

10.0

IBM MQ アプリケーション開発リファレンス



注記

本書および本書で紹介する製品をご使用になる前に、[2209 ページの『特記事項』](#)に記載されている情報をお読みください。

本版はバージョン 9 リリース 4 に適用され、新たな版で別段の記載があるまで、IBM® MQ その後のすべてのリリースおよび変更にも適用される。

情報を に送信する際、あなたは に IBM、その情報を任意の方法で使用または配布する非独占的権利を IBM 付与します。これにより、あなたに何らの義務も生じません。

© Copyright International Business Machines Corporation 2007 年, 2026.

目次

アプリケーションの開発に関する参照情報.....	7
MQI アプリケーション・リファレンス.....	7
定数.....	7
MQI で使用されるデータ・タイプ.....	177
関数呼び出し.....	617
オブジェクトの属性.....	791
戻りコード.....	869
MQI オプションの妥当性検査に関する規則.....	870
キュー型パブリッシュ/サブスクライブ・コマンド・メッセージ.....	873
マシン・エンコード.....	896
レポート・オプションおよびメッセージ・フラグ.....	898
データ変換出口.....	903
MQRFH2 エlementとして指定されるプロパティ.....	926
コード・ページ変換.....	935
64 ビット・プラットフォームでのコーディング標準.....	996
IBM i アプリケーション・プログラミングの参照情報 (ILE/RPG).....	1000
IBM i でのデータ・タイプの説明.....	1002
IBM i での関数呼び出し.....	1260
IBM i でのオブジェクトの属性.....	1380
IBM i の戻りコード (ILE RPG).....	1426
IBM i の MQI オプションの妥当性検査に関する規則 (ILE RPG).....	1428
IBM i でのマシン・エンコーディング.....	1430
IBM i でのレポート・オプションおよびメッセージ・フラグ.....	1433
IBM i 上でのデータ変換.....	1437
IBM i での変換処理.....	1437
IBM i での処理規則.....	1438
IBM i でのレポート・メッセージの変換.....	1442
IBM i での MQDXP (データ変換出口パラメーター).....	1443
IBM i での MQXCNCV (文字の変換).....	1448
IBM i での MQCONVX (データ変換出口).....	1453
ユーザー出口、API 出口、およびインストール可能サービス参照.....	1457
MQIEP 構造体.....	1457
データ変換出口リファレンス.....	1460
MQ_PUBLISH_EXIT - パブリッシュ出口.....	1464
チャネル出口呼び出しおよびデータ構造体.....	1473
クラスター・ワークロード出口呼び出しとデータ構造体.....	1539
API 出口参照.....	1565
インストール可能サービス・インターフェースの参照情報.....	1623
IBM i でのインストール可能サービス・インターフェースの参照情報.....	1687
IBM MQ .NET クラスとインターフェース.....	1726
MQAsyncStatus.NET クラス.....	1726
MQAuthenticationInformationRecord.NET クラス.....	1727
MQDestination.NET クラス.....	1728
MQEnvironment.NET クラス.....	1731
MQException.NET クラス.....	1734
MQGetMessageOptions.NET クラス.....	1734
MQManagedObject.NET クラス.....	1737
MQMessage.NET クラス.....	1740
MQProcess.NET クラス.....	1752
MQPropertyDescriptor.NET クラス.....	1754
MQPutMessageOptions.NET クラス.....	1756

MQQueue.NET クラス.....	1758
MQQueueManager.NET クラス.....	1766
MQSubscription.NET クラス.....	1779
MQTopic.NET クラス.....	1780
IMQObjectTrigger.NET インターフェース.....	1786
MQC.NET インターフェース.....	1787
.NET アプリケーション用の文字セット ID.....	1787
IBM MQ C++ クラス.....	1790
C++ と MQI の相互参照.....	1791
ImqAuthenticationRecord C++ クラス.....	1806
ImqBinary C++ クラス.....	1809
ImqCache C++ クラス.....	1810
ImqChannel C++ クラス.....	1813
ImqCICSBridgeHeader C++ クラス.....	1819
ImqDeadLetterHeader C++ クラス.....	1825
ImqDistributionList C++ クラス.....	1828
ImqError C++ クラス.....	1829
ImqGetMessageOptions C++ クラス.....	1830
ImqHeader C++ クラス.....	1833
ImqIMSBridgeHeader C++ クラス.....	1835
ImqItem C++ クラス.....	1838
ImqMessage C++ クラス.....	1839
ImqMessageTracker C++ クラス.....	1846
ImqNamelist C++ クラス.....	1848
ImqObject C++ クラス.....	1850
ImqProcess C++ クラス.....	1855
ImqPutMessageOptions C++ クラス.....	1857
ImqQueue C++ クラス.....	1859
ImqQueueManager C++ クラス.....	1869
ImqReferenceHeader C++ クラス.....	1885
ImqString C++ クラス.....	1888
ImqTrigger C++ クラス.....	1893
ImqWorkHeader C++ クラス.....	1896
IBM MQ classes for JMS オブジェクトのプロパティー.....	1898
IBM MQ classes for JMS オブジェクトのプロパティー間の依存関係.....	1902
APPLICATIONNAME.....	1904
ASYNCEXCEPTION.....	1905
バルオプション.....	1906
BALTYPE.....	1906
BALTIMEOUT.....	1907
BROKERCCDURSUBQ.....	1907
BROKERCCSUBQ.....	1908
BROKERCONQ.....	1908
BROKERDURSUBQ.....	1909
BROKERPUBQ.....	1909
BROKERPUBQMGR.....	1910
BROKERQMGR.....	1910
BROKERSUBQ.....	1910
BROKERVER.....	1911
CCDTURL.....	1912
CCSID.....	1912
CERTVALPO.....	1913
CHANNEL.....	1913
CLEANUP.....	1914
CLEANUPINT.....	1914
CONNECTIONNAMELIST.....	1915
CLIENTRECONNECTOPTIONS.....	1915

CLIENTRECONNECTTIMEOUT.....	1916
CLIENTID.....	1916
CLONESUPP.....	1917
COMPHDR.....	1917
COMPMSG.....	1918
CONNOPT.....	1918
CONNTAG.....	1919
説明.....	1920
DIRECTAUTH.....	1920
ENCODING.....	1921
EXPIRY.....	1922
FAILIFQUIESCE.....	1922
HOSTNAME.....	1923
LOCALADDRESS.....	1923
MAPNAMESTYLE.....	1924
MAXBUFFSIZE.....	1925
MDREAD.....	1925
MDWRITE.....	1926
MDMSGCTX.....	1926
MSGBATCHSZ.....	1927
MSGBODY.....	1927
MSGRETENTION.....	1928
MSGSELECTION.....	1928
MULTICAST.....	1929
OPTIMISTICPUBLICATION.....	1930
OUTCOMENOTIFICATION.....	1930
PERSISTENCE.....	1931
POLLINGINT.....	1931
PORT.....	1932
PRIORITY.....	1932
PROCESSDURATION.....	1933
PROVIDERVERSION.....	1933
PROXYHOSTNAME.....	1935
PROXYPORT.....	1936
PUBACKINT.....	1936
PUTASYNCALLOWED.....	1937
QMANAGER.....	1937
QUEUE.....	1938
READAHEADALLOWED.....	1938
READAHEADCLOSEPOLICY.....	1939
RECEIVECCSID.....	1939
RECEIVECONVERSION.....	1940
RECEIVEISOLATION.....	1940
RECEXIT.....	1941
RECEXITINIT.....	1941
REPLYTOSTYLE.....	1942
RESCANINT.....	1942
SECEXIT.....	1943
SECEXITINIT.....	1943
SENDCHECKCOUNT.....	1944
SENDEXIT.....	1944
SENDEXITINIT.....	1945
SHARECONVALLOWED.....	1945
SPARSESUBS.....	1946
SSLCIPHERSUITE.....	1946
SSLCRL.....	1947
SSLFIPSREQUIRED.....	1947
SSLPEERNAME.....	1948

SSLRESETCOUNT.....	1949
STATREFRESHINT.....	1949
SUBSTORE.....	1950
SYNCPOINTALLGETS.....	1950
TARGCLIENT.....	1951
TARGCLIENTMATCHING.....	1951
TEMPMODEL.....	1952
TEMPQPREFIX.....	1952
TEMPTOPICPREFIX.....	1953
トークンクライアント ID.....	1953
トークンクライアントシークレット.....	1954
トークンエンドポイント.....	1954
トークン HTTPS 証明書検証ポリシー.....	1955
トピック.....	1955
TRANSPORT.....	1956
WILDCARDFORMAT.....	1957
WMQ_CCDT_HTTPS_証明書検証ポリシー.....	1957
ENCODING プロパティ.....	1958
IBM MQ Message Service Client (XMS) for .NET リファレンス.....	1958
.NET インターフェース.....	1958
XMS オブジェクトのプロパティ.....	2038
Managed File Transfer アプリケーション開発リファレンス.....	2108
fteCreateTransfer を使用してプログラムを開始する例.....	2108
fteAnt : タスク Ant を実行する MFT.....	2110
MFT ユーザー出口カスタマイズ・リファレンス.....	2135
MFT エージェントのコマンド・キューに PUT できるメッセージ形式.....	2176
メッセージング REST API に関する参照情報.....	2176
REST API リソース.....	2176
特記事項.....	2209
プログラミング・インターフェース情報.....	2210
商標.....	2210

アプリケーションの開発に関する参照情報

このセクションにある各リンクを使用して、IBM MQ アプリケーションの開発に役立ててください。

- 7 ページの『MQI アプリケーション・リファレンス』
-  1000 ページの『IBM i アプリケーション・プログラミングの参照情報 (ILE/RPG)』
-  1437 ページの『IBM i 上でのデータ変換』
- 1457 ページの『ユーザー出口、API 出口、およびインストール可能サービス参照』
- 1726 ページの『IBM MQ .NET クラスとインターフェース』
- 1790 ページの『IBM MQ C++ クラス』
- 1898 ページの『IBM MQ classes for JMS オブジェクトのプロパティ』
- 2176 ページの『メッセージング REST API に関する参照情報』

関連タスク

[アプリケーションの開発](#)

関連資料

[IBM MQ Java ライブラリのためのクラス](#)

[IBM MQ classes for JMS](#)

MQI アプリケーション・リファレンス

このセクションにあるリンクは、Message Queue Interface (MQI) アプリケーションを開発するうえで役立ちます。

関連タスク

[IBM MQ での MQI アプリケーションの開発](#)

関連資料

996 ページの『64 ビット・プラットフォームでのコーディング標準』

この情報を使用して、64 ビット・プラットフォームでのコーディング標準および優先データ・タイプについて学びます。

[IBM MQ Java ライブラリのためのクラス](#)

[IBM MQ クラス JMS 2.0](#)

[IBM MQ クラス Jakarta Messaging](#)

定数

メッセージ・キュー・インターフェース (MQI) で使用可能な定数に関する情報。

MQ_* (ストリングの長さ)

表 1. 定数の値		
名前	10 進数値	16 進値
MQ_ABEND_CODE_LENGTH	4	X'00000004'
MQ_ACCOUNTING_TOKEN_LENGTH	32	X'00000020'
MQ_APPL_FUNCTION_NAME_LENGTH	10	X'0000000A'
MQ_APPL_IDENTITY_DATA_LENGTH	32	X'00000020'
MQ_APPL_NAME_LENGTH	28	X'0000001C'
MQ_APPL_ORIGIN_DATA_LENGTH	4	X'00000004'

表 1. 定数の値 (続き)		
名前	10 進数値	16 進値
MQ_APPL_TAG_LENGTH	28	X'0000001C'
MQ_ARM_SUFFIX_LENGTH	2	X'00000002'
MQ_ATTENTION_ID_LENGTH	4	X'00000004'
MQ_AUTH_INFO_CONN_NAME_LENGTH	264	X'00000108'
MQ_AUTH_INFO_DESC_LENGTH	64	X'00000040'
MQ_AUTH_INFO_NAME_LENGTH	48	X'00000030'
MQ_AUTH_INFO_OCSP_URL_LENGTH	256	X'00000100'
MQ_AUTHENTICATOR_LENGTH	8	X'00000008'
MQ_AUTO_REORG_CATALOG_LENGTH	44	X'0000002C'
MQ_AUTO_REORG_TIME_LENGTH	4	X'00000004'
MQ_BATCH_INTERFACE_ID_LENGTH	8	X'00000008'
MQ_BRIDGE_NAME_LENGTH	24	X'00000018'
MQ_CANCEL_CODE_LENGTH	4	X'00000004'
MQ_CF_STRUC_DESC_LENGTH	64	X'00000040'
MQ_CF_STRUC_NAME_LENGTH	12	X'0000000C'
MQ_CHANNEL_DATE_LENGTH	12	X'0000000C'
MQ_CHANNEL_DESC_LENGTH	64	X'00000040'
MQ_CHANNEL_NAME_LENGTH	20	X'00000014'
MQ_CHANNEL_TIME_LENGTH	8	X'00000008'
MQ_CHINIT_SERVICE_PARM_LENGTH	32	X'00000020'
MQ_CICS_FILE_NAME_LENGTH	8	X'00000008'
MQ_CLIENT_ID_LENGTH	23	X'00000017'
MQ_CLUSTER_NAME_LENGTH	48	X'00000030'
MQ_CONN_NAME_LENGTH	264	X'00000108'
MQ_CONN_TAG_LENGTH	128	X'00000080'
MQ_CONNECTION_ID_LENGTH	24	X'00000018'
MQ_CORREL_ID_LENGTH	24	X'00000018'
MQ_CREATION_DATE_LENGTH	12	X'0000000C'
MQ_CREATION_TIME_LENGTH	8	X'00000008'
MQ_DATE_LENGTH	12	X'0000000C'
MQ_DISTINGUISHED_NAME_LENGTH	1024	X'00000400'
MQ_DNS_GROUP_NAME_LENGTH	18	X'00000012'
MQ_EXIT_DATA_LENGTH	32	X'00000020'
MQ_EXIT_INFO_NAME_LENGTH	48	X'00000030'
MQ_EXIT_NAME_LENGTH	(value differs by platform or version)	
MQ_EXIT_PD_AREA_LENGTH	48	X'00000030'
MQ_EXIT_USER_AREA_LENGTH	16	X'00000010'
MQ_FACILITY_LENGTH	8	X'00000008'
MQ_FACILITY_LIKE_LENGTH	4	X'00000004'

表 1. 定数の値 (続き)		
名前	10 進数値	16 進値
MQ_FORMAT_LENGTH	8	X'00000008'
MQ_FUNCTION_LENGTH	4	X'00000004'
MQ_GROUP_ID_LENGTH	24	X'00000018'
MQ_LDAP_PASSWORD_LENGTH	32	X'00000020'
MQ_LISTENER_NAME_LENGTH	48	X'00000030'
MQ_LISTENER_DESC_LENGTH	64	X'00000040'
MQ_LOCAL_ADDRESS_LENGTH	48	X'00000030'
MQ_LTERM_OVERRIDE_LENGTH	8	X'00000008'
MQ_LU_NAME_LENGTH	8	X'00000008'
MQ_LUWID_LENGTH	16	X'00000010'
MQ_MAX_EXIT_NAME_LENGTH	128	X'00000080'
MQ_MAX_MCA_USER_ID_LENGTH	64	X'00000040'
MQ_MAX_PROPERTY_NAME_LENGTH	4095	X'00000FFF'
MQ_MAX_USER_ID_LENGTH	64	X'00000040'
MQ_MCA_JOB_NAME_LENGTH	28	X'0000001C'
MQ_MCA_NAME_LENGTH	20	X'00000014'
MQ_MCA_USER_DATA_LENGTH	32	X'00000020'
MQ_MCA_USER_ID_LENGTH	(value differs by platform or version)	(value differs by platform or version)
MQ_MFS_MAP_NAME_LENGTH	8	X'00000008'
MQ_MODE_NAME_LENGTH	8	X'00000008'
MQ_MSG_HEADER_LENGTH	4000	X'00000FA0'
MQ_MSG_ID_LENGTH	24	X'00000018'
MQ_MSG_TOKEN_LENGTH	16	X'00000010'
MQ_NAMELIST_DESC_LENGTH	64	X'00000040'
MQ_NAMELIST_NAME_LENGTH	48	X'00000030'
MQ_NHA_INSTANCE_NAME_LENGTH	48	X'00000030'
MQ_OBJECT_INSTANCE_ID_LENGTH	24	X'00000018'
MQ_OBJECT_NAME_LENGTH	48	X'00000030'
MQ_PASS_TICKET_APPL_LENGTH	8	X'00000008'
MQ_PASSWORD_LENGTH	12	X'0000000C'
MQ_PROCESS_APPL_ID_LENGTH	256	X'00000100'
MQ_PROCESS_DESC_LENGTH	64	X'00000040'
MQ_PROCESS_ENV_DATA_LENGTH	128	X'00000080'
MQ_PROCESS_NAME_LENGTH	48	X'00000030'
MQ_PROCESS_USER_DATA_LENGTH	128	X'00000080'
MQ_PROGRAM_NAME_LENGTH	20	X'00000014'
MQ_PUT_APPL_NAME_LENGTH	28	X'0000001C'
MQ_PUT_DATE_LENGTH	8	X'00000008'
MQ_PUT_TIME_LENGTH	8	X'00000008'

表 1. 定数の値 (続き)		
名前	10 進数値	16 進値
MQ_Q_DESC_LENGTH	64	X'00000040'
MQ_Q_MGR_DESC_LENGTH	64	X'00000040'
MQ_Q_MGR_IDENTIFIER_LENGTH	48	X'00000030'
MQ_Q_MGR_NAME_LENGTH	48	X'00000030'
MQ_Q_NAME_LENGTH	48	X'00000030'
MQ_QSG_NAME_LENGTH	4	X'00000004'
MQ_REMOTE_SYS_ID_LENGTH	4	X'00000004'
MQ_SECURITY_ID_LENGTH	40	X'00000028'
MQ_SELECTOR_LENGTH	10240	X'00002800'
MQ_SERVICE_ARGS_LENGTH	255	X'000000FF'
MQ_SERVICE_COMMAND_LENGTH	255	X'000000FF'
MQ_SERVICE_DESC_LENGTH	64	X'00000040'
MQ_SERVICE_NAME_LENGTH	32	X'00000020'
MQ_SERVICE_PATH_LENGTH	255	X'000000FF'
MQ_SERVICE_STEP_LENGTH	8	X'00000008'
MQ_SHORT_CONN_NAME_LENGTH	20	X'00000014'
MQ_SHORT_DNAME_LENGTH	256	X'00000100'
MQ_SSL_CIPHER_SPEC_LENGTH	32	X'00000020'
MQ_SSL_CRYPTO_HARDWARE_LENGTH	256	X'00000100'
MQ_SSL_HANDSHAKE_STAGE_LENGTH	32	X'00000020'
MQ_SSL_KEY_LIBRARY_LENGTH	44	X'0000002C'
MQ_SSL_KEY_MEMBER_LENGTH	8	X'00000008'
MQ_SSL_KEY_REPOSITORY_LENGTH	256	X'00000100'
MQ_SSL_PEER_NAME_LENGTH	1024	X'00000400'
MQ_SSL_SHORT_PEER_NAME_LENGTH	256	X'00000100'
MQ_START_CODE_LENGTH	4	X'00000004'
MQ_STORAGE_CLASS_DESC_LENGTH	64	X'00000040'
MQ_STORAGE_CLASS_LENGTH	8	X'00000008'
MQ_SUB_IDENTITY_LENGTH	128	X'00000080'
MQ_SUB_POINT_LENGTH	128	X'00000080'
MQ_SUITE_B_128_BIT	2	X'00000002'
MQ_SUITE_B_192_BIT	4	X'00000004'
MQ_SUITE_B_NONE	1	X'00000001'
MQ_SUITE_B_NOT_AVAILABLE	0	X'00000000'
MQ_TCP_NAME_LENGTH	8	X'00000008'
MQ_TIME_LENGTH	8	X'00000008'
MQ_TOPIC_DESC_LENGTH	64	X'00000040'
MQ_TOPIC_NAME_LENGTH	48	X'00000030'
MQ_TOPIC_STR_LENGTH	10240	X'00002800'

表 1. 定数の値 (続き)		
名前	10 進数値	16 進値
MQ_TOTAL_EXIT_DATA_LENGTH	999	X'000003E7'
MQ_TOTAL_EXIT_NAME_LENGTH	999	X'000003E7'
MQ_TP_NAME_LENGTH	64	X'00000040'
MQ_TPIPE_NAME_LENGTH	8	X'00000008'
MQ_TRAN_INSTANCE_ID_LENGTH	16	X'00000010'
MQ_TRANSACTION_ID_LENGTH	4	X'00000004'
MQ_TRIGGER_DATA_LENGTH	64	X'00000040'
MQ_TRIGGER_PROGRAM_NAME_LENGTH	8	X'00000008'
MQ_TRIGGER_TERM_ID_LENGTH	4	X'00000004'
MQ_TRIGGER_TRANS_ID_LENGTH	4	X'00000004'
MQ_USER_ID_LENGTH	12	X'0000000C'
MQ_VERSION_LENGTH	8	X'00000008'
MQ_XCF_GROUP_NAME_LENGTH	8	X'00000008'
MQ_XCF_MEMBER_NAME_LENGTH	16	X'00000010'

MQ_* (コマンド形式のストリングの長さ)

表 2. 定数の値		
名前	10 進数値	16 進値
MQ_ARCHIVE_PFX_LENGTH	36	X'00000024'
MQ_ARCHIVE_UNIT_LENGTH	8	X'00000008'
MQ_ASID_LENGTH	4	X'00000004'
MQ_AUTH_PROFILE_NAME_LENGTH	48	X'00000030'
MQ_CF_LEID_LENGTH	12	X'0000000C'
MQ_COMMAND_MQSC_LENGTH	32768	X'00008000'
MQ_DATA_SET_NAME_LENGTH	44	X'0000002C'
MQ_DB2_NAME_LENGTH	4	X'00000004'
MQ_DSG_NAME_LENGTH	8	X'00000008'
MQ_ENTITY_NAME_LENGTH	1024	X'00000400'
MQ_ENV_INFO_LENGTH	96	X'00000060'
MQ_IP_ADDRESS_LENGTH	48	X'00000030'
MQ_LOG_CORREL_ID_LENGTH	8	X'00000008'
MQ_LOG_EXTENT_NAME_LENGTH	24	X'00000018'
MQ_LOG_PATH_LENGTH	1024	X'00000400'
MQ_LRSN_LENGTH	12	X'0000000C'
MQ_ORIGIN_NAME_LENGTH	8	X'00000008'
MQ_PSB_NAME_LENGTH	8	X'00000008'
MQ_PST_ID_LENGTH	8	X'00000008'
MQ_Q_MGR_CPF_LENGTH	4	X'00000004'
MQ_RESPONSE_ID_LENGTH	24	X'00000018'
MQ_RBA_LENGTH	16	X'00000010'

表 2. 定数の値 (続き)		
名前	10 進数値	16 進値
MQ_SECURITY_PROFILE_LENGTH	40	X'00000028'
MQ_SERVICE_COMPONENT_LENGTH	48	X'00000030'
MQ_SUB_NAME_LENGTH	10240	X'00002800'
MQ_SYSP_SERVICE_LENGTH	32	X'00000020'
MQ_SYSTEM_NAME_LENGTH	8	X'00000008'
MQ_TASK_NUMBER_LENGTH	8	X'00000008'
MQ_TPIPE_PFX_LENGTH	4	X'00000004'
MQ_UOW_ID_LENGTH	256	X'00000100'
MQ_USER_DATA_LENGTH	10240	X'00002800'
MQ_VOLSER_LENGTH	6	X'00000006'

MQACH_* (API 出口チェーン領域ヘッダー構造体)

表 3. 定数の構造	
名前	構造体
MQACH_STRUC_ID	"ACH-"
MQACH_STRUC_ID_ARRAY	'A','C','H',' '

注: 記号-は、単一のブランク文字を表します。

表 4. 定数の値		
名前	10 進数値	16 進値
MQACH_VERSION_1	1	X'00000001'
MQACH_CURRENT_VERSION	1	X'00000001'
MQACH_LENGTH_1	(value differs by platform or version)	(value differs by platform or version)
MQACH_CURRENT_LENGTH	(value differs by platform or version)	(value differs by platform or version)

MQACT_* (会計トークン)

表 5. 定数名と値	
名前	値
MQACT_NONE	X'00...00' (32 個のヌル値)
MQACT_NONE_ARRAY	'\0','\0',... (32 個のヌル値)

MQACT_* (コマンド形式のアクション・オプション)

表 6. 定数の値		
名前	10 進数値	16 進値
MQACT_FORCE_REMOVE	1	X'00000001'
MQACT_ADVANCE_LOG	2	X'00000002'
MQACT_COLLECT_STATISTICS	3	X'00000003'
MQACT_PUBSUB	4	X'00000004'

MQACTP_* (アクション)

表 7. 定数の値		
名前	10 進数値	16 進値
MQACTP_NEW	0	X'00000000'
MQACTP_FORWARD	1	X'00000001'
MQACTP_REPLY	2	X'00000002'
MQACTP_REPORT	3	X'00000003'

MQACTT_* (会計トークン・タイプ)

表 8. 定数の値	
名前	16 進値
MQACTT_UNKNOWN	X'00'
MQACTT_CICS_LUOW_ID	X'01'
MQACTT_OS2_DEFAULT	X'04'
MQACTT_DOS_DEFAULT	X'05'
MQACTT_UNIX_NUMERIC_ID	X'06'
MQACTT_OS400_ACCOUNT_TOKEN	X'08'
MQACTT_WINDOWS_DEFAULT	X'09'
MQACTT_NT_SECURITY_ID	X'0B'
MQACTT_USER	X'19'

MQADOPT_* (新規 MCA チェックおよび新規 MCA タイプの採用)

新規 MCA チェックの採用

表 9. 定数の値		
名前	10 進数値	16 進値
MQADOPT_CHECK_NONE	0	X'00000000'
MQADOPT_CHECK_ALL	1	X'00000001'
MQADOPT_CHECK_Q_MGR_NAME	2	X'00000002'
MQADOPT_CHECK_NET_ADDR	4	X'00000004'

新規 MCA タイプの採用

表 10. 定数の値		
名前	10 進数値	16 進値
MQADOPT_TYPE_NO	0	X'00000000'
MQADOPT_TYPE_ALL	1	X'00000001'
MQADOPT_TYPE_SVR	2	X'00000002'
MQADOPT_TYPE_SDR	4	X'00000004'
MQADOPT_TYPE_RCVR	8	X'00000008'
MQADOPT_TYPE_CLUSRCVR	16	X'00000010'

MQAIR_* (認証情報レコード構造体)

表 11. 定数の構造	
名前	構造体
MQAIR_STRUC_ID	"AIR-"
MQAIR_STRUC_ID_ARRAY	'A','I','R',' '

注: 記号-は、単一のブランク文字を表します。

表 12. 定数の値		
名前	10 進数値	16 進値
MQAIR_VERSION_1	1	X'00000001'
MQAIR_VERSION_2	2	X'00000002'
MQAIR_CURRENT_VERSION	2	X'00000002'

MQAIT_* (認証情報のタイプ)

表 13. 定数の値		
名前	10 進数値	16 進値
MQAIT_ALL	0	X'00000000'
MQAIT_CRL_LDAP	1	X'00000001'
MQAIT_OCSP	2	X'00000002'
MQAIT_IDPW_OS	3	X'00000003'
MQAIT_IDPW_LDAP	4	X'00000004'

MQAS_* (コマンド形式の非同期状態値)

表 14. 定数の値		
名前	10 進数値	16 進値
MQAS_NONE	0	X'00000000'
MQAS_STARTED	1	X'00000001'
MQAS_START_WAIT	2	X'00000002'
MQAS_STOPPED	3	X'00000003'
MQAS_SUSPENDED	4	X'00000004'
MQAS_SUSPENDED_TEMPORARY	5	X'00000005'
MQAS_ACTIVE	6	X'00000006'
MQAS_INACTIVE	7	X'00000007'

MQAT_* (アプリケーション書き込みタイプ)

表 15. 定数の値		
名前	10 進数値	16 進値
MQAT_UNKNOWN	-1	X'FFFFFFFF'
MQAT_NO_CONTEXT	0	X'00000000'
MQAT_CICS	1	X'00000001'
MQAT_MVS	2	X'00000002'

表 15. 定数の値 (続き)		
名前	10 進数値	16 進値
MQAT_OS390	2	X'00000002'
MQAT_ZOS	2	X'00000002'
MQAT_IMS	3	X'00000003'
MQAT_OS2	4	X'00000004'
MQAT_DOS	5	X'00000005'
MQAT_AIX	6	X'00000006'
MQAT_UNIX	6	X'00000006'
MQAT_QMGR	7	X'00000007'
MQAT_OS400	8	X'00000008'
MQAT_WINDOWS	9	X'00000009'
MQAT_CICS_VSE	10	X'0000000A'
MQAT_WINDOWS_NT	11	X'0000000B'
MQAT_VMS	12	X'0000000C'
MQAT_GUARDIAN	13	X'0000000D'
MQAT_NSK	13	X'0000000D'
MQAT_VOS	14	X'0000000E'
MQAT_OPEN_TP1	15	X'0000000F'
MQAT_VM	18	X'00000012'
MQAT_IMS_BRIDGE	19	X'00000013'
MQAT_XCF	20	X'00000014'
MQAT_CICS_BRIDGE	21	X'00000015'
MQAT_NOTES_AGENT	22	X'00000016'
MQAT_TPF	23	X'00000017'
MQAT_USER	25	X'00000019'
MQAT_BROKER	26	X'0000001A'
MQAT_QMGR_PUBLISH	26	X'0000001A'
MQAT_JAVA	28	X'0000001C'
MQAT_DQM	29	X'0000001D'
MQAT_CHANNEL_INITIATOR	30	X'0000001E'
MQAT_WLM	31	X'0000001F'
MQAT_BATCH	32	X'00000020'
MQAT_RRS_BATCH	33	X'00000021'
MQAT_SIB	34	X'00000022'
MQAT_DEFAULT	(value differs by platform or version)	(value differs by platform or version)
MQAT_USER_FIRST	65536	X'00010000'
MQAT_USER_LAST	999999999	X'3B9AC9FF'

 MQAUSC_* (コマンド形式権限範囲値)

表 16. 定数の値		
名前	10 進数値	16 進値
MQAUSC_失敗	0	X'00000000'
MQAUSC_全接続	1	X'00000001'
MQAUSC_全チェック	2	X'00000002'

MQAUTH_* (コマンド形式の権限値)

表 17. 定数の値		
名前	10 進数値	16 進値
MQAUTH_NONE	0	X'00000000'
MQAUTH_ALT_USER_AUTHORITY	1	X'00000001'
MQAUTH_BROWSE	2	X'00000002'
MQAUTH_CHANGE	3	X'00000003'
MQAUTH_CLEAR	4	X'00000004'
MQAUTH_CONNECT	5	X'00000005'
MQAUTH_CREATE	6	X'00000006'
MQAUTH_DELETE	7	X'00000007'
MQAUTH_DISPLAY	8	X'00000008'
MQAUTH_INPUT	9	X'00000009'
MQAUTH_INQUIRE	10	X'0000000A'
MQAUTH_OUTPUT	11	X'0000000B'
MQAUTH_PASS_ALL_CONTEXT	12	X'0000000C'
MQAUTH_PASS_IDENTITY_CONTEXT	13	X'0000000D'
MQAUTH_SET	14	X'0000000E'
MQAUTH_SET_ALL_CONTEXT	15	X'0000000F'
MQAUTH_SET_IDENTITY_CONTEXT	16	X'00000010'
MQAUTH_CONTROL	17	X'00000011'
MQAUTH_CONTROL_EXTENDED	18	X'00000012'
MQAUTH_PUBLISH	19	X'00000013'
MQAUTH_SUBSCRIBE	20	X'00000014'
MQAUTH_RESUME	21	X'00000015'
MQAUTH_SYSTEM	22	X'00000016'

MQAUTHOPT_* (コマンド形式の権限オプション)

表 18. 定数の値		
名前	10 進数値	16 進値
MQAUTHOPT_CUMULATIVE	256	X'00000100'
MQAUTHOPT_ENTITY_EXPLICIT	1	X'00000001'
MQAUTHOPT_ENTITY_SET	2	X'00000002'
MQAUTHOPT_NAME_ALL_MATCHING	32	X'00000020'

表 18. 定数の値 (続き)		
名前	10 進数値	16 進値
MQAUTHOPT_NAME_AS_WILDCARD	64	X'00000040'
MQAUTHOPT_NAME_EXPLICIT	16	X'00000010'

MQAXC_* (API 出口コンテキスト構造体)

表 19. 定数の構造	
名前	構造体
MQAXC_STRUC_ID	"AXC-"
MQAXC_STRUC_ID_ARRAY	'A', 'X', 'C', '-'

注: 記号-は、単一のブランク文字を表します。

表 20. 定数の値		
名前	10 進数値	16 進値
MQAXC_VERSION_1	1	X'00000001'
MQAXC_CURRENT_VERSION	1	X'00000001'

MQAXP_* (API 出口パラメーター構造体)

表 21. 定数の構造	
名前	構造体
MQAXP_STRUC_ID	"AXP-"
MQAXP_STRUC_ID_ARRAY	'A', 'X', 'P', '-'

注: 記号-は、単一のブランク文字を表します。

表 22. 定数の値		
名前	10 進数値	16 進値
MQAXP_VERSION_1	1	X'00000001'
MQAXP_VERSION_2	2	X'00000002'
MQAXP_CURRENT_VERSION	2	X'00000002'

MQBA_* (バイト属性セクター)

表 23. 定数の値		
名前	10 進数値	16 進値
MQBA_FIRST	6001	X'00001771'
MQBA_LAST	8000	X'00001F40'

MQBACF_* (コマンド形式のバイト・パラメーター・タイプ)

表 24. 定数の値		
名前	10 進数値	16 進値
MQBACF_FIRST	7001	X'00001B59'
MQBACF_EVENT_ACCOUNTING_TOKEN	7001	X'00001B59'
MQBACF_EVENT_SECURITY_ID	7002	X'00001B5A'

表 24. 定数の値 (続き)		
名前	10 進数値	16 進値
MQBACF_RESPONSE_SET	7003	X'00001B5B'
MQBACF_RESPONSE_ID	7004	X'00001B5C'
MQBACF_EXTERNAL_UOW_ID	7005	X'00001B5D'
MQBACF_CONNECTION_ID	7006	X'00001B5E'
MQBACF_GENERIC_CONNECTION_ID	7007	X'00001B5F'
MQBACF_ORIGIN_UOW_ID	7008	X'00001B60'
MQBACF_Q_MGR_UOW_ID	7009	X'00001B61'
MQBACF_ACCOUNTING_TOKEN	7010	X'00001B62'
MQBACF_CORREL_ID	7011	X'00001B63'
MQBACF_GROUP_ID	7012	X'00001B64'
MQBACF_MSG_ID	7013	X'00001B65'
MQBACF_CF_LEID	7014	X'00001B66'
MQBACF_DESTINATION_CORREL_ID	7015	X'00001B67'
MQBACF_SUB_ID	7016	X'00001B68'
MQBACF_LAST_USED	7016	X'00001B68'

MQBL_* (mqAddString および mqSetString のバッファーク長)

表 25. 定数の値		
名前	10 進数値	16 進値
MQBL_NULL_TERMINATED	-1	X'FFFFFFFF'

MQBMHO_* (バッファークからメッセージ・ハンドルへの変換オプションおよび構造体)

バッファークからメッセージ・ハンドルへの変換オプション構造

表 26. 定数の構造	
名前	構造体
MQBMHO_STRUC_ID	"BMHO"
MQBMHO_STRUC_ID_ARRAY	'B', 'M', 'H', 'O'

注: 記号-は、単一のブランク文字を表します。

表 27. 定数の値		
名前	10 進数値	16 進値
MQBMHO_VERSION_1	1	X'00000001'
MQBMHO_CURRENT_VERSION	1	X'00000001'

バッファークからメッセージ・ハンドルへの変換オプション

表 28. 定数の値		
名前	10 進数値	16 進値
MQBMHO_NONE	0	X'00000000'

表 28. 定数の値 (続き)		
名前	10 進数値	16 進値
MQBMHO_DELETE_PROPERTIES	1	X'00000001'

MQBND_* (デフォルト・バインディング)

表 29. 定数の値		
名前	10 進数値	16 進値
MQBND_BIND_ON_OPEN	0	X'00000000'
MQBND_BIND_NOT_FIXED	1	X'00000001'
MQBND_BIND_ON_GROUP	2	X'00000002'

MQBO_* (開始オプションおよび構造体)

開始オプション構造体

表 30. 定数の構造	
名前	構造体
MQBO_STRUC_ID	"B0--"
MQBO_STRUC_ID_ARRAY	'B','0','-', '-'

注：記号-は、単一のブランク文字を表します。

表 31. 定数の値		
名前	10 進数値	16 進値
MQBO_VERSION_1	1	X'00000001'
MQBO_CURRENT_VERSION	1	X'00000001'

開始オプション

表 32. 定数の値		
名前	10 進数値	16 進値
MQBO_NONE	0	X'00000000'

MQBT_* (コマンド形式のブリッジ・タイプ)

表 33. 定数の値		
名前	10 進数値	16 進値
MQBT_OTMA	1	X'00000001'

MQCA_* (文字属性セクター)

表 34. 定数の値		
名前	10 進数値	16 進値
MQCA_ADMIN_TOPIC_NAME	2105	X'00000839'
MQCA_ALTERATION_DATE	2027	X'000007EB'
MQCA_ALTERATION_TIME	2028	X'000007EC'
MQCA_APPL_ID	2001	X'000007D1'

表 34. 定数の値 (続き)		
名前	10 進数値	16 進値
MQCA_AUTH_INFO_CONN_NAME	2053	X'00000805'
MQCA_AUTH_INFO_DESC	2046	X'000007FE'
MQCA_AUTH_INFO_NAME	2045	X'000007FD'
MQCA_AUTH_INFO_OCSP_URL	2109	X'0000083D'
MQCA_AUTO_REORG_CATALOG	2091	X'0000082B'
MQCA_AUTO_REORG_START_TIME	2090	X'0000082A'
MQCA_BACKOUT_REQ_Q_NAME	2019	X'000007E3'
MQCA_BASE_OBJECT_NAME	2002	X'000007D2'
MQCA_BASE_Q_NAME	2002	X'000007D2'
MQCA_BATCH_INTERFACE_ID	2068	X'00000814'
MQCA_CF_STRUC_DESC	2052	X'00000804'
MQCA_CF_STRUC_NAME	2039	X'000007F7'
MQCA_CHANNEL_AUTO_DEF_EXIT	2026	X'000007EA'
MQCA_CHILD	2101	X'00000835'
MQCA_CHINIT_SERVICE_PARM	2076	X'0000081C'
MQCA_CICS_FILE_NAME	2060	X'0000080C'
MQCA_CLUS_CHL_NAME	2124	X'0000084C'
MQCA_CLUSTER_DATE	2037	X'000007F5'
MQCA_CLUSTER_NAME	2029	X'000007ED'
MQCA_CLUSTER_NAMELIST	2030	X'000007EE'
MQCA_CLUSTER_Q_MGR_NAME	2031	X'000007EF'
MQCA_CLUSTER_TIME	2038	X'000007F6'
MQCA_CLUSTER_WORKLOAD_DATA	2034	X'000007F2'
MQCA_CLUSTER_WORKLOAD_EXIT	2033	X'000007F1'
MQCA_COMMAND_INPUT_Q_NAME	2003	X'000007D3'
MQCA_COMMAND_REPLY_Q_NAME	2067	X'00000813'
MQCA_CREATION_DATE	2004	X'000007D4'
MQCA_CREATION_TIME	2005	X'000007D5'
MQCA_DEAD_LETTER_Q_NAME	2006	X'000007D6'
MQCA_DEF_XMIT_Q_NAME	2025	X'000007E9'
MQCA_DNS_GROUP	2071	X'00000817'
MQCA_ENV_DATA	2007	X'000007D7'
MQCA_FIRST	2001	X'000007D1'
MQCA_IGQ_USER_ID	2041	X'000007F9'
MQCA_INITIATION_Q_NAME	2008	X'000007D8'
MQCA_LAST	4000	X'00000FA0'
MQCA_LAST_USED	2109	X'0000083D'
MQCA_LDAP_PASSWORD	2048	X'00000800'
MQCA_LDAP_USER_NAME	2047	X'000007FF'

表 34. 定数の値 (続き)		
名前	10 進数値	16 進値
MQCA_LU_GROUP_NAME	2072	X'00000818'
MQCA_LU_NAME	2073	X'00000819'
MQCA_LU62_ARM_SUFFIX	2074	X'0000081A'
MQCA_MODEL_DURABLE_Q	2096	X'00000830'
MQCA_MODEL_NON_DURABLE_Q	2097	X'00000831'
MQCA_MONITOR_Q_NAME	2066	X'00000812'
MQCA_NAMELIST_DESC	2009	X'000007D9'
MQCA_NAMELIST_NAME	2010	X'000007DA'
MQCA_NAMES	2020	X'000007E4'
MQCA_PARENT	2102	X'00000836'
MQCA_PASS_TICKET_APPL	2086	X'00000826'
MQCA_PROCESS_DESC	2011	X'000007DB'
MQCA_PROCESS_NAME	2012	X'000007DC'
MQCA_Q_DESC	2013	X'000007DD'
MQCA_Q_MGR_DESC	2014	X'000007DE'
MQCA_Q_MGR_IDENTIFIER	2032	X'000007F0'
MQCA_Q_MGR_NAME	2015	X'000007DF'
MQCA_Q_NAME	2016	X'000007E0'
MQCA_QSG_NAME	2040	X'000007F8'
MQCA_REMOTE_Q_MGR_NAME	2017	X'000007E1'
MQCA_REMOTE_Q_NAME	2018	X'000007E2'
MQCA_REPOSITORY_NAME	2035	X'000007F3'
MQCA_REPOSITORY_NAMELIST	2036	X'000007F4'
MQCA_RESUME_DATE	2098	X'00000832'
MQCA_RESUME_TIME	2099	X'00000833'
MQCA_SERVICE_DESC	2078	X'0000081E'
MQCA_SERVICE_NAME	2077	X'0000081D'
MQCA_SERVICE_START_ARGS	2080	X'00000820'
MQCA_SERVICE_START_COMMAND	2079	X'0000081F'
MQCA_SERVICE_STOP_ARGS	2082	X'00000822'
MQCA_SERVICE_STOP_COMMAND	2081	X'00000821'
MQCA_STDERR_DESTINATION	2084	X'00000824'
MQCA_STDOUT_DESTINATION	2083	X'00000823'
MQCA_SSL_CRL_NAMELIST	2050	X'00000802'
MQCA_SSL_CRYPTO_HARDWARE	2051	X'00000803'
MQCA_SSL_KEY_LIBRARY	2069	X'00000815'
MQCA_SSL_KEY_MEMBER	2070	X'00000816'
MQCA_SSL_KEY_REPOSITORY	2049	X'00000801'
MQCA_STORAGE_CLASS	2022	X'000007E6'

表 34. 定数の値 (続き)		
名前	10 進数値	16 進値
MQCA_STORAGE_CLASS_DESC	2042	X'000007FA'
MQCA_SYSTEM_LOG_Q_NAME	2065	X'00000811'
MQCA_TCP_NAME	2075	X'0000081B'
MQCA_TOPIC_DESC	2093	X'0000082D'
MQCA_TOPIC_NAME	2092	X'0000082C'
MQCA_TOPIC_STRING_FILTER	2108	X'0000083C'
MQCA_TOPIC_STRING	2094	X'0000082E'
MQCA_TPIPE_NAME	2085	X'00000825'
MQCA_TRIGGER_CHANNEL_NAME	2064	X'00000810'
MQCA_TRIGGER_DATA	2023	X'000007E7'
MQCA_TRIGGER_PROGRAM_NAME	2062	X'0000080E'
MQCA_TRIGGER_TERM_ID	2063	X'0000080F'
MQCA_TRIGGER_TRANS_ID	2061	X'0000080D'
MQCA_USER_DATA	2021	X'000007E5'
MQCA_USER_LIST	4000	X'00000FA0'
MQCA_VERSION	2120	X'00000848'
MQCA_XCF_GROUP_NAME	2043	X'000007FB'
MQCA_XCF_MEMBER_NAME	2044	X'000007FC'
MQCA_XMIT_Q_NAME	2024	X'000007E8'

MQCACF_* (コマンド形式の文字パラメーター・タイプ)

表 35. 定数の値		
名前	10 進数値	16 進値
MQCACF_FIRST	3001	X'00000BB9'
MQCACF_FROM_Q_NAME	3001	X'00000BB9'
MQCACF_TO_Q_NAME	3002	X'00000BBA'
MQCACF_FROM_PROCESS_NAME	3003	X'00000BBB'
MQCACF_TO_PROCESS_NAME	3004	X'00000BBC'
MQCACF_FROM_NAMELIST_NAME	3005	X'00000BBD'
MQCACF_TO_NAMELIST_NAME	3006	X'00000BBE'
MQCACF_FROM_CHANNEL_NAME	3007	X'00000BBF'
MQCACF_TO_CHANNEL_NAME	3008	X'00000BC0'
MQCACF_FROM_AUTH_INFO_NAME	3009	X'00000BC1'
MQCACF_TO_AUTH_INFO_NAME	3010	X'00000BC2'
MQCACF_Q_NAMES	3011	X'00000BC3'
MQCACF_PROCESS_NAMES	3012	X'00000BC4'
MQCACF_NAMELIST_NAMES	3013	X'00000BC5'
MQCACF_ESCAPE_TEXT	3014	X'00000BC6'
MQCACF_LOCAL_Q_NAMES	3015	X'00000BC7'

表 35. 定数の値 (続き)		
名前	10 進数値	16 進値
MQCACF_MODEL_Q_NAMES	3016	X'00000BC8'
MQCACF_ALIAS_Q_NAMES	3017	X'00000BC9'
MQCACF_REMOTE_Q_NAMES	3018	X'00000BCA'
MQCACF_SENDER_CHANNEL_NAMES	3019	X'00000BCB'
MQCACF_SERVER_CHANNEL_NAMES	3020	X'00000BCC'
MQCACF_REQUESTER_CHANNEL_NAMES	3021	X'00000BCD'
MQCACF_RECEIVER_CHANNEL_NAMES	3022	X'00000BCE'
MQCACF_OBJECT_Q_MGR_NAME	3023	X'00000BCF'
MQCACF_APPL_NAME	3024	X'00000BD0'
MQCACF_USER_IDENTIFIER	3025	X'00000BD1'
MQCACF_AUX_ERROR_DATA_STR_1	3026	X'00000BD2'
MQCACF_AUX_ERROR_DATA_STR_2	3027	X'00000BD3'
MQCACF_AUX_ERROR_DATA_STR_3	3028	X'00000BD4'
MQCACF_BRIDGE_NAME	3029	X'00000BD5'
MQCACF_STREAM_NAME	3030	X'00000BD6'
MQCACF_TOPIC	3031	X'00000BD7'
MQCACF_PARENT_Q_MGR_NAME	3032	X'00000BD8'
MQCACF_CORREL_ID	3033	X'00000BD9'
MQCACF_PUBLISH_TIMESTAMP	3034	X'00000BDA'
MQCACF_STRING_DATA	3035	X'00000BDB'
MQCACF_SUPPORTED_STREAM_NAME	3036	X'00000BDC'
MQCACF_REG_TOPIC	3037	X'00000BDD'
MQCACF_REG_TIME	3038	X'00000BDE'
MQCACF_REG_USER_ID	3039	X'00000BDF'
MQCACF_CHILD_Q_MGR_NAME	3040	X'00000BE0'
MQCACF_REG_STREAM_NAME	3041	X'00000BE1'
MQCACF_REG_Q_MGR_NAME	3042	X'00000BE2'
MQCACF_REG_Q_NAME	3043	X'00000BE3'
MQCACF_REG_CORREL_ID	3044	X'00000BE4'
MQCACF_EVENT_USER_ID	3045	X'00000BE5'
MQCACF_OBJECT_NAME	3046	X'00000BE6'
MQCACF_EVENT_Q_MGR	3047	X'00000BE7'
MQCACF_AUTH_INFO_NAMES	3048	X'00000BE8'
MQCACF_EVENT_APPL_IDENTITY	3049	X'00000BE9'
MQCACF_EVENT_APPL_NAME	3050	X'00000BEA'
MQCACF_EVENT_APPL_ORIGIN	3051	X'00000BEB'
MQCACF_SUBSCRIPTION_NAME	3052	X'00000BEC'
MQCACF_REG_SUB_NAME	3053	X'00000BED'
MQCACF_SUBSCRIPTION_IDENTITY	3054	X'00000BEE'

表 35. 定数の値 (続き)		
名前	10 進数値	16 進値
MQCACF_REG_SUB_IDENTITY	3055	X'00000BEF'
MQCACF_SUBSCRIPTION_USER_DATA	3056	X'00000BF0'
MQCACF_REG_SUB_USER_DATA	3057	X'00000BF1'
MQCACF_APPL_TAG	3058	X'00000BF2'
MQCACF_DATA_SET_NAME	3059	X'00000BF3'
MQCACF_UOW_START_DATE	3060	X'00000BF4'
MQCACF_UOW_START_TIME	3061	X'00000BF5'
MQCACF_UOW_LOG_START_DATE	3062	X'00000BF6'
MQCACF_UOW_LOG_START_TIME	3063	X'00000BF7'
MQCACF_UOW_LOG_EXTENT_NAME	3064	X'00000BF8'
MQCACF_PRINCIPAL_ENTITY_NAMES	3065	X'00000BF9'
MQCACF_GROUP_ENTITY_NAMES	3066	X'00000BFA'
MQCACF_AUTH_PROFILE_NAME	3067	X'00000BFB'
MQCACF_ENTITY_NAME	3068	X'00000BFC'
MQCACF_SERVICE_COMPONENT	3069	X'00000BFD'
MQCACF_RESPONSE_Q_MGR_NAME	3070	X'00000BFE'
MQCACF_CURRENT_LOG_EXTENT_NAME	3071	X'00000BFF'
MQCACF_RESTART_LOG_EXTENT_NAME	3072	X'00000C00'
MQCACF_MEDIA_LOG_EXTENT_NAME	3073	X'00000C01'
MQCACF_LOG_PATH	3074	X'00000C02'
MQCACF_COMMAND_MQSC	3075	X'00000C03'
MQCACF_Q_MGR_CPF	3076	X'00000C04'
MQCACF_USAGE_LOG_RBA	3078	X'00000C06'
MQCACF_USAGE_LOG_LRSN	3079	X'00000C07'
MQCACF_COMMAND_SCOPE	3080	X'00000C08'
MQCACF_ASID	3081	X'00000C09'
MQCACF_PSB_NAME	3082	X'00000C0A'
MQCACF_PST_ID	3083	X'00000C0B'
MQCACF_TASK_NUMBER	3084	X'00000C0C'
MQCACF_TRANSACTION_ID	3085	X'00000C0D'
MQCACF_Q_MGR_UOW_ID	3086	X'00000C0E'
MQCACF_ORIGIN_NAME	3088	X'00000C10'
MQCACF_ENV_INFO	3089	X'00000C11'
MQCACF_SECURITY_PROFILE	3090	X'00000C12'
MQCACF_CONFIGURATION_DATE	3091	X'00000C13'
MQCACF_CONFIGURATION_TIME	3092	X'00000C14'
MQCACF_FROM_CF_STRUC_NAME	3093	X'00000C15'
MQCACF_TO_CF_STRUC_NAME	3094	X'00000C16'
MQCACF_CF_STRUC_NAMES	3095	X'00000C17'

表 35. 定数の値 (続き)		
名前	10 進数値	16 進値
MQCACF_FAIL_DATE	3096	X'00000C18'
MQCACF_FAIL_TIME	3097	X'00000C19'
MQCACF_BACKUP_DATE	3098	X'00000C1A'
MQCACF_BACKUP_TIME	3099	X'00000C1B'
MQCACF_SYSTEM_NAME	3100	X'00000C1C'
MQCACF_CF_STRUC_BACKUP_START	3101	X'00000C1D'
MQCACF_CF_STRUC_BACKUP_END	3102	X'00000C1E'
MQCACF_CF_STRUC_LOG_Q_MGRS	3103	X'00000C1F'
MQCACF_FROM_STORAGE_CLASS	3104	X'00000C20'
MQCACF_TO_STORAGE_CLASS	3105	X'00000C21'
MQCACF_STORAGE_CLASS_NAMES	3106	X'00000C22'
MQCACF_DSG_NAME	3108	X'00000C24'
MQCACF_DB2_NAME	3109	X'00000C25'
MQCACF_SYSP_CMD_USER_ID	3110	X'00000C26'
MQCACF_SYSP_OTMA_GROUP	3111	X'00000C27'
MQCACF_SYSP_OTMA_MEMBER	3112	X'00000C28'
MQCACF_SYSP_OTMA_DRU_EXIT	3113	X'00000C29'
MQCACF_SYSP_OTMA_TPIPE_PFX	3114	X'00000C2A'
MQCACF_SYSP_ARCHIVE_PFX1	3115	X'00000C2B'
MQCACF_SYSP_ARCHIVE_UNIT1	3116	X'00000C2C'
MQCACF_SYSP_LOG_CORREL_ID	3117	X'00000C2D'
MQCACF_SYSP_UNIT_VOLSER	3118	X'00000C2E'
MQCACF_SYSP_Q_MGR_TIME	3119	X'00000C2F'
MQCACF_SYSP_Q_MGR_DATE	3120	X'00000C30'
MQCACF_SYSP_Q_MGR_RBA	3121	X'00000C31'
MQCACF_SYSP_LOG_RBA	3122	X'00000C32'
MQCACF_SYSP_SERVICE	3123	X'00000C33'
MQCACF_FROM_LISTENER_NAME	3124	X'00000C34'
MQCACF_TO_LISTENER_NAME	3125	X'00000C35'
MQCACF_FROM_SERVICE_NAME	3126	X'00000C36'
MQCACF_TO_SERVICE_NAME	3127	X'00000C37'
MQCACF_LAST_PUT_DATE	3128	X'00000C38'
MQCACF_LAST_PUT_TIME	3129	X'00000C39'
MQCACF_LAST_GET_DATE	3130	X'00000C3A'
MQCACF_LAST_GET_TIME	3131	X'00000C3B'
MQCACF_OPERATION_DATE	3132	X'00000C3C'
MQCACF_OPERATION_TIME	3133	X'00000C3D'
MQCACF_ACTIVITY_DESC	3134	X'00000C3E'
MQCACF_APPL_IDENTITY_DATA	3135	X'00000C3F'

表 35. 定数の値 (続き)		
名前	10 進数値	16 進値
MQCACF_APPL_ORIGIN_DATA	3136	X'00000C40'
MQCACF_PUT_DATE	3137	X'00000C41'
MQCACF_PUT_TIME	3138	X'00000C42'
MQCACF_REPLY_TO_Q	3139	X'00000C43'
MQCACF_REPLY_TO_Q_MGR	3140	X'00000C44'
MQCACF_RESOLVED_Q_NAME	3141	X'00000C45'
MQCACF_STRUC_ID	3142	X'00000C46'
MQCACF_VALUE_NAME	3143	X'00000C47'
MQCACF_SERVICE_START_DATE	3144	X'00000C48'
MQCACF_SERVICE_START_TIME	3145	X'00000C49'
MQCACF_SYSP_OFFLINE_RBA	3146	X'00000C4A'
MQCACF_SYSP_ARCHIVE_PFX2	3147	X'00000C4B'
MQCACF_SYSP_ARCHIVE_UNIT2	3148	X'00000C4C'
MQCACF_TO_TOPIC_NAME	3149	X'00000C4D'
MQCACF_FROM_TOPIC_NAME	3150	X'00000C4E'
MQCACF_TOPIC_NAMES	3151	X'00000C4F'
MQCACF_SUB_NAME	3152	X'00000C50'
MQCACF_DESTINATION_Q_MGR	3153	X'00000C51'
MQCACF_DESTINATION	3154	X'00000C52'
MQCACF_SUB_USER_ID	3156	X'00000C54'
MQCACF_SUB_USER_DATA	3159	X'00000C57'
MQCACF_SUB_SELECTOR	3160	X'00000C58'
MQCACF_LAST_PUB_DATE	3161	X'00000C59'
MQCACF_LAST_PUB_TIME	3162	X'00000C5A'
MQCACF_FROM_SUB_NAME	3163	X'00000C5B'
MQCACF_TO_SUB_NAME	3164	X'00000C5C'
MQCACF_LAST_MSG_TIME	3167	X'00000C5F'
MQCACF_LAST_MSG_DATE	3168	X'00000C60'
MQCACF_SUBSCRIPTION_POINT	3169	X'00000C61'
MQCACF_FILTER	3170	X'00000C62'
MQCACF_NONE	3171	X'00000C63'
MQCACF_ADMIN_TOPIC_NAMES	3172	X'00000C64'
MQCACF_ROUTING_FINGER_PRINT	3173	X'00000C65'
MQCACF_APPL_DESC	3174	X'00000C66'
MQCACF_Q_MGR_START_DATE	3175	X'00000C67'
MQCACF_Q_MGR_START_TIME	3176	X'00000C68'
MQCACF_FROM_COMM_INFO_NAME	3177	X'00000C69'
MQCACF_TO_COMM_INFO_NAME	3178	X'00000C6A'
MQCACF_CF_OFFLOAD_SIZE1	3179	X'00000C6B'

表 35. 定数の値 (続き)		
名前	10 進数値	16 進値
MQCACF_CF_OFFLOAD_SIZE2	3180	X'00000C6C'
MQCACF_CF_OFFLOAD_SIZE3	3181	X'00000C6D'
MQCACF_CF_SMDS_GENERIC_NAME	3182	X'00000C6E'
MQCACF_CF_SMDS	3183	X'00000C6F'
MQCACF_RECOVERY_DATE	3184	X'00000C70'
MQCACF_RECOVERY_TIME	3185	X'00000C71'
MQCACF_CF_SMDSCONN	3186	X'00000C72'
MQCACF_CF_STRUC_NAME	3187	X'00000C73'
MQCACF_ALTERNATE_USERID	3188	X'00000C74'
MQCACF_CHAR_ATTRS	3189	X'00000C75'
MQCACF_DYNAMIC_Q_NAME	3190	X'00000C76'
MQCACF_HOST_NAME	3191	X'00000C77'
MQCACF_MQCB_NAME	3192	X'00000C78'
MQCACF_OBJECT_STRING	3193	X'00000C79'
MQCACF_RESOLVED_LOCAL_Q_MGR	3194	X'00000C7A'
MQCACF_RESOLVED_LOCAL_Q_NAME	3195	X'00000C7B'
MQCACF_RESOLVED_OBJECT_STRING	3196	X'00000C7C'
MQCACF_RESOLVED_Q_MGR	3197	X'00000C7D'
MQCACF_SELECTION_STRING	3198	X'00000C7E'
MQCACF_XA_INFO	3199	X'00000C7F'
MQCACF_APPL_FUNCTION	3200	X'00000C80'
MQCACF_XQH_REMOTE_Q_NAME	3201	X'00000C81'
MQCACF_XQH_REMOTE_Q_MGR	3202	X'00000C82'
MQCACF_XQH_PUT_TIME	3203	X'00000C83'
MQCACF_XQH_PUT_DATE	3204	X'00000C84'
MQCACF_EXCL_OPERATOR_MESSAGES	3205	X'00000C85'
MQCACF_CSP_USER_IDENTIFIER	3206	X'00000C86'
MQCACF_AMQP_CLIENT_ID	3207	X'00000C87'
MQCACF_ARCHIVE_LOG_EXTENT_NAME	3208	X'00000C88'
MQCACF_APPL_IMMOVABLE_DATE	3209	X'00000C89'
MQCACF_APPL_IMMOVABLE_TIME	3210	X'00000C8A'
MQCACF_NHA_INSTANCE_NAME	3211	X'00000C8B'
 MQCACF_イベント重複元	3232	X'00000CA0'
MQCACF_LAST_USED	3232	X'00000CA0'

MQCACH_* (コマンド形式の文字チャネル・パラメーター・タイプ)

表 36. 定数の値		
名前	10 進数値	16 進値
MQCACH_FIRST	3501	X'00000DAD'

表 36. 定数の値 (続き)		
名前	10 進数値	16 進値
MQCACH_CHANNEL_NAME	3501	X'00000DAD'
MQCACH_DESC	3502	X'00000DAE'
MQCACH_MODE_NAME	3503	X'00000DAF'
MQCACH_TP_NAME	3504	X'00000DB0'
MQCACH_XMIT_Q_NAME	3505	X'00000DB1'
MQCACH_CONNECTION_NAME	3506	X'00000DB2'
MQCACH_MCA_NAME	3507	X'00000DB3'
MQCACH_SEC_EXIT_NAME	3508	X'00000DB4'
MQCACH_MSG_EXIT_NAME	3509	X'00000DB5'
MQCACH_SEND_EXIT_NAME	3510	X'00000DB6'
MQCACH_RCV_EXIT_NAME	3511	X'00000DB7'
MQCACH_CHANNEL_NAMES	3512	X'00000DB8'
MQCACH_SEC_EXIT_USER_DATA	3513	X'00000DB9'
MQCACH_MSG_EXIT_USER_DATA	3514	X'00000DBA'
MQCACH_SEND_EXIT_USER_DATA	3515	X'00000DBB'
MQCACH_RCV_EXIT_USER_DATA	3516	X'00000DBC'
MQCACH_USER_ID	3517	X'00000DBD'
MQCACH_PASSWORD	3518	X'00000DBE'
MQCACH_LOCAL_ADDRESS	3520	X'00000DC0'
MQCACH_LOCAL_NAME	3521	X'00000DC1'
MQCACH_LAST_MSG_TIME	3524	X'00000DC4'
MQCACH_LAST_MSG_DATE	3525	X'00000DC5'
MQCACH_MCA_USER_ID	3527	X'00000DC7'
MQCACH_CHANNEL_START_TIME	3528	X'00000DC8'
MQCACH_CHANNEL_START_DATE	3529	X'00000DC9'
MQCACH_MCA_JOB_NAME	3530	X'00000DCA'
MQCACH_LAST_LUWID	3531	X'00000DCB'
MQCACH_CURRENT_LUWID	3532	X'00000DCC'
MQCACH_FORMAT_NAME	3533	X'00000DCD'
MQCACH_MR_EXIT_NAME	3534	X'00000DCE'
MQCACH_MR_EXIT_USER_DATA	3535	X'00000DCF'
MQCACH_SSL_CIPHER_SPEC	3544	X'00000DD8'
MQCACH_SSL_PEER_NAME	3545	X'00000DD9'
MQCACH_SSL_HANDSHAKE_STAGE	3546	X'00000DDA'
MQCACH_SSL_SHORT_PEER_NAME	3547	X'00000ddb'
MQCACH_REMOTE_APPL_TAG	3548	X'00000DDC'
MQCACH_SSL_CERT_USER_ID	3549	X'00000DDD'
MQCACH_SSL_CERT_ISSUER_NAME	3550	X'00000DDE'
MQCACH_LU_NAME	3551	X'00000DDF'

表 36. 定数の値 (続き)		
名前	10 進数値	16 進値
MQCACH_IP_ADDRESS	3552	X'00000DE0'
MQCACH_TCP_NAME	3553	X'00000DE1'
MQCACH_LISTENER_NAME	3554	X'00000DE2'
MQCACH_LISTENER_DESC	3555	X'00000DE3'
MQCACH_LISTENER_START_DATE	3556	X'00000DE4'
MQCACH_LISTENER_START_TIME	3557	X'00000DE5'
MQCACH_SSL_KEY_RESET_DATE	3558	X'00000DE6'
MQCACH_SSL_KEY_RESET_TIME	3559	X'00000DE7'
MQCACH_LAST_USED	3559	X'00000DE7'

MQCADSD_* (CICS 情報ヘッダー ADS 記述子)

表 37. 定数の値		
名前	10 進数値	16 進値
MQCADSD_NONE	0	X'00000000'
MQCADSD_SEND	1	X'00000001'
MQCADSD_RECV	16	X'00000010'
MQCADSD_MSGFORMAT	256	X'00000100'

MQCAFTY_* (接続類縁性値)

表 38. 定数の値		
名前	10 進数値	16 進値
MQCAFTY_NONE	0	X'00000000'
MQCAFTY_PREFERRED	1	X'00000001'

MQCAMO_* (コマンド形式の文字モニター・パラメーター・タイプ)

表 39. 定数の値		
名前	10 進数値	16 進値
MQCAMO_FIRST	2701	X'00000A8D'
MQCAMO_CLOSE_DATE	2701	X'00000A8D'
MQCAMO_CLOSE_TIME	2702	X'00000A8E'
MQCAMO_CONN_DATE	2703	X'00000A8F'
MQCAMO_CONN_TIME	2704	X'00000A90'
MQCAMO_DISC_DATE	2705	X'00000A91'
MQCAMO_DISC_TIME	2706	X'00000A92'
MQCAMO_END_DATE	2707	X'00000A93'
MQCAMO_END_TIME	2708	X'00000A94'
MQCAMO_OPEN_DATE	2709	X'00000A95'
MQCAMO_OPEN_TIME	2710	X'00000A96'
MQCAMO_START_DATE	2711	X'00000A97'
MQCAMO_START_TIME	2712	X'00000A98'

表 39. 定数の値 (続き)		
名前	10 進数値	16 進値
MQCAMO_LAST_USED	2712	X'00000A98'

MQCBC_* (MQCBC 定数構造体)

表 40. 定数の構造	
名前	構造体
MQCBC_STRUC_ID	"CBC↵"
MQCBC_STRUC_ID_ARRAY	'C','B','C','↵'

注: 記号↵は、単一のブランク文字を表します。

表 41. 定数の値		
名前	10 進数値	16 進値
MQCBC_VERSION_1	1	X'00000001'
MQCBC_CURRENT_VERSION	1	X'00000001'

MQCBCF_* (MQCBC 定数フラグ)

表 42. 定数の値		
名前	10 進数値	16 進値
MQCBCF_NONE	0	X'00000000'
MQCBCF_READA_BUFFER_EMPTY	1	X'00000001'

MQCBCT_* (MQCBC 定数コールバック・タイプ)

表 43. 定数の値		
名前	10 進数値	16 進値
MQCBCT_START_CALL	1	X'00000001'
MQCBCT_STOP_CALL	2	X'00000002'
MQCBCT_REGISTER_CALL	3	X'00000003'
MQCBCT_DEREGISTER_CALL	4	X'00000004'
MQCBCT_EVENT_CALL	5	X'00000005'
MQCBCT_MSG_REMOVED	6	X'00000006'
MQCBCT_MSG_NOT_REMOVED	7	X'00000007'

MQCBD_* (MQCBD 定数構造体)

表 44. 定数の構造	
名前	構造体
MQCBD_STRUC_ID	"CBD↵"
MQCBD_STRUC_ID_ARRAY	'C','B','D','↵'

注: 記号↵は、単一のブランク文字を表します。

表 45. 定数の値		
名前	10 進数値	16 進値
MQCBD_VERSION_1	1	X'00000001'
MQCBD_CURRENT_VERSION	1	X'00000001'

MQCBDO_* (MQCBD 定数コールバック・オプション)

表 46. 定数の値		
名前	10 進数値	16 進値
MQCBDO_NONE	0	X'00000000'
MQCBDO_START_CALL	1	X'00000001'
MQCBDO_STOP_CALL	4	X'00000004'
MQCBDO_REGISTER_CALL	256	X'00000100'
MQCBDO_DEREGISTER_CALL	512	X'00000200'
MQCBDO_FAIL_IF QUIESCING	8192	X'00002000'

MQCBO_* (mqCreateBag のバッグ作成オプション)

表 47. 定数の値		
名前	10 進数値	16 進値
MQCBO_NONE	0	X'00000000'
MQCBO_USER_BAG	0	X'00000000'
MQCBO_ADMIN_BAG	1	X'00000001'
MQCBO_COMMAND_BAG	16	X'00000010'
MQCBO_SYSTEM_BAG	32	X'00000020'
MQCBO_GROUP_BAG	64	X'00000040'
MQCBO_LIST_FORM_ALLOWED	2	X'00000002'
MQCBO_LIST_FORM_INHIBITED	0	X'00000000'
MQCBO_REORDER_AS_REQUIRED	4	X'00000004'
MQCBO_DO_NOT_REORDER	0	X'00000000'
MQCBO_CHECK_SELECTORS	8	X'00000008'
MQCBO_DO_NOT_CHECK_SELECTORS	0	X'00000000'

MQCBT_* (MQCBD 定数。これはコールバック関数の型です。)

表 48. 定数の値		
名前	10 進数値	16 進値
MQCBT_MESSAGE_CONSUMER	1	X'00000001'
MQCBT_EVENT_HANDLER	2	X'00000002'

MQCC_* (完了コード)

表 49. 定数の値		
名前	10 進数値	16 進値
MQCC_OK	0	X'00000000'
MQCC_WARNING	1	X'00000001'

表 49. 定数の値 (続き)		
名前	10 進数値	16 進値
MQCC_FAILED	2	X'00000002'
MQCC_UNKNOWN	-1	X'FFFFFFFF'

MQCCSI_* (コード化文字セット ID)

表 50. 定数の値		
名前	10 進数値	16 進値
MQCCSI_UNDEFINED	0	X'00000000'
MQCCSI_DEFAULT	0	X'00000000'
MQCCSI_Q_MGR	0	X'00000000'
MQCCSI_INHERIT	-2	X'FFFFFFFE'
MQCCSI_EMBEDDED	-1	X'FFFFFFFF'
MQCCSI_APPL	-3	X'FFFFFFFD'

MQCCT_* (CICS 情報ヘッダー会話型タスク・オプション)

表 51. 定数の値		
名前	10 進数値	16 進値
MQCCT_YES	1	X'00000001'
MQCCT_NO	0	X'00000000'

MQCD_* (チャネル定義構造体)

表 52. 定数の値		
名前	10 進数値	16 進値
MQCD_VERSION_1	1	X'00000001'
MQCD_VERSION_2	2	X'00000002'
MQCD_VERSION_3	3	X'00000003'
MQCD_VERSION_4	4	X'00000004'
MQCD_VERSION_5	5	X'00000005'
MQCD_VERSION_6	6	X'00000006'
MQCD_VERSION_7	7	X'00000007'
MQCD_VERSION_8	8	X'00000008'
MQCD_VERSION_9	9	X'00000009'
MQCD_VERSION_10	10	X'0000000A'
MQCD_VERSION_11	11	X'0000000B'
MQCD_VERSION_12	12	X'0000000C'
 MQCD_VERSION_13	13	X'0000000D'
 MQCD_CURRENT_VERSION	12	X'0000000C'
 MQCD_CURRENT_VERSION	13	X'0000000D'
MQCD_LENGTH_4	(value differs by platform or version)	(value differs by platform or version)

表 52. 定数の値 (続き)		
名前	10 進数値	16 進値
MQCD_LENGTH_5	(value differs by platform or version)	(value differs by platform or version)
MQCD_LENGTH_6	(value differs by platform or version)	(value differs by platform or version)
MQCD_LENGTH_7	(value differs by platform or version)	(value differs by platform or version)
MQCD_LENGTH_8	(value differs by platform or version)	(value differs by platform or version)
MQCD_LENGTH_9	(value differs by platform or version)	(value differs by platform or version)
MQCD_LENGTH_10	(value differs by platform or version)	(value differs by platform or version)
MQCD_LENGTH_11	(value differs by platform or version)	(value differs by platform or version)
MQCD_LENGTH_12	(value differs by platform or version)	(value differs by platform or version)
 MQCD_LENGTH_13	(value differs by platform or version)	(value differs by platform or version)
MQCD_CURRENT_LENGTH	(value differs by platform or version)	(value differs by platform or version)

MQCDC_* (チャネル・データ変換)

表 53. 定数の値		
名前	10 進数値	16 進値
MQCDC_SENDER_CONVERSION	1	X'00000001'
MQCDC_NO_SENDER_CONVERSION	0	X'00000000'

MQCERT_* (証明書検証ポリシー・タイプ)

MQ_CERT_VAL_POLICY_DEFAULT	0	X'00000000'
MQ_CERT_VAL_POLICY_ANY	0	X'00000000'
MQ_CERT_VAL_POLICY_RFC5280	1	X'00000001'
MQ_CERT_VAL_POLICY_NONE	2	X'00000002'

MQCF_* (機能フラグ)

表 54. 定数の値		
名前	10 進数値	16 進値
MQCF_NONE	0	X'00000000'
MQCF_DIST_LISTS	1	X'00000001'

MQCFAC_* (CICS 情報ヘッダー機能)

表 55. 定数名と値	
名前	16 進値
MQCFAC_NONE	X'00...00' (8 つのヌル)

表 55. 定数名と値 (続き)	
名前	16 進値
MQCFAC_NONE_ARRAY	'\0','\0',... (8つのヌル)

MQCFBF_* (コマンド形式のバイト・ストリング・フィルター・パラメーター構造体)

表 56. 定数の値		
名前	10 進数値	16 進値
MQCFBF_STRUC_LENGTH_FIXED	20	X'00000014'

MQCFBS_* (コマンド形式のバイト・ストリング・パラメーター構造体)

表 57. 定数の値		
名前	10 進数値	16 進値
MQCFBS_STRUC_LENGTH_FIXED	16	X'00000010'

MQCFCC_* (コマンド形式のヘッダー制御オプション)

表 58. 定数の値		
名前	10 進数値	16 進値
MQCFCC_LAST	1	X'00000001'
MQCFCC_NOT_LAST	0	X'00000000'

MQCFGR_* (コマンド形式のグループ・パラメーター構造体)

表 59. 定数の値		
名前	10 進数値	16 進値
MQCFGR_STRUC_LENGTH	16	X'00000010'

MQCFH_* (コマンド形式のヘッダー構造体)

表 60. 定数の値		
名前	10 進数値	16 進値
MQCFH_STRUC_LENGTH	36	X'00000024'
MQCFH_VERSION_1	1	X'00000001'
MQCFH_VERSION_2	2	X'00000002'
MQCFH_VERSION_3	3	X'00000003'
MQCFH_CURRENT_VERSION	3	X'00000003'

MQCFIF_* (コマンド形式の整数フィルター・パラメーター構造体)

表 61. 定数の値		
名前	10 進数値	16 進値
MQCFIF_STRUC_LENGTH	20	X'00000014'

MQCFIL_* (コマンド形式の整数リスト・パラメーター構造体)

表 62. 定数の値		
名前	10 進数値	16 進値
MQCFIL_STRUC_LENGTH_FIXED	16	X'00000010'

MQCFIL64_* (コマンド形式の 64 ビット整数リスト・パラメーター構造体)

表 63. 定数の値		
名前	10 進数値	16 進値
MQCFIL64_STRUC_LENGTH_FIXED	16	X'00000010'

MQCFIN_* (コマンド形式の整数パラメーター構造体)

表 64. 定数の値		
名前	10 進数値	16 進値
MQCFIN_STRUC_LENGTH	16	X'00000010'

MQCFIN64_* (コマンド形式の 64 ビット整数パラメーター構造体)

表 65. 定数の値		
名前	10 進数値	16 進値
MQCFIN64_STRUC_LENGTH	24	X'00000018'

MQCFO_* (コマンド形式のリポジトリ最新表示オプションおよび コマンド形式のキュー除去オプション)

コマンド形式のリポジトリ最新表示オプション

表 66. 定数の値		
名前	10 進数値	16 進値
MQCFO_REFRESH_REPOSITORY_YES	1	X'00000001'
MQCFO_REFRESH_REPOSITORY_NO	0	X'00000000'

コマンド形式のキュー除去オプション

表 67. 定数の値		
名前	10 進数値	16 進値
MQCFO_REMOVE_QUEUES_YES	1	X'00000001'
MQCFO_REMOVE_QUEUES_NO	0	X'00000000'

MQCFOP_* (コマンド形式のフィルター演算子)

表 68. 定数の値		
名前	10 進数値	16 進値
MQCFOP_LESS	1	X'00000001'
MQCFOP_EQUAL	2	X'00000002'
MQCFOP_GREATER	4	X'00000004'
MQCFOP_NOT_LESS	6	X'00000006'

表 68. 定数の値 (続き)		
名前	10 進数値	16 進値
MQCFOP_NOT_EQUAL	5	X'00000005'
MQCFOP_NOT_GREATER	3	X'00000003'
MQCFOP_LIKE	18	X'00000012'
MQCFOP_NOT_LIKE	21	X'00000015'
MQCFOP_CONTAINS	10	X'0000000A'
MQCFOP_EXCLUDES	13	X'0000000D'
MQCFOP_CONTAINS_GEN	26	X'0000001A'
MQCFOP_EXCLUDES_GEN	29	X'0000001D'

MQCFR_* (CF 回復可能性)

表 69. 定数の値		
名前	10 進数値	16 進値
MQCFR_YES	1	X'00000001'
MQCFR_NO	0	X'00000000'

MQCFSF_* (コマンド形式のストリング・フィルター・パラメーター構造体)

表 70. 定数の値		
名前	10 進数値	16 進値
MQCFSF_STRUC_LENGTH_FIXED	24	X'00000018'

MQCFSL_* (コマンド形式のストリング・リスト・パラメーター構造体)

表 71. 定数の値		
名前	10 進数値	16 進値
MQCFSL_STRUC_LENGTH_FIXED	24	X'00000018'

MQCFST_* (コマンド形式のストリング・パラメーター構造体)

表 72. 定数の値		
名前	10 進数値	16 進値
MQCFST_STRUC_LENGTH_FIXED	20	X'00000014'

MQCFSTATUS_* (コマンド形式の CF 状況)

表 73. 定数の値		
名前	10 進数値	16 進値
MQCFSTATUS_NOT_FOUND	0	X'00000000'
MQCFSTATUS_ACTIVE	1	X'00000001'
MQCFSTATUS_IN_RECOVER	2	X'00000002'
MQCFSTATUS_IN_BACKUP	3	X'00000003'
MQCFSTATUS_FAILED	4	X'00000004'
MQCFSTATUS_NONE	5	X'00000005'
MQCFSTATUS_UNKNOWN	6	X'00000006'

表 73. 定数の値 (続き)		
名前	10 進数値	16 進値
MQCFSTATUS_ADMIN_INCOMPLETE	20	X'00000014'
MQCFSTATUS_NEVER_USED	21	X'00000015'
MQCFSTATUS_NO_BACKUP	22	X'00000016'
MQCFSTATUS_NOT_FAILED	23	X'00000017'
MQCFSTATUS_NOT_RECOVERABLE	24	X'00000018'
MQCFSTATUS_XES_ERROR	25	X'00000019'

MQCFT_* (コマンド形式の構造体のタイプ)

表 74. 定数の値		
名前	10 進数値	16 進値
MQCFT_NONE	0	X'00000000'
MQCFT_COMMAND	1	X'00000001'
MQCFT_RESPONSE	2	X'00000002'
MQCFT_INTEGER	3	X'00000003'
MQCFT_STRING	4	X'00000004'
MQCFT_INTEGER_LIST	5	X'00000005'
MQCFT_STRING_LIST	6	X'00000006'
MQCFT_EVENT	7	X'00000007'
MQCFT_USER	8	X'00000008'
MQCFT_BYTE_STRING	9	X'00000009'
MQCFT_TRACE_ROUTE	10	X'0000000A'
MQCFT_REPORT	12	X'0000000C'
MQCFT_INTEGER_FILTER	13	X'0000000D'
MQCFT_STRING_FILTER	14	X'0000000E'
MQCFT_BYTE_STRING_FILTER	15	X'0000000F'
MQCFT_COMMAND_XR	16	X'00000010'
MQCFT_XR_MSG	17	X'00000011'
MQCFT_XR_ITEM	18	X'00000012'
MQCFT_XR_SUMMARY	19	X'00000013'
MQCFT_GROUP	20	X'00000014'
MQCFT_STATISTICS	21	X'00000015'
MQCFT_ACCOUNTING	22	X'00000016'
MQCFT_INTEGER64	23	X'00000017'
MQCFT_INTEGER64_LIST	25	X'00000019'

MQCFTYPE_* (コマンド形式の CF タイプ)

表 75. 定数の値		
名前	10 進数値	16 進値
MQCFTYPE_APPL	0	X'00000000'
MQCFTYPE_ADMIN	1	X'00000001'

MQCFUNC_* (CICS 情報ヘッダー関数)

表 76. 定数の構造	
名前	構造体
MQCFUNC_MQCONN	"CONN"
MQCFUNC_MQGET	"GET"
MQCFUNC_MQINQ	"INQ"
MQCFUNC_MQOPEN	"OPEN"
MQCFUNC_MQPUT	"PUT"
MQCFUNC_MQPUT1	"PUT1"
MQCFUNC_NONE	" "
MQCFUNC_MQCONN_ARRAY	'C','O','N','N'
MQCFUNC_MQGET_ARRAY	'G','E','T',' '
MQCFUNC_MQINQ_ARRAY	'I','N','Q',' '
MQCFUNC_MQOPEN_ARRAY	'O','P','E','N'
MQCFUNC_MQPUT_ARRAY	'P','U','T',' '
MQCFUNC_MQPUT1_ARRAY	'P','U','T','1'
MQCFUNC_NONE_ARRAY	' ',' ',' ',' '

注: 記号は、単一のブランク文字を表します。

MQCGWI_* (CICS 情報ヘッダー待機間隔取得)

表 77. 定数の値		
名前	10 進数値	16 進値
MQCGWI_DEFAULT	-2	X'FFFFFFFE'

MQCHAD_* (チャンネル自動定義)

表 78. 定数の値		
名前	10 進数値	16 進値
MQCHAD_DISABLED	0	X'00000000'
MQCHAD_ENABLED	1	X'00000001'

MQCHIDS_* (コマンド形式の未確定状況)

表 79. 定数の値		
名前	10 進数値	16 進値
MQCHIDS_NOT_INDOUBT	0	X'00000000'
MQCHIDS_INDOUBT	1	X'00000001'

MQCHLD_* (コマンド形式のチャンネル処理)

表 80. 定数の値		
名前	10 進数値	16 進値
MQCHLD_ALL	-1	X'FFFFFFFF'
MQCHLD_DEFAULT	1	X'00000001'

表 80. 定数の値 (続き)		
名前	10 進数値	16 進値
MQCHLD_SHARED	2	X'00000002'
MQCHLD_PRIVATE	4	X'00000004'
MQCHLD_FIXSHARED	5	X'00000005'

MQCHS_* (コマンド形式のチャネル状況)

表 81. 定数の値		
名前	10 進数値	16 進値
MQCHS_INACTIVE	0	X'00000000'
MQCHS_BINDING	1	X'00000001'
MQCHS_STARTING	2	X'00000002'
MQCHS_RUNNING	3	X'00000003'
MQCHS_STOPPING	4	X'00000004'
MQCHS_RETRYING	5	X'00000005'
MQCHS_STOPPED	6	X'00000006'
MQCHS_REQUESTING	7	X'00000007'
MQCHS_PAUSED	8	X'00000008'
MQCHS_INITIALIZING	13	X'0000000D'
MQCHS_SWITCHING	14	X'0000000E'

MQCHSH_* (コマンド形式のチャネル共有再始動オプション)

表 82. 定数の値		
名前	10 進数値	16 進値
MQCHSH_RESTART_NO	0	X'00000000'
MQCHSH_RESTART_YES	1	X'00000001'

MQCHSR_* (コマンド形式のチャネル停止オプション)

表 83. 定数の値		
名前	10 進数値	16 進値
MQCHSR_STOP_NOT_REQUESTED	0	X'00000000'
MQCHSR_STOP_REQUESTED	1	X'00000001'

MQCHSSTATE_* (コマンド形式のチャネル副状態)

表 84. 定数の値		
名前	10 進数値	16 進値
MQCHSSTATE_OTHER	0	X'00000000'
MQCHSSTATE_END_OF_BATCH	100	X'00000064'
MQCHSSTATE_SENDING	200	X'000000C8'
MQCHSSTATE_RECEIVING	300	X'0000012C'
MQCHSSTATE_SERIALIZING	400	X'00000190'
MQCHSSTATE_RESYNCHING	500	X'000001F4'

表 84. 定数の値 (続き)		
名前	10 進数値	16 進値
MQCHSSTATE_HEARTBEATING	600	X'00000258'
MQCHSSTATE_IN_SCYEXIT	700	X'000002BC'
MQCHSSTATE_IN_RCVEXIT	800	X'00000320'
MQCHSSTATE_IN_SENDEXIT	900	X'00000384'
MQCHSSTATE_IN_MSGEXIT	1000	X'000003E8'
MQCHSSTATE_IN_MREXIT	1100	X'0000044C'
MQCHSSTATE_IN_CHADEXIT	1200	X'000004B0'
MQCHSSTATE_NET_CONNECTING	1250	X'000004E2'
MQCHSSTATE_SSL_HANDSHAKING	1300	X'00000514'
MQCHSSTATE_NAME_SERVER	1400	X'00000578'
MQCHSSTATE_IN_MQPUT	1500	X'000005DC'
MQCHSSTATE_IN_MQGET	1600	X'00000640'
MQCHSSTATE_IN_MQI_CALL	1700	X'000006A4'
MQCHSSTATE_COMPRESSING	1800	X'00000708'

MQCHT_* (チャンネル・タイプ)

表 85. 定数の値		
名前	10 進数値	16 進値
MQCHT_SENDER	1	X'00000001'
MQCHT_SERVER	2	X'00000002'
MQCHT_RECEIVER	3	X'00000003'
MQCHT_REQUESTER	4	X'00000004'
MQCHT_ALL	5	X'00000005'
MQCHT_CLNTCONN	6	X'00000006'
MQCHT_SVRCONN	7	X'00000007'
MQCHT_CLUSRCVR	8	X'00000008'
MQCHT_CLUSSDR	9	X'00000009'

MQCHTAB_* (コマンド形式のチャンネル・テーブル・タイプ)

表 86. 定数の値		
名前	10 進数値	16 進値
MQCHTAB_Q_MGR	1	X'00000001'
MQCHTAB_CLNTCONN	2	X'00000002'

MQCI_* (関連 ID)

表 87. 定数名と値	
名前	値
MQCI_NONE	X'00...00' (24 バイト)
MQCI_NONE_ARRAY	'\0', '\0', ... (24 バイト)

表 87. 定数名と値 (続き)	
名前	値
MQCI_NEW_SESSION	X'414D5121...'
MQCI_NEW_SESSION_ARRAY	'\x41','\x4D','\51','\x21',...

MQCIH_* (CICS 情報ヘッダー構造体およびフラグ)

CICS 情報ヘッダー構造体

表 88. 定数の構造	
名前	構造体
MQCIH_STRUC_ID	"CIH"
MQCIH_STRUC_ID_ARRAY	'C','I','H',' '

注: 記号-は、単一の空白文字を表します。

表 89. 定数の値		
名前	10 進数値	16 進値
MQCIH_VERSION_1	1	X'00000001'
MQCIH_VERSION_2	2	X'00000002'
MQCIH_CURRENT_VERSION	2	X'00000002'
MQCIH_LENGTH_1	164	X'000000A4'
MQCIH_LENGTH_2	180	X'000000B4'
MQCIH_CURRENT_LENGTH	180	X'000000B4'

CICS 情報ヘッダー・フラグ

表 90. 定数の値		
名前	10 進数値	16 進値
MQCIH_NONE	0	X'00000000'
MQCIH_PASS_EXPIRATION	1	X'00000001'
MQCIH_UNLIMITED_EXPIRATION	0	X'00000000'
MQCIH_REPLY_WITHOUT_NULLS	2	X'00000002'
MQCIH_REPLY_WITH_NULLS	0	X'00000000'
MQCIH_SYNC_ON_RETURN	4	X'00000004'
MQCIH_NO_SYNC_ON_RETURN	0	X'00000000'

MQCLCT_* (クラスター・キャッシュ・タイプ)

表 91. 定数の値		
名前	10 進数値	16 進値
MQCLCT_STATIC	0	X'00000000'
MQCLCT_DYNAMIC	1	X'00000001'

MQCLRS_* (コマンド形式のトピック・ストリング消去有効範囲)

表 92. 定数の値		
名前	10 進数値	16 進値
MQCLRS_LOCAL	1	X'00000001'
MQCLRS_GLOBAL	2	X'00000002'

MQCLRT_* (コマンド形式のトピック・ストリング消去タイプ)

表 93. 定数の値		
名前	10 進数値	16 進値
MQCLRT_RETAINED	1	X'00000001'

MQCLT_* (CICS 情報ヘッダー・リンク・タイプ)

表 94. 定数の値		
名前	10 進数値	16 進値
MQCLT_PROGRAM	1	X'00000001'
MQCLT_TRANSACTION	2	X'00000002'

MQCLWL_* (クラスター・ワークロード)

表 95. 定数の値		
名前	10 進数値	16 進値
MQCLWL_USEQ_LOCAL	0	X'00000000'
MQCLWL_USEQ_ANY	1	X'00000001'
MQCLWL_USEQ_AS_Q_MGR	-3	X'FFFFFFFFD'

MQCLXQ_* (クラスター伝送キュー・タイプ)

MQCLXQ_* は、DEFCLXQ キュー・マネージャー属性で設定できる値です。この **DEFCLXQ** 属性は、cluster-sender チャンネルがメッセージを受信し、cluster-receiver チャンネルに送信する際に、デフォルトでどの送信キューが選択されるかを制御します。

表 96. 定数の値		
名前	10 進数値	16 進値
MQCLXQ_SCTQ	0	X'00000000'
MQCLXQ_CHANNEL	1	X'00000001'

関連資料

[811 ページの『DefClusterXmitQueueType \(MQLONG\)』](#)

DefClusterXmitQueueType 属性は、cluster-sender チャンネルがメッセージを受信し、cluster-receiver チャンネルに送信する際に、デフォルトでどの送信キューが選択されるかを制御します。

[Change Queue Manager](#)

[Inquire Queue Manager](#)

[Inquire Queue Manager \(応答\)](#)

[698 ページの『MQINQ - オブジェクト属性の照会』](#)

MQINQ 呼び出しは、オブジェクトの属性が入っている整数の配列と一連の文字ストリングを戻します。

MQCMD_*(コマンド・コード)

表 97. 定数の値		
名前	10 進数値	16 進値
MQCMD_NONE	0	X'00000000'
MQCMD_CHANGE_Q_MGR	1	X'00000001'
MQCMD_INQUIRE_Q_MGR	2	X'00000002'
MQCMD_CHANGE_PROCESS	3	X'00000003'
MQCMD_COPY_PROCESS	4	X'00000004'
MQCMD_CREATE_PROCESS	5	X'00000005'
MQCMD_DELETE_PROCESS	6	X'00000006'
MQCMD_INQUIRE_PROCESS	7	X'00000007'
MQCMD_CHANGE_Q	8	X'00000008'
MQCMD_CLEAR_Q	9	X'00000009'
MQCMD_COPY_Q	10	X'0000000A'
MQCMD_CREATE_Q	11	X'0000000B'
MQCMD_DELETE_Q	12	X'0000000C'
MQCMD_INQUIRE_Q	13	X'0000000D'
MQCMD_REFRESH_Q_MGR	16	X'00000010'
MQCMD_RESET_Q_STATS	17	X'00000011'
MQCMD_INQUIRE_Q_NAMES	18	X'00000012'
MQCMD_INQUIRE_PROCESS_NAMES	19	X'00000013'
MQCMD_INQUIRE_CHANNEL_NAMES	20	X'00000014'
MQCMD_CHANGE_CHANNEL	21	X'00000015'
MQCMD_COPY_CHANNEL	22	X'00000016'
MQCMD_CREATE_CHANNEL	23	X'00000017'
MQCMD_DELETE_CHANNEL	24	X'00000018'
MQCMD_INQUIRE_CHANNEL	25	X'00000019'
MQCMD_PING_CHANNEL	26	X'0000001A'
MQCMD_RESET_CHANNEL	27	X'0000001B'
MQCMD_START_CHANNEL	28	X'0000001C'
MQCMD_STOP_CHANNEL	29	X'0000001D'
MQCMD_START_CHANNEL_INIT	30	X'0000001E'
MQCMD_START_CHANNEL_LISTENER	31	X'0000001F'
MQCMD_CHANGE_NAMELIST	32	X'00000020'
MQCMD_COPY_NAMELIST	33	X'00000021'
MQCMD_CREATE_NAMELIST	34	X'00000022'
MQCMD_DELETE_NAMELIST	35	X'00000023'
MQCMD_INQUIRE_NAMELIST	36	X'00000024'
MQCMD_INQUIRE_NAMELIST_NAMES	37	X'00000025'
MQCMD_ESCAPE	38	X'00000026'
MQCMD_RESOLVE_CHANNEL	39	X'00000027'

表 97. 定数の値 (続き)		
名前	10 進数値	16 進値
MQCMD_PING_Q_MGR	40	X'00000028'
MQCMD_INQUIRE_Q_STATUS	41	X'00000029'
MQCMD_INQUIRE_CHANNEL_STATUS	42	X'0000002A'
MQCMD_CONFIG_EVENT	43	X'0000002B'
MQCMD_Q_MGR_EVENT	44	X'0000002C'
MQCMD_PERFM_EVENT	45	X'0000002D'
MQCMD_CHANNEL_EVENT	46	X'0000002E'
MQCMD_DELETE_PUBLICATION	60	X'0000003C'
MQCMD_DEREGISTER_PUBLISHER	61	X'0000003D'
MQCMD_DEREGISTER_SUBSCRIBER	62	X'0000003E'
MQCMD_PUBLISH	63	X'0000003F'
MQCMD_REGISTER_PUBLISHER	64	X'00000040'
MQCMD_REGISTER_SUBSCRIBER	65	X'00000041'
MQCMD_REQUEST_UPDATE	66	X'00000042'
MQCMD_BROKER_INTERNAL	67	X'00000043'
MQCMD_ACTIVITY_MSG	69	X'00000045'
MQCMD_INQUIRE_CLUSTER_Q_MGR	70	X'00000046'
MQCMD_RESUME_Q_MGR_CLUSTER	71	X'00000047'
MQCMD_SUSPEND_Q_MGR_CLUSTER	72	X'00000048'
MQCMD_REFRESH_CLUSTER	73	X'00000049'
MQCMD_RESET_CLUSTER	74	X'0000004A'
MQCMD_TRACE_ROUTE	75	X'0000004B'
MQCMD_REFRESH_SECURITY	78	X'0000004E'
MQCMD_CHANGE_AUTH_INFO	79	X'0000004F'
MQCMD_COPY_AUTH_INFO	80	X'00000050'
MQCMD_CREATE_AUTH_INFO	81	X'00000051'
MQCMD_DELETE_AUTH_INFO	82	X'00000052'
MQCMD_INQUIRE_AUTH_INFO	83	X'00000053'
MQCMD_INQUIRE_AUTH_INFO_NAMES	84	X'00000054'
MQCMD_INQUIRE_CONNECTION	85	X'00000055'
MQCMD_STOP_CONNECTION	86	X'00000056'
MQCMD_INQUIRE_AUTH_RECS	87	X'00000057'
MQCMD_INQUIRE_ENTITY_AUTH	88	X'00000058'
MQCMD_DELETE_AUTH_REC	89	X'00000059'
MQCMD_SET_AUTH_REC	90	X'0000005A'
MQCMD_LOGGER_EVENT	91	X'0000005B'
MQCMD_RESET_Q_MGR	92	X'0000005C'
MQCMD_CHANGE_LISTENER	93	X'0000005D'
MQCMD_COPY_LISTENER	94	X'0000005E'

表 97. 定数の値 (続き)		
名前	10 進数値	16 進値
MQCMD_CREATE_LISTENER	95	X'0000005F'
MQCMD_DELETE_LISTENER	96	X'00000060'
MQCMD_INQUIRE_LISTENER	97	X'00000061'
MQCMD_INQUIRE_LISTENER_STATUS	98	X'00000062'
MQCMD_COMMAND_EVENT	99	X'00000063'
MQCMD_CHANGE_SECURITY	100	X'00000064'
MQCMD_CHANGE_CF_STRUC	101	X'00000065'
MQCMD_CHANGE_STG_CLASS	102	X'00000066'
MQCMD_CHANGE_TRACE	103	X'00000067'
MQCMD_ARCHIVE_LOG	104	X'00000068'
MQCMD_BACKUP_CF_STRUC	105	X'00000069'
MQCMD_CREATE_BUFFER_POOL	106	X'0000006A'
MQCMD_CREATE_PAGE_SET	107	X'0000006B'
MQCMD_CREATE_CF_STRUC	108	X'0000006C'
MQCMD_CREATE_STG_CLASS	109	X'0000006D'
MQCMD_COPY_CF_STRUC	110	X'0000006E'
MQCMD_COPY_STG_CLASS	111	X'0000006F'
MQCMD_DELETE_CF_STRUC	112	X'00000070'
MQCMD_DELETE_STG_CLASS	113	X'00000071'
MQCMD_INQUIRE_ARCHIVE	114	X'00000072'
MQCMD_INQUIRE_CF_STRUC	115	X'00000073'
MQCMD_INQUIRE_CF_STRUC_STATUS	116	X'00000074'
MQCMD_INQUIRE_CMD_SERVER	117	X'00000075'
MQCMD_INQUIRE_CHANNEL_INIT	118	X'00000076'
MQCMD_INQUIRE_QSG	119	X'00000077'
MQCMD_INQUIRE_LOG	120	X'00000078'
MQCMD_INQUIRE_SECURITY	121	X'00000079'
MQCMD_INQUIRE_STG_CLASS	122	X'0000007A'
MQCMD_INQUIRE_SYSTEM	123	X'0000007B'
MQCMD_INQUIRE_THREAD	124	X'0000007C'
MQCMD_INQUIRE_TRACE	125	X'0000007D'
MQCMD_INQUIRE_USAGE	126	X'0000007E'
MQCMD_MOVE_Q	127	X'0000007F'
MQCMD_RECOVER_BSDS	128	X'00000080'
MQCMD_RECOVER_CF_STRUC	129	X'00000081'
MQCMD_RESET_TPIPE	130	X'00000082'
MQCMD_RESOLVE_INDOUBT	131	X'00000083'
MQCMD_RESUME_Q_MGR	132	X'00000084'
MQCMD_REVERIFY_SECURITY	133	X'00000085'

表 97. 定数の値 (続き)		
名前	10 進数値	16 進値
MQCMD_SET_ARCHIVE	134	X'00000086'
MQCMD_SET_LOG	136	X'00000088'
MQCMD_SET_SYSTEM	137	X'00000089'
MQCMD_START_CMD_SERVER	138	X'0000008A'
MQCMD_START_Q_MGR	139	X'0000008B'
MQCMD_START_TRACE	140	X'0000008C'
MQCMD_STOP_CHANNEL_INIT	141	X'0000008D'
MQCMD_STOP_CHANNEL_LISTENER	142	X'0000008E'
MQCMD_STOP_CMD_SERVER	143	X'0000008F'
MQCMD_STOP_Q_MGR	144	X'00000090'
MQCMD_STOP_TRACE	145	X'00000091'
MQCMD_SUSPEND_Q_MGR	146	X'00000092'
MQCMD_INQUIRE_CF_STRUC_NAMES	147	X'00000093'
MQCMD_INQUIRE_STG_CLASS_NAMES	148	X'00000094'
MQCMD_CHANGE_SERVICE	149	X'00000095'
MQCMD_COPY_SERVICE	150	X'00000096'
MQCMD_CREATE_SERVICE	151	X'00000097'
MQCMD_DELETE_SERVICE	152	X'00000098'
MQCMD_INQUIRE_SERVICE	153	X'00000099'
MQCMD_INQUIRE_SERVICE_STATUS	154	X'0000009A'
MQCMD_START_SERVICE	155	X'0000009B'
MQCMD_STOP_SERVICE	156	X'0000009C'
MQCMD_DELETE_BUFFER_POOL	157	X'0000009D'
MQCMD_DELETE_PAGE_SET	158	X'0000009E'
MQCMD_CHANGE_BUFFER_POOL	159	X'0000009F'
MQCMD_CHANGE_PAGE_SET	160	X'000000A0'
MQCMD_INQUIRE_Q_MGR_STATUS	161	X'000000A1'
MQCMD_CREATE_LOG	162	X'000000A2'
MQCMD_STATISTICS_MQI	164	X'000000A4'
MQCMD_STATISTICS_Q	165	X'000000A5'
MQCMD_STATISTICS_CHANNEL	166	X'000000A6'
MQCMD_ACCOUNTING_MQI	167	X'000000A7'
MQCMD_ACCOUNTING_Q	168	X'000000A8'
MQCMD_INQUIRE_AUTH_SERVICE	169	X'000000A9'
MQCMD_CHANGE_TOPIC	170	X'000000AA'
MQCMD_COPY_TOPIC	171	X'000000AB'
MQCMD_CREATE_TOPIC	172	X'000000AC'
MQCMD_DELETE_TOPIC	173	X'000000AD'
MQCMD_INQUIRE_TOPIC	174	X'000000AE'

表 97. 定数の値 (続き)		
名前	10 進数値	16 進値
MQCMD_INQUIRE_TOPIC_NAMES	175	X'000000AF'
MQCMD_INQUIRE_SUBSCRIPTION	176	X'000000B0'
MQCMD_CREATE_SUBSCRIPTION	177	X'000000B1'
MQCMD_CHANGE_SUBSCRIPTION	178	X'000000B2'
MQCMD_DELETE_SUBSCRIPTION	179	X'000000B3'
MQCMD_COPY_SUBSCRIPTION	181	X'000000B5'
MQCMD_INQUIRE_SUB_STATUS	182	X'000000B6'
MQCMD_INQUIRE_TOPIC_STATUS	183	X'000000B7'
MQCMD_CLEAR_TOPIC_STRING	184	X'000000B8'
MQCMD_INQUIRE_PUBSUB_STATUS	185	X'000000B9'
MQCMD_PURGE_CHANNEL	195	X'000000C3'

MQCMDI_* (コマンド形式のコマンド情報値)

表 98. 定数の値		
名前	10 進数値	16 進値
MQCMDI_CMDScope_ACCEPTED	1	X'00000001'
MQCMDI_CMDScope_GENERATED	2	X'00000002'
MQCMDI_CMDScope_COMPLETED	3	X'00000003'
MQCMDI_QSG_DISP_COMPLETED	4	X'00000004'
MQCMDI_COMMAND_ACCEPTED	5	X'00000005'
MQCMDI_CLUSTER_REQUEST_QUEUED	6	X'00000006'
MQCMDI_CHANNEL_INIT_STARTED	7	X'00000007'
MQCMDI_RECOVER_STARTED	11	X'0000000B'
MQCMDI_BACKUP_STARTED	12	X'0000000C'
MQCMDI_RECOVER_COMPLETED	13	X'0000000D'
MQCMDI_SEC_TIMER_ZERO	14	X'0000000E'
MQCMDI_REFRESH_CONFIGURATION	16	X'00000010'
MQCMDI_SEC_SIGNOFF_ERROR	17	X'00000011'
MQCMDI_IMS_BRIDGE_SUSPENDED	18	X'00000012'
MQCMDI_DB2_SUSPENDED	19	X'00000013'
MQCMDI_DB2_OBSOLETE_MSGS	20	X'00000014'
MQCMDI_SEC_UPPERCASE	21	X'00000015'
MQCMDI_SEC_MIXEDCASE	22	X'00000016'

MQCMDL_* (コマンド・レベル)

表 99. 定数名と値	
名前	値
MQCMDL_LEVEL_800	800
MQCMDL_LEVEL_801	801
MQCMDL_LEVEL_802	802

表 99. 定数名と値 (続き)	
名前	値
MQCMDL_LEVEL_900	900
MQCMDL_LEVEL_901	901
MQCMDL_LEVEL_902	902
MQCMDL_LEVEL_903	903
MQCMDL_LEVEL_904	904
MQCMDL_LEVEL_905	905
MQCMDL_LEVEL_910	910
MQCMDL_LEVEL_912	912
MQCMDL_LEVEL_913	913
MQCMDL_LEVEL_914	914
MQCMDL_LEVEL_915	915
MQCMDL_LEVEL_920	920
MQCMDL_LEVEL_921	921
MQCMDL_LEVEL_922	922
MQCMDL_LEVEL_923	923
MQCMDL_LEVEL_924	924
MQCMDL_LEVEL_925	925
MQCMDL_LEVEL_930	930
MQCMDL_LEVEL_931	931
MQCMDL_LEVEL_932	932

MQCMHO_* (メッセージ・ハンドル作成オプションおよび構造体)

メッセージ・ハンドル作成オプション構造体

表 100. 定数の構造	
名前	構造体
MQCMHO_STRUC_ID	"CMHO"
MQCMHO_STRUC_ID_ARRAY	'C', 'M', 'H', 'O'

注: 記号-は、単一の空白文字を表します。

表 101. 定数の値		
名前	10 進数値	16 進値
MQCMHO_VERSION_1	1	X'00000001'
MQCMHO_CURRENT_VERSION	1	X'00000001'

メッセージ・ハンドル作成オプション

表 102. 定数の値		
名前	10 進数値	16 進値
MQCMHO_DEFAULT_VALIDATION	0	X'00000000'
MQCMHO_NO_VALIDATION	1	X'00000001'

表 102. 定数の値 (続き)		
名前	10 進数値	16 進値
MQCMHO_VALIDATE	2	X'00000002'
MQCMHO_NONE	0	X'00000000'

MQCNO_* (接続オプションおよび構造体)

接続オプション構造体

表 103. 定数の構造	
名前	構造体
MQCNO_STRUC_ID	"CNO-"
MQCNO_STRUC_ID_ARRAY	'C','N','O','_'

注: 記号-は、単一の空白文字を表します。

表 104. 定数の値		
名前	10 進数値	16 進値
MQCNO_VERSION_1	1	X'00000001'
MQCNO_VERSION_2	2	X'00000002'
MQCNO_VERSION_3	3	X'00000003'
MQCNO_VERSION_4	4	X'00000004'
MQCNO_VERSION_5	5	X'00000005'
MQCNO_CURRENT_VERSION	5	X'00000005'

接続オプション

表 105. 定数の値		
名前	10 進数値	16 進値
MQCNO_STANDARD_BINDING	0	X'00000000'
MQCNO_FASTPATH_BINDING	1	X'00000001'
MQCNO_SERIALIZE_CONN_TAG_Q_MGR	2	X'00000002'
MQCNO_SERIALIZE_CONN_TAG_QSG	4	X'00000004'
MQCNO_RESTRICT_CONN_TAG_Q_MGR	8	X'00000008'
MQCNO_RESTRICT_CONN_TAG_QSG	16	X'00000010'
MQCNO_HANDLE_SHARE_NONE	32	X'00000020'
MQCNO_HANDLE_SHARE_BLOCK	64	X'00000040'
MQCNO_HANDLE_SHARE_NO_BLOCK	128	X'00000080'
MQCNO_SHARED_BINDING	256	X'00000100'
MQCNO_ISOLATED_BINDING	512	X'00000200'
MQCNO_LOCAL_BINDING	1024	X'00000400'
MQCNO_CLIENT_BINDING	2048	X'00000800'
MQCNO_ACCOUNTING_MQI_ENABLED	4096	X'00001000'
MQCNO_ACCOUNTING_MQI_DISABLED	8192	X'00002000'
MQCNO_ACCOUNTING_Q_ENABLED	16384	X'00004000'

表 105. 定数の値 (続き)		
名前	10 進数値	16 進値
MQCNO_ACCOUNTING_Q_DISABLED	32768	X'00008000'
MQCNO_NO_CONV_SHARING	65536	X'00010000'
MQCNO_ALL_CONVS_SHARE	262144	X'00040000'
MQCNO_CD_FOR_OUTPUT_ONLY	524288	X'00080000'
MQCNO_USE_CD_SELECTION	1048576	X'00100000'
MQCNO_RECONNECT	16777216	X'01000000'
MQCNO_RECONNECT_AS_DEF	0	X'00000000'
MQCNO_RECONNECT_DISABLED	33554432	X'02000000'
MQCNO_RECONNECT_Q_MGR	67108864	X'04000000'
MQCNO_ACTIVITY_TRACE_ENABLED	134217728	X'08000000'
MQCNO_ACTIVITY_TRACE_DISABLED	268435456	X'10000000'
MQCNO_NONE	0	X'00000000'

MQCO_* (クローズ・オプション)

表 106. 定数の値		
名前	10 進数値	16 進値
MQCO_IMMEDIATE	0	X'00000000'
MQCO_NONE	0	X'00000000'
MQCO_DELETE	1	X'00000001'
MQCO_DELETE_PURGE	2	X'00000002'
MQCO_KEEP_SUB	4	X'00000004'
MQCO_REMOVE_SUB	8	X'00000008'
MQCO_QUIESCE	32	X'00000020'

MQCODL_* (CICS 情報ヘッダー出力データ長)

表 107. 定数の値		
名前	10 進数値	16 進値
MQCODL_AS_INPUT	-1	X'FFFFFFFF'

MQCOMPRESS_* (チャネル圧縮)

表 108. 定数の値		
名前	10 進数値	16 進値
MQCOMPRESS_NOT_AVAILABLE	-1	X'FFFFFFFF'
MQCOMPRESS_NONE	0	X'00000000'
MQCOMPRESS_RLE	1	X'00000001'
MQCOMPRESS_ZLIBFAST	2	X'00000002'
MQCOMPRESS_ZLIBHIGH	4	X'00000004'
MQCOMPRESS_SYSTEM	8	X'00000008'
 MQCOMPRESS_LZ4FAST	16	X'10000000'

表 108. 定数の値 (続き)		
名前	10 進数値	16 進値
 MQCOMPRESS_LZ4HIGH	32	X'20000000'
MQCOMPRESS_ANY	268435455	X'0FFFFFFF'

MQCONNID_* (接続 ID)

表 109. 定数名と値	
名前	値
MQCONNID_NONE	X'00...00' (24 スル)
MQCONNID_NONE_ARRAY	'\0','\0',... (24 スル)

MQCOPY_* (プロパティ・コピー・オプション)

表 110. 定数の値		
名前	10 進数値	16 進値
MQCOPY_NONE	0	X'00000000'
MQCOPY_ALL	1	X'00000001'
MQCOPY_FORWARD	2	X'00000002'
MQCOPY_PUBLISH	4	X'00000004'
MQCOPY_REPLY	8	X'00000008'
MQCOPY_REPORT	16	X'00000010'
MQCOPY_DEFAULT	22	X'00000016'

MQCQT_* (クラスター・キュー・タイプ)

表 111. 定数の値		
名前	10 進数値	16 進値
MQCQT_LOCAL_Q	1	X'00000001'
MQCQT_ALIAS_Q	2	X'00000002'
MQCQT_REMOTE_Q	3	X'00000003'
MQCQT_Q_MGR_ALIAS	4	X'00000004'

MQCRC_* (CICS 情報ヘッダー戻りコード)

表 112. 定数の値		
名前	10 進数値	16 進値
MQCRC_OK	0	X'00000000'
MQCRC_CICS_EXEC_ERROR	1	X'00000001'
MQCRC_MQ_API_ERROR	2	X'00000002'
MQCRC_BRIDGE_ERROR	3	X'00000003'
MQCRC_BRIDGE_ABEND	4	X'00000004'
MQCRC_APPLICATION_ABEND	5	X'00000005'
MQCRC_SECURITY_ERROR	6	X'00000006'
MQCRC_PROGRAM_NOT_AVAILABLE	7	X'00000007'

表 112. 定数の値 (続き)		
名前	10 進数値	16 進値
MQCRC_BRIDGE_TIMEOUT	8	X'00000008'
MQCRC_TRANSID_NOT_AVAILABLE	9	X'00000009'

MQCS_* (MQCBC 定数コンシューマー状態)

表 113. 定数の値		
名前	10 進数値	16 進値
MQCS_NONE	0	X'00000000'
MQCS_SUSPENDED_TEMPORARY	1	X'00000001'
MQCS_SUSPENDED_USER_ACTION	2	X'00000002'
MQCS_SUSPENDED	3	X'00000003'
MQCS_STOPPED	4	X'00000004'

MQCSC_* (CICS 情報ヘッダー開始コード)

表 114. 定数の構造	
名前	構造体
MQCSC_START	"S-"
MQCSC_STARTDATA	"SD-"
MQCSC_TERMINPUT	"TD-"
MQCSC_NONE	"--"
MQCSC_START_ARRAY	'S','-',',','-',',','-',',','-'
MQCSC_STARTDATA_ARRAY	'S','D','-',',','-',',','-',',','-'
MQCSC_TERMINPUT_ARRAY	'T','D','-',',','-',',','-',',','-'
MQCSC_NONE_ARRAY	'-','-',',','-',',','-',',','-'

注: 記号-は、単一のブランク文字を表します。

MQCSP_* (接続セキュリティ・パラメーター構造体および認証 タイプ)

接続セキュリティ・パラメーター構造体

表 115. 定数の構造	
名前	構造体
MQCSP_STRUC_ID	"CSP-"
MQCSP_STRUC_ID_ARRAY	'C','S','P','-',',','-',',','-'

注: 記号-は、単一のブランク文字を表します。

表 116. 定数の値		
名前	10 進数値	16 進値
MQCSP_VERSION_1	1	X'00000001'
MQCSP_VERSION_2	2	X'00000002'
MQCSP_VERSION_3	3	X'00000003'
MQCSP_CURRENT_VERSION	3	X'00000003'

接続セキュリティ・パラメーター認証 タイプ

表 117. 定数の値		
名前	10 進数値	16 進値
MQCSP_AUTH_NONE	0	X'00000000'
MQCSP_AUTH_USER_ID_AND_PWD	1	X'00000001'
MQCSP 認証 ID トークン	2	X'00000002'

MQCSRV_* (コマンド・サーバー・オプション)

表 118. 定数の値		
名前	10 進数値	16 進値
MQCSRV_CONVERT_NO	0	X'00000000'
MQCSRV_CONVERT_YES	1	X'00000001'
MQCSRV_DLQ_NO	0	X'00000000'
MQCSRV_DLQ_YES	1	X'00000001'

MQCT_* (キュー・マネージャー接続タグ)

表 119. 定数名と値	
名前	値
MQCT_NONE	X'00...00' (128 個のヌル値)
MQCT_NONE_ARRAY	'\0','\0',... (128 個のヌル値)

MQCTES_* (CICS 情報ヘッダー・タスク終了状況)

表 120. 定数の値		
名前	10 進数値	16 進値
MQCTES_NOSYNC	0	X'00000000'
MQCTES_COMMIT	256	X'00000100'
MQCTES_BACKOUT	4352	X'00001100'
MQCTES_ENDTASK	65536	X'00010000'

MQCTLO_* (MQCTL オプション構造体およびコンシューマー制御オプション)

MQCTL オプション構造体

表 121. 定数の構造	
名前	構造体
MQCTLO_STRUC_ID	"CTLO"
MQCTLO_STRUC_ID_ARRAY	'C','T','L','O'

注: 記号-は、単一のブランク文字を表します。

表 122. 定数の値		
名前	10 進数値	16 進値
MQCTLO_VERSION_1	1	X'00000001'
MQCTLO_CURRENT_VERSION	1	X'00000001'

MQCTL オプションのコンシューマー制御オプション

表 123. 定数の値		
名前	10 進数値	16 進値
MQCTLO_NONE	0	X'00000000'
MQCTLO_THREAD_AFFINITY	1	X'00000001'
MQCTLO_FAIL_IF_QUIESCING	8192	X'00002000'

MQCUOWC_* (CICS 情報ヘッダー作業単位制御)

表 124. 定数の値		
名前	10 進数値	16 進値
MQCUOWC_ONLY	273	X'00000111'
MQCUOWC_CONTINUE	65536	X'00010000'
MQCUOWC_FIRST	17	X'00000011'
MQCUOWC_MIDDLE	16	X'00000010'
MQCUOWC_LAST	272	X'00000110'
MQCUOWC_COMMIT	256	X'00000100'
MQCUOWC_BACKOUT	4352	X'00001100'

MQCXP_* (チャネル出口パラメーター構造体)

表 125. 定数の構造	
名前	構造体
MQCXP_STRUC_ID	"CXP"
MQCXP_STRUC_ID_ARRAY	'C','X','P',' '

注：記号␣は、単一のブランク文字を表します。

表 126. 定数の値		
名前	10 進数値	16 進値
MQCXP_VERSION_1	1	X'00000001'
MQCXP_VERSION_2	2	X'00000002'
MQCXP_VERSION_3	3	X'00000003'
MQCXP_VERSION_4	4	X'00000004'
MQCXP_VERSION_5	5	X'00000005'
MQCXP_VERSION_6	6	X'00000006'
MQCXP_VERSION_7	7	X'00000007'
MQCXP_VERSION_8	8	X'00000008'
MQCXP_VERSION_9	9	X'00000009'
MQCXP_CURRENT_VERSION	9	X'00000009'

MQDC_* (宛先クラス)

表 127. 定数の値		
名前	10 進数値	16 進値
MQDC_MANAGED	1	X'00000001'

表 127. 定数の値 (続き)		
名前	10 進数値	16 進値
MQDC_PROVIDED	2	X'00000002'

MQDCC_* (変換オプション、およびマスクと係数)

変換オプション

表 128. 定数の値		
名前	10 進数値	16 進値
MQDCC_DEFAULT_CONVERSION	1	X'00000001'
MQDCC_FILL_TARGET_BUFFER	2	X'00000002'
MQDCC_INT_DEFAULT_CONVERSION	4	X'00000004'
MQDCC_SOURCE_ENC_NATIVE	(value differs by platform or version)	(value differs by platform or version)
MQDCC_SOURCE_ENC_NORMAL	16	X'00000010'
MQDCC_SOURCE_ENC_REVERSED	32	X'00000020'
MQDCC_SOURCE_ENC_UNDEFINED	0	X'00000000'
MQDCC_TARGET_ENC_NATIVE	(value differs by platform or version)	(value differs by platform or version)
MQDCC_TARGET_ENC_NORMAL	256	X'00000100'
MQDCC_TARGET_ENC_REVERSED	512	X'00000200'
MQDCC_TARGET_ENC_UNDEFINED	0	X'00000000'
MQDCC_NONE	0	X'00000000'

変換オプションのマスクおよび係数

表 129. 定数の値		
名前	10 進数値	16 進値
MQDCC_SOURCE_ENC_MASK	240	X'000000F0'
MQDCC_TARGET_ENC_MASK	3840	X'00000F00'
MQDCC_SOURCE_ENC_FACTOR	16	X'00000010'
MQDCC_TARGET_ENC_FACTOR	256	X'00000100'

MQDELO_* (パブリッシュ/サブスクライブ削除オプション)

表 130. 定数の値		
名前	10 進数値	16 進値
MQDELO_NONE	0	X'00000000'
MQDELO_LOCAL	4	X'00000004'

MQDH_* (配布ヘッダー構造体)

表 131. 定数の構造	
名前	構造体
MQDH_STRUC_ID	"DHヘッ"

表 131. 定数の構造 (続き)	
名前	構造体
MQDH_STRUC_ID_ARRAY	'D','H',' ',' '

注: 記号-は、単一の空白文字を表します。

表 132. 定数の値		
名前	10 進数値	16 進値
MQDH_VERSION_1	1	X'00000001'
MQDH_CURRENT_VERSION	1	X'00000001'

MQDHF_* (配布ヘッダー・フラグ)

表 133. 定数の値		
名前	10 進数値	16 進値
MQDHF_NEW_MSG_IDS	1	X'00000001'
MQDHF_NONE	0	X'00000000'

MQDISCONNECT_* (コマンド形式の切断タイプ)

表 134. 定数の値		
名前	10 進数値	16 進値
MQDISCONNECT_NORMAL	0	X'00000000'
MQDISCONNECT_IMPLICIT	1	X'00000001'
MQDISCONNECT_Q_MGR	2	X'00000002'

MQDL_* (配布リスト)

表 135. 定数の値		
名前	10 進数値	16 進値
MQDL_SUPPORTED	1	X'00000001'
MQDL_NOT_SUPPORTED	0	X'00000000'

MQDLH_* (送達不能ヘッダー構造体)

表 136. 定数の構造	
名前	構造体
MQDLH_STRUC_ID	"DLH"
MQDLH_STRUC_ID_ARRAY	'D','L','H',' '

注: 記号-は、単一の空白文字を表します。

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

MQDLV_* (持続/非持続メッセージ送達)

表 137. 定数の値		
名前	10 進数値	16 進値
MQDLV_AS_PARENT	0	X'00000000'
MQDLV_ALL	1	X'00000001'
MQDLV_ALL_DUR	2	X'00000002'
MQDLV_ALL_AVAIL	3	X'00000003'

MQDMHO_* (メッセージ・ハンドル削除オプションおよび構造体)

メッセージ・ハンドル削除オプション構造体

表 138. 定数の構造	
名前	構造体
MQDMHO_STRUC_ID	"DMHO"
MQDMHO_STRUC_ID_ARRAY	'D','M','H','O'

注: 記号-は、単一の空白文字を表します。

表 139. 定数の値		
名前	10 進数値	16 進値
MQDMHO_VERSION_1	1	X'00000001'
MQDMHO_CURRENT_VERSION	1	X'00000001'

メッセージ・ハンドル削除オプション

表 140. 定数の値		
名前	10 進数値	16 進値
MQDMHO_NONE	0	X'00000000'

MQDMPO_* (メッセージ・プロパティ削除オプションおよび構造体)

メッセージ・プロパティ削除オプション構造体

表 141. 定数の構造	
名前	構造体
MQDMPO_STRUC_ID	"DMPO"
MQDMPO_STRUC_ID_ARRAY	'D','M','P','O'

注: 記号-は、単一の空白文字を表します。

表 142. 定数の値		
名前	10 進数値	16 進値
MQDMPO_VERSION_1	1	X'00000001'
MQDMPO_CURRENT_VERSION	1	X'00000001'

メッセージ・プロパティ削除オプション

表 143. 定数の値		
名前	10 進数値	16 進値
MQDMPO_DEL_FIRST	0	X'00000000'
MQDMPO_DEL_PROP_UNDER_CURSOR	1	X'00000001'
MQDMPO_NONE	0	X'00000000'

MQDNSWLM_* (DNS WLM)

表 144. 定数の値		
名前	10 進数値	16 進値
MQDNSWLM_NO	0	X'00000000'
MQDNSWLM_YES	1	X'00000001'

MQDT_* (宛先タイプ)

表 145. 定数の値		
名前	10 進数値	16 進値
MQDT_APPL	1	X'00000001'
MQDT_BROKER	2	X'00000002'

MQDXP_* (変換出口パラメーター構造体)

表 146. 定数の構造	
名前	構造体
MQDXP_STRUC_ID	"DXP-"
MQDXP_STRUC_ID_ARRAY	'D', 'X', 'P', '-'

注：記号-は、単一のブランク文字を表します。

表 147. 定数の値		
名前	10 進数値	16 進値
MQDXP_VERSION_1	1	X'00000001'
MQDXP_VERSION_2	2	X'00000002'
MQDXP_CURRENT_VERSION	2	X'00000002'

MQEC_* (シグナル値)

表 148. 定数の値		
名前	10 進数値	16 進値
MQEC_MSG_ARRIVED	2	X'00000002'
MQEC_WAIT_INTERVAL_EXPIRED	3	X'00000003'
MQEC_WAIT_CANCELED	4	X'00000004'
MQEC_Q_MGR QUIESCING	5	X'00000005'
MQEC_CONNECTION QUIESCING	6	X'00000006'

MQEI_* (有効期限)

表 149. 定数の値		
名前	10 進数値	16 進値
MQEI_UNLIMITED	-1	X'FFFFFFFF'

MQENC_* (エンコード)

MQENC_* (エンコード)

表 150. プラットフォームごとの定数の値			
名前	プラットフォーム	10 進数値	16 進値
MQENC_NATIVE	IBM i	273	X'00000111'
	Linux®	546	X'00000222'
	SPARC 上の Linux	273	X'00000111'
	x86 上の Linux	546	X'00000222'
	AIX® and Linux	273	X'00000111'
	Windows	546	X'00000222'
	Windows での Micro Focus COBOL	17	X'00000011'
	z/OS®	785	X'00000311'

MQENC_* (エンコード・マスク)

表 151. 定数の値		
名前	10 進数値	16 進値
MQENC_INTEGER_MASK	15	X'0000000F'
MQENC_DECIMAL_MASK	240	X'000000F0'
MQENC_FLOAT_MASK	3840	X'00000F00'
MQENC_RESERVED_MASK	-4096	X'FFFFFF000'

MQENC_* (2 進整数のエンコード)

表 152. 定数の値		
名前	10 進数値	16 進値
MQENC_INTEGER_UNDEFINED	0	X'00000000'
MQENC_INTEGER_NORMAL	1	X'00000001'
MQENC_INTEGER_REVERSED	2	X'00000002'

MQENC_* (パック 10 進整数のエンコード)

表 153. 定数の値		
名前	10 進数値	16 進値
MQENC_DECIMAL_UNDEFINED	0	X'00000000'
MQENC_DECIMAL_NORMAL	16	X'00000010'
MQENC_DECIMAL_REVERSED	32	X'00000020'

MQENC_* (浮動小数点数のエンコード)

表 154. 定数の値		
名前	10 進数値	16 進値
MQENC_FLOAT_UNDEFINED	0	X'00000000'
MQENC_FLOAT_IEEE_NORMAL	256	X'00000100'
MQENC_FLOAT_IEEE_REVERSED	512	X'00000200'
MQENC_FLOAT_S390	768	X'00000300'
MQENC_FLOAT_TNS	1024	X'00000400'

MQEPH_* (埋め込みコマンド形式のヘッダー構造体およびフラグ)

埋め込みコマンド形式のヘッダー構造体

表 155. 定数の構造	
名前	構造体
MQEPH_STRUC_ID	"EPH"
MQEPH_STRUC_ID_ARRAY	'E','P','H',' '

注: 記号␣は、単一のブランク文字を表します。

表 156. 定数の値		
名前	10 進数値	16 進値
MQEPH_STRUC_LENGTH_FIXED	68	X'00000044'
MQEPH_VERSION_1	1	X'00000001'
MQEPH_CURRENT_VERSION	1	X'00000001'

埋め込みコマンド形式のヘッダー・フラグ

表 157. 定数の値		
名前	10 進数値	16 進値
MQEPH_NONE	0	X'00000000'
MQEPH_CCSID_EMBEDDED	1	X'00000001'

MQET_* (コマンド形式のエスケープ・タイプ)

表 158. 定数の値		
名前	10 進数値	16 進値
MQET_MQSC	1	X'00000001'

MQEVO_* (コマンド形式のイベント発信元)

表 159. 定数の値		
名前	10 進数値	16 進値
MQEVO_OTHER	0	X'00000000'
MQEVO_CONSOLE	1	X'00000001'
MQEVO_INIT	2	X'00000002'
MQEVO_MSG	3	X'00000003'

表 159. 定数の値 (続き)		
名前	10 進数値	16 進値
MQEVO_MQSET	4	X'00000004'
MQEVO_INTERNAL	5	X'00000005'
MQEVO_MQSUB	6	X'00000006'
MQEVO_CTLMSG	7	X'00000007'
MQEVO_REST	8	X'00000008'

MQEVR_* (コマンド形式のイベント記録)

表 160. 定数の値		
名前	10 進数値	16 進値
MQEVR_DISABLED	0	X'00000000'
MQEVR_ENABLED	1	X'00000001'
MQEVR_EXCEPTION	2	X'00000002'
MQEVR_NO_DISPLAY	3	X'00000003'

MQEXPI_* (有効期限スキャン間隔)

表 161. 定数の値		
名前	10 進数値	16 進値
MQEXPI_OFF	0	X'00000000'

MQFB_* (フィードバック値)

表 162. 定数の値		
名前	10 進数値	16 進値
MQFB_NONE	0	X'00000000'
MQFB_SYSTEM_FIRST	1	X'00000001'
MQFB_QUIT	256	X'00000100'
MQFB_EXPIRATION	258	X'00000102'
MQFB_COA	259	X'00000103'
MQFB_COD	260	X'00000104'
MQFB_CHANNEL_COMPLETED	262	X'00000106'
MQFB_CHANNEL_FAIL_RETRY	263	X'00000107'
MQFB_CHANNEL_FAIL	264	X'00000108'
MQFB_APPL_CANNOT_BE_STARTED	265	X'00000109'
MQFB_TM_ERROR	266	X'0000010A'
MQFB_APPL_TYPE_ERROR	267	X'0000010B'
MQFB_STOPPED_BY_MSG_EXIT	268	X'0000010C'
MQFB_ACTIVITY	269	X'0000010D'
MQFB_XMIT_Q_MSG_ERROR	271	X'0000010F'
MQFB_PAN	275	X'00000113'
MQFB_NAN	276	X'00000114'
MQFB_STOPPED_BY_CHAD_EXIT	277	X'00000115'

表 162. 定数の値 (続き)		
名前	10 進数値	16 進値
MQFB_STOPPED_BY_PUBSUB_EXIT	279	X'00000117'
MQFB_NOT_A_REPOSITORY_MSG	280	X'00000118'
MQFB_BIND_OPEN_CLUSRCVR_DEL	281	X'00000119'
MQFB_MAX_ACTIVITIES	282	X'0000011A'
MQFB_NOT_FORWARDED	283	X'0000011B'
MQFB_NOT_DELIVERED	284	X'0000011C'
MQFB_UNSUPPORTED_FORWARDING	285	X'0000011D'
MQFB_UNSUPPORTED_DELIVERY	286	X'0000011E'
MQFB_DATA_LENGTH_ZERO	291	X'00000123'
MQFB_DATA_LENGTH_NEGATIVE	292	X'00000124'
MQFB_DATA_LENGTH_TOO_BIG	293	X'00000125'
MQFB_BUFFER_OVERFLOW	294	X'00000126'
MQFB_LENGTH_OFF_BY_ONE	295	X'00000127'
MQFB_IIH_ERROR	296	X'00000128'
MQFB_NOT_AUTHORIZED_FOR_IMS	298	X'0000012A'
MQFB_IMS_ERROR	300	X'0000012C'
MQFB_IMS_FIRST	301	X'0000012D'
MQFB_IMS_LAST	399	X'0000018F'
MQFB_CICS_INTERNAL_ERROR	401	X'00000191'
MQFB_CICS_NOT_AUTHORIZED	402	X'00000192'
MQFB_CICS_BRIDGE_FAILURE	403	X'00000193'
MQFB_CICS_CORREL_ID_ERROR	404	X'00000194'
MQFB_CICS_CCSDID_ERROR	405	X'00000195'
MQFB_CICS_ENCODING_ERROR	406	X'00000196'
MQFB_CICS_CIH_ERROR	407	X'00000197'
MQFB_CICS_UOW_ERROR	408	X'00000198'
MQFB_CICS_COMMAREA_ERROR	409	X'00000199'
MQFB_CICS_APPL_NOT_STARTED	410	X'0000019A'
MQFB_CICS_APPL_ABENDED	411	X'0000019B'
MQFB_CICS_DLQ_ERROR	412	X'0000019C'
MQFB_CICS_UOW_BACKED_OUT	413	X'0000019D'
MQFB_PUBLICATIONS_ON_REQUEST	501	X'000001F5'
MQFB_SUBSCRIBER_IS_PUBLISHER	502	X'000001F6'
MQFB_MSG_SCOPE_MISMATCH	503	X'000001F7'
MQFB_SELECTOR_MISMATCH	504	X'000001F8'
MQFB_IMS_NACK_1A_REASON_FIRST	600	X'00000258'
MQFB_IMS_NACK_1A_REASON_LAST	855	X'00000357'
MQFB_SYSTEM_LAST	65535	X'0000FFFF'
MQFB_APPL_FIRST	65536	X'00010000'

表 162. 定数の値 (続き)		
名前	10 進数値	16 進値
MQFB_APPL_LAST	999999999	X'3B9AC9FF'

MQFC_* (コマンド形式の強制オプション)

表 163. 定数の値		
名前	10 進数値	16 進値
MQFC_YES	1	X'00000001'
MQFC_NO	0	X'00000000'

MQFMT_* (フォーマット)

表 164. 定数名と値	
名前	値
MQFMT_NONE	"~~~~~"
MQFMT_ADMIN	"MQADMIN~"
MQFMT_CHANNEL_COMPLETED	"MQCHCOM~"
MQFMT_CICS	"MQCICS~~"
MQFMT_COMMAND_1	"MQCMD1~~"
MQFMT_COMMAND_2	"MQCMD2~~"
MQFMT_DEAD_LETTER_HEADER	"MQDEAD~~"
MQFMT_DIST_HEADER	"MQHDIST~"
MQFMT_EMBEDDED_PCF	"MQHEPCF~"
MQFMT_EVENT	"MQEVENT~"
MQFMT_IMS	"MQIMS~~~"
MQFMT_IMS_VAR_STRING	"MQIMSVS~"
MQFMT_MD_EXTENSION	"MQHME~~"
MQFMT_PCF	"MQPCF~~~"
MQFMT_REF_MSG_HEADER	"MQHREF~~"
MQFMT_RF_HEADER	"MQHRF~~~"
MQFMT_RF_HEADER_1	"MQHRF~~~"
MQFMT_RF_HEADER_2	"MQHRF2~~"
MQFMT_STRING	"MQSTR~~~"
MQFMT_TRIGGER	"MQTRIG~~"
MQFMT_WORK_INFO_HEADER	"MQHWIH~~"
MQFMT_XMIT_Q_HEADER	"MQXMIT~~"
MQFMT_NONE_ARRAY	'~','~','~','~','~','~','~','~','~'
MQFMT_ADMIN_ARRAY	'M','Q','A','D','M','I','N','~'
MQFMT_CHANNEL_COMPLETED_ARRAY	'M','Q','C','H','C','O','M','~'
MQFMT_CICS_ARRAY	'M','Q','C','I','C','S','~','~'
MQFMT_COMMAND_1_ARRAY	'M','Q','C','M','D','1','~','~'
MQFMT_COMMAND_2_ARRAY	'M','Q','C','M','D','2','~','~'

表 164. 定数名と値 (続き)	
名前	値
MQFMT_DEAD_LETTER_HEADER_ARRAY	'M','Q','D','E','A','D',' ',' '
MQFMT_DIST_HEADER_ARRAY	'M','Q','H','D','I','S','T',' '
MQFMT_EMBEDDED_PCF_ARRAY	'M','Q','H','E','P','C','F',' '
MQFMT_EVENT_ARRAY	'M','Q','E','V','E','N','T',' '
MQFMT_IMS_ARRAY	'M','Q','I','M','S',' ',' ',' '
MQFMT_IMS_VAR_STRING_ARRAY	'M','Q','I','M','S','V','S',' '
MQFMT_MD_EXTENSION_ARRAY	'M','Q','H','M','D','E',' ',' '
MQFMT_PCF_ARRAY	'M','Q','P','C','F',' ',' ',' '
MQFMT_REF_MSG_HEADER_ARRAY	'M','Q','H','R','E','F',' ',' '
MQFMT_RF_HEADER_ARRAY	'M','Q','H','R','F',' ',' ',' '
MQFMT_RF_HEADER_1_ARRAY	'M','Q','H','R','F',' ',' ',' '
MQFMT_RF_HEADER_2_ARRAY	'M','Q','H','R','F','2',' ',' '
MQFMT_STRING_ARRAY	'M','Q','S','T','R',' ',' ',' '
MQFMT_TRIGGER_ARRAY	'M','Q','T','R','I','G',' ',' '
MQFMT_WORK_INFO_HEADER_ARRAY	'M','Q','H','W','I','H',' ',' '
MQFMT_XMIT_Q_HEADER_ARRAY	'M','Q','X','M','I','T',' ',' '

注: 記号-は、単一のブランク文字を表します。

MQFUN_* (アプリケーション関数型)

表 165. 定数の値		
名前	10 進数値	16 進値
MQFUN_TYPE_UNKNOWN	0	X'00000000'
MQFUN_TYPE_JVM	1	X'00000001'
MQFUN_TYPE_PROGRAM	2	X'00000002'
MQFUN_TYPE_PROCEDURE	3	X'00000003'
MQFUN_TYPE_USERDEF	4	X'00000004'
MQFUN_TYPE_COMMAND	5	X'00000005'

MQGA_* (グループ属性セクター)

表 166. 定数の値		
名前	10 進数値	16 進値
MQGA_FIRST	8001	X'00001F41'
MQGA_LAST	9000	X'00002328'

MQGACF_* (コマンド形式のグループ・パラメーター・タイプ)

表 167. 定数の値		
名前	10 進数値	16 進値
MQGACF_FIRST	8001	X'00001F41'
MQGACF_COMMAND_CONTEXT	8001	X'00001F41'

表 167. 定数の値 (続き)		
名前	10 進数値	16 進値
MQGACF_COMMAND_DATA	8002	X'00001F42'
MQGACF_TRACE_ROUTE	8003	X'00001F43'
MQGACF_OPERATION	8004	X'00001F44'
MQGACF_ACTIVITY	8005	X'00001F45'
MQGACF_EMBEDDED_MQMD	8006	X'00001F46'
MQGACF_MESSAGE	8007	X'00001F47'
MQGACF_MQMD	8008	X'00001F48'
MQGACF_VALUE_NAMING	8009	X'00001F49'
MQGACF_Q_ACCOUNTING_DATA	8010	X'00001F4A'
MQGACF_Q_STATISTICS_DATA	8011	X'00001F4B'
MQGACF_CHL_STATISTICS_DATA	8012	X'00001F4C'
MQGACF_LAST_USED	8012	X'00001F4C'

MQGI_* (グループ ID)

表 168. 定数名と値	
名前	値
MQGI_NONE	X'00...00' (24 バイト)
MQGI_NONE_ARRAY	'\0', '\0', ... (24 バイト)

MQGMO_* (メッセージ取得オプションおよび構造体)

読み取りメッセージ・オプション構造体

表 169. 定数の構造	
名前	構造体
MQGMO_STRUC_ID	"GMO"
MQGMO_STRUC_ID_ARRAY	'G', 'M', 'O', '\0'

注: 記号\0は、単一の空白文字を表します。

表 170. 定数の値		
名前	10 進数値	16 進値
MQGMO_VERSION_1	1	X'00000001'
MQGMO_VERSION_2	2	X'00000002'
MQGMO_VERSION_3	3	X'00000003'
MQGMO_VERSION_4	4	X'00000004'
MQGMO_CURRENT_VERSION	4	X'00000004'

メッセージ読み取りオプション

表 171. 定数の値		
名前	10 進数値	16 進値
MQGMO_WAIT	1	X'00000001'

表 171. 定数の値 (続き)		
名前	10 進数値	16 進値
MQGMO_NO_WAIT	0	X'00000000'
MQGMO_SET_SIGNAL	8	X'00000008'
MQGMO_FAIL_IF QUIESCING	8192	X'00002000'
MQGMO_SYNCPOINT	2	X'00000002'
MQGMO_SYNCPOINT_IF_PERSISTENT	4096	X'00001000'
MQGMO_NO_SYNCPOINT	4	X'00000004'
MQGMO_MARK_SKIP_BACKOUT	128	X'00000080'
MQGMO_BROWSE_FIRST	16	X'00000010'
MQGMO_BROWSE_NEXT	32	X'00000020'
MQGMO_BROWSE_MSG_UNDER_CURSOR	2048	X'00008000'
MQGMO_BROWSE_HANDLE	17825808	X'01100010'
MQGMO_BROWSE_CO_OP	18874384	X'01200010'
MQGMO_MSG_UNDER_CURSOR	256	X'00000100'
MQGMO_LOCK	512	X'00000200'
MQGMO_UNLOCK	1024	X'00000400'
MQGMO_ACCEPT_TRUNCATED_MSG	64	X'00000040'
MQGMO_CONVERT	16384	X'00004000'
MQGMO_LOGICAL_ORDER	32768	X'00008000'
MQGMO_COMPLETE_MSG	65536	X'00010000'
MQGMO_ALL_MSGS_AVAILABLE	131072	X'00020000'
MQGMO_ALL_SEGMENTS_AVAILABLE	262144	X'00040000'
MQGMO_MARK_BROWSE_HANDLE	1048576	X'00100000'
MQGMO_MARK_BROWSE_CO_OP	2097152	X'00200000'
MQGMO_UNMARK_BROWSE_CO_OP	4194304	X'00400000'
MQGMO_UNMARK_BROWSE_HANDLE	8388608	X'00800000'
MQGMO_UNMARKED_BROWSE_MSG	16777216	X'01000000'
MQGMO_PROPERTIES_FORCE_MQRFH2	33554432	X'02000000'
MQGMO_NO_PROPERTIES	67108864	X'04000000'
MQGMO_PROPERTIES_IN_HANDLE	134217728	X'08000000'
MQGMO_PROPERTIES_COMPATIBILITY	268435456	X'10000000'
MQGMO_PROPERTIES_AS_Q_DEF	0	X'00000000'
MQGMO_NONE	0	X'00000000'

MQGS_* (グループ状況)

表 172. 定数名と値	
名前	値
MQGS_NOT_IN_GROUP	'-'
MQGS_MSG_IN_GROUP	'G'
MQGS_LAST_MSG_IN_GROUP	'L'

注: 記号-は、単一のブランク文字を表します。

MQHA_* (ハンドル・セクター)

表 173. 定数の値		
名前	10 進数値	16 進値
MQHA_FIRST	4001	X'00000FA1'
MQHA_BAG_HANDLE	4001	X'00000FA1'
MQHA_LAST_USED	4001	X'00000FA1'
MQHA_LAST	6000	X'00001770'

MQHB_* (バッグ・ハンドル)

表 174. 定数の値		
名前	10 進数値	16 進値
MQHB_UNUSABLE_HBAG	-1	X'FFFFFFFF'
MQHB_NONE	-2	X'FFFFFFFE'

MQHC_* (接続ハンドル)

表 175. 定数の値		
名前	10 進数値	16 進値
MQHC_DEF_HCONN	0	X'00000000'
MQHC_UNUSABLE_HCONN	-1	X'FFFFFFFF'
MQHC_UNASSOCIATED_HCONN	-3	X'FFFFFFFD'

MQHM_* (メッセージ・ハンドル)

表 176. 定数の値		
名前	10 進数値	16 進値
MQHM_UNUSABLE_HMSG	-1	X'FFFFFFFF'
MQHM_NONE	0	X'00000000'

MQHO_* (オブジェクト・ハンドル)

表 177. 定数の値		
名前	10 進数値	16 進値
MQHO_UNUSABLE_HOBJ	-1	X'FFFFFFFF'
MQHO_NONE	0	X'00000000'

MQHSTATE_* (コマンド形式のハンドル状態)

表 178. 定数の値		
名前	10 進数値	16 進値
MQHSTATE_INACTIVE	0	X'00000000'
MQHSTATE_ACTIVE	1	X'00000001'

MQIA_* (整数属性セクター)

表 179. 定数の値		
名前	10 進数値	16 進値
MQIA_ACCOUNTING_CONN_OVERRIDE	136	X'00000088'
MQIA_ACCOUNTING_INTERVAL	135	X'00000087'
MQIA_ACCOUNTING_MQI	133	X'00000085'
MQIA_ACCOUNTING_Q	134	X'00000086'
MQIA_ACTIVE_CHANNELS	100	X'00000064'
MQIA_ACTIVITY_CONN_OVERRIDE	239	X'000000EF'
MQIA_ACTIVITY_RECORDING	138	X'0000008A'
MQIA_ACTIVITY_TRACE	240	X'000000F0'
MQIA_ADOPTNEWMCA_CHECK	102	X'00000066'
MQIA_ADOPTNEWMCA_INTERVAL	104	X'00000068'
MQIA_ADOPTNEWMCA_TYPE	103	X'00000067'
MQIA_ADOPT_CONTEXT	260	X'00000104'
 MQIA_ADVANCED_CAPABILITY	273	X'00000111'
MQIA_AMQP_CAPABILITY	265	X'00000109'
MQIA_APPL_TYPE	1	X'00000001'
MQIA_ARCHIVE	60	X'0000003C'
MQIA_AUTHENTICATION_FAIL_DELAY	259	X'00000103'
MQIA_AUTHENTICATION_METHOD	266	X'0000010A'
MQIA_AUTH_INFO_TYPE	66	X'00000042'
 MQIA_AUTHOREV_SCOPE	277	X'00000115'
MQIA_AUTHORITY_EVENT	47	X'0000002F'
MQIA_AUTO_REORG_INTERVAL	174	X'000000AE'
MQIA_AUTO_REORGANIZATION	173	X'000000AD'
MQIA_BACKOUT_THRESHOLD	22	X'00000016'
MQIA_BASE_TYPE	193	X'000000C1'
MQIA_BATCH_INTERFACE_AUTO	86	X'00000056'
MQIA_BRIDGE_EVENT	74	X'0000004A'
MQIA_CF_LEVEL	70	X'00000046'
MQIA_CF_RECOVER	71	X'00000047'
MQIA_CHANNEL_AUTO_DEF	55	X'00000037'
MQIA_CHANNEL_AUTO_DEF_EVENT	56	X'00000038'
MQIA_CHANNEL_EVENT	73	X'00000049'
MQIA_CHECK_CLIENT_BINDING	258	X'00000102'
MQIA_CHECK_LOCAL_BINDING	257	X'00000101'
MQIA_CHINIT_ADAPTERS	101	X'00000065'
MQIA_CHINIT_CONTROL	119	X'00000077'
MQIA_CHINIT_DISPATCHERS	105	X'00000069'
MQIA_CHINIT_TRACE_AUTO_START	117	X'00000075'

表 179. 定数の値 (続き)		
名前	10 進数値	16 進値
MQIA_CHINIT_TRACE_TABLE_SIZE	118	X'00000076'
MQIA_CLUSTER_OBJECT_STATE	256	X'00000100'
MQIA_CLUSTER_PUB_ROUTE	255	X'000000FF'
MQIA_CLUSTER_Q_TYPE	59	X'0000003B'
MQIA_CLUSTER_WORKLOAD_LENGTH	58	X'0000003A'
MQIA_CLWL_MRU_CHANNELS	97	X'00000061'
MQIA_CLWL_Q_RANK	95	X'0000005F'
MQIA_CLWL_Q_PRIORITY	96	X'00000060'
MQIA_CLWL_USEQ	98	X'00000062'
MQIA_CMD_SERVER_AUTO	87	X'00000057'
MQIA_CMD_SERVER_CONTROL	120	X'00000078'
MQIA_CMD_SERVER_CONVERT_MSG	88	X'00000058'
MQIA_CMD_SERVER_DLQ_MSG	89	X'00000059'
MQIA_CODED_CHAR_SET_ID	2	X'00000002'
MQIA_COMM_EVENT	232	X'000000E8'
MQIA_COMMAND_EVENT	99	X'00000063'
MQIA_COMMAND_LEVEL	31	X'0000001F'
MQIA_CONFIGURATION_EVENT	51	X'00000033'
MQIA_CPI_LEVEL	27	X'0000001B'
MQIA_CURRENT_Q_DEPTH	3	X'00000003'
MQIA_DEF_BIND	61	X'0000003D'
MQIA_DEF_CLUSTER_XMIT_Q_TYPE	250	X'000000FA'
MQIA_DEF_INPUT_OPEN_OPTION	4	X'00000004'
MQIA_DEF_PERSISTENCE	5	X'00000005'
MQIA_DEF_PRIORITY	6	X'00000006'
MQIA_DEF_PUT_RESPONSE_TYPE	184	X'000000B8'
MQIA_DEF_READ_AHEAD	188	X'000000BC'
MQIA_DEFINITION_TYPE	7	X'00000007'
MQIA_DISPLAY_TYPE	262	X'00000106'
MQIA_DIST_LISTS	34	X'00000022'
MQIA_DNS_WLM	106	X'0000006A'
MQIA_DURABLE_SUB	175	X'000000AF'
MQIA_EXPIRY_INTERVAL	39	X'00000027'
MQIA_FIRST	1	X'00000001'
 MQIA_GROUP_UR	221	X'000000DD'
MQIA_HARDEN_GET_BACKOUT	8	X'00000008'
MQIA_HIGH_Q_DEPTH	36	X'00000024'
MQIA_IGQ_PUT_AUTHORITY	65	X'00000041'
MQIA_INDEX_TYPE	57	X'00000039'

表 179. 定数の値 (続き)		
名前	10 進数値	16 進値
MQIA_INHIBIT_EVENT	48	X'00000030'
MQIA_INHIBIT_GET	9	X'00000009'
MQIA_INHIBIT_PUB	181	X'000000B5'
MQIA_INHIBIT_PUT	10	X'0000000A'
MQIA_INHIBIT_SUB	182	X'000000B6'
 MQIA_INTRA_GROUP_QUEUING	64	X'00000040'
MQIA_IP_ADDRESS_VERSION	93	X'0000005D'
MQIA_KEY_REUSE_COUNT	267	X'0000010B'
MQIA_LAST	2000	X'000007D0'
MQIA_LAST_USED	277	X'00000115'
MQIA_LDAP_AUTHORMD	263	X'00000107'
MQIA_LDAP_NESTGRP	264	X'00000108'
MQIA_LDAP_SECURE_COMM	261	X'00000105'
MQIA_LISTENER_PORT_NUMBER	85	X'00000055'
MQIA_LISTENER_TIMER	107	X'0000006B'
MQIA_LOGGER_EVENT	94	X'0000005E'
MQIA_LU62_CHANNELS	108	X'0000006C'
MQIA_LOCAL_EVENT	49	X'00000031'
MQIA_MSG_MARK_BROWSE_INTERVAL	68	X'00000044'
MQIA_MAX_CHANNELS	109	X'0000006D'
MQIA_MAX_CLIENTS	172	X'000000AC'
MQIA_MAX_GLOBAL_LOCKS	83	X'00000053'
MQIA_MAX_HANDLES	11	X'0000000B'
MQIA_MAX_LOCAL_LOCKS	84	X'00000054'
MQIA_MAX_MSG_LENGTH	13	X'0000000D'
MQIA_MAX_OPEN_Q	80	X'00000050'
MQIA_MAX_PRIORITY	14	X'0000000E'
MQIA_MAX_PROPERTIES_LENGTH	192	X'000000C0'
MQIA_MAX_Q_DEPTH	15	X'0000000F'
MQIA_MAX_Q_TRIGGERS	90	X'0000005A'
MQIA_MAX_RECOVERY_TASKS	171	X'000000AB'
MQIA_MAX_UNCOMMITTED_MSGS	33	X'00000021'
MQIA_MCAST_BRIDGE	233	X'000000E9'
MQIA_MONITOR_INTERVAL	81	X'00000051'
MQIA_MONITORING_AUTO_CLUSSDR	124	X'0000007C'
MQIA_MONITORING_CHANNEL	122	X'0000007A'
MQIA_MONITORING_Q	123	X'0000007B'
MQIA_MSG_DELIVERY_SEQUENCE	16	X'00000010'
MQIA_MSG_DEQ_COUNT	38	X'00000026'

表 179. 定数の値 (続き)		
名前	10 進数値	16 進値
MQIA_MSG_ENQ_COUNT	37	X'00000025'
MQIA_NAME_COUNT	19	X'00000013'
MQIA_NAMELIST_TYPE	72	X'00000048'
MQIA_NPM_CLASS	78	X'0000004E'
MQIA_NPM_DELIVERY	196	X'000000C4'
MQIA_OPEN_INPUT_COUNT	17	X'00000011'
MQIA_OPEN_OUTPUT_COUNT	18	X'00000012'
MQIA_OUTBOUND_PORT_MAX	140	X'0000008C'
MQIA_OUTBOUND_PORT_MIN	110	X'0000006E'
MQIA_PAGESET_ID	62	X'0000003E'
MQIA_PERFORMANCE_EVENT	53	X'00000035'
MQIA_PLATFORM	32	X'00000020'
MQIA_PM_DELIVERY	195	X'000000C3'
MQIA_PROPERTY_CONTROL	190	X'000000BE'
MQIA_PROT_POLICY_CAPABILITY	251	X'000000FB'
MQIA_PROXY_SUB	199	X'000000C7'
MQIA_PUB_COUNT	215	X'000000D7'
MQIA_PUB_SCOPE	219	X'000000DB'
MQIA_PUBSUB_CLUSTER	249	X'000000F9'
MQIA_PUBSUB_MAXMSG_RETRY_COUNT	206	X'000000CE'
MQIA_PUBSUB_MODE	187	X'000000BB'
MQIA_PUBSUB_NP_MSG	203	X'000000CB'
MQIA_PUBSUB_NP_RESP	205	X'000000CD'
MQIA_PUBSUB_SYNC_PT	207	X'000000CF'
MQIA_Q_DEPTH_HIGH_EVENT	43	X'0000002B'
MQIA_Q_DEPTH_HIGH_LIMIT	40	X'00000028'
MQIA_Q_DEPTH_LOW_EVENT	44	X'0000002C'
MQIA_Q_DEPTH_LOW_LIMIT	41	X'00000029'
MQIA_Q_DEPTH_MAX_EVENT	42	X'0000002A'
MQIA_Q_SERVICE_INTERVAL	54	X'00000036'
MQIA_Q_SERVICE_INTERVAL_EVENT	46	X'0000002E'
MQIA_Q_TYPE	20	X'00000014'
MQIA_Q_USERS	82	X'00000052'
MQIA_QMGR_CFCONLOS	245	X'000000F5'
MQIA_QMOPT_CONS_COMMS_MSGS	155	X'0000009B'
MQIA_QMOPT_CONS_CRITICAL_MSGS	154	X'0000009A'
MQIA_QMOPT_CONS_ERROR_MSGS	153	X'00000099'
MQIA_QMOPT_CONS_INFO_MSGS	151	X'00000097'
MQIA_QMOPT_CONS_REORG_MSGS	156	X'0000009C'

表 179. 定数の値 (続き)		
名前	10 進数値	16 進値
MQIA_QMOPT_CONS_SYSTEM_MSGS	157	X'0000009D'
MQIA_QMOPT_CONS_WARNING_MSGS	152	X'00000098'
MQIA_QMOPT_CSMT_ON_ERROR	150	X'00000096'
MQIA_QMOPT_INTERNAL_DUMP	170	X'000000AA'
MQIA_QMOPT_LOG_COMMS_MSGS	162	X'000000A2'
MQIA_QMOPT_LOG_CRITICAL_MSGS	161	X'000000A1'
MQIA_QMOPT_LOG_ERROR_MSGS	160	X'000000A0'
MQIA_QMOPT_LOG_INFO_MSGS	158	X'0000009E'
MQIA_QMOPT_LOG_REORG_MSGS	163	X'000000A3'
MQIA_QMOPT_LOG_SYSTEM_MSGS	164	X'000000A4'
MQIA_QMOPT_LOG_WARNING_MSGS	159	X'0000009F'
MQIA_QMOPT_TRACE_COMMS	166	X'000000A6'
MQIA_QMOPT_TRACE_CONVERSION	168	X'000000A8'
MQIA_QMOPT_TRACE_REORG	167	X'000000A7'
MQIA_QMOPT_TRACE_MQI_CALLS	165	X'000000A5'
MQIA_QMOPT_TRACE_SYSTEM	169	X'000000A9'
MQIA_QSG_DISP	63	X'0000003F'
MQIA_READ_AHEAD	189	X'000000BD'
MQIA_RECEIVE_TIMEOUT	111	X'0000006F'
MQIA_RECEIVE_TIMEOUT_MIN	113	X'00000071'
MQIA_RECEIVE_TIMEOUT_TYPE	112	X'00000070'
MQIA_REMOTE_EVENT	50	X'00000032'
MQIA_RETENTION_INTERVAL	21	X'00000015'
MQIA_REVERSE_DNS_LOOKUP	254	X'000000FE'
MQIA_SCOPE	45	X'0000002D'
MQIA_SECURITY_CASE	141	X'0000008D'
MQIA_SERVICE_CONTROL	139	X'0000008B'
MQIA_SERVICE_TYPE	121	X'00000079'
MQIA_SHAREABILITY	23	X'00000017'
MQIA_SHARED_Q_Q_MGR_NAME	77	X'0000004D'
MQIA_SSL_EVENT	75	X'0000004B'
MQIA_SSL_FIPS_REQUIRED	92	X'0000005C'
MQIA_SSL_RESET_COUNT	76	X'0000004C'
MQIA_SSL_TASKS	69	X'00000045'
MQIA_START_STOP_EVENT	52	X'00000034'
MQIA_STATISTICS_CHANNEL	129	X'00000081'
MQIA_STATISTICS_AUTO_CLUSSDR	130	X'00000082'
MQIA_STATISTICS_INTERVAL	131	X'00000083'
MQIA_STATISTICS_MQI	127	X'0000007F'

表 179. 定数の値 (続き)		
名前	10 進数値	16 進値
MQIA_STATISTICS_Q	128	X'00000080'
MQIA_SUB_COUNT	204	X'000000CC'
MQIA_SUB_SCOPE	218	X'000000DA'
MQIA_SYNCPOINT	30	X'0000001E'
MQIA_TCP_CHANNELS	114	X'00000072'
MQIA_TCP_KEEP_ALIVE	115	X'00000073'
MQIA_TCP_STACK_TYPE	116	X'00000074'
MQIA_TIME_SINCE_RESET	35	X'00000023'
MQIA_TOPIC_DEF_PERSISTENCE	185	X'000000B9'
MQIA_TOPIC_NODE_COUNT	253	X'000000FD'
MQIA_TOPIC_TYPE	208	X'000000D0'
MQIA_TRACE_ROUTE_RECORDING	137	X'00000089'
MQIA_TREE_LIFE_TIME	183	X'000000B7'
MQIA_TRIGGER_CONTROL	24	X'00000018'
MQIA_TRIGGER_DEPTH	29	X'0000001D'
MQIA_TRIGGER_INTERVAL	25	X'00000019'
MQIA_TRIGGER_MSG_PRIORITY	26	X'0000001A'
MQIA_TRIGGER_TYPE	28	X'0000001C'
MQIA_TRIGGER_RESTART	91	X'0000005B'
MQIA_USAGE	12	X'0000000C'
MQIA_USE_DEAD_LETTER_Q	234	X'000000EA'
MQIA_USER_LIST	2000	X'000007D0'
MQIA_WILDCARD_OPERATION	216	X'000000D8'
MQIA_XR_CAPABILITY	243	X'000000F3'

MQIACF_* (コマンド形式の整数パラメーター・タイプ)

表 180. 定数の値		
名前	10 進数値	16 進値
MQIACF_FIRST	1001	X'000003E9'
MQIACF_Q_MGR_ATTRS	1001	X'000003E9'
MQIACF_Q_ATTRS	1002	X'000003EA'
MQIACF_PROCESS_ATTRS	1003	X'000003EB'
MQIACF_NAMELIST_ATTRS	1004	X'000003EC'
MQIACF_FORCE	1005	X'000003ED'
MQIACF_REPLACE	1006	X'000003EE'
MQIACF_PURGE	1007	X'000003EF'
MQIACF QUIESCE	1008	X'000003F0'
MQIACF_MODE	1008	X'000003F0'
MQIACF_ALL	1009	X'000003F1'

表 180. 定数の値 (続き)		
名前	10 進数値	16 進値
MQIACF_EVENT_APPL_TYPE	1010	X'000003F2'
MQIACF_EVENT_ORIGIN	1011	X'000003F3'
MQIACF_PARAMETER_ID	1012	X'000003F4'
MQIACF_ERROR_ID	1013	X'000003F5'
MQIACF_ERROR_IDENTIFIER	1013	X'000003F5'
MQIACF_SELECTOR	1014	X'000003F6'
MQIACF_CHANNEL_ATTRS	1015	X'000003F7'
MQIACF_OBJECT_TYPE	1016	X'000003F8'
MQIACF_ESCAPE_TYPE	1017	X'000003F9'
MQIACF_ERROR_OFFSET	1018	X'000003FA'
MQIACF_AUTH_INFO_ATTRS	1019	X'000003FB'
MQIACF_REASON_QUALIFIER	1020	X'000003FC'
MQIACF_COMMAND	1021	X'000003FD'
MQIACF_OPEN_OPTIONS	1022	X'000003FE'
MQIACF_OPEN_TYPE	1023	X'000003FF'
MQIACF_PROCESS_ID	1024	X'00000400'
MQIACF_THREAD_ID	1025	X'00000401'
MQIACF_Q_STATUS_ATTRS	1026	X'00000402'
MQIACF_UNCOMMITTED_MSGS	1027	X'00000403'
MQIACF_HANDLE_STATE	1028	X'00000404'
MQIACF_AUX_ERROR_DATA_INT_1	1070	X'0000042E'
MQIACF_AUX_ERROR_DATA_INT_2	1071	X'0000042F'
MQIACF_CONV_REASON_CODE	1072	X'00000430'
MQIACF_BRIDGE_TYPE	1073	X'00000431'
MQIACF_INQUIRY	1074	X'00000432'
MQIACF_WAIT_INTERVAL	1075	X'00000433'
MQIACF_OPTIONS	1076	X'00000434'
MQIACF_BROKER_OPTIONS	1077	X'00000435'
MQIACF_REFRESH_TYPE	1078	X'00000436'
MQIACF_SEQUENCE_NUMBER	1079	X'00000437'
MQIACF_INTEGER_DATA	1080	X'00000438'
MQIACF_REGISTRATION_OPTIONS	1081	X'00000439'
MQIACF_PUBLICATION_OPTIONS	1082	X'0000043A'
MQIACF_CLUSTER_INFO	1083	X'0000043B'
MQIACF_Q_MGR_DEFINITION_TYPE	1084	X'0000043C'
MQIACF_Q_MGR_TYPE	1085	X'0000043D'
MQIACF_ACTION	1086	X'0000043E'
MQIACF_SUSPEND	1087	X'0000043F'
MQIACF_BROKER_COUNT	1088	X'00000440'

表 180. 定数の値 (続き)		
名前	10 進数値	16 進値
MQIACF_APPL_COUNT	1089	X'00000441'
MQIACF_ANONYMOUS_COUNT	1090	X'00000442'
MQIACF_REG_REG_OPTIONS	1091	X'00000443'
MQIACF_DELETE_OPTIONS	1092	X'00000444'
MQIACF_CLUSTER_Q_MGR_ATTRS	1093	X'00000445'
MQIACF_REFRESH_INTERVAL	1094	X'00000446'
MQIACF_REFRESH_REPOSITORY	1095	X'00000447'
MQIACF_REMOVE_QUEUES	1096	X'00000448'
MQIACF_OPEN_INPUT_TYPE	1098	X'0000044A'
MQIACF_OPEN_OUTPUT	1099	X'0000044B'
MQIACF_OPEN_SET	1100	X'0000044C'
MQIACF_OPEN_INQUIRE	1101	X'0000044D'
MQIACF_OPEN_BROWSE	1102	X'0000044E'
MQIACF_Q_STATUS_TYPE	1103	X'0000044F'
MQIACF_Q_HANDLE	1104	X'00000450'
MQIACF_Q_STATUS	1105	X'00000451'
MQIACF_SECURITY_TYPE	1106	X'00000452'
MQIACF_CONNECTION_ATTRS	1107	X'00000453'
MQIACF_CONNECT_OPTIONS	1108	X'00000454'
MQIACF_CONN_INFO_TYPE	1110	X'00000456'
MQIACF_CONN_INFO_CONN	1111	X'00000457'
MQIACF_CONN_INFO_HANDLE	1112	X'00000458'
MQIACF_CONN_INFO_ALL	1113	X'00000459'
MQIACF_AUTH_PROFILE_ATTRS	1114	X'0000045A'
MQIACF_AUTHORIZATION_LIST	1115	X'0000045B'
MQIACF_AUTH_ADD_AUTHS	1116	X'0000045C'
MQIACF_AUTH_REMOVE_AUTHS	1117	X'0000045D'
MQIACF_ENTITY_TYPE	1118	X'0000045E'
MQIACF_COMMAND_INFO	1120	X'00000460'
MQIACF_CMDSCOPE_Q_MGR_COUNT	1121	X'00000461'
MQIACF_Q_MGR_SYSTEM	1122	X'00000462'
MQIACF_Q_MGR_EVENT	1123	X'00000463'
MQIACF_Q_MGR_DQM	1124	X'00000464'
MQIACF_Q_MGR_CLUSTER	1125	X'00000465'
MQIACF_QSG_DISPS	1126	X'00000466'
MQIACF_UOW_STATE	1128	X'00000468'
MQIACF_SECURITY_ITEM	1129	X'00000469'
MQIACF_CF_STRUC_STATUS	1130	X'0000046A'
MQIACF_UOW_TYPE	1132	X'0000046C'

表 180. 定数の値 (続き)		
名前	10 進数値	16 進値
MQIACF_CF_STRUC_ATTRS	1133	X'0000046D'
MQIACF_EXCLUDE_INTERVAL	1134	X'0000046E'
MQIACF_CF_STATUS_TYPE	1135	X'0000046F'
MQIACF_CF_STATUS_SUMMARY	1136	X'00000470'
MQIACF_CF_STATUS_CONNECT	1137	X'00000471'
MQIACF_CF_STATUS_BACKUP	1138	X'00000472'
MQIACF_CF_STRUC_TYPE	1139	X'00000473'
MQIACF_CF_STRUC_SIZE_MAX	1140	X'00000474'
MQIACF_CF_STRUC_SIZE_USED	1141	X'00000475'
MQIACF_CF_STRUC_ENTRIES_MAX	1142	X'00000476'
MQIACF_CF_STRUC_ENTRIES_USED	1143	X'00000477'
MQIACF_CF_STRUC_BACKUP_SIZE	1144	X'00000478'
MQIACF_MOVE_TYPE	1145	X'00000479'
MQIACF_MOVE_TYPE_MOVE	1146	X'0000047A'
MQIACF_MOVE_TYPE_ADD	1147	X'0000047B'
MQIACF_Q_MGR_NUMBER	1148	X'0000047C'
MQIACF_Q_MGR_STATUS	1149	X'0000047D'
MQIACF_DB2_CONN_STATUS	1150	X'0000047E'
MQIACF_SECURITY_ATTRS	1151	X'0000047F'
MQIACF_SECURITY_TIMEOUT	1152	X'00000480'
MQIACF_SECURITY_INTERVAL	1153	X'00000481'
MQIACF_SECURITY_SWITCH	1154	X'00000482'
MQIACF_SECURITY_SETTING	1155	X'00000483'
MQIACF_STORAGE_CLASS_ATTRS	1156	X'00000484'
MQIACF_USAGE_TYPE	1157	X'00000485'
MQIACF_BUFFER_POOL_ID	1158	X'00000486'
MQIACF_USAGE_TOTAL_PAGES	1159	X'00000487'
MQIACF_USAGE_UNUSED_PAGES	1160	X'00000488'
MQIACF_USAGE_PERSIST_PAGES	1161	X'00000489'
MQIACF_USAGE_NONPERSIST_PAGES	1162	X'0000048A'
MQIACF_USAGE_RESTART_EXTENTS	1163	X'0000048B'
MQIACF_USAGE_EXPAND_COUNT	1164	X'0000048C'
MQIACF_PAGESET_STATUS	1165	X'0000048D'
MQIACF_USAGE_TOTAL_BUFFERS	1166	X'0000048E'
MQIACF_USAGE_DATA_SET_TYPE	1167	X'0000048F'
MQIACF_USAGE_PAGESET	1168	X'00000490'
MQIACF_USAGE_DATA_SET	1169	X'00000491'
MQIACF_USAGE_BUFFER_POOL	1170	X'00000492'
MQIACF_MOVE_COUNT	1171	X'00000493'

表 180. 定数の値 (続き)		
名前	10 進数値	16 進値
MQIACF_EXPIRY_Q_COUNT	1172	X'00000494'
MQIACF_CONFIGURATION_OBJECTS	1173	X'00000495'
MQIACF_CONFIGURATION_EVENTS	1174	X'00000496'
MQIACF_SYSP_TYPE	1175	X'00000497'
MQIACF_SYSP_DEALLOC_INTERVAL	1176	X'00000498'
MQIACF_SYSP_MAX_ARCHIVE	1177	X'00000499'
MQIACF_SYSP_MAX_READ_TAPES	1178	X'0000049A'
MQIACF_SYSP_IN_BUFFER_SIZE	1179	X'0000049B'
MQIACF_SYSP_OUT_BUFFER_SIZE	1180	X'0000049C'
MQIACF_SYSP_OUT_BUFFER_COUNT	1181	X'0000049D'
MQIACF_SYSP_ARCHIVE	1182	X'0000049E'
MQIACF_SYSP_DUAL_ACTIVE	1183	X'0000049F'
MQIACF_SYSP_DUAL_ARCHIVE	1184	X'000004A0'
MQIACF_SYSP_DUAL_BSDS	1185	X'000004A1'
MQIACF_SYSP_MAX_CONNS	1186	X'000004A2'
MQIACF_SYSP_MAX_CONNS_FORE	1187	X'000004A3'
MQIACF_SYSP_MAX_CONNS_BACK	1188	X'000004A4'
MQIACF_SYSP_EXIT_INTERVAL	1189	X'000004A5'
MQIACF_SYSP_EXIT_TASKS	1190	X'000004A6'
MQIACF_SYSP_CHKPOINT_COUNT	1191	X'000004A7'
MQIACF_SYSP_OTMA_INTERVAL	1192	X'000004A8'
MQIACF_SYSP_Q_INDEX_DEFER	1193	X'000004A9'
MQIACF_SYSP_DB2_TASKS	1194	X'000004AA'
MQIACF_SYSP_RESLEVEL_AUDIT	1195	X'000004AB'
MQIACF_SYSP_ROUTING_CODE	1196	X'000004AC'
MQIACF_SYSP_SMF_ACCOUNTING	1197	X'000004AD'
MQIACF_SYSP_SMF_STATS	1198	X'000004AE'
MQIACF_SYSP_SMF_INTERVAL	1199	X'000004AF'
MQIACF_SYSP_TRACE_CLASS	1200	X'000004B0'
MQIACF_SYSP_TRACE_SIZE	1201	X'000004B1'
MQIACF_SYSP_WLM_INTERVAL	1202	X'000004B2'
MQIACF_SYSP_ALLOC_UNIT	1203	X'000004B3'
MQIACF_SYSP_ARCHIVE_RETAIN	1204	X'000004B4'
MQIACF_SYSP_ARCHIVE_WTOR	1205	X'000004B5'
MQIACF_SYSP_BLOCK_SIZE	1206	X'000004B6'
MQIACF_SYSP_CATALOG	1207	X'000004B7'
MQIACF_SYSP_COMPACT	1208	X'000004B8'
MQIACF_SYSP_ALLOC_PRIMARY	1209	X'000004B9'
MQIACF_SYSP_ALLOC_SECONDARY	1210	X'000004BA'

表 180. 定数の値 (続き)		
名前	10 進数値	16 進値
MQIACF_SYSP_PROTECT	1211	X'000004BB'
MQIACF_SYSP_QUIESCE_INTERVAL	1212	X'000004BC'
MQIACF_SYSP_TIMESTAMP	1213	X'000004BD'
MQIACF_SYSP_UNIT_ADDRESS	1214	X'000004BE'
MQIACF_SYSP_UNIT_STATUS	1215	X'000004BF'
MQIACF_SYSP_LOG_COPY	1216	X'000004C0'
MQIACF_SYSP_LOG_USED	1217	X'000004C1'
MQIACF_SYSP_LOG_SUSPEND	1218	X'000004C2'
MQIACF_SYSP_OFFLOAD_STATUS	1219	X'000004C3'
MQIACF_SYSP_TOTAL_LOGS	1220	X'000004C4'
MQIACF_SYSP_FULL_LOGS	1221	X'000004C5'
MQIACF_LISTENER_ATTRS	1222	X'000004C6'
MQIACF_LISTENER_STATUS_ATTRS	1223	X'000004C7'
MQIACF_SERVICE_ATTRS	1224	X'000004C8'
MQIACF_SERVICE_STATUS_ATTRS	1225	X'000004C9'
MQIACF_Q_TIME_INDICATOR	1226	X'000004CA'
MQIACF_OLDEST_MSG_AGE	1227	X'000004CB'
MQIACF_AUTH_OPTIONS	1228	X'000004CC'
MQIACF_Q_MGR_STATUS_ATTRS	1229	X'000004CD'
MQIACF_CONNECTION_COUNT	1230	X'000004CE'
MQIACF_Q_MGR_FACILITY	1231	X'000004CF'
MQIACF_CHINIT_STATUS	1232	X'000004D0'
MQIACF_CMD_SERVER_STATUS	1233	X'000004D1'
MQIACF_ROUTE_DETAIL	1234	X'000004D2'
MQIACF_RECORDED_ACTIVITIES	1235	X'000004D3'
MQIACF_MAX_ACTIVITIES	1236	X'000004D4'
MQIACF_DISCONTINUITY_COUNT	1237	X'000004D5'
MQIACF_ROUTE_ACCUMULATION	1238	X'000004D6'
MQIACF_ROUTE_DELIVERY	1239	X'000004D7'
MQIACF_OPERATION_TYPE	1240	X'000004D8'
MQIACF_BACKOUT_COUNT	1241	X'000004D9'
MQIACF_COMP_CODE	1242	X'000004DA'
MQIACF_ENCODING	1243	X'000004DB'
MQIACF_EXPIRY	1244	X'000004DC'
MQIACF_FEEDBACK	1245	X'000004DD'
MQIACF_MSG_FLAGS	1247	X'000004DF'
MQIACF_MSG_LENGTH	1248	X'000004E0'
MQIACF_MSG_TYPE	1249	X'000004E1'
MQIACF_OFFSET	1250	X'000004E2'

表 180. 定数の値 (続き)		
名前	10 進数値	16 進値
MQIACF_ORIGINAL_LENGTH	1251	X'000004E3'
MQIACF_PERSISTENCE	1252	X'000004E4'
MQIACF_PRIORITY	1253	X'000004E5'
MQIACF_REASON_CODE	1254	X'000004E6'
MQIACF_REPORT	1255	X'000004E7'
MQIACF_VERSION	1256	X'000004E8'
MQIACF_UNRECORDED_ACTIVITIES	1257	X'000004E9'
MQIACF_MONITORING	1258	X'000004EA'
MQIACF_ROUTE_FORWARDING	1259	X'000004EB'
MQIACF_SERVICE_STATUS	1260	X'000004EC'
MQIACF_Q_TYPES	1261	X'000004ED'
MQIACF_USER_ID_SUPPORT	1262	X'000004EE'
MQIACF_INTERFACE_VERSION	1263	X'000004EF'
MQIACF_AUTH_SERVICE_ATTRS	1264	X'000004F0'
MQIACF_USAGE_EXPAND_TYPE	1265	X'000004F1'
MQIACF_SYSP_CLUSTER_CACHE	1266	X'000004F2'
MQIACF_SYSP_DB2_BLOB_TASKS	1267	X'000004F3'
MQIACF_SYSP_WLM_INT_UNITS	1268	X'000004F4'
MQIACF_TOPIC_ATTRS	1269	X'000004F5'
MQIACF_PUBSUB_PROPERTIES	1271	X'000004F7'
MQIACF_DESTINATION_CLASS	1273	X'000004F9'
MQIACF_DURABLE_SUBSCRIPTION	1274	X'000004FA'
MQIACF_SUBSCRIPTION_SCOPE	1275	X'000004FB'
MQIACF_VARIABLE_USER_ID	1277	X'000004FD'
MQIACF_REQUEST_ONLY	1280	X'00000500'
MQIACF_PUB_PRIORITY	1283	X'00000503'
MQIACF_SUB_ATTRS	1287	X'00000507'
MQIACF_WILDCARD_SCHEMA	1288	X'00000508'
MQIACF_SUB_TYPE	1289	X'00000509'
MQIACF_MESSAGE_COUNT	1290	X'0000050A'
MQIACF_Q_MGR_PUBSUB	1291	X'0000050B'
MQIACF_Q_MGR_VERSION	1292	X'0000050C'
MQIACF_SUB_STATUS_ATTRS	1294	X'0000050E'
MQIACF_TOPIC_STATUS	1295	X'0000050F'
MQIACF_TOPIC_SUB	1296	X'00000510'
MQIACF_TOPIC_PUB	1297	X'00000511'
MQIACF_RETAINED_PUBLICATION	1300	X'00000514'
MQIACF_TOPIC_STATUS_ATTRS	1301	X'00000515'
MQIACF_TOPIC_STATUS_TYPE	1302	X'00000516'

表 180. 定数の値 (続き)		
名前	10 進数値	16 進値
MQIACF_SUB_OPTIONS	1303	X'00000517'
MQIACF_PUBLISH_COUNT	1304	X'00000518'
MQIACF_CLEAR_TYPE	1305	X'00000519'
MQIACF_CLEAR_SCOPE	1306	X'0000051A'
MQIACF_SUB_LEVEL	1307	X'0000051B'
MQIACF_ASYNC_STATE	1308	X'0000051C'
MQIACF_SUB_SUMMARY	1309	X'0000051D'
MQIACF_OBSOLETE_MSGS	1310	X'0000051E'
MQIACF_PUBSUB_STATUS	1311	X'0000051F'
MQIACF_PS_STATUS_TYPE	1314	X'00000522'
MQIACF_PUBSUB_STATUS_ATTRS	1318	X'00000526'
MQIACF_SELECTOR_TYPE	1321	X'00000529'
MQIACF_MCAST_REL_INDICATOR	1351	X'00000547'
MQIACF_CHLAUTH_TYPE	1352	X'00000548'
MQXR_DIAGNOSTICS_TYPE	1354	X'0000054A'
MQIACF_CHLAUTH_ATTRS	1355	X'0000054B'
MQIACF_OPERATION_ID	1356	X'0000054C'
MQIACF_API_CALLER_TYPE	1357	X'0000054D'
MQIACF_API_ENVIRONMENT	1358	X'0000054E'
MQIACF_TRACE_DETAIL	1359	X'0000054F'
MQIACF_HOBJ	1360	X'00000550'
MQIACF_CALL_TYPE	1361	X'00000551'
MQIACF_MQCB_OPERATION	1362	X'00000552'
MQIACF_MQCB_TYPE	1363	X'00000553'
MQIACF_MQCB_OPTIONS	1364	X'00000554'
MQIACF_CLOSE_OPTIONS	1365	X'00000555'
MQIACF_CTL_OPERATION	1366	X'00000556'
MQIACF_GET_OPTIONS	1367	X'00000557'
MQIACF_RECS_PRESENT	1368	X'00000558'
MQIACF_KNOWN_DEST_COUNT	1369	X'00000559'
MQIACF_UNKNOWN_DEST_COUNT	1370	X'0000055A'
MQIACF_INVALID_DEST_COUNT	1371	X'0000055B'
MQIACF_RESOLVED_TYPE	1372	X'0000055C'
MQIACF_PUT_OPTIONS	1373	X'0000055D'
MQIACF_BUFFER_LENGTH	1374	X'0000055E'
MQIACF_TRACE_DATA_LENGTH	1375	X'0000055F'
MQIACF_SMDS_EXPANDST	1376	X'00000560'
MQIACF_STRUC_LENGTH	1377	X'00000561'
MQIACF_ITEM_COUNT	1378	X'00000562'

表 180. 定数の値 (続き)		
名前	10 進数値	16 進値
MQIACF_EXPIRY_TIME	1379	X'00000563'
MQIACF_CONNECT_TIME	1380	X'00000564'
MQIACF_DISCONNECT_TIME	1381	X'00000565'
MQIACF_HSUB	1382	X'00000566'
MQIACF_SUBRQ_OPTIONS	1383	X'00000567'
MQIACF_XA_RMID	1384	X'00000568'
MQIACF_XA_FLAGS	1385	X'00000569'
MQIACF_XA_RETCODE	1386	X'0000056A'
MQIACF_XA_HANDLE	1387	X'0000056B'
MQIACF_XA_RETVAL	1388	X'0000056C'
MQIACF_STATUS_TYPE	1389	X'0000056D'
MQIACF_XA_COUNT	1390	X'0000056E'
MQIACF_SELECTOR_COUNT	1391	X'0000056F'
MQIACF_SELECTORS	1392	X'00000570'
MQIACF_INTATTR_COUNT	1393	X'00000571'
MQIACF_INTATTRS	1394	X'00000572'
MQIACF_SUBRQ_ACTION	1395	X'00000573'
MQIACF_NUM_PUBS	1396	X'00000574'
MQIACF_POINTER_SIZE	1397	X'00000575'
MQIACF_REMOVE_AUTHREC	1398	X'00000576'
MQIACF_XR_ATTRS	1399	X'00000577'
MQIACF_APPL_FUNCTION_TYPE	1400	X'00000578'
MQIACF_AMQP_ATTRS	1401	X'00000579'
MQIACF_EXPORT_TYPE	1402	X'0000057A'
MQIACF_EXPORT_ATTRS	1403	X'0000057B'
MQIACF_SYSTEM_OBJECTS	1404	X'0000057C'
MQIACF_CONNECTION_SWAP	1405	X'0000057D'
MQIACF_AMQP_DIAGNOSTICS_TYPE	1406	X'0000057E'
MQIACF_BUFFER_POOL_LOCATION	1408	X'00000580'
MQIACF_LDAP_CONNECTION_STATUS	1409	X'00000581'
MQIACF_SYSP_MAX_ACE_POOL	1410	X'00000582'
MQIACF_PAGECLAS	1411	X'00000583'
MQIACF_AUTH_REC_TYPE	1412	X'00000584'
MQIACF_SYSP_MAX_CONC_OFFLOADS	1413	X'00000585'
MQIACF_SYSP_ZHYPERWRITE	1414	X'00000586'
MQIACF_Q_MGR_STATUS_LOG	1415	X'00000587'
MQIACF_ARCHIVE_LOG_SIZE	1416	X'00000588'
MQIACF_MEDIA_LOG_SIZE	1417	X'00000589'
MQIACF_RESTART_LOG_SIZE	1418	X'0000058A'

表 180. 定数の値 (続き)		
名前	10 進数値	16 進値
MQIACF_REUSABLE_LOG_SIZE	1419	X'0000058B'
MQIACF_LOG_IN_USE	1420	X'0000058C'
MQIACF_LOG_UTILIZATION	1421	X'0000058D'
MQIACF_IGNORE_STATE	1423	X'0000058F'
MQIACF_MOVABLE_APPL_COUNT	1424	X'00000590'
MQIACF_APPL_INFO_ATTRS	1425	X'00000591'
MQIACF_APPL_MOVABLE	1426	X'00000592'
MQIACF_REMOTE_QMGR_ACTIVE	1427	X'00000593'
MQIACF_APPL_INFO_TYPE	1428	X'00000594'
MQIACF_APPL_INFO_APPL	1429	X'00000595'
MQIACF_APPL_INFO_QMGR	1430	X'00000596'
MQIACF_APPL_INFO_LOCAL	1431	X'00000597'
MQIACF_APPL_IMMOVABLE_COUNT	1432	X'00000598'
MQIACF_BALANCED	1433	X'00000599'
MQIACF_BALSTATE	1434	X'0000059A'
MQIACF_APPL_IMMOVABLE_REASON	1435	X'0000059B'
MQIACF_DS_ENCRYPTED	1436	X'0000059C'
MQIACF_CUR_Q_FILE_SIZE	1437	X'0000059D'
MQIACF_CUR_MAX_FILE_SIZE	1438	X'0000059E'
MQIACF_BALANCING_TYPE	1439	X'0000059F'
MQIACF_BALANCING_OPTIONS	1440	X'000005A0'
MQIACF_BALANCING_TIMEOUT	1441	X'000005A1'
MQIACF_SYSP_SMF_STAT_TIME_SECS	1442	X'000005A2'
MQIACF_SYSP_SMF_ACCT_TIME_MINS	1443	X'000005A3'
MQIACF_SYSP_SMF_ACCT_TIME_SECS	1444	X'000005A4'
 MQIACF_イベント重複カウント	1478	X'000005C6'
MQIACF_LAST_USED	1478	X'000005C6'

MQIACH_* (コマンド形式の整数チャネル・タイプ)

表 181. 定数の値		
名前	10 進数値	16 進値
MQIACH_FIRST	1501	X'000005DD'
MQIACH_XMIT_PROTOCOL_TYPE	1501	X'000005DD'
MQIACH_BATCH_SIZE	1502	X'000005DE'
MQIACH_DISC_INTERVAL	1503	X'000005DF'
MQIACH_SHORT_TIMER	1504	X'000005E0'
MQIACH_SHORT_RETRY	1505	X'000005E1'
MQIACH_LONG_TIMER	1506	X'000005E2'
MQIACH_LONG_RETRY	1507	X'000005E3'

表 181. 定数の値 (続き)		
名前	10 進数値	16 進値
MQIACH_PUT_AUTHORITY	1508	X'000005E4'
MQIACH_SEQUENCE_NUMBER_WRAP	1509	X'000005E5'
MQIACH_MAX_MSG_LENGTH	1510	X'000005E6'
MQIACH_CHANNEL_TYPE	1511	X'000005E7'
MQIACH_DATA_COUNT	1512	X'000005E8'
MQIACH_NAME_COUNT	1513	X'000005E9'
MQIACH_MSG_SEQUENCE_NUMBER	1514	X'000005EA'
MQIACH_DATA_CONVERSION	1515	X'000005EB'
MQIACH_IN_DOUBT	1516	X'000005EC'
MQIACH_MCA_TYPE	1517	X'000005ED'
MQIACH_SESSION_COUNT	1518	X'000005EE'
MQIACH_ADAPTER	1519	X'000005EF'
MQIACH_COMMAND_COUNT	1520	X'000005F0'
MQIACH_SOCKET	1521	X'000005F1'
MQIACH_PORT	1522	X'000005F2'
MQIACH_CHANNEL_INSTANCE_TYPE	1523	X'000005F3'
MQIACH_CHANNEL_INSTANCE_ATTRS	1524	X'000005F4'
MQIACH_CHANNEL_ERROR_DATA	1525	X'000005F5'
MQIACH_CHANNEL_TABLE	1526	X'000005F6'
MQIACH_CHANNEL_STATUS	1527	X'000005F7'
MQIACH_INDOUBT_STATUS	1528	X'000005F8'
MQIACH_LAST_SEQ_NUMBER	1529	X'000005F9'
MQIACH_LAST_SEQUENCE_NUMBER	1529	X'000005F9'
MQIACH_CURRENT_MSGS	1531	X'000005FB'
MQIACH_CURRENT_SEQ_NUMBER	1532	X'000005FC'
MQIACH_CURRENT_SEQUENCE_NUMBER	1532	X'000005FC'
MQIACH_SSL_RETURN_CODE	1533	X'000005FD'
MQIACH_MSGS	1534	X'000005FE'
MQIACH_BYTES_SENT	1535	X'000005FF'
MQIACH_BYTES_RCVD	1536	X'00000600'
MQIACH_BYTES_RECEIVED	1536	X'00000600'
MQIACH_BATCHES	1537	X'00000601'
MQIACH_BUFFERS_SENT	1538	X'00000602'
MQIACH_BUFFERS_RCVD	1539	X'00000603'
MQIACH_BUFFERS_RECEIVED	1539	X'00000603'
MQIACH_LONG_RETRIES_LEFT	1540	X'00000604'
MQIACH_SHORT_RETRIES_LEFT	1541	X'00000605'
MQIACH_MCA_STATUS	1542	X'00000606'
MQIACH_STOP_REQUESTED	1543	X'00000607'

表 181. 定数の値 (続き)		
名前	10 進数値	16 進値
MQIACH_MR_COUNT	1544	X'00000608'
MQIACH_MR_INTERVAL	1545	X'00000609'
MQIACH_NPM_SPEED	1562	X'0000061A'
MQIACH_HB_INTERVAL	1563	X'0000061B'
MQIACH_BATCH_INTERVAL	1564	X'0000061C'
MQIACH_NETWORK_PRIORITY	1565	X'0000061D'
MQIACH_KEEP_ALIVE_INTERVAL	1566	X'0000061E'
MQIACH_BATCH_HB	1567	X'0000061F'
MQIACH_SSL_CLIENT_AUTH	1568	X'00000620'
MQIACH_ALLOC_RETRY	1570	X'00000622'
MQIACH_ALLOC_FAST_TIMER	1571	X'00000623'
MQIACH_ALLOC_SLOW_TIMER	1572	X'00000624'
MQIACH_DISC_RETRY	1573	X'00000625'
MQIACH_PORT_NUMBER	1574	X'00000626'
MQIACH_HDR_COMPRESSION	1575	X'00000627'
MQIACH_MSG_COMPRESSION	1576	X'00000628'
MQIACH_CLWL_CHANNEL_RANK	1577	X'00000629'
MQIACH_CLWL_CHANNEL_PRIORITY	1578	X'0000062A'
MQIACH_CLWL_CHANNEL_WEIGHT	1579	X'0000062B'
MQIACH_CHANNEL_DISP	1580	X'0000062C'
MQIACH_INBOUND_DISP	1581	X'0000062D'
MQIACH_CHANNEL_TYPES	1582	X'0000062E'
MQIACH_ADAPS_STARTED	1583	X'0000062F'
MQIACH_ADAPS_MAX	1584	X'00000630'
MQIACH_DISPS_STARTED	1585	X'00000631'
MQIACH_DISPS_MAX	1586	X'00000632'
MQIACH_SSLTASKS_STARTED	1587	X'00000633'
MQIACH_SSLTASKS_MAX	1588	X'00000634'
MQIACH_CURRENT_CHL	1589	X'00000635'
MQIACH_CURRENT_CHL_MAX	1590	X'00000636'
MQIACH_CURRENT_CHL_TCP	1591	X'00000637'
MQIACH_CURRENT_CHL_LU62	1592	X'00000638'
MQIACH_ACTIVE_CHL	1593	X'00000639'
MQIACH_ACTIVE_CHL_MAX	1594	X'0000063A'
MQIACH_ACTIVE_CHL_PAUSED	1595	X'0000063B'
MQIACH_ACTIVE_CHL_STARTED	1596	X'0000063C'
MQIACH_ACTIVE_CHL_STOPPED	1597	X'0000063D'
MQIACH_ACTIVE_CHL_RETRY	1598	X'0000063E'
MQIACH_LISTENER_STATUS	1599	X'0000063F'

表 181. 定数の値 (続き)		
名前	10 進数値	16 進値
MQIACH_SHARED_CHL_RESTART	1600	X'00000640'
MQIACH_LISTENER_CONTROL	1601	X'00000641'
MQIACH_BACKLOG	1602	X'00000642'
MQIACH_XMITQ_TIME_INDICATOR	1604	X'00000644'
MQIACH_NETWORK_TIME_INDICATOR	1605	X'00000645'
MQIACH_EXIT_TIME_INDICATOR	1606	X'00000646'
MQIACH_BATCH_SIZE_INDICATOR	1607	X'00000647'
MQIACH_XMITQ_MSGS_AVAILABLE	1608	X'00000648'
MQIACH_CHANNEL_SUBSTATE	1609	X'00000649'
MQIACH_SSL_KEY_RESETS	1610	X'0000064A'
MQIACH_COMPRESSION_RATE	1611	X'0000064B'
MQIACH_COMPRESSION_TIME	1612	X'0000064C'
MQIACH_MAX_XMIT_SIZE	1613	X'0000064D'
MQIACH_DEF_CHANNEL_DISP	1614	X'0000064E'
MQIACH_SHARING_CONVERSATIONS	1615	X'0000064F'
MQIACH_MAX_SHARING_CONVS	1616	X'00000650'
MQIACH_CURRENT_SHARING_CONVS	1617	X'00000651'
MQIACH_MAX_INSTANCES	1618	X'00000652'
MQIACH_MAX_INSTS_PER_CLIENT	1619	X'00000653'
MQIACH_CLIENT_CHANNEL_WEIGHT	1620	X'00000654'
MQIACH_CONNECTION_AFFINITY	1621	X'00000655'
MQIACH_AUTH_INFO_TYPES	1622	X'00000656'
MQIACH_RESET_REQUESTED	1623	X'00000657'
MQIACH_BATCH_DATA_LIMIT	1624	X'00000658'
MQIACH_MSG_HISTORY	1625	X'00000659'
MQIACH_MULTICAST_PROPERTIES	1626	X'0000065A'
MQIACH_NEW_SUBSCRIBER_HISTORY	1627	X'0000065B'
MQIACH_MC_HB_INTERVAL	1628	X'0000065C'
MQIACH_USE_CLIENT_ID	1629	X'0000065D'
MQIACH_MQTT_KEEP_ALIVE	1630	X'0000065E'
MQIACH_IN_DOUBT_IN	1631	X'0000065F'
MQIACH_IN_DOUBT_OUT	1632	X'00000660'
MQIACH_MSGS_SENT<	1633	X'00000661'
MQIACH_MSGS_RECEIVED	1634	X'00000662'
MQIACH_MSGS_RCVD	1634	X'00000662'
MQIACH_PENDING_OUT	1635	X'00000663'
MQIACH_AVAILABLE_CIPHERSPECS	1636	X'00000664'
MQIACH_MATCH	1637	X'00000665'
MQIACH_USER_SOURCE	1638	X'00000666'

表 181. 定数の値 (続き)		
名前	10 進数値	16 進値
MQIACH_WARNING	1639	X'00000667'
MQIACH_DEF_RECONNECT	1640	X'00000668'
MQIACH_CHANNEL_SUMMARY_ATTRS	1642	X'0000066A'
MQIACH_PROTOCOL	1643	X'0000066B'
MQIACH_AMQPKEEPALIVE	1644	X'0000066C'
MQIACH_SECURITY_PROTOCOL	1645	X'0000066D'
 MQIACH_SPL_PROTECTION	1646	X'0000066E'
MQIACH_LAST_USED	1646	X'0000066E'

MQIAMO_* (コマンド形式の整数モニター・パラメーター・タイプ)

表 182. 定数の値		
名前	10 進数値	16 進値
MQIAMO_FIRST	701	X'000002BD'
MQIAMO_AVG_BATCH_SIZE	702	X'000002BE'
MQIAMO_AVG_Q_TIME	703	X'000002BF'
MQIAMO_BACKOUTS	704	X'000002C0'
MQIAMO_BROWSES	705	X'000002C1'
MQIAMO_BROWSE_MAX_BYTES	706	X'000002C2'
MQIAMO_BROWSE_MIN_BYTES	707	X'000002C3'
MQIAMO_BROWSES_FAILED	708	X'000002C4'
MQIAMO_CLOSSES	709	X'000002C5'
MQIAMO_COMMITS	710	X'000002C6'
MQIAMO_COMMITS_FAILED	711	X'000002C7'
MQIAMO_CONNS	712	X'000002C8'
MQIAMO_CONNS_MAX	713	X'000002C9'
MQIAMO_DISCS	714	X'000002CA'
MQIAMO_DISCS_IMPLICIT	715	X'000002CB'
MQIAMO_DISC_TYPE	716	X'000002CC'
MQIAMO_EXIT_TIME_AVG	717	X'000002CD'
MQIAMO_EXIT_TIME_MAX	718	X'000002CE'
MQIAMO_EXIT_TIME_MIN	719	X'000002CF'
MQIAMO_FULL_BATCHES	720	X'000002D0'
MQIAMO_GENERATED_MSGS	721	X'000002D1'
MQIAMO_GETS	722	X'000002D2'
MQIAMO_GET_MAX_BYTES	723	X'000002D3'
MQIAMO_GET_MIN_BYTES	724	X'000002D4'
MQIAMO_GETS_FAILED	725	X'000002D5'
MQIAMO_INCOMPLETE_BATCHES	726	X'000002D6'
MQIAMO_INQS	727	X'000002D7'

表 182. 定数の値 (続き)		
名前	10 進数値	16 進値
MQIAMO_MSGS	728	X'000002D8'
MQIAMO_NET_TIME_AVG	729	X'000002D9'
MQIAMO_NET_TIME_MAX	730	X'000002DA'
MQIAMO_NET_TIME_MIN	731	X'000002DB'
MQIAMO_OBJECT_COUNT	732	X'000002DC'
MQIAMO_OPENS	733	X'000002DD'
MQIAMO_PUT1S	734	X'000002DE'
MQIAMO_PUTS	735	X'000002DF'
MQIAMO_PUT_MAX_BYTES	736	X'000002E0'
MQIAMO_PUT_MIN_BYTES	737	X'000002E1'
MQIAMO_PUT_RETRIES	738	X'000002E2'
MQIAMO_Q_MAX_DEPTH	739	X'000002E3'
MQIAMO_Q_MIN_DEPTH	740	X'000002E4'
MQIAMO_Q_TIME_AVG	741	X'000002E5'
MQIAMO_Q_TIME_MAX	742	X'000002E6'
MQIAMO_Q_TIME_MIN	743	X'000002E7'
MQIAMO_SETS	744	X'000002E8'
MQIAMO_CONNS_FAILED	749	X'000002ED'
MQIAMO_OPENS_FAILED	751	X'000002EF'
MQIAMO_INQS_FAILED	752	X'000002F0'
MQIAMO_SETS_FAILED	753	X'000002F1'
MQIAMO_PUTS_FAILED	754	X'000002F2'
MQIAMO_PUT1S_FAILED	755	X'000002F3'
MQIAMO_CLOSSES_FAILED	757	X'000002F5'
MQIAMO_MSGS_EXPIRED	758	X'000002F6'
MQIAMO_MSGS_NOT_QUEUED	759	X'000002F7'
MQIAMO_MSGS_PURGED	760	X'000002F8'
MQIAMO_SUBS_DUR	764	X'000002FC'
MQIAMO_SUBS_NDUR	765	X'000002FD'
MQIAMO_SUBS_FAILED	766	X'000002FE'
MQIAMO_SUBRQS	767	X'000002FF'
MQIAMO_SUBRQS_FAILED	768	X'00000300'
MQIAMO_CBS	769	X'00000301'
MQIAMO_CBS_FAILED	770	X'00000302'
MQIAMO_CTLs	771	X'00000303'
MQIAMO_CTLs_FAILED	772	X'00000304'
MQIAMO_STATS	773	X'00000305'
MQIAMO_STATS_FAILED	774	X'00000306'
MQIAMO_SUB_DUR_HIGHWATER	775	X'00000307'

表 182. 定数の値 (続き)		
名前	10 進数値	16 進値
MQIAMO_SUB_DUR_LOWWATER	776	X'00000308'
MQIAMO_SUB_NDUR_HIGHWATER	777	X'00000309'
MQIAMO_SUB_NDUR_LOWWATER	778	X'0000030A'
MQIAMO_TOPIC_PUTS	779	X'0000030B'
MQIAMO_TOPIC_PUTS_FAILED	780	X'0000030C'
MQIAMO_TOPIC_PUT1S	781	X'0000030D'
MQIAMO_TOPIC_PUT1S_FAILED	782	X'0000030E'
MQIAMO_PUBLISH_MSG_COUNT	784	X'00000310'
MQIAMO_UNSUBS_DUR	786	X'00000312'
MQIAMO_UNSUBS_NDUR	787	X'00000313'
MQIAMO_UNSUBS_FAILED	788	X'00000314'
MQIAMO_INTERVAL	789	X'00000315'
MQIAMO_MSGS_SENT	790	X'00000316'
MQIAMO_BYTES_SENT	791	X'00000317'
MQIAMO_REPAIR_BYTES	792	X'00000318'
MQIAMO_FEEDBACK_MODE	793	X'00000319'
MQIAMO_RELIABILITY_TYPE	794	X'0000031A'
MQIAMO_LATE_JOIN_MARK	795	X'0000031B'
MQIAMO_NACKS_RCVD	796	X'0000031C'
MQIAMO_REPAIR_PKTS	797	X'0000031D'
MQIAMO_HISTORY_PKTS	798	X'0000031E'
MQIAMO_PENDING_PKTS	799	X'0000031F'
MQIAMO_PKT_RATE	800	X'00000320'
MQIAMO_MCAST_XMIT_RATE	801	X'00000321'
MQIAMO_MCAST_BATCH_TIME	802	X'00000322'
MQIAMO_MCAST_HEARTBEAT	803	X'00000323'
MQIAMO_DEST_DATA_PORT	804	X'00000324'
MQIAMO_DEST_REPAIR_PORT	805	X'00000325'
MQIAMO_ACKS_RCVD	806	X'00000326'
MQIAMO_ACTIVE_ACKERS	807	X'00000327'
MQIAMO_PKTS_SENT	808	X'00000328'
MQIAMO_TOTAL_REPAIR_PKTS	809	X'00000329'
MQIAMO_TOTAL_PKTS_SENT	810	X'0000032A'
MQIAMO_TOTAL_MSGS_SENT	811	X'0000032B'
MQIAMO_TOTAL_BYTES_SENT	812	X'0000032C'
MQIAMO_NUM_STREAMS	813	X'0000032D'
MQIAMO_ACK_FEEDBACK	814	X'0000032E'
MQIAMO_NACK_FEEDBACK	815	X'0000032F'
MQIAMO_PKTS_LOST	816	X'00000330'

表 182. 定数の値 (続き)		
名前	10 進数値	16 進値
MQIAMO_MSGS_RCVD	817	X'00000331'
MQIAMO_MSG_BYTES_RCVD	818	X'00000332'
MQIAMO_MSGS_DELIVERED	819	X'00000333'
MQIAMO_PKTS_PROCESSED	820	X'00000334'
MQIAMO_PKTS_DLVD	821	X'00000335'
MQIAMO_PKTS_DROPPED	822	X'00000336'
MQIAMO_PKTS_DUPLICATED	823	X'00000337'
MQIAMO_NACKS_CREATED	824	X'00000338'
MQIAMO_NACK_PKTS_SENT	825	X'00000339'
MQIAMO_REPAIR_PKTS_RQSTD	826	X'0000033A'
MQIAMO_REPAIR_PKTS_RCVD	827	X'0000033B'
MQIAMO_PKTS_REPAIRED	828	X'0000033C'
MQIAMO_TOTAL_MSGS_RCVD	829	X'0000033D'
MQIAMO_TOTAL_MSGS_BYTES_RCVD	830	X'0000033E'
MQIAMO_TOTAL_REPAIR_PKTS_RCVD	831	X'0000033F'
MQIAMO_TOTAL_REPAIR_PKTS_RQSTD	832	X'00000340'
MQIAMO_TOTAL_MSGS_PROCESSED	833	X'00000341'
MQIAMO_TOTAL_MSGS_SELECTED	834	X'00000342'
MQIAMO_TOTAL_MSGS_EXPIRED	835	X'00000343'
MQIAMO_TOTAL_MSGS_DELIVERED	836	X'00000344'
MQIAMO_TOTAL_MSGS_RETURNED	837	X'00000345'
MQIAMO_LAST_USED	837	X'00000345'

MQIAMO64_* (コマンド形式の 64 ビット整数モニター・パラメーター・タイプ)

表 183. 定数の値		
名前	10 進数値	16 進値
MQIAMO64_AVG_Q_TIME	703	X'000002BF'
MQIAMO64_Q_TIME_AVG	741	X'000002E5'
MQIAMO64_Q_TIME_MAX	742	X'000002E6'
MQIAMO64_Q_TIME_MIN	743	X'000002E7'
MQIAMO64_BROWSE_BYTES	745	X'000002E9'
MQIAMO64_BYTES	746	X'000002EA'
MQIAMO64_GET_BYTES	747	X'000002EB'
MQIAMO64_PUT_BYTES	748	X'000002EC'
MQIAMO64_TOPIC_PUT_BYTES	783	X'0000030F'
MQIAMO64_PUBLISH_MSG_BYTES	785	X'00000311'

MQIASY_* (整数システム・セクター)

表 184. 定数の値		
名前	10 進数値	16 進値
MQIASY_FIRST	-1	X'FFFFFFFF'
MQIASY_CODED_CHAR_SET_ID	-1	X'FFFFFFFF'
MQIASY_TYPE	-2	X'FFFFFFFE'
MQIASY_COMMAND	-3	X'FFFFFFFD'
MQIASY_MSG_SEQ_NUMBER	-4	X'FFFFFFFC'
MQIASY_CONTROL	-5	X'FFFFFFFB'
MQIASY_COMP_CODE	-6	X'FFFFFFFA'
MQIASY_REASON	-7	X'FFFFFFF9'
MQIASY_BAG_OPTIONS	-8	X'FFFFFFF8'
MQIASY_VERSION	-9	X'FFFFFFF7'
MQIASY_LAST_USED	-9	X'FFFFFFF7'
MQIASY_LAST	-2000	X'FFFFF830'

MQIAUT_* (IMS 情報ヘッダー認証子)

表 185. 定数名と値	
名前	値
MQIAUT_NONE	"~~~~~"
MQIAUT_NONE_ARRAY	' ',' ',' ',' ',' ',' ',' ',' '

注：記号~は、単一のブランク文字を表します。

MQIAV_* (整数属性値)

表 186. 定数の値		
名前	10 進数値	16 進値
MQIAV_NOT_APPLICABLE	-1	X'FFFFFFFF'
MQIAV_UNDEFINED	-2	X'FFFFFFFE'

MQICM_* (IMS 情報ヘッダー・コミット・モード)

表 187. 定数名と値	
名前	値
MQICM_COMMIT_THEN_SEND	'0'
MQICM_SEND_THEN_COMMIT	'1'

MQIDO_* (コマンド形式の未確定オプション)

表 188. 定数の値		
名前	10 進数値	16 進値
MQIDO_COMMIT	1	X'00000001'
MQIDO_BACKOUT	2	X'00000002'

MQIEP_* (インターフェース・エントリー・ポイント)

接続セキュリティ・パラメーター構造体

表 189. 定数の構造	
名前	構造体
MQIEP_STRUC_ID	"IEP"
MQIEP_STRUC_ID_ARRAY	'I','E','P',' '

注: 記号-は、単一の空白文字を表します。

表 190. 定数の値		
名前	10 進数値	16 進値
MQIEP_VERSION_1	1	X'00000001'
MQDXP_CURRENT_VERSION	1	X'00000001'

MQIGQ_* (グループ内キューイング)

表 191. 定数の値		
名前	10 進数値	16 進値
MQIGQ_DISABLED	0	X'00000000'
MQIGQ_ENABLED	1	X'00000001'

MQIGQPA_* (グループ内キューイング書き込み権限)

表 192. 定数の値		
名前	10 進数値	16 進値
MQIGQPA_DEFAULT	1	X'00000001'
MQIGQPA_CONTEXT	2	X'00000002'
MQIGQPA_ONLY_IGQ	3	X'00000003'
MQIGQPA_ALTERNATE_OR_IGQ	4	X'00000004'

MQIIH_* (IMS 情報ヘッダー構造体およびフラグ)

IMS 情報ヘッダー構造体

表 193. 定数の構造	
名前	構造体
MQIIH_STRUC_ID	"IIH"
MQIIH_STRUC_ID_ARRAY	'I','I','H',' '

注: 記号-は、単一の空白文字を表します。

表 194. 定数の値		
名前	10 進数値	16 進値
MQIIH_VERSION_1	1	X'00000001'
MQIIH_CURRENT_VERSION	1	X'00000001'
MQIIH_LENGTH_1	84	X'00000054'

IMS 情報ヘッダー・フラグ

表 195. 定数の値		
名前	10 進数値	16 進値
MQIIH_NONE	0	X'00000000'
MQIIH_PASS_EXPIRATION	1	X'00000001'
MQIIH_UNLIMITED_EXPIRATION	0	X'00000000'
MQIIH_REPLY_FORMAT_NONE	8	X'00000008'
MQIIH_IGNORE_PURG	16	X'00000010'
MQIIH_CM0_REQUEST_RESPONSE	32	X'00000020'

MQIMPO_* (メッセージ・プロパティー照会オプションおよび構造体)

メッセージ・プロパティー照会オプション構造体

表 196. 定数の構造	
名前	構造体
MQIMPO_STRUC_ID	"IMPO"
MQIMPO_STRUC_ID_ARRAY	'I','M','P','O'

注：記号-は、単一のブランク文字を表します。

表 197. 定数の値		
名前	10 進数値	16 進値
MQIMPO_VERSION_1	1	X'00000001'
MQIMPO_CURRENT_VERSION	1	X'00000001'

メッセージ・プロパティー照会オプション

表 198. 定数の値		
名前	10 進数値	16 進値
MQIMPO_CONVERT_TYPE	2	X'00000002'
MQIMPO_QUERY_LENGTH	4	X'00000004'
MQIMPO_INQ_FIRST	0	X'00000000'
MQIMPO_INQ_NEXT	8	X'00000008'
MQIMPO_INQ_PROP_UNDER_CURSOR	16	X'00000010'
MQIMPO_CONVERT_VALUE	32	X'00000020'
MQIMPO_NONE	0	X'00000000'

MQINBD_* (コマンド形式のインバウンド後処理)

表 199. 定数の値		
名前	10 進数値	16 進値
MQINBD_Q_MGR	0	X'00000000'
MQINBD_GROUP	3	X'00000003'

MQIND_* (特殊索引値)

表 200. 定数の値		
名前	10 進数値	16 進値
MQIND_NONE	-1	X'FFFFFFFF'
MQIND_ALL	-2	X'FFFFFFFE'

MQIPADDR_* (IP アドレス・バージョン)

表 201. 定数の値		
名前	10 進数値	16 進値
MQIPADDR_IPV4	0	X'00000000'
MQIPADDR_IPV6	1	X'00000001'

MQISS_* (IMS 情報ヘッダー・セキュリティ有効範囲)

表 202. 定数名と値	
名前	値
MQISS_CHECK	'C'
MQISS_FULL	'F'

MQIT_* (索引タイプ)

表 203. 定数の値		
名前	10 進数値	16 進値
MQIT_NONE	0	X'00000000'
MQIT_MSG_ID	1	X'00000001'
MQIT_CORREL_ID	2	X'00000002'
MQIT_MSG_TOKEN	4	X'00000004'
MQIT_GROUP_ID	5	X'00000005'

MQITEM_* (mqInquireItemInfo の項目タイプ)

表 204. 定数の値		
名前	10 進数値	16 進値
MQITEM_INTEGER	1	X'00000001'
MQITEM_STRING	2	X'00000002'
MQITEM_BAG	3	X'00000003'
MQITEM_BYTE_STRING	4	X'00000004'
MQITEM_INTEGER_FILTER	5	X'00000005'
MQITEM_STRING_FILTER	6	X'00000006'
MQITEM_INTEGER64	7	X'00000007'
MQITEM_BYTE_STRING_FILTER	8	X'00000008'

MQITII_* (IMS 情報ヘッダー・トランザクション・インスタンス ID)

表 205. 定数名と値	
名前	値
MQITII_NONE	X'00...00' (16 個のヌル)
MQITII_NONE_ARRAY	'\0','\0',... (16 個のヌル)

MQITS_* (IMS 情報ヘッダー・トランザクション状態)

表 206. 定数名と値	
名前	値
MQITS_IN_CONVERSATION	'C'
MQITS_NOT_IN_CONVERSATION	'-'
MQITS_ARCHITECTED	'A'

注: 記号-は、単一のブランク文字を表します。

MQKAI_* (KeepAlive 間隔)

表 207. 定数の値		
名前	10 進数値	16 進値
MQKAI_AUTO	-1	X'FFFFFFFF'

MQMASTER_* (マスター管理)

表 208. 定数の値		
名前	10 進数値	16 進値
MQMASTER_NO	0	X'00000000'
MQMASTER_YES	1	X'00000001'

MQMCAS_* (コマンド形式のメッセージ・チャネル・エージェント状況)

表 209. 定数の値		
名前	10 進数値	16 進値
MQMCAS_STOPPED	0	X'00000000'
MQMCAS_RUNNING	3	X'00000003'

MQMCAT_* (MCA タイプ)

表 210. 定数の値		
名前	10 進数値	16 進値
MQMCAT_PROCESS	1	X'00000001'
MQMCAT_THREAD	2	X'00000002'

MQMCD_* (パブリッシュ/サブスクライブ・オプション・タグの情報)

パブリッシュ/サブスクライブ・オプション・タグのメッセージ内容記述子 (mcd) タグ

表 211. 定数の値		
名前	10 進数値	16 進値
MQMCD_FOLDER_VERSION	1	X'00000001'

パブリッシュ/サブスクライブ・オプション・タグのタグ名

表 212. 定数名と値	
名前	値
MQMCD_MSG_DOMAIN	"Msd"
MQMCD_MSG_SET	"Set"
MQMCD_MSG_TYPE	"Type"
MQMCD_MSG_FORMAT	"Fmt"

パブリッシュ/サブスクライブ・オプション・タグの XML タグ名

表 213. 定数名と値	
名前	値
MQMCD_MSG_DOMAIN_B	"<Msd>"
MQMCD_MSG_DOMAIN_E	"</Msd>"
MQMCD_MSG_SET_B	"<Set>"
MQMCD_MSG_SET_E	"</Set>"
MQMCD_MSG_TYPE_B	"<Type>"
MQMCD_MSG_TYPE_E	"</Type>"
MQMCD_MSG_FORMAT_B	"<Fmt>"
MQMCD_MSG_FORMAT_E	"</Fmt>"

パブリッシュ/サブスクライブ・オプション・タグのタグ値

表 214. 定数名と値	
名前	値
MQMCD_DOMAIN_NONE	"none"
MQMCD_DOMAIN_NEON	"neon"
MQMCD_DOMAIN_MRM	"mrm"
MQMCD_DOMAIN_JMS_NONE	"jms_none"
MQMCD_DOMAIN_JMS_TEXT	"jms_text"
MQMCD_DOMAIN_JMS_OBJECT	"jms_object"
MQMCD_DOMAIN_JMS_MAP	"jms_map"
MQMCD_DOMAIN_JMS_STREAM	"jms_stream"
MQMCD_DOMAIN_JMS_BYTES	"jms_bytes"

MQMD_* (メッセージ記述子構造体)

表 215. 定数の構造	
名前	構造体
MQMD_STRUC_ID	"MD↵"
MQMD_STRUC_ID_ARRAY	'M','D','↵','↵'

注: 記号↵は、単一のブランク文字を表します。

表 216. 定数の値		
名前	10 進数値	16 進値
MQMD_VERSION_1	1	X'00000001'
MQMD_VERSION_2	2	X'00000002'
MQMD_CURRENT_VERSION	2	X'00000002'

MQMDE_* (メッセージ記述子拡張構造体)

表 217. 定数の構造	
名前	構造体
MQMDE_STRUC_ID	"MDE↵"
MQMDE_STRUC_ID_ARRAY	'M','D','E','↵'

注: 記号↵は、単一のブランク文字を表します。

表 218. 定数の値		
名前	10 進数値	16 進値
MQMDE_VERSION_2	2	X'00000002'
MQMDE_CURRENT_VERSION	2	X'00000002'
MQMDE_LENGTH_2	72	X'00000048'

MQMDEF_* (メッセージ記述子拡張フラグ)

表 219. 定数の値		
名前	10 進数値	16 進値
MQMDEF_NONE	0	X'00000000'

MQMDS_* (メッセージ送達順序)

表 220. 定数の値		
名前	10 進数値	16 進値
MQMDS_PRIORITY	0	X'00000000'
MQMDS_FIFO	1	X'00000001'

MQMF_* (メッセージ・フラグ)

表 221. 定数の値		
名前	10 進数値	16 進値
MQMF_SEGMENTATION_INHIBITED	0	X'00000000'
MQMF_SEGMENTATION_ALLOWED	1	X'00000001'

表 221. 定数の値 (続き)		
名前	10 進数値	16 進値
MQMF_MSG_IN_GROUP	8	X'00000008'
MQMF_LAST_MSG_IN_GROUP	16	X'00000010'
MQMF_SEGMENT	2	X'00000002'
MQMF_LAST_SEGMENT	4	X'00000004'
MQMF_NONE	0	X'00000000'

MQMHBO_* (メッセージ・ハンドルからバッファへの変換オプションおよび構造体)

メッセージ・ハンドルからバッファへの変換オプション構造

表 222. 定数の構造	
名前	構造体
MQMHBO_STRUC_ID	"MHBO"
MQMHBO_STRUC_ID_ARRAY	'M','H','B','O'

注: 記号-は、単一のブランク文字を表します。

表 223. 定数の値		
名前	10 進数値	16 進値
MQMHBO_VERSION_1	1	X'00000001'
MQMHBO_CURRENT_VERSION	1	X'00000001'

メッセージ・ハンドルからバッファへの変換オプション

表 224. 定数の値		
名前	10 進数値	16 進値
MQMHBO_PROPERTIES_IN_MQRFH2	1	X'00000001'
MQMHBO_DELETE_PROPERTIES	2	X'00000002'
MQMHBO_NONE	0	X'00000000'

MQMI_* (メッセージ ID)

表 225. 定数名と値	
名前	値
MQMI_NONE	X'00...00' (24 バイト)
MQMI_NONE_ARRAY	'\0','\0',... (24 バイト)

MQMMBI_* (メッセージ参照マーク間隔)

表 226. 定数の値		
名前	10 進数値	16 進値
MQMMBI_UNLIMITED	-1	X'FFFFFFFF'

MQMO_* (一致オプション)

表 227. 定数の値		
名前	10 進数値	16 進値
MQMO_MATCH_MSG_ID	1	X'00000001'
MQMO_MATCH_CORREL_ID	2	X'00000002'
MQMO_MATCH_GROUP_ID	4	X'00000004'
MQMO_MATCH_MSG_SEQ_NUMBER	8	X'00000008'
MQMO_MATCH_OFFSET	16	X'00000010'
MQMO_MATCH_MSG_TOKEN	32	X'00000020'
MQMO_NONE	0	X'00000000'

MQMODE_* (コマンド形式のモード・オプション)

表 228. 定数の値		
名前	10 進数値	16 進値
MQMODE_FORCE	0	X'00000000'
MQMODE_QUIESCE	1	X'00000001'
MQMODE_TERMINATE	2	X'00000002'

MQMON_* (モニター値)

表 229. 定数の値		
名前	10 進数値	16 進値
MQMON_NOT_AVAILABLE	-1	X'FFFFFFFF'
MQMON_NONE	-1	X'FFFFFFFF'
MQMON_Q_MGR	-3	X'FFFFFFFD'
MQMON_OFF	0	X'00000000'
MQMON_ON	1	X'00000001'
MQMON_DISABLED	0	X'00000000'
MQMON_ENABLED	1	X'00000001'
MQMON_LOW	17	X'00000011'
MQMON_MEDIUM	33	X'00000021'
MQMON_HIGH	65	X'00000041'

MQMT_* (メッセージ・タイプ)

表 230. 定数の値		
名前	10 進数値	16 進値
MQMT_SYSTEM_FIRST	1	X'00000001'
MQMT_REQUEST	1	X'00000001'
MQMT_REPLY	2	X'00000002'
MQMT_DATAGRAM	8	X'00000008'
MQMT_REPORT	4	X'00000004'
MQMT_MQE_FIELDS_FROM_MQE	112	X'00000070'

表 230. 定数の値 (続き)		
名前	10 進数値	16 進値
MQMT_MQE_FIELDS	113	X'00000071'
MQMT_SYSTEM_LAST	65535	X'0000FFFF'
MQMT_APPL_FIRST	65536	X'00010000'
MQMT_APPL_LAST	99999999	X'3B9AC9FF'

MQMTOK_* (メッセージ・トークン)

表 231. 定数名と値	
名前	値
MQMTOK_NONE	X'00...00' (16 個のヌル)
MQMTOK_NONE_ARRAY	'\0','\0',... (16 個のヌル)

MQNC_* (名前カウント)

表 232. 定数の値		
名前	10 進数値	16 進値
MQNC_MAX_NAMELIST_NAME_COUNT	256	X'00000100'

MQNPM_* (非持続性メッセージ・クラス)

表 233. 定数の値		
名前	10 進数値	16 進値
MQNPM_CLASS_NORMAL	0	X'00000000'
MQNPM_CLASS_HIGH	10	X'0000000A'

MQNPMS_* (非持続性メッセージ速度)

表 234. 定数の値		
名前	10 進数値	16 進値
MQNPMS_NORMAL	1	X'00000001'
MQNPMS_FAST	2	X'00000002'

MQNT_* (名前リスト・タイプ)

表 235. 定数の値		
名前	10 進数値	16 進値
MQNT_NONE	0	X'00000000'
MQNT_Q	1	X'00000001'
MQNT_CLUSTER	2	X'00000002'
MQNT_AUTH_INFO	4	X'00000004'
MQNT_ALL	1001	X'000003E9'

MQNVS_* (名前/値ストリングの名前)

表 236. 定数名と値	
名前	値
MQNVS_APPL_TYPE	"OPT_APP_GRP"
MQNVS_MSG_TYPE	"OPT_MSG_TYPE"

注: 記号-は、単一の空白文字を表します。

MQOA_* (オブジェクト属性のセクターの制限)

表 237. 定数の値		
名前	10 進数値	16 進値
MQOA_FIRST	1	X'00000001'
MQOA_LAST	9000	X'00002328'

MQOD_* (オブジェクト記述子構造体)

表 238. 定数の構造	
名前	構造体
MQOD_STRUC_ID	"OD"
MQOD_STRUC_ID_ARRAY	'O','D',' ',' '

注: 記号-は、単一の空白文字を表します。

表 239. 定数の値		
名前	10 進数値	16 進値
MQOD_VERSION_1	1	X'00000001'
MQOD_VERSION_2	2	X'00000002'
MQOD_VERSION_3	3	X'00000003'
MQOD_VERSION_4	4	X'00000004'
MQOD_CURRENT_VERSION	4	X'00000004'
MQOD_CURRENT_LENGTH	(value differs by platform or version)	(value differs by platform or version)

MQOII_* (オブジェクト・インスタンス ID)

表 240. 定数名と値	
名前	値
MQOII_NONE	X'00...00' (24 バイト)
MQOII_NONE_ARRAY	'\0','\0',... (24 バイト)

MQOL_* (元の長さ)

表 241. 定数の値		
名前	10 進数値	16 進値
MQOL_UNDEFINED	-1	X'FFFFFFFF'

MQOM_* (照会グループについての古い Db2 メッセージ・オプション)

表 242. 定数の値		
名前	10 進数値	16 進値
MQOM_NO	0	X'00000000'
MQOM_YES	1	X'00000001'

MQOO_* (オープン・オプション)

表 243. 定数の値		
名前	10 進数値	16 進値
MQOO_BIND_AS_Q_DEF	0	X'00000000'
MQOO_READ_AHEAD_AS_Q_DEF	0	X'00000000'
MQOO_INPUT_AS_Q_DEF	1	X'00000001'
MQOO_INPUT_SHARED	2	X'00000002'
MQOO_INPUT_EXCLUSIVE	4	X'00000004'
MQOO_BROWSE	8	X'00000008'
MQOO_OUTPUT	16	X'00000010'
MQOO_INQUIRE	32	X'00000020'
MQOO_SET	64	X'00000040'
MQOO_SAVE_ALL_CONTEXT	128	X'00000080'
MQOO_PASS_IDENTITY_CONTEXT	256	X'00000100'
MQOO_PASS_ALL_CONTEXT	512	X'00000200'
MQOO_SET_IDENTITY_CONTEXT	1024	X'00000400'
MQOO_SET_ALL_CONTEXT	2048	X'00000800'
MQOO_ALTERNATE_USER_AUTHORITY	4096	X'00001000'
MQOO_FAIL_IF QUIESCING	8192	X'00002000'
MQOO_BIND_ON_OPEN	16384	X'00004000'
MQOO_BIND_NOT_FIXED	32768	X'00008000'
MQOO_CO_OP	131072	X'00020000'
MQOO_RESOLVE_LOCAL_TOPIC	262144	X'00040000'
MQOO_NO_READ_AHEAD	524288	X'00080000'
MQOO_READ_AHEAD	1048576	X'00100000'
MQOO_BIND_ON_GROUP	4194304	X'00400000'

MQOO_* (下記は C++ のみで使用される)

表 244. 定数の値		
名前	10 進数値	16 進値
MQOO_RESOLVE_NAMES	65536	X'00010000'
MQOO_RESOLVE_LOCAL_Q	262144	X'00040000'

MQOP_* (MQCTL および MQCB の演算コード)

MQCTL の演算コード

表 245. 定数の値		
名前	10 進数値	16 進値
MQOP_START	1	X'00000001'
MQOP_START_WAIT	2	X'00000002'
MQOP_STOP	4	X'00000004'

MQCB の演算コード

表 246. 定数の値		
名前	10 進数値	16 進値
MQOP_REGISTER	256	X'00000100'
MQOP_DEREGISTER	512	X'00000200'

MQCTL および MQCB の演算コード

表 247. 定数の値		
名前	10 進数値	16 進値
MQOP_SUSPEND	65536	X'00010000'
MQOP_RESUME	131072	X'00020000'

MQOPEN_* (MQOPEN_PRIV 構造体に関連した値)

表 248. 定数の値		
名前	10 進数値	16 進値
MQOPEN_PRIV_VERSION_1	1	X'00000001'
MQOPEN_PRIV_CURRENT_VERSION	1	X'00000001'

MQOPER_* (アクティビティ操作)

表 249. 定数の値		
名前	10 進数値	16 進値
MQOPER_SYSTEM_FIRST	0	X'00000000'
MQOPER_UNKNOWN	0	X'00000000'
MQOPER_BROWSE	1	X'00000001'
MQOPER_DISCARD	2	X'00000002'
MQOPER_GET	3	X'00000003'
MQOPER_PUT	4	X'00000004'
MQOPER_PUT_REPLY	5	X'00000005'
MQOPER_PUT_REPORT	6	X'00000006'
MQOPER_RECEIVE	7	X'00000007'
MQOPER_SEND	8	X'00000008'
MQOPER_TRANSFORM	9	X'00000009'

表 249. 定数の値 (続き)		
名前	10 進数値	16 進値
MQOPER_PUBLISH	10	X'0000000A'
MQOPER_EXCLUDED_PUBLISH	11	X'0000000B'
MQOPER_DISCARDED_PUBLISH	12	X'0000000C'
MQOPER_SYSTEM_LAST	65535	X'0000FFFF'
MQOPER_APPL_FIRST	65536	X'00010000'
MQOPER_APPL_LAST	99999999	X'3B9AC9FF'

MQOT_* (オブジェクト・タイプおよび拡張オブジェクト・タイプ)

オブジェクト・タイプ

表 250. 定数の値		
名前	10 進数値	16 進値
MQOT_NONE	0	X'00000000'
MQOT_Q	1	X'00000001'
MQOT_NAMELIST	2	X'00000002'
MQOT_PROCESS	3	X'00000003'
MQOT_STORAGE_CLASS	4	X'00000004'
MQOT_Q_MGR	5	X'00000005'
MQOT_CHANNEL	6	X'00000006'
MQOT_AUTH_INFO	7	X'00000007'
MQOT_TOPIC	8	X'00000008'
MQOT_CF_STRUC	10	X'0000000A'
MQOT_LISTENER	11	X'0000000B'
MQOT_SERVICE	12	X'0000000C'
MQOT_RESERVED_1	999	X'000003E7'

拡張オブジェクト・タイプ

表 251. 定数の値		
名前	10 進数値	16 進値
MQOT_ALL	1001	X'000003E9'
MQOT_ALIAS_Q	1002	X'000003EA'
MQOT_MODEL_Q	1003	X'000003EB'
MQOT_LOCAL_Q	1004	X'000003EC'
MQOT_REMOTE_Q	1005	X'000003ED'
MQOT_SENDER_CHANNEL	1007	X'000003EF'
MQOT_SERVER_CHANNEL	1008	X'000003F0'
MQOT_REQUESTER_CHANNEL	1009	X'000003F1'
MQOT_RECEIVER_CHANNEL	1010	X'000003F2'
MQOT_CURRENT_CHANNEL	1011	X'000003F3'
MQOT_SAVED_CHANNEL	1012	X'000003F4'

表 251. 定数の値 (続き)		
名前	10 進数値	16 進値
MQOT_SVRCONN_CHANNEL	1013	X'000003F5'
MQOT_CLNTCONN_CHANNEL	1014	X'000003F6'
MQOT_SHORT_CHANNEL	1015	X'000003F7'
MQOT_CHLAUTH	1016	X'000003F8'
MQOT_REMOTE_Q_MGR_NAME	1017	X'000003F9'
MQOT_PROT_POLICY	1019	X'000003FB'
MQOT_TT_CHANNEL	1020	X'000003FC'
MQOT_AMQP_CHANNEL	1021	X'000003FD'
MQOT_AUTH_REC	1022	X'000003FE'

MQPA_* (書き込み権限)

表 252. 定数の値		
名前	10 進数値	16 進値
MQPA_DEFAULT	1	X'00000001'
MQPA_CONTEXT	2	X'00000002'
MQPA_ONLY_MCA	3	X'00000003'
MQPA_ALTERNATE_OR_MCA	4	X'00000004'

MQPD_* (プロパティ記述子、サポートおよびコンテキスト)

プロパティ記述子構造体

表 253. 定数の構造	
名前	構造体
MQPD_STRUC_ID	"PD _{rr} "
MQPD_STRUC_ID_ARRAY	'P', 'D', ' _{rr} ', ' _{rr} '

注: 記号_{rr}は、単一のブランク文字を表します。

表 254. 定数の値		
名前	10 進数値	16 進値
MQPD_VERSION_1	1	X'00000001'
MQPD_CURRENT_VERSION	1	X'00000001'

プロパティ記述子オプション

表 255. 定数の値		
名前	10 進数値	16 進値
MQPD_NONE	0	X'00000000'

プロパティ・サポート・オプション

表 256. 定数の値		
名前	10 進数値	16 進値
MQPD_SUPPORT_OPTIONAL	1	X'00000001'
MQPD_SUPPORT_REQUIRED	1048576	X'00100000'
MQPD_SUPPORT_REQUIRED_IF_LOCAL	1024	X'00000400'
MQPD_REJECT_UNSUP_MASK	-1048576	X'FFF00000'
MQPD_ACCEPT_UNSUP_IF_XMIT_MASK	1047552	X'000FFC00'
MQPD_ACCEPT_UNSUP_MASK	1023	X'000003FF'

プロパティ・コンテキスト

表 257. 定数の値		
名前	10 進数値	16 進値
MQPD_NO_CONTEXT	0	X'00000000'
MQPD_USER_CONTEXT	1	X'00000001'

MQPER_* (持続値)

表 258. 定数の値		
名前	10 進数値	16 進値
MQPER_PERSISTENCE_AS_PARENT	-1	X'FFFFFFFF'
MQPER_NOT_PERSISTENT	0	X'00000000'
MQPER_PERSISTENT	1	X'00000001'
MQPER_PERSISTENCE_AS_Q_DEF	2	X'00000002'
MQPER_PERSISTENCE_AS_TOPIC_DEF	2	X'00000002'

MQPL_* (プラットフォーム)

表 259. 定数の値		
名前	10 進数値	16 進値
MQPL_MVS	1	X'00000001'
MQPL_OS390	1	X'00000001'
MQPL_ZOS	1	X'00000001'
MQPL_OS2	2	X'00000002'
MQPL_AIX	3	X'00000003'
MQPL_UNIX	3	X'00000003'
MQPL_OS400	4	X'00000004'
MQPL_WINDOWS	5	X'00000005'
MQPL_WINDOWS_NT	11	X'0000000B'
MQPL_VMS	12	X'0000000C'
MQPL_NSK	13	X'0000000D'
MQPL_OPEN_TP1	15	X'0000000F'
MQPL_VM	18	X'00000012'

表 259. 定数の値 (続き)		
名前	10 進数値	16 進値
MQPL_TPF	23	X'00000017'
MQPL_VSE	27	X'0000001B'
MQPL_APPLIANCE	28	X'0000001C'
MQPL_NATIVE	1	X'00000001'

MQPMO_* (パブリッシュ・マスクの書き込みメッセージ・オプションおよび構造体)

書き込みメッセージ・オプション構造体

表 260. 定数の構造	
名前	構造体
MQPMO_STRUC_ID	"PMO_"
MQPMO_STRUC_ID_ARRAY	'P', 'M', 'O', '_'

注: 記号_は、単一のブランク文字を表します。

表 261. 定数の値		
名前	10 進数値	16 進値
MQPMO_VERSION_1	1	X'00000001'
MQPMO_VERSION_2	2	X'00000002'
MQPMO_VERSION_3	3	X'00000003'
MQPMO_CURRENT_VERSION	3	X'00000003'
MQPMO_CURRENT_LENGTH	(value differs by platform or version)	(value differs by platform or version)

メッセージ書き込みオプション

表 262. 定数の値		
名前	10 進数値	16 進値
MQPMO_SYNCPOINT	2	X'00000002'
MQPMO_NO_SYNCPOINT	4	X'00000004'
MQPMO_DEFAULT_CONTEXT	32	X'00000020'
MQPMO_NEW_MSG_ID	64	X'00000040'
MQPMO_NEW_CORREL_ID	128	X'00000080'
MQPMO_PASS_IDENTITY_CONTEXT	256	X'00000100'
MQPMO_PASS_ALL_CONTEXT	512	X'00000200'
MQPMO_SET_IDENTITY_CONTEXT	1024	X'00000400'
MQPMO_SET_ALL_CONTEXT	2048	X'00000800'
MQPMO_ALTERNATE_USER_AUTHORITY	4096	X'00001000'
MQPMO_FAIL_IF QUIESCING	8192	X'00002000'
MQPMO_NO_CONTEXT	16384	X'00004000'
MQPMO_LOGICAL_ORDER	32768	X'00008000'

表 262. 定数の値 (続き)		
名前	10 進数値	16 進値
MQPMO_ASYNC_RESPONSE	65536	X'00010000'
MQPMO_SYNC_RESPONSE	131072	X'00020000'
MQPMO_RESOLVE_LOCAL_Q	262144	X'00040000'
MQPMO_RETAIN	2097152	X'00200000'
MQPMO_MD_FOR_OUTPUT_ONLY	8388608	X'00800000'
MQPMO_SCOPE_QMGR	67108864	X'04000000'
MQPMO_SUPPRESS_REPLYTO	134217728	X'08000000'
MQPMO_NOT_OWN_SUBS	268435456	X'10000000'
MQPMO_RESPONSE_AS_Q_DEF	0	X'00000000'
MQPMO_RESPONSE_AS_TOPIC_DEF	0	X'00000000'
MQPMO_NONE	0	X'00000000'

パブリッシュ用のメッセージ書き込みオプション・マスク

表 263. 定数の値		
名前	10 進数値	16 進値
MQPMO_PUB_OPTIONS_MASK	2097152	X'00200000'

MQPMRF_* (メッセージ書き込みレコード・フィールド)

表 264. 定数の値		
名前	10 進数値	16 進値
MQPMRF_MSG_ID	1	X'00000001'
MQPMRF_CORREL_ID	2	X'00000002'
MQPMRF_GROUP_ID	4	X'00000004'
MQPMRF_FEEDBACK	8	X'00000008'
MQPMRF_ACCOUNTING_TOKEN	16	X'00000010'
MQPMRF_NONE	0	X'00000000'

MQPO_* (コマンド形式のページ・オプション)

表 265. 定数の値		
名前	10 進数値	16 進値
MQPO_YES	1	X'00000001'
MQPO_NO	0	X'00000000'

MQPRI_* (優先順位)

表 266. 定数の値		
名前	10 進数値	16 進値
MQPRI_PRIORITY_AS_Q_DEF	-1	X'FFFFFFFF'
MQPRI_PRIORITY_AS_PARENT	-2	X'FFFFFFFE'
MQPRI_PRIORITY_AS_PUBLISHED	-3	X'FFFFFFFD'

表 266. 定数の値 (続き)		
名前	10 進数値	16 進値
MQPRI_PRIORITY_AS_TOPIC_DEF	-1	X'FFFFFFFF'

MQPROP_* (キューおよびチャネル・プロパティ制御値および 最大プロパティ一長)

キューおよびチャネル・プロパティ制御値

表 267. 定数の値		
名前	10 進数値	16 進値
MQPROP_COMPATIBILITY	0	X'00000000'
MQPROP_NONE	1	X'00000001'
MQPROP_ALL	2	X'00000002'
MQPROP_FORCE_MQRFH2	3	X'00000003'

最大プロパティ一長

表 268. 定数の値		
名前	10 進数値	16 進値
MQPROP_UNRESTRICTED_LENGTH	-1	X'FFFFFFFF'

MQPRT_* (応答書き込み値)

表 269. 定数の値		
名前	10 進数値	16 進値
MQPRT_RESPONSE_AS_PARENT	0	X'00000000'
MQPRT_SYNC_RESPONSE	1	X'00000001'
MQPRT_ASYNC_RESPONSE	2	X'00000002'

MQPS_* (パブリッシュ/サブスクライブ)

コマンド形式のパブリッシュ/サブスクライブ状況

表 270. 定数の値		
名前	10 進数値	16 進値
MQPS_STATUS_INACTIVE	0	X'00000000'
MQPS_STATUS_STARTING	1	X'00000001'
MQPS_STATUS_STOPPING	2	X'00000002'
MQPS_STATUS_ACTIVE	3	X'00000003'
MQPS_STATUS_COMPAT	4	X'00000004'
MQPS_STATUS_ERROR	5	X'00000005'
MQPS_STATUS_REFUSED	6	X'00000006'

文字列としてのパブリッシュ/サブスクライブ・タグ

MQPS_COMMAND	"MQPSCommand"

MQPS_COMP_CODE	"MQPSCompCode"
MQPS_CORREL_ID	"MQPSCorrelId"
MQPS_DELETE_OPTIONS	"MQPSDelOpts"
MQPS_ERROR_ID	"MQPSErrorId"
MQPS_ERROR_POS	"MQPSErrorPos"
MQPS_INTEGER_DATA	"MQPSIntData"
MQPS_PARAMETER_ID	"MQPSParmId"
MQPS_PUBLICATION_OPTIONS	"MQSPSPubOpts"
MQPS_PUBLISH_TIMESTAMP	"MQSPSPubTime"
MQPS_Q_MGR_NAME	"MQPSQMgrName"
MQPS_Q_NAME	"MQPSQName"
MQPS_REASON	"MQPSReason"
MQPS_REASON_TEXT	"MQPSReasonText"
MQPS_REGISTRATION_OPTIONS	"MQPSRegOpts"
MQPS_SEQUENCE_NUMBER	"MQPSSeqNum"
MQPS_STREAM_NAME	"MQPSStreamName"
MQPS_STRING_DATA	"MQPSStringData"
MQPS_SUBSCRIPTION_IDENTITY	"MQPSSubIdentity"
MQPS_SUBSCRIPTION_NAME	"MQPSSubName"
MQPS_SUBSCRIPTION_USER_DATA	"MQPSSubUserData"
MQPS_TOPIC	"MQPSTopic"
MQPS_USER_ID	"MQPSUserId"

空白で囲まれた文字列としてのパブリッシュ/サブスクライブ・タグ

MQPS_COMMAND_B	"~MQPSCommand~"
MQPS_COMP_CODE_B	"~MQPSCompCode~"
MQPS_CORREL_ID_B	"~MQPSCorrelId~"
MQPS_DELETE_OPTIONS_B	"~MQPSDelOpts~"
MQPS_ERROR_ID_B	"~MQPSErrorId~"
MQPS_ERROR_POS_B	"~MQPSErrorPos~"
MQPS_INTEGER_DATA_B	"~MQPSIntData~"
MQPS_PARAMETER_ID_B	"~MQSPSParmId~"
MQPS_PUBLICATION_OPTIONS_B	"~MQSPSPubOpts~"
MQPS_PUBLISH_TIMESTAMP_B	"~MQSPSPubTime~"
MQPS_Q_MGR_NAME_B	"~MQPSQMgrName~"
MQPS_Q_NAME_B	"~MQPSQName~"

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

注：記号¬は、単一のブランク文字を表します。

ストリングとしてのパブリッシュ/サブスクライブ・コマンド・タグ値

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

ブランクで囲まれたストリングとしてのパブリッシュ/サブスクライブ・コマンド・タグ値

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

注：記号¬は、単一のブランク文字を表します。

ストリングとしてのパブリッシュ/サブスクライブ・オプション・タグ値

MQPS_ADD_NAME	"AddName"
MQPS_ANONYMOUS	"Anon"
MQPS_CORREL_ID_AS_IDENTITY	"CorrelAsId"

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

空白で囲まれた文字列としてのパブリッシュ/サブスクライブ・オプション・タグ値

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

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

注：記号¬は、単一のブランク文字を表します。

MQPSC_* (パブリッシュ/サブスクライブ・オプション・タグ・パブリッシュ/サブスクライブ・コマンド・フォルダー (psc) のタグ)

表 271. 定数の値		
名前	10 進数値	16 進値
MQPSC_FOLDER_VERSION	1	X'00000001'

MQPSC_* (パブリッシュ/サブスクライブ・オプション・タグのタグ名)

MQPSC_COMMAND	"Command"
MQPSC_REGISTRATION_OPTION	"RegOpt"
MQPSC_PUBLICATION_OPTION	"PubOpt"
MQPSC_DELETE_OPTION	"DelOpt"
MQPSC_TOPIC	"Topic"
MQPSC_SUBSCRIPTION_POINT	"SubPoint"
MQPSC_FILTER	"Filter"
MQPSC_Q_MGR_NAME	"QMgrName"
MQPSC_Q_NAME	"QName"
MQPSC_PUBLISH_TIMESTAMP	"PubTime"
MQPSC_SEQUENCE_NUMBER	"SeqNum"

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

MQPSC_* (パブリッシュ/サブスクライブ・オプション・タグのXML タグ名)

MQPSC_COMMAND_B	"<Command>"
MQPSC_COMMAND_E	"</Command>"
MQPSC_REGISTRATION_OPTION_B	"<RegOpt>"
MQPSC_REGISTRATION_OPTION_E	"</RegOpt>"
MQPSC_PUBLICATION_OPTION_B	"<PubOpt>"
MQPSC_PUBLICATION_OPTION_E	"</PubOpt>"
MQPSC_DELETE_OPTION_B	"<DelOpt>"
MQPSC_DELETE_OPTION_E	"</DelOpt>"
MQPSC_TOPIC_B	"<Topic>"
MQPSC_TOPIC_E	"</Topic>"
MQPSC_SUBSCRIPTION_POINT_B	"<SubPoint>"
MQPSC_SUBSCRIPTION_POINT_E	"</SubPoint>"
MQPSC_FILTER_B	"<Filter>"
MQPSC_FILTER_E	"</Filter>"
MQPSC_Q_MGR_NAME_B	"<QMgrName>"
MQPSC_Q_MGR_NAME_E	"</QMgrName>"
MQPSC_Q_NAME_B	"<QName>"
MQPSC_Q_NAME_E	"</QName>"
MQPSC_PUBLISH_TIMESTAMP_B	"<PubTime>"
MQPSC_PUBLISH_TIMESTAMP_E	"</PubTime>"
MQPSC_SEQUENCE_NUMBER_B	"<SeqNum>"
MQPSC_SEQUENCE_NUMBER_E	"</SeqNum>"
MQPSC_SUBSCRIPTION_NAME_B	"<SubName>"
MQPSC_SUBSCRIPTION_NAME_E	"</SubName>"
MQPSC_SUBSCRIPTION_IDENTITY_B	"<SubIdentity>"
MQPSC_SUBSCRIPTION_IDENTITY_E	"</SubIdentity>"
MQPSC_SUBSCRIPTION_USER_DATA_B	"<SubUserData>"
MQPSC_SUBSCRIPTION_USER_DATA_E	"</SubUserData>"
MQPSC_CORREL_ID_B	"<CorrelId>"
MQPSC_CORREL_ID_E	"</CorrelId>"

MQPSC_* (ストリングとしてのパブリッシュ/サブスクライブ・オプション・タグ・パブリッシャー値)

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

MQPSC_* (ストリングとしてのパブリッシュ/サブスクライブ・オプション・タグ名値)

MQPSC_ADD_NAME	"AddName"
MQPSC_CORREL_ID_AS_IDENTITY	"CorrelAsId"
MQPSC_DEREGISTER_ALL	"DeregAll"
MQPSC_DUPLICATES_OK	"DupsOK"
MQPSC_FULL_RESPONSE	"FullResp"
MQPSC_INFORM_IF_RETAINED	"InformIfRet"
MQPSC_IS_RETAINED_PUB	"IsRetainedPub"
MQPSC_JOIN_SHARED	"JoinShared"
MQPSC_JOIN_EXCLUSIVE	"JoinExcl"
MQPSC_LEAVE_ONLY	"LeaveOnly"
MQPSC_LOCAL	"Local"
MQPSC_LOCKED	"Locked"
MQPSC_NEW_PUBS_ONLY	"NewPubsOnly"
MQPSC_NO_ALTERATION	"NoAlter"
MQPSC_NON_PERSISTENT	"NonPers"
MQPSC_OTHER_SUBS_ONLY	"OtherSubsOnly"
MQPSC_PERSISTENT	"Pers"
MQPSC_PERSISTENT_AS_PUBLISH	"PersAsPub"
MQPSC_PERSISTENT_AS_Q	"PersAsQueue"
MQPSC_NONE	"None"
MQPSC_PUB_ON_REQUEST_ONLY	"PubOnReqOnly"
MQPSC_RETAIN_PUB	"RetainPub"
MQPSC_VARIABLE_USER_ID	"VariableUserId"

MQPSCR_* (パブリッシュ/サブスクライブ・オプション)

パブリッシュ/サブスクライブ・オプション・タグのパブリッシュ/サブスクライブ応答フォルダー (pscr) のタグ

表 272. 定数の値		
名前	10 進数値	16 進値
MQPSCR_FOLDER_VERSION	1	X'00000001'

パブリッシュ/サブスクライブ・オプション・タグのタグ名

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

パブリッシュ/サブスクライブ・オプション・タグの XML タグ名

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

パブリッシュ/サブスクライブ・オプション・タグのタグ値

MQPSCR_OK	"ok"
MQPSCR_WARNING	"warning"
MQPSCR_ERROR	"error"

MQPSM_* (パブリッシュ/サブスクライブ・モード)

表 273. 定数の値		
名前	10 進数値	16 進値
MQPSM_DISABLED	0	X'00000000'
MQPSM_COMPAT	1	X'00000001'
MQPSM_ENABLED	2	X'00000002'

MQPSPROP_* (パブリッシュ/サブスクライブ・メッセージ・プロパティ)

表 274. 定数の値		
名前	10 進数値	16 進値
MQPSPROP_NONE	0	X'00000000'
MQPSPROP_COMPAT	1	X'00000001'
MQPSPROP_RFH2	2	X'00000002'
MQPSPROP_MSGPROP	3	X'00000003'

MQPSST_* (コマンド形式のパブリッシュ/サブスクライブ状況タイプ)

表 275. 定数の値		
名前	10 進数値	16 進値
MQPSST_ALL	0	X'00000000'
MQPSST_LOCAL	1	X'00000001'
MQPSST_PARENT	2	X'00000002'
MQPSST_CHILD	3	X'00000003'

MQPUBO_* (パブリッシュ/サブスクライブ・パブリケーション・オプション)

表 276. 定数の値		
名前	10 進数値	16 進値
MQPUBO_NONE	0	X'00000000'
MQPUBO_CORREL_ID_AS_IDENTITY	1	X'00000001'
MQPUBO_RETAIN_PUBLICATION	2	X'00000002'
MQPUBO_OTHER_SUBSCRIBERS_ONLY	4	X'00000004'
MQPUBO_NO_REGISTRATION	8	X'00000008'
MQPUBO_IS_RETAINED_PUBLICATION	16	X'00000010'

MQPXP_* (パブリッシュ/サブスクライブ経路指定出口パラメーター構造体)

表 277. 定数の構造	
名前	構造体
MQPXP_STRUC_ID	"PXP"
MQPXP_STRUC_ID_ARRAY	'P','X','P','P'

注: 記号-は、単一の空白文字を表します。

表 278. 定数の値		
名前	10 進数値	16 進値
MQPXP_VERSION_1	1	X'00000001'
MQPXP_CURRENT_VERSION	1	X'00000001'

MQQA_* (キュー属性)

読み取り禁止値

表 279. 定数の値		
名前	10 進数値	16 進値
MQQA_GET_INHIBITED	1	X'00000001'
MQQA_GET_ALLOWED	0	X'00000000'

書き込み禁止値

表 280. 定数の値		
名前	10 進数値	16 進値
MQQA_PUT_INHIBITED	1	X'00000001'

表 280. 定数の値 (続き)		
名前	10 進数値	16 進値
MQQA_PUT_ALLOWED	0	X'00000000'

キューの共有可能性

表 281. 定数の値		
名前	10 進数値	16 進値
MQQA_SHAREABLE	1	X'00000001'
MQQA_NOT_SHAREABLE	0	X'00000000'

バックアウトの強化

表 282. 定数の値		
名前	10 進数値	16 進値
MQQA_BACKOUT_HARDENED	1	X'00000001'
MQQA_BACKOUT_NOT_HARDENED	0	X'00000000'

MQQDT_* (キュー定義タイプ)

表 283. 定数の値		
名前	10 進数値	16 進値
MQQDT_PREDEFINED	1	X'00000001'
MQQDT_PERMANENT_DYNAMIC	2	X'00000002'
MQQDT_TEMPORARY_DYNAMIC	3	X'00000003'
MQQDT_SHARED_DYNAMIC	4	X'00000004'

MQQF_* (キュー・フラグ)

表 284. 定数の値		
名前	10 進数値	16 進値
MQQF_LOCAL_Q	1	X'00000001'
MQQF_CLWL_USEQ_ANY	64	X'00000040'
MQQF_CLWL_USEQ_LOCAL	128	X'00000080'

MQQMDT_* (コマンド形式のキュー・マネージャー定義タイプ)

表 285. 定数の値		
名前	10 進数値	16 進値
MQQMDT_EXPLICIT_CLUSTER_SENDER	1	X'00000001'
MQQMDT_AUTO_CLUSTER_SENDER	2	X'00000002'
MQQMDT_AUTO_EXP_CLUSTER_SENDER	4	X'00000004'
MQQMDT_CLUSTER_RECEIVER	3	X'00000003'

MQQMF_* (キュー・マネージャー・フラグ)

表 286. 定数の値		
名前	10 進数値	16 進値
MQQMF_REPOSITORY_Q_MGR	2	X'00000002'
MQQMF_CLUSSDR_USER_DEFINED	8	X'00000008'
MQQMF_CLUSSDR_AUTO_DEFINED	16	X'00000010'
MQQMF_AVAILABLE	32	X'00000020'

MQQMFACT_* (コマンド形式のキュー・マネージャー機能)

表 287. 定数の値		
名前	10 進数値	16 進値
MQQMFACT_IMS_BRIDGE	1	X'00000001'
MQQMFACT_DB2	2	X'00000002'

MQQMSTA_* (コマンド形式のキュー・マネージャー状況)

表 288. 定数の値		
名前	10 進数値	16 進値
MQQMSTA_STARTING	1	X'00000001'
MQQMSTA_RUNNING	2	X'00000002'
MQQMSTA QUIESCING	3	X'00000003'

MQQMT_* (コマンド形式のキュー・マネージャー・タイプ)

表 289. 定数の値		
名前	10 進数値	16 進値
MQQMT_NORMAL	0	X'00000000'
MQQMT_REPOSITORY	1	X'00000001'

MQQO_* (コマンド形式の静止オプション)

表 290. 定数の値		
名前	10 進数値	16 進値
MQQO_YES	1	X'00000001'
MQQO_NO	0	X'00000000'

MQQS_* (量子耐性アルゴリズムのオプション)

表 291. 定数の値		
名前	10 進数値	16 進値
MQQS_不明	-1	X'FFFFFFFF'
MQQS_UNPROTECTED	0	X'00000000'
MQQS_HYBMLKEM	1	X'00000001'
MQQS_MLKEM	2	X'00000002'

MQQSGD_* (キュー共有グループ処理)

表 292. 定数の値		
名前	10 進数値	16 進値
MQQSGD_ALL	-1	X'FFFFFFFF'
MQQSGD_Q_MGR	0	X'00000000'
MQQSGD_COPY	1	X'00000001'
MQQSGD_SHARED	2	X'00000002'
MQQSGD_GROUP	3	X'00000003'
MQQSGD_PRIVATE	4	X'00000004'
MQQSGD_LIVE	6	X'00000006'

MQQSGS_* (コマンド形式のキュー共有グループ状況)

表 293. 定数の値		
名前	10 進数値	16 進値
MQQSGS_UNKNOWN	0	X'00000000'
MQQSGS_CREATED	1	X'00000001'
MQQSGS_ACTIVE	2	X'00000002'
MQQSGS_INACTIVE	3	X'00000003'
MQQSGS_FAILED	4	X'00000004'
MQQSGS_PENDING	5	X'00000005'

MQQSIE_* (コマンド形式のキュー・サービス間隔イベント)

表 294. 定数の値		
名前	10 進数値	16 進値
MQQSIE_NONE	0	X'00000000'
MQQSIE_HIGH	1	X'00000001'
MQQSIE_OK	2	X'00000002'

MQQSO_* (コマンド形式の SET、BROWSE、INPUT のキュー状況オープン・オプション)

表 295. 定数の値		
名前	10 進数値	16 進値
MQQSO_NO	0	X'00000000'
MQQSO_YES	1	X'00000001'
MQQSO_SHARED	1	X'00000001'
MQQSO_EXCLUSIVE	2	X'00000002'

MQQSOT_* (コマンド形式のキュー状況オープン・タイプ)

表 296. 定数の値		
名前	10 進数値	16 進値
MQQSOT_ALL	1	X'00000001'

表 296. 定数の値 (続き)		
名前	10 進数値	16 進値
MQQSOT_INPUT	2	X'00000002'
MQQSOT_OUTPUT	3	X'00000003'

MQQSR_* (量子耐性が必要なオプション)

表 297. 定数の値		
名前	10 進数値	16 進値
MQQSR_OPTIONAL	0	X'00000000'
MQQSR_必須	1	X'00000001'

MQQSUM_* (コマンド形式のキュー状況未コミット・メッセージ)

表 298. 定数の値		
名前	10 進数値	16 進値
MQQSUM_YES	1	X'00000001'
MQQSUM_NO	0	X'00000000'

MQQT_* (キュー・タイプおよび拡張キュー・タイプ)

キュー・タイプ

表 299. 定数の値		
名前	10 進数値	16 進値
MQQT_LOCAL	1	X'00000001'
MQQT_MODEL	2	X'00000002'
MQQT_ALIAS	3	X'00000003'
MQQT_REMOTE	6	X'00000006'
MQQT_CLUSTER	7	X'00000007'

拡張キュー・タイプ

表 300. 定数の値		
名前	10 進数値	16 進値
MQQT_ALL	1001	X'000003E9'

MQRC_* (理由コード)

表 301. 定数の値		
名前	10 進数値	16 進値
MQRC_NONE	0	X'00000000'
MQRC_APPL_FIRST	900	X'00000384'
MQRC_APPL_LAST	999	X'000003E7'
MQRC_ALIAS_BASE_Q_TYPE_ERROR	2001	X'000007D1'
MQRC_ALREADY_CONNECTED	2002	X'000007D2'

表 301. 定数の値 (続き)		
名前	10 進数値	16 進値
MQRC_BACKED_OUT	2003	X'000007D3'
MQRC_BUFFER_ERROR	2004	X'000007D4'
MQRC_BUFFER_LENGTH_ERROR	2005	X'000007D5'
MQRC_CHAR_ATTR_LENGTH_ERROR	2006	X'000007D6'
MQRC_CHAR_ATTRS_ERROR	2007	X'000007D7'
MQRC_CHAR_ATTRS_TOO_SHORT	2008	X'000007D8'
MQRC_CONNECTION_BROKEN	2009	X'000007D9'
MQRC_DATA_LENGTH_ERROR	2010	X'000007DA'
MQRC_DYNAMIC_Q_NAME_ERROR	2011	X'000007DB'
MQRC_ENVIRONMENT_ERROR	2012	X'000007DC'
MQRC_EXPIRY_ERROR	2013	X'000007DD'
MQRC_FEEDBACK_ERROR	2014	X'000007DE'
MQRC_GET_INHIBITED	2016	X'000007E0'
MQRC_HANDLE_NOT_AVAILABLE	2017	X'000007E1'
MQRC_HCONN_ERROR	2018	X'000007E2'
MQRC_HOBJ_ERROR	2019	X'000007E3'
MQRC_INHIBIT_VALUE_ERROR	2020	X'000007E4'
MQRC_INT_ATTR_COUNT_ERROR	2021	X'000007E5'
MQRC_INT_ATTR_COUNT_TOO_SMALL	2022	X'000007E6'
MQRC_INT_ATTRS_ARRAY_ERROR	2023	X'000007E7'
MQRC_SYNCPOINT_LIMIT_REACHED	2024	X'000007E8'
MQRC_MAX_CONNS_LIMIT_REACHED	2025	X'000007E9'
MQRC_MD_ERROR	2026	X'000007EA'
MQRC_MISSING_REPLY_TO_Q	2027	X'000007EB'
MQRC_MSG_TYPE_ERROR	2029	X'000007ED'
MQRC_MSG_TOO_BIG_FOR_Q	2030	X'000007EE'
MQRC_MSG_TOO_BIG_FOR_Q_MGR	2031	X'000007EF'
MQRC_NO_MSG_AVAILABLE	2033	X'000007F1'
MQRC_NO_MSG_UNDER_CURSOR	2034	X'000007F2'
MQRC_NOT_AUTHORIZED	2035	X'000007F3'
MQRC_NOT_OPEN_FOR_BROWSE	2036	X'000007F4'
MQRC_NOT_OPEN_FOR_INPUT	2037	X'000007F5'
MQRC_NOT_OPEN_FOR_INQUIRE	2038	X'000007F6'
MQRC_NOT_OPEN_FOR_OUTPUT	2039	X'000007F7'
MQRC_NOT_OPEN_FOR_SET	2040	X'000007F8'
MQRC_OBJECT_CHANGED	2041	X'000007F9'
MQRC_OBJECT_IN_USE	2042	X'000007FA'
MQRC_OBJECT_TYPE_ERROR	2043	X'000007FB'
MQRC_OD_ERROR	2044	X'000007FC'

表 301. 定数の値 (続き)		
名前	10 進数値	16 進値
MQRC_OPTION_NOT_VALID_FOR_TYPE	2045	X'000007FD'
MQRC_OPTIONS_ERROR	2046	X'000007FE'
MQRC_PERSISTENCE_ERROR	2047	X'000007FF'
MQRC_PERSISTENT_NOT_ALLOWED	2048	X'00000800'
MQRC_PRIORITY_EXCEEDS_MAXIMUM	2049	X'00000801'
MQRC_PRIORITY_ERROR	2050	X'00000802'
MQRC_PUT_INHIBITED	2051	X'00000803'
MQRC_Q_DELETED	2052	X'00000804'
MQRC_Q_FULL	2053	X'00000805'
MQRC_Q_NOT_EMPTY	2055	X'00000807'
MQRC_Q_SPACE_NOT_AVAILABLE	2056	X'00000808'
MQRC_Q_TYPE_ERROR	2057	X'00000809'
MQRC_Q_MGR_NAME_ERROR	2058	X'0000080A'
MQRC_Q_MGR_NOT_AVAILABLE	2059	X'0000080B'
MQRC_REPORT_OPTIONS_ERROR	2061	X'0000080D'
MQRC_SECOND_MARK_NOT_ALLOWED	2062	X'0000080E'
MQRC_SECURITY_ERROR	2063	X'0000080F'
MQRC_SELECTOR_COUNT_ERROR	2065	X'00000811'
MQRC_SELECTOR_LIMIT_EXCEEDED	2066	X'00000812'
MQRC_SELECTOR_ERROR	2067	X'00000813'
MQRC_SELECTOR_NOT_FOR_TYPE	2068	X'00000814'
MQRC_SIGNAL_OUTSTANDING	2069	X'00000815'
MQRC_SIGNAL_REQUEST_ACCEPTED	2070	X'00000816'
MQRC_STORAGE_NOT_AVAILABLE	2071	X'00000817'
MQRC_SYNCPOINT_NOT_AVAILABLE	2072	X'00000818'
MQRC_TRIGGER_CONTROL_ERROR	2075	X'0000081B'
MQRC_TRIGGER_DEPTH_ERROR	2076	X'0000081C'
MQRC_TRIGGER_MSG_PRIORITY_ERR	2077	X'0000081D'
MQRC_TRIGGER_TYPE_ERROR	2078	X'0000081E'
MQRC_TRUNCATED_MSG_ACCEPTED	2079	X'0000081F'
MQRC_TRUNCATED_MSG_FAILED	2080	X'00000820'
MQRC_UNKNOWN_ALIAS_BASE_Q	2082	X'00000822'
MQRC_UNKNOWN_OBJECT_NAME	2085	X'00000825'
MQRC_UNKNOWN_OBJECT_Q_MGR	2086	X'00000826'
MQRC_UNKNOWN_REMOTE_Q_MGR	2087	X'00000827'
MQRC_WAIT_INTERVAL_ERROR	2090	X'0000082A'
MQRC_XMIT_Q_TYPE_ERROR	2091	X'0000082B'
MQRC_XMIT_Q_USAGE_ERROR	2092	X'0000082C'
MQRC_NOT_OPEN_FOR_PASS_ALL	2093	X'0000082D'

表 301. 定数の値 (続き)		
名前	10 進数値	16 進値
MQRC_NOT_OPEN_FOR_PASS_IDENT	2094	X'0000082E'
MQRC_NOT_OPEN_FOR_SET_ALL	2095	X'0000082F'
MQRC_NOT_OPEN_FOR_SET_IDENT	2096	X'00000830'
MQRC_CONTEXT_HANDLE_ERROR	2097	X'00000831'
MQRC_CONTEXT_NOT_AVAILABLE	2098	X'00000832'
MQRC_SIGNAL1_ERROR	2099	X'00000833'
MQRC_OBJECT_ALREADY_EXISTS	2100	X'00000834'
MQRC_OBJECT_DAMAGED	2101	X'00000835'
MQRC_RESOURCE_PROBLEM	2102	X'00000836'
MQRC_ANOTHER_Q_MGR_CONNECTED	2103	X'00000837'
MQRC_UNKNOWN_REPORT_OPTION	2104	X'00000838'
MQRC_STORAGE_CLASS_ERROR	2105	X'00000839'
MQRC_COD_NOT_VALID_FOR_XCF_Q	2106	X'0000083A'
MQRC_XWAIT_CANCELED	2107	X'0000083B'
MQRC_XWAIT_ERROR	2108	X'0000083C'
MQRC_SUPPRESSED_BY_EXIT	2109	X'0000083D'
MQRC_FORMAT_ERROR	2110	X'0000083E'
MQRC_SOURCE_CCSID_ERROR	2111	X'0000083F'
MQRC_SOURCE_INTEGER_ENC_ERROR	2112	X'00000840'
MQRC_SOURCE_DECIMAL_ENC_ERROR	2113	X'00000841'
MQRC_SOURCE_FLOAT_ENC_ERROR	2114	X'00000842'
MQRC_TARGET_CCSID_ERROR	2115	X'00000843'
MQRC_TARGET_INTEGER_ENC_ERROR	2116	X'00000844'
MQRC_TARGET_DECIMAL_ENC_ERROR	2117	X'00000845'
MQRC_TARGET_FLOAT_ENC_ERROR	2118	X'00000846'
MQRC_NOT_CONVERTED	2119	X'00000847'
MQRC_CONVERTED_MSG_TOO_BIG	2120	X'00000848'
MQRC_TRUNCATED	2120	X'00000848'
MQRC_NO_EXTERNAL_PARTICIPANTS	2121	X'00000849'
MQRC_PARTICIPANT_NOT_AVAILABLE	2122	X'0000084A'
MQRC_OUTCOME_MIXED	2123	X'0000084B'
MQRC_OUTCOME_PENDING	2124	X'0000084C'
MQRC_BRIDGE_STARTED	2125	X'0000084D'
MQRC_BRIDGE_STOPPED	2126	X'0000084E'
MQRC_ADAPTER_STORAGE_SHORTAGE	2127	X'0000084F'
MQRC_UOW_IN_PROGRESS	2128	X'00000850'
MQRC_ADAPTER_CONN_LOAD_ERROR	2129	X'00000851'
MQRC_ADAPTER_SERV_LOAD_ERROR	2130	X'00000852'
MQRC_ADAPTER_DEFS_ERROR	2131	X'00000853'

表 301. 定数の値 (続き)		
名前	10 進数値	16 進値
MQRC_ADAPTER_DEFS_LOAD_ERROR	2132	X'00000854'
MQRC_ADAPTER_CONV_LOAD_ERROR	2133	X'00000855'
MQRC_BO_ERROR	2134	X'00000856'
MQRC_DH_ERROR	2135	X'00000857'
MQRC_MULTIPLE_REASONS	2136	X'00000858'
MQRC_OPEN_FAILED	2137	X'00000859'
MQRC_ADAPTER_DISC_LOAD_ERROR	2138	X'0000085A'
MQRC_CNO_ERROR	2139	X'0000085B'
MQRC_CICS_WAIT_FAILED	2140	X'0000085C'
MQRC_DLH_ERROR	2141	X'0000085D'
MQRC_HEADER_ERROR	2142	X'0000085E'
MQRC_SOURCE_LENGTH_ERROR	2143	X'0000085F'
MQRC_TARGET_LENGTH_ERROR	2144	X'00000860'
MQRC_SOURCE_BUFFER_ERROR	2145	X'00000861'
MQRC_TARGET_BUFFER_ERROR	2146	X'00000862'
MQRC_IIH_ERROR	2148	X'00000864'
MQRC_PCF_ERROR	2149	X'00000865'
MQRC_DBCS_ERROR	2150	X'00000866'
MQRC_OBJECT_NAME_ERROR	2152	X'00000868'
MQRC_OBJECT_Q_MGR_NAME_ERROR	2153	X'00000869'
MQRC_RECS_PRESENT_ERROR	2154	X'0000086A'
MQRC_OBJECT_RECORDS_ERROR	2155	X'0000086B'
MQRC_RESPONSE_RECORDS_ERROR	2156	X'0000086C'
MQRC_ASID_MISMATCH	2157	X'0000086D'
MQRC_PMO_RECORD_FLAGS_ERROR	2158	X'0000086E'
MQRC_PUT_MSG_RECORDS_ERROR	2159	X'0000086F'
MQRC_CONN_ID_IN_USE	2160	X'00000870'
MQRC_Q_MGR QUIESCING	2161	X'00000871'
MQRC_Q_MGR_STOPPING	2162	X'00000872'
MQRC_DUPLICATE_RECOV_COORD	2163	X'00000873'
MQRC_PMO_ERROR	2173	X'0000087D'
MQRC_API_EXIT_NOT_FOUND	2182	X'00000886'
MQRC_API_EXIT_LOAD_ERROR	2183	X'00000887'
MQRC_REMOTE_Q_NAME_ERROR	2184	X'00000888'
MQRC_INCONSISTENT_PERSISTENCE	2185	X'00000889'
MQRC_GMO_ERROR	2186	X'0000088A'
MQRC_CICS_BRIDGE_RESTRICTION	2187	X'0000088B'
MQRC_STOPPED_BY_CLUSTER_EXIT	2188	X'0000088C'
MQRC_CLUSTER_RESOLUTION_ERROR	2189	X'0000088D'

表 301. 定数の値 (続き)		
名前	10 進数値	16 進値
MQRC_CONVERTED_STRING_TOO_BIG	2190	X'0000088E'
MQRC_TMC_ERROR	2191	X'0000088F'
MQRC_PAGESET_FULL	2192	X'00000890'
MQRC_STORAGE_MEDIUM_FULL	2192	X'00000890'
MQRC_PAGESET_ERROR	2193	X'00000891'
MQRC_NAME_NOT_VALID_FOR_TYPE	2194	X'00000892'
MQRC_UNEXPECTED_ERROR	2195	X'00000893'
MQRC_UNKNOWN_XMIT_Q	2196	X'00000894'
MQRC_UNKNOWN_DEF_XMIT_Q	2197	X'00000895'
MQRC_DEF_XMIT_Q_TYPE_ERROR	2198	X'00000896'
MQRC_DEF_XMIT_Q_USAGE_ERROR	2199	X'00000897'
MQRC_MSG_MARKED_BROWSE_CO_OP	2200	X'00000898'
MQRC_NAME_IN_USE	2201	X'00000899'
MQRC_CONNECTION QUIESCING	2202	X'0000089A'
MQRC_CONNECTION_STOPPING	2203	X'0000089B'
MQRC_ADAPTER_NOT_AVAILABLE	2204	X'0000089C'
MQRC_MSG_ID_ERROR	2206	X'0000089E'
MQRC_CORREL_ID_ERROR	2207	X'0000089F'
MQRC_FILE_SYSTEM_ERROR	2208	X'000008A0'
MQRC_NO_MSG_LOCKED	2209	X'000008A1'
MQRC_SOAP_DOTNET_ERROR	2210	X'000008A2'
MQRC_SOAP_AXIS_ERROR	2211	X'000008A3'
MQRC_SOAP_URL_ERROR	2212	X'000008A4'
MQRC_FILE_NOT_AUDITED	2216	X'000008A8'
MQRC_CONNECTION_NOT_AUTHORIZED	2217	X'000008A9'
MQRC_MSG_TOO_BIG_FOR_CHANNEL	2218	X'000008AA'
MQRC_CALL_IN_PROGRESS	2219	X'000008AB'
MQRC_RMH_ERROR	2220	X'000008AC'
MQRC_Q_MGR_ACTIVE	2222	X'000008AE'
MQRC_Q_MGR_NOT_ACTIVE	2223	X'000008AF'
MQRC_Q_DEPTH_HIGH	2224	X'000008B0'
MQRC_Q_DEPTH_LOW	2225	X'000008B1'
MQRC_Q_SERVICE_INTERVAL_HIGH	2226	X'000008B2'
MQRC_Q_SERVICE_INTERVAL_OK	2227	X'000008B3'
MQRC_RFH_HEADER_FIELD_ERROR	2228	X'000008B4'
MQRC_RAS_PROPERTY_ERROR	2229	X'000008B5'
MQRC_UNIT_OF_WORK_NOT_STARTED	2232	X'000008B8'
MQRC_CHANNEL_AUTO_DEF_OK	2233	X'000008B9'
MQRC_CHANNEL_AUTO_DEF_ERROR	2234	X'000008BA'

表 301. 定数の値 (続き)		
名前	10 進数値	16 進値
MQRC_CFH_ERROR	2235	X'000008BB'
MQRC_CFIL_ERROR	2236	X'000008BC'
MQRC_CFIN_ERROR	2237	X'000008BD'
MQRC_CFSL_ERROR	2238	X'000008BE'
MQRC_CFST_ERROR	2239	X'000008BF'
MQRC_INCOMPLETE_GROUP	2241	X'000008C1'
MQRC_INCOMPLETE_MSG	2242	X'000008C2'
MQRC_INCONSISTENT_CCSDS	2243	X'000008C3'
MQRC_INCONSISTENT_ENCODINGS	2244	X'000008C4'
MQRC_INCONSISTENT_UOW	2245	X'000008C5'
MQRC_INVALID_MSG_UNDER_CURSOR	2246	X'000008C6'
MQRC_MATCH_OPTIONS_ERROR	2247	X'000008C7'
MQRC_MDE_ERROR	2248	X'000008C8'
MQRC_MSG_FLAGS_ERROR	2249	X'000008C9'
MQRC_MSG_SEQ_NUMBER_ERROR	2250	X'000008CA'
MQRC_OFFSET_ERROR	2251	X'000008CB'
MQRC_ORIGINAL_LENGTH_ERROR	2252	X'000008CC'
MQRC_SEGMENT_LENGTH_ZERO	2253	X'000008CD'
MQRC_UOW_NOT_AVAILABLE	2255	X'000008CF'
MQRC_WRONG_GMO_VERSION	2256	X'000008D0'
MQRC_WRONG_MD_VERSION	2257	X'000008D1'
MQRC_GROUP_ID_ERROR	2258	X'000008D2'
MQRC_INCONSISTENT_BROWSE	2259	X'000008D3'
MQRC_XQH_ERROR	2260	X'000008D4'
MQRC_SRC_ENV_ERROR	2261	X'000008D5'
MQRC_SRC_NAME_ERROR	2262	X'000008D6'
MQRC_DEST_ENV_ERROR	2263	X'000008D7'
MQRC_DEST_NAME_ERROR	2264	X'000008D8'
MQRC_TM_ERROR	2265	X'000008D9'
MQRC_CLUSTER_EXIT_ERROR	2266	X'000008DA'
MQRC_CLUSTER_EXIT_LOAD_ERROR	2267	X'000008DB'
MQRC_CLUSTER_PUT_INHIBITED	2268	X'000008DC'
MQRC_CLUSTER_RESOURCE_ERROR	2269	X'000008DD'
MQRC_NO_DESTINATIONS_AVAILABLE	2270	X'000008DE'
MQRC_CONN_TAG_IN_USE	2271	X'000008DF'
MQRC_PARTIALLY_CONVERTED	2272	X'000008E0'
MQRC_CONNECTION_ERROR	2273	X'000008E1'
MQRC_OPTION_ENVIRONMENT_ERROR	2274	X'000008E2'
MQRC_CD_ERROR	2277	X'000008E5'

表 301. 定数の値 (続き)		
名前	10 進数値	16 進値
MQRC_CLIENT_CONN_ERROR	2278	X'000008E6'
MQRC_CHANNEL_STOPPED_BY_USER	2279	X'000008E7'
MQRC_HCONFIG_ERROR	2280	X'000008E8'
MQRC_FUNCTION_ERROR	2281	X'000008E9'
MQRC_CHANNEL_STARTED	2282	X'000008EA'
MQRC_CHANNEL_STOPPED	2283	X'000008EB'
MQRC_CHANNEL_CONV_ERROR	2284	X'000008EC'
MQRC_SERVICE_NOT_AVAILABLE	2285	X'000008ED'
MQRC_INITIALIZATION_FAILED	2286	X'000008EE'
MQRC_TERMINATION_FAILED	2287	X'000008EF'
MQRC_UNKNOWN_Q_NAME	2288	X'000008F0'
MQRC_SERVICE_ERROR	2289	X'000008F1'
MQRC_Q_ALREADY_EXISTS	2290	X'000008F2'
MQRC_USER_ID_NOT_AVAILABLE	2291	X'000008F3'
MQRC_UNKNOWN_ENTITY	2292	X'000008F4'
MQRC_UNKNOWN_AUTH_ENTITY	2293	X'000008F5'
MQRC_UNKNOWN_REF_OBJECT	2294	X'000008F6'
MQRC_CHANNEL_ACTIVATED	2295	X'000008F7'
MQRC_CHANNEL_NOT_ACTIVATED	2296	X'000008F8'
MQRC_UOW_CANCELED	2297	X'000008F9'
MQRC_FUNCTION_NOT_SUPPORTED	2298	X'000008FA'
MQRC_SELECTOR_TYPE_ERROR	2299	X'000008FB'
MQRC_COMMAND_TYPE_ERROR	2300	X'000008FC'
MQRC_MULTIPLE_INSTANCE_ERROR	2301	X'000008FD'
MQRC_SYSTEM_ITEM_NOT_ALTERABLE	2302	X'000008FE'
MQRC_BAG_CONVERSION_ERROR	2303	X'000008FF'
MQRC_SELECTOR_OUT_OF_RANGE	2304	X'00000900'
MQRC_SELECTOR_NOT_UNIQUE	2305	X'00000901'
MQRC_INDEX_NOT_PRESENT	2306	X'00000902'
MQRC_STRING_ERROR	2307	X'00000903'
MQRC_ENCODING_NOT_SUPPORTED	2308	X'00000904'
MQRC_SELECTOR_NOT_PRESENT	2309	X'00000905'
MQRC_OUT_SELECTOR_ERROR	2310	X'00000906'
MQRC_STRING_TRUNCATED	2311	X'00000907'
MQRC_SELECTOR_WRONG_TYPE	2312	X'00000908'
MQRC_INCONSISTENT_ITEM_TYPE	2313	X'00000909'
MQRC_INDEX_ERROR	2314	X'0000090A'
MQRC_SYSTEM_BAG_NOT_ALTERABLE	2315	X'0000090B'
MQRC_ITEM_COUNT_ERROR	2316	X'0000090C'

表 301. 定数の値 (続き)		
名前	10 進数値	16 進値
MQRC_FORMAT_NOT_SUPPORTED	2317	X'0000090D'
MQRC_SELECTOR_NOT_SUPPORTED	2318	X'0000090E'
MQRC_ITEM_VALUE_ERROR	2319	X'0000090F'
MQRC_HBAG_ERROR	2320	X'00000910'
MQRC_PARAMETER_MISSING	2321	X'00000911'
MQRC_CMD_SERVER_NOT_AVAILABLE	2322	X'00000912'
MQRC_STRING_LENGTH_ERROR	2323	X'00000913'
MQRC_INQUIRY_COMMAND_ERROR	2324	X'00000914'
MQRC_NESTED_BAG_NOT_SUPPORTED	2325	X'00000915'
MQRC_BAG_WRONG_TYPE	2326	X'00000916'
MQRC_ITEM_TYPE_ERROR	2327	X'00000917'
MQRC_SYSTEM_BAG_NOT_DELETABLE	2328	X'00000918'
MQRC_SYSTEM_ITEM_NOT_DELETABLE	2329	X'00000919'
MQRC_CODED_CHAR_SET_ID_ERROR	2330	X'0000091A'
MQRC_MSG_TOKEN_ERROR	2331	X'0000091B'
MQRC_MISSING_WIH	2332	X'0000091C'
MQRC_WIH_ERROR	2333	X'0000091D'
MQRC_RFH_ERROR	2334	X'0000091E'
MQRC_RFH_STRING_ERROR	2335	X'0000091F'
MQRC_RFH_COMMAND_ERROR	2336	X'00000920'
MQRC_RFH_PARM_ERROR	2337	X'00000921'
MQRC_RFH_DUPLICATE_PARM	2338	X'00000922'
MQRC_RFH_PARM_MISSING	2339	X'00000923'
MQRC_CHAR_CONVERSION_ERROR	2340	X'00000924'
MQRC_UCS2_CONVERSION_ERROR	2341	X'00000925'
MQRC_DB2_NOT_AVAILABLE	2342	X'00000926'
MQRC_OBJECT_NOT_UNIQUE	2343	X'00000927'
MQRC_CONN_TAG_NOT_RELEASED	2344	X'00000928'
MQRC_CF_NOT_AVAILABLE	2345	X'00000929'
MQRC_CF_STRUC_IN_USE	2346	X'0000092A'
MQRC_CF_STRUC_LIST_HDR_IN_USE	2347	X'0000092B'
MQRC_CF_STRUC_AUTH_FAILED	2348	X'0000092C'
MQRC_CF_STRUC_ERROR	2349	X'0000092D'
MQRC_CONN_TAG_NOT_USABLE	2350	X'0000092E'
MQRC_GLOBAL_UOW_CONFLICT	2351	X'0000092F'
MQRC_LOCAL_UOW_CONFLICT	2352	X'00000930'
MQRC_HANDLE_IN_USE_FOR_UOW	2353	X'00000931'
MQRC_UOW_ENLISTMENT_ERROR	2354	X'00000932'
MQRC_UOW_MIX_NOT_SUPPORTED	2355	X'00000933'

表 301. 定数の値 (続き)		
名前	10 進数値	16 進値
MQRC_WXP_ERROR	2356	X'00000934'
MQRC_CURRENT_RECORD_ERROR	2357	X'00000935'
MQRC_NEXT_OFFSET_ERROR	2358	X'00000936'
MQRC_NO_RECORD_AVAILABLE	2359	X'00000937'
MQRC_OBJECT_LEVEL_INCOMPATIBLE	2360	X'00000938'
MQRC_NEXT_RECORD_ERROR	2361	X'00000939'
MQRC_BACKOUT_THRESHOLD_REACHED	2362	X'0000093A'
MQRC_MSG_NOT_MATCHED	2363	X'0000093B'
MQRC_JMS_FORMAT_ERROR	2364	X'0000093C'
MQRC_SEGMENTS_NOT_SUPPORTED	2365	X'0000093D'
MQRC_WRONG_CF_LEVEL	2366	X'0000093E'
MQRC_CONFIG_CREATE_OBJECT	2367	X'0000093F'
MQRC_CONFIG_CHANGE_OBJECT	2368	X'00000940'
MQRC_CONFIG_DELETE_OBJECT	2369	X'00000941'
MQRC_CONFIG_REFRESH_OBJECT	2370	X'00000942'
MQRC_CHANNEL_SSL_ERROR	2371	X'00000943'
MQRC_PARTICIPANT_NOT_DEFINED	2372	X'00000944'
MQRC_CF_STRUC_FAILED	2373	X'00000945'
MQRC_API_EXIT_ERROR	2374	X'00000946'
MQRC_API_EXIT_INIT_ERROR	2375	X'00000947'
MQRC_API_EXIT_TERM_ERROR	2376	X'00000948'
MQRC_EXIT_REASON_ERROR	2377	X'00000949'
MQRC_RESERVED_VALUE_ERROR	2378	X'0000094A'
MQRC_NO_DATA_AVAILABLE	2379	X'0000094B'
MQRC_SCO_ERROR	2380	X'0000094C'
MQRC_KEY_REPOSITORY_ERROR	2381	X'0000094D'
MQRC_CRYPTO_HARDWARE_ERROR	2382	X'0000094E'
MQRC_AUTH_INFO_REC_COUNT_ERROR	2383	X'0000094F'
MQRC_AUTH_INFO_REC_ERROR	2384	X'00000950'
MQRC_AIR_ERROR	2385	X'00000951'
MQRC_AUTH_INFO_TYPE_ERROR	2386	X'00000952'
MQRC_AUTH_INFO_CONN_NAME_ERROR	2387	X'00000953'
MQRC_LDAP_USER_NAME_ERROR	2388	X'00000954'
MQRC_LDAP_USER_NAME_LENGTH_ERR	2389	X'00000955'
MQRC_LDAP_PASSWORD_ERROR	2390	X'00000956'
MQRC_SSL_ALREADY_INITIALIZED	2391	X'00000957'
MQRC_SSL_CONFIG_ERROR	2392	X'00000958'
MQRC_SSL_INITIALIZATION_ERROR	2393	X'00000959'
MQRC_Q_INDEX_TYPE_ERROR	2394	X'0000095A'

表 301. 定数の値 (続き)		
名前	10 進数値	16 進値
MQRC_CFBS_ERROR	2395	X'0000095B'
MQRC_SSL_NOT_ALLOWED	2396	X'0000095C'
MQRC_JSSE_ERROR	2397	X'0000095D'
MQRC_SSL_PEER_NAME_MISMATCH	2398	X'0000095E'
MQRC_SSL_PEER_NAME_ERROR	2399	X'0000095F'
MQRC_UNSUPPORTED_CIPHER_SUITE	2400	X'00000960'
MQRC_SSL_CERTIFICATE_REVOKED	2401	X'00000961'
MQRC_SSL_CERT_STORE_ERROR	2402	X'00000962'
MQRC_CLIENT_EXIT_LOAD_ERROR	2406	X'00000966'
MQRC_CLIENT_EXIT_ERROR	2407	X'00000967'
MQRC_UOW_COMMITTED	2408	X'00000968'
MQRC_SSL_KEY_RESET_ERROR	2409	X'00000969'
MQRC_UNKNOWN_COMPONENT_NAME	2410	X'0000096A'
MQRC_LOGGER_STATUS	2411	X'0000096B'
MQRC_COMMAND_MQSC	2412	X'0000096C'
MQRC_COMMAND_PCF	2413	X'0000096D'
MQRC_CFIF_ERROR	2414	X'0000096E'
MQRC_CFSF_ERROR	2415	X'0000096F'
MQRC_CFGR_ERROR	2416	X'00000970'
MQRC_MSG_NOT_ALLOWED_IN_GROUP	2417	X'00000971'
MQRC_FILTER_OPERATOR_ERROR	2418	X'00000972'
MQRC_NESTED_SELECTOR_ERROR	2419	X'00000973'
MQRC_EPH_ERROR	2420	X'00000974'
MQRC_RFH_FORMAT_ERROR	2421	X'00000975'
MQRC_CFBF_ERROR	2422	X'00000976'
MQRC_CLIENT_CHANNEL_CONFLICT	2423	X'00000977'
MQRC_SD_ERROR	2424	X'00000978'
MQRC_TOPIC_STRING_ERROR	2425	X'00000979'
MQRC_STS_ERROR	2426	X'0000097A'
MQRC_NO_SUBSCRIPTION	2428	X'0000097C'
MQRC_SUBSCRIPTION_IN_USE	2429	X'0000097D'
MQRC_STAT_TYPE_ERROR	2430	X'0000097E'
MQRC_SUB_USER_DATA_ERROR	2431	X'0000097F'
MQRC_SUB_ALREADY_EXISTS	2432	X'00000980'
MQRC_IDENTITY_MISMATCH	2434	X'00000982'
MQRC_ALTER_SUB_ERROR	2435	X'00000983'
MQRC_DURABILITY_NOT_ALLOWED	2436	X'00000984'
MQRC_NO_RETAINED_MSG	2437	X'00000985'
MQRC_SRO_ERROR	2438	X'00000986'

表 301. 定数の値 (続き)		
名前	10 進数値	16 進値
MQRC_SUB_NAME_ERROR	2440	X'00000988'
MQRC_OBJECT_STRING_ERROR	2441	X'00000989'
MQRC_PROPERTY_NAME_ERROR	2442	X'0000098A'
MQRC_SEGMENTATION_NOT_ALLOWED	2443	X'0000098B'
MQRC_CBD_ERROR	2444	X'0000098C'
MQRC_CTLO_ERROR	2445	X'0000098D'
MQRC_NO_CALLBACKS_ACTIVE	2446	X'0000098E'
MQRC_CALLBACK_NOT_REGISTERED	2448	X'00000990'
MQRC_OPTIONS_CHANGED	2457	X'00000999'
MQRC_READ_AHEAD_MSGS	2458	X'0000099A'
MQRC_SELECTOR_SYNTAX_ERROR	2459	X'0000099B'
MQRC_HMSG_ERROR	2460	X'0000099C'
MQRC_CMHO_ERROR	2461	X'0000099D'
MQRC_DMHO_ERROR	2462	X'0000099E'
MQRC_SMPO_ERROR	2463	X'0000099F'
MQRC_IMPO_ERROR	2464	X'000009A0'
MQRC_PROPERTY_NAME_TOO_BIG	2465	X'000009A1'
MQRC_PROP_VALUE_NOT_CONVERTED	2466	X'000009A2'
MQRC_PROP_TYPE_NOT_SUPPORTED	2467	X'000009A3'
MQRC_PROPERTY_VALUE_TOO_BIG	2469	X'000009A5'
MQRC_PROP_CONV_NOT_SUPPORTED	2470	X'000009A6'
MQRC_PROPERTY_NOT_AVAILABLE	2471	X'000009A7'
MQRC_PROP_NUMBER_FORMAT_ERROR	2472	X'000009A8'
MQRC_PROPERTY_TYPE_ERROR	2473	X'000009A9'
MQRC_PROPERTIES_TOO_BIG	2478	X'000009AE'
MQRC_PUT_NOT_RETAINED	2479	X'000009AF'
MQRC_ALIAS_TARGTYPE_CHANGED	2480	X'000009B0'
MQRC_DMPO_ERROR	2481	X'000009B1'
MQRC_PD_ERROR	2482	X'000009B2'
MQRC_CALLBACK_TYPE_ERROR	2483	X'000009B3'
MQRC_CBD_OPTIONS_ERROR	2484	X'000009B4'
MQRC_MAX_MSG_LENGTH_ERROR	2485	X'000009B5'
MQRC_CALLBACK_ROUTINE_ERROR	2486	X'000009B6'
MQRC_CALLBACK_LINK_ERROR	2487	X'000009B7'
MQRC_OPERATION_ERROR	2488	X'000009B8'
MQRC_BMHO_ERROR	2489	X'000009B9'
MQRC_UNSUPPORTED_PROPERTY	2490	X'000009BA'
MQRC_PROP_NAME_NOT_CONVERTED	2492	X'000009BC'
MQRC_GET_ENABLED	2494	X'000009BE'

表 301. 定数の値 (続き)		
名前	10 進数値	16 進値
MQRC_MODULE_NOT_FOUND	2495	X'000009BF'
MQRC_MODULE_INVALID	2496	X'000009C0'
MQRC_MODULE_ENTRY_NOT_FOUND	2497	X'000009C1'
MQRC_MIXED_CONTENT_NOT_ALLOWED	2498	X'000009C2'
MQRC_MSG_HANDLE_IN_USE	2499	X'000009C3'
MQRC_HCONN_ASYNC_ACTIVE	2500	X'000009C4'
MQRC_MHBO_ERROR	2501	X'000009C5'
MQRC_PUBLICATION_FAILURE	2502	X'000009C6'
MQRC_SUB_INHIBITED	2503	X'000009C7'
MQRC_SELECTOR_ALWAYS_FALSE	2504	X'000009C8'
MQRC_XEPO_ERROR	2507	X'000009CB'
MQRC_DURABILITY_NOT_ALTERABLE	2509	X'000009CD'
MQRC_TOPIC_NOT_ALTERABLE	2510	X'000009CE'
MQRC_SUBLEVEL_NOT_ALTERABLE	2512	X'000009D0'
MQRC_PROPERTY_NAME_LENGTH_ERR	2513	X'000009D1'
MQRC_DUPLICATE_GROUP_SUB	2514	X'000009D2'
MQRC_GROUPING_NOT_ALTERABLE	2515	X'000009D3'
MQRC_SELECTOR_INVALID_FOR_TYPE	2516	X'000009D4'
MQRC_HOBJ QUIESCED	2517	X'000009D5'
MQRC_HOBJ QUIESCED_NO_MSGS	2518	X'000009D6'
MQRC_SELECTION_STRING_ERROR	2519	X'000009D7'
MQRC_RES_OBJECT_STRING_ERROR	2520	X'000009D8'
MQRC_CONNECTION_SUSPENDED	2521	X'000009D9'
MQRC_INVALID_DESTINATION	2522	X'000009DA'
MQRC_INVALID_SUBSCRIPTION	2523	X'000009DB'
MQRC_SELECTOR_NOT_ALTERABLE	2524	X'000009DC'
MQRC_RETAINED_MSG_Q_ERROR	2525	X'000009DD'
MQRC_RETAINED_NOT_DELIVERED	2526	X'000009DE'
MQRC_RFH_RESTRICTED_FORMAT_ERR	2527	X'000009DF'
MQRC_CONNECTION_STOPPED	2528	X'000009E0'
MQRC_ASYNC_UOW_CONFLICT	2529	X'000009E1'
MQRC_ASYNC_XA_CONFLICT	2530	X'000009E2'
MQRC_PUBSUB_INHIBITED	2531	X'000009E3'
MQRC_MSG_HANDLE_COPY_FAILURE	2532	X'000009E4'
MQRC_DEST_CLASS_NOT_ALTERABLE	2533	X'000009E5'
MQRC_OPERATION_NOT_ALLOWED	2534	X'000009E6'
MQRC_ACTION_ERROR	2535	X'000009E7'
MQRC_CHANNEL_NOT_AVAILABLE	2537	X'000009E9'
MQRC_HOST_NOT_AVAILABLE	2538	X'000009EA'

表 301. 定数の値 (続き)		
名前	10 進数値	16 進値
MQRC_CHANNEL_CONFIG_ERROR	2539	X'000009EB'
MQRC_UNKNOWN_CHANNEL_NAME	2540	X'000009EC'
MQRC_LOOPING_PUBLICATION	2541	X'000009ED'
MQRC_ALREADY_JOINED	2542	X'000009EE'
MQRC_スタンバイ_キュー_マネージャ	2543	X'000009EF'
MQRC_RECONNECTING	2544	X'000009F0'
MQRC_RECONNECTED	2545	X'000009F1'
MQRC_RECONNECT_QMID_MISMATCH	2546	X'000009F2'
MQRC_RECONNECT_INCOMPATIBLE	2547	X'000009F3'
MQRC_RECONNECT_FAILED	2548	X'000009F4'
MQRC_CALL_INTERRUPTED	2549	X'000009F5'
MQRC_一致するサブスクリプションなし	2550	X'000009F6'
MQRC_SELECTION_NOT_AVAILABLE	2551	X'000009F7'
MQRC_CHANNEL_SSL_WARNING	2552	X'000009F8'
MQRC_OCSP_URL_ERROR	2553	X'000009F9'
MQRC_CONTENT_ERROR	2554	X'000009FA'
MQRC_RECONNECT_Q_MGR_REQD	2555	X'000009FB'
MQRC_再接続タイムアウト	2556	X'000009FC'
MQRC_PUBLISH_EXIT_ERROR	2557	X'000009FD'
MQRC_COMMINFO_ERROR	2558	X'000009FE'
MQRC_DEF_SYNCPOINT_INHIBITED	2559	X'000009FF'
MQRC_MULTICAST_ONLY	2560	X'00000A00'
MQRC_データセット利用不可	2561	X'00000A01'
MQRC_グループ化が許可されていません	2562	X'00000A02'
MQRC_グループアドレスエラー	2563	X'00000A03'
MQRC_MULTICAST_CONFIG_ERROR	2564	X'00000A04'
MQRC_マルチキャストインターフェースエラー	2565	X'00000A05'
MQRC_MULTICAST_SEND_ERROR	2566	X'00000A06'
MQRC_MULTICAST_INTERNAL_ERROR	2567	X'00000A07'
MQRC_CONNECTION_NOT_AVAILABLE	2568	X'00000A08'
MQRC_SYNCPOINT_NOT_ALLOWED	2569	X'00000A09'
MQRC_SSL_代替プロバイダー必須	2570	X'00000A0A'
MQRC_MCAST_PUB_STATUS	2571	X'00000A0B'
MQRC_MCAST_サブステータス	2572	X'00000A0C'
MQRC_PRECONN_EXIT_LOAD_ERROR	2573	X'00000A0D'
MQRC_PRECONN_EXIT_NOT_FOUND	2574	X'00000A0E'
MQRC_PRECONN_EXIT_ERROR	2575	X'00000A0F'
MQRC_CD_ARRAY_ERROR	2576	X'00000A10'
MQRC_CHANNEL_BLOCKED	2577	X'00000A11'

表 301. 定数の値 (続き)		
名前	10 進数値	16 進値
MQRC_CHANNEL_BLOCKED_WARNING	2578	X'00000A12'
MQRC_サブスクリプション作成	2579	X'00000A13'
MQRC_サブスクリプション削除	2580	X'00000A14'
MQRC_サブスクリプション変更	2581	X'00000A15'
MQRC_サブスクリプション更新	2582	X'00000A16'
MQRC_INSTALLATION_MISMATCH	2583	X'00000A17'
MQRC_NOT_PRIVILEGED	2584	X'00000A18'
MQRC_プロパティ無効	2586	X'00000A1A'
MQRC_HMSG_利用不可	2587	X'00000A1B'
MQRC_EXIT_PROPS_NOT_SUPPORTED	2588	X'00000A1C'
MQRC_INSTALLATION_MISSING	2589	X'00000A1D'
MQRC_FASTPATH_利用不可	2590	X'00000A1E'
MQRC_CIPHER_SPEC_NOT_SUITE_B	2591	X'00000A1F'
MQRC_SUITE_B_ERROR	2592	X'00000A20'
MQRC_CERT_VAL_POLICY_ERROR	2593	X'00000A21'
MQRC_PASSWORD_PROTECTION_ERROR	2594	X'00000A22'
MQRC_CSP_ERROR	2595	X'00000A23'
MQRC_CERT_LABEL_NOT_ALLOWED	2596	X'00000A24'
MQRC_ADMIN_TOPIC_STRING_ERROR	2598	X'00000A26'
MQRC_AMQP_利用不可	2599	X'00000A27'
MQRC_CCDT_URL_ERROR	2600	X'00000A28'
MQRC_Q_MGR_RECONNECT_REQUESTED	2601	X'00000A29'
MQRC_BNO_ERROR	2602	X'00000A2A'
MQRC_OUTBOUND_SNI_NOT_VALID	2603	X'00000A2B'
MQRC_HTTPS_キーストアエラー	2604	X'00000A2C'
MQRC_REOPEN_EXCL_INPUT_ERROR	6100	X'000017D4'
MQRC_REOPEN_INQUIRE_ERROR	6101	X'000017D5'
MQRC_REOPEN_SAVED_CONTEXT_ERR	6102	X'000017D6'
MQRC_REOPEN_TEMPORARY_Q_ERROR	6103	X'000017D7'
MQRC_ATTRIBUTE_LOCKED	6104	X'000017D8'
MQRC_CURSOR_NOT_VALID	6105	X'000017D9'
MQRC_ENCODING_ERROR	6106	X'000017DA'
MQRC_STRUC_ID_ERROR	6107	X'000017DB'
MQRC_NULL_POINTER	6108	X'000017DC'
MQRC_NO_CONNECTION_REFERENCE	6109	X'000017DD'
MQRC_NO_BUFFER	6110	X'000017DE'
MQRC_BINARY_DATA_LENGTH_ERROR	6111	X'000017DF'
MQRC_BUFFER_NOT_AUTOMATIC	6112	X'000017E0'
MQRC_INSUFFICIENT_BUFFER	6113	X'000017E1'

表 301. 定数の値 (続き)		
名前	10 進数値	16 進値
MQRC_INSUFFICIENT_DATA	6114	X'000017E2'
MQRC_DATA_TRUNCATED	6115	X'000017E3'
MQRC_ZERO_LENGTH	6116	X'000017E4'
MQRC_NEGATIVE_LENGTH	6117	X'000017E5'
MQRC_NEGATIVE_OFFSET	6118	X'000017E6'
MQRC_INCONSISTENT_FORMAT	6119	X'000017E7'
MQRC_INCONSISTENT_OBJECT_STATE	6120	X'000017E8'
MQRC_CONTEXT_OBJECT_NOT_VALID	6121	X'000017E9'
MQRC_CONTEXT_OPEN_ERROR	6122	X'000017EA'
MQRC_STRUC_LENGTH_ERROR	6123	X'000017EB'
MQRC_NOT_CONNECTED	6124	X'000017EC'
MQRC_NOT_OPEN	6125	X'000017ED'
MQRC_DISTRIBUTION_LIST_EMPTY	6126	X'000017EE'
MQRC_INCONSISTENT_OPEN_OPTIONS	6127	X'000017EF'
MQRC_WRONG_VERSION	6128	X'000017F0'
MQRC_REFERENCE_ERROR	6129	X'000017F1'
MQRC_XR_利用不可	6130	X'000017F2'
MQRC_SUB_JOIN_NOT_ALTERABLE	29440	X'00007200'

MQRCCF_* (コマンド形式のヘッダー理由コード)

プログラマ応答の詳細については、[PCF 理由コード](#)を参照してください。

表 302. 定数の値		
名前	10 進数値	16 進値
MQRCCF_CFH_TYPE_ERROR	3001	X'00000BB9'
MQRCCF_CFH_LENGTH_ERROR	3002	X'00000BBA'
MQRCCF_CFH_VERSION_ERROR	3003	X'00000BBB'
MQRCCF_CFH_MSG_SEQ_NUMBER_ERR	3004	X'00000BBC'
MQRCCF_CFH_CONTROL_ERROR	3005	X'00000BBD'
MQRCCF_CFH_PARM_COUNT_ERROR	3006	X'00000BBE'
MQRCCF_CFH_COMMAND_ERROR	3007	X'00000BBF'
MQRCCF_COMMAND_FAILED	3008	X'00000BC0'
MQRCCF_CFIN_LENGTH_ERROR	3009	X'00000BC1'
MQRCCF_CFST_LENGTH_ERROR	3010	X'00000BC2'
MQRCCF_CFST_STRING_LENGTH_ERR	3011	X'00000BC3'
MQRCCF_FORCE_VALUE_ERROR	3012	X'00000BC4'
MQRCCF_STRUCTURE_TYPE_ERROR	3013	X'00000BC5'
MQRCCF_CFIN_PARM_ID_ERROR	3014	X'00000BC6'
MQRCCF_CFST_PARM_ID_ERROR	3015	X'00000BC7'
MQRCCF_MSG_LENGTH_ERROR	3016	X'00000BC8'

表 302. 定数の値 (続き)		
名前	10 進数値	16 進値
MQRCCF_CFIN_DUPLICATE_PARM	3017	X'00000BC9'
MQRCCF_CFST_DUPLICATE_PARM	3018	X'00000BCA'
MQRCCF_PARM_COUNT_TOO_SMALL	3019	X'00000BCB'
MQRCCF_PARM_COUNT_TOO_BIG	3020	X'00000BCC'
MQRCCF_Q_ALREADY_IN_CELL	3021	X'00000BCD'
MQRCCF_Q_TYPE_ERROR	3022	X'00000BCE'
MQRCCF_MD_FORMAT_ERROR	3023	X'00000BCF'
MQRCCF_CFSL_LENGTH_ERROR	3024	X'00000BD0'
MQRCCF_REPLACE_VALUE_ERROR	3025	X'00000BD1'
MQRCCF_CFIL_DUPLICATE_VALUE	3026	X'00000BD2'
MQRCCF_CFIL_COUNT_ERROR	3027	X'00000BD3'
MQRCCF_CFIL_LENGTH_ERROR	3028	X'00000BD4'
MQRCCF_QUIESCE_VALUE_ERROR	3029	X'00000BD5'
MQRCCF_MODE_VALUE_ERROR	3029	X'00000BD5'
MQRCCF_MSG_SEQ_NUMBER_ERROR	3030	X'00000BD6'
MQRCCF_PING_DATA_COUNT_ERROR	3031	X'00000BD7'
MQRCCF_PING_DATA_COMPARE_ERROR	3032	X'00000BD8'
MQRCCF_CFSL_PARM_ID_ERROR	3033	X'00000BD9'
MQRCCF_CHANNEL_TYPE_ERROR	3034	X'00000BDA'
MQRCCF_PARM_SEQUENCE_ERROR	3035	X'00000BDB'
MQRCCF_XMIT_PROTOCOL_TYPE_ERR	3036	X'00000BDC'
MQRCCF_BATCH_SIZE_ERROR	3037	X'00000BDD'
MQRCCF_DISC_INT_ERROR	3038	X'00000BDE'
MQRCCF_SHORT_RETRY_ERROR	3039	X'00000BDF'
MQRCCF_SHORT_TIMER_ERROR	3040	X'00000BE0'
MQRCCF_LONG_RETRY_ERROR	3041	X'00000BE1'
MQRCCF_LONG_TIMER_ERROR	3042	X'00000BE2'
MQRCCF_SEQ_NUMBER_WRAP_ERROR	3043	X'00000BE3'
MQRCCF_MAX_MSG_LENGTH_ERROR	3044	X'00000BE4'
MQRCCF_PUT_AUTH_ERROR	3045	X'00000BE5'
MQRCCF_PURGE_VALUE_ERROR	3046	X'00000BE6'
MQRCCF_CFIL_PARM_ID_ERROR	3047	X'00000BE7'
MQRCCF_MSG_TRUNCATED	3048	X'00000BE8'
MQRCCF_CCSID_ERROR	3049	X'00000BE9'
MQRCCF_ENCODING_ERROR	3050	X'00000BEA'
MQRCCF_QUEUES_VALUE_ERROR	3051	X'00000BEB'
MQRCCF_DATA_CONV_VALUE_ERROR	3052	X'00000BEC'
MQRCCF_INDOUBT_VALUE_ERROR	3053	X'00000BED'
MQRCCF_ESCAPE_TYPE_ERROR	3054	X'00000BEE'

表 302. 定数の値 (続き)		
名前	10 進数値	16 進値
MQRCCF_REPOS_VALUE_ERROR	3055	X'00000BEF'
MQRCCF_CHANNEL_TABLE_ERROR	3062	X'00000BF6'
MQRCCF_MCA_TYPE_ERROR	3063	X'00000BF7'
MQRCCF_CHL_INST_TYPE_ERROR	3064	X'00000BF8'
MQRCCF_CHL_STATUS_NOT_FOUND	3065	X'00000BF9'
MQRCCF_CFSL_DUPLICATE_PARM	3066	X'00000BFA'
MQRCCF_CFSL_TOTAL_LENGTH_ERROR	3067	X'00000BFB'
MQRCCF_CFSL_COUNT_ERROR	3068	X'00000BFC'
MQRCCF_CFSL_STRING_LENGTH_ERR	3069	X'00000BFD'
MQRCCF_BROKER_DELETED	3070	X'00000BFE'
MQRCCF_STREAM_ERROR	3071	X'00000BFF'
MQRCCF_TOPIC_ERROR	3072	X'00000C00'
MQRCCF_NOT_REGISTERED	3073	X'00000C01'
MQRCCF_Q_MGR_NAME_ERROR	3074	X'00000C02'
MQRCCF_INCORRECT_STREAM	3075	X'00000C03'
MQRCCF_Q_NAME_ERROR	3076	X'00000C04'
MQRCCF_NO_RETAINED_MSG	3077	X'00000C05'
MQRCCF_DUPLICATE_IDENTITY	3078	X'00000C06'
MQRCCF_INCORRECT_Q	3079	X'00000C07'
MQRCCF_CORREL_ID_ERROR	3080	X'00000C08'
MQRCCF_NOT_AUTHORIZED	3081	X'00000C09'
MQRCCF_UNKNOWN_STREAM	3082	X'00000C0A'
MQRCCF_REG_OPTIONS_ERROR	3083	X'00000C0B'
MQRCCF_PUB_OPTIONS_ERROR	3084	X'00000C0C'
MQRCCF_UNKNOWN_BROKER	3085	X'00000C0D'
MQRCCF_Q_MGR_CCSID_ERROR	3086	X'00000C0E'
MQRCCF_DEL_OPTIONS_ERROR	3087	X'00000C0F'
MQRCCF_CLUSTER_NAME_CONFLICT	3088	X'00000C10'
MQRCCF_REPOS_NAME_CONFLICT	3089	X'00000C11'
MQRCCF_CLUSTER_Q_USAGE_ERROR	3090	X'00000C12'
MQRCCF_ACTION_VALUE_ERROR	3091	X'00000C13'
MQRCCF_COMMS_LIBRARY_ERROR	3092	X'00000C14'
MQRCCF_NETBIOS_NAME_ERROR	3093	X'00000C15'
MQRCCF_BROKER_COMMAND_FAILED	3094	X'00000C16'
MQRCCF_CFST_CONFLICTING_PARM	3095	X'00000C17'
MQRCCF_PATH_NOT_VALID	3096	X'00000C18'
MQRCCF_PARM_SYNTAX_ERROR	3097	X'00000C19'
MQRCCF_PWD_LENGTH_ERROR	3098	X'00000C1A'
MQRCCF_FILTER_ERROR	3150	X'00000C4E'

表 302. 定数の値 (続き)		
名前	10 進数値	16 進値
MQRCCF_WRONG_USER	3151	X'00000C4F'
MQRCCF_DUPLICATE_SUBSCRIPTION	3152	X'00000C50'
MQRCCF_SUB_NAME_ERROR	3153	X'00000C51'
MQRCCF_SUB_IDENTITY_ERROR	3154	X'00000C52'
MQRCCF_SUBSCRIPTION_IN_USE	3155	X'00000C53'
MQRCCF_SUBSCRIPTION_LOCKED	3156	X'00000C54'
MQRCCF_ALREADY_JOINED	3157	X'00000C55'
MQRCCF_OBJECT_IN_USE	3160	X'00000C58'
MQRCCF_UNKNOWN_FILE_NAME	3161	X'00000C59'
MQRCCF_FILE_NOT_AVAILABLE	3162	X'00000C5A'
MQRCCF_DISC_RETRY_ERROR	3163	X'00000C5B'
MQRCCF_ALLOC_RETRY_ERROR	3164	X'00000C5C'
MQRCCF_ALLOC_SLOW_TIMER_ERROR	3165	X'00000C5D'
MQRCCF_ALLOC_FAST_TIMER_ERROR	3166	X'00000C5E'
MQRCCF_PORT_NUMBER_ERROR	3167	X'00000C5F'
MQRCCF_CHL_SYSTEM_NOT_ACTIVE	3168	X'00000C60'
MQRCCF_ENTITY_NAME_MISSING	3169	X'00000C61'
MQRCCF_PROFILE_NAME_ERROR	3170	X'00000C62'
MQRCCF_AUTH_VALUE_ERROR	3171	X'00000C63'
MQRCCF_AUTH_VALUE_MISSING	3172	X'00000C64'
MQRCCF_OBJECT_TYPE_MISSING	3173	X'00000C65'
MQRCCF_CONNECTION_ID_ERROR	3174	X'00000C66'
MQRCCF_LOG_TYPE_ERROR	3175	X'00000C67'
MQRCCF_PROGRAM_NOT_AVAILABLE	3176	X'00000C68'
MQRCCF_PROGRAM_AUTH_FAILED	3177	X'00000C69'
MQRCCF_NONE_FOUND	3200	X'00000C80'
MQRCCF_SECURITY_SWITCH_OFF	3201	X'00000C81'
MQRCCF_SECURITY_REFRESH_FAILED	3202	X'00000C82'
MQRCCF_PARM_CONFLICT	3203	X'00000C83'
MQRCCF_COMMAND_INHIBITED	3204	X'00000C84'
MQRCCF_OBJECT_BEING_DELETED	3205	X'00000C85'
MQRCCF_STORAGE_CLASS_IN_USE	3207	X'00000C87'
MQRCCF_OBJECT_NAME_RESTRICTED	3208	X'00000C88'
MQRCCF_OBJECT_LIMIT_EXCEEDED	3209	X'00000C89'
MQRCCF_OBJECT_OPEN_FORCE	3210	X'00000C8A'
MQRCCF_DISPOSITION_CONFLICT	3211	X'00000C8B'
MQRCCF_Q_MGR_NOT_IN_QSG	3212	X'00000C8C'
MQRCCF_ATTR_VALUE_FIXED	3213	X'00000C8D'
MQRCCF_NAMELIST_ERROR	3215	X'00000C8F'

表 302. 定数の値 (続き)		
名前	10 進数値	16 進値
MQRCCF_NO_CHANNEL_INITIATOR	3217	X'00000C91'
MQRCCF_CHANNEL_INITIATOR_ERROR	3218	X'00000C92'
MQRCCF_COMMAND_LEVEL_CONFLICT	3222	X'00000C96'
MQRCCF_Q_ATTR_CONFLICT	3223	X'00000C97'
MQRCCF_EVENTS_DISABLED	3224	X'00000C98'
MQRCCF_COMMAND_SCOPE_ERROR	3225	X'00000C99'
MQRCCF_COMMAND_REPLY_ERROR	3226	X'00000C9A'
MQRCCF_FUNCTION_RESTRICTED	3227	X'00000C9B'
MQRCCF_PARM_MISSING	3228	X'00000C9C'
MQRCCF_PARM_VALUE_ERROR	3229	X'00000C9D'
MQRCCF_COMMAND_LENGTH_ERROR	3230	X'00000C9E'
MQRCCF_COMMAND_ORIGIN_ERROR	3231	X'00000C9F'
MQRCCF_LISTENER_CONFLICT	3232	X'00000CA0'
MQRCCF_LISTENER_STARTED	3233	X'00000CA1'
MQRCCF_LISTENER_STOPPED	3234	X'00000CA2'
MQRCCF_CHANNEL_ERROR	3235	X'00000CA3'
MQRCCF_CF_STRUC_ERROR	3236	X'00000CA4'
MQRCCF_UNKNOWN_USER_ID	3237	X'00000CA5'
MQRCCF_UNEXPECTED_ERROR	3238	X'00000CA6'
MQRCCF_NO_XCF_PARTNER	3239	X'00000CA7'
MQRCCF_CFGR_PARM_ID_ERROR	3240	X'00000CA8'
MQRCCF_CFIF_LENGTH_ERROR	3241	X'00000CA9'
MQRCCF_CFIF_OPERATOR_ERROR	3242	X'00000CAA'
MQRCCF_CFIF_PARM_ID_ERROR	3243	X'00000CAB'
MQRCCF_CFSF_FILTER_VAL_LEN_ERR	3244	X'00000CAC'
MQRCCF_CFSF_LENGTH_ERROR	3245	X'00000CAD'
MQRCCF_CFSF_OPERATOR_ERROR	3246	X'00000CAE'
MQRCCF_CFSF_PARM_ID_ERROR	3247	X'00000CAF'
MQRCCF_TOO_MANY_FILTERS	3248	X'00000CB0'
MQRCCF_LISTENER_RUNNING	3249	X'00000CB1'
MQRCCF_LSTR_STATUS_NOT_FOUND	3250	X'00000CB2'
MQRCCF_SERVICE_RUNNING	3251	X'00000CB3'
MQRCCF_SERV_STATUS_NOT_FOUND	3252	X'00000CB4'
MQRCCF_SERVICE_STOPPED	3253	X'00000CB5'
MQRCCF_CFBS_DUPLICATE_PARM	3254	X'00000CB6'
MQRCCF_CFBS_LENGTH_ERROR	3255	X'00000CB7'
MQRCCF_CFBS_PARM_ID_ERROR	3256	X'00000CB8'
MQRCCF_CFBS_STRING_LENGTH_ERR	3257	X'00000CB9'
MQRCCF_CFGR_LENGTH_ERROR	3258	X'00000CBA'

表 302. 定数の値 (続き)		
名前	10 進数値	16 進値
MQRCCF_CFGR_PARM_COUNT_ERROR	3259	X'00000CBB'
MQRCCF_CONN_NOT_STOPPED	3260	X'00000CBC'
MQRCCF_SERVICE_REQUEST_PENDING	3261	X'00000CBD'
MQRCCF_NO_START_CMD	3262	X'00000CBE'
MQRCCF_NO_STOP_CMD	3263	X'00000CBF'
MQRCCF_CFBF_LENGTH_ERROR	3264	X'00000CC0'
MQRCCF_CFBF_PARM_ID_ERROR	3265	X'00000CC1'
MQRCCF_CFBF_OPERATOR_ERROR	3266	X'00000CC2'
MQRCCF_CFBF_FILTER_VAL_LEN_ERR	3267	X'00000CC3'
MQRCCF_LISTENER_STILL_ACTIVE	3268	X'00000CC4'
MQRCCF_DEF_XMIT_Q_CLUS_ERROR	3269	X'00000CC5'
MQRCCF_TOPICSTR_ALREADY_EXISTS	3300	X'00000CE4'
MQRCCF_SHARING_CONVS_ERROR	3301	X'00000CE5'
MQRCCF_SHARING_CONVS_TYPE	3302	X'00000CE6'
MQRCCF_SECURITY_CASE_CONFLICT	3303	X'00000CE7'
MQRCCF_TOPIC_TYPE_ERROR	3305	X'00000CE9'
MQRCCF_MAX_INSTANCES_ERROR	3306	X'00000CEA'
MQRCCF_MAX_INSTS_PER_CLNT_ERR	3307	X'00000CEB'
MQRCCF_TOPIC_STRING_NOT_FOUND	3308	X'00000CEC'
MQRCCF_SUBSCRIPTION_POINT_ERR	3309	X'00000CED'
MQRCCF_SUB_ALREADY_EXISTS	3311	X'00000CEF'
MQRCCF_UNKNOWN_OBJECT_NAME	3312	X'00000CF0'
MQRCCF_REMOTE_Q_NAME_ERROR	3313	X'00000CF1'
MQRCCF_DURABILITY_NOT_ALLOWED	3314	X'00000CF2'
MQRCCF_HOBJ_ERROR	3315	X'00000CF3'
MQRCCF_DEST_NAME_ERROR	3316	X'00000CF4'
MQRCCF_INVALID_DESTINATION	3317	X'00000CF5'
MQRCCF_PUBSUB_INHIBITED	3318	X'00000CF6'
MQRCCF_CHLAUTH_TYPE_ERROR	3326	X'00000CFE'
MQRCCF_CHLAUTH_ACTION_ERROR	3327	X'00000CFF'
MQRCCF_CHLAUTH_USERSRC_ERROR	3335	X'00000D07'
MQRCCF_WRONG_CHLAUTH_TYPE	3336	X'00000D08'
MQRCCF_CHLAUTH_ALREADY_EXISTS	3337	X'00000D09'
MQRCCF_CHLAUTH_NOT_FOUND	3338	X'00000D0A'
MQRCCF_WRONG_CHLAUTH_ACTION	3339	X'00000D0B'
MQRCCF_WRONG_CHLAUTH_USERSRC	3340	X'00000D0C'
MQRCCF_CHLAUTH_WARN_ERROR	3341	X'00000D0D'
MQRCCF_WRONG_CHLAUTH_MATCH	3342	X'00000D0E'
MQRCCF_IPADDR_RANGE_CONFLICT	3343	X'00000D0F'

表 302. 定数の値 (続き)		
名前	10 進数値	16 進値
MQRCCF_CHLAUTH_MAX_EXCEEDED	3344	X'00000D10'
MQRCCF_IPADDR_ERROR	3345	X'00000D11'
MQRCCF_IPADDR_RANGE_ERROR	3346	X'00000D12'
MQRCCF_PROFILE_NAME_MISSING	3347	X'00000D13'
MQRCCF_CHLAUTH_CLNTUSER_ERROR	3348	X'00000D14'
MQRCCF_CHLAUTH_NAME_ERROR	3349	X'00000D15'
MQRCCF_SUITE_B_ERROR	3353	X'00000D19'
MQRCCF_PSCLUS_DISABLED_TOPDEF	3359	X'00000D1F'
MQRCCF_PSCLUS_TOPIC_EXISTS	3360	X'00000D20'
MQRCCF_INVALID_PROTOCOL	3365	X'00000D25'
MQRCCF_ACCESS_BLOCKED	3382	X'00000D36'
 MQRCCF_属性値エラー_QSG_QM	3390	X'00000D3E'
MQRCCF_OBJECT_ALREADY_EXISTS	4001	X'00000FA1'
MQRCCF_OBJECT_WRONG_TYPE	4002	X'00000FA2'
MQRCCF_LIKE_OBJECT_WRONG_TYPE	4003	X'00000FA3'
MQRCCF_OBJECT_OPEN	4004	X'00000FA4'
MQRCCF_ATTR_VALUE_ERROR	4005	X'00000FA5'
MQRCCF_UNKNOWN_Q_MGR	4006	X'00000FA6'
MQRCCF_Q_WRONG_TYPE	4007	X'00000FA7'
MQRCCF_OBJECT_NAME_ERROR	4008	X'00000FA8'
MQRCCF_ALLOCATE_FAILED	4009	X'00000FA9'
MQRCCF_HOST_NOT_AVAILABLE	4010	X'00000FAA'
MQRCCF_CONFIGURATION_ERROR	4011	X'00000FAB'
MQRCCF_CONNECTION_REFUSED	4012	X'00000FAC'
MQRCCF_ENTRY_ERROR	4013	X'00000FAD'
MQRCCF_SEND_FAILED	4014	X'00000FAE'
MQRCCF_RECEIVED_DATA_ERROR	4015	X'00000FAF'
MQRCCF_RECEIVE_FAILED	4016	X'00000FB0'
MQRCCF_CONNECTION_CLOSED	4017	X'00000FB1'
MQRCCF_NO_STORAGE	4018	X'00000FB2'
MQRCCF_NO_COMMS_MANAGER	4019	X'00000FB3'
MQRCCF_LISTENER_NOT_STARTED	4020	X'00000FB4'
MQRCCF_BIND_FAILED	4024	X'00000FB8'
MQRCCF_CHANNEL_INDOUBT	4025	X'00000FB9'
MQRCCF_MQCONN_FAILED	4026	X'00000FBA'
MQRCCF_MQOPEN_FAILED	4027	X'00000FBB'
MQRCCF_MQGET_FAILED	4028	X'00000FBC'
MQRCCF_MQPUT_FAILED	4029	X'00000FBD'
MQRCCF_PING_ERROR	4030	X'00000FBE'

表 302. 定数の値 (続き)		
名前	10 進数値	16 進値
MQRCCF_CHANNEL_IN_USE	4031	X'00000FBF'
MQRCCF_CHANNEL_NOT_FOUND	4032	X'00000FC0'
MQRCCF_UNKNOWN_REMOTE_CHANNEL	4033	X'00000FC1'
MQRCCF_REMOTE_QM_UNAVAILABLE	4034	X'00000FC2'
MQRCCF_REMOTE_QM_TERMINATING	4035	X'00000FC3'
MQRCCF_MQINQ_FAILED	4036	X'00000FC4'
MQRCCF_NOT_XMIT_Q	4037	X'00000FC5'
MQRCCF_CHANNEL_DISABLED	4038	X'00000FC6'
MQRCCF_USER_EXIT_NOT_AVAILABLE	4039	X'00000FC7'
MQRCCF_COMMIT_FAILED	4040	X'00000FC8'
MQRCCF_WRONG_CHANNEL_TYPE	4041	X'00000FC9'
MQRCCF_CHANNEL_ALREADY_EXISTS	4042	X'00000FCA'
MQRCCF_DATA_TOO_LARGE	4043	X'00000FCB'
MQRCCF_CHANNEL_NAME_ERROR	4044	X'00000FCC'
MQRCCF_XMIT_Q_NAME_ERROR	4045	X'00000FCD'
MQRCCF_MCA_NAME_ERROR	4047	X'00000FCF'
MQRCCF_SEND_EXIT_NAME_ERROR	4048	X'00000FD0'
MQRCCF_SEC_EXIT_NAME_ERROR	4049	X'00000FD1'
MQRCCF_MSG_EXIT_NAME_ERROR	4050	X'00000FD2'
MQRCCF_RCV_EXIT_NAME_ERROR	4051	X'00000FD3'
MQRCCF_XMIT_Q_NAME_WRONG_TYPE	4052	X'00000FD4'
MQRCCF_MCA_NAME_WRONG_TYPE	4053	X'00000FD5'
MQRCCF_DISC_INT_WRONG_TYPE	4054	X'00000FD6'
MQRCCF_SHORT_RETRY_WRONG_TYPE	4055	X'00000FD7'
MQRCCF_SHORT_TIMER_WRONG_TYPE	4056	X'00000FD8'
MQRCCF_LONG_RETRY_WRONG_TYPE	4057	X'00000FD9'
MQRCCF_LONG_TIMER_WRONG_TYPE	4058	X'00000FDA'
MQRCCF_PUT_AUTH_WRONG_TYPE	4059	X'00000FDB'
MQRCCF_KEEP_ALIVE_INT_ERROR	4060	X'00000FDC'
MQRCCF_MISSING_CONN_NAME	4061	X'00000FDD'
MQRCCF_CONN_NAME_ERROR	4062	X'00000FDE'
MQRCCF_MQSET_FAILED	4063	X'00000FDF'
MQRCCF_CHANNEL_NOT_ACTIVE	4064	X'00000FE0'
MQRCCF_TERMINATED_BY_SEC_EXIT	4065	X'00000FE1'
MQRCCF_DYNAMIC_Q_SCOPE_ERROR	4067	X'00000FE3'
MQRCCF_CELL_DIR_NOT_AVAILABLE	4068	X'00000FE4'
MQRCCF_MR_COUNT_ERROR	4069	X'00000FE5'
MQRCCF_MR_COUNT_WRONG_TYPE	4070	X'00000FE6'
MQRCCF_MR_EXIT_NAME_ERROR	4071	X'00000FE7'

表 302. 定数の値 (続き)		
名前	10 進数値	16 進値
MQRCCF_MR_EXIT_NAME_WRONG_TYPE	4072	X'00000FE8'
MQRCCF_MR_INTERVAL_ERROR	4073	X'00000FE9'
MQRCCF_MR_INTERVAL_WRONG_TYPE	4074	X'00000FEA'
MQRCCF_NPM_SPEED_ERROR	4075	X'00000FEB'
MQRCCF_NPM_SPEED_WRONG_TYPE	4076	X'00000FEC'
MQRCCF_HB_INTERVAL_ERROR	4077	X'00000FED'
MQRCCF_HB_INTERVAL_WRONG_TYPE	4078	X'00000FEE'
MQRCCF_CHAD_ERROR	4079	X'00000FEF'
MQRCCF_CHAD_WRONG_TYPE	4080	X'00000FF0'
MQRCCF_CHAD_EVENT_ERROR	4081	X'00000FF1'
MQRCCF_CHAD_EVENT_WRONG_TYPE	4082	X'00000FF2'
MQRCCF_CHAD_EXIT_ERROR	4083	X'00000FF3'
MQRCCF_CHAD_EXIT_WRONG_TYPE	4084	X'00000FF4'
MQRCCF_SUPPRESSED_BY_EXIT	4085	X'00000FF5'
MQRCCF_BATCH_INT_ERROR	4086	X'00000FF6'
MQRCCF_BATCH_INT_WRONG_TYPE	4087	X'00000FF7'
MQRCCF_NET_PRIORITY_ERROR	4088	X'00000FF8'
MQRCCF_NET_PRIORITY_WRONG_TYPE	4089	X'00000FF9'
MQRCCF_CHANNEL_CLOSED	4090	X'00000FFA'
MQRCCF_Q_STATUS_NOT_FOUND	4091	X'00000FFB'
MQRCCF_SSL_CIPHER_SPEC_ERROR	4092	X'00000FFC'
MQRCCF_SSL_PEER_NAME_ERROR	4093	X'00000FFD'
MQRCCF_SSL_CLIENT_AUTH_ERROR	4094	X'00000FFE'
MQRCCF_RETAINED_NOT_SUPPORTED	4095	X'00000FFF'
 MQRCCF_KWD_VALUE_WRONG_TYPE	4096	X'00001000'

MQRCN_* (クライアントの再接続定数)

表 303. 定数の値		
名前	10 進数値	16 進値
MQRCN_NO	0	X'00000000'
MQRCN_YES	1	X'00000001'
MQRCN_Q_MGR	2	X'00000002'
MQRCN_DISABLED	3	X'00000003'

MQRCVTIME_* (受信タイムアウト・タイプ)

表 304. 定数の値		
名前	10 進数値	16 進値
MQRCVTIME_MULTIPLY	0	X'00000000'
MQRCVTIME_ADD	1	X'00000001'

表 304. 定数の値 (続き)		
名前	10 進数値	16 進値
MQRCVTIME_EQUAL	2	X'00000002'

MQREADA_* (先読み値)

表 305. 定数の値		
名前	10 進数値	16 進値
MQREADA_NO	0	X'00000000'
MQREADA_YES	1	X'00000001'
MQREADA_DISABLED	2	X'00000002'
MQREADA_INHIBITED	3	X'00000003'
MQREADA_BACKLOG	4	X'00000004'

MQRECORDING_* (記録オプション)

表 306. 定数の値		
名前	10 進数値	16 進値
MQRECORDING_DISABLED	0	X'00000000'
MQRECORDING_Q	1	X'00000001'
MQRECORDING_MSG	2	X'00000002'

MQREGO_* (パブリッシュ/サブスクライブ登録オプション)

表 307. 定数の値		
名前	10 進数値	16 進値
MQREGO_NONE	0	X'00000000'
MQREGO_CORREL_ID_AS_IDENTITY	1	X'00000001'
MQREGO_ANONYMOUS	2	X'00000002'
MQREGO_LOCAL	4	X'00000004'
MQREGO_DIRECT_REQUESTS	8	X'00000008'
MQREGO_NEW_PUBLICATIONS_ONLY	16	X'00000010'
MQREGO_PUBLISH_ON_REQUEST_ONLY	32	X'00000020'
MQREGO_DEREGISTER_ALL	64	X'00000040'
MQREGO_INCLUDE_STREAM_NAME	128	X'00000080'
MQREGO_INFORM_IF_RETAINED	256	X'00000100'
MQREGO_DUPLICATES_OK	512	X'00000200'
MQREGO_NON_PERSISTENT	1024	X'00000400'
MQREGO_PERSISTENT	2048	X'00000800'
MQREGO_PERSISTENT_AS_PUBLISH	4096	X'00001000'
MQREGO_PERSISTENT_AS_Q	8192	X'00002000'
MQREGO_ADD_NAME	16384	X'00004000'
MQREGO_NO_ALTERATION	32768	X'00008000'
MQREGO_FULL_RESPONSE	65536	X'00010000'
MQREGO_JOIN_SHARED	131072	X'00020000'

表 307. 定数の値 (続き)		
名前	10 進数値	16 進値
MQREGO_JOIN_EXCLUSIVE	262144	X'00040000'
MQREGO_LEAVE_ONLY	524288	X'00080000'
MQREGO_VARIABLE_USER_ID	1048576	X'00100000'
MQREGO_LOCKED	2097152	X'00200000'

MQRFH_* (規則および書式ヘッダーの構造体およびフラグ)

規則および書式ヘッダー構造体

表 308. 定数の構造	
名前	構造体
MQRFH_STRUC_ID	"RFH"
MQRFH_STRUC_ID_ARRAY	'R','F','H',' '

注: 記号は、単一のブランク文字を表します。

表 309. 定数の値		
名前	10 進数値	16 進値
MQRFH_VERSION_1	1	X'00000001'
MQRFH_VERSION_2	2	X'00000002'
MQRFH_STRUC_LENGTH_FIXED	32	X'00000020'
MQRFH_STRUC_LENGTH_FIXED_2	36	X'00000024'

規則および書式ヘッダー・フラグ

表 310. 定数の値		
名前	10 進数値	16 進値
MQRFH_NONE	0	X'00000000'
MQRFH_NO_FLAGS	0	X'00000000'

MQRFH2_* (パブリッシュ/サブスクライブ・オプション・タグの RFH2 トップレベル・フォルダー・タグ)

表 311. 定数の値		
名前	10 進数値	16 進値
MQRFH2_NAME_VALUE_VERSION	1	X'00000001'

MQRFH2_* (パブリッシュ/サブスクライブ・オプション・タグのタグ名)

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

MQRFH2_* (パブリッシュ/サブスクライブ・オプション・タグのXML タグ名)

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

MQRL_* (戻り長)

表 312. 定数の値		
名前	10 進数値	16 進値
MQRL_UNDEFINED	-1	X'FFFFFFFF'

MQRMH_* (参照メッセージ・ヘッダー構造体)

表 313. 定数の構造	
名前	構造体
MQRMH_STRUC_ID	"RMH-"
MQRMH_STRUC_ID_ARRAY	'R','M','H','-'

注：記号-は、単一のブランク文字を表します。

表 314. 定数の値		
名前	10 進数値	16 進値
MQRMH_VERSION_1	1	X'00000001'
MQRMH_CURRENT_VERSION	1	X'00000001'

MQRMHF_* (参照メッセージ・ヘッダー・フラグ)

表 315. 定数の値		
名前	10 進数値	16 進値
MQRMHF_LAST	1	X'00000001'
MQRMHF_NOT_LAST	0	X'00000000'

MQRO_* (レポート・オプション)

表 316. 定数の値		
名前	10 進数値	16 進値
MQRO_EXCEPTION	16777216	X'01000000'
MQRO_EXCEPTION_WITH_DATA	50331648	X'03000000'
MQRO_EXCEPTION_WITH_FULL_DATA	117440512	X'07000000'
MQRO_EXPIRATION	2097152	X'00200000'

表 316. 定数の値 (続き)		
名前	10 進数値	16 進値
MQRO_EXPIRATION_WITH_DATA	6291456	X'00600000'
MQRO_EXPIRATION_WITH_FULL_DATA	14680064	X'00E00000'
MQRO_COA	256	X'00000100'
MQRO_COA_WITH_DATA	768	X'00000300'
MQRO_COA_WITH_FULL_DATA	1792	X'00000700'
MQRO_COD	2048	X'00000800'
MQRO_COD_WITH_DATA	6144	X'00001800'
MQRO_COD_WITH_FULL_DATA	14336	X'00003800'
MQRO_PAN	1	X'00000001'
MQRO_NAN	2	X'00000002'
MQRO_ACTIVITY	4	X'00000004'
MQRO_NEW_MSG_ID	0	X'00000000'
MQRO_PASS_MSG_ID	128	X'00000080'
MQRO_COPY_MSG_ID_TO_CORREL_ID	0	X'00000000'
MQRO_PASS_CORREL_ID	64	X'00000040'
MQRO_DEAD_LETTER_Q	0	X'00000000'
MQRO_DISCARD_MSG	134217728	X'08000000'
MQRO_PASS_DISCARD_AND_EXPIRY	16384	X'00004000'
MQRO_NONE	0	X'00000000'

MQRO_* (レポート・オプション・マスク)

表 317. 定数の値		
名前	10 進数値	16 進値
MQRO_REJECT_UNSUP_MASK	270270464	X'101C0000'
MQRO_ACCEPT_UNSUP_MASK	-270532353	X'EFE000FF'
MQRO_ACCEPT_UNSUP_IF_XMIT_MASK	261888	X'0003FF00'

MQROUTE_* (経路トレース)

経路トレース最大アクティビティー (MQIACF_MAX_ACTIVITIES)

表 318. 定数の値		
名前	10 進数値	16 進値
MQROUTE_UNLIMITED_ACTIVITIES	0	X'00000000'

経路トレース詳細 (MQIACF_ROUTE_DETAIL)

表 319. 定数の値		
名前	10 進数値	16 進値
MQROUTE_DETAIL_LOW	2	X'00000002'
MQROUTE_DETAIL_MEDIUM	8	X'00000008'
MQROUTE_DETAIL_HIGH	32	X'00000020'

経路トレース転送 (MQIACF_ROUTE_FORWARDING)

表 320. 定数の値		
名前	10 進数値	16 進値
MQROUTE_FORWARD_ALL	256	X'00000100'
MQROUTE_FORWARD_IF_SUPPORTED	512	X'00000200'
MQROUTE_FORWARD_REJ_UNSUP_MASK	-65536	X'FFFF0000'

経路トレース送達 (MQIACF_ROUTE_DELIVERY)

表 321. 定数の値		
名前	10 進数値	16 進値
MQROUTE_DELIVER_YES	4096	X'00001000'
MQROUTE_DELIVER_NO	8192	X'00002000'
MQROUTE_DELIVER_REJ_UNSUP_MASK	-65536	X'FFFF0000'

経路トレース累積 (MQIACF_ROUTE_ACCUMULATION)

表 322. 定数の値		
名前	10 進数値	16 進値
MQROUTE_ACCUMULATE_NONE	65539	X'00010003'
MQROUTE_ACCUMULATE_IN_MSG	65540	X'00010004'
MQROUTE_ACCUMULATE_AND_REPLY	65541	X'00010005'

MQRP_* (コマンド形式の置き換えオプション)

表 323. 定数の値		
名前	10 進数値	16 進値
MQRP_YES	1	X'00000001'
MQRP_NO	0	X'00000000'

MQRQ_* (コマンド形式の理由修飾子)

表 324. 定数の値		
名前	10 進数値	16 進値
MQRQ_CONN_NOT_AUTHORIZED	1	X'00000001'
MQRQ_OPEN_NOT_AUTHORIZED	2	X'00000002'
MQRQ_CLOSE_NOT_AUTHORIZED	3	X'00000003'
MQRQ_CMD_NOT_AUTHORIZED	4	X'00000004'
MQRQ_Q_MGR_STOPPING	5	X'00000005'
MQRQ_Q_MGR QUIESCING	6	X'00000006'
MQRQ_CHANNEL_STOPPED_OK	7	X'00000007'
MQRQ_CHANNEL_STOPPED_ERROR	8	X'00000008'
MQRQ_CHANNEL_STOPPED_RETRY	9	X'00000009'
MQRQ_CHANNEL_STOPPED_DISABLED	10	X'0000000A'
MQRQ_BRIDGE_STOPPED_OK	11	X'0000000B'

表 324. 定数の値 (続き)		
名前	10 進数値	16 進値
MQRQ_BRIDGE_STOPPED_ERROR	12	X'0000000C'
MQRQ_SSL_HANDSHAKE_ERROR	13	X'0000000D'
MQRQ_SSL_CIPHER_SPEC_ERROR	14	X'0000000E'
MQRQ_SSL_CLIENT_AUTH_ERROR	15	X'0000000F'
MQRQ_SSL_PEER_NAME_ERROR	16	X'00000010'
MQRQ_SUB_NOT_AUTHORIZED	17	X'00000011'
MQRQ_SUB_DEST_NOT_AUTHORIZED	18	X'00000012'
MQRQ_SSL_UNKNOWN_REVOCATION	19	X'00000013'
MQRQ_SYS_CONN_NOT_AUTHORIZED	20	X'00000014'
MQRQ_CHANNEL_BLOCKED_ADDRESS	21	X'00000015'
MQRQ_CHANNEL_BLOCKED_USERID	22	X'00000016'
MQRQ_CHANNEL_BLOCKED_NOACCESS	23	X'00000017'
MQRQ_MAX_ACTIVE_CHANNELS	24	X'00000018'
MQRQ_MAX_CHANNELS	25	X'00000019'
MQRQ_SVRCONN_INST_LIMIT	26	X'0000001A'
MQRQ_CLIENT_INST_LIMIT!	27	X'0000001B'
MQRQ_CAF_NOT_INSTALLED	28	X'0000001C'
MQRQ_CSP_NOT_AUTHORIZED	29	X'0000001D'
MQRQ_FAILOVER_PERMITTED	30	X'0000001E'
MQRQ_FAILOVER_NOT_PERMITTED	31	X'0000001F'
MQRQ_STANDBY_ACTIVATED	32	X'00000020'
MQRQ_REPLICA_ACTIVATED	33	X'00000021'
 MQRQ_CONN_AUTHORIZED	65	X'00000041'
 MQRQ_OPEN_AUTHORIZED	66	X'00000042'
 MQRQ_SUB_AUTHORIZED	67	X'00000043'
 MQRQ_SUB_DEST_AUTHORIZED	68	X'00000044'

MQRT_* (コマンド形式の最新表示タイプ)

表 325. 定数の値		
名前	10 進数値	16 進値
MQRT_CONFIGURATION	1	X'00000001'
MQRT_EXPIRY	2	X'00000002'
MQRT_NSPROC	3	X'00000003'
MQRT_PROXYSUB	4	X'00000004'

MQRU_* (要求のみ)

表 326. 定数の値		
名前	10 進数値	16 進値
MQRU_PUBLISH_ON_REQUEST	1	X'00000001'
MQRU_PUBLISH_ALL	2	X'00000002'

MQSCA_* (TLS クライアント認証)

表 327. 定数の値		
名前	10 進数値	16 進値
MQSCA_REQUIRED	0	X'00000000'
MQSCA_OPTIONAL	1	X'00000001'

MQSCO_* (TLS 構成オプション)

TLS 構成オプションの構造体

表 328. 定数の構造	
名前	構造体
MQSCO_STRUC_ID	"SCO"
MQSCO_STRUC_ID_ARRAY	'S','C','O',' '

注: 記号-は、単一のブランク文字を表します。

表 329. 定数の値		
名前	10 進数値	16 進値
MQSCO_VERSION_1	1	X'00000001'
MQSCO_VERSION_2	2	X'00000002'
MQSCO_VERSION_3	3	X'00000003'
MQSCO_VERSION_4	4	X'00000004'
MQSCO_VERSION_5	5	X'00000005'
MQSCO_VERSION_6	6	X'00000006'
 MQSCO_VERSION_7	7	X'00000007'
MQSCO_CURRENT_VERSION	 7,  6	 X'00000007',  X'00000006'

注: 記号-は、単一のブランク文字を表します。

TLS 構成オプション・キー・リセット・カウンタ

表 330. 定数の値		
名前	10 進数値	16 進値
MQSCO_RESET_COUNT_DEFAULT	0	X'00000000'

コマンド形式のキュー定義有効範囲

表 331. 定数の値		
名前	10 進数値	16 進値
MQSCO_Q_MGR	1	X'00000001'
MQSCO_CELL	2	X'00000002'

MQSCOPE_* (パブリッシュ有効範囲)

表 332. 定数の値		
名前	10 進数値	16 進値
MQSCOPE_ALL	0	X'00000000'
MQSCOPE_AS_PARENT	1	X'00000001'
MQSCOPE_QMGR	4	X'00000004'

MQSCYC_* (セキュリティ・ケース)

表 333. 定数の値		
名前	10 進数値	16 進値
MQSCYC_UPPER	0	X'00000000'
MQSCYC_MIXED	1	X'00000001'

MQSD_* (オブジェクト記述子構造体)

表 334. 定数名と構造体	
名前	構造体
MQSD_STRUC_ID	"SD-"
MQSD_STRUC_ID_ARRAY	'S','D',' ',' '

注: 記号-は、単一のブランク文字を表します。

表 335. 定数の値		
名前	10 進数値	16 進値
MQSD_VERSION_1	1	X'00000001'
MQSD_CURRENT_VERSION	1	X'00000001'

MQSECITEM_* (コマンド形式のセキュリティ項目)

表 336. 定数の値		
名前	10 進数値	16 進値
MQSECITEM_ALL	0	X'00000000'
MQSECITEM_MQADMIN	1	X'00000001'
MQSECITEM_MQNLIST	2	X'00000002'
MQSECITEM_MQPROC	3	X'00000003'
MQSECITEM_MQQUEUE	4	X'00000004'
MQSECITEM_MQCONN	5	X'00000005'
MQSECITEM_MQCMDS	6	X'00000006'
MQSECITEM_MXADMIN	7	X'00000007'
MQSECITEM_MXNLIST	8	X'00000008'
MQSECITEM_MXPROC	9	X'00000009'
MQSECITEM_MXQUEUE	10	X'0000000A'
MQSECITEM_MXTOPIC	11	X'0000000B'

MQSECPROT_* (セキュリティ・プロトコル・タイプ)

表 337. 定数の値		
名前	10 進数値	16 進値
MQSECPROT_NONE	0	X'00000000'
MQSECPROT_SSLV30	1	X'00000001'
MQSECPROT_TLSV10	2	X'00000002'
MQSECPROT_TLSV12	4	X'00000004'

MQSECSW_* (コマンド形式のセキュリティ・スイッチおよびスイッチ状態)

コマンド形式のセキュリティ・スイッチ

表 338. 定数の値		
名前	10 進数値	16 進値
MQSECSW_PROCESS	1	X'00000001'
MQSECSW_NAMELIST	2	X'00000002'
MQSECSW_Q	3	X'00000003'
MQSECSW_TOPIC	4	X'00000004'
MQSECSW_CONTEXT	6	X'00000006'
MQSECSW_ALTERNATE_USER	7	X'00000007'
MQSECSW_COMMAND	8	X'00000008'
MQSECSW_CONNECTION	9	X'00000009'
MQSECSW_SUBSYSTEM	10	X'0000000A'
MQSECSW_COMMAND_RESOURCES	11	X'0000000B'
MQSECSW_Q_MGR	15	X'0000000F'
MQSECSW_QSG	16	X'00000010'

コマンド形式のセキュリティ・スイッチ状態

表 339. 定数の値		
名前	10 進数値	16 進値
MQSECSW_OFF_FOUND	21	X'00000015'
MQSECSW_ON_FOUND	22	X'00000016'
MQSECSW_OFF_NOT_FOUND	23	X'00000017'
MQSECSW_ON_NOT_FOUND	24	X'00000018'
MQSECSW_OFF_ERROR	25	X'00000019'
MQSECSW_ON_OVERRIDDEN	26	X'0000001A'

MQSECTYPE_* (コマンド形式のセキュリティ・タイプ)

表 340. 定数の値		
名前	10 進数値	16 進値
MQSECTYPE_AUTHSERV	1	X'00000001'
MQSECTYPE_SSL	2	X'00000002'

表 340. 定数の値 (続き)		
名前	10 進数値	16 進値
MQSECTYPE_CLASSES	3	X'00000003'

MQSEG_* (セグメンテーション)

表 341. 定数名と値	
名前	値
MQSEG_INHIBITED	'␣'
MQSEG_ALLOWED	'A'

注: 記号␣は、単一のブランク文字を表します。

MQSEL_* (特殊セクター値)

表 342. 定数の値		
名前	10 進数値	16 進値
MQSEL_ANY_SELECTOR	-30001	X'FFFF8ACF'
MQSEL_ANY_USER_SELECTOR	-30002	X'FFFF8ACE'
MQSEL_ANY_SYSTEM_SELECTOR	-30003	X'FFFF8ACD'
MQSEL_ALL_SELECTORS	-30001	X'FFFF8ACF'
MQSEL_ALL_USER_SELECTORS	-30002	X'FFFF8ACE'
MQSEL_ALL_SYSTEM_SELECTORS	-30003	X'FFFF8ACD'

MQSELTYPE_* (セクター・タイプ)

表 343. 定数の値		
名前	10 進数値	16 進値
MQSELTYPE_NONE	0	X'00000000'
MQSELTYPE_STANDARD	1	X'00000001'
MQSELTYPE_EXTENDED	2	X'00000002'

MQSID_* (セキュリティー ID)

表 344. 定数名と値	
名前	値
MQSID_NONE	X'00...00' (40 スル)
MQSID_NONE_ARRAY	'\0', '\0', ... (40 スル)

MQSIDT_* (セキュリティー ID タイプ)

表 345. 定数名と値	
名前	16 進値
MQSIDT_NONE	X'00'
MQSIDT_NT_SECURITY_ID	X'01'
MQSIDT_WAS_SECURITY_ID	X'02'

MQSMPO_* (メッセージ・プロパティの設定オプションおよび構造体)

メッセージ・プロパティ設定オプション構造体

表 346. 定数の構造	
名前	構造体
MQSMPO_STRUC_ID	"SMPO"
MQSMPO_STRUC_ID_ARRAY	'S','M','P','O'

注: 記号-は、単一の空白文字を表します。

表 347. 定数の値		
名前	10 進数値	16 進値
MQSMPO_VERSION_1	1	X'00000001'
MQSMPO_CURRENT_VERSION	1	X'00000001'

メッセージ・プロパティ設定オプション

表 348. 定数の値		
名前	10 進数値	16 進値
MQSMPO_SET_FIRST	0	X'00000000'
MQSMPO_SET_PROP_UNDER_CURSOR	1	X'00000001'
MQSMPO_SET_PROP_AFTER_CURSOR	2	X'00000002'
MQSMPO_APPEND_PROPERTY	4	X'00000004'
MQSMPO_SET_PROP_BEFORE_CURSOR	8	X'00000008'
MQSMPO_NONE	0	X'00000000'

MQSO_* (サブスクライブ・オプション)

表 349. 定数の値		
名前	10 進数値	16 進値
MQSO_NONE	0	X'00000000'
MQSO_NON_DURABLE	0	X'00000000'
MQSO_READ_AHEAD_AS_Q_DEF	0	X'00000000'
MQSO_ALTER	1	X'00000001'
MQSO_CREATE	2	X'00000002'
MQSO_RESUME	4	X'00000004'
MQSO_DURABLE	8	X'00000008'
MQSO_GROUP_SUB	16	X'00000010'
MQSO_MANAGED	32	X'00000020'
MQSO_SET_IDENTITY_CONTEXT	64	X'00000040'
MQSO_FIXED_USERID	256	X'00000100'
MQSO_ANY_USERID	512	X'00000200'
MQSO_PUBLICATIONS_ON_REQUEST	2048	X'00000800'
MQSO_NEW_PUBLICATIONS_ONLY	4096	X'00001000'
MQSO_FAIL_IF_QUIESCING	8192	X'00002000'

表 349. 定数の値 (続き)		
名前	10 進数値	16 進値
MQSO_ALTERNATE_USER_AUTHORITY	262144	X'00040000'
MQSO_WILDCARD_CHAR	1048576	X'00100000'
MQSO_WILDCARD_TOPIC	2097152	X'00200000'
MQSO_SET_CORREL_ID	4194304	X'00400000'
MQSO_SCOPE_QMGR	67108864	X'04000000'
MQSO_NO_READ_AHEAD	134217728	X'08000000'
MQSO_READ_AHEAD	268435456	X'10000000'

MQSP_* (同期点の可用性)

表 350. 定数の値		
名前	10 進数値	16 進値
MQSP_AVAILABLE	1	X'00000001'
MQSP_NOT_AVAILABLE	0	X'00000000'

MQSPL_* (セキュリティ・ポリシー保護オプション)

表 351. 定数の値		
名前	10 進数値	16 進値
MQSPL_PASSTHRU	0	X'00000000'
MQSPL_REMOVE	1	X'00000001'
MQSPL_AS_POLICY	2	X'00000002'

MQSQQM_* (共有キューのキュー・マネージャー名)

表 352. 定数の値		
名前	10 進数値	16 進値
MQSQQM_USE	0	X'00000000'
MQSQQM_IGNORE	1	X'00000001'

MQSR_* (アクション)

表 353. 定数の値		
名前	10 進数値	16 進値
MQSR_ACTION_PUBLICATION	1	X'00000001'

MQSRO_* (サブスクリプション要求オプション構造体)

表 354. 定数の構造	
名前	構造体
MQSRO_STRUC_ID	"SRO-"
MQSRO_STRUC_ID_ARRAY	'S', 'R', 'O', ' '

注: 記号-は、単一のブランク文字を表します。

表 355. 定数の値		
名前	10 進数値	16 進値
MQSRO_VERSION_1	1	X'00000001'
MQSRO_CURRENT_VERSION	1	X'00000001'
MQSRO_NONE	0	X'00000000'
MQSRO_FAIL_IF QUIESCING	8192	X'00002000'

MQSS_* (セグメント状況)

表 356. 定数名と構造体	
名前	構造体
MQSS_NOT_A_SEGMENT	'-'
MQSS_SEGMENT	'S'
MQSS_LAST_SEGMENT	'L'

注：記号-は、単一のブランク文字を表します。

MQSSL_* (TLSFIPS 要件)

注：

-  WindowsAIX、Linux、およびにおいて、IBM MQ は GSKit 9 IBM Crypto for C (ICC) 暗号モジュールを通じてへの準拠を実現 FIPS 140-3 しています。この FIPS 140-3 モジュールに関連する NIST 認証は、<https://csrc.nist.gov/projects/cryptographic-module-validation-program/certificate/4755> で確認できます。
-  Linux s390xFIPS 現在、.への対応は行っていません。IBM MQ このプラットフォームをご利用のお客様でサポートが必要な FIPS 場合は、以前のバージョンのままご利用ください。このプラットフォーム上で暗号モジュールが IBM Crypto for C (ICC) 認証を取得次第、今後の FixPack でサポートが FIPS 140-3 有効化されます。
-  このソフトウェアに含まれる暗号 FIPS 140-3 モジュールは、2024 IBM Semeru Runtime 年 8 月に NIST の承認を受けた。IBM Semeru Runtime 11+IBM MQ 10.0.0 TLS を使用したおよび IBM MQ classes for Java のクライアント接続の IBM MQ classes for JMS 処理機能を、FIPS 140-3 およびに Java 8 追加しました。<https://csrc.nist.gov/projects/cryptographic-module-validation-program/certificate/4755> この FIPS 140-3 モジュールに関連する NIST 認証は、以下で確認できます。
- Containers では IBM MQ、3.2.0 およびキューマネージャ IBM MQ Operator ーのコンテナイメージ 9.4.0.0 以降は、UBI 9 をベースとしています。FIPS 140-3 Containers における IBM MQ コンプライアンスは、現在審査中です。

 が有効になっている FIPS 場合、コンテナ制御 IBM MQ プロセスでは、認定 FIPS 140-3 OpenSSL モジュールが使用されます。NIST 認証の詳細については、以下の URL をご覧ください：<https://access.redhat.com/compliance/fips>。IBM MQ コンテナイメージ内で実行されるキューマネージャは、ベースイメージのプラットフォームバージョンと同じ FIPS IBM MQ 認証レベルを持っています。

表 357. 定数の値		
名前	10 進数値	16 進値
MQSSL_FIPS_NO	0	X'00000000'
MQSSL_FIPS_YES	1	X'00000001'

MQSTAT_* (状況オプション)

表 358. 定数の値		
名前	10 進数値	16 進値
MQSTAT_TYPE_ASYNC_ERROR	0	X'00000000'
MQSTAT_TYPE_RECONNECTION	0	X'00000000'
MQSTAT_TYPE_RECONNECTION_ERROR	0	X'00000000'

MQSTS_* (状況報告構造体)

表 359. 定数の構造	
名前	構造体
MQSTS_STRUC_ID	"STAT"
MQSTS_STRUC_ID_ARRAY	'S','T','A','T'

注: 記号-は、単一のブランク文字を表します。

表 360. 定数の値		
名前	10 進数値	16 進値
MQSTS_VERSION_1	1	X'00000001'
MQSTS_CURRENT_VERSION	1	X'00000001'

MQSUB_* (永続サブスクリプション)

永続許可サブスクリプション

表 361. 定数の値		
名前	10 進数値	16 進値
MQSUB_DURABLE_AS_PARENT	0	X'00000000'
MQSUB_DURABLE_ALLOWED	1	X'00000001'
MQSUB_DURABLE_INHIBITED	2	X'00000002'

永続サブスクリプションの範囲

表 362. 定数の値		
名前	10 進数値	16 進値
MQSUB_DURABLE_ALL	-1	X'FFFFFFFF'
MQSUB_DURABLE_YES	1	X'00000001'
MQSUB_DURABLE_NO	2	X'00000002'

MQSUBTYPE_* (コマンド形式のサブスクリプション・タイプ)

表 363. 定数の値		
名前	10 進数値	16 進値
MQSUBTYPE_API	1	X'00000001'
MQSUBTYPE_ADMIN	2	X'00000002'
MQSUBTYPE_PROXY	3	X'00000003'
MQSUBTYPE_ALL	-1	X'FFFFFFFF'

表 363. 定数の値 (続き)		
名前	10 進数値	16 進値
MQSUBTYPE_USER	-2	X'FFFFFFFFE'

MQSUS_* (コマンド形式の中断状況)

表 364. 定数の値		
名前	10 進数値	16 進値
MQSUS_YES	1	X'00000001'
MQSUS_NO	0	X'00000000'

MQSVC_* (サービス)

サービス・タイプ

表 365. 定数の値		
名前	10 進数値	16 進値
MQSVC_TYPE_COMMAND	0	X'00000000'
MQSVC_TYPE_SERVER	1	X'00000001'

サービス制御

表 366. 定数の値		
名前	10 進数値	16 進値
MQSVC_CONTROL_Q_MGR	0	X'00000000'
MQSVC_CONTROL_Q_MGR_START	1	X'00000001'
MQSVC_CONTROL_MANUAL	2	X'00000002'

サービス状況

表 367. 定数の値		
名前	10 進数値	16 進値
MQSVC_STATUS_STOPPED	0	X'00000000'
MQSVC_STATUS_STARTING	1	X'00000001'
MQSVC_STATUS_RUNNING	2	X'00000002'
MQSVC_STATUS_STOPPING	3	X'00000003'
MQSVC_STATUS_RETRYING	4	X'00000004'

MQSYNCPOINT_* (コマンド形式のパブリッシュ/サブスクライブ・マイグレーションの同期点値)

表 368. 定数の値		
名前	10 進数値	16 進値
MQSYNCPOINT_YES	0	X'00000000'
MQSYNCPOINT_IFPER	1	X'00000001'

MQSYSP_* (コマンド形式のシステム・パラメーター値)

表 369. 定数の値		
名前	10 進数値	16 進値
MQSYSP_NO	0	X'00000000'
MQSYSP_YES	1	X'00000001'
MQSYSP_EXTENDED	2	X'00000002'
MQSYSP_TYPE_INITIAL	10	X'0000000A'
MQSYSP_TYPE_SET	11	X'0000000B'
MQSYSP_TYPE_LOG_COPY	12	X'0000000C'
MQSYSP_TYPE_LOG_STATUS	13	X'0000000D'
MQSYSP_TYPE_ARCHIVE_TAPE	14	X'0000000E'
MQSYSP_ALLOC_BLK	20	X'00000014'
MQSYSP_ALLOC_TRK	21	X'00000015'
MQSYSP_ALLOC_CYL	22	X'00000016'
MQSYSP_STATUS_BUSY	30	X'0000001E'
MQSYSP_STATUS_PREMOUNT	31	X'0000001F'
MQSYSP_STATUS_AVAILABLE	32	X'00000020'
MQSYSP_STATUS_UNKNOWN	33	X'00000021'
MQSYSP_STATUS_ALLOC_ARCHIVE	34	X'00000022'
MQSYSP_STATUS_COPYING_BSDS	35	X'00000023'
MQSYSP_STATUS_COPYING_LOG	36	X'00000024'

MQTA_* (トピック属性)

ワイルドカード

表 370. 定数の値		
名前	10 進数値	16 進値
MQTA_BLOCK	1	X'00000001'
MQTA_PASSTHRU	2	X'00000002'

許可されるサブスクリプション

表 371. 定数の値		
名前	10 進数値	16 進値
MQTA_SUB_AS_PARENT	0	X'00000000'
MQTA_SUB_INHIBITED	1	X'00000001'
MQTA_SUB_ALLOWED	2	X'00000002'

プロキシ・サブスクリプション伝搬

表 372. 定数の値		
名前	10 進数値	16 進値
MQTA_PROXY_SUB_FORCE	1	X'00000001'

表 372. 定数の値 (続き)		
名前	10 進数値	16 進値
MQTA_PROXY_SUB_FIRSTUSE	2	X'00000002'

許可されるパブリケーション

表 373. 定数の値		
名前	10 進数値	16 進値
MQTA_PUB_AS_PARENT	0	X'00000000'
MQTA_PUB_INHIBITED	1	X'00000001'
MQTA_PUB_ALLOWED	2	X'00000002'

MQTC_* (トリガー制御)

表 374. 定数の値		
名前	10 進数値	16 進値
MQTC_OFF	0	X'00000000'
MQTC_ON	1	X'00000001'

MQTCPKEEP_* (TCP キープアライブ)

表 375. 定数の値		
名前	10 進数値	16 進値
MQTCPKEEP_NO	0	X'00000000'
MQTCPKEEP_YES	1	X'00000001'

MQTCPSTACK_* (TCP スタック・タイプ)

表 376. 定数の値		
名前	10 進数値	16 進値
MQTCPSTACK_SINGLE	0	X'00000000'
MQTCPSTACK_MULTIPLE	1	X'00000001'

MQTIME_* (コマンド形式の時間単位)

表 377. 定数の値		
名前	10 進数値	16 進値
MQTIME_UNIT_MINS	0	X'00000000'
MQTIME_UNIT_SECS	1	X'00000001'

MQTM_* (トリガー・メッセージ構造体)

表 378. 定数の構造	
名前	構造体
MQTM_STRUC_ID	"TM--"
MQTM_STRUC_ID_ARRAY	'T','M',' ',' '

注: 記号-は、単一のブランク文字を表します。

表 379. 定数の値		
名前	10 進数値	16 進値
MQTM_VERSION_1	1	X'00000001'
MQTM_CURRENT_VERSION	1	X'00000001'

MQTMC_* (トリガー・メッセージ文字フォーマット構造体)

表 380. 定数の構造	
名前	構造体
MQTMC_STRUC_ID	"TMC"
MQTMC_STRUC_ID_ARRAY	'T','M','C',' '
MQTMC_VERSION_1	"1"
MQTMC_VERSION_2	"2"
MQTMC_CURRENT_VERSION	"2"
MQTMC_VERSION_1_ARRAY	' ',' ',' ','1'
MQTMC_VERSION_2_ARRAY	' ',' ',' ','2'
MQTMC_CURRENT_VERSION_ARRAY	' ',' ',' ','2'

MQTOPT_* (トピック・タイプ)

表 381. 定数の値		
名前	10 進数値	16 進値
MQTOPT_LOCAL	0	X'00000000'
MQTOPT_CLUSTER	1	X'00000001'
MQTOPT_ALL	2	X'00000002'

MQTRAXSTR_* (チャネル・イニシエーター・トレース自動始動)

表 382. 定数の値		
名前	10 進数値	16 進値
MQTRAXSTR_NO	0	X'00000000'
MQTRAXSTR_YES	1	X'00000001'

MQTSCOPE_* (サブスクリプション有効範囲)

表 383. 定数の値		
名前	10 進数値	16 進値
MQTSCOPE_QMGR	1	X'00000001'
MQTSCOPE_ALL	2	X'00000002'

MQTT_* (トリガー・タイプ)

表 384. 定数の値		
名前	10 進数値	16 進値
MQTT_NONE	0	X'00000000'
MQTT_FIRST	1	X'00000001'
MQTT EVERY	2	X'00000002'

表 384. 定数の値 (続き)		
名前	10 進数値	16 進値
MQTT_DEPTH	3	X'00000003'

MQTYPE_* (プロパティ・データ・タイプ)

表 385. 定数の値		
名前	10 進数値	16 進値
MQTYPE_AS_SET	0	X'00000000'
MQTYPE_NULL	2	X'00000002'
MQTYPE_BOOLEAN	4	X'00000004'
MQTYPE_BYTE_STRING	8	X'00000008'
MQTYPE_INT8	16	X'00000010'
MQTYPE_INT16	32	X'00000020'
MQTYPE_INT32	64	X'00000040'
MQTYPE_LONG	64	X'00000040'
MQTYPE_INT64	128	X'00000080'
MQTYPE_FLOAT32	256	X'00000100'
MQTYPE_FLOAT64	512	X'00000200'
MQTYPE_STRING	1024	X'00000400'

MQUA_* (パブリッシュ/サブスクライブ・ユーザー属性セクター)

表 386. 定数の値		
名前	10 進数値	16 進値
MQUA_FIRST	65536	X'00010000'
MQUA_LAST	999999999	X'3B9AC9FF'

MQUIDSUPP_* (コマンド形式のユーザー ID サポート)

表 387. 定数の値		
名前	10 進数値	16 進値
MQUIDSUPP_NO	0	X'00000000'
MQUIDSUPP_YES	1	X'00000001'

MQUNDELIVERED_* (コマンド形式のパブリッシュ/サブスクライブ・マイグレーションの未配布値)

表 388. 定数の値		
名前	10 進数値	16 進値
MQUNDELIVERED_NORMAL	0	X'00000000'
MQUNDELIVERED_SAFE	1	X'00000001'
MQUNDELIVERED_DISCARD	2	X'00000002'
MQUNDELIVERED_KEEP	3	X'00000003'

MQUOWST_* (コマンド形式の UOW 状態)

表 389. 定数の値		
名前	10 進数値	16 進値
MQUOWST_NONE	0	X'00000000'
MQUOWST_ACTIVE	1	X'00000001'
MQUOWST_PREPARED	2	X'00000002'
MQUOWST_UNRESOLVED	3	X'00000003'

MQUOWT_* (コマンド形式の UOW タイプ)

表 390. 定数の値		
名前	10 進数値	16 進値
MQUOWT_Q_MGR	0	X'00000000'
MQUOWT_CICS	1	X'00000001'
MQUOWT_RRS	2	X'00000002'
MQUOWT_IMS	3	X'00000003'
MQUOWT_XA	4	X'00000004'

MQUS_* (キューの使用法)

表 391. 定数の値		
名前	10 進数値	16 進値
MQUS_NORMAL	0	X'00000000'
MQUS_TRANSMISSION	1	X'00000001'

MQUSAGE_* (コマンド形式のページ・セット使用状況値およびデータ・セット使用状況値)

コマンド形式のページ・セット使用状況値

表 392. 定数の値		
名前	10 進数値	16 進値
MQUSAGE_PS_AVAILABLE	0	X'00000000'
MQUSAGE_PS_DEFINED	1	X'00000001'
MQUSAGE_PS_OFFLINE	2	X'00000002'
MQUSAGE_PS_NOT_DEFINED	3	X'00000003'
MQUSAGE_PS_SUSPENDED	4	X'00000004'
MQUSAGE_EXPAND_USER	1	X'00000001'
MQUSAGE_EXPAND_SYSTEM	2	X'00000002'
MQUSAGE_EXPAND_NONE	3	X'00000003'

コマンド形式のデータ・セット使用状況値

表 393. 定数の値		
名前	10 進数値	16 進値
MQUSAGE_DS_OLDEST_ACTIVE_UOW	10	X'0000000A'

表 393. 定数の値 (続き)		
名前	10 進数値	16 進値
MQUSAGE_DS_OLDEST_PS_RECOVERY	11	X'0000000B'
MQUSAGE_DS_OLDEST_CF_RECOVERY	12	X'0000000C'

MQVL_* (値の長さ)

表 394. 定数の値		
名前	10 進数値	16 進値
MQVL_NULL_TERMINATED	-1	X'FFFFFFFF'
MQVL_EMPTY_STRING	0	X'00000000'

MQVU_* (可変ユーザー ID)

表 395. 定数の値		
名前	10 進数値	16 進値
MQVU_FIXED_USER	1	X'00000001'
MQVU_ANY_USER	2	X'00000002'

MQWDR_* (クラスター・ワークロード出口宛先レコード構造体)

表 396. 定数の構造	
名前	構造体
MQWDR_STRUC_ID	"WDR-"
MQWDR_STRUC_ID_ARRAY	'W', 'D', 'R', '-'

注: 記号-は、単一のブランク文字を表します。

表 397. 定数の値		
名前	10 進数値	16 進値
MQWDR_VERSION_1	1	X'00000001'
MQWDR_VERSION_2	2	X'00000002'
MQWDR_CURRENT_VERSION	2	X'00000002'
MQWDR_LENGTH_1	124	X'0000007C'
MQWDR_LENGTH_2	136	X'00000088'
MQWDR_CURRENT_LENGTH	136	X'00000088'

MQWI_* (待機間隔)

表 398. 定数の値		
名前	10 進数値	16 進値
MQWI_UNLIMITED	-1	X'FFFFFFFF'

MQWIH_* (ワークロード情報ヘッダー構造体およびフラグ)

ワークロード情報ヘッダー構造体

表 399. 定数の構造	
名前	構造体
MQWIH_STRUC_ID	"WIH-"
MQWIH_STRUC_ID_ARRAY	'W', 'I', 'H', '-'

注: 記号-は、単一のブランク文字を表します。

表 400. 定数の値		
名前	10 進数値	16 進値
MQWIH_VERSION_1	1	X'00000001'
MQWIH_CURRENT_VERSION	1	X'00000001'
MQWIH_LENGTH_1	120	X'00000078'
MQWIH_CURRENT_LENGTH	120	X'00000078'

ワークロード情報ヘッダー・フラグ

表 401. 定数の値		
名前	10 進数値	16 進値
MQWIH_NONE	0	X'00000000'

MQWQR_* (クラスター・ワークロード出口キュー・レコード構造体)

表 402. 定数の構造	
名前	構造体
MQWQR_STRUC_ID	"WQR-"
MQWQR_STRUC_ID_ARRAY	'W', 'Q', 'R', '-'

注: 記号-は、単一のブランク文字を表します。

表 403. 定数の値		
名前	10 進数値	16 進値
MQWQR_VERSION_1	1	X'00000001'
MQWQR_VERSION_2	2	X'00000002'
MQWQR_VERSION_3	3	X'00000003'
MQWQR_CURRENT_VERSION	3	X'00000003'
MQWQR_LENGTH_1	200	X'000000C8'
MQWQR_LENGTH_2	208	X'000000D0'
MQWQR_LENGTH_3	212	X'000000D4'
MQWQR_CURRENT_LENGTH	212	X'000000D4'

MQWS_* (ワイルドカード・スキーマ)

表 404. 定数の値		
名前	10 進数値	16 進値
MQWS_DEFAULT	0	X'00000000'
MQWS_CHAR	1	X'00000001'
MQWS_TOPIC	2	X'00000002'

MQWXP_* (クラスター・ワークロード出口パラメーター構造体)

MQWXP_* (クラスター・ワークロード出口パラメーター構造体)

表 405. 定数の構造	
名前	構造体
MQWXP_STRUC_ID	"WXP-"
MQWXP_STRUC_ID_ARRAY	'W','X','P','-'

注: 記号-は、単一のブランク文字を表します。

表 406. 定数の値		
名前	10 進数値	16 進値
MQWXP_VERSION_1	1	X'00000001'
MQWXP_VERSION_2	2	X'00000002'
MQWXP_VERSION_3	3	X'00000003'
MQWXP_VERSION_4	4	X'00000004'
MQWXP_CURRENT_VERSION	4	X'00000004'

MQWXP_* (クラスター・ワークロード・フラグ)

表 407. 定数の値		
名前	10 進数値	16 進値
MQWXP_PUT_BY_CLUSTER_CHL	2	X'00000002'

関連資料

1546 ページの『[MQWXP - クラスター・ワークロード出口のパラメーター構造体内のフィールド](#)』
MQWXP - クラスター・ワークロード出口のパラメーター構造体内のフィールドの説明

MQXACT_* (API 呼び出し側のタイプ)

表 408. 定数の値		
名前	10 進数値	16 進値
MQXACT_EXTERNAL	1	X'00000001'
MQXACT_INTERNAL	2	X'00000002'

MQXC_* (出口コマンド)

表 409. 定数の値		
名前	10 進数値	16 進値
MQXC_MQOPEN	1	X'00000001'

表 409. 定数の値 (続き)		
名前	10 進数値	16 進値
MQXC_MQCLOSE	2	X'00000002'
MQXC_MQGET	3	X'00000003'
MQXC_MQPUT	4	X'00000004'
MQXC_MQPUT1	5	X'00000005'
MQXC_MQINQ	6	X'00000006'
MQXC_MQSET	8	X'00000008'
MQXC_MQBACK	9	X'00000009'
MQXC_MQCMIT	10	X'0000000A'

MQXCC_* (出口応答)

表 410. 定数の値		
名前	10 進数値	16 進値
MQXCC_OK	0	X'00000000'
MQXCC_SUPPRESS_FUNCTION	-1	X'FFFFFFFF'
MQXCC_SKIP_FUNCTION	-2	X'FFFFFFFE'
MQXCC_SEND_AND_REQUEST_SEC_MSG	-3	X'FFFFFFFD'
MQXCC_SEND_SEC_MSG	-4	X'FFFFFFFC'
MQXCC_SUPPRESS_EXIT	-5	X'FFFFFFFB'
MQXCC_CLOSE_CHANNEL	-6	X'FFFFFFFA'
MQXCC_REQUEST_ACK	-7	X'FFFFFFF9'
MQXCC_FAILED	-8	X'FFFFFFF8'

MQXDR_* (出口応答)

表 411. 定数の値		
名前	10 進数値	16 進値
MQXDR_OK	0	X'00000000'
MQXDR_CONVERSION_FAILED	1	X'00000001'

MQXE_* (環境)

表 412. 定数の値		
名前	10 進数値	16 進値
MQXE_OTHER	0	X'00000000'
MQXE_MCA	1	X'00000001'
MQXE_MCA_SVRCONN	2	X'00000002'
MQXE_COMMAND_SERVER	3	X'00000003'
MQXE_MQSC	4	X'00000004'

MQXEPO_* (エントリー・ポイント登録オプション構造体および出口オプション)

エントリー・ポイント登録オプション構造体

表 413. 定数の構造	
名前	構造体
MQXEPO_STRUC_ID	"XEPO"
MQXEPO_STRUC_ID_ARRAY	'X','E','P','O'

注: 記号-は、単一のブランク文字を表します。

表 414. 定数の値		
名前	10 進数値	16 進値
MQXEPO_VERSION_1	1	X'00000001'
MQXEPO_CURRENT_VERSION	1	X'00000001'

出口オプション

表 415. 定数の値		
名前	10 進数値	16 進値
MQXEPO_NONE	0	X'00000000'

MQXF_* (API 関数 ID)

表 416. 定数の値		
名前	10 進数値	16 進値
MQXF_INIT	1	X'00000001'
MQXF_TERM	2	X'00000002'
MQXF_CONN	3	X'00000003'
MQXF_CONNX	4	X'00000004'
MQXF_DISC	5	X'00000005'
MQXF_OPEN	6	X'00000006'
MQXF_CLOSE	7	X'00000007'
MQXF_PUT1	8	X'00000008'
MQXF_PUT	9	X'00000009'
MQXF_GET	10	X'0000000A'
MQXF_DATA_CONV_ON_GET	11	X'0000000B'
MQXF_INQ	12	X'0000000C'
MQXF_SET	13	X'0000000D'
MQXF_BEGIN	14	X'0000000E'
MQXF_CMIT	15	X'0000000F'
MQXF_BACK	16	X'00000010'
MQXF_STAT	18	X'00000012'
MQXF_CB	19	X'00000013'
MQXF_CTL	20	X'00000014'
MQXF_CALLBACK	21	X'00000015'

表 416. 定数の値 (続き)		
名前	10 進数値	16 進値
MQXF_SUB	22	X'00000016'
MQXF_SUBRQ	23	X'00000017'
MQXF_XACLOSE	24	X'00000018'
MQXF_XACOMMIT	25	X'00000019'
MQXF_XACOMplete	26	X'0000001A'
MQXF_XAEND	27	X'0000001B'
MQXF_XAFORGET	28	X'0000001C'
MQXF_XAOPEN	29	X'0000001D'
MQXF_XAPREPARE	30	X'0000001E'
MQXF_XARECOVER	31	X'0000001F'
MQXF_XAROLLBACK	32	X'00000020'
MQXF_XASTART	33	X'00000021'
MQXF_AXREG	34	X'00000022'
MQXF_AXUNREG	35	X'00000023'

MQXP_* (API 交差出口パラメーター構造体)

表 417. 定数の構造	
名前	構造体
MQXP_STRUC_ID	"XP↵"
MQXP_STRUC_ID_ARRAY	'X','P','↵','↵'

注：記号↵は、単一のブランク文字を表します。

表 418. 定数の値		
名前	10 進数値	16 進値
MQXP_VERSION_1	1	X'00000001'

MQXPDA_* (問題判別域)

表 419. 定数名と値	
名前	値
MQXPDA_NONE	X'00...00' (48 スル値)
MQXPDA_NONE_ARRAY	'\0','\0',... (48 スル値)

MQXPT_* (トランスポート・タイプ)

表 420. 定数の値		
名前	10 進数値	16 進値
MQXPT_ALL	-1	X'FFFFFFFF'
MQXPT_LOCAL	0	X'00000000'
MQXPT_LU62	1	X'00000001'
MQXPT_TCP	2	X'00000002'
MQXPT_NETBIOS	3	X'00000003'

表 420. 定数の値 (続き)		
名前	10 進数値	16 進値
MQXPT_SPX	4	X'00000004'
MQXPT_DECNET	5	X'00000005'
MQXPT_UDP	6	X'00000006'

MQXQH_* (伝送キュー・ヘッダー構造体)

表 421. 定数の構造	
名前	構造体
MQXQH_STRUC_ID	"XQH-"
MQXQH_STRUC_ID_ARRAY	'X','Q','H','-'

注: 記号-は、単一のブランク文字を表します。

表 422. 定数の値		
名前	10 進数値	16 進値
MQXQH_VERSION_1	1	X'00000001'
MQXQH_CURRENT_VERSION	1	X'00000001'

MQXR_* (出口の理由)

表 423. 定数の値		
名前	10 進数値	16 進値
MQXR_BEFORE	1	X'00000001'
MQXR_AFTER	2	X'00000002'
MQXR_CONNECTION	3	X'00000003'
MQXR_INIT	11	X'0000000B'
MQXR_TERM	12	X'0000000C'
MQXR_MSG	13	X'0000000D'
MQXR_XMIT	14	X'0000000E'
MQXR_SEC_MSG	15	X'0000000F'
MQXR_INIT_SEC	16	X'00000010'
MQXR_RETRY	17	X'00000011'
MQXR_AUTO_CLUSSDR	18	X'00000012'
MQXR_AUTO_RECEIVER	19	X'00000013'
MQXR_CLWL_OPEN	20	X'00000014'
MQXR_CLWL_PUT	21	X'00000015'
MQXR_CLWL_MOVE	22	X'00000016'
MQXR_CLWL_REPOS	23	X'00000017'
MQXR_CLWL_REPOS_MOVE	24	X'00000018'
MQXR_END_BATCH	25	X'00000019'
MQXR_ACK_RECEIVED	26	X'0000001A'
MQXR_AUTO_SVRCONN	27	X'0000001B'
MQXR_AUTO_CLUSRCVR	28	X'0000001C'

表 423. 定数の値 (続き)		
名前	10 進数値	16 進値
MQXR_SEC_PARMS	29	X'0000001D'

MQXR2_* (出口応答 2)

表 424. 定数の値		
名前	10 進数値	16 進値
MQXR2_PUT_WITH_DEF_ACTION	0	X'00000000'
MQXR2_PUT_WITH_DEF_USERID	1	X'00000001'
MQXR2_PUT_WITH_MSG_USERID	2	X'00000002'
MQXR2_USE_AGENT_BUFFER	0	X'00000000'
MQXR2_USE_EXIT_BUFFER	4	X'00000004'
MQXR2_DEFAULT_CONTINUATION	0	X'00000000'
MQXR2_CONTINUE_CHAIN	8	X'00000008'
MQXR2_SUPPRESS_CHAIN	16	X'00000010'
MQXR2_STATIC_CACHE	0	X'00000000'
MQXR2_DYNAMIC_CACHE	32	X'00000020'

MQXT_* (出口 ID)

表 425. 定数の値		
名前	10 進数値	16 進値
MQXT_API_CROSSING_EXIT	1	X'00000001'
MQXT_API_EXIT	2	X'00000002'
MQXT_CHANNEL_SEC_EXIT	11	X'0000000B'
MQXT_CHANNEL_MSG_EXIT	12	X'0000000C'
MQXT_CHANNEL_SEND_EXIT	13	X'0000000D'
MQXT_CHANNEL_RCV_EXIT	14	X'0000000E'
MQXT_CHANNEL_MSG_RETRY_EXIT	15	X'0000000F'
MQXT_CHANNEL_AUTO_DEF_EXIT	16	X'00000010'
MQXT_CLUSTER_WORKLOAD_EXIT	20	X'00000014'
MQXT_PUBSUB_ROUTING_EXIT	21	X'00000015'

MQXUA_* (出口のユーザー域値)

表 426. 定数名と値	
名前	値
MQXUA_NONE	X'00...00' (16 個のヌル)
MQXUA_NONE_ARRAY	'\0', '\0', ... (16 個のヌル)

MQXWD_* (出口待機記述子構造体)

表 427. 定数の構造	
名前	構造体
MQXWD_STRUC_ID	"XWD-"
MQXWD_STRUC_ID_ARRAY	'X', 'W', 'D', '-'

注: 記号-は、単一のブランク文字を表します。

表 428. 定数の値		
名前	10 進数値	16 進値
MQXWD_VERSION_1	1	X'00000001'

MQZAC_* (アプリケーション・コンテキスト構造体)

表 429. 定数の構造	
名前	構造体
MQZAC_STRUC_ID	"ZAC-"
MQZAC_STRUC_ID_ARRAY	'Z', 'A', 'C', '-'

注: 記号-は、単一のブランク文字を表します。

表 430. 定数の値		
名前	10 進数値	16 進値
MQZAC_VERSION_1	1	X'00000001'
MQZAC_CURRENT_VERSION	1	X'00000001'

MQZAD_* (権限データ構造体)

表 431. 定数の構造	
名前	構造体
MQZAD_STRUC_ID	"ZAD-"
MQZAD_STRUC_ID_ARRAY	'Z', 'A', 'D', '-'

注: 記号-は、単一のブランク文字を表します。

表 432. 定数の値		
名前	10 進数値	16 進値
MQZAD_VERSION_1	1	X'00000001'
MQZAD_VERSION_2	2	X'00000002'
MQZAD_CURRENT_VERSION	2	X'00000002'

MQZAET_* (インストール可能サービス・エンティティ・タイプ)

表 433. 定数の値		
名前	10 進数値	16 進値
MQZAET_NONE	0	X'00000000'
MQZAET_PRINCIPAL	1	X'00000001'
MQZAET_GROUP	2	X'00000002'

表 433. 定数の値 (続き)		
名前	10 進数値	16 進値
MQZAET_UNKNOWN	3	X'00000003'

MQZAO_* (インストール可能サービス許可)

表 434. 定数の値		
名前	10 進数値	16 進値
MQZAO_CONNECT	1	X'00000001'
MQZAO_BROWSE	2	X'00000002'
MQZAO_INPUT	4	X'00000004'
MQZAO_OUTPUT	8	X'00000008'
MQZAO_INQUIRE	16	X'00000010'
MQZAO_SET	32	X'00000020'
MQZAO_PASS_IDENTITY_CONTEXT	64	X'00000040'
MQZAO_PASS_ALL_CONTEXT	128	X'00000080'
MQZAO_SET_IDENTITY_CONTEXT	256	X'00000100'
MQZAO_SET_ALL_CONTEXT	512	X'00000200'
MQZAO_ALTERNATE_USER_AUTHORITY	1024	X'00000400'
MQZAO_PUBLISH	2048	X'00000800'
MQZAO_SUBSCRIBE	4096	X'00001000'
MQZAO_RESUME	8192	X'00002000'
MQZAO_ALL_MQI	16383	X'00003FFF'
MQZAO_CREATE	65536	X'00010000'
MQZAO_DELETE	131072	X'00020000'
MQZAO_DISPLAY	262144	X'00040000'
MQZAO_CHANGE	524288	X'00080000'
MQZAO_CLEAR	1048576	X'00100000'
MQZAO_CONTROL	2097152	X'00200000'
MQZAO_CONTROL_EXTENDED	4194304	X'00400000'
MQZAO_AUTHORIZE	8388608	X'00800000'
MQZAO_ALL_ADMIN	16646144	X'00FE0000'
MQZAO_ALL	16662527	X'00FE3FFF'
MQZAO_REMOVE	16777216	X'01000000'
MQZAO_NONE	0	X'00000000'

MQZAS_* (インストール可能サービスのサービス・インターフェース・バージョン)

表 435. 定数の値		
名前	10 進数値	16 進値
MQZAS_VERSION_1	1	X'00000001'
MQZAS_VERSION_2	2	X'00000002'
MQZAS_VERSION_3	3	X'00000003'

表 435. 定数の値 (続き)		
名前	10 進数値	16 進値
MQZAS_VERSION_4	4	X'00000004'
MQZAS_VERSION_5	5	X'00000005'
MQZAS_VERSION_6	6	X'00000006'

MQZAT_* (認証タイプ)

表 436. 定数の値		
名前	10 進数値	16 進値
MQZAT_INITIAL_CONTEXT	0	X'00000000'
MQZAT_CHANGE_CONTEXT	1	X'00000001'

MQZCI_* (インストール可能サービス継続標識)

表 437. 定数の値		
名前	10 進数値	16 進値
MQZCI_DEFAULT	0	X'00000000'
MQZCI_CONTINUE	0	X'00000000'
MQZCI_STOP	1	X'00000001'

MQZED_* (エンティティ・データ構造体)

表 438. 定数の構造	
名前	構造体
MQZED_STRUC_ID	"ZED↵"
MQZED_STRUC_ID_ARRAY	'Z','E','D','↵'

注: 記号↵は、単一の空白文字を表します。

表 439. 定数の値		
名前	10 進数値	16 進値
MQZED_VERSION_1	1	X'00000001'
MQZED_VERSION_2	2	X'00000002'
MQZED_CURRENT_VERSION	2	X'00000002'

MQZFP_* (解放パラメーター構造体)

表 440. 定数の構造	
名前	構造体
MQZFP_STRUC_ID	"ZFP↵"
MQZFP_STRUC_ID_ARRAY	'Z','F','P','↵'

注: 記号↵は、単一の空白文字を表します。

表 441. 定数の値		
名前	10 進数値	16 進値
MQZFP_VERSION_1	1	X'00000001'

表 441. 定数の値 (続き)		
名前	10 進数値	16 進値
MQZFP_CURRENT_VERSION	1	X'00000001'

MQZIC_* (アイデンティティ・コンテキスト構造体)

表 442. 定数の構造	
名前	構造体
MQZIC_STRUC_ID	"ZIC↵"
MQZIC_STRUC_ID_ARRAY	'Z','I','C','↵'

注：記号↵は、単一のブランク文字を表します。

表 443. 定数の値		
名前	10 進数値	16 進値
MQZIC_VERSION_1	1	X'00000001'
MQZIC_CURRENT_VERSION	1	X'00000001'

MQZID_* (サービスの関数 ID)

すべてのサービスに共通の関数 ID

表 444. 定数の値		
名前	10 進数値	16 進値
MQZID_INIT	0	X'00000000'
MQZID_TERM	1	X'00000001'

権限サービスの関数 ID

表 445. 定数の値		
名前	10 進数値	16 進値
MQZID_INIT_AUTHORITY	0	X'00000000'
MQZID_TERM_AUTHORITY	1	X'00000001'
MQZID_CHECK_AUTHORITY	2	X'00000002'
MQZID_COPY_ALL_AUTHORITY	3	X'00000003'
MQZID_DELETE_AUTHORITY	4	X'00000004'
MQZID_SET_AUTHORITY	5	X'00000005'
MQZID_GET_AUTHORITY	6	X'00000006'
MQZID_GET_EXPLICIT_AUTHORITY	7	X'00000007'
MQZID_REFRESH_CACHE	8	X'00000008'
MQZID_ENUMERATE_AUTHORITY_DATA	9	X'00000009'
MQZID_AUTHENTICATE_USER	10	X'0000000A'
MQZID_FREE_USER	11	X'0000000B'
MQZID_INQUIRE	12	X'0000000C'
MQZID_CHECK_PRIVILEGED	13	X'0000000D'

ネーム・サービスの関数 ID

表 446. 定数の値		
名前	10 進数値	16 進値
MQZID_INIT_NAME	0	X'00000000'
MQZID_TERM_NAME	1	X'00000001'
MQZID_LOOKUP_NAME	2	X'00000002'
MQZID_INSERT_NAME	3	X'00000003'
MQZID_DELETE_NAME	4	X'00000004'

ユーザー ID サービスの関数 ID

表 447. 定数の値		
名前	10 進数値	16 進値
MQZID_INIT_USERID	0	X'00000000'
MQZID_TERM_USERID	1	X'00000001'
MQZID_FIND_USERID	2	X'00000002'

MQZIO_* (インストール可能サービス初期化オプション)

表 448. 定数の値		
名前	10 進数値	16 進値
MQZIO_PRIMARY	0	X'00000000'
MQZIO_SECONDARY	1	X'00000001'

MQZNS_* (ネーム・サービス・インターフェース・バージョン)

表 449. 定数の値		
名前	10 進数値	16 進値
MQZNS_VERSION_1	1	X'00000001'

MQZSE_* (インストール可能サービス列挙開始標識)

表 450. 定数の値		
名前	10 進数値	16 進値
MQZSE_START	1	X'00000001'
MQZSE_CONTINUE	0	X'00000000'

MQZSL_* (インストール可能サービス・セクター標識)

表 451. 定数の値		
名前	10 進数値	16 進値
MQZSL_NOT_RETURNED	0	X'00000000'
MQZSL_RETURNED	1	X'00000001'

MQZTO_* (インストール可能サービス終了オプション)

表 452. 定数の値		
名前	10 進数値	16 進値
MQZTO_PRIMARY	0	X'00000000'
MQZTO_SECONDARY	1	X'00000001'

MQZUS_* (ユーザー ID サービス・インターフェース・バージョン)

表 453. 定数の値		
名前	10 進数値	16 進値
MQZUS_VERSION_1	1	X'00000001'

MQI で使用されるデータ・タイプ

Message Queue Interface (MQI) で使用できるデータ・タイプに関する情報。各データ・タイプと関連する言語の説明、フィールド、および言語宣言。

MQI のデータ型とプログラミング

基本データ型と構造化データ型について、また C 言語、COBOL、High Level Assembler を用いたプログラミングを通じて、MQI の使い方をご紹介します。

基本データ・タイプ

MQI または出口関数で使用するデータ型に関する情報。これらは詳細に説明され、その後、サポートされているプログラミング言語で基本データ型を宣言する方法を示す例が続きます。

MQI または終了関数で使用するデータ型は、以下のいずれかです：

- 基本データ・タイプ
- 基本データ・タイプの集合体 (配列または構造体)

MQI または終了関数では、以下の基本データ型が使用されます：

表 454. 基本データ・タイプ名、タイプ、および説明		
基本データ・タイプの名前	データ・タイプ	説明
MQBOOL	Boolean	MQBOOL データ・タイプはブール値を表します。値 0 は偽を表します。その他の値は真を表します。MQBOOL は MQLONG データ・タイプに関する限り位置合わせしなければなりません。

表 454. 基本データ・タイプ名、タイプ、および説明 (続き)

基本データ・タイプの名前	データ・タイプ	説明
MQBYTE	Byte	MQBYTE データ・タイプは、1 バイトのデータを表します。バイトに対して特定の解釈は設定されません。単にビット・ストリングとして処理されるだけで、2 進数または文字として処理されることはありません。特殊な位置合わせは必要ありません。 MQBYTE データが、異なる文字セットやエンコード方式を使用するキュー・マネージャー間で送信される場合、MQBYTE データは変換されません。MQMD 構造体の <i>MsgId</i> および <i>CorrelId</i> フィールドがこれにあたります。 MQBYTE の配列は、キュー・マネージャーに認識されない主記憶領域を表すために使用されることがあります。例えば、その領域にアプリケーション・メッセージ・データまたは構造体が入っている可能性がある場合です。この領域の境界合わせは、その領域に含まれるデータの性質と合っていなければなりません。 C プログラミング言語では、MQBYTE の配列として示されている関数パラメーターには任意のデータ・タイプを使用できます。なぜならば、この種のパラメーターの受け渡しは常にアドレスに基づいて行われ、C 言語では、関数パラメーターは <code>void</code> を示すポインターとして宣言されるからです。

表 454. 基本データ・タイプ名、タイプ、および説明 (続き)		
基本データ・タイプの名前	データ・タイプ	説明
MQBYTEn	n バイトのストリング	各 MQBYTEn データ・タイプは n バイトのストリングを表します。 n は、8、16、24、32、40、または 128 のいずれかの値です。各バイトは MQBYTE データ・タイプにより記述されます。特殊な位置合わせは必要ありません。 バイト・ストリング内のデータがストリングの定義長より短い場合は、ストリングの定義長に達するまで、データをヌルで埋める必要があります。 キュー・マネージャーは、(例えば MQGET 呼び出しで) アプリケーションにバイト・ストリングを戻すときは、ストリングの定義長に達するまでヌルを埋め込みます。 バイト・ストリング・フィールドの長さを定義する名前定数を使用することができます。これらは以下に列挙されています 7 ページの『定数』
MQCHAR	文字	MQCHAR データ・タイプは、1 バイト文字か、2 バイトまたはマルチバイト文字の 1 バイトを表します。特殊な位置合わせは必要ありません。 MQCHAR データが、異なる文字セットやエンコード方式を使用するキュー・マネージャー間で送信される場合は、普通、データが正しく解釈されるように MQCHAR データを変換する必要があります。MQMD 構造体の MQCHAR データの場合は、キュー・マネージャーが自動的にこれを行います。アプリケーション・メッセージ・データ内の MQCHAR データの変換は、MQGET 呼び出しの際に指定される MQGMO_CONVERT オプションによって制御されます。このオプションの詳細については 326 ページの『MQGMO - 読み取りメッセージ・オプション』 を参照してください。

表 454. 基本データ・タイプ名、タイプ、および説明 (続き)

基本データ・タイプの名前	データ・タイプ	説明
MQCHARn	n 文字のストリング	各 MQCHARn データ・タイプは、n 文字のストリングを表します。n は、4、8、12、20、28、32、48、64、128、または 256 のいずれかの値です。各文字は MQCHAR データ・タイプにより定義されます。特殊な位置合わせは必要ありません。 ストリング内のデータがストリングの定義長より短い場合は、ストリングの定義長になるまで、データをブランクで埋める必要があります。ブランクを埋め込む代わりにヌル文字を使用して、ストリングを途中で終了させることができる場合があります。この場合、ヌル文字とそれに続く文字は、ストリングの定義長に達するまでブランクとして取り扱われます。ヌル文字を使用できる位置は、呼び出し記述およびデータ・タイプ記述に示されています。 キュー・マネージャーは、(例えば MQGET 呼び出しで) アプリケーションに文字ストリングを戻すときは常に、ストリングの定義長に達するまでブランクを埋め込みます。キュー・マネージャーは、ストリングを区切るためにヌル文字を使用することはありません。 文字ストリング・フィールドの長さを定義する 名前定数を使用することができます。7 ページの『定数』にリストしています。
MQFLOAT32	32 ビットの浮動小数点数	MQFLOAT32 データ・タイプは、標準の IEEE 浮動小数点形式を使用して表される 32 ビット浮動小数点数です。MQFLOAT32 は、4 バイトの境界に位置合わせされていなければなりません。 z/OS 上で C の MQFLOAT32 を使用するには、FLOAT(IEEE) コンパイラー・フラグを使用する必要があります。 COBOL の MQFLOAT32 の使用は、IEEE 形式の浮動小数点数をサポートするコンパイラーに制限されています。FLOAT(NATIVE) コンパイラー・フラグを使用する必要が生じることがあります。

表 454. 基本データ・タイプ名、タイプ、および説明 (続き)		
基本データ・タイプの名前	データ・タイプ	説明
MQFLOAT64	64 ビットの浮動小数点数	MQFLOAT64 データ・タイプは、標準の IEEE 浮動小数点形式を使用して表される 64 ビット浮動小数点数です。MQFLOAT64 は、8 バイトの境界に位置合わせされていなければなりません。 z/OS 上で C の MQFLOAT64 を使用するには、FLOAT(IEEE) コンパイラー・フラグを使用する必要があります。 COBOL の MQFLOAT64 の使用は、IEEE 形式の浮動小数点数をサポートするコンパイラーに制限されています。FLOAT(NATIVE) コンパイラー・フラグを使用する必要が生じることがあります。
MQHCONFIG	構成ハンドル	MQHCONFIG データ・タイプは、構成ハンドル、つまり特定のインストール可能なサービス用に構成されているコンポーネントを表します。構成ハンドルは、その本来の境界に位置合わせされる必要があります。 アプリケーションが、このハンドル内に保管されるデータの形式に依存してはなりません。有効な場合、この値は以降の MQI 呼び出しで使用可能となりますが、この目的以外のいかなる意味も持ちません。
MQHCONN	接続ハンドル	MQHCONN データ・タイプは、接続ハンドル、つまり特定のキュー・マネージャーへの接続を表します。接続ハンドルは、4 バイトの境界に位置合わせされていなければなりません。 アプリケーションが、このハンドル内に保管されるデータの形式に依存してはなりません。有効な場合、この値は以降の MQI 呼び出しで使用可能となりますが、この目的以外のいかなる意味も持ちません。

表 454. 基本データ・タイプ名、タイプ、および説明 (続き)		
基本データ・タイプの名前	データ・タイプ	説明
MQHMSG	メッセージ・ハンドル	MQHMSG データ・タイプは、メッセージにアクセスできるようにするメッセージ・ハンドルを表します。メッセージ・ハンドルは、8 バイト境界に位置合わせされていなければなりません。 アプリケーションが、このハンドル内に保管されるデータの形式に依存してはなりません。有効な場合、この値は以降の MQI 呼び出しで使用可能となりますが、この目的以外のいかなる意味も持ちません。
MQHOBJ	オブジェクト・ハンドル	MQHOBJ データ・タイプは、オブジェクトへのアクセスを提供するオブジェクト・ハンドルを表します。オブジェクト・ハンドルは、4 バイトの境界に位置合わせされていなければなりません。 アプリケーションが、このハンドル内に保管されるデータの形式に依存してはなりません。有効な場合、この値は以降の MQI 呼び出しで使用可能となりますが、この目的以外のいかなる意味も持ちません。
MQINT8	8 ビットの符号付き整数	MQINT8 データ・タイプは 8 ビットの符号付き整数で、コンテキストにより特に制限されていない限り、-128 から +127 までの範囲内の任意の値をとることができます。
MQINT16	16 ビットの符号付き整数	MQINT16 データ・タイプは 16 ビットの符号付き整数で、コンテキストにより特に制限されていない限り、-32 768 から +32 767 までの範囲内の任意の値をとることができます。MQINT16 は、2 バイトの境界に位置合わせされていなければなりません。
MQINT32	32 ビットの符号付き整数	MQINT32 データ・タイプは 32 ビットの符号付き 2 進整数で、コンテキストにより特に制限されていない限り、-2 147 483 648 から +2 147 483 647 の範囲内の任意の値をとることができます。 MQLONG の定義を参照してください。

表 454. 基本データ・タイプ名、タイプ、および説明 (続き)		
基本データ・タイプの名前	データ・タイプ	説明
MQINT64	64 ビットの符号付き整数	MQINT64 データ・タイプは 64 ビットの符号付き整数で、コンテキストにより特に制限されていない限り、-9 223 372 036 854 775 808 から +9 223 372 036 854 775 807 の範囲内に任意の値をとることができます。 COBOL の場合、有効範囲は -999 999 999 999 999 から +999 999 999 999 999 に制限されます。64 ビットの整数は、8 バイトの境界に位置合わせされていなければなりません。
MQLONG	32 ビットの符号付き整数	MQLONG データ・タイプは 32 ビットの符号付き 2 進整数で、コンテキストにより特に制限されていない限り、-2 147 483 648 から +2 147 483 647 の範囲内の任意の値をとることができます。 COBOL では、有効範囲は -999 999 999 から +999 999 999 に制限されています。MQLONG は、4 バイトの境界に位置合わせされていなければなりません。
MQPID	プロセス ID	IBM MQ プロセス ID。 これは、MQ のトレース・ダンプおよび FFST™ ダンプで使用される ID と同じ ID ですが、オペレーティング・システムのプロセス ID とは異なる場合があります。
MQPTR	ポインター	MQPTR データ・タイプは、すべてのタイプのデータのアドレスです。ポインターは、その本来の境界に位置合わせされていなければなりません。これは、IBM i では 16 バイトの境界、他のプラットフォームでは 8 バイトの境界になります。 プログラム言語の中には、タイプ付きポインターをサポートするものがあります。これは MQI でもまれに使用されることがあります (例えば、C プログラミング言語の PMQCHAR および PMQLONG)。

表 454. 基本データ・タイプ名、タイプ、および説明 (続き)		
基本データ・タイプの名前	データ・タイプ	説明
MQTID	スレッド ID	IBM MQ スレッド ID。 これは、MQ のトレース・ダンプおよび FFST™ ダンプで使用される ID と同じ ID ですが、オペレーティング・システムのスレッド ID とは異なる場合があります。
MQUINT8	8 ビットの符号なし整数	MQUINT8 データ・タイプは 8 ビットの符号なし整数で、コンテキストにより特に制限されていない限り、0 から +255 までの範囲内の任意の値をとることができます。
MQUINT16	16 ビットの符号なし整数	MQUINT16 データ・タイプは 16 ビットの符号なし整数で、コンテキストにより特に制限されていない限り、0 から +65 535 までの範囲内の任意の値をとることができます。 MQUINT16 は、2 バイトの境界に位置合わせされていなければなりません。
MQUINT32	32 ビットの符号なし整数	MQUINT32 データ・タイプは 32 ビットの符号なし 2 進整数です。 <u>MQULONG</u> の定義を参照してください。
MQUINT64	64 ビットの符号なし整数	MQUINT64 データ・タイプは 64 ビットの符号なし整数で、コンテキストにより特に制限されていない限り、0 から +18 446 744 073 709 551 615 の範囲内の任意の値をとることができます。 COBOL の場合、有効範囲は 0 から +999 999 999 999 999 999 に制限されます。64 ビットの整数は、8 バイトの境界に位置合わせされていなければなりません。
MQULONG	32 ビットの符号なし整数	MQULONG データ・タイプは 32 ビットの符号なし 2 進整数で、コンテキストにより特に制限されていない限り、0 から +4 294 967 294 の範囲内の任意の値をとることができます。 COBOL では、有効範囲は 0 から +999 999 999 に制限されています。MQULONG は、4 バイトの境界に位置合わせされていなければなりません。

表 454. 基本データ・タイプ名、タイプ、および説明 (続き)		
基本データ・タイプの名前	データ・タイプ	説明
PMQACH	ポインター	タイプ MQACH のデータ構造体へのポインター
PMQAIR	ポインター	タイプ MQAIR のデータ構造体へのポインター
PMQAXC	ポインター	タイプ MQAXC のデータ構造体へのポインター
PMQAXP	ポインター	タイプ MQAXP のデータ構造体へのポインター
PMQBMHO	ポインター	タイプ MQBMHO のデータ構造体へのポインター
PMQBO	ポインター	タイプ MQBO のデータ構造体へのポインター
PMQBOOL	ポインター	タイプ MQBOOL のデータへのポインター
PMQBYTE	ポインター	タイプ MQBYTE のデータへのポインター
PMQBYTEn	ポインター	MQBYTEn のデータ・タイプへのポインター。n は 8、16、24、32、40、128 のいずれかです。
PMQCBC	ポインター	タイプ MQCBC のデータ構造体へのポインター
PMQCBD	ポインター	タイプ MQCBD のデータ構造体へのポインター
PMQCHAR	ポインター	タイプ MQCHAR のデータへのポインター
PMQCHARN	ポインター	MQCHARN のデータ・タイプへのポインター。n は 4、8、12、20、28、32、48、64、128、256、264 のいずれかです。
PMQCHARV	ポインター	タイプ MQCHARV のデータ構造体へのポインター
PMQCIH	ポインター	タイプ MQCIH のデータ構造体へのポインター
PMQCMHO	ポインター	タイプ MQCMHO のデータ構造体へのポインター
PMQCNO	ポインター	タイプ MQCNO のデータ構造体へのポインター
PMQCSP	ポインター	タイプ MQCSP のデータ構造体へのポインター
PMQCTLO	ポインター	タイプ MQCTLO のデータ構造体へのポインター
PMQDH	ポインター	タイプ MQDH のデータ構造体へのポインター

表 454. 基本データ・タイプ名、タイプ、および説明 (続き)		
基本データ・タイプの名前	データ・タイプ	説明
PMQDHO	ポインター	タイプ MQDHO のデータ構造体へのポインター
PMQDLH	ポインター	タイプ MQDLH のデータ構造体へのポインター
PMQDMHO	ポインター	タイプ MQDMHO のデータ構造体へのポインター
PMQDMPO	ポインター	タイプ MQDMPO のデータ構造体へのポインター
PMQEPH	ポインター	タイプ MQEPH のデータ構造体へのポインター
PMQFLOAT32	ポインター	タイプ MQFLOAT32 のデータ構造体へのポインター
PMQFLOAT64	ポインター	タイプ MQFLOAT64 のデータ構造体へのポインター
PMQFUNC	ポインター	関数へのポインター
PMQGMO	ポインター	タイプ MQGMO のデータ構造体へのポインター
PMQHCONFIG	ポインター	タイプ MQHCONFIG のデータへのポインター
PMQHCONN	ポインター	タイプ MQHCONN のデータへのポインター
PMQHMSG	ポインター	タイプ MQHMSG のデータへのポインター
PMQHOBJ	ポインター	タイプ MQHOBJ のデータへのポインター
PMQIIH	ポインター	タイプ MQIIH のデータ構造体へのポインター
PMQIMPO	ポインター	タイプ MQIMPO のデータ構造体へのポインター
PMQINT8	ポインター	タイプ MQINT8 のデータへのポインター
PMQINT16	ポインター	タイプ MQINT16 のデータへのポインター
PMQINT32	ポインター	タイプ MQINT32 のデータへのポインター
PMQINT64	ポインター	タイプ MQINT64 のデータへのポインター
PMQLONG	ポインター	タイプ MQLONG のデータへのポインター
PMQMD	ポインター	タイプ MQMD の構造体へのポインター

表 454. 基本データ・タイプ名、タイプ、および説明 (続き)		
基本データ・タイプの名前	データ・タイプ	説明
PMQMDE	ポインター	タイプ MQMDE のデータ構造体へのポインター
PMQMD1	ポインター	タイプ MQMD1 のデータ構造体へのポインター
PMQMD2	ポインター	タイプ MQMD2 のデータ構造体へのポインター
PMQMHBO	ポインター	タイプ MQMHBO のデータ構造体へのポインター
PMQOD	ポインター	タイプ MQOD のデータ構造体へのポインター
PMQOR	ポインター	タイプ MQOR のデータ構造体へのポインター
PMQPD	ポインター	タイプ MQPD のデータ構造体へのポインター
PMQPID	ポインター	プロセス ID へのポインター
PMQMD	ポインター	タイプ MQMD のデータ構造体へのポインター
PMQPMO	ポインター	タイプ MQPMO のデータ構造体へのポインター
PMQPTR	ポインター	タイプ MQPTR のデータへのポインター
PMQRFH	ポインター	タイプ MQRFH のデータ構造体へのポインター
PMQRFH2	ポインター	タイプ MQRFH2 のデータ構造体へのポインター
PMQRMH	ポインター	タイプ MQRMH のデータ構造体へのポインター
PMQRR	ポインター	タイプ MQRR のデータ構造体へのポインター
PMQSCO	ポインター	タイプ MQSCO のデータ構造体へのポインター
PMQSD	ポインター	タイプ MQSD のデータ構造体へのポインター
PMQSMPO	ポインター	タイプ MQSMPO のデータ構造体へのポインター
PMQSRO	ポインター	タイプ MQSRO のデータ構造体へのポインター
PMSSTS	ポインター	タイプ MQSTS のデータ構造体へのポインター
PMQTID	ポインター	スレッド ID へのポインター
PMQTM	ポインター	タイプ MQTM のデータ構造体へのポインター

表 454. 基本データ・タイプ名、タイプ、および説明 (続き)		
基本データ・タイプの名前	データ・タイプ	説明
PMQTMCM2	ポインター	タイプ MQTMCM2 のデータ構造体へのポインター
PMQUINT8	ポインター	データ・タイプ MQUINT8 へのポインター
PMQUINT16	ポインター	データ・タイプ MQUINT16 へのポインター
PMQUINT32	ポインター	データ・タイプ MQUINT32 へのポインター
PMQUINT64	ポインター	データ・タイプ MQUINT64 へのポインター
PMQULONG	ポインター	データ・タイプ MQULONG へのポインター
PMQVOID	ポインター	
PMQWIH	ポインター	タイプ MQWIH のデータ構造体へのポインター
PMQXQH	ポインター	タイプ MQXQH のデータ構造体へのポインター

C のデータ型宣言

表 455. C データ・タイプ名と表記	
データ・タイプ	表現
MQBOOL	<code>typedef MQLONG MQBOOL;</code>
MQBYTE	<code>typedef unsigned char MQBYTE;</code>
MQBYTE8	<code>typedef MQBYTE MQBYTE8[8];</code>
MQBYTE16	<code>typedef MQBYTE MQBYTE16[16];</code>
MQBYTE24	<code>typedef MQBYTE MQBYTE24[24];</code>
MQBYTE32	<code>typedef MQBYTE MQBYTE32[32];</code>
MQBYTE40	<code>typedef MQBYTE MQBYTE40[40];</code>
MQCHAR	<code>typedef char MQCHAR;</code>
MQCHAR4	<code>typedef MQCHAR MQCHAR4[4];</code>
MQCHAR8	<code>typedef MQCHAR MQCHAR8[8];</code>

表 455. C データ・タイプ名と表記 (続き)	
データ・タイプ	表現
MQCHAR12	<code>typedef MQCHAR MQCHAR12[12];</code>
MQCHAR20	<code>typedef MQCHAR MQCHAR20[20];</code>
MQCHAR28	<code>typedef MQCHAR MQCHAR28[28];</code>
MQCHAR32	<code>typedef MQCHAR MQCHAR32[32];</code>
MQCHAR48	<code>typedef MQCHAR MQCHAR48[48];</code>
MQCHAR64	<code>typedef MQCHAR MQCHAR64[64];</code>
MQCHAR128	<code>typedef MQCHAR MQCHAR128[128];</code>
MQCHAR256	<code>typedef MQCHAR MQCHAR256[256];</code>
MQFLOAT32	<code>typedef float MQFLOAT32;</code>
MQFLOAT64	<code>typedef double MQFLOAT64;</code>
MQHCONFIG	<code>typedef void MQPOINTER MQHCONFIG;</code>
MQHCONN	<code>typedef MQLONG MQHCONN;</code>
MQHOBJ	<code>typedef MQLONG MQHOBJ;</code>
MQINT8	<code>typedef signed char MQINT8;</code>
MQINT16	<code>typedef short MQINT16;</code>

表 455. C データ・タイプ名と表記 (続き)	
データ・タイプ	表現
MQINT64	UNIX 64 ビットの UNIX の場合: <pre>typedef long;</pre> AIX 32 AIX ビットでは: <pre>typedef int64_t;</pre> IBM i z/OS Linux Linux、IBM i、および z/OS の場合: <pre>typedef long long;</pre> Windows On Windows: <pre>typedef _int64;</pre>
MQLONG	IBM i On IBM i: <pre>typedef long MQLONG;</pre> z/OS ALW 他のプラットフォーム: <pre>if defined(MQ_64_BIT) typedef int MQLONG; else typedef long MQLONG;</pre>
MQPID	<pre>typedef MQLONG MQPID;</pre>
MQPTR	<pre>typedef void MQPOINTER MQPTR;</pre>
MQTID	<pre>typedef MQLONG MQTID;</pre>
MQUINT8	<pre>typedef unsigned char MQUINT8;</pre>
MQUINT16	<pre>typedef unsigned short MQUINT16;</pre>

表 455. C データ・タイプ名と表記 (続き)	
データ・タイプ	表現
MQUINT64	 64 ビットの UNIX の場合: <pre>typedef unsigned long;</pre>  32 AIX ビットでは : <pre>typedef uint64_t;</pre>    Linux, IBM i、および z/OS の場合: <pre>typedef unsigned long long;</pre>  On Windows: <pre>typedef unsigned _int64;</pre>
MQULONG	 On IBM i: <pre>typedef unsigned long MQULONG;</pre>   他のプラットフォーム: <pre>if defined(MQ_64_BIT) typedef unsigned int MQULONG; else typedef unsigned long MQULONG;</pre>
PMQBO	<pre>typedef MQBO MQPOINTER PMQBO;</pre>
PMQBOOL	<pre>typedef MQBOOL MQPOINTER PMQBOOL;</pre>
PMQBYTE	<pre>typedef MQBYTE MQPOINTER PMQBYTE;</pre>
PMQBYTE8	<pre>typedef MQBYTE8[8] MQPOINTER PMQBYTE8[8];</pre>
PMQBYTE16	<pre>typedef MQBYTE16[16] MQPOINTER PMQBYTE16[16];</pre>
PMQBYTE24	<pre>typedef MQBYTE24[24] MQPOINTER PMQBYTE24[24];</pre>
PMQBYTE32	<pre>typedef MQBYTE32[32] MQPOINTER PMQBYTE32[32];</pre>
PMQBYTE40	<pre>typedef MQBYTE40[40] MQPOINTER PMQBYTE40[40];</pre>
PMQBYTE128	<pre>typedef MQBYTE128[128] MQPOINTER PMQBYTE128[128];</pre>
PMQCHAR	<pre>typedef MQCHAR MQPOINTER PMQCHAR;</pre>

表 455. C データ・タイプ名と表記 (続き)	
データ・タイプ	表現
PMQCHAR4	<code>typedef MQCHAR4[4] MQPOINTER PMQCHAR4[4];</code>
PMQCHAR8	<code>typedef MQCHAR8[8] MQPOINTER PMQCHAR8[8];</code>
PMQCHAR12	<code>typedef MQCHAR12[12] MQPOINTER PMQCHAR12[12];</code>
PMQCHAR20	<code>typedef MQCHAR20[20] MQPOINTER PMQCHAR20[20];</code>
PMQCHAR28	<code>typedef MQCHAR28[28] MQPOINTER PMQCHAR28[28];</code>
PMQCHAR32	<code>typedef MQCHAR32[32] MQPOINTER PMQCHAR32[32];</code>
PMQCHAR48	<code>typedef MQCHAR48[48] MQPOINTER PMQCHAR48[48];</code>
PMQCHAR64	<code>typedef MQCHAR64[64] MQPOINTER PMQCHAR64[64];</code>
PMQCHAR128	<code>typedef MQCHAR128[128] MQPOINTER PMQCHAR128[128];</code>
PMQCHAR256	<code>typedef MQCHAR256[256] MQPOINTER PMQCHAR256[256];</code>
PMQCHAR264	<code>typedef MQCHAR264[264] MQPOINTER PMQCHAR264[264];</code>
PMQCIH	<code>typedef MQCIH MQPOINTER PMQCIH;</code>
PMQCNO	<code>typedef MQCNO MQPOINTER PMQCNO;</code>
PMQDLH	<code>typedef MQDLH MQPOINTER PMQDLH;</code>
PMQFUNC	<code>typedef void MQPOINTER PMQFUNC;</code>
PMQFLOAT32	<code>typedef MQFLOAT32 MQPOINTER PMQFLOAT32;</code>
PMQFLOAT64	<code>typedef MQFLOAT64 MQPOINTER PMQFLOAT64;</code>
PMQGM0	<code>typedef MQGM0 MQPOINTER PMQGM0;</code>
PMQHCONFIG	<code>typedef MQHCONFIG MQPOINTER PMQHCONFIG;</code>
PMQHCONN	<code>typedef MQHCONN MQPOINTER PMQHCONN;</code>
PMQHOBJ	<code>typedef MQHOBJ MQPOINTER PMQHOBJ;</code>

表 455. C データ・タイプ名と表記 (続き)	
データ・タイプ	表現
PMQIIH	<code>typedef MQIIH MQPOINTER PMQIIH;</code>
PMQINT8	<code>typedef MQINT8 MQPOINTER PMQINT8;</code>
PMQINT16	<code>typedef MQINT16 MQPOINTER PMQINT16;</code>
PMQLONG	<code>typedef MQLONG MQPOINTER PMQLONG;</code>
PMQMD	<code>typedef MQMD MQPOINTER PMQMD;</code>
PMQMD1	<code>typedef MQMD1[1] MQPOINTER PMQMD1[1];</code>
PMQMDE	<code>typedef MQMDE MQPOINTER PMQMDE;</code>
PMQOD	<code>typedef MQOD MQPOINTER PMQOD;</code>
PMQPMO	<code>typedef MQPMO MQPOINTER PMQPMO;</code>
PMQPTR	<code>typedef MQPTR MQPOINTER PMQPTR;</code>
PMQRFH	<code>typedef MQRFH MQPOINTER PMQRFH;</code>
PMQRFH2	<code>typedef MQRFH2[2] MQPOINTER PMQRFH2[2];</code>
PMQRMH	<code>typedef MQRMH MQPOINTER PMQRMH;</code>
PMQTM	<code>typedef MQTM MQPOINTER PMQTM;</code>
PMQTM2	<code>typedef MQTM2[2] MQPOINTER PMQTM2[2];</code>
PMQUINT8	<code>typedef MQUINT8 MQPOINTER PMQUINT8;</code>
PMQUINT16	<code>typedef MQUINT16 MQPOINTER PMQUINT16;</code>
PMQULONG	<code>typedef MQULONG MQPOINTER PMQULONG;</code>
PMQVOID	<code>typedef void MQPOINTER PMQVOID;</code>
PMQWIH	<code>typedef MQWIH MQPOINTER PMQWIH;</code>
PMQXQH	<code>typedef MQXQH MQPOINTER PMQXQH;</code>

表 455. C データ・タイプ名と表記 (続き)	
データ・タイプ	表現
PPMQBO	<code>typedef PMQBO MQPOINTER PPMQBO;</code>
PPMQBYTE	<code>typedef PMQBYTE MQPOINTER PPMQBYTE;</code>
PPMQCHAR	<code>typedef PMQCHAR MQPOINTER PPMQCHAR;</code>
PPMQCNO	<code>typedef PMQCNO MQPOINTER PPMQCNO;</code>
PPMQGMO	<code>typedef PMQGMO MQPOINTER PPMQGMO;</code>
PPMQHCONN	<code>typedef PMQHCONN MQPOINTER PPMQHCONN;</code>
PPMQHOBJ	<code>typedef PMQHOBJ MQPOINTER PPMQHOBJ;</code>
PPMQLONG	<code>typedef PMQLONG MQPOINTER PPMQLONG;</code>
PPMQMD	<code>typedef PMQMD MQPOINTER PPMQMD;</code>
PPMQOD	<code>typedef PMQOD MQPOINTER PPMQOD;</code>
PPMQPMO	<code>typedef PMQPMO MQPOINTER PPMQPMO;</code>
PPMQULONG	<code>typedef PMQULONG MQPOINTER PPMQULONG;</code>
PPMQVOID	<code>typedef PMQVOID MQPOINTER PPMQVOID;</code>
ここで、defined(MQ_64_BIT) は 64 ビット・プラットフォームを意味します。	

MQPOINTER マクロ変数の説明については、[203 ページ](#)の『データ・タイプ』を参照してください。

COBOL データ型の宣言

表 456. COBOL データ・タイプ名と表記	
データ・タイプ	表現
MQBOOL	<code>PIC S9(9) BINARY</code>
MQBYTE	<code>PIC X</code>
MQBYTE8	<code>PIC X(8)</code>
MQBYTE16	<code>PIC X(16)</code>

表 456. COBOL データ・タイプ名と表記 (続き)	
データ・タイプ	表現
MQBYTE24	PIC X(24)
MQBYTE32	PIC X(32)
MQBYTE40	PIC X(40)
MQCHAR	PIC X
MQCHAR4	PIC X(4)
MQCHAR8	PIC X(8)
MQCHAR12	PIC X(12)
MQCHAR20	PIC X(20)
MQCHAR28	PIC X(28)
MQCHAR32	PIC X(32)
MQCHAR48	PIC X(48)
MQCHAR64	PIC X(64)
MQCHAR128	PIC X(128)
MQCHAR256	PIC X(256)
MQFLOAT32	USAGE COMP-1
MQFLOAT64	USAGE COMP-2
MQHCONN	z/OS 上 PIC S9(9) COMP-5 その他のプラットフォームの場合 PIC S9(9) BINARY
MQHOBJ	PIC S9(9) BINARY
MQINT8	PIC S9(2) BINARY

表 456. COBOL データ・タイプ名と表記 (続き)	
データ・タイプ	表現
MQINT16	PIC S9(4) BINARY
MQINT64	PIC S9(18) BINARY
MQLONG	PIC S9(9) BINARY
MQPTR	POINTER
MQUINT8	PIC 9(2) BINARY
MQUINT16	PIC 9(4) BINARY
MQUINT64	PIC 9(18) BINARY
MQULONG	PIC 9(9) BINARY

PL/I データ型の宣言

表 457. PL/I データ・タイプ名と表記	
データ・タイプ	表現
MQBOOL	fixed bin(31)
MQBYTE	char(1)
MQBYTE8	char(8)
MQBYTE16	char(16)
MQBYTE24	char(24)
MQBYTE32	char(32)
MQBYTE40	char(40)
MQCHAR	char(1)
MQCHAR4	char(4)
MQCHAR8	char(8)
MQCHAR12	char(12)

表 457. PL/I データ・タイプ名と表記 (続き)	
データ・タイプ	表現
MQCHAR20	char(20)
MQCHAR28	char(28)
MQCHAR32	char(32)
MQCHAR48	char(48)
MQCHAR64	char(64)
MQCHAR128	char(128)
MQCHAR256	char(256)
MQFLOAT32	binary float(21) ieee
MQFLOAT64	binary float(52) ieee
MQHCONN	fixed bin(31)
MQHOBJ	fixed bin(31)
MQINT8	fixed bin(7)
MQINT16	fixed bin(15)
MQINT64	fixed bin(63)
MQLONG	fixed bin(31)
MQPTR	pointer
MQUINT8	fixed bin(8)
MQUINT16	fixed bin(16)
MQUINT64	fixed bin(64)
MQULONG	fixed bin(32)

High Level Assembler データ型の宣言

表 458. System/390 アセンブラーのデータ・タイプ名と表記	
データ・タイプ	表現
MQBOOL	DS F
MQBYTE	DS XL1
MQBYTE8	DS XL8
MQBYTE16	DS XL16
MQBYTE24	DS XL24
MQBYTE32	DS XL32
MQBYTE40	DS XL40
MQCHAR	DS CL1
MQCHAR4	DS CL4
MQCHAR8	DS CL8
MQCHAR12	DS CL12
MQCHAR20	DS CL20
MQCHAR28	DS CL28
MQCHAR32	DS CL32
MQCHAR48	DS CL48
MQCHAR64	DS CL64
MQCHAR128	DS CL128
MQCHAR256	DS CL256
MQFLOAT32	DS EB
MQFLOAT64	DS DB
MQHCONN	DS F

表 458. System/390 アセンブラーのデータ・タイプ名と表記 (続き)	
データ・タイプ	表現
MQHOB	DS F
MQINT8	DS XL1
MQINT16	DS H
MQINT64	DS D
MQLONG	DS F
MQPTR	DS F
MQUINT8	DS XL1
MQUINT16	DS H
MQUINT64	DS D
MQULONG	DS F

構造データ型

構造体データ型の概要、MQI の構造体を一貫してマッピングするためのルール、および各構造体データ型の説明で使用する規約。

- [199 ページの『MQI の呼び出しや終了関数で使用する構造体データ型の概要』](#)
- [200 ページの『メッセージ・データで使用する構造体データ・タイプの要約』](#)
- [201 ページの『MQI 構造体を一貫してマッピングするためのルール』](#)
- [201 ページの『各構造体データ・タイプ記述で使用する規則』](#)

MQI の呼び出しや終了関数で使用する構造体データ型の概要

表 459. MQI の呼び出しや終了関数で使用する構造体データ型		
構造体	説明	使用される呼び出し
MQACH	API 出口チェーン・ヘッダー	
MQAIR	認証情報レコード	MQCONN
MQAXC	API 出口コンテキスト	
MQAXP	API 出口パラメーター	
MQBMHO	バッファからメッセージ・ハンドルへの変換オプション	MQBUFMH
MQBO	開始オプション	MQBEGIN
MQCBD	コールバック記述子	MQCB

表 459. MQI の呼び出しや終了関数で使用する構造体データ型 (続き)		
構造体	説明	使用される呼び出し
MQCBO	バッグ作成オプション	mqCreateBag
MQCHARV	可変長ストリング	MQINQMP
MQCNO	接続オプション	MQCONN
MQCSP	セキュリティー・パラメーター。	MQCONN
MQCTLO	コールバック・オプション	MQCTL
MQDMPO	メッセージ・プロパティー削除オプション	MQDLTMP
MQGMO	読み取りメッセージ・オプション	MQGet
MQIMPO	メッセージ・プロパティー照会オプション	MQINQMP
MQMD	メッセージ記述子	MQBUFMH , MQMHBUE , MQCB , MQGET , MQPUT , MQPUT1
MQMHBO	メッセージ・ハンドルからバッファへの変換オプション	MQMHBUE
MQOD	オブジェクト記述子	MQOPEN , MQPUT1
MQOR	オブジェクト・レコード	MQOPEN , MQPUT1
MQPD	プロパティー記述子	MQSETMP
MQPMO	書き込みメッセージ・オプション	MQPUT , MQPUT1
MQPMR	書き込みメッセージ・レコード	MQPUT , MQPUT1
MQRR	応答レコード	MQOPEN , MQPUT , MQPUT1
MQSCO	TLS 構成オプション	MQCONN
MQSD	サブスクリプション記述子	MQSUB
MQSMPO	メッセージ・プロパティー設定オプション	MQSETMP
MQSRO	サブスクリプション要求オプション	MQSUBRQ
MQSTS	状況報告構造体	MQSTAT

メッセージ・データで使用する構造体データ・タイプの要約

表 460. メッセージ・データに使用される構造体データ・タイプ	
構造体	説明
MQCIH	CICS 情報ヘッダー
MQCFH	PCF ヘッダー
MQEPH	組み込み PCF ヘッダー
MQDH	配布ヘッダー
MQDLH	送達不能 (未配布メッセージ) ヘッダー
MQIIH	IMS 情報ヘッダー

表 460. メッセージ・データに使用される構造体データ・タイプ (続き)	
構造体	説明
MQMDE	拡張メッセージ記述子
MQRFH	規則およびフォーマット・ヘッダー
MQRFH2	規則および書式ヘッダー 2
MQRMH	参照メッセージ・ヘッダー
MQTM	トリガー・メッセージ
MQTMC2	トリガー・メッセージ (文字形式 2)
MQWIH	作業情報ヘッダー
MQXQH	伝送キュー・ヘッダー

注: MQDXP 構造体 (データ変換出口パラメーター) と、関連するデータ変換呼び出しについては [903 ページ](#) の『データ変換出口』を参照してください。

MQI 構造体を一貫してマッピングするためのルール

プログラミング言語は、構造体のサポートのレベルが異なるため、特定の規則を取り入れて、各プログラミング言語で矛盾なく MQI 構造体がマップされるようにしています。

- 構造体は、その本来の境界に位置合わせされる必要があります。
 - 一般に MQI 構造体は、4 バイトの境界で位置合わせする必要があります。
 - IBM i の場合、ポインターの含まれる構造体には 16 バイトの位置合わせが必要です。MQCNO、MQOD、MQPMO はこの種の構造体です。
- 構造内の各フィールドは、その本来の境界に位置合わせされる必要があります。
 - MQLONG と等しいデータ・タイプをもつフィールドは、4 バイトの境界で位置合わせされる必要があります。
 - MQPTR と等しいデータ・タイプを持つフィールドは、IBM i の場合は 16 バイトの境界で、その他の環境では 4 バイトの境界で位置合わせされる必要があります。
 - その他のフィールドは 1 バイトの境界に位置合わせされます。
- 構造体の長さは、その境界合わせの倍数にする必要があります。
 - 一般に、MQI 構造体の長さは、4 バイトの倍数となります。
 - IBM i の場合、ポインターの含まれる構造体の長さは 16 バイトの倍数になります。
- 必要に応じて埋め込みバイトまたはフィールドを追加し、確実に上記の規則に準拠するようにします。

各構造体データ・タイプ記述で使用される規則

各構造体データ・タイプの記述には、以下の情報が含まれます。

- 構造体の目的と使用法についての概説。
- 構造体の中のフィールドの説明。プログラム言語からは独立した形で説明します。
- サポートされる各プログラム言語における構造体の宣言方法の例。

各構造体データ・タイプの記述は、次に示すセクションで構成されています。

構造体名

構造体の名前。名前の後ろには、その構造体中のフィールドの要約が続きます。

概要

構造体の目的と使用法の簡潔な説明。

フィールド

フィールドの説明。各フィールドについて、フィールドの名前の後に、小括弧 () 内の基本データ・タイプが続きます。テキストでは、フィールド名はイタリック書体を使用して表示されます。例えば、*Version* のようになります。

また、フィールドの目的について説明し、そのフィールドが取ることのできる値のリストも示しています。定数の名前は、例えば、MQGMO_STRUC_ID のように英大文字で示されています。同じ接頭部をもつ一組の定数は、* 文字を使用して、MQIA_* のように示されています。

フィールドの記述の中で、以下の用語が使用されます。

入力

呼び出しを行うときに、ユーザーはフィールドに情報を与えます。

出力

呼び出しが完了または失敗したときに、キュー・マネージャーがそのフィールドに情報を戻します。

入出力

呼び出しを行うときにユーザーがフィールドに情報を与え、呼び出しが完了または失敗したときにキュー・マネージャーが情報を変更します。

初期値

MQI で与えられたデータ定義ファイルの中の各フィールドの初期値を示すテーブル。

C 宣言

C の場合の構造体の一般的な宣言。

COBOL 宣言

COBOL の場合の構造体の一般的な宣言。

PL/I 宣言

PL/I の場合の構造体の一般的な宣言。

高水準アセンブラ宣言

System/390 アセンブラ言語の場合の構造体の一般的な宣言。

Visual Basic の宣言

Visual Basic の構造体の一般的な宣言。

C プログラミング

C 言語から MQI を使用するための情報。

- [202 ページの『ヘッダー・ファイル』](#)
- [203 ページの『関数』](#)
- [203 ページの『未定義データ・タイプのパラメーター』](#)
- [203 ページの『データ・タイプ』](#)
- [204 ページの『2 進ストリングの取り扱い』](#)
- [204 ページの『文字ストリングの取り扱い』](#)
- [204 ページの『構造体の初期値』](#)
- [205 ページの『動的構造体の初期値』](#)
- [205 ページの『C++ からの使用』](#)
- [205 ページの『表記規則』](#)

ヘッダー・ファイル

表 461. C ヘッダー・ファイル	
ファイル	内容
CMQC	メイン MQI 用の関数原型、データ・タイプ、および名前付き定数

表 461. C ヘッダー・ファイル (続き)	
ファイル	内容
CMQXC	データ変換出口用の関数原型、データ・タイプ、および名前付き定数
CMQEC	メインの MQI、データ変換出口、およびインターフェース・エントリー・ポイント構造体 (CMQEC は CMQXC および CMQC を含む) 用の関数プロトタイプ、データ・タイプ、および名前付き定数
CMQSTRC	MQI の定数定義を同等のテキストに変換する関数。  重要:  IBM MQ 9.1 から z/OS に適用できます。このヘッダー・ファイルを使用するプログラムは、LONGNAME コンパイラー・オプションを使用してコンパイルする必要があります。

アプリケーションの移植性を向上させるために、次のように、ヘッダー・ファイルの名前を `#include` プリプロセッサ指示に小文字でコーディングしてください。

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

関数

アドレスによって渡されるパラメーターは、必ずしも関数を呼び出すたびに指定する必要はありません。

- 入力専用 のパラメーター、およびタイプが MQHCONN、MQHOBJ、または MQLONG のパラメーターを値で渡します。
- その他のパラメーターをすべてアドレスによって渡します。

特に必要なパラメーターがない場合は、パラメーター・データのアドレスの代わりに、ヌル・ポインターを関数呼び出しのパラメーターとして使用することができます。これが可能なパラメーターは、呼び出し記述子で識別されます。

関数の値としてパラメーターが戻されることはありません。これを C の用語で表現すれば、すべての関数は `void` を戻すということになります。

関数の属性は、MQENTRY マクロ変数によって定義されます。このマクロ変数の値は、環境に応じて異なります。

未定義データ・タイプのパラメーター

MQGET 関数、MQPUT 関数、および MQPUT1 関数の **Buffer** パラメーターには、未定義のデータ・タイプがあります。このパラメーターは、アプリケーションのメッセージ・データを送受信するために使用されます。

この種類のパラメーターは、C の例で MQBYTE の配列として示されています。この方法でパラメーターを宣言することは可能ですが、通常は、メッセージのデータ・レイアウトを記述する特定の構造体として宣言の方が便利です。実際の関数パラメーターは `void` を示すポインターとして宣言し、任意の種類のデータのアドレスを関数呼び出しのパラメーターとして指定します。

データ・タイプ

データ・タイプはすべて、C `typedef` 文を使って定義します。各データ・タイプごとに、対応するポインター・データ・タイプも定義します。ポインター・データ・タイプの名前は、基本データ・タイプ、またはポインターを指示するためにその接頭部に P の付いた構造体データ・タイプの名前です。ポインターの属性は、MQPOINTER マクロ変数を使って定義します。このマクロ変数の値は環境によって決まります。ポインターのデータ・タイプがどのように宣言されるのかを次に示します。

```
#define MQPOINTER *          /* depends on environment */
...
```

```
typedef MQLONG MQPOINTER PMQLONG; /* pointer to MQLONG */
typedef MQMD MQPOINTER PMQMD; /* pointer to MQMD */
```

2 進ストリングの取り扱い

バイナリー・データのストリングは、MQBYTEn データ・タイプの 1 つとして宣言します。

このタイプのフィールドをコピー、比較、または設定するときは、常に C 関数 **memcpy**、**memcmp**、または **memset** を使用します。例えば、

```
#include <string.h>
#include "cmqc.h"

MQMD MyMsgDesc;

memcpy(MyMsgDesc.MsgId,          /* set "MsgId" field to nulls */
       MQMI_NONE,               /* ...using named constant */
       sizeof(MyMsgDesc.MsgId));

memset(MyMsgDesc.CorrelId,       /* set "CorrelId" field to nulls */
       0x00,                   /* ...using a different method */
       sizeof(MQBYTE24));
```

ストリング関数 **strcpy**、**strcmp**、**strncpy**、および **strncmp** は使用しないでください。これらの関数は、MQBYTEn データ・タイプを使用して宣言されたデータについては正しく働かないからです。

文字ストリングの取り扱い

キュー・マネージャーは、文字データをアプリケーションに戻すときに、文字データがフィールドの定義された長さになるように必ずブランク文字を埋め込みます。キュー・マネージャーは、ヌル終了ストリングを戻しません。

したがって、このようなストリングをコピー、比較、または連結するときは、ストリング関数 **strncpy**、**strncmp**、**strncat** を使用します。

ストリングをヌル (**strcpy**、**strcmp**、**strcat**) で終了する必要があるストリング関数は使用しないでください。また、ストリングの長さを判別するために関数 **strlen** を使用しないでください。代わりに **sizeof** 関数を使用して、フィールドの長さを判別してください。

構造体の初期値

ヘッダー・ファイルには、メッセージ・キューイング構造体のインスタンスを宣言した時点でそれらの MQ 構造体に初期値を提供するために使用できる各種のマクロ変数が定義されています。

これらのマクロ変数では、MQxxx_DEFAULT の形式の名前を使用します。この MQxxx は構造体の名前を表します。使用方法は次のとおりです。

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

一部の文字フィールド (例えば、ほとんどの構造体にある *StrucId* フィールド、または MQMD にある *Format* フィールドなど) では、MQI は特定の有効値を定義します。各有効値について、次に示す 2 つのマクロ変数があります。

- 暗黙のヌルを除いた長さがフィールドの定義長に正確に一致するストリングとして値を定義するマクロ変数。例えば、MQMD の *Format* フィールドの場合、以下のマクロ変数が提供されます (↵ は単一のブランク文字を表します)。

```
#define MQFMT_STRING "MQSTR↵↵↵"
```

この形式は **memcpy** 関数および **memcmp** 関数で使用します。

- 値を文字の配列として定義するマクロ変数。このマクロ変数の名前は、ストリング形式の名前に接尾部「_ARRAY」が付けられます。以下に例を示します。


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

このフォームを使用して、MQMD_DEFAULT マクロ変数によって提供される値とは異なる値を持つ構造のインスタンスを宣言するときに、このフィールドを初期化します。(これは必ずしも必要ではありません。一部の環境では、両方の状態で値のストリング形式を使用することができます。しかし、配列形式は C++ プログラミング言語との互換性を確保するためには必須なので、宣言には配列形式を使用することができます。)

動的構造体の初期値

構造体に可変数のインスタンスが必要な場合、そのインスタンスは通常、calloc 関数または malloc 関数を使用して動的に取得されるメイン・ストレージの内部に作成されます。このような構造体内のフィールドを初期設定する場合は、次の技法を検討してください。

1. 構造体を初期化するために、適切な MQxxx_DEFAULT マクロ変数を使用して、構造体のインスタンスを宣言する。このインスタンスが他のインスタンスのモデルになります。

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

必要に応じて、宣言に static キーワードまたは auto キーワードをコーディングすると、存続時間中のモデル・インスタンスを静的または動的な状態にすることができます。

2. calloc 関数または malloc 関数を使用して、構造体の動的インスタンス用のストレージを取得します。

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

3. memcpy 関数を使用して、モデル・インスタンスを動的インスタンスにコピーします。

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

C++ からの使用

C++ プログラミング言語では、C++ コンパイラーの使用時にのみ組み込まれる次の追加文がヘッダー・ファイルに含まれています。

```
#ifdef __cplusplus  
extern "C" {  
#endif  
  
/* rest of header file */  
  
#ifdef __cplusplus  
}  
#endif
```

表記規則

この情報は、関数の呼び出し方とパラメーターの宣言方法を示しています。

パラメーターが、サイズの固定されていない配列となる場合があります。このような場合は、小文字の n を使用して数値定数を表しています。そのパラメーターの宣言をエンコードするときは、必要な数値で n を置き換えます。

COBOL プログラミング

プログラミング言語「COBOL」から「MQI」を利用するための情報。

- [206 ページの『COPY ファイル』](#)
- [207 ページの『構造体』](#)
- [207 ページの『ポインター』](#)
- [208 ページの『名前付き定数』](#)

COPY ファイル

MQI を使用する COBOL アプリケーション・プログラムの作成のために多様な COPY ファイルが提供されています。COPY ファイルには、名前付き定数を含むファイルが 2 つあり、各構造体についてファイルが 2 つずつあります。

各構造体は、初期値付きの形式と初期値なしの形式の 2 つの形式が提供されています。

- 初期値付きの構造体は、COBOL プログラムの WORKING-STORAGE SECTION で使用します。この構造体は、名前の接尾部に V (Value) の文字が付いた COPY ファイルに入っています。
- 初期値なしの構造体は、COBOL プログラムの LINKAGE SECTION で使用します。この構造体は、名前の接尾部に L (Linkage) の文字が付いた COPY ファイルに入っています。

COPY ファイルの概要は次の表にまとめられています。記載してあるファイルがすべての環境で利用できるわけではありません。

表 462. COBOL の COPY ファイル		
ファイル (初期値付き)	ファイル (初期値なし)	内容
CMQAIRV	CMQAIRL	認証情報レコード
CMQBOV	CMQBOL	開始オプション構造体
CMQCIHV	CMQCIHL	CICS 情報ヘッダー構造体
CMQCN OV	CMQCNOL	接続オプション構造体
CMQD HV	CMQDHL	配布ヘッダー構造体
CMQDL HV	CMQDLHL	送達不能ヘッダー構造体
CMQDXPV	CMQDXPL	データ変換出口パラメーター構造体
CMQGM OV	CMQGMOL	読み取りメッセージ・オプション構造体
CMQIIHV	CMQIIHL	IMS 情報ヘッダー構造体
CMQMDV	CMQMDL	メッセージ記述子構造体
CMQMDEV	CMQMDEL	拡張メッセージ記述子構造体
CMQMD1V	CMQMD1L	メッセージ記述子構造体 - バージョン 1
CMQODV	CMQODL	オブジェクト記述子構造体
CMQORV	CMQORL	オブジェクト・レコード構造体
CMQPM OV	CMQPMOL	書き込みメッセージ・オプション構造体
CMQRFHV	CMQRFHL	規則および書式ヘッダー構造体
CMQRFH2V	CMQRFH2L	規則および書式ヘッダー構造体、バージョン 2
CMQRMHV	CMQRMHL	参照メッセージ・ヘッダー構造体
CMQRRV	CMQRRL	応答レコード構造体
CMQSCOV	CMQSCOL	TLS 構成オプション
CMQTMV	CMQTML	トリガー・メッセージ構造体
CMQTM CV	CMQTMCL	トリガー・メッセージ構造体 (文字形式)
CMQTM C2V	CMQTM C2L	トリガー・メッセージ構造体 (文字形式) - バージョン 2

表 462. COBOL の COPY ファイル (続き)		
ファイル (初期値付き)	ファイル (初期値なし)	内容
CMQWIHV	CMQWIHL	作業情報ヘッダー構造体
CMQXQHV	CMQXQHL	伝送キュー・ヘッダー構造体
CMQV	-	メイン MQI 用の名前付き定数
CMQXV	-	データ変換出口用の名前付き定数
CMQMD2V	CMQMD2L	メッセージ記述子構造体 - バージョン 2

構造体

COPY ファイルでは、各構造体宣言は、レベル 10 の項目で始めます。この項目で開始すると、レベル 01 の宣言をコーディングし、構造体宣言の後続部分で COPY 文を使用してコピーすることにより、構造体の複数のインスタンスを宣言することができます。適切なインスタンスを参照するには、IN キーワードを使用します。

```
* Declare two instances of MQMD
01 MY-MQMD.
   COPY CMQMDV.
01 MY-OTHER-MQMD.
   COPY CMQMDV.

*
* Set MSGTYPE field in MY-OTHER-MQMD
MOVE MQMT-REQUEST TO MQMD-MSGTYPE IN MY-OTHER-MQMD.
```

構造体は適切な境界で位置合わせします。COPY 文を使用してレベル 01 項目以外の項目の後に続く構造体を組み込む場合は、その構造体がレベル 01 項目の開始点から見て適当なオフセットから始まっていることを確認してください。ほとんどの MQI 構造体では 4 バイトの位置合わせが必要です。例外として、IBM i では、MQCNO、MQOD、MQPMO に対して 16 バイトの位置合わせが必要となります。

このセクションでは、構造体内のフィールドの名前は接頭部なしで表記してあります。COBOL では、フィールド名の前に、構造体の名前とハイフンを使用した接頭部が付きます。ただし、構造体名の末尾に数字が付いていて、その構造体が元の構造体の 2 番目以降のバージョンであることが示されている場合、そのような数字は接頭部では省略されます。COBOL では、フィールド名は大文字で表記します (必要な場合には小文字の表記または混合表記も使用できます)。例えば、で記述された *MsgType* フィールドは、COBOL385 ページの『MQMD - メッセージ記述子』では MQMD-MSGTYPE となる。

接尾部 V の付いた構造体は、すべてのフィールドの初期値を使用して宣言します。したがって、設定する必要があるのは、提供されている初期値と異なる値を必要とするフィールドだけです。

ポインター

構造体によっては、その構造体とは連続していないオプション・データをアドレッシングすることが必要な場合があります。MQOD 構造体によってアドレッシングされる MQOR レコードや MQRR レコードはその例です。

構造体の中には、このようなオプション・データをアドレッシングするためのフィールドがあり、ポインター・データ・タイプで宣言されます。ただし、COBOL では、ポインター・データ・タイプはどの環境でもサポートされません。この場合には、構造体の先頭からのデータのオフセットを含むフィールドを使用すると、オプション・データをアドレッシングできます。

アプリケーションを各環境間で移植する場合、対象となるすべての環境でポインター・データ・タイプが使用できるかどうかを確認します。ポインター・データ・タイプが使用できない環境では、アプリケーションでのオプション・データのアドレッシングは、ポインター・フィールドではなくオフセット・フィールドを使って実行しなければなりません。

ポインターがサポートされていない環境では、適切な長さのバイト・ストリングとしてポインター・フィールドを宣言します。この場合の初期値はすべてヌルのバイト・ストリングになります。オフセット・フィールドを使用している場合は、この初期値を変更しないでください。

名前付き定数

この情報では、定数の名前にはアンダースコア文字 (_) が名前の一部として含まれていることが示されています。COBOL では、下線の代わりにハイフン (-) 文字を使用します。

文字ストリング値を持つ定数は、ストリング区切り文字 (') として単一引用符を使用します。一部の環境では、適切なコンパイラ・オプションを指定して、コンパイラが二重引用符の代わりにストリング区切り文字として単一引用符を受け入れるようにする必要があります。

名前付き定数は、COPY ファイルではレベル 10 項目として宣言されています。定数を使用するには、レベル 01 項目を明示的に宣言し、次にその定数の宣言内でコピーするために使用する COPY 文を宣言します。

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

上記の方式では、定数が参照されない場合であっても、プログラム内のストレージを定数が占有してしまいます。同じ実行単位内の多数の別個のプログラムにこのような定数を組み込む場合は、同じ定数の複数のコピーが存在することになり、主ストレージを不必要に消費する結果になります。次のいずれかの技法を使用して、この事態を回避してください。

- GLOBAL 文節をレベル 01 宣言に追加する。

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

これは実行単位内で 1 組の定数のみにストレージを割り当てます。定数を参照できるプログラムは、レベル 01 宣言を含んでいるプログラムだけではなく、その実行単位内のすべてのプログラムです。

注：GLOBAL 文節は、すべての環境でサポートされるわけではありません。

- 各プログラムが参照する定数だけを、それぞれのプログラムに手動操作でコピーする。COPY ステートメントを使用して、すべての定数を一括でプログラムにコピーしないでください。

表記規則

このトピックの前のセクションでは、呼び出しの発行方法とパラメータの宣言方法を説明しています。パラメーターが、サイズの固定されていないテーブルまたは文字ストリングとなる場合もあります。このような場合は、小文字の n を使用して数値定数を表しています。そのパラメーターの宣言をエンコードするときは、必要な数値で n を置き換えます。

High Level Assembler ・ プログラミング

System/390 アセンブラ言語から MQI を使用するための情報。

- [208 ページの『マクロ』](#)
- [209 ページの『構造体』](#)
- [209 ページの『CMQVERA マクロ』](#)
- [210 ページの『表記規則』](#)

マクロ

すべての名前付き定数用のマクロが 2 つと、個々の構造体について 1 つずつマクロがあります。これらのファイルは、以下の表に要約されています。

表 463. アセンブラーのマクロ	
ファイル	内容
CMQA	メイン MQI 用の名前付き定数 (等価)
CMQCIHA	CICS 情報ヘッダー構造体
CMQCNOA	接続オプション構造体
CMQDLHA	送達不能ヘッダー構造体
CMQDXPA	データ変換出口パラメーター構造体
CMQGMOA	読み取りメッセージ・オプション構造体
CMQIIHA	IMS 情報ヘッダー構造体
CMQMDA	メッセージ記述子構造体
CMQMDEA	拡張メッセージ記述子構造体
CMQODA	オブジェクト記述子構造体
CMQPMOA	書き込みメッセージ・オプション構造体
CMQRFHA	規則および書式ヘッダー構造体
CMQRFH2A	規則および書式ヘッダー構造体、バージョン 2
CMQRMHA	参照メッセージ・ヘッダー構造体
CMQTMA	トリガー・メッセージ構造体
CMQTMCA	トリガー・メッセージ構造体 (文字形式) - バージョン 2
CMQVERA	構造体のバージョン管理
CMQWIHA	作業情報ヘッダー構造体
CMQXA	データ変換出口用の名前付き定数
CMQXPA	API 交差出口パラメーター構造体
CMQXQHA	伝送キュー・ヘッダー構造体

構造体

構造体はマクロにより生成されます。各マクロには、マクロのアクションを制御するための各種のパラメーターがあります。ユーザーの OIDS ログイン・ソースの構成に関する情報については、[210 ページ](#)の『[構造体](#)』

CMQVERA マクロ

このマクロを使用すると、構造体マクロの DCLVER パラメーターにデフォルト値を設定することができます。

CMQVERA によって指定される値が構造体マクロによって使用されるのは、その構造体マクロの起動時に DCLVER パラメーターを省略する場合だけです。このデフォルト値は、CMQVERA マクロと DCLVER パラメーターをコーディングすることによって設定します。

DCLVER=CURRENT

デフォルトのバージョンは現在 (最新) のバージョンに設定されます。

DCLVER=SPECIFIED

デフォルトのバージョンは VERSION パラメーターによって指定されるバージョンに設定されます。

DCLVER パラメーターを指定し、その値は大文字にする必要があります。CMQVERA によって指定される値は、次に CMQVERA を呼び出すまでの、またはアセンブリーの終了までのデフォルト値として残ります。CMQVERA を省略する場合のデフォルトは、DCLVER=CURRENT です。

表記規則

その他のトピックでは、呼び出しの呼び出し方法およびパラメーターの宣言方法を示します。パラメーターが、サイズの固定されていない配列または文字ストリングとなる場合があります。そのため、小文字の n を使用して数値定数を表しています。そのパラメーターの宣言をエンコードするときは、必要な数値で n を置き換えます。

構造体

構造体はマクロにより生成されます。各マクロには、マクロのアクションを制御するための各種のパラメーターがあります。

注：IBM MQ 構造体の新しいバージョンが導入されることがあります。新バージョンに新しくフィールドが加えられることによって、以前は 256 バイト未満であった構造体がそれより大きくなる可能性もあります。このため、256 バイトより大きい可能性のある構造体で正しく作業するために、IBM MQ 構造体をコピーするか、または IBM MQ 構造体をヌルに設定することを意図したアセンブラー命令を作成してください。また、DCLVER マクロ・パラメーターか CMQVERA マクロを VERSION パラメーターと共に使用することによって、構造体の特定のバージョンを宣言することもできます。

- [210 ページの『構造体の名前を指定する』](#)
- [210 ページの『構造体の形式を指定する』](#)
- [211 ページの『構造体のバージョンを制御する』](#)
- [211 ページの『別の構造体内に組み込む構造体を宣言する』](#)
- [211 ページの『フィールドへの初期値の指定』](#)
- [211 ページの『リスト出力の有無の制御』](#)

構造体の名前を指定する

同一構造体の複数のインスタンスを宣言するために、マクロは、構造体内の各フィールドの名前にユーザーが指定可能なストリングと下線から成る接頭部を付けます。

使用されるストリングは、マクロの呼び出し時に指定されたラベルです。ラベルが指定されていない場合は、構造体の名前を使用して接頭部が作成されます。

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

この構造体宣言ではデフォルトの接頭部を使用しています。

構造体の形式を指定する

構造体宣言は、DSECT パラメーターが制御する次に示す 2 つの形式のどちらかのマクロを使用して生成することができます。

DSECT=YES

アセンブラーの DSECT 命令を使用して、新しいデータ・セクションが開始されます。DSECT 文の直後に構造体定義が続きます。マクロ呼び出しのラベルがデータ・セクションの名前として使用されます。ラベルが指定されていない場合は、構造体の名前が使用されます。

DSECT=NO

アセンブラーの DC 命令を使用して、ルーチン内の現在位置に構造体が定義されます。フィールドは、関連のパラメーターをマクロ呼び出しにコーディングすることにより指定できる値により初期設定されます。マクロの呼び出しで値が指定されないフィールドは、デフォルト値で初期化されます。

指定する値は大文字でなければなりません。DSECT パラメーターを指定しないなら、DSECT=NO が想定されます。

構造体のバージョンを制御する

デフォルトでは、通常、各構造体の最新のバージョンがマクロによって宣言されます。

VERSION マクロ・パラメーターを使用すれば、構造体の *Version* フィールドの値を指定することができますが、このパラメーターは *Version* フィールドの初期値を定義するだけで、実際に宣言されている構造体のバージョンまで制御することはできません。宣言される構造体のバージョンを調整するには、DCLVER パラメーターを使用します。

DCLVER=CURRENT

宣言されるバージョンは、現在 (最新) のバージョンです。

DCLVER=SPECIFIED

宣言されるのは VERSION パラメーターにより指定されるバージョンです。VERSION パラメーターを省略する場合のデフォルトはバージョン 1 です。

VERSION パラメーターを指定する場合は、自己定義の数値定数、または必要なバージョンの名前定数を値として指定する必要があります (例えば、MQCNO_VERSION_3)。指定されている値が他にも存在する場合は、たとえ VERSION の値が有効な値であったとしても、その構造体は DCLVER=CURRENT が指定されているものとして宣言されます。

指定する値は大文字でなければなりません。DCLVER パラメーターを省略する場合、使用される値は MQDCLVER グローバル・マクロ変数から取られます。CMQVERA マクロを使用してこの変数を設定することができます。

別の構造体内に組み込む構造体を宣言する

ある構造体を別の構造体のコンポーネントとして宣言するには、NESTED パラメーターを使用します。

NESTED=YES

構造体宣言が別の構造体内部にネストされます。

NESTED=NO

構造体宣言は別の構造体内部にネストされません。

指定する値は大文字でなければなりません。NESTED パラメーターを省略する場合は NESTED=NO が想定されます。

フィールドへの初期値の指定

構造体内のフィールドの名前 (接頭部なし) を、必要な値と共にマクロ呼び出しのパラメーターとしてコーディングすることで、そのフィールドを初期設定するために使用する値を指定することができます。

例えば、*MsgType* フィールドを MQMT_REQUEST により初期設定し、*ReplyToQ* フィールドをストリング「MY_REPLY_TO_QUEUE」により初期設定して、メッセージ記述子構造体を宣言するには、以下を使用します。

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

名前付き定数 (等価) をマクロ呼び出しの値として指定する場合は、CMQA マクロを使用して、その名前付き定数を定義します。文字ストリングの値は単一引用符で囲まないでください。

リスト出力の有無の制御

構造体宣言をアセンブラー・リストに表示するかどうかを、LIST パラメーターを使って制御することができます。

LIST=YES

構造体宣言がアセンブラー・リストに表示されます。

LIST=NO

構造体宣言はアセンブラー・リストに表示されません。

指定する値は大文字でなければなりません。LIST パラメーターを省略する場合は LIST=NO が想定されます。

MQAIR - 認証情報レコード

MQAIR 構造体を使用すると、IBM MQ MQI client として実行されているアプリケーションで、クライアント接続に使用する認証子に関する情報を指定できます。この構造体は、MQCONNX 呼び出しの入力パラメーターです。

可用性

MQAIR 構造は、以下のクライアントで利用可能です：

-  AIX
-  Linux
-  Windows

文字セットとエンコード

MQAIR 内のデータは、ローカルキューマネージャの文字セットおよびエンコーディングでなければならない。これらはキューマネージャ **CodedCharSetId** 属性および MQENC_NATIVE によって指定される。

フィールド

注：次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 464. MQAIR のフィールド		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
<u>StrucId</u> (構造識別子)	MQAIR_STRUC_ID	'AIR~'
<u>バージョン</u> (構造バージョン番号)	MQAIR_VERSION_1	1
<u>AuthInfoType</u> (認証情報の種類)	MQAIT_CRL_LDAP	1
<u>AuthInfoConnName</u> (LDAP CRL サーバーの接続名)	なし	ヌル・ストリングまたはブランク
<u>LDAPUserNamePtr</u> (LDAP のユーザー名)	なし	ヌル・ポインターまたはヌル・バイト
<u>LDAPUserNameOffset</u> (LDAP ユーザー名の MQSCO 先頭からのオフセット)	なし	0
<u>LDAPUserNameLength</u> (LDAP ユーザー名の長さ)	なし	0
<u>LDAPPassword</u> (LDAP サーバーへのアクセス用パスワード)	なし	ヌル・ストリングまたはブランク
注：Version が MQAIR_VERSION_2 より小さい場合、残りのフィールドは無視されます。		
<u>OCSPResponderURL</u> (URL OCSP レスポンダが連絡可能な場所)	なし	ヌル・ストリングまたはブランク

表 464. MQAIR のフィールド (続き)

フィールド名と説明	定数の名前	定数の初期値 (存在する場合)
注: - 記号-は、単一のブランク文字を表します。 - C プログラミング言語では、マクロ変数 MQAIR_DEFAULT には、表にリストされている値が含まれています。このマクロ変数を以下の方法で使用して、構造体のフィールドに初期値を設定します。 <pre>MQAIR MyAIR = {MQAIR_DEFAULT};</pre>		

言語ごとの宣言

MQAIR の C 宣言

```
typedef struct tagMQAIR MQAIR;
struct tagMQAIR {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;           /* Structure version number */
    MQLONG     AuthInfoType;      /* Type of authentication
                                   information */
    MQCHAR264  AuthInfoConnName; /* Connection name of CRL LDAP
                                   server */
    PMQCHAR    LDAPUserNamePtr;   /* Address of LDAP user name */
    MQLONG     LDAPUserNameOffset; /* Offset of LDAP user name from start
                                   of MQAIR structure */
    MQLONG     LDAPUserNameLength; /* Length of LDAP user name */
    MQCHAR32   LDAPPASSWORD;      /* Password to access LDAP server */
    MQCHAR256  OCSPResponderURL; /* URL of OCSP responder */
};
```

MQAIR の COBOL 宣言

```
**  MQAIR structure
10 MQAIR.
**  Structure identifier
15 MQAIR-STRUCID      PIC X(4).
**  Structure version number
15 MQAIR-VERSION     PIC S9(9) BINARY.
**  Type of authentication information
15 MQAIR-AUTHINFOTYPE PIC S9(9) BINARY.
**  Connection name of CRL LDAP server
15 MQAIR-AUTHINFOCONNNAME PIC X(264).
**  Address of LDAP user name
15 MQAIR-LDAPUSERNAMEPTR  POINTER.
**  Offset of LDAP user name from start of MQAIR structure
15 MQAIR-LDAPUSERNAMEOFFSET PIC S9(9) BINARY.
**  Length of LDAP user name
15 MQAIR-LDAPUSERNAMELENGTH PIC S9(9) BINARY.
**  Password to access LDAP server
15 MQAIR-LDAPPASSWORD    PIC X(32).
**  URL of OCSP responder
15 MQAIR-OCSPRESPONDERURL PIC X(256).
```

MQAIR の Visual Basic 宣言

```
Type MQAIR
    StrucId      As String*4   'Structure identifier'
    Version      As Long       'Structure version number'
    AuthInfoType As Long       'Type of authentication information'
    AuthInfoConnName As String*264 'Connection name of CRL LDAP server'
    LDAPUserNamePtr As MQPTR    'Address of LDAP user name'
    LDAPUserNameOffset As Long  'Offset of LDAP user name from start
                                'of MQAIR structure'
    LDAPUserNameLength As Long  'Length of LDAP user name'
    LDAPPASSWORD As String*32  'Password to access LDAP server'
End Type
```

MQAIR のフィールド詳細

MQAIR 構造体を使用すると、IBM MQ MQI client として実行されているアプリケーションで、クライアント接続に使用する認証子に関する情報を指定できます。この構造体は、MQCONN 呼び出しの入力パラメーターです。

フィールド

注：次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 465. MQAIR のフィールド		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
StrucId (構造識別子)	MQAIR_STRUC_ID	'AIR~'
バージョン (構造バージョン番号)	MQAIR_VERSION_1	1
AuthInfoType (認証情報の種類)	MQAIT_CRL_LDAP	1
AuthInfoConnName (LDAP CRL サーバーの接続名)	なし	ヌル・ストリングまたはブランク
LDAPUserNamePtr (LDAP のユーザー名)	なし	ヌル・ポインターまたはヌル・バイト
LDAPUserNameOffset (LDAP ユーザー名の MQSCO 先頭からのオフセット)	なし	0
LDAPUserNameLength (LDAP ユーザー名の長さ)	なし	0
LDAPPassword (LDAP サーバーへのアクセス用パスワード)	なし	ヌル・ストリングまたはブランク
注：Version が MQAIR_VERSION_2 より小さい場合、残りのフィールドは無視されます。		
OCSPResponderURL (URL OCSP レスポンダが連絡可能な場所)	なし	ヌル・ストリングまたはブランク
注： 1. 記号~は、単一のブランク文字を表します。 2. C プログラミング言語では、マクロ変数 MQAIR_DEFAULT には、表にリストされている値が含まれています。このマクロ変数を以下の方法で使用して、構造体のフィールドに初期値を設定します。 <pre>MQAIR MyAIR = {MQAIR_DEFAULT};</pre>		

StrucId (MQCHAR4)

これは認証情報レコード構造の構造識別子です。常に入力フィールドです。その値は MQAIR_STRUC_ID です。

値は次のものでなければなりません。

MQAIR_STRUC_ID

認証情報レコードの ID。

C プログラミング言語では、定数 MQAIR_STRUC_ID_ARRAY も定義されている。これは MQAIR_STRUC_ID と同じ値を持ちますが、文字列ではなく文字の配列です。

Version (MQLONG)

これは認証情報レコード構造のバージョン番号です。常に入力フィールドです。

値は次のいずれかでなければなりません。

MQAIR_VERSION_1

Version-1 認証情報レコード。

これはこのフィールドの初期値です。

MQAIR_VERSION_2

Version-2 認証情報レコード。

以下の定数は、現行バージョンのバージョン番号を指定しています。

MQAIR_CURRENT_VERSION

認証情報レコードの現行バージョン。

AuthInfoType (MQLONG)

これは、レコードに含まれる認証情報のタイプです。

この値は、以下の 2 つのパラメーターのいずれかです。

MQAIT_CRL_LDAP

LDAP サーバーを使用する証明書失効検査

MQAIT_OCSP

OCSP を使用する証明書失効検査

値が無効の場合、呼び出しは失敗し、理由コード MQRC_AUTH_INFO_TYPE_ERROR が戻ります。

これは入力フィールドです。フィールドの初期値は、MQAIT_CRL_LDAP です。

AuthInfoConnName (MQCHAR264)

これは、LDAP サーバーが稼働しているホストのホスト名またはネットワーク・アドレスのどちらかです。この後にオプションのポート番号を括弧で囲んで指定できます。デフォルトのポート番号は 389 です。

値がフィールドの長さより短い場合、値をヌル文字で終了するか、フィールドの長さまでブランクを埋め込みます。値が無効の場合、呼び出しは失敗し、理由コード MQRC_AUTH_INFO_CONN_NAME_ERROR が戻ります。

これは入力フィールドです。このフィールドの長さは MQ_AUTH_INFO_CONN_NAME_LENGTH によって指定されます。初期値は、C 言語ではヌル・ストリングですが、その他のプログラミング言語ではブランク文字です。

LDAPUserNamePtr (PMQCHAR)

これは LDAP ユーザー名です。

これは、LDAP CRL サーバーへのアクセスを試行するユーザーの Distinguished Name から成ります。値が *LDAPUserNameLength* で指定されている長さより短い場合は、値をヌル文字で終了するか、*LDAPUserNameLength* の長さまでブランクを埋め込む必要があります。このフィールドは *LDAPUserNameLength* がゼロの場合、無視されます。

LDAP ユーザー名は、次の 2 つの方法で提供できます。

- *LDAPUserNamePtr* ポインター・フィールドを使用する

この場合、アプリケーションは MQAIR 構造体とは別個にストリングを宣言でき、そのストリングのアドレスに *LDAPUserNamePtr* を設定できます。

他の環境へ移植できる形式のポインター・データ・タイプをサポートするプログラミング言語 (C プログラミング言語など) の場合は、*LDAPUserNamePtr* の使用を検討してください。

- *LDAPUserNameOffset* オフセット・フィールドを使用する

この場合、アプリケーションは MQSCO 構造体と、その後に続く MQAIR レコードの配列、およびその後に続く LDAP ユーザー名のストリングを含む、複合の構造体を宣言する必要があります。さらに

LDAPUserNameOffset を、MQAIR 構造体の先頭からの該当する名前ストリングのオフセットに設定する必要があります。この値が正しいこと、および値が *MQLONG* 内に収まることを確認してください (最も制限の大きいプログラミング言語は COBOL で、有効範囲は -999 999 999 から +999 999 999 です)。

ポインターのデータ・タイプをサポートしていないプログラミング言語や、他の環境に移植できない形式のポインター・データ・タイプを実装しているプログラミング言語 (COBOL プログラミング言語など) の場合には、*LDAPUserNameOffset* の使用を検討してください。

どちらの方法を選んでも、*LDAPUserNamePtr* または *LDAPUserNameOffset* のいずれか一方のみを使用します。両方ともゼロでない場合、呼び出しは失敗し、理由コード *MQRC_LDAP_USER_NAME_ERROR* が戻ります。

これは入力フィールドです。このフィールドの初期値は、ポインターをサポートするプログラミング言語のヌル・ポインターです。それ以外の場合は、すべてヌルのバイトのストリングです。

注: プログラミング言語がポインターのデータ・タイプをサポートしていないプラットフォームでは、このフィールドは適切な長さのバイト・ストリングとして宣言されます。

LDAPUserNameOffset (MQLONG)

これは、MQAIR 構造体の先頭から LDAP ユーザー名のオフセットをバイト数で表したものです。

オフセットの値は、正負どちらの値にもなります。*LDAPUserNameLength* がゼロの場合、このフィールドは無視されます。

LDAP ユーザー名の指定には、*LDAPUserNamePtr* または *LDAPUserNameOffset* のどちらか一方を使用します。両方とも使用することはできません。詳細については *LDAPUserNamePtr* フィールドの説明を参照してください。

これは入力フィールドです。このフィールドの初期値は 0 です。

LDAPUserNameLength (MQLONG)

これは、*LDAPUserNamePtr* フィールドまたは *LDAPUserNameOffset* フィールドで指定された LDAP ユーザー名のバイト単位の長さです。

値はゼロから *MQ_DISTINGUISHED_NAME_LENGTH* までの範囲内でなければなりません。値が無効の場合、呼び出しは失敗し、理由コード *MQRC_LDAP_USER_NAME_LENGTH_ERR* が戻ります。

関係する LDAP サーバーがユーザー名を必要としない場合は、このフィールドをゼロに設定します。

これは入力フィールドです。このフィールドの初期値は 0 です。

LDAPPassword (MQCHAR32)

これは、LDAP CRL サーバーへのアクセスに必要なパスワードです。値がフィールドの長さより短い場合、値をヌル文字で終了するか、フィールドの長さまでブランクを埋め込みます。

LDAP サーバーがパスワードを必要としない場合や、LDAP ユーザー名を省略する場合は、*LDAPPassword* はヌルかブランクでなければなりません。LDAP ユーザー名を省略するときに、*LDAPPassword* をヌルかブランクにしないと、呼び出しは失敗し、理由コード *MQRC_LDAP_PASSWORD_ERROR* が戻ります。

これは入力フィールドです。このフィールドの長さは *MQ_LDAP_PASSWORD_LENGTH* によって指定されます。初期値は、C 言語ではヌル・ストリングですが、その他のプログラミング言語ではブランク文字です。

OCSPResponderURL (MQCHAR256)

OCSP 応答側に関する接続詳細を表す MQAIR 構造体の場合、このフィールドには、その応答側に接続できる URL が入ります。

このフィールドの値は HTTP URL です。このフィールドは、AuthorityInfoAccess (AIA) 証明書拡張の URL よりも優先されます。

以下の両方が該当する場合以外は、値は無視されます。

- MQAIR 構造体がバージョン 2 以降である（「バージョン」フィールドが MQAIR_VERSION_2 以上に設定されている）。
- 「AuthInfoType」フィールドが MQAIT_OCSP に設定されている。

このフィールドに正しいフォーマットの HTTP URL が含まれていない場合（かつ、無視されない場合）、MQCONNХ 呼び出しは失敗し、理由コード MQRC_OCSP_URL_ERROR が戻されます。

このフィールドは、大文字と小文字が区別されます。先頭は、小文字のストリング http:// にする必要があります。OCSP サーバーの実装環境によっては、URL の残りの部分は、大文字小文字が区別されることもあります。

このフィールドはデータ変換の対象にはなりません。

MQBMHO - バッファからメッセージ・ハンドルへの変換オプション

アプリケーションでは、MQBMHO 構造体を使用することによって、バッファからメッセージ・ハンドルを生成する方法を制御するためのオプションを指定することができます。この構造体は、MQBUFМH 呼び出しの入力パラメーターです。

文字セットとエンコード

MQBMHO 内のデータは、アプリケーションの文字セットおよびアプリケーションのエンコーディング（MQENC_NATIVE）でなければならない。

フィールド

注：次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 466. MQBMHO のフィールド		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
StrucId (構造識別子)	MQBMHO_STRUC_ID	'BMHO'
バージョン (構造バージョン番号)	MQBMHO_VERSION_1	1
オプション (MQBMHO の動作を制御するオプション)	MQBMHO_NONE	0
注： 1. C プログラミング言語では、マクロ変数 MQBMHO_DEFAULT には、表にリストされている値が含まれています。このマクロ変数を以下の方法で使用して、構造体のフィールドに初期値を設定します。 <pre>MQBMHO MyBMHO = {MQBMHO_DEFAULT};</pre>		

言語ごとの宣言

MQBMHO の C 宣言

```
typedef struct tagMQBMHO MQBMHO;
struct tagMQBMHO {
    MQCHAR4 StrucId;           /* Structure identifier */
    MQLONG  Version;          /* Structure version number */
    MQLONG  Options;          /* Options that control the action of
                               MQBUFМH */
};
```

MQBMHO の COBOL 宣言

```
**      MQBMHO structure
10 MQBMHO.
**      Structure identifier
15 MQBMHO-STRUCID          PIC X(4).
**      Structure version number
15 MQBMHO-VERSION          PIC S9(9) BINARY.
**      Options that control the action of MQBUFMH
15 MQBMHO-OPTIONS          PIC S9(9) BINARY.
```

MQBMHO の PL/I 宣言

```
Dcl
  1 MQBMHO based,
    3 StrucId          char(4),          /* Structure identifier */
    3 Version          fixed bin(31), /* Structure version number */
    3 Options          fixed bin(31), /* Options that control the action
                                     of MQBUFMH */
```

MQBMHO の高水準アセンブラ宣言

```
MQBMHO          DSECT
MQBMHO_STRUCID   DS   CL4   Structure identifier
MQBMHO_VERSION   DS   F     Structure version number
MQBMHO_OPTIONS   DS   F     Options that control the
*                  action of MQBUFMH
MQBMHO_LENGTH    EQU   *-MQBMHO
MQBMHO_AREA      DS     CL(MQBMHO_LENGTH)
```

MQBMHO のフィールド詳細

MQBMHO バッファからメッセージハンドルへのオプション構造体により、アプリケーションはバッファからメッセージハンドルが生成される方法を制御するオプションを指定できます。この構造体は、MQBUFMH 呼び出しの入力パラメーターです。

フィールド

注: 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 467. MQBMHO のフィールド		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
StrucId (構造識別子)	MQBMHO_STRUC_ID	'BMHO'
バージョン (構造バージョン番号)	MQBMHO_VERSION_1	1
オプション (MQBMHO の動作を制御するオプション)	MQBMHO_NONE	0
注: 1. C プログラミング言語では、マクロ変数 MQBMHO_DEFAULT には、表にリストされている値が含まれています。このマクロ変数を以下の方法で使用して、構造体のフィールドに初期値を設定します。 <pre>MQBMHO MyBMHO = {MQBMHO_DEFAULT};</pre>		

StrucId (MQCHAR4)

これは、バッファからメッセージへのハンドル構造体の構造体識別子です。常に入力フィールドです。その値は MQBMHO_STRUC_ID です。

値は次のものでなければなりません。

MQBMHO_STRUC_ID

バッファからメッセージ・ハンドルへの変換構造の ID。

C プログラミング言語では、定数 MQBMHO_STRUC_ID_ARRAY も定義されている。これは MQBMHO_STRUC_ID と同じ値を持ちますが、文字列ではなく文字の配列です。

Version (MQLONG)

これはバッファからメッセージへのハンドル構造体のバージョン番号です。常に入力フィールドです。値は次のものでなければなりません。

MQBMHO_VERSION_1

バッファからメッセージハンドル構造体へのバージョン番号。

以下の定数は、現行バージョンのバージョン番号を指定しています。

MQBMHO_CURRENT_VERSION

バッファからメッセージ・ハンドルへの変換構造の現行バージョン。

Options (MQLONG)

バッファからメッセージ・ハンドルへの変換の構造体 - Options フィールド。

値は次のいずれかです。

MQBMHO_DELETE_PROPERTIES

メッセージ・ハンドルに追加されるプロパティが、バッファから削除される。呼び出しが失敗すると、プロパティは削除されません。

デフォルトのオプション: 記載されたオプションが必要ない場合は、以下のオプションを使用してください:

MQBMHO_NONE

指定されるオプションはありません。

これは常に入力フィールドです。フィールドの初期値は、MQBMHO_DELETE_PROPERTIES です。

MQBNO - バランシング・オプション

以下の表に、この構造体の各フィールドを要約します。

フィールド

注: 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 468. MQBNO のフィールド		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
StrucId (構造識別子)	MQBNO_STRUC_ID	'BNO-'
バージョン (構造バージョン番号)	MQBNO_VERSION_1	1
ApplicationType (構造内で設定されたバランス調整オプションの種類)	MQBNO_VALTYPE_SIMPLE	0
タイムアウト (再調整がアプリケーションの動作を中断する可能性のあるタイムアウト時間)	MQBNO_TIMEOUT_AS_DEFAULT	0
BalanceOptions (発行アプリケーションによって設定されたバランス調整オプション)	MQBNO_OPTIONS_NONE	0

表 468. MQBNO のフィールド (続き)

フィールド名と説明	定数の名前	定数の初期値（存在する場合）
注： 1. 記号-は、単一のブランク文字を表します。 2. C プログラミング言語では、マクロ変数 MQBNO_DEFAULT に、表にリストされている値が含まれています。このマクロ変数を以下の方法で使用して、構造体のフィールドに初期値を設定します。 <pre>MQBNO MyBNO = {MQBNO_DEFAULT};</pre>		

言語ごとの宣言

MQBNO の C 宣言

```
typedef struct tagMQBNO MQBNO;
struct tagMQBNO {
    MQCHAR4    StrucId;          /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQLONG     Type;             /* Type of balancing options set in the
                                structure */
    MQLONG     Timeout;          /* Timeout after which re-balancing might
                                interrupt application activity */
    MQLONG     BalanceOptions;   /* Balancing options set by the issuing
                                application */
};
```

MQBNO の COBOL 宣言

```
**      MQBNO structure
10      MQBNO.
**      Structure identifier
15      MQBNO-STRUCID          PIC X(4).
**      Structure version number
15      MQBNO-VERSION          PIC S9(9) BINARY.
**      Type of balancing options set in the structure
15      MQBNO-TYPE             PIC S9(9) BINARY.
**      Timeout after which re-balancing might interrupt application activity
15      MQBNO-TIMEOUT          PIC S9(9) BINARY.
**      Balancing options set by the issuing application
15      MQBNO-BALANCEOPTIONS    PIC S9(9) BINARY.
```

MQBNO の PL/I 宣言

```
dcl
1 MQBNO based,
3 StrucId          char(4),      /* Structure identifier */
3 Version          fixed bin(31), /* Structure version number */
3 Type             fixed bin(31), /* Type of balancing options set in the
                                structure*/
3 Timeout          fixed bin(31), /* Timeout after which re-balancing might
                                interrupt application activity */
3 BalanceOptions    fixed bin(31), /* Balancing options set by the issuing
                                application*/
```

関連資料

265 ページの『MQCNO - 接続オプション』

MQCNO 構造により、アプリケーションはキューマネージャへの接続に関連するオプションを指定できます。この構造体は、MQCONN 呼び出しの入出力パラメーターです。

MQBNO のフィールド詳細

MQBNO バランス調整オプション構造体のフィールドの詳細。

フィールド

注: 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 469. MQBNO のフィールド		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
StrucId (構造識別子)	MQBNO_STRUC_ID	'BNO↵'
バージョン (構造バージョン番号)	MQBNO_VERSION_1	1
ApplicationType (構造内で設定されたバランス調整オプションの種類)	MQBNO_VALTYPE_SIMPLE	0
タイムアウト (再調整がアプリケーションの動作を中断する可能性のあるタイムアウト時間)	MQBNO_TIMEOUT_AS_DEFAULT	0
BalanceOptions (発行アプリケーションによって設定されたバランス調整オプション)	MQBNO_OPTIONS_NONE	0
注: 1. 記号↵は、単一のブランク文字を表します。 2. C プログラミング言語では、マクロ変数 MQBNO_DEFAULT に、表にリストされている値が含まれています。このマクロ変数を以下の方法で使用して、構造体のフィールドに初期値を設定します。 <pre>MQBNO MyBNO = {MQBNO_DEFAULT};</pre>		

StrucId (MQCHAR4)

これは、バランス調整オプション構造体の構造体識別子です。常に入力フィールドです。その初期値は BNO です。

値は次のものでなければなりません。

BNO

バランス調整オプション構造体の識別子。

C プログラミング言語では、定数 MQBNO_STRUC_ID_ARRAY も定義されている。この定数は BNO と同じ値を持ちますが、文字列ではなく文字の配列です。

StrucId に有効な値を指定する必要があります。そうでなければ、MQRC_BNO_ERROR が返されます。

Version (MQLONG)

これは、バランス調整オプション構造体のバージョン番号です。常に入力フィールドです。

値は次のものでなければなりません。

MQBNO_VERSION_1

バランス調整オプション構造体のバージョン番号。

Version に有効な値を指定する必要があります。そうでなければ、MQRC_BNO_ERROR が返されます。

ApplicationType (MQLONG)

構造内のバランシング・オプション・セットのタイプ。

指定できる値は以下のとおりです。

MQBNO_BALTYPE_SIMPLE

単純なバランシング。一様クラスターでのアプリケーションの再平衡化の影響で説明されているものに加えて、特定のルールは適用されません。

MQBNO_BALTYPE_REQREP

要求/応答のバランシング。各 MQPUT 呼び出しの後で、対応する MQGET 呼び出しが応答メッセージに対して予期されます。そのようなメッセージが受信されるか、要求メッセージ EXPIRY が超過するまで、バランシングは遅延されます。

これは常に入力フィールドです。このフィールドの初期値は MQBNO_BALTYPE_SIMPLE です。

ApplicationType フィールドまたは MQRC_BNO_ERROR のいずれかの値のみを指定する必要があります。

注：MQBNO_BALTYPE_RA_MANAGED のこのフィールドの追加値は、IBM MQ リソース・アダプターが JEE 環境用に使用するために予約されています。アプリケーションがこの値を直接提供するのはエラーですが、例えば、アプリケーション状況を照会するときにレポートを作成することができます。

Timeout (MQLONG)

再バランシングによってアプリケーション・アクティビティーが中断される可能性がある **Timeout**。

指定できる値は以下のとおりです。

MQBNO_TIMEOUT_AS_DEFAULT

設定されたデフォルトのタイムアウト値。

MQBNO_TIMEOUT_IMMEDIATE

即時タイムアウトが発生します。

MQBNO_TIMEOUT_NEVER

タイムアウトは起こりません。

このフィールドの初期値は MQBNO_TIMEOUT_AS_DEFAULT です。

Timeout フィールドまたは MQRC_BNO_ERROR が戻される場合は、定義済みの値、または 0 から 999999999 秒までの値のみを指定する必要があります。

BalanceOptions (MQLONG)

発行側アプリケーションによって設定されるバランシング・オプション。

可能な値は以下の通りです：

MQBNO_OPTIONS_NONE

オプションは設定されていません。

MQBNO_OPTIONS_IGNORE_TRANS

このオプションを設定すると、トランザクションの途中でであっても、アプリケーションをリバランスすることができます。

このフィールドの初期値は MQBNO_OPTIONS_NONE です。

BalanceOptions フィールドの論理または文字を使用して、定義された値の任意の組み合わせを指定できます。有効でない値があると、MQRC_BNO_ERROR が戻されます。

関連資料

265 ページの『MQCNO - 接続オプション』

MQCNO 構造により、アプリケーションはキューマネージャへの接続に関連するオプションを指定できます。この構造体は、MQCONN 呼び出しの入出力パラメーターです。

MQBO - 開始オプション

MQBO 構造体を使用すると、アプリケーションで、作業単位の作成に関するオプションを指定できます。この構造体は、MQBEGIN 呼び出しの入出力パラメーターです。

可用性

MQBO 構造は、以下のプラットフォームで利用可能です：

-  AIX
-  IBM i
-  Linux
-  Windows

MQBO 構造体は、IBM MQ MQI clients では使用できません。

文字セットとエンコード

MQBO 内のデータは、キューマネージャ **CodedCharSetId** 属性およびローカルキューマネージャのエンコーディング (MQENC_NATIVE で指定) によって指定される文字セットでなければならない。ただし、アプリケーションが MQ MQI クライアントとして実行されている場合、構造体はクライアントの文字セットとエンコードに従っている必要があります。

フィールド

注: 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 470. MQBO における MQBO のフィールド		
フィールド名と説明	定数の名前	定数の初期値 (存在する場合)
<u>StrucId</u> (構造識別子)	MQBO_STRUC_ID	'B0'
<u>バージョン</u> (構造バージョン番号)	MQBO_VERSION_1	1
<u>オプション</u> (MQBEGIN の動作を制御するオプション)	MQBO_NONE	0
注: 1. 記号'は、単一のブランク文字を表します。 2. C プログラミング言語では、マクロ変数 MQBO_DEFAULT には、表にリストされている値が含まれています。このマクロ変数を以下の方法で使用して、構造体のフィールドに初期値を設定します。 <pre>MQBO MyBO = {MQBO_DEFAULT};</pre>		

言語ごとの宣言

MQBO の C 宣言

```
typedef struct tagMQBO MQBO;
struct tagMQBO {
    MQCHAR4 StrucId; /* Structure identifier */
    MQLONG Version; /* Structure version number */
    MQLONG Options; /* Options that control the action of MQBEGIN */
};
```

MQBO の COBOL 宣言

```
** MQBO structure
10 MQBO.
** Structure identifier
15 MQBO-STRUCID PIC X(4).
** Structure version number
15 MQBO-VERSION PIC S9(9) BINARY.
** Options that control the action of MQBEGIN
15 MQBO-OPTIONS PIC S9(9) BINARY.
```

MQBO の PL/I 宣言

```

dcl
  1 MQBO based,
    3 StrucId char(4),           /* Structure identifier */
    3 Version fixed bin(31), /* Structure version number */
    3 Options fixed bin(31); /* Options that control the action of
                               MQBEGIN */

```

MQBO の Visual Basic 宣言

```

Type MQBO
  StrucId As String*4 'Structure identifier'
  Version As Long     'Structure version number'
  Options As Long     'Options that control the action of MQBEGIN'
End Type

```

MQBO のフィールド詳細

MQBO Begin オプション構造は、アプリケーションが作業単位の作成に関連するオプションを指定することを可能にします。この構造体は、MQBEGIN 呼び出しの入出力パラメーターです。

フィールド

注: 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 471. MQBO のフィールド		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
StrucId (構造識別子)	MQBO_STRUC_ID	'B0'
バージョン (構造バージョン番号)	MQBO_VERSION_1	1
オプション (MQBEGIN の動作を制御するオプション)	MQBO_NONE	0
注: 1. 記号'は、単一のブランク文字を表します。 2. C プログラミング言語では、マクロ変数 MQBO_DEFAULT には、表にリストされている値が含まれています。このマクロ変数を以下の方法で使用して、構造体のフィールドに初期値を設定します。 MQBO MyBO = {MQBO_DEFAULT};		

StrucId (MQCHAR4)

これは開始オプション構造体の構造体識別子です。常に入力フィールドです。その値は MQBO_STRUC_ID です。

値は次のものでなければなりません。

MQBO_STRUC_ID

開始オプション構造体の識別子。

C プログラミング言語では、定数 MQBO_STRUC_ID_ARRAY も定義されている。これは MQBO_STRUC_ID と同じ値を持ちますが、文字列ではなく文字の配列です。

Version (MQLONG)

これは開始オプション構造体のバージョン番号です。常に入力フィールドです。

値は次のものでなければなりません。

MQBO_VERSION_1

begin オプション構造体のバージョン番号。

以下の定数は、現行バージョンのバージョン番号を指定しています。

MQBO_CURRENT_VERSION

現在の begin オプション構造体のバージョン。

Options (MQLONG)

このフィールドは常に入力フィールドです。 初期値は、MQBO_NONE です。

値は次のものでなければなりません。

MQBO_NONE

指定されるオプションはありません。

MQCBC - コールバック・コンテキスト

MQCBC 構造を使用して、コールバック関数に渡されるコンテキスト情報を指定します。 この構造は、メッセージ・コンシューマー・ルーチンに対する呼び出しの入出力パラメーターです。

可用性

MQCBC 構造は、以下のプラットフォームで利用可能です：

- ▶ **AIX** AIX
- ▶ **IBM i** IBM i
- ▶ **Linux** Linux
- ▶ **Windows** Windows
- ▶ **z/OS** z/OS

および、これらのシステムに接続された IBM MQ MQI clients。

バージョン

現在の MQCBC のバージョンは MQCBC_VERSION_2 です。

文字セットとエンコード

MQCBC 内のデータは、キューマネージャ **CodedCharSetId** 属性で指定された文字セットおよびローカルキューマネージャのエンコーディング（MQENC_NATIVE で指定）でなければならない。しかし、アプリケーションが MQ MQI クライアントとして実行している場合、構造はクライアントの文字セット内およびエンコード内にあります。

フィールド

注：

- 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。
- MQCBC 構造には初期値が存在しません。 この構造体は、コールバック・ルーチンにパラメーターとして渡されます。 キュー・マネージャーが構造体を初期化します。アプリケーションはこの構造体を初期化しません。

表 472. MQCBC のフィールド	
フィールド	説明
StrucID	構造体 ID

表 472. MQCBC のフィールド (続き)	
フィールド	説明
バージョン	構造体のバージョン番号
CallType	機能が呼び出された理由
Hobj	オブジェクト・ハンドル
CallbackArea	コールバック関数が使用するフィールド
ConnectionArea	コールバック関数が使用するフィールド
CompCode	完了コード
理由 (Reason)	理由コード
状態	現行コンシューマーの状態の指示
DataLength	メッセージ長
BufferLength	メッセージ・バッファの長さ (バイト数)
フラグ	汎用フラグ
注: Version が MQCBC_VERSION_2 より前の場合、このフィールドは無視されます。	
ReconnectDelay	再接続を試みるまでのミリ秒数

言語ごとの宣言

MQCBC の C 宣言

```
typedef struct tagMQCBC MQCBC;
struct tagMQCBC {
    MCHAR4    StrucId;           /* Structure identifier */
    MQLONG    Version;          /* Structure version number */
    MQLONG    CallType;         /* Why Function was called */
    MQHOBj    Hobj;             /* Object Handle */
    MQPTR     CallbackArea;     /* Callback data passed to the function */
    MQPTR     ConnectionArea;   /* MQCTL data area passed to the function */
    MQLONG    CompCode;         /* Completion Code */
    MQLONG    Reason;           /* Reason Code */
    MQLONG    State;            /* Consumer State */
    MQLONG    DataLength;       /* Message Data Length */
    MQLONG    BufferLength;      /* Buffer Length */
    MQLONG    Flags;            /* Flags containing information about
                                this consumer */

    /* Ver:1 */
    MQLONG    ReconnectDelay;   /* Number of milliseconds before */
    /* Ver:2 */ }               /* reconnect attempt */
```

MQCBC の COBOL 宣言

```
** MQCBC structure
10  MQCBC.
** Structure Identifier
15  MQCBC-STRUCID                     PIC X(4).
** Structure Version
15  MQCBC-VERSION                     PIC S9(9) BINARY.
** Call Type
15  MQCBC-CALLTYPE                     PIC S9(9) BINARY.
** Object Handle
15  MQCBC-HOBJ                       PIC S9(9) BINARY.
** Callback User Area
15  MQCBC-CALLBACKAREA                POINTER
** Connection Area
15  MQCBC-CONNECTIONAREA              POINTER
** Completion Code
15  MQCBC-COMPCODE                     PIC S9(9) BINARY.
** Reason Code
15  MQCBC-REASON                       PIC S9(9) BINARY.
```

```

** Consumer State
15 MQCBC-STATE PIC S9(9) BINARY.
** Data Length
15 MQCBC-DATALENGTH PIC S9(9) BINARY.
** Buffer Length
15 MQCBC-BUFFERLENGTH PIC S9(9) BINARY.
** Flags
15 MQCBC-FLAGS PIC S9(9) BINARY.
** Ver:1 **
** Number of milliseconds before reconnect attempt
15 MQCBC-RECONNECTDELAY PIC S9(9) BINARY.
** Ver:2 **

```

MQCBC の PL/I 宣言

```

dcl
1 MQCBC based,
3 StrucId char(4), /* Structure identifier */
3 Version fixed bin(31), /* Structure version */
3 CallType fixed bin(31), /* Callback type */
3 Hobj fixed bin(31), /* Object Handle */
3 CallbackArea pointer, /* User area passed to the function */
3 ConnectionArea pointer, /* Connection User Area */
3 CompCode fixed bin(31); /* Completion Code */
3 Reason fixed bin(31); /* Reason Code */
3 State fixed bin(31); /* Consumer State */
3 DataLength fixed bin(31); /* Message Data Length */
3 BufferLength fixed bin(31); /* Message Buffer length */
3 Flags fixed bin(31); /* Consumer Flags */
/* Ver:1 */
3 ReconnectDelay fixed bin(31); /* Number of milliseconds before */
/* Ver:2 */ /* reconnect attempt */

```

MQCBC の高水準アセンブラ宣言

MQCBC	DSECT	
MQCBC	DS 0F	Force fullword alignment
MQCBC_STRUCID	DS CL4	Structure identifier
MQCBC_VERSION	DS F	Structure version number
MQCBC_CALLTYPE	DS F	Why Function was called
MQCBC_HOBJ	DS F	Object Handle
MQCBC_CALLBACKAREA	DS A	Callback data passed to the function
MQCBC_CONNECTIONAREA	DS A	MQCTL Data area passed to the function
MQCBC_COMPCODE	DS F	Completion Code
MQCBC_REASON	DS F	Reason Code
MQCBC_STATE	DS F	Consumer State
MQCBC_DATALENGTH	DS F	Message Data Length
MQCBC_BUFFERLENGTH	DS F	Buffer Length
MQCBC_FLAGS	DS F	Flags containing information about this consumer
MQCBC_RECONNECTDELAY	DS F	Number of milliseconds before reconnect
MQCBC_LENGTH	EQU *-MQCBC	
	ORG MQCBC	
MQCBC_AREA	DS CL(MQCBC_LENGTH)	

MQCBC のフィールド詳細

MQCBC コールバックコンテキスト構造体は、コールバック関数に渡されるコンテキスト情報を指定するために使用されます。この構造は、メッセージ・コンシューマー・ルーチンに対する呼び出しの入出力パラメーターです。

フィールド

注：

- 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。子トピックも同じ順序に従います。
- MQCBC 構造には初期値が存在しません。この構造体は、コールバック・ルーチンにパラメーターとして渡されます。キュー・マネージャーが構造体を初期化します。アプリケーションはこの構造体を初期化しません。

表 473. MQCBC のフィールド	
フィールド	説明
<u>StrucID</u>	構造体 ID
<u>バージョン</u>	構造体のバージョン番号
<u>CallType</u>	機能が呼び出された理由
<u>Hobj</u>	オブジェクト・ハンドル
<u>CallbackArea</u>	コールバック関数が使用するフィールド
<u>ConnectionArea</u>	コールバック関数が使用するフィールド
<u>CompCode</u>	完了コード
理由 (Reason)	理由コード
<u>状態</u>	現行コンシューマーの状態の指示
<u>DataLength</u>	メッセージ長
<u>BufferLength</u>	メッセージ・バッファの長さ (バイト数)
<u>フラグ</u>	汎用フラグ
注: Version が MQCBC_VERSION_2 より前の場合、このフィールドは無視されます。	
<u>ReconnectDelay</u>	再接続を試みるまでのミリ秒数

StrucID (MQCHAR4)

これはコールバックコンテキスト構造体の構造体識別子です。常に入力フィールドです。その値は MQCBC_STRUC_ID です。

値は次のものでなければなりません。

MQCBC_STRUC_ID

コールバック・コンテキスト構造の ID。

C プログラミング言語では、定数 MQCBC_STRUC_ID_ARRAY も定義されている。これは MQCBC_STRUC_ID と同じ値を持ちますが、文字列ではなく文字の配列です。

Version (MQLONG)

これはコールバックコンテキスト構造体のバージョン番号です。常に入力フィールドです。

値は次のものでなければなりません。

MQCBC_VERSION_1

バージョン 1 コールバックコンテキスト構造体。

以下の定数は、現行バージョンのバージョン番号を指定しています。

MQCBC_CURRENT_VERSION

コールバック・コンテキストの構造の現行バージョン。

コールバック関数は、常に最新バージョンの構造体を渡されます。

CallType (MQLONG)

この機能が呼び出された理由に関する情報を格納するフィールド。以下の値が定義されます。

メッセージ送達呼び出しタイプ: これらの呼び出しタイプには、メッセージに関する情報が含まれます。これらの呼び出しタイプの場合は、**DataLength** および **BufferLength** パラメーターは有効です。

MQCBCT_MSG_REMOVED

メッセージ・コンシューマー機能が呼び出され、メッセージがオブジェクト・ハンドルから破壊除去されました。

CompCode の値が MQCC_WARNING の場合は、*Reason* フィールドの値は MQRC_TRUNCATED_MSG_ACCEPTED か、データ変換の問題を示すコードの 1 つです。

MQCBCT_MSG_NOT_REMOVED

メッセージ・コンシューマー機能が呼び出され、メッセージはまだオブジェクト・ハンドルから破壊除去されていません。*MsgToken* を使用すると、メッセージをオブジェクト・ハンドルから破壊除去できます。

メッセージが除去されていない理由は以下のとおりです。

- MQGMO オプションがブラウザ操作 MQGMO_BROWSE_* を要求した
- メッセージが使用可能なバッファより大きく、MQGMO オプションが MQGMO_ACCEPT_TRUNCATED_MSG を指定していない

CompCode の値が MQCC_WARNING の場合は、*Reason* フィールドの値は MQRC_TRUNCATED_MSG_FAILED か、データ変換の問題を示すコードの 1 つです。

コールバック制御呼び出しタイプ: これらの呼び出しタイプには、コールバックの制御に関する情報が含まれ、メッセージに関する詳細情報は含まれません。これらの呼び出しタイプを要求するには、MQCBD 構造中で [Options](#) を使用します。

これらの呼び出しタイプの場合は、**DataLength** および **BufferLength** パラメーターは無効です。

MQCBCT_REGISTER_CALL

この呼び出しタイプの目的は、コールバック関数がいくつかの初期セットアップを実行できるようにすることです。

コールバック関数は、コールバックの登録直後に呼び出されます。つまり、MQOP_REGISTER の *Operation* フィールドの値を使用する MQCB 呼び出しから戻る際に呼び出されます。

この呼び出しタイプは、メッセージ・コンシューマーとイベント・ハンドラーの両方に使用されます。

要求された場合、このタイプがコールバック関数の最初の呼び出しになります。

Reason フィールドの値は MQRC_NONE です。

MQCBCT_START_CALL

この呼び出しタイプの目的は、コールバック関数が開始時にいくつかのセットアップを実行できるようにすることです。例えば、以前の停止時に終結処理されたリソースの再インストールなどが含まれます。

このコールバック関数は、MQOP_START または MQOP_START_WAIT を使用して接続を開始する際に呼び出されます。

コールバック関数が別のコールバック関数内に登録されている場合、この呼び出しタイプはコールバックが戻るときに呼び出されます。

この呼び出しタイプは、メッセージ・コンシューマー専用です。

Reason フィールドの値は MQRC_NONE です。

MQCBCT_STOP_CALL

この呼び出しタイプの目的は、コールバック関数がしばらくの間停止している際にいくつかの終結処理を実行できるようにすることです。例えば、メッセージのコンシューム中に獲得した追加リソースの終結処理などが含まれます。

コールバック関数は、MQOP_STOP の *Operation* フィールドの値を使用して MQCTL 呼び出しを発行する際に呼び出されます。

この呼び出しタイプは、メッセージ・コンシューマー専用です。

Reason フィールドの値が設定され、停止の理由が示されます。

MQCBCT_DEREGISTER_CALL

この呼び出しタイプの目的は、コールバック関数がコンシューム・プロセスの終わりに最終終結処理を実行できるようにすることです。このコールバック関数は、以下の時点で呼び出されます。

- MQOP_DEREGISTER を指定した MQCB 呼び出しを使用してコールバック関数が登録解除される。
- キューがクローズされるために、暗黙的な登録解除が発生する時点。この場合、コールバック関数はオブジェクト・ハンドルとして MQHO_UNUSABLE_HOBJ を渡されます。
- MQDISC 呼び出しが完了する時点。暗黙的なクローズが発生し、そのために登録解除が発生します。この場合、接続は即時に切断されず、実行中のトランザクションはまだコミットされません。

これらのいずれかのアクションがコールバック関数自体の内部で取られる場合、コールバックが戻るとアクションが呼び出されます。

この呼び出しタイプは、メッセージ・コンシューマーとイベント・ハンドラーの両方に使用されます。

要求された場合、このタイプがコールバック関数の最後の呼び出しになります。

Reason フィールドの値が設定され、停止の理由が示されます。

MQCBCT_EVENT_CALL

イベント・ハンドラー機能

キュー・マネージャーまたは接続の停止時か静止時に、イベント・ハンドラー機能がメッセージなしで呼び出されています。

この呼び出しを使用して、すべてのコールバック関数に対し適切な処置を行うことができます。

メッセージ・コンシューマー機能

メッセージ・コンシューマー機能は、オブジェクト・ハンドルに固有のエラー (*CompCode*=MQCC_FAILED) が検出されている時点で、メッセージなしで呼び出されています (例えば、*Reason* コード = MQRC_GET_INHIBITED)。

Reason フィールドの値が設定され、呼び出しの理由が示されます。

MQCBCT_MC_EVENT_CALL

イベント・ハンドラー機能がマルチキャスト・イベントに対して呼び出されています。イベント・ハンドラーは、標準の IBM MQ イベントではなく、IBM MQ マルチキャスト・イベントに対して送信されます。

MQCBCT_MC_EVENT_CALL の詳細については、[マルチキャスト例外報告](#)を参照してください。

Hobj (MQHOBJ)

これは、メッセージ・コンシューマーに対する呼び出しのオブジェクト・ハンドルです。

イベント・ハンドラーの場合、この値は MQHO_NONE です。

メッセージがキューから除去されてはいない場合、アプリケーションはこのハンドルとメッセージ読み取りオプション・ブロック中のメッセージ・トークンを使用して、メッセージを読み取ることができます。

これは常に入力フィールドです。このフィールドの初期値は、MQHO_UNUSABLE_HOBJ です。

CallbackArea (MQPTR)

これは、コールバック関数ができるフィールドです。

キュー・マネージャーは、このフィールドの内容に基づいて決定せず、MQCBD 構造中の [CallbackArea](#) フィールドから、コールバック関数の定義に使用する MQCB 呼び出し上のパラメーターが変更せずに渡されます。

CallbackArea に対する変更は、*Hobj* に関するコールバック関数を呼び出すたびに保存されます。このフィールドは、他のハンドルに関するコールバック関数と共有されません。

これは、コールバック関数への入出力フィールドです。このフィールドの初期値は、ヌル・ポインターまたはヌル・バイトです。

ConnectionArea (MQPTR)

これは、コールバック関数を使用できるフィールドです。

キュー・マネージャーは、このフィールドの内容に基づいて決定を行いません。これは、コールバック関数の制御に使われる MQCTL 呼び出しのパラメーターである、MQCTLO 構造内の [ConnectionArea](#) フィールドから不変のまま渡されます。

コールバック関数がこのフィールドに対して加える変更は、コールバック関数を呼び出すたびに保存されます。この領域を使用して、すべてのコールバック関数で共有される情報を渡すことができます。この領域は、*CallbackArea* とは違って、接続ハンドルに関するすべてのコールバックで共通です。

これは入出力フィールドです。このフィールドの初期値は、ヌル・ポインターまたはヌル・バイトです。

CompCode (MQLONG)

このフィールドは完了コードです。これは、メッセージのコンシュームに関する問題があったかどうかを示します。

値は、次のいずれか 1 つです。

MQCC_OK

正常終了。

MQCC_WARNING

警告 (部分完了)

MQCC_FAILED

呼び出し失敗

これは入力フィールドです。このフィールドの初期値は、MQCC_OK です。

理由 (MQLONG)

これは、*CompCode* を修飾する理由コードです。

これは入力フィールドです。このフィールドの初期値は MQRC_NONE です。

State (MQLONG)

現行コンシューマーの状態に関する指示。ゼロ以外の理由コードがコンシューマー関数に渡される場合、このフィールドはアプリケーションにとって非常に重要となります。

理由コードごとに動作をコーディングする必要がなくなるので、このフィールドを使用するとアプリケーション・プログラミングを単純化できます。

これは入力フィールドです。フィールドの初期値は、MQCS_NONE です。

表 474.		
State	キュー・マネージャーのアクション	定数の値
<i>MQCS_NONE</i> この理由コードは、追加の理由情報のない通常の呼び出しを表します。	なし。これは通常の操作です。	0
<i>MQCS_SUSPENDED_TEMPORARY</i> これらの理由コードは、一時的な条件を表します。	コールバック・ルーチンが呼び出されて条件が報告されてから、中断されます。特定の期間の後で、システムが操作を再試行することもあります。同じ条件が発生することになります。	1

表 474. (続き)		
State	キュー・マネージャーのアクション	定数の値
<i>MQCS_SUSPENDED_USER_ACTION</i> これらの理由コードは、解決するためにコールバックがアクションを取る必要がある条件を表します。	コンシューマーが中断され、コールバック・ルーチンが呼び出されて条件が報告されます。コールバック・ルーチンが、条件を解決し (可能な場合)、接続の RESUME または閉鎖を行う必要があります。	2
<i>MQCS_SUSPENDED</i> これらの理由コードは、以後メッセージのコールバックが行えなくなる障害を表します。	キュー・マネージャーはコールバック関数を自動的に中断します。コールバック関数を再開すると、おそらく同じ理由コードを再び受け取ります。	3
<i>MQCS_STOPPED</i> これらの理由コードは、メッセージのコンシュームの終わりを表します。	例外ハンドラーおよび <i>MQCBDO_STOP_CALL</i> を指定したコールバックに配布されます。これ以上のメッセージはコンシュームできません。	4

DataLength (MQLONG)

これは、メッセージ内のアプリケーション・データの長さ (バイト数) です。この値がゼロの場合は、メッセージにアプリケーション・データがないことを意味します。

DataLength フィールドには、メッセージの長さは含まれますが、コンシューマーに渡されるメッセージ・データの長さは必ずしも含まれるとは限りません。メッセージが切り捨てられている可能性もあります。実際にコンシューマーに渡されているデータの量を判別するには、MQGMO 中で [ReturnedLength](#) フィールドを使用してください。

メッセージが切り捨てられていることが理由コードに示される場合は、**DataLength** フィールドを使用して、実際のメッセージの大きさを判別できます。これにより、メッセージ・データを収容するのに必要なバッファのサイズを判別してから、MQCB 呼び出しを発行し、適切な値を用いて [MaxMsgLength](#) を更新することができます。

MQGMO_CONVERT オプションが指定されている場合、変換されたメッセージが **DataLength** の戻り値より大きくなる場合があります。この場合、おそらくアプリケーションが MQCB 呼び出しを発行して、[MaxMsgLength](#) が **DataLength** としてキュー・マネージャーから戻される値より大きくなるよう更新する必要があります。

メッセージ切り捨ての問題が発生しないようにするには、[MaxMsgLength](#) を MQCBD_FULL_MSG_LENGTH と指定します。これにより、キュー・マネージャーはデータ変換後のメッセージ全長に対応するバッファを割り振ります。ただし、このオプションが指定された場合でも、要求を正しく処理するのに十分なストレージが使用できない可能性があることに注意してください。アプリケーションが、戻される理由コードを常に検査する必要があります。例えば、メッセージを変換するのに十分なストレージを割り振ることができない場合は、メッセージは変換されずにアプリケーションに戻されます。

これはメッセージ・コンシューマー関数に対する入力フィールドで、イベント・ハンドラー関数には関係ありません。

BufferLength (MQLONG)

このフィールドは、この機能に渡されているメッセージ・バッファの長さ (バイト数) です。

このバッファは、コンシューマーに関して定義された [MaxMsgLength](#) 値と、MQGMO 内の [ReturnedLength](#) 値の両方よりも大きくなる場合があります。

実際のメッセージ長は、[DataLength](#) フィールドで提供されます。

コールバック関数の所要時間中、アプリケーションは独自の目的でバッファ全体を使用できます。

これはメッセージ・コンシューマー関数の入力フィールドです。例外ハンドラー関数とは関係ありません。

フラグ (MQLONG)

このコンシューマーに関する情報を含むフラグ。

以下のオプションが定義されます。

MQCBCF_READA_BUFFER_EMPTY

このフラグは、MQCO_QUIESCE オプションを使用した前の MQCLOSE 呼び出しが理由コード MQRC_READ_AHEAD_MSGS で失敗した場合に戻ることがあります。

このコードは、最後の先読みメッセージが戻され、バッファが空になったことを示しています。アプリケーションが MQCO_QUIESCE オプションを使用して別の MQCLOSE 呼び出しを発行すると、正常に行われます。

このフラグが設定されたメッセージがアプリケーションに確実に与えられるとは限らないことに注意してください。これは、現在の選択基準と一致しないメッセージが依然として先読みバッファ内に入っている可能性があるためです。この場合、理由コード MQRC_HOBJ_QUIESCED でコンシューマー関数が呼び出されます。

先読みバッファが完全に空の場合は、MQCBCF_READA_BUFFER_EMPTY フラグおよび理由コード MQRC_HOBJ_QUIESCED_NO_MSGS でコンシューマーが呼び出されます。

これはメッセージ・コンシューマー関数に対する入力フィールドで、イベント・ハンドラー関数には関係ありません。

ReconnectDelay (MQLONG)

ReconnectDelay は、再接続を試みるまでキュー・マネージャーが待つ時間の長さを示します。このフィールドをイベント・ハンドラーによって変更し、再接続の遅延または停止を完全に変更することができます。

ReconnectDelay フィールドは、コールバック・コンテキストにおける Reason フィールドの値が MQRC_RECONNECTING である場合にのみ使用してください。

イベント・ハンドラーに入ったときの ReconnectDelay の値は、再接続を試みるまでキュー・マネージャーが待つミリ秒数です。次の表は、イベントハンドラからの戻り値としてキューマネージャの動作を変更するために設定可能な値を一覧表示したものです。

表 475. ReconnectDelay の値		
名前	値	説明
MQRD_NO_RECONNECT	-1	それ以上再接続を試みません。アプリケーションにはエラーが戻されます。
MQRD_NO_DELAY	0	すぐに再接続を試みます。
Milliseconds	>0	接続を再試行するまで、このミリ秒数の間待ちます。

MQCBD - コールバック記述子

MQCBD 構造体は、コールバック関数と、キューマネージャによるその使用を制御するオプションを指定するために使用されます。この構造体は、MQCB 呼び出しの入力パラメーターです。

可用性

MQCBD 構造は、以下のプラットフォームで利用可能です：

-  AIX
-  IBM i
-  Linux
-  Windows

-  z/OS

および、これらのシステムに接続された IBM MQ MQI clients。

バージョン

MQCBD の現在のバージョンは MQCBD_VERSION_1 です。

文字セットとエンコード

MQCBD 内のデータは、キュー・マネージャ **CodedCharSetId** 属性で指定された文字セットおよび MQENC_NATIVE で指定されたローカル・キュー・マネージャのエンコーディングでなければならない。ただし、アプリケーションが MQ MQI クライアントとして実行されている場合、構造体はクライアントの文字セットとエンコードに従っている必要があります。

フィールド

注：次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 476. MQCBD のフィールド		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
StrucID (構造識別子)	MQCBD_STRUC_ID	'CBD↵'
バージョン (構造バージョン番号)	MQCBD_VERSION_1	1
CallbackType (コールバック関数の種類)	MQCBT_MESSAGE_CONSUMER	1
オプション (メッセージ消費を制御するオプション)	MQCBDO_NONE	0
CallbackArea (使用するコールバック関数のフィールド)	なし	ヌル・ポインターまたはヌル・ブランク
CallbackFunction (関数が API 呼び出しとして呼び出されるかどうか)	なし	ヌル・ポインターまたはヌル・ブランク
CallbackName (関数が動的リンクされたプログラムとして呼び出されるかどうか)	なし	ヌル・ストリングまたはブランク
MaxMsgLength (読み取れる最長メッセージの長さ)	MQCBD_FULL_MSG_LENGTH	-1
注： 1. 記号↵は、単一のブランク文字を表します。 2. ヌル・ストリングまたはブランクの値は、C プログラミング言語ではヌル・ストリングを表し、他のプログラミング言語ではブランク文字を表します。 3. C プログラミング言語では、マクロ変数 MQCBD_DEFAULT には、表にリストされている値が含まれています。このマクロ変数を以下の方法で使用して、構造体のフィールドに初期値を設定します。 <pre>MQCBD MyCBD = {MQCBD_DEFAULT};</pre>		

言語ごとの宣言

MQCBD の C 宣言

```
typedef struct tagMQCBD MQCBD;
struct tagMQCBD {
```

```

MQCHAR4    StrucId;           /* Structure identifier */
MQLONG     Version;           /* Structure version number */
MQLONG     CallbackType;      /* Callback function type */
MQLONG     Options;           /* Options controlling message
                               consumption */
MQPTR      CallbackArea;      /* User data passed to the function */
MQPTR      CallbackFunction;   /* Callback function pointer */
MQCHAR128   CallbackName;     /* Callback name */
MQLONG     MaxMsgLength;      /* Maximum message length */
};

```

MQCBD の COBOL 宣言

```

** MQCBCD structure
10  MQCBD.
** Structure Identifier
15  MQCBD-STRUCID                PIC X(4).
** Structure Version
15  MQCBD-VERSION              PIC S9(9) BINARY.
** Callback Type
15  MQCBD-CALLBACKTYPE         PIC S9(9) BINARY.
** Options
15  MQCBD-OPTIONS              PIC S9(9) BINARY.
** Callback User Area
15  MQCBD-CALLBACKAREA         POINTER
** Callback Function Pointer
15  MQCBD-CALLBACKFUNCTION     FUNCTION-POINTER
** Callback Program Name
15  MQCBD-CALLBACKNAME         PIC X(128)
** Maximum Message Length
15  MQCBD-MAXMSGLENGTH         PIC S9(9) BINARY.

```

MQCBD の PL/I 宣言

```

dcl
1  MQCBD based,
3  StrucId          char(4),          /* Structure identifier*/
3  Version          fixed bin(31),    /* Structure version*/
3  CallbackType     fixed bin(31),    /* Callback function type */
3  Options          fixed bin(31),    /* Options */
3  CallbackArea     pointer,          /* User area passed to the function */
3  CallbackFunction pointer,          /* Callback Function Pointer */
3  CallbackName     char(128),        /* Callback Program Name */
3  MaxMsgLength     fixed bin(31);    /* Maximum Message Length */

```

MQCBD のフィールド詳細

MQCBD コールバック記述子構造体は、コールバック関数と、キューマネージャによるその使用を制御するオプションを指定するために使用されます。この構造体は、MQCB 呼び出しの入力パラメーターです。

フィールド

注: 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。子トピックも同じ順序に従います。

表 477. MQCBD のフィールド		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
StrucID (構造識別子)	MQCBD_STRUC_ID	'CBD'
バージョン (構造バージョン番号)	MQCBD_VERSION_1	1
CallbackType (コールバック関数の種類)	MQCBT_MESSAGE_CONSUMER	1
オプション (メッセージ消費を制御するオプション)	MQCBDO_NONE	0
CallbackArea (使用するコールバック関数のフィールド)	なし	ヌル・ポインターまたはヌル・ブランク

表 477. MQCBD のフィールド (続き)		
フィールド名と説明	定数の名前	定数の初期値 (存在する場合)
<u>CallbackFunction</u> (関数が API 呼び出しとして呼び出されるかどうか)	なし	ヌル・ポインターまたはヌル・ブランク
<u>CallbackName</u> (関数が動的リンクされたプログラムとして呼び出されるかどうか)	なし	ヌル・ストリングまたはブランク
<u>MaxMsgLength</u> (読み取れる最長メッセージの長さ)	MQCBD_FULL_MSG_LENGTH	-1
注: 1. 記号-は、単一のブランク文字を表します。 2. ヌル・ストリングまたはブランクの値は、C プログラミング言語ではヌル・ストリングを表し、他のプログラミング言語ではブランク文字を表します。 3. C プログラミング言語では、マクロ変数 MQCBD_DEFAULT には、表にリストされている値が含まれています。このマクロ変数を以下の方法で使用して、構造体のフィールドに初期値を設定します。 <pre>MQCBD MyCBD = {MQCBD_DEFAULT};</pre>		

StrucID (MQCHAR4)

これはコールバック記述子構造体の構造体識別子です。常に入力フィールドです。その値は MQCBD_STRUC_ID です。

値は次のものでなければなりません。

MQCBD_STRUC_ID

コールバック記述子構造の ID。

C プログラミング言語では、定数 MQCBD_STRUC_ID_ARRAY も定義されている。これは MQCBD_STRUC_ID と同じ値を持ちますが、文字列ではなく文字の配列です。

Version (MQLONG)

これはコールバック記述子構造体のバージョン番号です。常に入力フィールドです。

値は次のものでなければなりません。

MQCBD_VERSION_1

バージョン 1 コールバック記述子構造体。

以下の定数は、現行バージョンのバージョン番号を指定しています。

MQCBD_CURRENT_VERSION

コールバック記述子の構造の現行バージョン。

CallbackType (MQLONG)

コールバック記述子構造体 - `CallbackType` フィールド

これは、コールバック関数のタイプです。値は次のいずれかです。

MQCBT_MESSAGE_CONSUMER

このコールバックをメッセージ・コンシューマー関数として定義します。

メッセージ・コンシューマー・コールバック関数は、指定された選択基準と一致するメッセージがオブジェクト・ハンドル上で使用可能であり、接続が開始されている場合に呼び出されます。

MQCBT_EVENT_HANDLER

このコールバックを非同期イベント・ルーチンとして定義します。これはハンドルのメッセージをコンシュームするためには使用されません。

Hobj は、イベント・ハンドラーを定義する MQCB 呼び出しでは必要ないので、指定すると無視されます。

イベント・ハンドラーは、メッセージ・コンシューマー環境全体に影響が及ぶ場合に呼び出されます。コンシューマー関数は、例えばキュー・マネージャーまたは接続が停止中または静止中であるといったイベントが発生する場合に、メッセージなしで呼び出されます。MQRC_GET_INHIBITED などの、単一のメッセージ・コンシューマーに固有の条件に関するイベント・ハンドラーは呼び出されません。

接続が開始されているか停止しているかにかかわらず、イベントはアプリケーションに配布されますが、以下の環境は例外です。

- CICS 環境 z/OS について
- 非スレッド・アプリケーション

呼び出し元がこれらのいずれかの値を渡さない場合は、呼び出しは *Reason* コード MQRC_CALLBACK_TYPE_ERROR で失敗します。

これは常に入力フィールドです。このフィールドの初期値は、MQCBT_MESSAGE_CONSUMER です。

Options (MQLONG)

コールバック記述子構造体 - Options フィールド

以下に説明する 1 つ以上のオプションを指定できます。複数のオプションを指定するには、値を加算する (同じ定数を複数回加算しない) か、ビット単位 OR 演算を使用して値を結合 (プログラミング言語でビット演算がサポートされる場合) します。

MQCBDO_FAIL_IF QUIESCING

MQCB 呼び出しは、キュー・マネージャーが静止状態にあるときは失敗します。

z/OS では、接続 (CICS または IMS アプリケーション用の) が静止状態になっている場合、このオプションは MQCB 呼び出しを強制的に失敗させる役割もします。

MQCB 呼び出しで渡される MQGMO オプション内に MQGMO_FAIL_IF QUIESCING を指定すると、これらの静止時にメッセージ・コンシューマーに対して通知が行われます。

制御オプション: 以下のオプションでは、コンシューマーの状態の変更時に、メッセージなしで、コールバック関数が呼び出されるかどうかを制御します。

MQCBDO_REGISTER_CALL

このコールバック関数は、呼び出しタイプ MQCBCT_REGISTER_CALL で呼び出されます。

MQCBDO_START_CALL

このコールバック関数は、呼び出しタイプ MQCBCT_START_CALL で呼び出されます。

MQCBDO_STOP_CALL

このコールバック関数は、呼び出しタイプ MQCBCT_STOP_CALL で呼び出されます。

MQCBDO_DEREGISTER_CALL

このコールバック関数は、呼び出しタイプ MQCBCT_DEREGISTER_CALL で呼び出されます。

MQCBDO_EVENT_CALL

このコールバック関数は、呼び出しタイプ MQCBCT_EVENT_CALL で呼び出されます。

MQCBDO_MC_EVENT_CALL

コールバック関数は、呼び出しタイプ MQCBCT_MC_EVENT_CALL で呼び出されます。

これらの呼び出しタイプについて詳しくは、[CallType](#) を参照してください。

デフォルト・オプション: 上記のいずれのオプションも必要でない場合は、以下のオプションを使用します。

MQCBDO_NONE

この値は、他のオプションが指定されなかったことを示すために使用します。すべてのオプションはデフォルト値であるとみなされます。

MQCBDO_NONE は、プログラムの文書化を支援するために定義します。このオプションは、他のオプションと組み合わせて使用するオプションではありません。ただし、このオプションの値はゼロと等価なので、他のオプションと組み合わせて使用しても、エラーとして検出されることはありません。

これは入力フィールドです。Options フィールドの初期値は、MQCBDO_NONE です。

CallbackArea (MQPTR)

コールバック記述子構造 - CallbackArea フィールド

これは、コールバック関数ができるフィールドです。

キュー・マネージャーは、このフィールドの内容に基づいて決定せず、MQCBC 構造中の CallbackArea フィールドから変更されずに渡されます。これはコールバック関数宣言上のパラメーターになります。

この値は、現在コールバックが定義されていない、値が MQOP_REGISTER の Operation のみで使用され、前の定義と置き換えられません。

これは、コールバック関数への入出力フィールドです。このフィールドの初期値は、ヌル・ポインターまたはヌル・バイトです。

CallbackFunction (MQPTR)

コールバック記述子構造 - CallbackFunction フィールド

このコールバック関数は、関数呼び出しとして呼び出されます。

このフィールドを使用して、コールバック関数へのポインターを指定します。

CallbackFunction または CallbackName のどちらかを指定しなければなりません。両方とも指定すると、理由コード MQRC_CALLBACK_ROUTINE_ERROR が戻されます。

CallbackName と CallbackFunction のどちらも設定されていないと、呼び出しは失敗し、理由コード MQRC_CALLBACK_ROUTINE_ERROR が戻されます。

このオプションは、次の環境ではサポートされません: プログラム言語およびコンパイラーが関数ポインター参照をサポートしていない。このようなシチュエーションでは、呼び出しは理由コード MQRC_CALLBACK_ROUTINE_ERROR で失敗します。

 z/OS では、OS リンケージ規約を使用してこの機能呼び出すことを予期しなければなりません。例えば、C プログラミング言語では、以下のように指定します。

```
#pragma linkage(MQCB_FUNCTION,OS)
```

これは入力フィールドです。このフィールドの初期値は、ヌル・ポインターまたはヌル・バイトです。

CallbackName (MQCHAR128)

コールバック記述子構造 - CallbackName フィールド

このコールバック関数は、動的リンク・プログラムとして呼び出されます。

CallbackFunction または CallbackName のどちらかを指定しなければなりません。両方とも指定すると、理由コード MQRC_CALLBACK_ROUTINE_ERROR が戻されます。

CallbackName と CallbackFunction のどちらも設定されない場合は、呼び出しは理由コード MQRC_CALLBACK_ROUTINE_ERROR で失敗します。

モジュールは、最初使用するコールバック・ルーチンが登録される際にロードされ、最後に使用するコールバック・ルーチンが登録解除される際にアンロードされます。

以下の本文で注記されている場合を除き、フィールド中で名前は組み込みブランクなしで左寄せされます。名前自体はフィールドの長さまでブランクを埋め込まれます。以下の説明では、大括弧 ([]) はオプション情報を表します。

IBM i

コールバック名は、次のいずれの形式にすることができます。

- Library "/" Program
- Library "/" ServiceProgram "("FunctionName")"

例えば、MyLibrary/MyProgram(MyFunction)などです。

ライブラリー名を *LIBL にすることができます。ライブラリー名とプログラム名は両方とも、最大 10 文字に制限されます。

AIX and Linux

コールバック名は、動的ロード可能なモジュールまたはライブラリーの名前で、このライブラリー中にある機能の名前が接尾部になります。関数名は括弧で囲む必要があります。ライブラリー名の前にはオプションでディレクトリー・パスを付けることができます。

```
[path]library(function)
```

パスを指定しないと、システム検索パスが使用されます。

この名前は最大 128 文字までに制限されています。

Windows

コールバック名は、ダイナミック・リンク・ライブラリーの名前で、このライブラリー中にある関数の名前が接尾部になります。機能名を括弧で囲まなければなりません。オプションで、以下のようにライブラリー名の前にディレクトリー・パスおよびドライブを付けることができます。

```
[d:][path]library(function)
```

ドライブとパスを指定しないと、システム検索パスが使用されます。

この名前は最大 128 文字までに制限されています。

z/OS

コールバック名は、LINK または LOAD マクロの EP パラメーター上の仕様にとって有効なロード・モジュールの名前です。

この名前は最大 8 文字までに制限されています。

z/OS CICS

コールバック名は、EXEC CICS LINK コマンドマクロの PROGRAM パラメータで指定可能なロードモジュールの名前です。

この名前は最大 8 文字までに制限されています。

プログラムは、インストール済みの PROGRAM 定義の REMOTESYTEM オプションを使用してリモートとして定義するか、または動的ルーティング・プログラムによって定義することができます。

プログラムが API 呼び出し IBM MQ を使用するには、遠隔 CICS 地域が IBM MQ 接続されている必要があります。しかし、MQCBC 構造中の Hobj フィールドはリモート・システム中では無効であることに注意してください。

CallbackName をロードしようとして障害が発生すると、以下のいずれかのエラー・コードがアプリケーションに戻されます。

- MQRC_MODULE_NOT_FOUND
- MQRC_MODULE_INVALID
- MQRC_MODULE_ENTRY_NOT_FOUND

また、メッセージがエラー・ログに書き込まれ、ロードが試行されたモジュールの名前と、オペレーティング・システムからの失敗理由コードが含まれます。

これは入力フィールドです。このフィールドの初期値は、ヌル・ストリングまたはブランクです。

MaxMsgLength (MQLONG)

これは、ハンドルから読み取り、コールバック・ルーチンに渡すことができる、最長メッセージの長さ (バイト数) です。

メッセージの長さがこれより長い場合は、コールバック・ルーチンは *MaxMsgLength* バイトのメッセージと以下の理由コードを受け取ります。

- MQRC_TRUNCATED_MSG_FAILED または
- MQGMO_ACCEPT_TRUNCATED_MSG を指定した場合は MQRC_TRUNCATED_MSG_ACCEPTED

実際のメッセージ長は、MQCBC 構造の *DataLength* フィールド中に提供されます。

以下のような特殊値が定義されます。

MQCBD_FULL_MSG_LENGTH

バッファ長はシステムにより、切り捨てなしでメッセージを返すように調整されます。

メッセージを受け取るためのバッファを割り振るのに使用できるメモリーが足りない場合は、システムは MQRC_STORAGE_NOT_AVAILABLE 理由コードでコールバック関数を呼び出します。

例えば、データ変換を要求し、メッセージ・データを変換するために使用できる十分なメモリーがない場合、未変換のメッセージはコールバック関数に渡されます。

これは入力フィールドです。 *MaxMsgLength* フィールドの初期値は、MQCBD_FULL_MSG_LENGTH です。

MQCHARV - 可変長ストリング

MQCHARV 構造体は、可変長ストリングを記述するのに使用します。

可用性

MQCHARV 構造体は、以下のプラットフォームで利用可能です：

-  AIX
-  IBM i
-  Linux
-  Windows

および、これらのシステムに接続された IBM MQ MQI clients。

文字セットとエンコード

MQCHARV 内のデータは、MQENC_NATIVE で指定されるローカルキューマネージャのエンコーディングおよび構造体内の VSCCSID フィールドの文字セットでなければならない。アプリケーションが MQ クライアントとして実行している場合、構造体はクライアントのエンコード内になければなりません。エンコードによって表記が変わる文字セットもあります。VSCCSID がこれらの文字セットの 1 つである場合、MQCHARV のその他のフィールドと同じエンコードが使用されます。VSCCSID で識別される文字セットは、2 バイト文字セット (DBCS) も可能です。

使用法

MQCHARV 構造体は、それを含む構造体と連続していない可能性のあるデータを扱う。このデータをアドレッシングするには、ポインター・データ・タイプで宣言されるフィールドを使用できます。COBOL では、ポインター・データ・タイプはどの環境でもサポートされないので注意してください。この場合には、MQCHARV を含む構造体の先頭からのデータのオフセットを含むフィールドを使用すると、データをアドレッシングできます。

フィールド

注: 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 478. MQCHARV のフィールド		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
<u>VSPtr</u> （可変長文字列へのポインタ）	なし	ヌル・ポインタまたはヌル・バイト。
<u>VSOOffset</u> （この MQCHARV 構造体を含む構造体の先頭からの可変長文字列のオフセット（バイト単位））	なし	0
<u>VSBufSize</u> (VSPtr または VSOOffset フィールドが指すバッファのバイト単位のサイズ)	MQVS_USE_VSLENGTH	0
<u>VSLength</u> (VSPtr または VSOOffset フィールドが指す可変長文字列のバイト単位の長さ)	なし	0
<u>VSCCSID</u> (VSPtr または VSOOffset フィールドによってアドレス指定される可変長文字列の文字セット識別子)	MQCCSI_APPL	-3
注: C プログラミング言語では、マクロ変数 MQCHARV_DEFAULT に表に記載された値が設定されています。この変数を以下の方法で使用すると、構造体のフィールドに初期値を設定できます。 MQCHARV MyVarStr = {MQCHARV_DEFAULT};		

言語ごとの宣言

MQCHARV の C 宣言

```
typedef struct tagMQCHARV MQCHARV;  
struct tagMQCHARV {  
    MQPTR      VSPtr;                /* Address of variable length string */  
    MQLONG     VSOOffset;            /* Offset of variable length string */  
    MQLONG     VSBufSize;            /* Size of buffer */  
    MQLONG     VSLength;            /* Length of variable length string */  
    MQLONG     VSCCSID;              /* CCSID of variable length string */  
};
```

MQCHARV の COBOL 宣言

```
** MQCHARV structure  
10  MQCHARV.  
** Address of variable length string  
15  MQCHARV-VSPTR      POINTER.  
** Offset of variable length string  
15  MQCHARV-VSOFFSET   PIC S9(9) BINARY.  
** Size of buffer  
15  MQCHARV-VSBUFSIZE  PIC S9(9) BINARY.  
** Length of variable length string  
15  MQCHARV-VSLENGTH  PIC S9(9) BINARY.  
** CCSID of variable length string  
15  MQCHARV-VSCCSID    PIC S9(9) BINARY.
```

注: COBOL アプリケーションを環境間で移植する場合、対象となるすべての環境でポインタデータ型が利用可能かどうかを確認する必要があります。ポインタ・データ・タイプが使用できない環境では、アプリケーションでのデータのアドレッシングは、ポインタ・フィールドではなくオフセット・フィールドを使って実行しなければなりません。ポインタがサポートされていない環境では、ポインタフィールドを適切な長さのバイト列として宣言でき、初期値は全ヌルバイト列とします。オフセット・フィールドを使用している場合は、この初期値を変更しないでください。提供されているサンプル集に変更を加えずに行う方法の一例を以下に示します。

```
COPY CMQCHRVV REPLACING POINTER BY ==BINARY PIC S9(9)==.
```

CMQCHRVV を、使用するサンプル集と交換できます。

MQCHARV の PL/I 宣言

```
decl
1 MQCHARV based,
3 VSPtr      pointer,      /* Address of variable length string */
3 VSOffset   fixed bin(31), /* Offset of variable length string */
3 VBufSize   fixed bin(31), /* Size of buffer */
3 VSLength   fixed bin(31), /* Length of variable length string */
3 VSCCSID    fixed bin(31); /* CCSID of variable length string */
```

MQCHARV の高水準アセンブラ宣言

```
MQCHARV          DSECT
MQCHARV_VSPTR     DS    F      Address of variable length string
MQCHARV_VSOFFSET  DS    F      Offset of variable length string
MQCHARV_VBUFSIZE  DS    F      Size of buffer
MQCHARV_VSLNGTH   DS    F      Length of variable length string
MQCHARV_VSCCSID   DS    F      CCSID of variable length string
*
MQCHARV_LENGTH    EQU    *-MQCHARV
ORG    MQCHARV
MQCHARV_AREA      DS     CL(MQCHARV_LENGTH)
```

MQCHARV のフィールド詳細

MQCHARV 構造体は、可変長ストリングを記述するのに使用します。

フィールド

注: 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 479. MQCHARV のフィールド		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
<u>VSPtr</u> （可変長文字列へのポインタ）	なし	ヌル・ポインタまたはヌル・バイト。
<u>VSOffset</u> （この MQCHARV 構造体を含む構造体の先頭からの可変長文字列のオフセット（バイト単位））	なし	0
<u>VBufSize</u> (VSPtr または VSOffset フィールドが指すバッファのバイト単位のサイズ)	MQVS_USE_VSLENGTH	0
<u>VSLength</u> （VSPtr または VSOffset フィールドが指す可変長文字列のバイト単位の長さ）	なし	0
<u>VSCCSID</u> (VSPtr または VSOffset フィールドによってアドレス指定される可変長文字列の文字セット識別子)	MQCCSI_APPL	-3
注: C プログラミング言語では、マクロ変数 MQCHARV_DEFAULT に表に記載された値が設定されています。この変数を以下の方法で使用すると、構造体のフィールドに初期値を設定できます。		
MQCHARV MyVarStr = {MQCHARV_DEFAULT};		

VSPtr (MQPTR)

これは可変長ストリングを指すポインタです。

VSPtr フィールドか VSOffset フィールドのどちらかを使用して可変長ストリングを指定できますが、両方とも指定することはできません。

このフィールドの初期値は、ヌル・ポインターまたはヌル・バイトです。

VSOffset (MQLONG)

MQCHARV または MQCHARV を含む構造体の先頭からの可変長ストリングのオフセットをバイト数で表したものです。

MQCHARV 構造体が別の構造体に組み込まれている場合、この値は、この MQCHARV 構造体が含まれる構造体の先頭からの可変長ストリングのオフセット (バイト単位) です。MQCHARV 構造体が別の構造体に組み込まれていない場合、例えば、これが関数呼び出しにおけるパラメーターとして指定された場合、オフセットは MQCHARV 構造体の先頭からの相対位置です。

オフセットの値は、正負どちらの値にもなります。

VSPtr フィールドか VSOffset フィールドのどちらかを使用して可変長ストリングを指定できますが、両方とも指定することはできません。

このフィールドの初期値は 0 です。

VSBufSize (MQLONG)

これは、VSPtr または VSOffset フィールドでアドレス指定されたバッファのバイトのサイズです。

MQCHARV 構造は、関数呼び出しの出力フィールドとして使用されます。このフィールドは、指定されたバッファの長さで初期化される必要があります。VLength の値が VSBufSize よりも大きい場合は、VSBufSize バイトのデータがバッファの呼び出し元に戻されます。

この値は、ゼロ以上であるか、または以下の認識される特殊値でなければなりません。

MQVS_USE_VSLENGTH

指定されている場合、MQCHARV 構造の VLength フィールドからバッファの長さが取られます。構造を出力フィールドとして使用し、バッファが指定されているときは、この値を使用しないでください。

これはこのフィールドの初期値です。

VLength (MQLONG)

VSPtr フィールドまたは VSOffset フィールドで指定された可変長ストリングのバイト単位の長さ。

このフィールドの初期値は 0 です。この値は、ゼロ以上、または認識される以下の特殊値のいずれかでなければなりません。

MQVS_NULL_TERMINATED

MQVS_NULL_TERMINATED を指定しないと、ストリングの一部として VLength のバイトが組み込まれます。ヌル文字があってもストリングは区切られません。

MQVS_NULL_TERMINATED を指定すると、ストリング中の最初のヌルでそのストリングが区切られます。ヌル自体はそのストリングの一部として組み込まれません。

注: MQVS_NULL_TERMINATED を指定する場合、VSCCSID で指定されたコード・セットからのヌル文字を使用して、ストリングを終了します。

例えば、UTF-16 (CCSID 1200、13488、および 17584) では 2 バイトの Unicode エンコードで、ヌルはすべてゼロの 16 ビットの数値で表されます。UTF-16 では、すべてゼロに設定された 1 バイトが文字の一部になっていることは一般的ですが (例えば、7 ビットの ASCII 文字)、偶数バイト境界に 2 つの「ゼロ」バイトがある場合のみストリングはヌル終了になります。奇数境界に 2 つの「ゼロ」バイトがある場合は、それらのバイトが有効な文字の個々の部分である場合に読み取れます。例えば、x'01' x'00 x'00' x'30' は 2 つの有効な Unicode 文字なので、ストリングはヌル終了しません。

VSCCSID (MQLONG)

これは、**VSPtr** フィールドまたは **VSoffset** フィールドで指定された可変長ストリングの文字セット ID です。

このフィールドの初期値は、MQ によって定義され、現行プロセスの本物の文字セット ID に変更される必要があることを示す **MQCCSI_APPL** です。そのため、**MQCCSI_APPL** という定数の値が可変長ストリングに関連付けられることはありません。

コンパイル単位の定数 **MQCCSI_APPL** に別の値を定義すると、このフィールドの初期値を変更できます。それを行う方法は、使用しているアプリケーションのプログラミング言語によって異なります。

 システム z/OS 上、**MQCCSI_APPL** が使用する CCSID デフォルトアプリケーションは以下のよう

- DLL インターフェースを使用するバッチ LE アプリケーションの場合、**MQCONN** の発行時の現行ロケールに関連付けられている CODESET がデフォルトです (デフォルト値は 1047)。
- バッチ MQ スタブの 1 つとバインドされているバッチ LE アプリケーションの場合、**MQCONN** の後の最初の MQI 呼び出しの発行時の現行ロケールに関連付けられている CODESET がデフォルトです (デフォルト値は 1047)。
- スレッド z/OS UNIX System Services 上で実行されるバッチ非 LE アプリケーションの場合、デフォルト値は THLICCSID、MQI が最初に呼び出された時点での値となります (デフォルト値は **MQCONN** 1047 です)。
- その他のバッチ・アプリケーションの場合、キュー・マネージャーの CCSID がデフォルトです。

注：非バッチアプリケーションの場合、.CCSID を手動で定義する必要があります。

MQCCSI_APPL の再定義

以下の例は、様々なプログラミング言語で **MQCCSI_APPL** の値を上書きする方法を示しています。の値を変更すべきです **MQCCSI_APPL**。これにより、各可変長文字列ごとに個別に **VSCCSID** を設定する必要がなくなり、正しい CCSID が使用されることが保証されます。

これらの例では、CCSID が 1208 に設定されています。この値を、必要な値に変更します。この値はデフォルト値になるので、**MQCHARV** の特定のインスタンスで **VSCCSID** を設定して指定変更できます。

C の使用

```
#define MQCCSI_APPL 1208
#include <cmqc.h>
```

COBOL の使用

```
COPY CMQXYZV REPLACING -3 BY 1208.
```

PL/I の使用

```
%MQCCSI_APPL = '1208';
%include syslib(cmqp);
```

High Level Assembler 使用法

```
MQCCSI_APPL EQU 1208
CMQA LIST=NO
```

MQCIH - CICS bridge ヘッダー

MQCIH 構造は、MQ ネットワークを CICS 介して送信される CICS bridge メッセージのヘッダー情報を記述する。

サポートされている IBM MQ プラットフォームであれば、MQCIH 構造を含むメッセージを作成して送信できますが、この構造を使用できるのは IBM MQ for z/OS キュー CICS bridge マネージャのみです。したが

って、メッセージが非キュー z/OS マネージャから CICS に到達するためには、メッセージがルーティングされる少なくとも 1 つの z/OS キューマネージャを、あなたのキューマネージャネットワークが包含している必要があります。

によってサポートされているすべての CICSIBM MQ 9.0.0 バージョン、およびそれ以降のバージョンは、提供された CICS バージョンのブリッジを使用します。アダプタ CICSIBM MQ および IBM MQCICS bridge コンポーネントの設定に関する詳細については、ドキュメント CICS の「[MQ への接続の設定](#)」セクションを参照してください。

可用性

MQCIH 構造体は以下のプラットフォームで利用可能です：

-  AIX
-  Linux
-  Windows
-  z/OS

および、これらのシステムに接続された IBM MQ MQI clients。

形式名

MQFMT_CICS

バージョン

MQCIH の現在のバージョンは MQCIH_VERSION_2 です。構造体のより新しいバージョンにのみ存在するフィールドは、以下の説明においてその旨が明記されています。

サポートされるプログラム言語用に提供されているヘッダー・ファイル、COPY ファイル、および INCLUDE ファイルには、MQCIH の最新バージョンが含まれており、*Version* フィールドの初期値は、MQCIH_VERSION_2 となっています。

文字セットとエンコード

MQCIH 構造体およびアプリケーションメッセージデータに使用される文字セットとエンコーディングには特別な条件が適用されます：

- CICS bridge・キューを所有するキュー・マネージャに接続するアプリケーションは、そのキュー・マネージャの文字セットとエンコードで記述した MQCIH 構造体を渡す必要があります。この場合には MQCIH 構造体のデータ変換が実行されないためです。
- 他のキュー・マネージャに接続するアプリケーションは、サポートされている任意の文字セットとエンコードで記述した MQCIH 構造体を渡すことができます。CICS bridge・キューを所有するキュー・マネージャに接続された受信側のメッセージ・チャネル・エージェントが、MQCIH 構造体を変換するからです。
- MQCIH 構造体の後に続くアプリケーション・メッセージ・データは、MQCIH 構造体と同じ文字セットとエンコードで記述されていなければなりません。MQCIH 構造体の *CodedCharSetId* フィールドおよび *Encoding* フィールドを使用して、そのアプリケーション・メッセージ・データの文字セットとエンコードを指定することはできません。

データがキュー・マネージャにサポートされる組み込み形式でない場合、アプリケーション・メッセージ・データを変換するために、ユーザーはデータ変換出口を提供することが必要です。

使用法

アプリケーションが、246 ページの表 481 に示された初期値と同じ値を必要とし、かつブリッジが AUTH=LOCAL または AUTH=IDENTIFY で動作している場合、メッセージから MQCIH 構造体を省略できます。それ以外の場合、この構造体は必要です。

ブリッジは、MQCIH 構造体のバージョン 1 とバージョン 2 のどちらをも受け入れています。ただし、3270 トランザクションにおいては、バージョン 2 の構造体を使用する必要があります。

「要求」フィールドと記されているフィールドでは、ブリッジに送信されるメッセージ内の値を、そのアプリケーション側で適切に設定しておく必要があります。これらのフィールドはブリッジに対する入力となります。

応答フィールドと記されているフィールドは、CICS bridge が、ブリッジからアプリケーションに送信する応答メッセージ内に設定します。ReturnCode、Function、CompCode、Reason、AbendCode などのフィールド内には、エラー情報が戻されます。ただし、すべてのケースでこれらすべてのフィールドが設定されているとは限りません。以下の表は、の異なる値に対してどの ReturnCode フィールドが設定されるかを示しています。

表 480. MQCIH 構造体のエラー情報フィールドの内容 (MQCIH)				
ReturnCode	Function	CompCode	Reason	AbendCode
MQCRC_OK	-	-	-	-
MQCRC_BRIDGE_ERROR	-	-	MQFB_CICS_*	-
MQCRC_MQ_API_ERROR MQCRC_BRIDGE_TIMEOUT	MQ 呼び出し名	MQ CompCode	MQ Reason	-
MQCRC_CICS_EXEC_ERROR MQCRC_SECURITY_ERROR MQCRC_PROGRAM_NOT_AVAILABLE MQCRC_TRANSID_NOT_AVAILABLE	CICS EIBFN	CICS EIBRESP	CICS EIBRESP2	-
MQCRC_BRIDGE_ABEND MQCRC_APPLICATION_ABEND	-	-	-	CICS ABCODE

フィールド

注: 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 481. MQCIH のフィールド		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
StrucId (構造識別子)	MQCIH_STRUC_ID	'CIH '
バージョン (構造バージョン番号)	MQCIH_VERSION_2	2
StrucLength (MQCIH 構造の長さ)	MQCIH_LENGTH_2	180
エンコーディング (予約済み)	なし	0
CodedCharSetId (予約済み)	なし	0
フォーマット (MQ に続くデータのフォーマット名)	MQFMT_NONE	ブランク
フラグ (フラグ)	MQCIH_NONE	0
ReturnCode (ブリッジからの戻りコード)	MQCRC_OK	0
CompCode (MQ 完了コードまたは EIBRESP CICS)	MQCC_OK	0

表 481. MQCIH のフィールド (続き)

フィールド名と説明	定数の名前	定数の初期値 (存在する場合)
理由 (MQ reason またはフィールドバックコード、または CICSEIBRESP2)	MQRC_NONE	0
UOW 制御 (作業単位制御)	MQCUOWC_ONLY	273
GetWaitInterval (ブリッジタスクによって発行された MQGET 呼び出しの待機間隔)	MQCGWI_DEFAULT	-2
LinkType (リンクタイプ)	MQCLT_PROGRAM	1
OutputDataLength (出力 COMMAREA データ長)	MQCODL_AS_INPUT	-1
FacilityKeepTime (ブリッジ施設解放時間)	なし	0
ADS 記述子 (ADS 記述子の送信/受信)	MQCADSD_NONE	0
ConversationalTask (タスクが会話形式で処理可能かどうか)	MQCCT_NO	0
TaskEndStatus (タスク終了時点の状況)	MQCTES_NOSYNC	0
施設 (ブリッジ施設トークン)	MQCFAC_NONE	Null
関数 (MQ 呼び出し名または EIBFN 関数 CICS)	MQCFUNC_NONE	ブランク
AbendCode (アベンドコード)	なし	ブランク
認証ツール (パスワードまたはパスチケット)	なし	ブランク
Reserved1 (予約済み)	なし	ブランク
ReplyToFormat (MQ 返信メッセージのフォーマット名)	MQFMT_NONE	ブランク
RemoteSysId (使用するリモート CICS システム ID)	なし	ブランク
RemoteTransId (使用する CICS RTRANSID)	なし	ブランク
TransactionId (添付する取引)	なし	ブランク
FacilityLike (端末エミュレーション属性)	なし	ブランク
AttentionId (AID キー)	なし	ブランク
StartCode (トランザクション開始コード)	MQCSC_NONE	ブランク
CancelCode (abend トランザクションコード)	なし	ブランク
NextTransactionId (次に添付する取引)	なし	ブランク
Reserved2 (予約済み)	なし	ブランク
Reserved3 (予約済み)	なし	ブランク
注: Version が MQCIH_VERSION_2 より小さい場合は、残りのフィールドは表示されません。		
CursorPosition (カーソル位置)	なし	0
ErrorOffset (メッセージ内の誤差のオフセット)	なし	0
InputItem (入力項目)	なし	0
Reserved4 (予約済み)	なし	0

表 481. MQCIH のフィールド (続き)

フィールド名と説明	定数の名前	定数の初期値 (存在する場合)
注: 1. 記号-は、単一のブランク文字を表します。 2. C プログラミング言語では、マクロ変数 MQCIH_DEFAULT には、表にリストされている値が含まれています。このマクロ変数を以下の方法で使用して、構造体のフィールドに初期値を設定します。 <pre>MQCIH MyCIH = {MQCIH_DEFAULT};</pre>		

言語ごとの宣言

MQCIH の C 宣言

```
typedef struct tagMQCIH MQCIH;
struct tagMQCIH {
    MQCHAR4  StrucId;           /* Structure identifier */
    MQLONG   Version;          /* Structure version number */
    MQLONG   StrucLength;      /* Length of MQCIH structure */
    MQLONG   Encoding;         /* Reserved */
    MQLONG   CodedCharSetId;    /* Reserved */
    MQCHAR8  Format;           /* MQ format name of data that follows
                               MQCIH */
    MQLONG   Flags;            /* Flags */
    MQLONG   ReturnCode;       /* Return code from bridge */
    MQLONG   CompCode;        /* MQ completion code or CICS EIBRESP */
    MQLONG   Reason;          /* MQ reason or feedback code, or CICS
                               EIBRESP2 */
    MQLONG   UOWControl;       /* Unit-of-work control */
    MQLONG   GetWaitInterval; /* Wait interval for MQGET call issued
                               by bridge task */
    MQLONG   LinkType;         /* Link type */
    MQLONG   OutputDataLength; /* Output COMMAREA data length */
    MQLONG   FacilityKeepTime; /* Bridge facility release time */
    MQLONG   ADSDescriptor;     /* Send/receive ADS descriptor */
    MQLONG   ConversationalTask; /* Whether task can be conversational */
    MQLONG   TaskEndStatus;     /* Status at end of task */
    MQBYTE8  Facility;         /* Bridge facility token */
    MQCHAR4  Function;         /* MQ call name or CICS EIBFN
                               function */
    MQCHAR4  AbendCode;        /* Abend code */
    MQCHAR8  Authenticator;    /* Password or passticket */
    MQCHAR8  Reserved1;        /* Reserved */
    MQCHAR8  ReplyToFormat;    /* MQ format name of reply message */
    MQCHAR4  RemoteSysId;      /* Reserved */
    MQCHAR4  RemoteTransId;    /* Reserved */
    MQCHAR4  TransactionId;    /* Transaction to attach */
    MQCHAR4  FacilityLike;     /* Terminal emulated attributes */
    MQCHAR4  AttentionId;      /* AID key */
    MQCHAR4  StartCode;        /* Transaction start code */
    MQCHAR4  CancelCode;       /* Abend transaction code */
    MQCHAR4  NextTransactionId; /* Next transaction to attach */
    MQCHAR8  Reserved2;        /* Reserved */
    MQCHAR8  Reserved3;        /* Reserved */
    MQLONG   CursorPosition;   /* Cursor position */
    MQLONG   ErrorOffset;      /* Offset of error in message */
    MQLONG   InputItem;        /* Reserved */
    MQLONG   Reserved4;        /* Reserved */
};
```

MQCIH の COBOL 宣言

```
**      MQCIH structure
10 MQCIH.
**      Structure identifier
15 MQCIH-STRUCID          PIC X(4).
**      Structure version number
15 MQCIH-VERSION          PIC S9(9) BINARY.
**      Length of MQCIH structure
15 MQCIH-STRUCLNGTH      PIC S9(9) BINARY.
```

```

**      Reserved
15 MQCIH-ENCODING          PIC S9(9) BINARY.
**      Reserved
15 MQCIH-CODEDCHARSETID   PIC S9(9) BINARY.
**      MQ format name of data that follows MQCIH
15 MQCIH-FORMAT           PIC X(8).
**      Flags
15 MQCIH-FLAGS            PIC S9(9) BINARY.
**      Return code from bridge
15 MQCIH-RETURNCODE       PIC S9(9) BINARY.
**      MQ completion code or CICS EIBRESP
15 MQCIH-COMPCODE         PIC S9(9) BINARY.
**      MQ reason or feedback code, or CICS EIBRESP2
15 MQCIH-REASON           PIC S9(9) BINARY.
**      Unit-of-work control
15 MQCIH-UOWCONTROL       PIC S9(9) BINARY.
**      Wait interval for MQGET call issued by bridge task
15 MQCIH-GETWAITINTERVAL  PIC S9(9) BINARY.
**      Link type
15 MQCIH-LINKTYPE         PIC S9(9) BINARY.
**      Output COMMAREA data length
15 MQCIH-OUTPUTDATALENGTH PIC S9(9) BINARY.
**      Bridge facility release time
15 MQCIH-FACILITYKEEPTIME PIC S9(9) BINARY.
**      Send/receive ADS descriptor
15 MQCIH-ADSDESCRIPTOR    PIC S9(9) BINARY.
**      Whether task can be conversational
15 MQCIH-CONVERSATIONALTASK PIC S9(9) BINARY.
**      Status at end of task
15 MQCIH-TASKENDSTATUS    PIC S9(9) BINARY.
**      Bridge facility token
15 MQCIH-FACILITY         PIC X(8).
**      MQ call name or CICS EIBFN function
15 MQCIH-FUNCTION         PIC X(4).
**      Abend code
15 MQCIH-ABENDCODE        PIC X(4).
**      Password or passticket
15 MQCIH-AUTHENTICATOR    PIC X(8).
**      Reserved
15 MQCIH-RESERVED1        PIC X(8).
**      MQ format name of reply message
15 MQCIH-REPLYTOFORMAT    PIC X(8).
**      Reserved
15 MQCIH-REMOTESYSID      PIC X(4).
**      Reserved
15 MQCIH-REMOTETRANSID    PIC X(4).
**      Transaction to attach
15 MQCIH-TRANSACTIONID    PIC X(4).
**      Terminal emulated attributes
15 MQCIH-FACILITYLIKE     PIC X(4).
**      AID key
15 MQCIH-ATTENTIONID      PIC X(4).
**      Transaction start code
15 MQCIH-STARTCODE        PIC X(4).
**      Abend transaction code
15 MQCIH-CANCELCODE       PIC X(4).
**      Next transaction to attach
15 MQCIH-NEXTTRANSACTIONID PIC X(4).
**      Reserved
15 MQCIH-RESERVED2        PIC X(8).
**      Reserved
15 MQCIH-RESERVED3        PIC X(8).
**      Cursor position
15 MQCIH-CURSORPOSITION   PIC S9(9) BINARY.
**      Offset of error in message
15 MQCIH-ERROROFFSET      PIC S9(9) BINARY.
**      Reserved
15 MQCIH-INPUTITEM        PIC S9(9) BINARY.
**      Reserved
15 MQCIH-RESERVED4        PIC S9(9) BINARY.

```

MQCIH の PL/I 宣言

```

dcl
  1 MQCIH based,
    3 StrucId          char(4),          /* Structure identifier */
    3 Version          fixed bin(31), /* Structure version number */
    3 StrucLength       fixed bin(31), /* Length of MQCIH structure */
    3 Encoding         fixed bin(31), /* Reserved */
    3 CodedCharSetId    fixed bin(31), /* Reserved */

```

3 Format	char(8),	/* MQ format name of data that follows MQCIH */
3 Flags	fixed bin(31),	/* Flags */
3 ReturnCode	fixed bin(31),	/* Return code from bridge */
3 CompCode	fixed bin(31),	/* MQ completion code or CICS EIBRESP */
3 Reason	fixed bin(31),	/* MQ reason or feedback code, or CICS EIBRESP2 */
3 UOWControl	fixed bin(31),	/* Unit-of-work control */
3 GetWaitInterval	fixed bin(31),	/* Wait interval for MQGET call issued by bridge task */
3 LinkType	fixed bin(31),	/* Link type */
3 OutputDataLength	fixed bin(31),	/* Output COMMAREA data length */
3 FacilityKeepTime	fixed bin(31),	/* Bridge facility release time */
3 ADSDescriptor	fixed bin(31),	/* Send/receive ADS descriptor */
3 ConversationalTask	fixed bin(31),	/* Whether task can be conversational */
3 TaskEndStatus	fixed bin(31),	/* Status at end of task */
3 Facility	char(8),	/* Bridge facility token */
3 Function	char(4),	/* MQ call name or CICS EIBFN function */
3 AbendCode	char(4),	/* Abend code */
3 Authenticator	char(8),	/* Password or passticket */
3 Reserved1	char(8),	/* Reserved */
3 ReplyToFormat	char(8),	/* MQ format name of reply message */
3 RemoteSysId	char(4),	/* Reserved */
3 RemoteTransId	char(4),	/* Reserved */
3 TransactionId	char(4),	/* Transaction to attach */
3 FacilityLike	char(4),	/* Terminal emulated attributes */
3 AttentionId	char(4),	/* AID key */
3 StartCode	char(4),	/* Transaction start code */
3 CancelCode	char(4),	/* Abend transaction code */
3 NextTransactionId	char(4),	/* Next transaction to attach */
3 Reserved2	char(8),	/* Reserved */
3 Reserved3	char(8),	/* Reserved */
3 CursorPosition	fixed bin(31),	/* Cursor position */
3 ErrorOffset	fixed bin(31),	/* Offset of error in message */
3 InputItem	fixed bin(31),	/* Reserved */
3 Reserved4	fixed bin(31);	/* Reserved */

MQCIH の高水準アセンブラ宣言

MQCIH	DSECT	
MQCIH_STRUCID	DS	CL4 Structure identifier
MQCIH_VERSION	DS	F Structure version number
MQCIH_STRUCLNGTH	DS	F Length of MQCIH structure
MQCIH_ENCODING	DS	F Reserved
MQCIH_CODEDCHARSETID	DS	F Reserved
MQCIH_FORMAT	DS	CL8 MQ format name of data that follows MQCIH
*		
MQCIH_FLAGS	DS	F Flags
MQCIH_RETURNCODE	DS	F Return code from bridge
MQCIH_COMPCODE	DS	F MQ completion code or CICS EIBRESP
MQCIH_REASON	DS	F MQ reason or feedback code, or CICS EIBRESP2
*		
MQCIH_UOWCONTROL	DS	F Unit-of-work control
MQCIH_GETWAITINTERVAL	DS	F Wait interval for MQGET call issued by bridge task
*		
MQCIH_LINKTYPE	DS	F Link type
MQCIH_OUTPUTDATALENGTH	DS	F Output COMMAREA data length
MQCIH_FACILITYKEEPTIME	DS	F Bridge facility release time
MQCIH_ADSDESCRIPTOR	DS	F Send/receive ADS descriptor
MQCIH_CONVERSATIONALTASK	DS	F Whether task can be conversational
MQCIH_TASKENDSTATUS	DS	F Status at end of task
MQCIH_FACILITY	DS	XL8 Bridge facility token
MQCIH_FUNCTION	DS	CL4 MQ call name or CICS EIBFN function
MQCIH_ABENDCODE	DS	CL4 Abend code
MQCIH_AUTHENTICATOR	DS	CL8 Password or passticket
MQCIH_RESERVED1	DS	CL8 Reserved
MQCIH_REPLYTOFORMAT	DS	CL8 MQ format name of reply message
MQCIH_REMOTESYSID	DS	CL4 Reserved
MQCIH_REMOTETRANSID	DS	CL4 Reserved
MQCIH_TRANSACTIONID	DS	CL4 Transaction to attach
MQCIH_FACILITYLIKE	DS	CL4 Terminal emulated attributes
MQCIH_ATTENTIONID	DS	CL4 AID key
MQCIH_STARTCODE	DS	CL4 Transaction start code
MQCIH_CANCELCODE	DS	CL4 Abend transaction code
MQCIH_NEXTTRANSACTIONID	DS	CL4 Next transaction to attach
MQCIH_RESERVED2	DS	CL8 Reserved

MQCIH_RESERVED3	DS	CL8	Reserved
MQCIH_CURSORPOSITION	DS	F	Cursor position
MQCIH_ERROROFFSET	DS	F	Offset of error in message
MQCIH_INPUTITEM	DS	F	Reserved
MQCIH_RESERVED4	DS	F	Reserved
*			
MQCIH_LENGTH	EQU	*-MQCIH	
	ORG	MQCIH	
MQCIH_AREA	DS	CL(MQCIH_LENGTH)	

MQCIH の Visual Basic 宣言

```

Type MQCIH
  StrucId      As String*4 'Structure identifier'
  Version      As Long     'Structure version number'
  StrucLength  As Long     'Length of MQCIH structure'
  Encoding     As Long     'Reserved'
  CodedCharSetId As Long   'Reserved'
  Format       As String*8 'MQ format name of data that follows'
                'MQCIH'
  Flags       As Long     'Flags'
  ReturnCode  As Long     'Return code from bridge'
  CompCode    As Long     'MQ completion code or CICS EIBRESP'
  Reason      As Long     'MQ reason or feedback code, or CICS'
                'EIBRESP2'
  UOWControl  As Long     'Unit-of-work control'
  GetWaitInterval As Long 'Wait interval for MQGET call issued'
                'by bridge task'
  LinkType    As Long     'Link type'
  OutputDataLength As Long 'Output COMMAREA data length'
  FacilityKeepTime As Long 'Bridge facility release time'
  ADSDescriptor As Long   'Send/receive ADS descriptor'
  ConversationalTask As Long 'Whether task can be conversational'
  TaskEndStatus As Long   'Status at end of task'
  Facility     As MQBYTE8 'Bridge facility token'
  Function     As String*4 'MQ call name or CICS EIBFN function'
  AbendCode    As String*4 'Abend code'
  Authenticator As String*8 'Password or passticket'
  Reserved1    As String*8 'Reserved'
  ReplyToFormat As String*8 'MQ format name of reply message'
  RemoteSysId  As String*4 'Reserved'
  RemoteTransId As String*4 'Reserved'
  TransactionId As String*4 'Transaction to attach'
  FacilityLike  As String*4 'Terminal emulated attributes'
  AttentionId   As String*4 'AID key'
  StartCode     As String*4 'Transaction start code'
  CancelCode    As String*4 'Abend transaction code'
  NextTransactionId As String*4 'Next transaction to attach'
  Reserved2     As String*8 'Reserved'
  Reserved3     As String*8 'Reserved'
  CursorPosition As Long    'Cursor position'
  ErrorOffset   As Long    'Offset of error in message'
  InputItem     As Long    'Reserved'
  Reserved4     As Long    'Reserved'
End Type

```

MQCIH のフィールド詳細

MQCIH 構造は、MQ ネットワークを CICS 介して送信される CICS bridge メッセージのヘッダー情報を記述する。

フィールド

注：次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 482. MQCIH における MQCIH のフィールド		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
<u>StrucId</u> (構造識別子)	MQCIH_STRUC_ID	'CIH'
<u>バージョン</u> (構造バージョン番号)	MQCIH_VERSION_2	2
<u>StrucLength</u> (MQCIH 構造の長さ)	MQCIH_LENGTH_2	180

表 482. MQCIH における MQCIH のフィールド (続き)

フィールド名と説明	定数の名前	定数の初期値 (存在する場合)
<u>エンコーディング</u> (予約済み)	なし	0
<u>CodedCharSetId</u> (予約済み)	なし	0
<u>フォーマット</u> (MQ に続くデータのフォーマット名)	MQFMT_NONE	ブランク
<u>フラグ</u> (フラグ)	MQCIH_NONE	0
<u>ReturnCode</u> (ブリッジからの戻りコード)	MQCRC_OK	0
<u>CompCode</u> (MQ 完了コードまたは EIBRESP CICS)	MQCC_OK	0
<u>理由</u> (MQ reason またはフィールドバックコード、または CICSEIBRESP2)	MQRC_NONE	0
<u>UOW 制御</u> (作業単位制御)	MQCUOWC_ONLY	273
<u>GetWaitInterval</u> (ブリッジタスクによって発行された MQGET 呼び出しの待機間隔)	MQCGWI_DEFAULT	-2
<u>LinkType</u> (リンクタイプ)	MQCLT_PROGRAM	1
<u>OutputDataLength</u> (出力 COMMAREA データ長)	MQCODL_AS_INPUT	-1
<u>FacilityKeepTime</u> (ブリッジ施設解放時間)	なし	0
<u>ADS 記述子</u> (ADS 記述子の送信/受信)	MQCADSD_NONE	0
<u>ConversationalTask</u> (タスクが会話形式で処理可能かどうか)	MQCCT_NO	0
<u>TaskEndStatus</u> (タスク終了時点の状況)	MQCTES_NOSYNC	0
<u>施設</u> (ブリッジ施設トークン)	MQCFAC_NONE	Null
<u>関数</u> (MQ 呼び出し名または EIBFN 関数 CICS)	MQCFUNC_NONE	ブランク
<u>AbendCode</u> (アベンドコード)	なし	ブランク
<u>認証ツール</u> (パスワードまたはパスチケット)	なし	ブランク
<u>Reserved1</u> (予約済み)	なし	ブランク
<u>ReplyToFormat</u> (MQ 返信メッセージのフォーマット名)	MQFMT_NONE	ブランク
<u>RemoteSysId</u> (使用するリモート CICS システム ID)	なし	ブランク
<u>RemoteTransId</u> (使用する CICS RTRANSID)	なし	ブランク
<u>TransactionId</u> (添付する取引)	なし	ブランク
<u>FacilityLike</u> (端末エミュレーション属性)	なし	ブランク
<u>AttentionId</u> (AID キー)	なし	ブランク
<u>StartCode</u> (トランザクション開始コード)	MQCSC_NONE	ブランク
<u>CancelCode</u> (abend トランザクションコード)	なし	ブランク
<u>NextTransactionId</u> (次に添付する取引)	なし	ブランク
<u>Reserved2</u> (予約済み)	なし	ブランク

表 482. MQCIH における MQCIH のフィールド (続き)		
フィールド名と説明	定数の名前	定数の初期値 (存在する場合)
<u>Reserved3</u> (予約済み)	なし	ブランク
注: <i>Version</i> が MQCIH_VERSION_2 より小さい場合は、残りのフィールドは表示されません。		
<u>CursorPosition</u> (カーソル位置)	なし	0
<u>ErrorOffset</u> (メッセージ内の誤差のオフセット)	なし	0
<u>InputItem</u> (入力項目)	なし	0
<u>Reserved4</u> (予約済み)	なし	0
注: 1. 記号-は、単一のブランク文字を表します。 2. C プログラミング言語では、マクロ変数 MQCIH_DEFAULT には、表にリストされている値が含まれています。このマクロ変数を以下の方法で使用して、構造体のフィールドに初期値を設定します。 <pre>MQCIH MyCIH = {MQCIH_DEFAULT};</pre>		

StrucId (MQCHAR4)

これは情報 CICS ヘッダ構造の構造識別子です。常に入力フィールドです。その値は MQCIH_STRUC_ID です。

値は次のものでなければなりません。

MQCIH_STRUC_ID

CICS 情報ヘッダー構造体の ID。

C プログラミング言語では、定数 MQCIH_STRUC_ID_ARRAY も定義されている。これは MQCIH_STRUC_ID と同じ値を持ちますが、文字列ではなく文字の配列です。

Version (MQLONG)

このフィールドは要求フィールドです。初期値は、MQCIH_VERSION_2 です。

値は次のいずれかでなければなりません。

MQCIH_VERSION_1

バージョン 1 の CICS 情報ヘッダー構造体。

MQCIH_VERSION_2

バージョン 2 の CICS 情報ヘッダー構造体。

これより新しいバージョンの構造体にのみ存在するフィールドは、そのフィールドの説明にその旨記載されています。以下の定数は、現行バージョンのバージョン番号を指定しています。

MQCIH_CURRENT_VERSION

CICS 情報ヘッダー構造体の現行バージョン。

StrucLength (MQLONG)

このフィールドは要求フィールドです。初期値は MQCIH_LENGTH_2 です。

値は次のいずれかでなければなりません。

MQCIH_LENGTH_1

バージョン 1 の CICS 情報ヘッダー構造体の長さ。

MQCIH_LENGTH_2

バージョン 2 の CICS 情報ヘッダー構造体の長さ。

以下の定数は、現行バージョンの長さを指定しています。

MQCIH_CURRENT_LENGTH

現行バージョンの CICS 情報ヘッダー構造体の長さ。

Encoding (MQLONG)

このフィールドは予約フィールドです。その価値は重要ではない。初期値は、ゼロです。

MQCIH 構造体の後のサポートされる構造体のエンコードは、MQCIH 構造体自体のエンコードと同じで、先行の IBM MQ ヘッダーから取られます。

CodedCharSetId (MQLONG)

CodedCharSetId は予約フィールドです。その価値は重要ではない。このフィールドの初期値は 0 です。

MQCIH 構造体の後に続くサポートされる構造体の文字セット ID は、MQCIH 構造体自体の文字セット ID と同じで、先行の IBM MQ ヘッダーから取得されます。

Format (MQCHAR8)

このフィールドは、MQCIH 構造体の後に続くデータの IBM MQ 形式名を示します。

MQPUT または MQPUT1 呼び出しでは、アプリケーションは、このフィールドをデータに適切な値に設定する必要があります。このフィールドのコーディング規則は、MQMD の *Format* フィールドのコーディング規則と同じです。

ReplyToFormat フィールドの値が MQFMT_NONE である場合は、この形式名が応答メッセージでも使用されます。

- DPL 要求の場合、*Format* は COMMAREA の形式名でなければなりません。
- 3270 要求の場合、*Format* は CSQCBDCI でなければならず、応答メッセージの場合、ブリッジは *Format* を CSQCBDCO に設定します。

これらの形式に対するデータ変換出口は、それを実行するキュー・マネージャーにインストールする必要があります。

要求メッセージによってエラー応答メッセージが生成された場合、エラー応答メッセージの形式名は MQFMT_STRING となります。

このフィールドは要求フィールドです。このフィールドの長さは MQ_FORMAT_LENGTH によって指定されます。このフィールドの初期値は MQFMT_NONE です。

フラグ (MQLONG)

このフィールドは要求フィールドです。このフィールドの初期値は、MQCIH_NONE です。

値は次のものでなければなりません。

MQCIH_NONE

フラグなし。

MQCIH_PASS_EXPIRATION

応答メッセージには以下のものが含まれています。

- 要求メッセージと同じ満了レポート・オプション。
- 要求メッセージからの残りの満了時間 (ブリッジの処理時間は未調整)。

この値を省略した場合、有効期限は無制限 に設定されます。

MQCIH_REPLY_WITHOUT_NULLS

CICS DPL プログラム要求の応答メッセージ長が調整され、DPL プログラムによって戻された COMMAREA の末尾の後書きヌル (X'00') が取り除かれます。この値が設定されていないと、ヌルが意味を持つことがあり、この場合 COMMAREA 全体が戻されます。

MQCIH_SYNC_ON_RETURN

DPL 要求のための CICS リンクでは SYNCONRETURN オプションが使用されます。これによって、CICS は、要求を他の CICS 領域に送る場合に、プログラムの完了時に同期点を取るようになります。ブリッジはどの CICS 領域に要求を送るかを指定しません。この制御は、CICS プログラム定義またはワークロード・バランシング機能によって行われます。

ReturnCode (MQLONG)

このフィールドの値は、CICS bridge で実行された処理の結果を示すブリッジからの戻りコードです。このフィールドは応答フィールドで、初期値は MQCRC_OK です。

Function、*CompCode*、*Reason*、および *AbendCode* の各フィールドに、追加情報が格納されることがあります (246 ページの表 480 を参照)。値は、次のいずれか 1 つです。

MQCRC_APPLICATION_ABEND

(5, X'005') アプリケーションが異常終了した。

MQCRC_BRIDGE_ABEND

(4, X'004') CICS bridge が異常終了した。

MQCRC_BRIDGE_ERROR

(3, X'003') CICS bridge でエラーが検出された。

MQCRC_BRIDGE_TIMEOUT

(8, X'008') 指定された時間内に現行作業単位内の 2 番目以降のメッセージを受信しなかった。

MQCRC_CICS_EXEC_ERROR

(1, X'001') EXEC CICS 文でエラーが検出された。

MQCRC_MQ_API_ERROR

(2, X'002') MQ 呼び出しでエラーが検出された。

MQCRC_OK

(0, X'000') エラーなし。

MQCRC_PROGRAM_NOT_AVAILABLE

(7, X'007') プログラムが使用できない。

MQCRC_SECURITY_ERROR

(6, X'006') セキュリティー・エラーが発生した。

MQCRC_TRANSID_NOT_AVAILABLE

(9, X'009') トランザクションが使用できない。

CompCode (MQLONG)

このフィールドは応答フィールドです。初期値は、MQCC_OK です。

このフィールドで返される値は、に依存します *ReturnCode*。246 ページの表 480 を参照してください。

理由 (MQLONG)

このフィールドは応答フィールドです。初期値は MQRC_NONE です

このフィールドで返される値は、に依存します *ReturnCode*。246 ページの表 480 を参照してください。

UOWControl (MQLONG)

このフィールドは要求フィールドです。CICS bridge によって実行される作業単位の処理を制御します。このフィールドの初期値は MQCUOWC_ONLY です

ブリッジに対して、単一ランザクションの実行を要求することも、1つの作業単位内で1つ以上のプログラムの実行を要求することもできます。このフィールドでは、CICS bridge で別の作業単位を開始するか、要求された機能を現行の作業単位の中で実行するか、それとも、作業単位をコミットまたはバックアウトして終了させるかを指定します。データ伝送の流れを最適化するために、様々な組み合わせがサポートされます。

値は次のいずれかでなければなりません。

MQCUOWC_ONLY

作業単位を開始し、機能を実行した上で、その作業単位をコミットする。

MQCUOWC_CONTINUE

現行の作業単位の追加データ (3270 のみ)。

MQCUOWC_FIRST

作業単位を開始し、機能を実行する。

MQCUOWC_MIDDLE

現行の作業単位の中で機能を実行する。

MQCUOWC_LAST

機能を実行した上で、その作業単位をコミットする。

MQCUOWC_COMMIT

作業単位をコミットする (DPL のみ)。

MQCUOWC_BACKOUT

作業単位をバックアウトする (DPL のみ)。

GetWaitInterval (MQLONG)

このフィールドは要求フィールドです。初期値は MQCGWI_DEFAULT です

このフィールドは、*UOWControl* の値が MQCUOWC_FIRST である場合にだけ適用されます。これによって、送信側アプリケーションでは、ブリッジで発行された MQGET 呼び出しが、このメッセージによって開始された作業単位に関する 2 番目およびそれ以降の要求メッセージを待機するおおよその時間をミリ秒単位で指定できます。ブリッジで使用されているデフォルトの待機間隔は、この機能によって指定変更されます。次の特殊値を使用することができます。

MQCGWI_DEFAULT

デフォルト待機間隔。

この値を使用すると、CICS bridge の開始時に指定された時間だけブリッジが待機します。

MQWI_UNLIMITED

無制限の待機間隔。

LinkType (MQLONG)

このフィールドは要求フィールドです。初期値は MQCLT_PROGRAM です

この値は、ブリッジがリンクを試みるオブジェクトのタイプを指定します。値は、次のいずれかでなければなりません。

MQCLT_PROGRAM

DPL プログラム。

MQCLT_TRANSACTION

3270 トランザクション。

OutputDataLength (MQLONG)

このフィールドは、DPL プログラムにのみ使用される要求フィールドです。その初期値は MQCODL_AS_INPUT です

この値は、応答メッセージでクライアントに戻されるユーザー・データの長さです。この長さには、8 バイトのプログラム名も含まれます。リンクされたプログラムに渡される COMMAREA の長さは、このフィ

ールドの長さ、要求メッセージ内のユーザー・データの長さから 8 を引いた長さのどちらか大きい方です。

注: メッセージ内のユーザー・データの長さは、MQCIH 構造体を除いたメッセージの長さです。

リクエストメッセージ内のユーザーデータの長さが未満の場合 *OutputDataLength*、コマンド LINK のオプション *DATALength* が使用され、を別の CICS リージョンへ効率的に機能転送できるよう LINK になります。

次の特殊値を使用することができます。

MQCODL_AS_INPUT

出力長を入力長と同じにする。

リンクされたプログラムに渡す COMMAREA が必ず十分なサイズになるように、応答の要求がなくてもこの値が必要になることがあります。

FacilityKeepTime (MQLONG)

FacilityKeepTime は、ユーザー・トランザクションが終了した後、ブリッジ機能が保持される長さ (秒数) です。

疑似会話型トランザクションの場合は、予想される疑似会話時間に相当する値を指定します。疑似会話の最後のトランザクションの場合は、ゼロを指定します。その他のタイプのトランザクションの場合も、ゼロを指定します。

このフィールドは、3270 トランザクションにのみ使用される要求フィールドです。このフィールドの初期値は 0 です。

ADSDescriptor (MQLONG)

このフィールドは、SEND および RECEIVE BMS 要求で ADS 記述子を送信するかどうかを指定する標識です。

以下の値が定義されます。

MQCADSD_NONE

ADS 記述子の送受信を行わない。

MQCADSD_SEND

ADS 記述子の送信。

MQCADSD_RECV

ADS 記述子の受信。

MQCADSD_MSGFORMAT

ADS 記述子用メッセージ・フォーマットの使用。

この値を指定すると ADS 記述子の送受信の際に、ADS 記述子の長形式が使用されます。長形式では、各フィールドは 4 バイトの境界で位置合わせされます。

ADSDescriptor フィールドを次のように設定します。

- ADS 記述子を使用していない場合は、このフィールドを MQCADSD_NONE に設定します。
- ADS 記述子を使用している場合で、各環境における CCSID が同一の場合は、このフィールドを MQCADSD_SEND と MQCADSD_RECV の合計に設定します。
- ADS 記述子を使用している場合で、各環境における CCSID が異なっている場合は、このフィールドを MQCADSD_SEND、MQCADSD_RECV、MQCADSD_MSGFORMAT の合計に設定します。

これは、3270 トランザクションにのみ使用される要求フィールドです。このフィールドの初期値は、MQCADSD_NONE です。

ConversationalTask (MQLONG)

このフィールドは、タスクを許可して詳細についての要求を発行するか、またはタスクを停止して異常終了メッセージを発行するかを指定する標識です。

値は次のいずれかのオプションです。

MQCCT_YES

タスクは会話型です。

MQCCT_NO

タスクは非会話型です。

このフィールドは、3270 トランザクションにのみ使用される要求フィールドです。このフィールドの初期値は MQCCT_NO です。

TaskEndStatus (MQLONG)

このフィールドは応答フィールドです。タスク終了時のユーザー・トランザクションの状況を示します。このフィールドは 3270 トランザクション専用で使用され、初期値は MQCTES_NOSYNC である。

次のいずれかの値が戻されます。

MQCTES_NOSYNC

同期していない。

ユーザー・トランザクションはまだ完了しておらず、同期点に達していません。この場合、MQMD 内の *MsgType* フィールドは MQMT_REQUEST になります。

MQCTES_COMMIT

作業単位をコミットする。

ユーザー・トランザクションはまだ完了していませんが、最初の作業単位の同期点に達しています。この場合、MQMD 内の *MsgType* フィールドは MQMT_DATAGRAM になります。

MQCTES_BACKOUT

作業単位をバックアウトする。

ユーザー・トランザクションはまだ完了していません。現行の作業単位がバックアウトされます。この場合、MQMD 内の *MsgType* フィールドは MQMT_DATAGRAM になります。

MQCTES_ENDTASK

タスクを終了する。

ユーザー・トランザクションは終了(または異常終了)しました。この場合、MQMD 内の *MsgType* フィールドは MQMT_REPLY になります。

Facility (MQBYTE8)

このフィールドは、8 バイトのブリッジ機能トークンを示します。

ブリッジ機能トークンにより、疑似会話内の複数のトランザクションの同じブリッジ機能 (仮想 3270 端末) が使用可能になります。疑似会話の最初のメッセージ (メッセージが 1 つだけの場合はそのメッセージ) では、値 MQCFAC_NONE に設定します。この値によって、このメッセージに新しいブリッジ機能を割り振るように CICS に指示します。入力メッセージでゼロ以外の *FacilityKeepTime* が指定されていると、応答メッセージでブリッジ機能トークンが戻されます。その後の疑似会話内の入力メッセージでは、同じブリッジ機能トークンを使用しなければなりません。

以下のような特殊値が定義されます。

MQCFAC_NONE

機能トークンは指定されていない。

C 言語の場合、定数 MQCFAC_NONE_ARRAY も定義されます。これは、MQCFAC_NONE と同じ値ですが、ストリングではなく文字の配列です。

このフィールドは、3270 トランザクションにのみ使用される要求フィールドおよび応答フィールドです。このフィールドの長さは、MQ_FACILITY_LENGTH で指定します。このフィールドの初期値は、MQCFAC_NONE です。

Function (MQCHAR4)

このフィールドは応答フィールドです。このフィールドの長さは MQ_FUNCTION_LENGTH によって指定されます。このフィールドの初期値は MQCFUNC_NONE です

このフィールドに戻される値は、*ReturnCode* の値によって決まります。[246 ページの表 480](#) を参照してください。*Function* に IBM MQ 呼び出し名が格納されている場合は、次のような値が戻されます。

MQCFUNC_MQCONN

MQCONN 呼び出し。

MQCFUNC_MQGET

MQGET 呼び出し。

MQCFUNC_MQINQ

MQINQ 呼び出し。

MQCFUNC_MQOPEN

MQOPEN 呼び出し。

MQCFUNC_MQPUT

MQPUT 呼び出し。

MQCFUNC_MQPUT1

MQPUT1 呼び出し。

MQCFUNC_NONE

呼び出しなし。

C プログラミング言語では、上記のすべてについて、定数 MQCFUNC_*_ARRAY も定義されます。この定数の値は、対応する MQCFUNC_* 定数と同じですが、ストリングの代わりに文字の配列を使用します。

AbendCode (MQCHAR4)

AbendCode は応答フィールドです。このフィールドの長さは MQ_ABEND_CODE_LENGTH によって指定されます。このフィールドの初期値は 4 個の空白文字です。

このフィールドで返される値は、`ReturnCode field` の値が `MQCRC\_APPLICATION\_ABEND` または `MQCRC\_BRIDGE\_ABEND` の場合にのみ意味を持ちます。この場合、*AbendCode* には CICS ABCODE 値が入ります。

Authenticator (MQCHAR8)

このフィールドの値は、パスワードまたはパスチケットです。

ユーザー ID の認証が CICS bridge でアクティブな場合は、メッセージの送信側を認証するために、*Authenticator* を使用して MQMD の ID コンテキスト内のユーザー ID を指定します。

これは要求フィールドです。このフィールドの長さは MQ_AUTHENTICATOR_LENGTH によって指定されます。このフィールドの初期値は 8 空白です。

Reserved1 (MQCHAR8)

このフィールドは予約フィールドです。値は 8 個の空白でなければなりません。

ReplyToFormat (MQCHAR8)

このフィールドの値は、現行メッセージに回答して送信される応答メッセージの IBM MQ 形式名です。

このフィールドのコーディング規則は、MQMD の *Format* フィールドのコーディング規則と同じです。

このフィールドは、DPL プログラムにのみ使用される要求フィールドです。このフィールドの長さは MQ_FORMAT_LENGTH によって指定されます。このフィールドの初期値は MQFMT_NONE です。

RemoteSysId (MQCHAR4)

このフィールドは、要求を処理する CICS システムの CICS システム ID を示します。

このフィールドがブランクの場合、CICS システム要求は、ブリッジ・モニターと同じ CICS システムで処理されます。使用される SYSID は応答メッセージの中で戻されます。

3270 疑似会話では、会話内の後続のすべてのメッセージが、初期応答で戻されるリモート SYSID を指定する必要があります。SYSID を指定する場合、次の状態になければなりません。

- アクティブである。
- IBM MQ 要求キューへのアクセス権限を持っている。
- ブリッジ・モニターの CICS システムの CICS ISC リンクからアクセスできる

RemoteTransId (MQCHAR4)

このフィールドは、オプションの要求フィールドです。このフィールドの長さは MQ_TRANSACTION_ID_LENGTH で指定されます

指定された場合、このフィールドは CICS START の RTRANSID 値として使用されます。

TransactionId (MQCHAR4)

このフィールドは要求フィールドです。この長さは、MQ_TRANSACTION_ID_LENGTH によって設定されます。このフィールドの初期値は 4 ブランクです。

が MQCLT_TRANSACTION の値を持つ *LinkType* 場合、*TransactionId* は実行対象のユーザートランザクションのトランザクション識別子である。この場合は空白以外の値を指定してください。

LinkType が値 MQCLT_PROGRAM を持つ場合、*TransactionId* は、該当する作業単位内のすべてのプログラムの実行で使用するトランザクション・コードです。値としてブランクを指定した場合は、CICS DPL ブリッジのデフォルト・トランザクション・コード (CKBP) が使用されます。非ブランクの値を指定する場合は、初期プログラムが CSQCBP00 のローカル・トランザクションとして CICS に定義してある値を使用する必要があります。このフィールドは、*UOWControl* の値が MQCUOWC_FIRST または MQCUOWC_ONLY である場合にだけ適用されます。

FacilityLike (MQCHAR4)

FacilityLike は、ブリッジ機能のモデルとして使用する設置済み端末の名前です。

空白の値は、ブリッジトランザクションプロファイル定義から取得されるか、デフォルト *FacilityLike* 値が使用されることを意味します。

このフィールドは、3270 トランザクションにのみ使用される要求フィールドです。このフィールドの長さは MQ_FACILITY_LIKE_LENGTH によって指定されます。このフィールドの初期値は 4 ブランクです。

AttentionId (MQCHAR4)

このフィールドの値によって、トランザクション開始時の AID キーの初期値が決まります。これは 1 バイト値で、左寄せされます。

AttentionId は、3270 トランザクションにのみ使用される要求フィールドです。このフィールドの長さは MQ_ATTENTION_ID_LENGTH によって指定されます。このフィールドの初期値は 4 ブランクです。

StartCode (MQCHAR4)

このフィールドの値は、端末トランザクション、または START によって開始されたトランザクションをブリッジがエミュレートするかどうかを示す標識です。

値は次のいずれかでなければなりません。

MQCSC_START

開始。

MQCSC_STARTDATA

データを開始する。

MQCSC_TERMINPUT

端末の入力。

MQCSC_NONE

なし。

C プログラミング言語では、上記のすべてについて、定数 `MQCSC_*_ARRAY` も定義されます。この定数の値は、対応する `MQCSC_*` 定数と同じですが、string の代わりに文字の配列を使用します。

ブリッジからの応答において、このフィールドは、フィールド *NextTransactionId* に含まれる次のトランザクション ID に対応する開始コードに設定される。応答では、次の開始コードが使用されます。

- `MQCSC_START`
- `MQCSC_STARTDATA`
- `MQCSC_TERMINPUT`

CICS Transaction Server 1.2 の場合、このフィールドは要求フィールドのみです。応答での値は未定義です。

CICS Transaction Server 1.3 以降のリリースでは、このフィールドは要求フィールドでもあり、応答フィールドでもあります。

このフィールドは、3270 トランザクションにのみ使用されます。このフィールドの長さは `MQ_START_CODE_LENGTH` によって指定されます。このフィールドの初期値は、`MQCSC_NONE` です。

CancelCode (MQCHAR4)

このフィールドの値は、トランザクション (通常は、さらにデータを要求する会話型トランザクション) を終了するために使用される異常終了コードです。それ以外の場合は、このフィールドはブランクに設定されます。

このフィールドは、3270 トランザクションにのみ使用される要求フィールドです。このフィールドの長さは `MQ_CANCEL_CODE_LENGTH` によって指定されます。このフィールドの初期値は 4 ブランクです。

NextTransactionId (MQCHAR4)

この値は、ユーザー・トランザクション (通常は `EXEC CICS RETURN TRANSID`) が戻す次のトランザクションの名前です。次のトランザクションがない場合は、このフィールドはブランクに設定されます。

このフィールドは、3270 トランザクションにのみ使用される応答フィールドです。このフィールドの長さは、`MQ_TRANSACTION_ID_LENGTH` で指定します。このフィールドの初期値は 4 ブランクです。

Reserved2 (MQCHAR8)

このフィールドは予約フィールドです。値は 8 個のブランクでなければなりません。

Reserved3 (MQCHAR8)

このフィールドは予約フィールドです。値は 8 個のブランクでなければなりません。

CursorPosition (MQLONG)

このフィールドの値によって、トランザクション開始時の初期カーソル位置が示されます。会話型トランザクションの場合、カーソル位置は `RECEIVE` ベクトルに入っています。

このフィールドは、3270 トランザクションにのみ使用される要求フィールドです。このフィールドの初期値は 0 です。このフィールドは、MQCIH_VERSION_2Version 未満の場合に存在しません

ErrorOffset (MQLONG)

ErrorOffset フィールドは、ブリッジ出口で検出された無効なデータの位置を示します。このフィールドは、メッセージの先頭から無効なデータの位置までのオフセットを示します。

ErrorOffset は、3270 トランザクションにのみ使用される応答フィールドです。このフィールドの初期値は 0 です。このフィールドは、MQCIH_VERSION_2Version 未満の場合に存在しません

InputItem (MQLONG)

このフィールドは予約フィールドです。値はゼロでなければなりません。
このフィールドは、MQCIH_VERSION_2Version 未満の場合に存在しません

Reserved4 (MQLONG)

このフィールドは予約フィールドです。値はゼロでなければなりません。
このフィールドは、MQCIH_VERSION_2Version 未満の場合に存在しません

MQCMHO - メッセージ・ハンドル作成オプション

MQCMHO 構造体を使用すると、アプリケーションで、メッセージ・ハンドルを作成する方法を制御するオプションを指定できます。この構造は、MQCRTMH 呼び出しの入力パラメーターです。

可用性

この MQCMHO 構造は、以下のプラットフォームで利用可能です：

-  AIX
-  IBM i
-  Linux
-  Windows
-  z/OS

および、これらのシステムに接続された IBM MQ MQI clients。

文字セットとエンコード

入力データは、アプリケーションの文字セットおよびアプリケーション MQCMHO のエンコーディング (MQENC_NATIVE) でなければならない。

フィールド

注：次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 483. MQCMHO のフィールド		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
StrucId (構造識別子)	MQCMHO_STRUC_ID	'CMHO'
バージョン (構造バージョン番号)	MQCMHO_VERSION_1	1

表 483. MQCMHO のフィールド (続き)		
フィールド名と説明	定数の名前	定数の初期値 (存在する場合)
<u>オプション</u> (オプション)	MQCMHO_DEFAULT_VAL IDATION	0
注: 1. C プログラミング言語では、マクロ変数 MQCMHO_DEFAULT には、表にリストされている値が含まれています。この変数を以下の方法で使用すると、構造体のフィールドに初期値を設定できます。 <pre>MQCMHO MyCMHO = {MQCMHO_DEFAULT};</pre>		

言語ごとの宣言

MQCMHO の C 宣言

```
struct tagMQCMHO {
    MQCHAR4    StrucId;          /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQLONG     Options;          /* Options that control the action of MQCRTMH */
};
```

MQCMHO の COBOL 宣言

```
**      MQCMHO structure
10 MQCMHO.
**      Structure identifier
15 MQCMHO-STRUCID          PIC X(4).
**      Structure version number
15 MQCMHO-VERSION          PIC S9(9) BINARY.
**      Options that control the action of MQCRTMH
15 MQCMHO-OPTIONS          PIC S9(9) BINARY.
```

MQCMHO の PL/I 宣言

```
dcl
1 MQCMHO based,
3 StrucId          char(4),          /* Structure identifier */
3 Version          fixed bin(31), /* Structure version number */
3 Options          fixed bin(31), /* Options that control the action of MQCRTMH */
```

MQCMHO の高水準アセンブラ宣言

```
MQCMHO          DSECT
MQCMHO_STRUCID  DS    CL4    Structure identifier
MQCMHO_VERSION  DS    F      Structure version number
MQCMHO_OPTIONS  DS    F      Options that control the action of
*                                MQCRTMH
MQCMHO_LENGTH   EQU    *-MQCMHO
MQCMHO_AREA     DS    CL(MQCMHO_LENGTH)
```

MQCMHO のフィールド詳細

メッセージ **MQCMHO** ハンドル作成オプション構造体により、アプリケーションはメッセージハンドルの作成方法を制御するオプションを指定できます。この構造は、**MQCRTMH** 呼び出しの入力パラメーターです。

フィールド

注: 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 484. MQCMHO のフィールド

フィールド名と説明	定数の名前	定数の初期値（存在する場合）
StrucId (構造識別子)	MQCMHO_STRUC_ID	'CMHO'
バージョン (構造バージョン番号)	MQCMHO_VERSION_1	1
オプション (オプション)	MQCMHO_DEFAULT_VALIDATION	0
注： 1. C プログラミング言語では、マクロ変数 MQCMHO_DEFAULT には、表にリストされている値が含まれています。この変数を以下の方法で使用すると、構造体のフィールドに初期値を設定できます。 <pre>MQCMHO MyCMHO = {MQCMHO_DEFAULT};</pre>		

StrucId (MQCHAR4)

これは、メッセージハンドル作成オプション構造体の構造体識別子です。常に入力フィールドです。その値は MQCMHO_STRUC_ID です。

値は次のものでなければなりません。

MQCMHO_STRUC_ID

メッセージ・ハンドル作成オプション構造の ID。

C プログラミング言語の場合、定数 **MQCMHO_STRUC_ID_ARRAY** も定義されます。これはと同じ値 **MQCMHO_STRUC_ID** を持ちますが、文字列ではなく文字の配列です。

Version (MQLONG)

このフィールドは常に入力フィールドです。初期値は、MQCMHO_VERSION_1 です。

これは構造体のバージョン番号です。値は以下のものでなければなりません。

MQCMHO_VERSION_1

バージョン 1 のメッセージ・ハンドル作成オプション構造。

以下の定数は、現行バージョンのバージョン番号を指定しています。

MQCMHO_CURRENT_VERSION

メッセージ・ハンドル作成オプション構造の現行バージョン。

Options (MQLONG)

このフィールドは常に入力フィールドです。初期値は、MQCMHO_DEFAULT_VALIDATION です。

以下のいずれかのオプションを指定できます。

MQCMHO_VALIDATE

MQSETMP を呼び出してこのメッセージ・ハンドル中のプロパティを設定する際には、プロパティ名が妥当性検査されて、以下のことが確認されます。

- 無効文字が含まれていない。
- 開始 JMS しない または usr。JMS ただし、以下のものを除く：
 - JMSCorrelationID
 - JMSReplyTo
 - JMSType

- JMSXGroupID
- JMSXGroupSeq

これらの名前は JMS プロパティー用に予約されています。

- 以下のいずれかのキーワードではない (小文字と大文字のすべての組み合わせを含む)。

- AND
- BETWEEN
- ESCAPE
- FALSE
- IN
- IS
- LIKE
- NOT
- NULL
- または
- TRUE

- 本文は開始されません。またはルート。（Root.MQMD を除く。）

プロパティが MQ で定義されている場合 (mq.*) 名前が認識されると、プロパティ記述子フィールドはプロパティの正しい値に設定される。プロパティが認識されていない場合は、プロパティ記述子の *Support* フィールドは **MQPD_OPTIONAL** に設定されます。

MQCMHO_DEFAULT_VALIDATION

この値は、プロパティ名のデフォルト・レベルの妥当性検査が行われることを指定します。

デフォルト・レベルの妥当性検査とは、**MQCMHO_VALIDATE** に指定されているレベルに相当します。

この値がデフォルト値です。

MQCMHO_NO_VALIDATION

プロパティ名に対する妥当性検査は行われません。**MQCMHO_VALIDATE** の説明を参照してください。

デフォルト・オプション: 上記のオプションがどれも必要でない場合には、以下のオプションを使用できます。

MQCMHO_NONE

すべてのオプションでデフォルト値が想定されます。この値を使用して、他のオプションが指定されていないことを示します。**MQCMHO_NONE** は、プログラムの文書化を支援します。このオプションは、他のオプションと組み合わせて使用するオプションではありません。ただし、このオプションの値はゼロなので、他のオプションと組み合わせて使用されていても、そのことを検出することはできません。

MQCNO - 接続オプション

MQCNO 構造により、アプリケーションはキューマネージャへの接続に関連するオプションを指定できます。この構造体は、MQCONN 呼び出しの入出力パラメーターです。

共用ハンドルの使用方法および MQCONN 呼び出しの詳細については、[MQCONN による共用 \(スレッド独立\) 接続](#)を参照してください。

可用性

MQCNO 構造は、以下のプラットフォームで利用可能です：

-  AIX

-  IBM i
-  Linux
-  Windows

および、これらのシステムに接続された IBM MQ MQI clients。

バージョン

サポート対象のプログラミング言語向けに提供されるヘッダー、COPY、および INCLUDE ファイルには、最新のバージョンの MQCNO が含まれていますが、フィールド *Version* の初期値は MQCNO_VERSION_1 に設定されています。version-1 構造体に存在しないフィールドを使用するには、アプリケーションは必要なバージョン番号を *Version* フィールドに設定する必要があります。

文字セットとエンコード

MQCNO 内のデータは、キューマネージャ **CodedCharSetId** 属性で指定された文字セットおよびローカルキューマネージャのエンコーディング (MQENC_NATIVE で指定) でなければならない。ただし、アプリケーションを IBM MQ MQI client として実行する場合は、構造体はクライアントの文字セットとエンコードに従っている必要があります。

フィールド

注: 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 485. MQCNO における MQCNO のフィールド		
フィールド名と説明	定数の名前	定数の初期値 (存在する場合)
StrucId (構造識別子)	MQCNO_STRUC_ID	'CNO~'
バージョン (構造バージョン番号)	MQCNO_VERSION_1	1
オプション (MQCONN の動作を制御するオプション)	MQCNO_NONE	0
注: <i>Version</i> が MQCNO_VERSION_2 より小さい場合は、残りのフィールドは無視されます。		
ClientConnOffset (クライアント接続用 MQCD 構造体のオフセット)	なし	0
ClientConnPtr (クライアント接続用 MQCD 構造体のアドレス)	なし	ヌル・ポインターまたはヌル・バイト
注: <i>Version</i> が MQCNO_VERSION_3 より小さい場合、残りのフィールドは無視されます。		
ConnTag (キューマネージャ接続タグ)	MQCT_NONE	Null
注: <i>Version</i> が MQCNO_VERSION_4 より小さい場合、残りのフィールドは無視されます。		
SSLConfigPtr (クライアント接続用の MQSCO 構造体のアドレス)	なし	ヌル・ポインターまたはヌル・バイト
SSLConfigOffset (クライアント接続用 MQSCO 構造体のオフセット)	なし	0
注: <i>Version</i> が MQCNO_VERSION_5 より小さい場合、残りのフィールドは無視されます。		
ConnectionId (一意の接続 ID)	なし	ヌル・ポインターまたはヌル・バイト

表 485. MQCNO における MQCNO のフィールド (続き)

フィールド名と説明	定数の名前	定数の初期値 (存在する場合)
<u>SecurityParmsOffset</u> (セキュリティパラメータ用 MQCSP 構造体のオフセット)	なし	ヌル・ポインターまたはヌル・バイト
<u>SecurityParmsPtr</u> (セキュリティパラメータ用の MQCSP 構造体のアドレス)	なし	ヌル・ポインターまたはヌル・バイト
注: <i>Version</i> が MQCNO_VERSION_6 より小さい場合は、残りのフィールドは無視されます。		
<u>予約済み</u> (予約済みフィールド)	なし	構造体を 64 ビット境界外に埋め込む予約フィールド。
<u>CCDTUrlLength</u> (CCDT URL 長さ)	なし	<u>CCDTUrlPtr</u> または <u>CCDTUrlOffset</u> によって示されるストリングの長さ。
<u>CCDTUrlPtr</u> (CCDT URL ポインタ)	なし	接続に使用するクライアント接続チャンネル・テーブルの場所を示す URL が含まれるストリングへのポインター。
<u>CCDTUrlOffset</u> (CCDT URL オフセット)	なし	接続に使用するクライアント接続チャンネル・テーブルの場所を示す URL が含まれるストリングからのオフセット (バイト単位)。
注: <i>Version</i> が MQCNO_VERSION_7 より小さい場合、残りのフィールドは無視されます。		
<u>ApplName</u> (アプリケーションによって設定された名前)	なし	キュー・マネージャーへの接続を識別するために、アプリケーションによって設定された名前。
<u>Reserved2</u> (予約フィールド)	なし	構造体を 64 ビット境界外に埋め込む予約フィールド。
注: <i>Version</i> が MQCNO_VERSION_8 より小さい場合は、残りのフィールドは無視されます。		
<u>BalanceParmsOffset</u>	なし	MQBNO 構造体へのオフセット (バイト単位)。
<u>BalanceParmsPtr</u>	なし	MQBNO 構造体の場所へのポインター。
注: 1. 記号-は、単一のブランク文字を表します。 2. C プログラミング言語では、マクロ変数 MQCNO_DEFAULT には、表にリストされている値が含まれています。このマクロ変数を以下の方法で使用して、構造体のフィールドに初期値を設定します。		
<pre>MQCNO MyCNO = {MQCNO_DEFAULT};</pre>		

言語ごとの宣言

MQCNO の C 宣言

```
typedef struct tagMQCNO MQCNO;
struct tagMQCNO {
    MQCHAR4    StructId;          /* Structure identifier */
    MQLONG     Version;           /* Structure version number */
    MQLONG     Options;           /* Options that control the action of
    MQCONNX */
    MQLONG     ClientConnOffset;  /* Offset of MQCD structure for client
    connection */
    MQPTR      ClientConnPtr;     /* Address of MQCD structure for client
    connection */
    MQBYTE128  ConnTag;           /* Queue manager connection tag */
    PMQSCO     SSLConfigPtr;      /* Address of MQSCO structure for client
    connection */
    MQLONG     SSLConfigOffset;   /* Offset of MQSCO structure for client
    connection */
    MQBYTE24   ConnectionId;      /* Unique connection identifier */
    MQLONG     SecurityParmsOffset /* Security fields */
    PMQCSP     SecurityParmsPtr  /* Security parameters */
    MQLONG     CCDUrlLength       /* Length of string identified by Ptr or offset */
    MQLONG     CCDUrlOffset       /* Offset in bytes to URL of client connection channel */
    PMQURL     CCDUrlPtr          /* Address of string containing URL */
    MQBYTE4    Reserved          /* Reserved field to pad out to 64 bit boundary */
    MQCHAR28   ApplName           /* Name set by the application to identify the connection to
    the queue manager */
    MQBYTE4    Reserved2         /* Reserved field to pad out to 64 bit boundary */
    MQLONG     BalanceParmsOffset /* Offset of the MQBMO structure */
    PMQBMO     BalanceParmsPtr    /* Address of the location of the MQBMO structure */
};
```

MQCNO の COBOL 宣言

```
**      MQCNO structure
10 MQCNO.
**      Structure identifier
15 MQCNO-STRUCID          PIC X(4).
**      Structure version number
15 MQCNO-VERSION          PIC S9(9) BINARY.
**      Options that control the action of MQCONNX
15 MQCNO-OPTIONS          PIC S9(9) BINARY.
**      Offset of MQCD structure for client connection
15 MQCNO-CLIENTCONNOFFSET PIC S9(9) BINARY.
**      Address of MQCD structure for client connection
15 MQCNO-CLIENTCONNPTR    POINTER.
**      Queue manager connection tag
15 MQCNO-CONNTAG          PIC X(128).
**      Address of MQSCO structure for client connection
15 MQCNO-SSLCONFIGPTR     POINTER.
**      Offset of MQSCO structure for client connection
15 MQCNO-SSLCONFIGOFFSET  PIC S9(9) BINARY.
**      Unique connection identifier
15 MQCNO-CONNECTIONID     PIC X(24).
**      Offset of MQCSP structure for security parameters
15 MQCNO-SECURITYPARMSOFFSET PIC S9(9) BINARY.
**      Address of MQCSP structure for security parameters
15 MQCNO-SECURITYPARMSPTR POINTER.
**      Length of string identified by CCDTURLOFFSET or CCDURLPTR
15 MQCNO-CCDTURLLENGTH
**      Pointer to a string which contains a URL, to identify the location of the client
connection channel
15 MQCNO-CCDTURLPTR
**      Address of string which contains a URL that identifies the location of the client
connection channel table
15 MQCNO-CCDTURLOFFSET
**      Reserved field to pad to 64 bit boundary
15 MQCNO-RESERVED
**      Name set by the application to identify the connection to the queue manager
15 MQCNO-APPLNAME
**      Reserved field to pad to 64 bit boundary
15 MQCNO-RESERVED2
**      Address of the MQBMO structure
15 MQCNO-BALANCEPARMSOFFSET
**      Pointer to the MQBMO structure
15 MQCNO-BALANCEPARMSPTR
```


MQCNO の PL/I 宣言

```

dcl
  1 MQCNO based,
    3 StrucId          char(4),          /* Structure identifier */
    3 Version          fixed bin(31),    /* Structure version number */
    3 Options          fixed bin(31),    /* Options that control the action
                                     of MQCONN */
    3 ClientConnOffset fixed bin(31),    /* Offset of MQCD structure for
                                     client connection */
    3 ClientConnPtr    pointer,          /* Address of MQCD structure for
                                     client connection */
    3 ConnTag          char(128),        /* Queue manager connection tag */
    3 SSLConfigPtr     pointer,          /* Address of MQSCO structure for
                                     client connection */
    3 SSLConfigOffset  fixed bin(31),    /* Offset of MQSCO structure for
                                     client connection */
    3 ConnectionId     char(24),        /* Unique connection identifier */
    3 SecurityParmsOffset fixed bin(31) /* Offset of MQCSP structure for
                                     security parameters */
    3 SecurityParmsPtr pointer,          /* Address of MQCSP structure for
                                     security parameters */
    3 CCDURLLength     fixed bin(31)    /* Length of string identified by CCDURLPtr
                                     or CCDURLOffset */
    3 CCDURLOffset     fixed bin(31)    /* Offset in bytes to URL of client connection channel */
    3 CCDURLPtr        pointer          /* Pointer to string containing URL */
    3 Reserved         char(4)         /* Reserved field to pad out to 64 bit boundary */
    3 ApplName         char(28)        /* Name set by the application to identify the
                                     connection to
                                     the queue manager */
    3 Reserved2        char(4)         /* Reserved field to pad out to 64 bit boundary */
    3 BalanceParmsOffset fixed bin(31) /* Offset of the MQBMO structure */
    3 BalanceParmsPtr  pointer          /* Address of the MQBMO structure */

```

MQCNO の高水準アセンブラ宣言

MQCNO	DSECT	
MQCNO_STRUCID	DS CL4	Structure identifier
MQCNO_VERSION	DS F	Structure version number
MQCNO_OPTIONS	DS F	Options that control the action of MQCONN
* MQCNO_CLIENTCONNOFFSET	DS F	Offset of MQCD structure for client connection
* MQCNO_CLIENTCONNPTR	DS F	Address of MQCD structure for client connection
* MQCNO_CONNTAG	DS XL128	Queue manager connection tag
MQCNO_CONNECTIONID	DS XL24	Unique connection identifier
MQCNO_SSLCONFIGOFFSET	DS F	Offset of MQCSP structure for security parameters
* MQCNO_SSLCONFIGPTR	DS F	Address of MQCSP structure for security parameters
* MQCNO_LENGTH	EQU *-MQCNO	
	ORG MQCNO	
MQCNO_AREA	DS CL(MQCNO_LENGTH)	
MQCNO_CCDURLLENGTH	DS F	Length of string identified by CCDURLPTR or CCDURLOFFSET
* MQCNO_CCDTURLOFFSET	DS F	Offset in bytes to URL of client connection channel
MQCNO_CCDURLPTR	DS F	Pointer to string containing URL
RESERVED	DS XL4	Reserved field to pad out to 64 bit boundary
APPLNAME	DS CL28	Name set by the application to identify the connection to the queue manager
* RESERVED2	DS XL4	Reserved field to pad out to 64 bit boundary
MQCNO_BALANCEPARMSOFFSET	DS F	Offset of the MQBMO structure
MQCNO_BALANCEPARMSPTR	DS F	Address of the MQBMO structure

MQCNO の Visual Basic 宣言

Type	MQCNO	
StrucId	As String*4	'Structure identifier'
Version	As Long	'Structure version number'
Options	As Long	'Options that control the action of 'MQCONNX'
ClientConnOffset	As Long	'Offset of MQCD structure for client 'connection'
ClientConnPtr	As MQPTR	'Address of MQCD structure for client 'connection'
ConnTag	As MQBYTE128	'Queue manager connection tag'

SSLConfigPtr	As MQPTR	'Address of MQSCO structure for client' 'connection'
SSLConfigOffset	As Long	'Offset of MQSCO structure for client' 'connection'
ConnectionId	As MQBYTE24	'Unique connection identifier'
SecurityParmsOffset	As Long	'Offset of MQCSP structure for security' 'parameters'
SecurityParmsPtr	As MQPTR	'Address of MQCSP structure for security' 'parameters'
CCDTUrlLength	As Long	'Length of string identified by CCDTUrlPtr' 'or CCDTUrlOffset'
CCDTUrlOffset	As Long	'Offset in bytes to URL of client connection channel'
CCDTUrlPtr	As MQPTR	'Pointer to string containing URL'
Reserved	As MQBYTE4	'Reserved field to pad out to 64 bit boundary'
ApplName	As String*28	'Name set by the application to identify the connection to' 'the queue manager'
Reserved2	As MQBYTE4	'Reserved field to pad out to 64 bit boundary'
BalanceParmsOffset	As Long	'Offset in bytes to MQBNO structure'
BalanceParmsPtr	As MQPTR	'Address of MQBNO structure'
End Type		

関連タスク

[MQCONN の使用法](#)

MQCNO のフィールド詳細

MQCNO Connect オプション構造体により、アプリケーションはキューマネージャへの接続に関連するオプションを指定できます。この構造体は、MQCONN 呼び出しの入出力パラメーターです。

フィールド

注: 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 486. MQCNO のフィールド		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
StrucId (構造識別子)	MQCNO_STRUC_ID	'CNO~'
バージョン (構造バージョン番号)	MQCNO_VERSION_1	1
オプション (MQCONN の動作を制御するオプション)	MQCNO_NONE	0
注: Version が MQCNO_VERSION_2 より小さい場合は、残りのフィールドは無視されます。		
ClientConnOffset (クライアント接続用 MQCD 構造のオフセット)	なし	0
ClientConnPtr (クライアント接続用 MQCD 構造体のアドレス)	なし	ヌル・ポインターまたはヌル・バイト
注: Version が MQCNO_VERSION_3 より小さい場合、残りのフィールドは無視されます。		
ConnTag (キューマネージャ接続タグ)	MQCT_NONE	Null
注: Version が MQCNO_VERSION_4 より小さい場合、残りのフィールドは無視されます。		
SSLConfigPtr (クライアント接続用の MQSCO 構造体のアドレス)	なし	ヌル・ポインターまたはヌル・バイト
SSLConfigOffset (クライアント接続用 MQSCO 構造体のオフセット)	なし	0
注: Version が MQCNO_VERSION_5 より小さい場合、残りのフィールドは無視されます。		

表 486. MQCNO のフィールド (続き)		
フィールド名と説明	定数の名前	定数の初期値 (存在する場合)
<u>ConnectionId</u> (一意の接続 ID)	なし	ヌル・ポインターまたはヌル・バイト
<u>SecurityParmsOffset</u> (セキュリティパラメータ用 MQSCO 構造体のオフセット)	なし	ヌル・ポインターまたはヌル・バイト
<u>SecurityParmsPtr</u> (セキュリティパラメータ用の MQSCO 構造体のアドレス)	なし	ヌル・ポインターまたはヌル・バイト
注: <i>Version</i> が MQCNO_VERSION_6 より小さい場合は、残りのフィールドは無視されます。		
<u>予約済み</u> (予約済みフィールド)	なし	構造体を 64 ビット境界外に埋め込む予約フィールド。
<u>CCDTUrlLength</u> (CCDT URL 長さ)	なし	<u>CCDTUrlPtr</u> または <u>CCDTUrlOffset</u> によって示される字符串の長さ。
<u>CCDTUrlPtr</u> (CCDT URL ポインタ)	なし	接続に使用するクライアント接続チャンネルテーブルの場所を識別する URL を含む文字列へのポインタ。
<u>CCDTUrlOffset</u> (CCDT URL オフセット)	なし	接続に使用するクライアント接続チャンネルテーブルの位置を識別する URL を含む文字列からのバイト単位のオフセット。
注: <i>Version</i> が MQCNO_VERSION_7 より小さい場合、残りのフィールドは無視されます。		
<u>ApplName</u> (アプリケーションによって設定された名前)	なし	キュー・マネージャーへの接続を識別するために、アプリケーションによって設定された名前。
<u>Reserved2</u> (予約フィールド)	なし	構造体を 64 ビット境界外に埋め込む予約フィールド。
注: <i>Version</i> が MQCNO_VERSION_8 より小さい場合は、残りのフィールドは無視されます。		
<u>BalanceParmsOffset</u>	なし	MQBNO 構造体へのオフセット (バイト単位)。
<u>BalanceParmsPtr</u>	なし	MQBNO 構造体の場所へのポインター。

表 486. MQCNO のフィールド (続き)		
フィールド名と説明	定数の名前	定数の初期値 (存在する場合)
注: 1. 記号-は、単一のブランク文字を表します。 2. C プログラミング言語では、マクロ変数 MQCNO_DEFAULT には、表にリストされている値が含まれています。このマクロ変数を以下の方法で使用して、構造体のフィールドに初期値を設定します。 <pre>MQCNO MyCNO = {MQCNO_DEFAULT};</pre>		

StrucId (MQCHAR4)

これは接続オプション構造体の構造体識別子です。常に入力フィールドです。その値は MQCNO_STRUC_ID です。

値は次のものでなければなりません。

MQCNO_STRUC_ID

接続オプション構造体の識別子。

C プログラミング言語では、定数 MQCNO_STRUC_ID_ARRAY も定義されている。この定数は MQCNO_STRUC_ID と同じ値を持ちますが、文字列ではなく文字の配列です。

Version (MQLONG)

Version は常に入力フィールドです。初期値は、MQCNO_VERSION_1 です。

値は次のいずれかでなければなりません。

MQCNO_VERSION_1

バージョン 1 の接続オプション構造体。

MQCNO_VERSION_2

バージョン 2 の接続オプション構造体。

MQCNO_VERSION_3

バージョン 3 の接続オプション構造体。

MQCNO_VERSION_4

バージョン 4 の接続オプション構造体。

MQCNO_VERSION_5

バージョン 5 の接続オプション構造体。

MQCNO_VERSION_6

バージョン 6 の接続オプション構造体。

MQCNO_VERSION_7

バージョン 7 の接続オプション構造体。

MQCNO_VERSION_8

バージョン 8 の接続オプション構造体。

これより新しいバージョンの構造体にのみ存在するフィールドは、そのフィールドの説明にその旨記載されています。以下の定数は、現行バージョンのバージョン番号を指定しています。

MQCNO_CURRENT_VERSION

接続オプション構造体の現行バージョン。

Options (MQLONG)

MQCONNx のアクションを制御するオプション。

会計オプション

以下のオプションは、**AccountingConnOverride** キュー・マネージャー属性を MQMON_ENABLED に設定した場合の、アカウンティングのタイプを制御します。

MQCNO_ACCOUNTING_MQI_ENABLED

キュー・マネージャー定義で **MQIAccounting** 属性を MQMON_OFF に設定することによりモニター・データ収集を使用不可にした場合、このフラグを設定すると、MQI アカウンティング・データ収集が使用可能になります。

MQCNO_ACCOUNTING_MQI_DISABLED

キュー・マネージャー定義で **MQIAccounting** 属性を MQMON_OFF に設定することによりモニター・データ収集を使用不可にした場合、このフラグを設定すると、MQI アカウンティング・データ収集が停止します。

MQCNO_ACCOUNTING_Q_ENABLED

キュー・マネージャー定義で **MQIAccounting** 属性を MQMON_OFF に設定することによりキュー・アカウンティング・データ収集を使用不可にした場合、このフラグを設定すると、キュー定義の **MQIAccounting** フィールドにキュー・マネージャーを指定したキューでは、アカウンティング・データ収集が使用可能になります。

MQCNO_ACCOUNTING_Q_DISABLED

キュー・マネージャー定義で **MQIAccounting** 属性を MQMON_OFF に設定することによりキュー・アカウンティング・データ収集を使用不可にした場合、このフラグを設定すると、キュー定義の **MQIAccounting** フィールドにキュー・マネージャーを指定したキューでは、アカウンティング・データ収集がオフに切り替わります。

これらのフラグが定義されない場合、接続のアカウンティングは、キュー・マネージャーの属性で定義されたとおりとなります。

製本オプション

以下のオプションは、使用する IBM MQ バインディングのタイプを制御します。次のオプションのうち 1 つのみを指定します。

MQCNO_STANDARD_BINDING

アプリケーションとローカル・キュー・マネージャー・エージェント (キューイング操作を管理するコンポーネント) がそれぞれ別の実行単位 (通常は、別のプロセス) で実行されます。この調整により、キュー・マネージャーの整合性は保持されます。つまり、キュー・マネージャーが誤ったプログラムから保護されます。

キューマネージャが複数のバインディングタイプをサポートしている場合、MQCNO_STANDARD_BINDING を設定すると、キューマネージャは `'bindings'` スタンス Connection 内の `'bindings'` ファイル `qm.ini` の `'DefaultBindType attribute'` を使用して、実際のバインディングタイプを選択します。このスタanzas が定義されていない、または値を使用できない、あるいはその値がアプリケーションに対して適切ではない場合、キュー・マネージャーは適切なバインディング・タイプを選択します。キュー・マネージャーは、接続オプションで実際に使用されるバインディング・タイプを設定します。

アプリケーションが十分にテストされていないか、確実性や信頼性に欠けている可能性がある場合は、MQCNO_STANDARD_BINDING を使用します。MQCNO_STANDARD_BINDING はデフォルトです。

このオプションはすべての環境でサポートされます。

mqm ライブラリーにリンクしている場合、まずはデフォルトのバインド・タイプを使用した標準のサーバー接続が試行されます。基礎となるサーバー・ライブラリーのロードに失敗した場合、代わりにクライアント接続が試行されます。

- MQCONN (MQCNO_STANDARD_BINDING の指定時は MQCONNX) の動作を変更するには、MQ_CONNECT_TYPE 環境変数を以下のオプションの 1 つに設定します。この例外として、MQ_CONNECT_TYPE を LOCAL または STANDARD に設定して MQCNO_FASTPATH_BINDING を指定した場合、関連する変更をアプリケーションに加えることなく管理者がファスト・パス接続をダウングレードできることに注意してください。

表 487. MQCONN または MQCONNX の動作を変更する MQ_CONNECT_TYPE の値	
値	意味
CLIENT	クライアント接続のみが試行されます。
FASTPATH	この値は以前のリリースではサポートされていましたが、現在は指定されても無視されます。
ローカル	サーバー接続のみが試行されます。ファスト・パス接続が、通常のサーバー接続にダウングレードされます。
STANDARD	以前のリリースとの互換性のためにサポートされず。現在この値は、LOCAL として処理されます。

- MQCONNX の呼び出し時に MQ_CONNECT_TYPE 環境変数が設定されていない場合、デフォルトのバインド・タイプを使用した標準サーバー接続が試行されます。サーバー・ライブラリーのロードに失敗した場合、クライアント接続が試みられます。

MQCNO_FASTPATH_BINDING

アプリケーションとローカル・キュー・マネージャー・エージェントが同じ実行単位で実行されます。これは、アプリケーションとローカル・キュー・マネージャー・エージェントが別々の実行単位で実行される通常のバインディング方式とは対照的です。

キュー・マネージャーがこのタイプのバインディングをサポートしていない場合、MQCNO_FASTPATH_BINDING は無視されます。この場合の処理は、このオプションが指定されていない場合と同じように実行されます。

複数のプロセスのリソース消費量がアプリケーションで使用される全リソースと比較して多い場合では、MQCNO_FASTPATH_BINDING を指定すると有効な場合があります。ファースト・パス・バインディングを使用するアプリケーションのことを、承認されたアプリケーションと呼びます。

ファースト・パス・バインディングを使用するかどうかを決める際には、必ず以下の重要事項を考慮に入れてください。

- MQCNO_FASTPATH_BINDING オプションを使用する場合、キュー・マネージャーに属するメッセージや他のデータ領域をアプリケーションが変更または破壊することは防止されません。このオプションは、これらの問題を十分に評価した状況においてのみ使用してください。
- アプリケーションで、非同期信号およびタイマー割り込み (sigkill など) を MQCNO_FASTPATH_BINDING と組み合わせて使用しないでください。また、共用メモリー・セグメントを使用する場合にもいくつかの制限があります。
- キュー・マネージャーから切断するには、アプリケーションは MQDISC 呼び出しを使用する必要があります。
- endmqm コマンドでキュー・マネージャーを終了する場合は、その前に必ずアプリケーションを停止してください。
- IBM i** IBM i では、ジョブを QMQMADM グループに属するユーザー・プロファイルの下で実行しなければなりません。さらに、プログラムの異常終了は絶対に避けてください。予期しない結果が発生することがあります。
- Linux** **AIX** AIX and Linux では、mqm ユーザー ID と mqm グループ ID が、それぞれ有効なユーザー ID およびグループ ID である必要があります。アプリケーションをこの方法で実行するには、ユーザー ID mqm およびグループ ID mqm がプログラムを所有するように構成した上で、プログラムに setuid 許可ビットおよび setgid 許可ビットを設定します。
- Windows** プログラムを実行するユーザーは Windows、グループ mqm の一員でなければなりません。32 ビットアプリケーションでは Fastpath バインディングはサポートされていません。

MQCNO_FASTPATH_BINDING オプションは、次の環境でサポートされます。

-  AIX
-  IBM i
-  Linux
-  Windows

•  z/OS z/OS では、このオプションは受け入れられますが、無視されます。

トラステッド・アプリケーションの使用方法の詳細については、[トラステッド・アプリケーションの制約事項](#)を参照してください。

MQCNO_SHARED_BINDING

MQCNO_SHARED_BINDING を指定すると、アプリケーションとローカル・キュー・マネージャー・エージェントは一部のリソースを共有します。キュー・マネージャーがこのタイプのバインディングをサポートしていない場合、MQCNO_SHARED_BINDING は無視されます。処理は、オプションが指定されなかったものとして続行します。

MQCNO_ISOLATED_BINDING

この場合、アプリケーション・プロセスとローカル・キュー・マネージャー・エージェントは、リソースを共有しないという点で相互に分離しています。キュー・マネージャーがこのタイプのバインディングをサポートしていない場合、MQCNO_ISOLATED_BINDING は無視されます。処理は、オプションが指定されなかったものとして続行します。

MQCNO_CLIENT_BINDING

このオプションを指定すると、アプリケーションはクライアント接続のみを試行します。このオプションには以下の制約があります。

-  MQCNO_CLIENT_BINDING は、z/OS 上では無視されます。
- MQCNO_CLIENT_BINDING は、MQCNO_STANDARD_BINDING 以外の MQCNO バインディング・オプションとともに指定されると MQRC_OPTIONS_ERROR を出して拒否されます。
- Java もしくは .NET には、バインド・タイプを選択する独自の仕組みがあるため、MQCNO_CLIENT_BINDING を使用できません。

MQCNO_LOCAL_BINDING

このオプションを指定すると、アプリケーションはサーバー接続を試行します。MQCNO_FASTPATH_BINDING、MQCNO_ISOLATED_BINDING、または MQCNO_SHARED_BINDING のいずれかが同時に指定されていると、接続はそちらのタイプに代わります。それについてはこのセクションで説明されています。そうでない場合は、デフォルトのバインド・タイプを使用した通常のサーバー接続が試行されます。MQCNO_LOCAL_BINDING には以下の制約があります。

-  MQCNO_LOCAL_BINDING は、z/OS 上では無視されます。
- MQCNO_LOCAL_BINDING は、MQCNO_RECONNECT_AS_DEF 以外の MQCNO 再接続オプションとともに指定されると MQRC_OPTIONS_ERROR を出して拒否されます。
- Java もしくは .NET には、バインド・タイプを選択する独自の仕組みがあるため、MQCNO_LOCAL_BINDING を使用できません。

以下のプラットフォームでは、使用するバインディングのタイプを制御するために、Options フィールドで指定したバインド・タイプとともに環境変数 MQ_CONNECT_TYPE を使用できます。

-  AIX
-  Linux
-  Windows

この環境変数を指定する場合、値は **FASTPATH** または **STANDARD** でなければなりません。それ以外の値の場合、この環境変数は無視されます。この環境変数の値には、大/小文字の区別があります。詳しくは、[MQCONN](#) 環境変数を参照してください。

この環境変数と *Options* フィールドは、以下のように相互作用します。

- この環境変数を省略したり、サポートされていない値をこの環境変数に入れたりすると、ファスト・パス・バインディングを使用するかどうかは、Options フィールドでのみ決定できます。
- サポートされている値をこの環境変数に入れると、環境変数と Options フィールドの両方にファスト・パス・バインディングを指定した場合に限りファスト・パス・バインディングが使用されます。

接続タグのオプション

以下のオプションは、接続タグの使用を制御 ConnTag しします。これらは、キュー IBM MQ for z/OS ・マネージャに接続している場合のみサポートされます。が MQCNO_VERSION_3 未満 Version の場合、それらは無効である。これらのオプションのうち、いずれか一つだけを選択してください。マルチプラットフォームのキューマネージャから接続する場合は、MQCNO_GENERATE_CONN_TAG を指定してください。

Multi MQCNO_GENERATE_CONN_TAG

キュー・マネージャがこの接続に関連付けた接続タグを、出力 MQCNO 構造体で返します。

キューマネージャが単一のアプリケーションインスタンスと見なすすべての接続に対して、返される接続タグは同一です。

z/OS MQCNO_SERIALIZE_CONN_TAG_Q_MGR

このオプションは、ローカル・キュー・マネージャ内での接続タグの排他使用を要求します。接続タグがローカル・キュー・マネージャ内で使用中の場合、MQCONN 呼び出しは、理由コード MQRC_CONN_TAG_IN_USE で失敗します。ローカル・キュー・マネージャが属するキュー共有グループ内の別の場所で接続タグが使用されていても、呼び出しの結果には影響しません。

z/OS MQCNO_SERIALIZE_CONN_TAG_QSG

このオプションは、ローカル・キュー・マネージャが属するキュー共有グループ内での接続タグの排他使用を要求します。接続タグがキュー共有グループ内で使用中の場合、MQCONN 呼び出しは、理由コード MQRC_CONN_TAG_IN_USE で失敗します。

z/OS MQCNO_RESTRICT_CONN_TAG_Q_MGR

このオプションは、ローカル・キュー・マネージャ内での接続タグの共有を要求します。接続タグがローカル・キュー・マネージャ内ですでに使用されていたとしても、アプリケーションが、そのタグを使用しているユーザーと同じ処理範囲で実行している限り、MQCONN 呼び出しは正常に行われます。この条件に適合しない場合、MQCONN 呼び出しは、理由コード MQRC_CONN_TAG_IN_USE で失敗します。呼び出しの結果は、ローカル・キュー・マネージャが属するキュー共有グループ内にある接続タグの使用によって影響を受けることはありません。

- 接続タグを共有するには、同じ MVS アドレス・スペース内でアプリケーションを実行する必要があります。接続タグを使用するアプリケーションがクライアント・アプリケーションである場合、MQCNO_RESTRICT_CONN_TAG_Q_MGR は使用できません。

z/OS MQCNO_RESTRICT_CONN_TAG_QSG

このオプションは、ローカル・キュー・マネージャが属するキュー共有グループ内での接続タグの共有を要求します。接続タグがキュー共有グループ内で既に使用されていても、要求アプリケーションがそのタグの既存ユーザーと同じ処理範囲で実行されていて、そのタグの既存ユーザーと同じキュー・マネージャに接続されていれば、MQCONN 呼び出しは正常に行われます。

これらの条件に適合しない場合、MQCONN 呼び出しは、理由コード MQRC_CONN_TAG_IN_USE で失敗します。

- 接続タグを共有するには、同じ MVS アドレス・スペース内でアプリケーションを実行する必要があります。接続タグを使用するアプリケーションがクライアント・アプリケーションである場合、MQCNO_RESTRICT_CONN_TAG_QSG は使用できません。

オプションが指定されていない場合、ConnTag は使用されません。

Multi ハンドル共有オプション

以下のオプションは、次の環境でサポートされます。

-  AIX
-  IBM i
-  Linux
-  Windows

これらは、同じプロセス内の異なるスレッド (並列処理の単位) 間のハンドルの共有を制御します。以下のオプションのうち 1 つのみ指定できます。

MQCNO_HANDLE_SHARE_NONE

このオプションは、接続およびオブジェクト・ハンドルを使用できるのは、ハンドルの割り振りを行ったスレッド (つまり、MQCONN、MQCONNEX、または MQOPEN 呼び出しを発行したスレッド) だけであることを示します。同じプロセスに属する他のスレッドはそのハンドルを使用できません。

MQCNO_HANDLE_SHARE_BLOCK

このオプションは、プロセスの 1 つのスレッドが割り振った接続およびオブジェクト・ハンドルを、同じプロセスに属する他のスレッドが使用できることを示します。ただし、特定のハンドルを使用できるのは一度に 1 つのスレッドだけです。つまり、ハンドルの順次使用だけが許可されています。すでに別のスレッドによって使用されているハンドルを使用しようとすると、そのハンドルが使用可能になるまで呼び出しはブロック (待機) されます。

MQCNO_HANDLE_SHARE_NO_BLOCK

これは MQCNO_HANDLE_SHARE_BLOCK と同じですが、ハンドルが別のスレッドによって使用中である場合、ハンドルが使用可能になるまで呼び出しがブロックするのではなく、MQCC_FAILED および MQRC_CALL_IN_PROGRESS を戻して即座に完了します。

スレッドは、以下のようにゼロまたは 1 個の非共有ハンドルを持つことができます。

- MQCNO_HANDLE_SHARE_NONE を指定する MQCONN または MQCONNEX 呼び出しは、最初の呼び出しで新しい非共有ハンドル、および 2 番目以降の呼び出しで同じ非共有ハンドルを返します (介入する MQDISC 呼び出しがないという前提です)。2 番目以降の呼び出しの理由コードは、MQRC_ALREADY_CONNECTED です。
- MQCNO_HANDLE_SHARE_BLOCK または MQCNO_HANDLE_SHARE_NO_BLOCK を指定する各 MQCONNEX 呼び出しは、呼び出しごとに新しい共有ハンドルを返します。

オブジェクト・ハンドルは、オブジェクト・ハンドルを作成した MQOPEN 呼び出しで指定される接続ハンドルと同じ共有プロパティを継承します。また、作業単位は、作業単位を開始するために使用される接続ハンドルと同じ共有プロパティを継承します。共有ハンドルを使用するあるスレッドで開始された作業単位は、同じハンドルを使用する別のスレッドで更新できます。

ハンドル共有オプションを指定しない場合、環境別に次のようなデフォルトが指定されます。

-  Microsoft Transaction Server (MTS) 環境では、デフォルトは MQCNO_HANDLE_SHARE_BLOCK と同じです。
- その他の環境では、デフォルトは MQCNO_HANDLE_SHARE_NONE と同じです。

再接続オプション

再接続オプションにより、ある接続が再接続可能かどうかが決まります。再接続可能なのはクライアント接続のみです。

MQCNO_RECONNECT_AS_DEF

再接続オプションは、デフォルト値に解決されます。デフォルトが設定されない場合、このオプションの値は DISABLED に解決されます。このオプションの値はサーバーに渡され、PCF および MQSC によって照会できるようになります。

MQCNO_RECONNECT

MQCONN の **QmgrName** パラメーターの値と整合する任意のキュー・マネージャーにアプリケーションを再接続できます。MQCNO_RECONNECT オプションは、クライアント・アプリケーションと、最初にそのアプリケーションとの接続が確立されていたキュー・マネージャーとの間に親和性がない場合のみ使用してください。このオプションの値はサーバーに渡され、PCF および MQSC によって照会できるようになります。

MQCNO_RECONNECT_DISABLED

アプリケーションを再接続できません。このオプションの値はサーバーに渡されません。

MQCNO_RECONNECT_Q_MGR

このアプリケーションは、最初にそれと接続されていたキュー・マネージャーにのみ再接続できます。この値は、あるクライアントが再接続可能であり、そのクライアント・アプリケーションと、最初にそのアプリケーションとの接続が確立されていたキュー・マネージャーとの間に親和性がある場合のみ使用してください。高可用性キュー・マネージャーの待機インスタンスにクライアントを自動再接続する場合にこの値を選択します。このオプションの値はサーバーに渡され、PCF および MQSC によって照会できるようになります。

MQCNO_RECONNECT、MQCNO_RECONNECT_DISABLED、および MQCNO_RECONNECT_Q_MGR オプションは、クライアント接続にのみ使用してください。このオプションをバインディング接続に使用すると、MQCONN が失敗し、完了コード MQCC_FAILED と理由コード MQRC_OPTIONS_ERROR が戻されます。自動クライアント再接続は IBM MQ classes for Java でサポートされていません

会話共有オプション

以下のオプションは、TCP/IP クライアント接続にのみ適用されます。SNA、SPX、および NetBios の各チャネルの場合、これらの値は無視され、チャネルは以前のバージョンの製品のように実行されます。

MQCNO_NO_CONV_SHARING

このオプションは、会話の共有を許可しません。

会話のロードが非常に重く、そのため会話の共有が存在するチャネル・インスタンスのサーバー接続エンドで競合が起きる可能性がある場合には、MQCNO_NO_CONV_SHARING を使用するとよいでしょう。MQCNO_NO_CONV_SHARING は、会話の共有をサポートするチャネルに接続する場合は sharecnv(1) と同様に動作し、会話の共有をサポートしないチャネルに接続する場合は sharecnv(0) と同様に動作します。

MQCNO_ALL_CONVS_SHARE

このオプションは、会話の共有を許可します。アプリケーションは、チャネルのインスタンス上の接続の数の制限を設けません。このオプションがデフォルト値です。

チャネル・インスタンスの共有が可能であるものの、チャネルのサーバー接続エンドの *SharingConversations* (SHARECNV) 定義が 1 に設定されていることをアプリケーションが示す場合、共有はされず、アプリケーションに対して警告は出されません。

同じように、共有可能であることがアプリケーションから示されているが、サーバー接続の *SharingConversations* の定義がゼロに設定されているという場合も、警告は出されません。そしてアプリケーションは、IBM WebSphere® MQ 7.0 より前のバージョンのクライアントと同じ動作を示します。会話の共有に関するアプリケーションの設定は無視されます。

MQCNO_NO_CONV_SHARING と MQCNO_ALL_CONVS_SHARE は相互に排他的です。特定の接続に対して両方のオプションが指定されると、接続は理由コード MQRC_OPTIONS_ERROR によって拒否されます。

チャンネル定義オプション

以下のオプションは、MQCNO で渡されるチャンネル定義構造体の使用を制御します。

MQCNO_CD_FOR_OUTPUT_ONLY

このオプションは、成功した MQCONN 呼び出しで使用されたチャンネル名を戻すという目的でのみ MQCNO 内でのチャンネル定義構造体の使用を許可します。

有効なチャンネル定義構造体がない場合、呼び出しは理由コード MQRC_CD_ERROR で失敗します。

アプリケーションがクライアントとして実行されていない場合、このオプションは無視されます。

戻されたチャンネル名は、その後の MQCNO_USE_CD_SELECTION オプションを使用する MQCONN 呼び出しでも使用できます。その際、同じチャンネル定義を使って再接続することができます。これは、複数の適用可能なチャンネル定義がクライアント・チャンネル・テーブルにある場合に役に立ちます。

MQCNO_USE_CD_SELECTION

このオプションは、MQCNO で渡されたチャンネル定義構造体に含まれるチャンネル名を使用して MQCONN 呼び出しが接続を行うことを許可します。

MQSERVER 環境変数を設定すると、その環境変数で定義されているチャンネル定義が使用されます。MQSERVER が設定されていない場合は、クライアント・チャンネル・テーブルが使用されます。

一致するチャンネル名とキュー・マネージャー名を持つチャンネル定義が見つからない場合、呼び出しは理由コード MQRC_Q_MGR_NAME_ERROR で失敗します。

有効なチャンネル定義構造体がない場合、呼び出しは理由コード MQRC_CD_ERROR で失敗します。

アプリケーションがクライアントとして実行されていない場合、このオプションは無視されます。

デフォルトの選択肢

上記のいずれのオプションも必要ない場合、次のオプションを使用できます。

MQCNO_NONE

オプションが指定されていません。

MQCNO_NONE は、プログラムの文書化を支援するために使用します。このオプションは、他の MQCNO_* オプションと組み合わせて使用するオプションではありません。ただし、このオプションの値はゼロと等価なので、他のオプションと組み合わせて使用しても、エラーとして検出されることはありません。

ClientConnOffset (MQLONG)

ClientConnOffset は、MQCNO 構造体の先頭からの MQCD チャンネル定義構造体のオフセットをバイト数で表したものです。オフセットの値は、正負どちらの値にもなります。このフィールドは入力フィールドです。初期値は 0 です。

ClientConnOffset は、MQCONN 呼び出しを発行するアプリケーションが IBM MQ MQI client として実行されている場合にのみ使用されます。このフィールドの使用の詳細は、ClientConnPtr フィールドの説明を参照してください。

Version が MQCNO_VERSION_2 より前の場合、このフィールドは無視されます。

ClientConnPtr (MQPTR)

ClientConnPtr は入力フィールドです。初期値は、ポインターをサポートするプログラミング言語ではヌル・ポインターです。それ以外の場合は、すべてヌルのバイト・ストリングです。

ClientConnOffset および ClientConnPtr は、MQCONN 呼び出しを発行するアプリケーションが IBM MQ MQI client として実行されている場合にのみ使用されます。アプリケーションは、この 2 つのフィールドのいずれかを指定して、必要な値を持つ MQCD チャンネル定義構造体を提供することにより、クライアント接続チャンネルの定義を制御できます。

アプリケーションが IBM MQ MQI client として実行されているが、MQCD 構造体を提供しない場合は、MQSERVER 環境変数を使用してチャンネル定義が選択されます。MQSERVER が設定されていない場合は、クライアント・チャンネル・テーブルが使用されます。

アプリケーションが IBM MQ MQI client として実行されていない場合は、*ClientConnOffset* および *ClientConnPtr* は無視されます。

アプリケーションが MQCD 構造体を提供する場合、ここで示すフィールドを必要な値に設定します。MQCD 内のその他のフィールドは無視されます。文字ストリングは、フィールドの長さまでブランクで埋め込むことも、ヌル文字で終了することもできます。MQCD 構造体のフィールドに関する詳細は、[1482 ページの『フィールド』](#)を参照してください。

表 488. MQCD のフィールド

MQCD のフィールド	値
<i>ChannelName</i>	チャンネル名。
<i>Version</i>	構造体のバージョン番号。MQCD_VERSION_7 より小さい値は不可。
<i>TransportType</i>	サポートされる任意のトランスポート・タイプ。
<i>ModeName</i>	LU 6.2 モード名。
<i>TpName</i>	LU 6.2 トランザクション・プログラム名。
<i>SecurityExit</i>	チャンネル・セキュリティ出口の名前。
<i>SendExit</i>	チャンネル送信出口の名前。
<i>ReceiveExit</i>	チャンネル受信出口の名前。
<i>MaxMsgLength</i>	クライアント接続チャンネルを通して送信できるメッセージの最大長 (バイト数)。
<i>SecurityUserData</i>	セキュリティ出口のユーザー・データ。
<i>SendUserData</i>	送信出口のユーザー・データ。
<i>ReceiveUserData</i>	受信出口のユーザー・データ。
<i>UserIdentifier</i>	LU 6.2 セッションの確立に使用されるユーザー ID。
<i>Password</i>	LU 6.2 セッションの確立に使用されるパスワード。
<i>ConnectionName</i>	接続名。
<i>HeartbeatInterval</i>	ハートビート・フローの間隔 (秒数)。
<i>StrucLength</i>	MQCD 構造体の長さ。
<i>ExitNameLength</i>	<i>SendExitPtr</i> と <i>ReceiveExitPtr</i> でアドレッシングされた出口名の長さ。または <i>ReceiveExitPtr</i> がヌルポインタ以外の値 <i>SendExitPtr</i> に設定されている場合、ゼロより大きくなければならない。
<i>ExitDataLength</i>	<i>SendUserDataPtr</i> と <i>ReceiveUserDataPtr</i> でアドレッシングされた出口データの長さ。または <i>ReceiveUserDataPtr</i> がヌルポインタ以外の値 <i>SendUserDataPtr</i> に設定されている場合、ゼロより大きくなければならない。
<i>SendExitsDefined</i>	<i>SendExitPtr</i> でアドレッシングされた送信出口の数。ゼロの場合は、 <i>SendExit</i> と <i>SendUserData</i> で出口の名前とデータを提供します。ゼロより大きい場合、 <i>SendExitPtr</i> と <i>SendUserDataPtr</i> が出口の名称とデータを提供し、 <i>SendExit</i> ととが空白 <i>SendUserData</i> でなければなりません。
<i>ReceiveExitsDefined</i>	<i>ReceiveExitPtr</i> でアドレッシングされる受信出口の数。ゼロの場合は、 <i>ReceiveExit</i> と <i>ReceiveUserData</i> で出口の名前とデータを提供します。ゼロより大きい場合、 <i>ReceiveExitPtr</i> と <i>ReceiveUserDataPtr</i> が出口の名称とデータを提供し、 <i>ReceiveExit</i> ととが空白 <i>ReceiveUserData</i> でなければなりません。

表 488. MQCD のフィールド (続き)

MQCD のフィールド	値
<i>SendExitPtr</i>	最初の送信出口の名前のアドレス。
<i>SendUserDataPtr</i>	最初の送信出口のデータのアドレス。
<i>ReceiveExitPtr</i>	最初の受信出口の名前のアドレス。
<i>ReceiveUserDataPtr</i>	最初の受信出口のデータのアドレス。
<i>LongRemoteUserIdLength</i>	長いリモート・ユーザー ID の長さ。
<i>LongRemoteUserIdPtr</i>	長いリモート・ユーザー ID のアドレス。
<i>RemoteSecurityId</i>	リモート・セキュリティ ID。
<i>SSLCipherSpec</i>	TLS CipherSpec.
<i>SSLPeerNamePtr</i>	TLS ピア名のアドレス。
<i>SSLPeerNameLength</i>	TLS ピア名の長さ。
<i>KeepAliveInterval</i>	チャンネルのキープアライブ・タイミングとして通信スタックに渡された値。
<i>LocalAddress</i>	ローカル通信アドレス (使用するローカル・ネットワーク・アダプターの IP アドレスを含む) および発信接続で使用するポートの範囲。

チャンネル定義構造体は、次の 2 つの方法で提供できます。

- *ClientConnOffset* オフセット・フィールドを使用する

この場合、アプリケーションは MQCNO と後に続くチャンネル定義構造体 MQCD を含む、複合構造体を宣言する必要があります。さらに *ClientConnOffset* を、MQCNO の先頭からのそのチャンネル定義構造体のオフセットに設定する必要があります。このオフセットが正しいか、確かめてください。*ClientConnPtr* には、ヌル・ポインターまたはヌル・バイトを設定する必要があります。

ポインターのデータ・タイプをサポートしていないプログラミング言語や、他の環境に移植できない方式のポインター・データ・タイプをインプリメントしているプログラミング言語 (COBOL プログラミング言語など) の場合には、*ClientConnOffset* を使用してください。

Visual Basic プログラミング言語では、以下の複合構造が呼び出されました。MQCNOCD は、ヘッダー・ファイル CMQXB.BAS; に用意されています。この構造体には、MQCNO 構造体とそれに続く MQCD 構造体が含まれています。MQCNOCD_DEFAULTS サブルーチンを呼び出すことによって MQCNOCD を初期化します。MQCNOCD は以下のように使用します MQCONNXAny 呼び出しのバリエーション。これについて詳しくは、MQCONNX 呼び出しの説明を参照してください。

- *ClientConnPtr* ポインター・フィールドを使用する

この場合、アプリケーションは MQCNO 構造体とは別個にチャンネル定義構造体を宣言でき、*ClientConnPtr* をそのチャンネル定義構造体のアドレスに設定できます。*ClientConnOffset* に 0 を設定します。

他の環境へ移植できる形式のポインター・データ・タイプをサポートするプログラミング言語 (例えば、C プログラミング言語など) には、*ClientConnPtr* を使用してください。

C プログラミング言語では、MQCONNX 呼び出し用として、MQCD_DEFAULT で提供される初期値よりも適した構造体の初期値を提供するために、マクロ変数 MQCD_CLIENT_CONN_DEFAULT を使用できます。

どちらの方法を選んでも、*ClientConnOffset* または *ClientConnPtr* のいずれか一方しか使用できません。両方ともゼロでない場合、呼び出しは失敗し、理由コード MQRC_CLIENT_CONN_ERROR が戻ります。

MQCONNX 呼び出しが完了した後は、MQCD 構造体が参照されることはありません。

Version が MQCNO_VERSION_2 より前の場合、このフィールドは無視されます。

注：プログラミング言語がそのポインターのデータ・タイプをサポートしないプラットフォームでは、このフィールドは初期値がすべてヌルのバイト・ストリングである、適当な長さのバイト・ストリングとして宣言されます。

ConnTag (MQBYTE128)

Multi

接続タグは、概念的には接続 ID に似ていますが、複数の関連する接続にまたがり、それらを単一のアプリケーション・インスタンスとして識別する場合があります。マルチプラットフォームでは、接続タグは接続時にキューマネージャによって生成されます。

詳しくは、[接続 ID](#) および [アプリケーション・インスタンス](#) を参照してください。

生成された接続タグは人間が読める semi 形式です。つまり、それらは、ローカル文字セットの文字列のように MQSC で表示およびフィルタリングできます。IBM MQ によって関連すると認識された接続には、自動的に同じ接続タグが割り当てられます。この割り当ては、特に [アプリケーション・バランシング](#) にとって重要です。

生成された接続タグは、以下の 3 つの方法で表示できます。

- MQCONN 呼び出しでの MQCNO 構造体の出力 (`MQCNO_GENERATE_CONN_TAG` が指定されている場合)。
- `DISPLAY CONN` (またはそれに相当するプログラム) からの出力。
- `DISPLAY APSTATUS` (またはそれに相当するもの) からの出力。

アプリケーションが終了すると、または、MQDISC 呼び出しが行われると、タグは無効になります。

IBM MQ for z/OS での ConnTag (MQBYTE128)

z/OS

接続タグは、概念的には接続 ID に似ていますが、複数の関連する接続にまたがり、それらを単一のアプリケーション・インスタンスとして識別する場合があります。IBM MQ for z/OS では、接続タグは、アプリケーションによって提供される入力フィールドであり、そのアプリケーション・インスタンスからの接続をシリアル化するために `MQCNO_*_CONN_TAG` オプションと組み合わせて使用されます。

同時に接続することを意図したアプリケーションの複数のインスタンスが存在する場合、それぞれがこのフィールドに固有値を提供する必要があります。詳細については、これらの [接続タグ・オプション](#) の説明を参照してください。

注：

- IBM MQ for z/OS では、実行時にアプリケーションと関連付けられた接続タグを管理的に判別する方法はありません。
- ASCII と EBCDIC のどちらの場合も、大文字、小文字、および大/小文字混合の MQ で始まる接続タグ値は、IBM 製品による使用のため予約済みです。これらの文字で始まる接続タグ値は使用しないでください。

タグを必要としない場合は、次の特殊値を使用します。

MQCT_NONE

値は、フィールドの長さについては 2 進ゼロです。

C 言語の場合、定数 `MQCT_NONE_ARRAY` も定義されます。この定数は、`MQCT_NONE` と同じ値ですが、ストリングではなく文字の配列です。

ConnTag フィールドは、z/OS キュー・マネージャーと接続するときに使用されます。

このフィールドの長さは `MQ_CONN_TAG_LENGTH` によって指定されます。Version が `MQCNO_VERSION_3` より前の場合、このフィールドは無視されます。

Multi

接続タグの使用方法については、を [282 ページの『ConnTag \(MQBYTE128\)』](#) 参照してください IBM MQ for Multiplatforms。

SSLConfigPtr (PMQSCO)

SSLConfigPtr は入力フィールドです。初期値は、ポインターをサポートするプログラミング言語ではヌル・ポインターです。それ以外の場合は、すべてヌルのバイト・ストリングです。

SSLConfigPtr および SSLConfigOffset は、MQCONNX 呼び出しを発行するアプリケーションが IBM MQ MQI client として実行されており、チャンネル・プロトコルが TCP/IP の場合にのみ使用してください。アプリケーションが IBM MQ クライアントとして実行されていない場合、またはチャンネル・プロトコルが TCP/IP でない場合、SSLConfigPtr と SSLConfigOffset は無視されます。

または SSLConfigPtr 指定 SSLConfigOffset し、さらにまたは ClientConnPtr のいずれかを指定することで、アプリケーションはクライアント接続における TLSClientConnOffset の使用を制御できます。TLS 情報をこの方法で指定した場合、環境変数 MQSSLKEYR および MQSSLCRYP は無視されます。クライアント・チャンネル定義テーブル (CCDT) の TLS 関連情報も無視されます。

TLS 情報は、以下の場合のみ指定されます。

- クライアント・プロセスの最初の MQCONNX 呼び出し
- キュー・マネージャーへの以前のすべての TLS 接続が MQDISC を使って終了されている場合、後続の MQCONNX 呼び出し

これらは単にプロセス規模の TLS 環境が初期化できる状態でしかありません。TLS 環境がすでに存在する場合に TLS 情報を指定して MQCONNX 呼び出しが発行されると、呼び出し時の TLS 情報は無視され、既存の TLS 情報を使って接続が行われます。この場合、呼び出しは完了コード MQCC_WARNING および理由コード MQRC_SSL_ALREADY_INITIALIZED を戻します。

MQSCO 構造体は、SSLConfigPtr にアドレスを指定するか、または SSLConfigOffset にオフセットを指定することによって、MQCD 構造体と同様に提供されます。この方法の詳細については、ClientConnPtr の説明を参照してください。ただし、と SSLConfigPtr のうち 1 つしか使用できません。両方を使用すると、理由コード SSLConfigOffset MQRC_SSL_CONFIG_ERROR で呼び出しが失敗します。両方がゼロでない場合。

MQCONNX 呼び出しが完了した後は、MQSCO 構造体が参照されることはありません。

このフィールドは、Version が MQCNO_VERSION_4 より前の場合には無視されます。

注：プログラミング言語がポインターのデータ・タイプをサポートしていないプラットフォームでは、このフィールドは適切な長さのバイト・ストリングとして宣言されます。

SSLConfigOffset (MQLONG)

SSLConfigOffset は、MQCNO 構造体の先頭からの MQSCO 構造体のオフセットをバイト数で表したものです。オフセットの値は、正負どちらの値にもなります。このフィールドは入力フィールドです。初期値は 0 です。

SSLConfigOffset は、MQCONNX 呼び出しを発行するアプリケーションが IBM MQ MQI client として実行されている場合にのみ使用されます。このフィールドの使用方法の詳細は、SSLConfigPtr フィールドの説明を参照してください。

このフィールドは、Version が MQCNO_VERSION_4 より前の場合には無視されます。

ConnectionId (MQBYTE24)

この接続 ID は 24 バイトの固有の ID であり、IBM MQ はこれを使用することにより、アプリケーションを確実に識別することができます。アプリケーションはこの ID を、PUT 呼び出しと GET 呼び出しの関連で使用します。この出力パラメーターの初期値は、すべてのプログラミング言語で 24 ヌル・バイトです。

接続がどのように確立されたかに関係なく、キュー・マネージャーは確立されたすべての接続に対して固有の ID を割り当てます。MQCONNX がバージョン 5 の MQCNO との接続を確立した場合、アプリケーションは、戻される MQCNO から ConnectionId を確認できます。割り当てられる ID は確実に、IBM MQ が生成する他のすべての ID の中で固有のものとなります。例えば、CorrelId、MsgID、GroupId などです。

PCF コマンド Inquire Connection または MQSC コマンド DISPLAY CONN を使って ConnectionId を使用して、長時間実行されている作業単位を識別します。MQSC コマンド (CONN) で使用される ConnectionId は、ここで戻される ConnectionId から派生します。PCF の Inquire および Stop Connection コマンドは、ここで戻される ConnectionId を、変更なしで使用できます。

PCF コマンド Stop Connection または MQSC コマンド STOP CONN を使って ConnectionId を指定することにより、ConnectionId を使用して、長時間実行されている作業単位を強制的に終了することができます。これらのコマンドの使用の詳細については、[Stop Connection](#) および [STOP CONN](#) を参照してください。

バージョンが MQCNO_VERSION_5 より前の場合、このフィールドは戻されません。

このフィールドの長さは MQ_CONNECTION_ID_LENGTH によって指定されます。

SecurityParmsOffset (MQLONG)

SecurityParmsOffset は、MQCNO 構造体の先頭からの MQCSP 構造体のオフセットをバイト数で表したものです。オフセットの値は、正負どちらの値にもなります。このフィールドは入力フィールドです。初期値は 0 です。

Version が MQCNO_VERSION_5 より前の場合、このフィールドは無視されます。

MQCSP 構造体は 286 ページの『[MQCSP - セキュリティー・パラメーター](#)』で定義されます。

SecurityParmsPtr (PMQCSP)

SecurityParmsPtr は MQCSP 構造体のアドレスです。これは、許可サービスによる認証用のユーザー ID とパスワードを指定するために使用されます。このフィールドは入力フィールドです。初期値はヌル・ポインターまたはヌル・バイトです。

Version が MQCNO_VERSION_5 より前の場合、このフィールドは無視されます。

MQCSP 構造体は 286 ページの『[MQCSP - セキュリティー・パラメーター](#)』で定義されます。

Reserved

構造体を 64 ビット境界外に埋め込む予約フィールド。フィールドの初期値は、フィールドの長さを示す 2 進ゼロです。

このフィールドは、Version が MQCNO_VERSION_6 より前の場合には無視されます。

CCDTUrlLength (MQLONG)

CCDTUrlLength CCDTUrlPtr または CCDTUrlOffset のいずれかで識別される文字列の長さです。この文字列には、接続に使用するクライアント接続チャネルテーブルの場所を識別する URL が含まれています。フィールドの初期値は 0 です。最大値は、オペレーティングシステムの最大ファイルパス長です。

CCDTUrlLength は、MQCONN 呼び出しを発行するアプリケーションが IBM MQ MQI client として実行されている場合にのみ使用されます。

これは、[MQCHLLIB](#) および [MQCHLTAB](#) の環境変数を設定することに対する、プログラムによる代替手段です。

アプリケーションがクライアントとして実行されていない場合、CCDTUrlLength は無視されます。

このフィールドは、Version が MQCNO_VERSION_6 より前の場合には無視されます。

CCDTUrlPtr (PMQCHAR)

CCDTUrlPtr は、接続に使用するクライアント接続チャネル・テーブルの場所を示す URL が含まれるストリングへの、オプションのポインターです。このフィールドは入力フィールドです。ポインターをサポートするプログラミング言語の場合は、ヌル・ポインターが初期値になり、それ以外の場合は、すべてヌルのバイト・ストリングが初期値になります。

CCDTUrlPtr は、MQCONN 呼び出しを発行するアプリケーションが IBM MQ MQI client として実行されている場合にのみ使用されます。

重要: CCDTUrlPtr と CCDTUrlOffset のいずれか 1 つだけ使用できます。両方のフィールドがゼロ以外の場合、呼び出しは理由コード MQRC_CCDT_URL_ERROR で失敗します。

これは、[MQCHLLIB](#) および [MQCHLTAB](#) の環境変数を設定することに対する、プログラムによる代替手段です。

アプリケーションがクライアントとして実行されていない場合、CCDTUrlPtr は無視されます。

このフィールドは、Version が MQCNO_VERSION_6 より前の場合には無視されます。

CCDTUrlOffset (MQLONG)

CCDTUrlOffset は、MQCNO 構造体の始まりから、接続に使用するクライアント接続チャンネル・テーブルの場所を示す URL が含まれる文字列までの、オフセットです (バイト単位)。オフセットの値は、正負どちらの値にもなります。フィールドの初期値は 0 です。

CCDTUrlOffset は、MQCONN 呼び出しを発行するアプリケーションが IBM MQ MQI client として実行されている場合にのみ使用されます。

重要: CCDTUrlPtr と CCDTUrlOffset のいずれか 1 つだけ使用できます。両方のフィールドがゼロ以外の場合、呼び出しは理由コード MQRC_CCDT_URL_ERROR で失敗します。

これは、[MQCHLLIB](#) および [MQCHLTAB](#) の環境変数を設定することに対する、プログラムによる代替手段です。

アプリケーションがクライアントとして実行されていない場合、CCDTUrlOffset は無視されます。

このフィールドは、Version が MQCNO_VERSION_6 より前の場合には無視されます。

ApplName (MQCHAR28)

キュー・マネージャーへの接続を識別するために、アプリケーションによって設定された名前。フィールドの初期値は MQAN_NONE_ARRAY (ブランク文字) です。

このフィールドは、Version が MQCNO_VERSION_7 より前の場合や、値がブランクに設定されている場合には無視されます。

 このフィールドは、z/OS では設定できません。これを行おうとすると、MQRC_CNO_ERROR 理由コードを受け取ります。

Reserved2 (MQBYTE4)

構造体を 64 ビット境界外に埋め込む予約フィールド。フィールドの初期値は、フィールドの長さを示す 2 進ゼロです。

このフィールドは、Version が MQCNO_VERSION_7 より前の場合には無視されます。

BalanceParmsOffset

タイプ MQBNO の構造体のメモリー・ロケーション。これには、アプリケーションの平衡化動作に関する情報が含まれます。アプリケーションがクライアント・チャンネルを介して接続していない限り、構造は完全に無視されます。

このフィールドは、MQCNO_VERSION_8 Version 未満の場合には無視されます。

詳細については、[MQBNO](#) を参照してください。

このフィールドを指定すると、286 ページの『BalanceParmsPtr』フィールドを指定できません。両方のフィールドを指定しようとすると、MQRC_CNO_ERROR が表示されます。このフィールドはクライアント接続にのみ関係するため、このフィールドを他の接続タイプに指定すると、MQRC_CNO_ERROR が発生します。

BalanceParmsPtr

アプリケーションの平衡化動作に関する情報を含む、タイプ MQBNO の構造体のメモリー・ロケーションへのポインターです。アプリケーションがクライアント・チャンネルを介して接続していない限り、構造は完全に無視されます。

このフィールドは、MQCNO_VERSION_8Version 未満の場合に無視されます。

詳細については、[MQBNO](#) を参照してください。

このフィールドを指定すると、285 ページの『BalanceParmsOffset』フィールドを指定できません。両方のフィールドを指定しようとすると、MQRC_CNO_ERROR が表示されます。このフィールドはクライアント接続にのみ関係するため、このフィールドを他の接続タイプに指定すると、MQRC_CNO_ERROR が発生します。

関連タスク

[MQCONN](#) の使用法

MQCSP - セキュリティー・パラメーター

接続セキュリティパラメータ IBM MQ 構造は、アプリケーションが MQCONN 呼び出しにおいて認証情報をキューマネージャに伝達するために使用される。また、機密データを暗号化するパスワード IBM MQ 保護システムで使用される初期キーを提供するためにも使用できます。

MQCSP_AUTH_USER_ID_AND_PWD に *AuthenticationType* 設定して、バージョン 1 のユーザー ID とパスワードを含める。

バージョン 2 で初期キー情報を提供すると、デフォルト値は *AuthenticationType* MQCSP_AUTH_NONE になります。

から IBM MQ 9.3.4、認証トークン情報を含めるには *AuthenticationType* を使用します。

MQCSP_AUTH_USER_ID_AND_PWD または MQCSP_AUTH_ID_TOKEN のいずれかを使用できますが、両方同時に使用することはできません。

警告: 場合によっては、クライアントアプリケーションの MQCSP 構造体内のパスワードまたは認証トークンが、ネットワーク上で平文で送信されることがある。クライアントアプリケーションのパスワードと認証トークンが適切に保護されるようにするには、[MQCSP パスワード保護](#)を参照してください。

可用性

MQCSP 構造は、サポートされている IBM MQ すべてのプラットフォームで利用可能です。

バージョン

サポート対象のプログラミング言語向けに提供されるヘッダー、COPY、および INCLUDE ファイルには、最新の MQCSP バージョンが含まれていますが、フィールド *Version* の初期値は MQCSP_VERSION_1 に設定されています。version-1 構造体に存在しないフィールドを使用するには、アプリケーションは必要なバージョン番号を *Version* フィールドに設定する必要があります。

文字セットとエンコード

MQCSP 内のデータは、ローカルキューマネージャの文字セットおよびエンコーディングでなければならない。これらはそれぞれキューマネージャ **CodedCharSetId** 属性および MQENC_NATIVE によって指定される。

フィールド

注: 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 489. MQCSP における MQCSP のフィールド

フィールド名と説明	定数の名前	定数の初期値（存在する場合）
StrucId (構造識別子)	MQCSP_STRUC_ID	'CSP-'
バージョン (構造バージョン番号)	MQCSP_VERSION_1	1
AuthenticationType (認証の種類)	なし	MQCSP_AUTH_NONE
Reserved1 (ポインタのアラインメントに必要 IBM i)	なし	ヌル・ストリングまたはブランク
CSPUserIdPtr (ユーザー ID のアドレス)	なし	ヌル・ポインターまたはヌル・バイト
CSPUserIdOffset (ユーザー ID のオフセット)	なし	0
CSPUserIdLength (ユーザー ID の長さ)	なし	0
Reserved2 (ポインタのアラインメントに必要 IBM i)	なし	ヌル・ストリングまたはブランク
CSPPasswordPtr (パスワードのアドレス)	なし	ヌル・ポインターまたはヌル・バイト
CSPPasswordOffset (パスワードのオフセット)	なし	0
CSPPasswordLength (パスワードの長さ)	なし	0
注：が より MQCSP_VERSION_2 小さい Version 場合、残りのフィールドは無視される。		
Reserved3 (ポインタのアラインメントに必要 IBM i)	なし	ヌル・ストリングまたはブランク
InitialKeyPtr	なし	ヌルポインタまたは空白
InitialKeyOffset (パスワード保護システムの初期キーのオフセット)	なし	0
InitialKeyLength (パスワード保護システムの初期キーの長さ)	なし	0
注：が より MQCSP_VERSION_3 小さい Version 場合、残りのフィールドは無視される。		
Reserved4 (ポインタのアラインメントに必要 IBM i)	なし	ヌル・ストリングまたはブランク
TokenPtr (認証トークンのアドレス)	なし	ヌルポインタまたは空白
TokenKeyOffset (認証トークンのオフセット)	なし	0
TokenLength (認証トークンの長さ)	なし	0
注：		
1. 記号-は、単一のブランク文字を表します。		
2. C プログラミング言語では、マクロ変数 MQCSP_DEFAULT には、表にリストされている値が含まれています。この変数を以下の方法で使用する、構造体のフィールドに初期値を設定できます。		
<pre>MQCSP MyCSP = {MQCSP_DEFAULT};</pre>		

言語ごとの宣言

MQCSP の C 宣言

```
typedef struct tagMQCSP MQCSP;
struct tagMQCSP {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQLONG     AuthenticationType; /* Type of authentication */
    MQBYTE4    Reserved1;        /* Required for IBM i pointer alignment */
    MQPTR      CSPUserIdPtr;      /* Address of user ID */
    MQLONG     CSPUserIdOffset;   /* Offset of user ID */
    MQLONG     CSPUserIdLength;   /* Length of user ID */
    MQBYTE8    Reserved2;        /* Required for IBM i pointer alignment */
    MQPTR      CSPPasswordPtr;    /* Address of password */
    MQLONG     CSPPasswordOffset; /* Offset of password */
    MQLONG     CSPPasswordLength; /* Length of password */
/* Ver:1 */

    MQBYTE8    Reserved3;        /* Required for IBM i pointer alignment */
    MQPTR      InitialKeyPtr;     /* Address of initial key */
    MQLONG     InitialKeyOffset;  /* Offset of initial key */
    MQLONG     InitialKeyLength;  /* Length of initial key */
/* Ver:2 */
};
```

```
typedef struct tagMQCSP MQCSP;
struct tagMQCSP {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQLONG     AuthenticationType; /* Type of authentication */
    MQBYTE4    Reserved1;        /* Required for IBM i pointer alignment */
    MQPTR      CSPUserIdPtr;      /* Address of user ID */
    MQLONG     CSPUserIdOffset;   /* Offset of user ID */
    MQLONG     CSPUserIdLength;   /* Length of user ID */
    MQBYTE8    Reserved2;        /* Required for IBM i pointer alignment */
    MQPTR      CSPPasswordPtr;    /* Address of password */
    MQLONG     CSPPasswordOffset; /* Offset of password */
    MQLONG     CSPPasswordLength; /* Length of password */
/* Ver:1 */

    MQBYTE8    Reserved3;        /* Required for IBM i pointer alignment */
    MQPTR      InitialKeyPtr;     /* Address of initial key */
    MQLONG     InitialKeyOffset;  /* Offset of initial key */
    MQLONG     InitialKeyLength;  /* Length of initial key */
/* Ver:2 */

    MQBYTE8    Reserved4;        /* Required for IBM i pointer alignment */
    MQPTR      TokenPtr;          /* Address of token */
    MQLONG     TokenOffset;       /* Offset of token */
    MQLONG     TokenLength;       /* Length of token */
/* Ver:3 */
};
```

MQCSP の COBOL 宣言

```
**      MQCSP structure
10 MQCSP.
**      Structure identifier
15 MQCSP-STRUCID      PIC X(4).
**      Structure version number
15 MQCSP-VERSION      PIC S9(9) BINARY.
**      Type of authentication
15 MQCSP-AUTHENTICATIONTYPE PIC S9(9) BINARY.
**      Required for IBM i pointer alignment
15 MQCSP-RESERVED1    PIC X(4).
**      Address of user ID
15 MQCSP-CSPUSERIDPTR    POINTER.
**      Offset of user ID
15 MQCSP-CSPUSERIDOFFSET PIC S9(9) BINARY.
**      Length of user ID
15 MQCSP-CSPUSERIDLENGTH PIC S9(9) BINARY.
**      Required for IBM i pointer alignment
15 MQCSP-RESERVED2    PIC X(4).
**      Address of password
15 MQCSP-CSPPASSWORDPTR    POINTER.
**      Offset of password
```

```

15 MQCSP-CSPPASSWORDOFFSET PIC S9(9) BINARY.
** Length of password
15 MQCSP-CSPPASSWORDLENGTH PIC S9(9) BINARY.
** Ver:1 **

** Reserved
15 MQCSP-RESERVED3 PIC X(8).
** Address of initial key
15 MQCSP-INITIALKEYPTR POINTER.
** Offset of initial key
15 MQCSP-INITIALKEYOFFSET PIC S9(9) BINARY.
** Length of initial key
15 MQCSP-INITIALKEYLENGTH PIC S9(9) BINARY.
** Ver:2 **

```

```

** MQCSP structure
10 MQCSP.
** Structure identifier
15 MQCSP-STRUCID PIC X(4).
** Structure version number
15 MQCSP-VERSION PIC S9(9) BINARY.
** Type of authentication
15 MQCSP-AUTHENTICATIONTYPE PIC S9(9) BINARY.
** Required for IBM i pointer alignment
15 MQCSP-RESERVED1 PIC X(4).
** Address of user ID
15 MQCSP-CSPUSERIDPTR POINTER.
** Offset of user ID
15 MQCSP-CSPUSERIDOFFSET PIC S9(9) BINARY.
** Length of user ID
15 MQCSP-CSPUSERIDLENGTH PIC S9(9) BINARY.
** Required for IBM i pointer alignment
15 MQCSP-RESERVED2 PIC X(4).
** Address of password
15 MQCSP-CSPPASSWORDPTR POINTER.
** Offset of password
15 MQCSP-CSPPASSWORDOFFSET PIC S9(9) BINARY.
** Length of password
15 MQCSP-CSPPASSWORDLENGTH PIC S9(9) BINARY.
** Ver:1 **

** Reserved
15 MQCSP-RESERVED3 PIC X(8).
** Address of initial key
15 MQCSP-INITIALKEYPTR POINTER.
** Offset of initial key
15 MQCSP-INITIALKEYOFFSET PIC S9(9) BINARY.
** Length of initial key
15 MQCSP-INITIALKEYLENGTH PIC S9(9) BINARY.
** Ver:2 **

** Reserved
15 MQCSP-RESERVED4 PIC X(8).
** Address of token
15 MQCSP-TOKENPTR POINTER.
** Offset of token
15 MQCSP-TOKENOFFSET PIC S9(9) BINARY.
** Length of token
15 MQCSP-TOKENLENGTH PIC S9(9) BINARY.
** Ver:3 **

```

MQCSP の PL/I 宣言

```

dcl
  1 MQCSP based,
    3 StrucId          char(4),          /* Structure identifier */
    3 Version          fixed bin(31),    /* Structure version number */
    3 AuthenticationType fixed bin(31),  /* Type of authentication */
    3 Reserved1        char(4),          /* Required for IBM i pointer
                                          alignment */
    3 CSPUserIdPtr      pointer,          /* Address of user ID */
    3 CSPUserIdOffset   fixed bin(31),    /* Offset of user ID */
    3 CSPUserIdLength   fixed bin(31),    /* Length of user ID */
    3 Reserved2        char(8),          /* Required for IBM i pointer
                                          alignment */
    3 CSPPasswordPtr     pointer,          /* Address of password */
    3 CSPPasswordOffset fixed bin(31),    /* Offset of user ID */
    3 CSPPasswordLength fixed bin(31),    /* Length of user ID */

```

```

/* Version 1 */

3 Reserved3      char(8),          /* Reserved */
3 InitialKeyPtr   pointer,          /* Address of initial key */
3 InitialKeyOffset fixed bin(31), /* Offset of initial key */
3 InitialKeyLength fixed bin(31); /* Length of initial key */
/* Version 2 */

dcl
1 MQCSP based,
3 StrucId          char(4),          /* Structure identifier */
3 Version          fixed bin(31), /* Structure version number */
3 AuthenticationType fixed bin(31), /* Type of authentication */
3 Reserved1        char(4),          /* Required for IBM i pointer
                                     alignment */
3 CSPUserIdPtr     pointer,          /* Address of user ID */
3 CSPUserIdOffset  fixed bin(31), /* Offset of user ID */
3 CSPUserIdLength  fixed bin(31), /* Length of user ID */
3 Reserved2        char(8),          /* Required for IBM i pointer
                                     alignment */
3 CSPPasswordPtr   pointer,          /* Address of password */
3 CSPPasswordOffset fixed bin(31), /* Offset of user ID */
3 CSPPasswordLength fixed bin(31), /* Length of user ID */
/* Version 1 */

3 Reserved3      char(8),          /* Reserved */
3 InitialKeyPtr   pointer,          /* Address of initial key */
3 InitialKeyOffset fixed bin(31), /* Offset of initial key */
3 InitialKeyLength fixed bin(31); /* Length of initial key */
/* Version 2 */

3 Reserved4      char(8),          /* Reserved */
3 TokenPtr        pointer,          /* Address of Token */
3 TokenOffset     fixed bin(31), /* Offset of Token */
3 TokenLength     fixed bin(31); /* Length of Token */
/* Version 3 */

```

MQCSP の Visual Basic 宣言

```

Type MQCSP
    StrucId          As String*4    'Structure identifier'
    Version          As Long        'Structure version number'
    AuthenticationType As Long      'Type of authentication'
    Reserved1        As MQBYTE4     'Required for IBM i pointer'
                                     'alignment'
    CSPUserIdPtr     As MQPTR       'Address of user ID'
    CSPUserIdOffset  As Long        'Offset of user ID'
    CSPUserIdLength  As Long        'Length of user ID'
    Reserved2        As MQBYTE8     'Required for IBM i pointer'
                                     'alignment'
    CSPPasswordPtr   As MQPTR       'Address of password'
    CSPPasswordOffset As Long      'Offset of password'
    CSPPasswordLength As Long      'Length of password'
End Type

```

MQCSP のフィールド詳細

接続セキュリティパラメータ IBM MQ 構造は、アプリケーションが MQCONN 呼び出しにおいて認証情報をキューマネージャに伝達するために使用される。また、機密データを暗号化するパスワード IBM MQ 保護システムで使用される初期キーを提供するためにも使用できます。

フィールド

注：次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 490. MQCSP における MQCSP のフィールド		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
<u>StrucId</u> (構造識別子)	MQCSP_STRUC_ID	'CSP'
<u>バージョン</u> (構造バージョン番号)	MQCSP_VERSION_1	1

表 490. MQCSP における MQCSP のフィールド (続き)

フィールド名と説明	定数の名前	定数の初期値 (存在する場合)
<u>AuthenticationType</u> (認証の種類)	なし	MQCSP_AUTH_NONE
<u>Reserved1</u> (ポインタのアラインメントに必要な IBM i)	なし	ヌル・ストリングまたはブランク
<u>CSPUserIdPtr</u> (ユーザー ID のアドレス)	なし	ヌル・ポインターまたはヌル・バイト
<u>CSPUserIdOffset</u> (ユーザー ID のオフセット)	なし	0
<u>CSPUserIdLength</u> (ユーザー ID の長さ)	なし	0
<u>Reserved2</u> (ポインタのアラインメントに必要な IBM i)	なし	ヌル・ストリングまたはブランク
<u>CSPPasswordPtr</u> (パスワードのアドレス)	なし	ヌル・ポインターまたはヌル・バイト
<u>CSPPasswordOffset</u> (パスワードのオフセット)	なし	0
<u>CSPPasswordLength</u> (パスワードの長さ)	なし	0
注: が より MQCSP_VERSION_2 小さい <i>Version</i> 場合、残りのフィールドは無視される。		
<u>Reserved3</u> (ポインタのアラインメントに必要な IBM i)	なし	ヌル・ストリングまたはブランク
<u>InitialKeyPtr</u>	なし	ヌルポインタまたは空白
<u>InitialKeyOffset</u> (パスワード保護システムの初期キーのオフセット)	なし	0
<u>InitialKeyLength</u> (パスワード保護システムの初期キーの長さ)	なし	0
注: が より MQCSP_VERSION_3 小さい <i>Version</i> 場合、残りのフィールドは無視される。		
<u>Reserved4</u> (ポインタのアラインメントに必要な IBM i)	なし	ヌル・ストリングまたはブランク
<u>TokenPtr</u> (認証トークンのアドレス)	なし	ヌルポインタまたは空白
<u>TokenKeyOffset</u> (認証トークンのオフセット)	なし	0
<u>TokenLength</u> (認証トークンの長さ)	なし	0
注: 1. 記号-は、単一のブランク文字を表します。 2. C プログラミング言語では、マクロ変数 MQCSP_DEFAULT には、表にリストされている値が含まれています。この変数を以下の方法で使用すると、構造体のフィールドに初期値を設定できます。 <pre>MQCSP MyCSP = {MQCSP_DEFAULT};</pre>		

StrucId (MQCHAR4)

これはセキュリティパラメータ構造体の構造体識別子です。常に入力フィールドです。その値は MQCSP_STRUC_ID です。

値は次のものでなければなりません。

MQCSP_STRUC_ID

セキュリティ・パラメータ構造体の ID。

C プログラミング言語では、定数 MQCSP_STRUC_ID_ARRAY も定義されている。これは MQCSP_STRUC_ID と同じ値を持ちますが、文字列ではなく文字の配列です。

Version (MQLONG)

MQCSP 構造のバージョン番号

値は次のものでなければなりません。

MQCSP_VERSION_1

バージョン 1 のセキュリティ・パラメータ構造体。バージョン 1 では、MQCSP 構造体にユーザー ID とパスワードを含めることで、キューマネージャとの認証を行うことができます。

MQCSP_VERSION_2

Version-2 セキュリティパラメータ構造体。バージョン 2 では、キュー管理者と認証するためのユーザー ID とパスワードを含めることができ、パスワード保護に使用される初期キーを指定できます。

MQCSP_VERSION_3

Version-3 セキュリティパラメータ構造体。バージョン 3 では、MQCSP 構造体にユーザー ID とパスワード、または認証トークンのいずれかを指定することで、キューマネージャへの認証が可能となります。パスワード保護に使用される初期キーを指定することもできます。

以下の定数は、現行バージョンのバージョン番号を指定しています。

MQCSP_CURRENT_VERSION

セキュリティ・パラメータ構造体の現行バージョン。

これは常に入力フィールドです。このフィールドの初期値は MQCSP_VERSION_3 です。

AuthenticationType (MQLONG)

AuthenticationType は入力フィールドです。初期値は、MQCSP_AUTH_NONE です。

これは、実行する認証のタイプです。有効な値は以下のとおりです。

MQCSP_AUTH_NONE

ユーザー ID とパスワード、または認証トークンのフィールドを使用しないでください。

MQCSP_AUTH_USER_ID_AND_PWD

MQCSP 構造内のユーザー ID とパスワードを使用して認証します。

MQCSP 認証 ID トークン

MQCSP 構造内の認証トークンを使用して認証する。

デフォルト値は MQCSP_AUTH_NONE です。デフォルト設定の場合、パスワード保護は行われません。

認証が必要な場合は、を設定する必要があります **MQCSP.AuthenticationType**

MQCSP_AUTH_USER_ID_AND_PWD または MQCSP_AUTH_ID_TOKEN へ

詳しくは、以下を参照してください

- [MQCSP パスワード保護](#)
- [認証トークンを使用した作業](#)

Reserved1 (MQBYTE4)



IBM i 上のポインターの位置合わせに必要な予約フィールド。

このフィールドの初期値は、すべてヌルです。

CSPUserIdPtr (MQPTR)

認証に使用するユーザー ID のアドレス。

これは入力フィールドです。このフィールドの初期値は、ポインターをサポートするプログラミング言語のヌル・ポインターです。それ以外の場合は、すべてヌルのバイトのストリングです。Version が MQCNO_VERSION_5 より前の場合、このフィールドは無視されます。

このフィールドには、キューマネージャの [CONNAUTH](#) フィールドに **IDPWOS AUTHTYPE** が指定されている場合、オペレーティングシステムのユーザー ID が含まれることがあります。

Windows では、これに完全修飾したドメイン・ユーザー ID を指定できます。

このフィールドには、キューマネージャの [CONNAUTH](#) フィールドに **IDPWLDAP AUTHTYPE** が指定されている場合、LDAP のユーザー ID が含まれることがあります。

CSPUserIdOffset (MQLONG)

認証に使用するユーザー ID のオフセット（バイト単位）。オフセットの値は、正負どちらの値にもなります。

これは入力フィールドです。このフィールドの初期値は 0 です。

CSPUserIdLength (MQLONG)

認証に使用するユーザー ID の長さ。

ユーザー ID の最大長は、プラットフォームによって異なります。[ユーザー ID](#) を参照してください。ユーザー ID の長さが許可される最大長を超える場合、認証要求は MQRC_CSP_ERROR で失敗します。以前のバージョンの IBM MQ、返されるエラーは MQRC_NOT_AUTHORIZED です。

このフィールドは入力フィールドです。このフィールドの初期値は 0 です。

Reserved2 (MQBYTE8)



IBM i 上のポインターの位置合わせに必要な予約フィールド。

このフィールドの初期値は、すべてヌルです。

CSPPasswordPtr (MQPTR)

認証に使用するパスワードのアドレス。

これは入力フィールドです。このフィールドの初期値は、ポインターをサポートするプログラミング言語のヌル・ポインターです。それ以外の場合は、すべてヌルのバイトのストリングです。Version が MQCNO_VERSION_5 より前の場合、このフィールドは無視されます。

このフィールドには空のパスワードが含まれる場合があります。設定によってはオペレーティングシステムまたは LDAP のパスワード検証で拒否されますが、認証メソッドに渡される前には IBM MQ 拒否されません。

CSPPasswordOffset (MQLONG)

これは認証に使用するパスワードのバイト単位のオフセットです。オフセットの値は、正負どちらの値にもなります。

これは入力フィールドです。このフィールドの初期値は 0 です。

CSPPasswordLength (MQLONG)

認証に使用するパスワードの長さ。

パスワードの最大長は MQ_CSP_PASSWORD_LENGTH であり、256 文字です。パスワードの長さが許可される最大長を超える場合、認証要求は MQRC_CSP_ERROR で失敗します。以前のバージョンの IBM MQ、返されるエラーは MQRC_NOT_AUTHORIZED です。

このフィールドは入力フィールドです。このフィールドの初期値は 0 です。

Reserved3 (MQBYTE8)



IBM i 上のポインタの位置合わせに必要な予約フィールド。

このフィールドの初期値は、すべてヌルです。

InitialKeyPtr (MQPTR)



パスワード保護システムの初期キーのアドレス。

これは入力フィールドです。このフィールドの初期値は、ポインタをサポートするプログラミング言語のヌル・ポインタです。それ以外の場合は、すべてヌルのバイトのストリングです。このフィールドは、MQCSP_VERSION_2Version 未満の場合に無視されます。

このフィールドは、マルチプラットフォーム上で実行されている MQI クライアントに IBM MQ のみ適用されます。

IBM MQMQI クライアントは、パスワード保護システム IBM MQ を利用して、一部のフィールドに対して暗号化された値を指定することができます。MQSCO 構造で指定されたキーリポジトリのパスワードを暗号化するために初期キーを使用した場合、同じクライアントアプリケーションの MQCSP 構造に初期キーフィールドを含めることを確認してください。

暗号化アルゴリズムは、これらの値を暗号化および復号化するために初期鍵を使用します。これらのフィールドの値をユーティリティ **runmqicred** で暗号化する際に初期キーが指定された場合、クライアントがキューマネージャに接続する際には、同じ初期キーを指定する必要があります。

このフィールドを使用して指定された初期キーは、MQS MQI KEYFILE 環境変数またはクライアント設定ファイルの Security スタンス内の MQIInitialKeyFile プロパティを使用して指定された初期キーを上書きします。

初期キーを指定するには *InitialKeyPtr*、または *InitialKeyOffset* のいずれかを使用できますが、両方を使用することはできません。

関連資料

- [初期キーの供給 IBM MQ MQI client AIX, Linux, and Windows](#)
- [コンポーネント設定 IBM MQ ファイルにおけるパスワードの保護](#)
- [runmqicred \(クライアントパスワード IBM MQ を保護\)](#)
- [548 ページの『KeyRepoPasswordPtr \(MQPTR\)』](#)

InitialKeyOffset (MQLONG)



パスワード保護システムの初期キーが MQCSP 構造体の先頭から始まるオフセット (バイト単位)。オフセットの値は、正負どちらの値にもなります。

初期キーを指定するには *InitialKeyPtr*、または *InitialKeyOffset* のいずれかを使用できますが、両方を使用することはできません。詳細については、フィールド *InitialKeyPtr* の説明を参照してください。

これは入力フィールドです。このフィールドの初期値は 0 です。このフィールドは、MQCSP_VERSION_2Version 未満の場合に無視されます。

関連資料

-  [初期キーの供給 IBM MQ MQI client AIX, Linux, and Windows](#)
- [コンポーネント設定 IBM MQ ファイルにおけるパスワードの保護](#)

InitialKeyLength (MQLONG)



パスワード保護システムの初期キーの長さ。

これは入力フィールドです。このフィールドの初期値は 0 です。このフィールドは、MQCSP_VERSION_2Version 未満の場合に無視されます。

Reserved4 (MQBYTE8)



IBM i 上のポインタの位置合わせに必要な予約フィールド。

このフィールドの初期値は、すべてヌルです。

TokenPtr (MQPTR)



キューマネージャとの認証に使用される認証トークンのアドレス。

これは入力フィールドです。このフィールドの初期値は、ポインタをサポートするプログラミング言語のヌル・ポインタです。それ以外の場合は、すべてヌルのバイトのストリングです。このフィールドは、MQCSP_VERSION_3Version 未満の場合に無視されます。

このフィールドは、Windows または Linux Linux AIX システム上で動作するキュー IBM MQ マネージャへの IBM MQ MQI clients 接続に関連します。

認証トークンを指定するには *TokenPtr*、または *TokenOffset* のいずれかを使用できますが、両方を使用することはできません。

関連資料

- [アプリケーションにおける認証トークンの使用](#)
- [認証トークンを使用した作業](#)

TokenOffset (MQLONG)



これは、MQCSP 構造体の先頭からの認証トークンのオフセット（バイト単位）です。オフセットの値は、正負どちらの値にもなります。

これは入力フィールドです。このフィールドの初期値は 0 です。このフィールドは、MQCSP_VERSION_3Version 未満の場合に無視されます。

トークンを指定するには *TokenPtr*、または *TokenOffset* のいずれかを使用できますが、両方を使用することはできません。詳細については、フィールド *TokenPtr* の説明を参照してください。

関連資料

- [アプリケーションにおける認証トークンの使用](#)
- [認証トークンを使用した作業](#)

TokenLength (MQLONG)



これは、キュー管理者と認証を行う際に使用される認証トークンの長さです。

認証トークンの最大長はMQ_CSP_TOKEN_LENGTHで、8192 バイトです。許可される最大長TokenLengthを超える場合、認証要求は失敗し、MQRC_CSP_ERROR エラーが発生します。

これは入力フィールドです。このフィールドの初期値は0です。このフィールドは、MQCSP_VERSION_3Version 未満の場合に無視されます。

関連資料

- [アプリケーションにおける認証トークンの使用](#)
- [認証トークンを使用した作業](#)

MQCTLO - コールバック制御オプション構造

MQCTLO 構造体は、コールバック制御関数に関係したオプションを指定するために使用されます。この構造体は、MQCTL 呼び出しにおける入力および出力パラメータです。

可用性

MQCTLO 構造体は以下のプラットフォームで利用可能です：

-  AIX
-  IBM i
-  Linux
-  Windows
-  z/OS

および、これらのシステムに接続された IBM MQ MQI clients。

バージョン

MQCTLO の現在のバージョンは MQCTLO_VERSION_1 です。

文字セットとエンコード

MQCTLO 内のデータは、キュー・マネージャ **CodedCharSetId** 属性によって指定される文字セットおよびMQENC_NATIVEによって指定されるローカル・キュー・マネージャのエンコーディングでなければならない。ただし、アプリケーションがMQ MQI クライアントとして実行されている場合、構造体はクライアントの文字セットとエンコードに従っている必要があります。

フィールド

注：次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 491. MQCTLO のフィールド		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
StrucID (構造識別子)	MQCTLO_STRUC_ID	'CTLO'
バージョン (構造バージョン番号)	MQCTLO_VERSION_1	1
オプション (オプション)	MQCTLO_NONE	Null
オプション (予約済み)	予約フィールド	
ConnectionArea (使用するコールバック関数のフィールド)	なし	ヌル・ポインターまたはヌル・バイト

表 491. MQCTLO のフィールド (続き)		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
注: 1. C プログラミング言語では、マクロ変数 MQCTLO_DEFAULT には、表にリストされている値が含まれています。このマクロ変数を以下の方法で使用して、構造体のフィールドに初期値を設定します。 <pre>MQCTLO MyCTLO = {MQCTLO_DEFAULT};</pre>		

言語ごとの宣言

MQCTLO の C 宣言

```
typedef struct tagMQCTLO MQCTLO;
struct tagMQCTLO {
    MQCHAR4    StrucId;        /* Structure identifier */
    MQLONG     Version;        /* Structure version number */
    MQLONG     Options;        /* Options that control the action of MQCTL */
    MQLONG     Reserved;       /* Reserved field */

    MQPTR      ConnectionArea; /* Connection work area passed to the function */
};
```

MQCTLO の COBOL 宣言

```
** MQCTLO structure
10  MQCTLO.
** Structure Identifier
15  MQCTLO-STRUCID                PIC X(4).
** Structure Version
15  MQCTLO-VERSION               PIC S9(9) BINARY.
** Options
15  MQCTLO-OPTIONS               PIC S9(9) BINARY.
** Reserved
15  MQCTLO-RESERVED              PIC S9(9) BINARY.
** ConnectionArea
15  MQCTLO-CONNECTIONAREA        POINTER
```

MQCTLO の PL/I 宣言

```
dcl
1  MQCTLO based,
3  StrucId          char(4),        /* Structure identifier */
3  Version          fixed bin(31), /* Structure version */
3  Options          fixed bin(31), /* Options */
3  Reserved         fixed bin(31),
3  ConnectionArea   pointer;        /* Connection work area */
```

MQCTLO のフィールド詳細

MQCTLO 構造体は、コールバック制御関数に関係したオプションを指定するために使用されます。この構造体は、MQCTL 呼び出しにおける入力および出力パラメータです。

フィールド

注: 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 492. MQCTLO のフィールド		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
<u>StrucID</u> (構造識別子)	MQCTLO_STRUC_ID	'CTLO'

表 492. MQCTLO のフィールド (続き)

フィールド名と説明	定数の名前	定数の初期値 (存在する場合)
<u>バージョン</u> (構造バージョン番号)	MQCTLO_VERSION_1	1
<u>オプション</u> (オプション)	MQCTLO_NONE	Null
<u>オプション</u> (予約済み)	予約フィールド	
<u>ConnectionArea</u> (使用するコールバック関数のフィールド)	なし	ヌル・ポインターまたはヌル・バイト
注: 1. C プログラミング言語では、マクロ変数 MQCTLO_DEFAULT には、表にリストされている値が含まれています。このマクロ変数を以下の方法で使用して、構造体のフィールドに初期値を設定します。 <pre>MQCTLO MyCTLO = {MQCTLO_DEFAULT};</pre>		

StrucID (MQCHAR4)

これは制御オプション構造体の構造体識別子です。常に入力フィールドです。その値は MQCTLO_STRUC_ID です。

値は次のものでなければなりません。

MQCTLO_STRUC_ID

制御オプション構造体の識別子。

C プログラミング言語では、定数 MQCTLO_STRUC_ID_ARRAY も定義されている。これは MQCTLO_STRUC_ID と同じ値を持ちますが、文字列ではなく文字の配列です。

Version (MQLONG)

これは構造体のバージョン番号です。値は以下のものでなければなりません。

MQCTLO_VERSION_1

バージョン 1 の制御オプション構造。

以下の定数は、現行バージョンのバージョン番号を指定しています。

MQCTLO_CURRENT_VERSION

制御オプション構造の現行バージョン。

これは常に入力フィールドです。フィールドの初期値は、MQCTLO_VERSION_1 です。

Options (MQLONG)

MQCTL のアクションを制御するオプション。

MQCTLO_FAIL_IF_QUIESCING

キュー・マネージャーまたは接続が静止状態にある場合、MQCTL 呼び出しが強制的に失敗します。

MQCB 呼び出しで渡される MQGMO オプション内に MQGMO_FAIL_IF_QUIESCING を指定すると、これらの静止時にメッセージ・コンシューマーに対して通知が行われます。

MQCTLO_THREAD_AFFINITY

このオプションは、アプリケーションで、同じ接続に関するすべてのメッセージ・コンシューマーが同じスレッド上で呼び出される必要があることを、システムに通知します。このスレッドは、接続が停止するまでコンシューマーのすべての呼び出しで使用されます。

デフォルト・オプション: 上記のいずれのオプションも必要でない場合は、以下のオプションを使用します。

MQCTLO_NONE

この値は、他のオプションが指定されなかったことを示すために使用します。すべてのオプションはデフォルト値であるとみなされます。MQCTLO_NONE は、プログラムの文書化を支援するために定義します。このオプションは、他のオプションと組み合わせて使用するオプションではありません。ただし、このオプションの値はゼロと等価なので、他のオプションと組み合わせて使用しても、エラーとして検出されることはありません。

これは入力フィールドです。Options フィールドの初期値は、MQCTLO_NONE です。

Reserved (MQLONG)

これは予約フィールドです。値はゼロでなければなりません。

ConnectionArea (MQPTR)

これは、コールバック関数ができるフィールドです。

キュー・マネージャーはこのフィールドの内容に基づいて決定せず、このフィールドの内容は、コールバックの入力パラメーターである MQCBC 構造内の ConnectionArea フィールドにそのまま渡されます。

MQOP_START および MQOP_START_WAIT 以外のすべての操作の場合、このフィールドは無視されます。

これは、コールバック関数への入出力フィールドです。このフィールドの初期値は、ヌル・ポインターまたはヌル・バイトです。

MQDH - 配布ヘッダー

MQDH 構造体は、メッセージが伝送キューに格納されている配布リスト・メッセージである場合に、そのメッセージ内の追加データを記述します。配布リスト・メッセージとは、複数の宛先キューに送信されるメッセージです。追加データは、MQDH 構造体、MQOR レコードの配列、MQPMR レコードの配列の順番で構成されています。この構造体は、メッセージを伝送キューに直接書き込んだり、伝送キューからメッセージを除去したりする特殊なアプリケーション (例えば、メッセージ・チャンネル・エージェント) で使用されます。メッセージを配布リストに書き込むアプリケーションは、この構造体を使用してはなりません。代わりに、MQOD 構造体を使用して配布リストに宛先を定義し、MQPMO 構造体を使用してメッセージのプロパティを指定したり個々の宛先に送信されるメッセージに関する情報を受信したりする必要があります。

可用性

MQDH 構造体は以下のプラットフォームで利用可能です：

-  AIX
-  IBM i
-  Linux
-  Windows

および、これらのシステムに接続された IBM MQ MQI clients。

形式名

MQFMT_DIST_HEADER

文字セットとエンコード

MQDH 内のデータは、キューマネージャ属性および MQENC_NATIVE で指定 CodedCharSetId されるローカルキューマネージャのエンコーディングによって与えられる文字セットでなければならない。

MQDH の文字セットおよびエンコードは、以下の構造体の CodedCharSetId および Encoding フィールドに設定する必要があります。

- MQMD (MQDH 構造体がメッセージ・データの開始点にある場合)
- MQDH 構造体に先行するヘッダー構造体 (上記以外の場合)

使用法

アプリケーションがディストリビューションリストにメッセージを送信する際、宛先の一部またはすべてがリモートの場合、キューマネージャはアプリケーションメッセージデータに MQXQH および MQDH 構造体を接頭辞として付加し、メッセージに関連する送信キューに配置する。したがって、伝送キューにメッセージが入っている場合は、データの順序は以下のようになります。

- MQXQH 構造
- MQDH 構造体、および、MQOR レコードと MQPMR レコードの配列
- アプリケーション・メッセージ・データ

宛先によっては、キュー・マネージャはこのようなメッセージを複数生成したり、別々の伝送キューに入れたりすることがあります。この場合には、これらのメッセージ内の MQDH 構造体によって、アプリケーションでオープンされている配布リストに定義された宛先ごとのサブセットをそれぞれ識別します。

配布リスト・メッセージを伝送キューに直接書き込むアプリケーションは、上記の順序に従い、MQDH 構造体が正しいことを確認する必要があります。MQDH 構造体が無効な場合、キュー・マネージャで MQPUT または MQPUT1 呼び出しが失敗し、理由コード MQRC_DH_ERROR が戻される可能性があります。

メッセージを配布リスト形式でキューに格納できるのは、そのキューが配布リスト・メッセージをサポートできるように定義した場合だけです。で説明されているキュー **DistLists** 属 [830 ページ](#)の『キューの属性』性を参照してください。配布リストをサポートしないキューにアプリケーションから配布リスト・メッセージを直接書き込んだ場合、キュー・マネージャは配布リストを個別のメッセージに分割し、代わりにそれらのメッセージをキューに入れます。

フィールド

注: 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 493. MQDH のフィールド		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
StrucId (構造識別子)	MQDH_STRUC_ID	'DH--'
バージョン (構造バージョン番号)	MQDH_VERSION_1	1
StrucLength (MQDH 構造の長さ+後続レコード)	なし	0
エンコーディング (MQPMR レコード配列に続くデータの数字エンコーディング)	なし	0
CodedCharSetId (MQPMR レコードの配列に続くデータの文字セット識別子)	MQCCSI_UNDEFINED	0
フォーマット (MQPMR レコード配列に続くデータのフォーマット名)	MQFMT_NONE	ブランク
フラグ (一般的なフラグ)	MQDHF_NONE	0
PutMsgRecFields (MQPMR フィールドの存在を示すフラグ)	MQPMRF_NONE	0
RecsPresent (存在するオブジェクトレコードの数)	なし	0
ObjectRecOffset (MQDH の先頭からの最初のオブジェクトレコードのオフセット)	なし	0

表 493. MQDH のフィールド (続き)

フィールド名と説明	定数の名前	定数の初期値 (存在する場合)
PutMsgRecOffset (最初のプットメッセージレコードの MQDH 先頭からのオフセット)	なし	0

注:

1. 記号-は、単一のブランク文字を表します。
2. C プログラミング言語では、マクロ変数 MQDH_DEFAULT には、表にリストされている値が含まれています。このマクロ変数を以下の方法で使用して、構造体のフィールドに初期値を設定します。

```
MQDH MyDH = {MQDH_DEFAULT};
```

言語ごとの宣言**MQDH の C 宣言**

```
typedef struct tagMQDH MQDH;
struct tagMQDH {
    MQCHAR4  StrucId;           /* Structure identifier */
    MQLONG   Version;          /* Structure version number */
    MQLONG   StrucLength;      /* Length of MQDH structure plus following
                                MQOR and MQPMR records */
    MQLONG   Encoding;         /* Numeric encoding of data that follows
                                the MQOR and MQPMR records */
    MQLONG   CodedCharSetId;   /* Character set identifier of data that
                                follows the MQOR and MQPMR records */
    MQCHAR8  Format;           /* Format name of data that follows the
                                MQOR and MQPMR records */
    MQLONG   Flags;            /* General flags */
    MQLONG   PutMsgRecFields;  /* Flags indicating which MQPMR fields are
                                present */
    MQLONG   RecsPresent;      /* Number of MQOR records present */
    MQLONG   ObjectRecOffset;  /* Offset of first MQOR record from start
                                of MQDH */
    MQLONG   PutMsgRecOffset;  /* Offset of first MQPMR record from start
                                of MQDH */
};
```

MQDH の COBOL 宣言

```
**      MQDH structure
10      MQDH.
**      Structure identifier
15      MQDH-STRUCID      PIC X(4).
**      Structure version number
15      MQDH-VERSION      PIC S9(9) BINARY.
**      Length of MQDH structure plus following MQOR and MQPMR records
15      MQDH-STRUCLNGTH  PIC S9(9) BINARY.
**      Numeric encoding of data that follows the MQOR and MQPMR records
15      MQDH-ENCODING     PIC S9(9) BINARY.
**      Character set identifier of data that follows the MQOR and MQPMR
**      records
15      MQDH-CODEDCHARSETID PIC S9(9) BINARY.
**      Format name of data that follows the MQOR and MQPMR records
15      MQDH-FORMAT       PIC X(8).
**      General flags
15      MQDH-FLAGS        PIC S9(9) BINARY.
**      Flags indicating which MQPMR fields are present
15      MQDH-PUTMSGRECFIELDS PIC S9(9) BINARY.
**      Number of MQOR records present
15      MQDH-RECSPRESENT  PIC S9(9) BINARY.
**      Offset of first MQOR record from start of MQDH
15      MQDH-OBJECTRECOFFSET PIC S9(9) BINARY.
**      Offset of first MQPMR record from start of MQDH
15      MQDH-PUTMSGRECOFFSET PIC S9(9) BINARY.
```

MQDH の PL/I 宣言

```
dc1
1 MQDH based,
3 StrucId          char(4),          /* Structure identifier */
3 Version          fixed bin(31), /* Structure version number */
3 StrucLength      fixed bin(31), /* Length of MQDH structure plus
                                   following MQOR and MQPMR
                                   records */
3 Encoding         fixed bin(31), /* Numeric encoding of data that
                                   follows the MQOR and MQPMR
                                   records */
3 CodedCharSetId   fixed bin(31), /* Character set identifier of data
                                   that follows the MQOR and MQPMR
                                   records */
3 Format           char(8),          /* Format name of data that follows
                                   the MQOR and MQPMR records */
3 Flags           fixed bin(31), /* General flags */
3 PutMsgRecFields  fixed bin(31), /* Flags indicating which MQPMR
                                   fields are present */
3 RecsPresent      fixed bin(31), /* Number of MQOR records present */
3 ObjectRecOffset  fixed bin(31), /* Offset of first MQOR record from
                                   start of MQDH */
3 PutMsgRecOffset  fixed bin(31); /* Offset of first MQPMR record from
                                   start of MQDH */
```

MQDH の Visual Basic 宣言

```
Type MQDH
StrucId          As String*4 'Structure identifier'
Version          As Long     'Structure version number'
StrucLength      As Long     'Length of MQDH structure plus following'
                                   'MQOR and MQPMR records'
Encoding         As Long     'Numeric encoding of data that follows'
                                   'the MQOR and MQPMR records'
CodedCharSetId   As Long     'Character set identifier of data that'
                                   'follows the MQOR and MQPMR records'
Format           As String*8 'Format name of data that follows the'
                                   'MQOR and MQPMR records'
Flags           As Long     'General flags'
PutMsgRecFields  As Long     'Flags indicating which MQPMR fields are'
                                   'present'
RecsPresent      As Long     'Number of MQOR records present'
ObjectRecOffset  As Long     'Offset of first MQOR record from start'
                                   'of MQDH'
PutMsgRecOffset  As Long     'Offset of first MQPMR record from start'
                                   'of MQDH'
End Type
```

MQDH のフィールド詳細

MQDH 配布ヘッダー構造は、メッセージが送信キューに格納された配布リストメッセージである場合に、そのメッセージに含まれる追加データを記述します。配布リスト・メッセージとは、複数の宛先キューに送信されるメッセージです。追加データは、MQDH 構造体、MQOR レコードの配列、MQPMR レコードの配列の順番で構成されています。

フィールド

注: 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 494. MQDH のフィールド		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
<u>StrucId</u> (構造識別子)	MQDH_STRUC_ID	'DH--'
<u>バージョン</u> (構造バージョン番号)	MQDH_VERSION_1	1
<u>StrucLength</u> (MQDH 構造の長さ+後続レコード)	なし	0
<u>エンコーディング</u> (MQPMR レコード配列に続くデータの数字エンコーディング)	なし	0

表 494. MQDH のフィールド (続き)		
フィールド名と説明	定数の名前	定数の初期値 (存在する場合)
<u>CodedCharSetId</u> (MQPMR レコードの配列に続くデータの文字セット識別子)	MQCCSI_UNDEFINED	0
<u>フォーマット</u> (MQPMR レコード配列に続くデータのフォーマット名)	MQFMT_NONE	ブランク
<u>フラグ</u> (一般的なフラグ)	MQDHF_NONE	0
<u>PutMsgRecFields</u> (MQPMR フィールドの存在を示すフラグ)	MQPMRF_NONE	0
<u>RecsPresent</u> (存在するオブジェクトレコードの数)	なし	0
<u>ObjectRecOffset</u> (MQDH の先頭からの最初のオブジェクトレコードのオフセット)	なし	0
<u>PutMsgRecOffset</u> (最初のプットメッセージレコードの MQDH 先頭からのオフセット)	なし	0
注: 1. 記号-は、単一のブランク文字を表します。 2. C プログラミング言語では、マクロ変数 MQDH_DEFAULT には、表にリストされている値が含まれています。このマクロ変数を以下の方法で使用して、構造体のフィールドに初期値を設定します。 <pre>MQDH MyDH = {MQDH_DEFAULT};</pre>		

StrucId (MQCHAR4)

これは、配布ヘッダー構造体の構造体識別子です。常に入力フィールドです。その値は MQDH_STRUC_ID です。

値は次のものでなければなりません。

MQDH_STRUC_ID

配布ヘッダー構造体の識別子。

C プログラミング言語では、定数 MQDH_STRUC_ID_ARRAY も定義されている。これは MQDH_STRUC_ID と同じ値を持ちますが、文字列ではなく文字の配列です。

Version (MQLONG)

値は次のものでなければなりません。

MQDH_VERSION_1

配布ヘッダー構造体のバージョン番号。

以下の定数は、現行バージョンのバージョン番号を指定しています。

MQDH_CURRENT_VERSION

配布ヘッダー構造体の現行バージョン。

フィールドの初期値は、MQDH_VERSION_1 です。

StrucLength (MQLONG)

これは、MQDH 構造体の先頭から、MQOR および MQPMR レコードの配列に続くメッセージ・データの先頭までのバイト数です。データの順序は、以下のとおりです。

- MQDH 構造体

- MQOR レコードの配列
- MQPMR レコードの配列
- メッセージ・データ

MQOR および MQPMR レコードの配列は、MQDH 構造体に含まれるオフセットによってアドレッシングされます。これらのオフセットにより、1 つ以上の MQDH 構造体、レコードの配列、およびメッセージ・データの間に未使用のバイトが生じる場合、それらの未使用のバイトを *StrucLength* の値に含める必要がありますが、それらのバイトの内容はキュー・マネージャーによって保持されません。MQPMR レコードの配列は MQOR レコードの配列より先に処理されても構いません。

このフィールドの初期値は 0 です。

Encoding (MQLONG)

これは、MQOR レコードと MQPMR レコードの配列に続くデータの数値エンコード方式です。これは、MQDH 構造体自体の数値データには適用されません。

MQPUT または MQPUT1 呼び出しでは、アプリケーションは、このフィールドをデータに適切な値に設定する必要があります。

このフィールドの初期値は 0 です。

CodedCharSetId (MQLONG)

これは、MQOR レコードと MQPMR レコードの配列に続くデータの文字セット ID です。これは、MQDH 構造体自体の文字データには適用されません。

MQPUT または MQPUT1 呼び出しでは、アプリケーションは、このフィールドをデータに適切な値に設定する必要があります。次の特殊値を使用することができます。

MQCCSI_INHERIT

この構造体の文字セット ID を継承する。

この構造体の後に続くデータの文字データは、この構造体に設定されているのと同じ文字セットになります。

キュー・マネージャーは、メッセージで送信される構造体の中のこの値を、構造体の実際の文字セット ID に変更します。エラーが発生しない場合、MQGET 呼び出しは、値 MQCCSI_INHERIT を戻しません。

MQCCSI_INHERIT は、MQMD の *PutApplType* フィールドの値が MQAT_BROKER である場合は使用できません。

この値はマルチプラットフォームおよびこれらのシステムに接続されたクライアント IBM MQ でサポートされています。

このフィールドの初期値は MQCCSI_UNDEFINED です。

Format (MQCHAR8)

これは、MQOD レコードと MQPMR レコードの配列の出現後に続くデータの形式名です。

MQPUT または MQPUT1 呼び出しでは、アプリケーションは、このフィールドをデータに適切な値に設定する必要があります。このフィールドのコーディングの規則は、MQMD の *Format* フィールドの場合と同じです。

このフィールドの初期値は MQFMT_NONE です。

フラグ (MQLONG)

次のフラグを指定できます。

MQDHF_NEW_MSG_IDS

配布リスト内の宛先ごとにそれぞれ新しいメッセージ ID を生成します。これは、書き込みメッセージ・レコードが存在しない場合、またはレコードは存在するが *MsgId* フィールドが含まれていない場合にのみ設定してください。

このフラグを使用すると、メッセージ ID の生成が、配布リスト・メッセージが最終的に個々のメッセージに分割されるときまで延期されます。これにより、配布リスト・メッセージを介してやりとりする必要のある制御情報の量を最小限に抑えられます。

アプリケーションによりメッセージが配布リストに書き込まれると、キュー・マネージャーは、以下の記述がどちらも該当する場合に MQDHF を生成し、その中に MQDHF_NEW_MSG_IDS を設定します。

- アプリケーションによって提供された書き込みメッセージ・レコードがないか、提供されたレコードに *MsgId* フィールドが含まれていません。
- MQMD の *MsgId* フィールドが MQMI_NONE であるか、MQPMO の *Options* フィールドに MQPMO_NEW_MSG_ID が含まれます。

フラグが必要ない場合は、次を指定します。

MQDHF_NONE

フラグは指定されていません。MQDHF_NONE は、プログラムの文書化を支援するために定義します。この定数は他の定数と組み合わせて使用するようには意図されていません。ただし、この定数の値はゼロなので、ほかの実数と組み合わせて使用しても、検出されることはありません。

フィールドの初期値は、MQDHF_NONE です。

PutMsgRecFields (MQLONG)

以下のフラグを 0 個以上指定できます。

MQPMRF_MSG_ID

メッセージ ID フィールドがある。

MQPMRF_CORREL_ID

相関 ID フィールドがある。

MQPMRF_GROUP_ID

グループ ID フィールドがある。

MQPMRF_FEEDBACK

フィードバック・フィールドがある。

MQPMRF_ACCOUNTING_TOKEN

会計トークン・フィールドがある。

MQPMR フィールドがない場合は、以下を指定します。

MQPMRF_NONE

書き込みメッセージ・レコードのフィールドがない。MQPMRF_NONE は、プログラムの文書化を支援するために定義します。この定数は他の定数と組み合わせて使用するようには意図されていません。ただし、この定数の値はゼロなので、ほかの実数と組み合わせて使用しても、検出されることはありません。

フィールドの初期値は、MQPMRF_NONE です。

RecsPresent (MQLONG)

これは宛先の数です。配布リストには常に少なくとも 1 つの宛先が含まれていなければならないため、*RecsPresent* は常にゼロより大きくなければなりません。

このフィールドの初期値は 0 です。

ObjectRecOffset (MQLONG)

これには、MQOR オブジェクト・レコードの配列内の最初のレコードのオフセットを宛先キューの名前も含めてバイト単位で指定します。この配列には *RecsPresent* 個のレコードがあります。これらのレコード (および最初のオブジェクト・レコードと前のフィールドの間にスキップされたバイト数) は、*StrucLength* フィールドで指定された長さに含まれます。

配布リストには常に少なくとも 1 つの宛先が含まれていなければならないため、*ObjectRecOffset* は常にゼロより大きくなければなりません。

このフィールドの初期値は 0 です。

PutMsgRecOffset (MQLONG)

これには、MQPMR 書き込みメッセージ・レコードの配列内の最初のレコードのオフセットをメッセージ・プロパティも含めてバイト単位で指定します。存在する場合、この配列には *RecsPresent* 個のレコードがあります。これらのレコード (最初の書き込みメッセージ・レコードと前のフィールドの間にスキップされたバイト数を含む) は、*StrucLength* フィールドに指定された長さに組み込まれます。

書き込みメッセージ・レコードはオプションです。レコードが指定されない場合、*PutMsgRecOffset* はゼロになり、*PutMsgRecFields* の値は MQPMRF_NONE になります。

このフィールドの初期値は 0 です。

MQDLH - 送達不能ヘッダー

MQDLH 構造は、デッドレター (未配達メッセージ) キュー上のメッセージのアプリケーションメッセージデータに付加される情報を記述する。メッセージがデッドレターキューに到達するのは、キューマネージャまたはメッセージチャネルエージェントがメッセージをそのキューにリダイレクトした場合、あるいはアプリケーションがメッセージを直接そのキューに配置した場合のいずれかである。

形式名

MQFMT_DEAD_LETTER_HEADER

文字セットとエンコード

MQDLH 構造体のフィールドは、``および *Encoding* `` *CodedCharSetId* フィールドで指定される文字セットおよびエンコーディングで構成される。これらは MQDLH 構造体の前のヘッダー構造内で指定するか、または MQDLH がアプリケーション・メッセージ・データの先頭にある場合には MQMD 構造体内に指定します。

文字セットは、キュー名に有効な文字用の 1 バイト文字を持つ文字セットでなければなりません。

/ Java 用の IBM MQ クラスを使用している場合、MQMD で定義された JMS コードページが仮想 Java マシンでサポートされていないと、MQDLH は UTF-8 文字セットで書き込まれます。

使用法

メッセージを直接デッドレターキューに配置するアプリケーションは、メッセージデータに MQDLH 構造体をプレフィックスとして付加し、フィールドを適切な値で初期化しなければならない。ただし、キュー・マネージャには、MQDLH 構造体が存在することや、そのフィールドに対して有効な値が指定されていることは、必ずしも必要ではありません。

メッセージがデッドレターキューに収まらないほど長い場合、アプリケーションは次のいずれかを実行する必要があります：

- メッセージデータをデッドレターキューに収まるように切り詰める。
- 補助記憶装置にメッセージを記録し、これを示す例外レポートメッセージをデッドレターキューに配置する。

- メッセージを廃棄して、エラーをその発信元に戻す。廃棄するメッセージが、メッセージ・チャンネル・エージェントが通信チャンネルから受け取ったメッセージなどの重大メッセージである (またはその可能性がある) 場合は、発信元にメッセージのコピーがまだあるときだけ、メッセージを廃棄してください。

上記のいずれの処置が適切であるか (適切なものがある場合) は、アプリケーションの設計によって異なります。

1つのセグメントからなるメッセージが書き込まれ、そのメッセージの前に MQDLH 構造体が指定されていると、キュー・マネージャーでは特殊な処理が実行されます。詳細については、MQMDE 構造体の説明を参照してください。

デッドレターキューへのメッセージの配置

メッセージがデッドレターキューに置かれる場合、MQPUT または MQPUT1 呼び出しに使用される MQMD 構造体は、以下の例外を除き、メッセージに関連付けられた MQMD (通常は MQGET 呼び出しで返される MQMD) と同一でなければならない：

- *CodedCharSetId* および *Encoding* フィールド。これらのフィールドは、MQDLH 構造体にあるフィールドに使用されている文字セットまたはエンコード (それが何であっても) に設定します。
- MQDLH 構造体からデータが始まることを示すために、*Format* フィールドを MQFMT_DEAD_LETTER_HEADER に設定します。
- コンテキスト・フィールド (*AccountingToken*、*ApplIdentityData*、*ApplOriginData*、*PutApplName*、*PutApplType*、*PutDate*、*PutTime*、*UserIdentifier*) は、状況に該当するコンテキスト・オプションを使用して設定します。
 - 先行メッセージと関連性のないメッセージをデッドレターキューに配置するアプリケーションは、MQPMO_DEFAULT_CONTEXT オプションを使用しなければならない。これにより、キューマネージャはメッセージ記述子の全てのコンテキストフィールドをデフォルト値に設定する。
 - 受信したメッセージをデッドレターキューに配置するサーバーアプリケーションは、元のコンテキスト情報を保持するために MQPMO_PASS_ALL_CONTEXT オプションを使用する必要があります。
 - 受信したメッセージへの返信をデッドレターキューに配置するサーバーアプリケーションは、MQPMO_PASS_IDENTITY_CONTEXT オプションを使用しなければならない。これにより識別情報は保持されるが、発信元情報はサーバーアプリケーション自身の情報に設定される。
 - 通信チャンネルから受信したメッセージをデッドレターキューに配置するメッセージチャンネルエージェントは、元のコンテキスト情報を保持するために MQPMO_SET_ALL_CONTEXT オプションを使用しなければならない。

MQDLH 構造体自体の中では、フィールドを以下のように設定してください。

- *CodedCharSetId*、*Encoding*、および *Format* フィールドは、MQDLH 構造体の後にあるデータを記述する値 (通常は、元のメッセージ記述子からの値) に設定します。
- コンテキストフィールド *PutApplType*、*PutApplName*、*PutDate*、および *PutTime* を、メッセージをデッドレターキューに配置するアプリケーションに適した値に設定してください。これらの値は元のメッセージとは関係ありません。
- その他のフィールドは、それぞれ適切な値に設定します。

すべてのフィールドの値が有効であること、および文字フィールドがフィールドの定義長まで空白で埋め込まれていることを確認します。MQDLH 構造体では、キュー・マネージャーがヌル文字とそれに続く文字を空白に変換しません。そのため、ヌル文字を使用して文字データを途中で終了させないでください。

デッドレターキューからのメッセージ取得

デッドレターキューからメッセージを取得するアプリケーションは、メッセージが MQDLH 構造体で始まっていることを検証する必要があります。アプリケーションは、MQDLH 構造体が存在しているかどうかを、メッセージ記述子 MQMD の中の *Format* フィールドを調べることによって判別することができます。つまり、フィールドの値が MQFMT_DEAD_LETTER_HEADER のときは、メッセージ・データは、MQDLH 構造

体から始まります。また、アプリケーションがデッドレターキューから取得するメッセージは、元々の長さがキューの制限を超えていた場合、切り詰められる可能性がある点に注意してください。

フィールド

注：次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 495. MQDLH のフィールド		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
<u>StrucId</u> (構造識別子)	MQDLH_STRUC_ID	'DLH-'
<u>バージョン</u> (構造バージョン番号)	MQDLH_VERSION_1	1
<u>理由</u> (デッドレターキューに理由メッセージが到着)	MQRC_NONE	0
<u>DestQName</u> (元の宛先キュー名)	なし	ヌル・ストリングまたはブランク
<u>DestQMgrName</u> (元の宛先キューマネージャ名)	なし	ヌル・ストリングまたはブランク
<u>エンコーディング</u> (MQDLH に続くデータの数字による符号化)	なし	0
<u>CodedCharSetId</u> (MQDLH に続くデータの文字セット識別子)	MQCCSI_UNDEFINED	0
<u>フォーマット</u> (MQDLH に続くデータのフォーマット名)	MQFMT_NONE	ブランク
<u>PutApplType</u> (メッセージをデッドレターキューに配置するアプリケーションの種類)	なし	0
<u>PutApplName</u> (デッドレターキューにメッセージを配置したアプリケーション名)	なし	ヌル・ストリングまたはブランク
<u>PutDate</u> (メッセージがデッドレターキューに置かれた日付)	なし	ヌル・ストリングまたはブランク
<u>PutTime</u> (メッセージがデッドレターキューに置かれた時刻)	なし	ヌル・ストリングまたはブランク
注： - 記号-は、単一のブランク文字を表します。 - ヌル・ストリングまたはブランクの値は、C 言語ではヌル・ストリングを表し、他のプログラミング言語ではブランク文字を表します。 - C プログラミング言語では、マクロ変数 MQDLH_DEFAULT には、表にリストされている値が含まれています。このマクロ変数を以下の方法で使用して、構造体のフィールドに初期値を設定します。 <pre>MQDLH MyDLH = {MQDLH_DEFAULT};</pre>		

言語ごとの宣言

MQDLH の C 宣言

```
typedef struct tagMQDLH MQDLH;
struct tagMQDLH {
    MQCHAR4    StrucId;          /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
};
```



```

    MQLONG    Reason;          /* Reason message arrived on dead-letter
                                (undelivered-message) queue */
    MQCHAR48  DestQName;       /* Name of original destination queue */
    MQCHAR48  DestQMGrName;    /* Name of original destination queue
                                manager */
    MQLONG    Encoding;        /* Numeric encoding of data that follows
                                MQDLH */
    MQLONG    CodedCharSetId;  /* Character set identifier of data that
                                follows MQDLH */
    MQCHAR8   Format;          /* Format name of data that follows
                                MQDLH */
    MQLONG    PutApplType;     /* Type of application that put message on
                                dead-letter (undelivered-message)
                                queue */
    MQCHAR28  PutApplName;     /* Name of application that put message on
                                dead-letter (undelivered-message)
                                queue */
    MQCHAR8   PutDate;         /* Date when message was put on dead-letter
                                (undelivered-message) queue */
    MQCHAR8   PutTime;         /* Time when message was put on the
                                dead-letter (undelivered-message)
                                queue */
};

```

MQDLH の COBOL 宣言

```

**  MQDLH structure
10  MQDLH.
**  Structure identifier
15  MQDLH-STRUCID          PIC X(4).
**  Structure version number
15  MQDLH-VERSION         PIC S9(9) BINARY.
**  Reason message arrived on dead-letter (undelivered-message) queue
15  MQDLH-REASON          PIC S9(9) BINARY.
**  Name of original destination queue
15  MQDLH-DESTQNAME       PIC X(48).
**  Name of original destination queue manager
15  MQDLH-DESTQMGRNAME    PIC X(48).
**  Numeric encoding of data that follows MQDLH
15  MQDLH-ENCODING        PIC S9(9) BINARY.
**  Character set identifier of data that follows MQDLH
15  MQDLH-CODEDCHARSETID  PIC S9(9) BINARY.
**  Format name of data that follows MQDLH
15  MQDLH-FORMAT          PIC X(8).
**  Type of application that put message on dead-letter
**  (undelivered-message) queue
15  MQDLH-PUTAPPLTYPE     PIC S9(9) BINARY.
**  Name of application that put message on dead-letter
**  (undelivered-message) queue
15  MQDLH-PUTAPPLNAME     PIC X(28).
**  Date when message was put on dead-letter (undelivered-message)
**  queue
15  MQDLH-PUTDATE         PIC X(8).
**  Time when message was put on the dead-letter (undelivered-message)
**  queue
15  MQDLH-PUTTIME         PIC X(8).

```

MQDLH の PL/I 宣言

```

dcl
  1 MQDLH based,
    3 StrucId          char(4),          /* Structure identifier */
    3 Version          fixed bin(31),    /* Structure version number */
    3 Reason           fixed bin(31),    /* Reason message arrived on
                                dead-letter (undelivered-message)
                                queue */
    3 DestQName        char(48),         /* Name of original destination
                                queue */
    3 DestQMGrName     char(48),         /* Name of original destination queue
                                manager */
    3 Encoding         fixed bin(31),    /* Numeric encoding of data that
                                follows MQDLH */
    3 CodedCharSetId   fixed bin(31),    /* Character set identifier of data
                                that follows MQDLH */
    3 Format            char(8),          /* Format name of data that follows
                                MQDLH */
    3 PutApplType      fixed bin(31),    /* Type of application that put
                                message on dead-letter
                                (undelivered-message) queue */

```

3 PutApplName	char(28),	/* Name of application that put message on dead-letter (undelivered-message) queue */
3 PutDate	char(8),	/* Date when message was put on dead-letter (undelivered-message) queue */
3 PutTime	char(8);	/* Time when message was put on the dead-letter (undelivered-message) queue */

MQDLH の高水準アセンブラ宣言

MQDLH	DSECT	
MQDLH_STRUCID	DS CL4	Structure identifier
MQDLH_VERSION	DS F	Structure version number
MQDLH_REASON	DS F	Reason message arrived on dead-letter (undelivered-message) queue
*		
MQDLH_DESTQNAME	DS CL48	Name of original destination queue
MQDLH_DESTQMGRNAME	DS CL48	Name of original destination queue manager
*		
MQDLH_ENCODING	DS F	Numeric encoding of data that follows MQDLH
*		
MQDLH_CODEDCHARSETID	DS F	Character set identifier of data that follows MQDLH
*		
MQDLH_FORMAT	DS CL8	Format name of data that follows MQDLH
MQDLH_PUTAPPLTYPE	DS F	Type of application that put message on dead-letter (undelivered-message) queue
*		
MQDLH_PUTAPPLNAME	DS CL28	Name of application that put message on dead-letter (undelivered-message) queue
*		
MQDLH_PUTDATE	DS CL8	Date when message was put on dead-letter (undelivered-message) queue
*		
MQDLH_PUTTIME	DS CL8	Time when message was put on the dead-letter (undelivered-message) queue
*		
*		
MQDLH_LENGTH	EQU *-MQDLH	
	ORG MQDLH	
MQDLH_AREA	DS CL(MQDLH_LENGTH)	

MQDLH の Visual Basic 宣言

```

Type MQDLH
    StrucId As String*4 'Structure identifier'
    Version As Long 'Structure version number'
    Reason As Long 'Reason message arrived on dead-letter (undelivered-message) queue'
    DestQName As String*48 'Name of original destination queue'
    DestQMgrName As String*48 'Name of original destination queue manager'
    Encoding As Long 'Numeric encoding of data that follows MQDLH'
    CodedCharSetId As Long 'Character set identifier of data that follows MQDLH'
    Format As String*8 'Format name of data that follows MQDLH'
    PutApplType As Long 'Type of application that put message on dead-letter (undelivered-message) queue'
    PutApplName As String*28 'Name of application that put message on dead-letter (undelivered-message) queue'
    PutDate As String*8 'Date when message was put on dead-letter (undelivered-message) queue'
    PutTime As String*8 'Time when message was put on the dead-letter (undelivered-message) queue'
End Type

```

MQDLH のフィールド詳細

MQDLH デッドレターヘッダー構造は、デッドレター（未配達メッセージ）キュー上のメッセージのアプリケーションメッセージデータに付加される情報を記述する。メッセージがデッドレターキューに到達するのは、キューマネージャまたはメッセージチャネルエージェントがメッセージをそのキューにリダイレクトした場合、あるいはアプリケーションがメッセージを直接そのキューに配置した場合のいずれかである。

フィールド

注：次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 496. MQDLH のフィールド

フィールド名と説明	定数の名前	定数の初期値（存在する場合）
StrucId (構造識別子)	MQDLH_STRUC_ID	'DLH'
バージョン (構造バージョン番号)	MQDLH_VERSION_1	1
理由 (デッドレターキューに理由メッセージが到着)	MQRC_NONE	0
DestQName (元の宛先キュー名)	なし	ヌル・ストリングまたはブランク
DestQMgrName (元の宛先キューマネージャ名)	なし	ヌル・ストリングまたはブランク
エンコーディング (MQDLH に続くデータの数字による符号化)	なし	0
CodedCharSetId (MQDLH に続くデータの文字セット識別子)	MQCCSI_UNDEFINED	0
フォーマット (MQDLH に続くデータのフォーマット名)	MQFMT_NONE	ブランク
PutApplType (メッセージをデッドレターキューに配置するアプリケーションの種類)	なし	0
PutApplName (デッドレターキューにメッセージを配置したアプリケーション名)	なし	ヌル・ストリングまたはブランク
PutDate (メッセージがデッドレターキューに置かれた日付)	なし	ヌル・ストリングまたはブランク
PutTime (メッセージがデッドレターキューに置かれた時刻)	なし	ヌル・ストリングまたはブランク

注:

1. 記号-は、単一のブランク文字を表します。
2. ヌル・ストリングまたはブランクの値は、C 言語ではヌル・ストリングを表し、他のプログラミング言語ではブランク文字を表します。
3. C プログラミング言語では、マクロ変数 MQDLH_DEFAULT には、表にリストされている値が含まれています。このマクロ変数を以下の方法で使用して、構造体のフィールドに初期値を設定します。

```
MQDLH MyDLH = {MQDLH_DEFAULT};
```

StrucId (MQCHAR4)

これはデッドレターヘッダー構造の構造識別子です。常に入力フィールドです。その値は MQDLH_STRUC_ID です。

値は次のものでなければなりません。

MQDLH_STRUC_ID

デッドレターヘッダー構造体の識別子。

C プログラミング言語では、定数 MQDLH_STRUC_ID_ARRAY も定義されている。これは MQDLH_STRUC_ID と同じ値を持ちますが、文字列ではなく文字の配列です。

Version (MQLONG)

Version は構造体のバージョン番号です。

値は次のものでなければなりません。

MQDLH_VERSION_1

送達不能ヘッダー構造体のバージョン番号。

以下の定数は、現行バージョンのバージョン番号を指定しています。

MQDLH_CURRENT_VERSION

送達不能ヘッダー構造体の現行バージョン。

フィールドの初期値は、MQDLH_VERSION_1 です。

理由 (MQLONG)

Reason フィールドは、メッセージが、元の宛先キューではなく、送達不能キューに置かれた理由を示しています。

これは、メッセージが、元の宛先キューではなく、送達不能キューに置かれた理由を識別します。理由コードは、MQFB_* または MQRC_* の値のいずれか 1 つ (例えば、MQRC_Q_FULL) でなければなりません。発生し得る一般的な MQFB_* 値の詳細については [385 ページの『MQMD - メッセージ記述子』](#)、フィールド *Feedback* の説明を参照してください。

値が MQFB_IMS_FIRST から MQFB_IMS_LAST の範囲にある場合、実際 IMS のエラーコードは、フィールド *Reason* の値から MQFB_IMS_ERROR を差し引くことで特定できます。

MQFB_* 値の中には、このフィールドにしか出てこないものもあります。これらの値は、送達不能キューに転送されたりポジトリ・メッセージ、トリガー・メッセージ、または伝送キュー・メッセージに関係しています。次のとおりです。

MQFB_APPL_CANNOT_BE_STARTED (X'00000109')

トリガー・メッセージを処理するアプリケーションが、トリガー・メッセージの *ApplId* フィールドで指名されたアプリケーションを開始することができません ([587 ページの『MQTM - トリガー・メッセージ』](#)を参照)。

例えば、z/OS では、CKTI CICS トランザクションがトリガー・メッセージを処理するアプリケーションとなります。

MQFB_APPL_TYPE_ERROR (X'0000010B')

トリガー・メッセージを処理するアプリケーションが、トリガー・メッセージの *ApplType* フィールドが無効であるために、アプリケーションを開始することができません ([587 ページの『MQTM - トリガー・メッセージ』](#)を参照)。

例えば、z/OS では、CKTI CICS トランザクションがトリガー・メッセージを処理するアプリケーションとなります。

MQFB_BIND_OPEN_CLUSRCVR_DEL (X'00000119')

メッセージは、MQOO_BIND_ON_OPEN オプションによって開かれたクラスター・キューに向けられた SYSTEM.CLUSTER.TRANSMIT.QUEUE 上にありましたが、その宛先キューにメッセージを伝送するために使用されるリモート・クラスター受信側チャンネルが、そのメッセージの送信前に削除されました。MQOO_BIND_ON_OPEN が指定されているため、メッセージの伝送には、そのキューが開かれていたときに選択されたチャンネルしか使用できません。このチャンネルが使用可能でなくなったため、このメッセージは送達不能キューに置かれています。

MQFB_NOT_A_REPOSITORY_MSG (X'00000118')

メッセージはリポジトリ・メッセージではない。

MQFB_STOPPED_BY_CHAD_EXIT (X'00000115')

メッセージはチャンネル自動定義出口によって停止された。

MQFB_STOPPED_BY_MSG_EXIT (X'0000010D')

メッセージはチャンネル・メッセージ出口によって停止された。

MQFB_TM_ERROR (X'0000010A')

MQMD の *Format* フィールドには MQFMT_TRIGGER が指定されていますが、メッセージが有効な MQTM 構造体で始まっていません。例えば、*StrucId* 簡略記号目印が有効でないか、*Version* が認識されていない、またはトリガー・メッセージの長さが短すぎて MQTM 構造体が入らない場合などがあります。

例えば、z/OS では、CKTI CICS トランザクションがトリガー・メッセージを処理し、このフィードバック・コードを生成できるアプリケーションとなります。

MQFB_XMIT_Q_MSG_ERROR (X'0000010F')

メッセージ・チャネル・エージェントが、伝送キューにあるメッセージが正しい形式になっていないことを検出しました。メッセージ・チャネル・エージェントは、このフィードバック・コードを使用してメッセージを送達不能キューに書き込みます。

一般的な原因の 1 つは、メッセージが直接伝送キューに入れられたため、メッセージに必要な XQH ヘッダーがないことです。アプリケーションが MQXQH ヘッダーを作成する場合を除き、メッセージはリモート・キューを介して伝送キューに入れる必要があります。

このフィールドの初期値は MQRC_NONE です。

DestQName (MQCHAR48)

DestQName は、メッセージの元の宛先であったメッセージ・キューの名前です。

このフィールドの長さは MQ_Q_NAME_LENGTH によって指定されます。このフィールドの初期値は、C 言語ではヌル・ストリングであり、他のプログラミング言語では 48 桁のブランク文字です。

DestQMgrName (MQCHAR48)

DestQMgrName は、メッセージの元の宛先であったキュー・マネージャーの名前です。

このフィールドの長さは MQ_Q_MGR_NAME_LENGTH で指定します。このフィールドの初期値は、C 言語ではヌル・ストリングであり、他のプログラミング言語では 48 桁のブランク文字です。

Encoding (MQLONG)

Encoding は、MQDLH 構造体のあとに続くデータ (通常は、元のメッセージから取られたデータ) の数値エンコード方式です。これは、MQDLH 構造体自体の数値データには適用されません。

MQPUT または MQPUT1 呼び出しでは、アプリケーションは、このフィールドをデータに適切な値に設定する必要があります。

このフィールドの初期値は 0 です。

CodedCharSetId (MQLONG)

CodedCharSetId は、MQDLH 構造体を通過するデータ (通常は、元のメッセージのデータ) の文字セット ID です。これは、MQDLH 構造体自体の文字データには適用されません。

MQPUT または MQPUT1 呼び出しでは、アプリケーションは、このフィールドをデータに適切な値に設定する必要があります。以下のような特別な値を使用することができます。

MQCCSI_INHERIT

この構造体の後に続くデータの文字データは、この構造体に設定されているのと同じ文字セットになります。

キュー・マネージャーは、メッセージで送信される構造体の中のこの値を、構造体の実際の文字セット ID に変更します。エラーが発生しない限り、値 MQCCSI_INHERIT が MQGET 呼び出しによって返されることはありません。

MQCCSI_INHERIT は、MQMD の *PutApplType* フィールドの値が MQAT_BROKER である場合は使用できません。

この値は、次の環境でサポートされます。

-  AIX
-  IBM i
-  Linux
-  Windows

および、これらのシステムに接続された IBM MQ クライアント。

このフィールドの初期値は MQCCSI_UNDEFINED です。

Format (MQCHAR8)

Format は、MQDLH 構造体のあとに続くデータ (通常は元のメッセージから取られたデータ) の形式名です。

MQPUT または MQPUT1 呼び出しでは、アプリケーションは、このフィールドをデータに適切な値に設定する必要があります。このフィールドのコーディング規則は、MQMD の *Format* フィールドのコーディング規則と同じです。

このフィールドの長さは MQ_FORMAT_LENGTH によって指定されます。このフィールドの初期値は MQFMT_NONE です。

PutApplType (MQLONG)

PutApplType は、送達不能 (未配布メッセージ) キューにメッセージ入れるアプリケーションのタイプです。

このフィールドは、メッセージ記述子 MQMD のフィールド *PutApplType* と同じ意味を持ちます (詳細は [385 ページの『MQMD - メッセージ記述子』](#) 参照)。

キュー・マネージャーがメッセージを送達不能キューに宛先変更する場合は、*PutApplType* の値は、MQAT_QMGR になります。

このフィールドの初期値は 0 です。

PutApplName (MQCHAR28)

PutApplName は、送達不能 (未配布メッセージ) キューにメッセージ入れるアプリケーションの名前です。

名前の形式は、*PutApplType* フィールドによって異なります。形式はリリース間で変わることがあります。フィールド *PutApplName* の説明を参照してください。 [385 ページの『MQMD - メッセージ記述子』](#)

キュー・マネージャーがメッセージを送達不能キューに宛先変更する場合は、*PutApplName* には、キュー・マネージャー名の最初の 28 文字が (必要に応じてブランクが埋め込まれて) 設定されています。

このフィールドの長さは MQ_PUT_APPL_NAME_LENGTH によって指定されます。このフィールドの初期値は、C 言語ではヌル・ストリングですが、その他のプログラミング言語では 28 桁のブランク文字です。

PutDate (MQCHAR8)

PutDate は、メッセージが送達不能 (未配布メッセージ) キューに書き込まれたときの日付です。

キュー・マネージャーがこのフィールドを生成する際に使用する日付の形式は、以下のとおりです。

- YYYYMMDD

文字は、以下のものを表します。

YYYY

年 (4 桁の数字)

MM

月 (01 から 12 まで)

DD

日 (01 から 31 まで)

PutDate および *PutTime* フィールドには、グリニッジ標準時 (GMT) が使用されます。GMT に正確に合わせたシステム・クロックに従います。

このフィールドの長さは `MQ_PUT_DATE_LENGTH` によって指定されます。初期値は、C 言語ではヌル・ストリングであり、他のプログラミング言語では 8 桁のブランク文字です。

PutTime (MQCHAR8)

PutTime は、メッセージが送達不能 (未配布メッセージ) キューに書き込まれた時間です。

キュー・マネージャーがこのフィールドを生成する際に使用する時刻の形式は、以下のとおりです。

- HHMMSSSTH

文字は、以下のものを表します。

HH

時間 (00 から 23 まで)

MM

分 (00 から 59 まで)

SS

秒 (00 から 59 まで。下記の注を参照)

T

10 分の 1 秒 (0 から 9 まで)

H

100 分の 1 秒 (0 から 9 まで)

注: システム・クロックが非常に正確な時間標準に同期している場合は、ごくまれですが、*PutTime* の秒数として 60 または 61 が戻されることがあります。これは、グローバル時間標準にうるう秒が挿入されたときに発生します。

PutDate および *PutTime* フィールドには、グリニッジ標準時 (GMT) が使用されます。GMT に正確に合わせたシステム・クロックに従います。

このフィールドの長さは `MQ_PUT_TIME_LENGTH` によって指定されます。初期値は、C 言語ではヌル・ストリングであり、他のプログラミング言語では 8 桁のブランク文字です。

MQDMHO - メッセージ・ハンドル削除オプション

MQDMHO 構造体を使用すると、アプリケーションで、メッセージ・ハンドルを削除する方法を制御するオプションを指定できます。この構造は、**MQDLTMH** 呼び出しの入力パラメーターです。

文字セットとエンコード

入力データは、アプリケーションの文字セットおよびアプリケーション **MQDMHO** のエンコーディング (**MQENC_NATIVE**) でなければならない。

フィールド

注: 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 497. MQDMHO のフィールド		
フィールド名と説明	定数の名前	定数の初期値 (存在する場合)
<u>StrucId</u> (構造識別子)	MQDMHO_STRUC_ID	'DMHO'
<u>バージョン</u> (構造バージョン番号)	MQDMHO_VERSION_1	1
<u>オプション</u> (オプション)	MQDMHO_NONE	0

表 497. MQDMHO のフィールド (続き)		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
注: 1. C プログラミング言語では、マクロ変数 MQDMHO_DEFAULT には、表にリストされている値が含まれています。この変数を以下の方法で使用すると、構造体のフィールドに初期値を設定できます。 <pre>MQDMHO MyDMHO = {MQDMHO_DEFAULT};</pre>		

言語ごとの宣言

MQDMHO の C 宣言

```
typedef struct tagMQDMHO;
struct tagMQDMHO {
    MQCHAR4    StrucId;          /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQLONG     Options;          /* Options that control the action of MQDLTMH */
};
```

MQDMHO の COBOL 宣言

```
**      MQDMHO structure
10 MQDMHO.
**      Structure identifier
15 MQDMHO-STRUCID      PIC X(4).
**      Structure version number
15 MQDMHO-VERSION     PIC S9(9) BINARY.
**      Options that control the action of MQDLTMH
15 MQDMHO-OPTIONS     PIC S9(9) BINARY.
```

MQDMHO の PL/I 宣言

```
dcl
1 MQDMHO based,
3 StrucId      char(4),          /* Structure identifier */
3 Version      fixed bin(31), /* Structure version number */
3 Options      fixed bin(31), /* Options that control the action of MQDLTMH */
```

MQDMHO の高水準アセンブラ宣言

```
MQDMHO          DSECT
MQDMHO_STRUCID  DS    CL4    Structure identifier
MQDMHO_VERSION  DS    F      Structure version number
MQDMHO_OPTIONS  DS    F      Options that control the action of
*                  MQDLTMH
MQDMHO_LENGTH   EQU    *-MQDMHO
MQDMHO_AREA     DS    CL(MQDMHO_LENGTH)
```

MQDMHO のフィールド詳細

メッセージ **MQDMHO** ハンドル削除オプション構造体は、アプリケーションがメッセージハンドルの削除方法を制御するオプションを指定できるようにします。この構造は、**MQDLTMH** 呼び出しの入力パラメーターです。

フィールド

注: 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 498. MQDMHO のフィールド		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
StrucId (構造識別子)	MQDMHO_STRUC_ID	'DMHO'
バージョン (構造バージョン番号)	MQDMHO_VERSION_1	1
オプション (オプション)	MQDMHO_NONE	0
注： 1. C プログラミング言語では、マクロ変数 MQDMHO_DEFAULT には、表にリストされている値が含まれています。この変数を以下の方法で使用すると、構造体のフィールドに初期値を設定できます。 <pre>MQDMHO MyDMHO = {MQDMHO_DEFAULT};</pre>		

StrucId (MQCHAR4)

これは削除メッセージハンドルオプション構造体の構造体識別子です。常に入力フィールドです。その値は MQDMHO_STRUC_ID です。

値は次のものでなければなりません。

MQDMHO_STRUC_ID

削除メッセージハンドルオプション構造体の識別子。

C プログラミング言語の場合、定数 **MQDMHO_STRUC_ID_ARRAY** も定義されます。これはと同じ値 **MQDMHO_STRUC_ID** を持ちますが、文字列ではなく文字の配列です。

Version (MQLONG)

これは構造体のバージョン番号です。値は次のものでなければなりません。

MQDMHO_VERSION_1

バージョン 1 のメッセージ・ハンドル削除オプション構造。

以下の定数は、現行バージョンのバージョン番号を指定しています。

MQDMHO_CURRENT_VERSION

メッセージ・ハンドル削除オプション構造の現行バージョン。

これは常に入力フィールドです。このフィールドの初期値は **MQDMHO_VERSION_1** です。

Options (MQLONG)

値は次のものでなければなりません。

MQDMHO_NONE

指定されるオプションはありません。

これは常に入力フィールドです。このフィールドの初期値は **MQDMHO_NONE** です。

MQDMPO - メッセージ・プロパティ削除オプション

MQDMPO 構造により、アプリケーションはメッセージのプロパティが削除される方法を制御するオプションを指定できます。この構造体は、MQDLTMP 呼び出しの入力パラメーターです。

文字セットとエンコード

MQDMPO 内のデータは、アプリケーションの文字セットおよびアプリケーションのエンコーディング (MQENC_NATIVE) でなければなりません。

フィールド

注: 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 499. MQDPMO のフィールド		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
StrucId (構造識別子)	MQDPMO_STRUC_ID	'DPMO'
バージョン (構造バージョン番号)	MQDPMO_VERSION_1	1
オプション (MQDPMO の動作を制御するオプション)	MQDLTMP のアクションを制御するオプション	MQDPMO_NONE
注: 1. C プログラミング言語では、マクロ変数 MQDPMO_DEFAULT には、表にリストされている値が含まれています。このマクロ変数を以下の方法で使用して、構造体のフィールドに初期値を設定します。 <pre>MQDPMO MyDPMO = {MQDPMO_DEFAULT};</pre>		

言語ごとの宣言

MQDPMO の C 宣言

```
typedef struct tagMQDPMO MQDPMO;
struct tagMQDPMO {
    MQCHAR4 StrucId;          /* Structure identifier */
    MQLONG  Version;          /* Structure version number */
    MQLONG  Options;          /* Options that control the action of
                               MQDLTMP */
};
```

MQDPMO の COBOL 宣言

```
**      MQDPMO structure
10      MQDPMO.
**      Structure identifier
15      MQDPMO-STRUCID          PIC X(4).
**      Structure version number
15      MQDPMO-VERSION          PIC S9(9) BINARY.
**      Options that control the action of MQDLTMP
15      MQDPMO-OPTIONS          PIC S9(9) BINARY.
```

MQDPMO の PL/I 宣言

```
Dcl
  1 MQDPMO based,
    3 StrucId      char(4),          /* Structure identifier */
    3 Version      fixed bin(31),    /* Structure version number */
    3 Options      fixed bin(31),    /* Options that control the action
                                     of MQDLTMP */
```

MQDPMO の高水準アセンブラ宣言

```
MQDPMO          DSECT
MQDPMO_STRUCID   DS    CL4  Structure identifier
MQDPMO_VERSION   DS    F    Structure version number
MQDPMO_OPTIONS   DS    F    Options that control the
*                  action of MQDLTMP
MQDPMO_LENGTH    EQU    *-MQDPMO
MQDPMO_AREA      DS    CL(MQDPMO_LENGTH)
```

MQDMPO のフィールド詳細

MQDMPO メッセージプロパティ削除オプション構造体により、アプリケーションはメッセージのプロパティが削除される方法を制御するオプションを指定できます。この構造体は、MQDLTMP 呼び出しの入力パラメーターです。

フィールド

注: 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 500. MQDMPO のフィールド		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
StrucId (構造識別子)	MQDMPO_STRUC_ID	'DMP0'
バージョン (構造バージョン番号)	MQDMPO_VERSION_1	1
オプション (MQDMPO の動作を制御するオプション)	MQDLTMP のアクションを制御するオプション	MQDMPO_NONE
注: 1. C プログラミング言語では、マクロ変数 MQDMPO_DEFAULT には、表にリストされている値が含まれています。このマクロ変数を以下の方法で使用して、構造体のフィールドに初期値を設定します。 <pre>MQDMPO MyDMP0 = {MQDMPO_DEFAULT};</pre>		

StrucId (MQCHAR4)

これは削除メッセージプロパティオプション構造体の構造体識別子です。常に入力フィールドです。その値は MQDMPO_STRUC_ID です。

値は次のものでなければなりません。

MQDMPO_STRUC_ID

削除メッセージプロパティオプション構造体の識別子。

C プログラミング言語では、定数 MQDMPO_STRUC_ID_ARRAY も定義されている。これは MQDMPO_STRUC_ID と同じ値を持ちますが、文字列ではなく文字の配列です。

Version (MQLONG)

メッセージプロパティオプション構造体の削除 - バージョンフィールド

これは構造体のバージョン番号です。値は次のものでなければなりません。

MQDMPO_VERSION_1

メッセージ・プロパティ削除オプション構造のバージョン番号。

以下の定数は、現行バージョンのバージョン番号を指定しています。

MQDMPO_CURRENT_VERSION

メッセージ・プロパティ削除オプション構造の現行バージョン。

これは常に入力フィールドです。このフィールドの初期値は、MQDMPO_VERSION_1 です。

Options (MQLONG)

メッセージプロパティオプション構造の削除 - オプションフィールド

位置オプション: 以下は、プロパティ・カーソルと比較したプロパティの相対位置に関するオプションです。

MQDMPO_DEL_FIRST

指定された名前と一致する最初のプロパティに対して削除します。

MQDMPO_DEL_PROP_UNDER_CURSOR

プロパティ・カーソルによって指し示されるプロパティが削除されます。このプロパティは、MQIMPO_INQ_FIRST または MQIMPO_INQ_NEXT オプションのいずれかを使用して最後に照会されたものです。

プロパティ・カーソルは、メッセージ・ハンドルが再使用されるときにリセットされます。また、MQGET 呼び出しの MQGMO 構造体、または MQPUT 呼び出しの MQPMO 構造体の *MsgHandle* フィールドにメッセージ・ハンドルが指定された場合にも、リセットされます。

プロパティ・カーソルがまだ確立されていない時点でこのオプションを使用すると、その呼び出しは完了コード MQCC_FAILED および理由 MQRC_PROPERTY_NOT_AVAILABLE で失敗します。この呼び出しはまた、プロパティ・カーソルが指しているプロパティが既に削除されている場合にも、完了コード MQCC_FAILED および理由 MQRC_PROPERTY_NOT_AVAILABLE で失敗します。

これらのオプションがいずれも必須でない場合は、次のオプションを使用できます。

MQDMPO_NONE

指定されるオプションはありません。

このフィールドは常に入力フィールドです。このフィールドの初期値は、MQDMPO_DEL_FIRST です。

MQEPH - 組み込み PCF ヘッダー

MQEPH 構造体は、メッセージがプログラマブル・コマンド・フォーマット (PCF) メッセージである場合に、そのメッセージ内の追加データを記述します。*PCFHeader* フィールドでは、この構造体続く PCF パラメーターを定義します。これにより、PCF メッセージ・データの後に他のヘッダーを続けることができます。

形式名

MQFMT_EMBEDDED_PCF

文字セットとエンコード

MQEPH 内のデータは、キュー・マネージャ **CodedCharSetId** 属性によって指定される文字セットおよび MQENC_NATIVE によって指定されるローカル・キュー・マネージャのエンコーディングでなければならない。

MQEPH の文字セットとエンコーディングを、MQMD 内の および *Encoding CodedCharSetId* フィールドに設定する (MQEPH 構造体がメッセージデータの先頭にある場合)、または MQEPH 構造体の前にあるヘッダー構造体に設定する (その他のすべてのケース)。

使用法

コマンドをコマンド・サーバーまたは PCF を受け入れるその他のキュー・マネージャーのサーバーに送信するために MQEPH 構造体を使用することはできません。

同様に、コマンド・サーバーまたは PCF を受け入れるその他のキュー・マネージャーのサーバーでも、MQEPH 構造体を含む応答やイベントは生成されません。

フィールド

注: 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 501. MQEPH の MQEPH フィールド		
フィールド名と説明	定数の名前	定数の初期値 (存在する場合)
<u>StrucId</u> (構造識別子)	MQEPH_STRUC_ID	'EPH'

表 501. MQEPH の MQEPH フィールド (続き)		
フィールド名と説明	定数の名前	定数の初期値 (存在する場合)
<u>バージョン</u> (構造バージョン番号)	MQEPH_VERSION_1	1
StrucLength (MQEPH 構造体の長さ+それに続く MQCFH およびパラメータ構造体の長さ)	MQEPH_STRUC_LENGTH_FIXED	68
<u>エンコーディング</u> (最後の PCF パラメータ構造体に続くデータの数字エンコーディング)	なし	0
<u>CodedCharSetId</u> (最後の PCF パラメータ構造体に続くデータの文字セット識別子)	MQCCSI_UNDEFINED	0
<u>フォーマット</u> (最後の PCF パラメータ構造体に続くデータのフォーマット名)	MQFMT_NONE	ブランク
<u>フラグ</u> (フラグ)	MQEPH_NONE	0
<u>PCF ヘッダー</u> (プログラマブルコマンドフォーマット (PCF) ヘッダー)	定義された名前と値 325 ページの表 503	0
注: 1. 記号-は、単一のブランク文字を表します。 2. C プログラミング言語では、マクロ変数 MQEPH_DEFAULT には、表にリストされている値が含まれています。このマクロ変数を以下の方法で使用して、構造体のフィールドに初期値を設定します。 <pre>MQEPH MyEPH = {MQEPH_DEFAULT};</pre>		

言語ごとの宣言

MQEPH 用 C 宣言

```
typedef struct tagMQEPH MQEPH;
struct tagMQDH {
    MQCHAR4  StrucId;           /* Structure identifier */
    MQLONG   Version;          /* Structure version number */
    MQLONG   StrucLength;      /* Total length of MQEPH including the MQCFH
                               and parameter structures that follow it */
    MQLONG   Encoding;         /* Numeric encoding of data that follows last
                               PCF parameter structure */
    MQLONG   CodedCharSetId;    /* Character set identifier of data that
                               follows last PCF parameter structure */
    MQCHAR8   Format;          /* Format name of data that follows last PCF
                               parameter structure */
    MQLONG   Flags;            /* Flags */
    MQCFH     PCFHeader;       /* Programmable command format header */
};
```

COBOL MQEPH に関する宣言

```
**      MQEPH structure
10 MQEPH.
**      Structure identifier
15 MQEPH-STRUCID          PIC X(4).
**      Structure version number
15 MQEPH-VERSION          PIC S9(9) BINARY.
**      Total length of MQEPH structure including the MQCFH
**      and parameter structures that follow it
15 MQEPH-STRUCLength      PIC S9(9) BINARY.
**      Numeric encoding of data that follows last
**      PCF structure
15 MQEPH-ENCODING         PIC S9(9) BINARY.
**      Character set identifier of data that
**      follows last PCF parameter structure
15 MQEPH-CODEDCHARSETID   PIC S9(9) BINARY.
```

```

**      Format name of data that follows last PCF
**      parameter structure
15 MQEPH-FORMAT          PIC X(8).
**      Flags
15 MQEPH-FLAGS          PIC S9(9) BINARY.
**      Programmable command format header
15 MQEPH-PCFHEADER.
**      Structure type
20 MQEPH-PCFHEADER-TYPE          PIC S9(9) BINARY.
**      Structure length
20 MQEPH-PCFHEADER-STRULENGTH    PIC S9(9) BINARY.
**      Structure version number
20 MQEPH-PCFHEADER-VERSION       PIC S9(9) BINARY.
**      Command identifier
20 MQEPH-PCFHEADER-COMMAND       PIC S9(9) BINARY.
**      Message sequence number
20 MQEPH-PCFHEADER-MSGSEQNUMBER  PIC S9(9) BINARY.
**      Control options
20 MQEPH-PCFHEADER-CONTROL       PIC S9(9) BINARY.
**      Completion code
20 MQEPH-PCFHEADER-COMPCODE      PIC S9(9) BINARY.
**      Reason code qualifying completion code
20 MQEPH-PCFHEADER-REASON        PIC S9(9) BINARY.
**      Count of parameter structures
20 MQEPH-PCFHEADER-PARAMETERCOUNT PIC S9(9) BINARY.

```

PL/I MQEPH に関する宣言

```

dcl
1 MQEPH based,
3 StrucId          char(4),          /* Structure identifier */
3 Version          fixed bin(31), /* Structure version number */
3 StrucLength      fixed bin(31), /* Total Length of MQEPH including the
                                   MQCFH and parameter structures that
                                   follow it
3 Encoding          fixed bin(31), /* Numeric encoding of data that follows
                                   last PCF parameter structure
3 CodedCharSetId    fixed bin(31), /* Character set identifier of data that
                                   follows last PCF parameter structure
3 Format            char(8),          /* Format name of data that follows last
                                   PCF parameter structure */
3 Flags            fixed bin(31), /* Flags */
3 PCFHeader,
5 Type             fixed bin(31), /* Structure type */
5 StrucLength      fixed bin(31), /* Structure length */
5 Version          fixed bin(31), /* Structure version number */
5 Command          fixed bin(31), /* Command identifier */
5 MsgseqNumber     fixed bin(31), /* Message sequence number */
5 Control          fixed bin(31), /* Control options */
5 CompCode         fixed bin(31), /* Completion code */
5 Reason           fixed bin(31), /* Reason code qualifying completion code */
5 ParameterCount   fixed bin(31); /* Count of parameter structures */

```

MQEPH の高水準アセンブラ宣言

MQEPH	DSECT	
MQEPH_STRUCID	DS CL4	Structure identifier
MQEPH_VERSION	DS F	Structure version number
MQEPH_STRULENGTH	DS F	Total length of MQEPH including the
*		MQCFH and parameter structures that
		follow it
MQEPH_ENCODING	DS F	Numeric encoding of data that follows
*		last PCF parameter structure
MQEPH_CODEDCHARSETID	DS F	Character set identifier of data that
*		follows last PCF parameter structure
MQEPH_FORMAT	DS CL8	Format name of data that follows last
*		PCF parameter structure
MQEPH_FLAGS	DS F	Flags
MQEPH_PCFHEADER	DS 0F	Force fullword alignment
MQEPH_PCFHEADER_TYPE	DS F	Structure type
MQEPH_PCFHEADER_STRULENGTH	DS F	Structure length
MQEPH_PCFHEADER_VERSION	DS F	Structure version number
MQEPH_PCFHEADER_COMMAND	DS F	Command identifier
MQEPH_PCFHEADER_MSGSEQNUMBER	DS F	Structure length
MQEPH_PCFHEADER_CONTROL	DS F	Control options
MQEPH_PCFHEADER_COMPCODE	DS F	Completion code
MQEPH_PCFHEADER_REASON	DS F	Reason code qualifying completion code
MQEPH_PCFHEADER_PARAMETER COUNT	DS F	Count of parameter structures
MQEPH_PCFHEADER_LENGTH	EQU	*-MQEPH_PCFHEADER

```

MQEPH_PCFHEADER_AREA      ORG  MQEPH_PCFHEADER
*                           DS    CL(MQEPH_PCFHEADER_LENGTH)
MQEPH_LENGTH              EQU  *-MQEPH
                           ORG  MQEPH
MQEPH_AREA                DS    CL(MQEPH_LENGTH)

```

MQEPH の Visual Basic 宣言

```

Type MQEPH
    StrucId      As String*4 'Structure identifier'
    Version      As Long     'Structure version number'
    StrucLength  As Long     'Total length of MQEPH structure including the MQCFH'
                                'and parameter structures that follow it'
    Encoding     As Long     'Numeric encoding of data that follows last'
                                'PCF parameter structure'
    CodedCharSetId As Long   'Character set identifier of data that'
                                'follows last PCF parameter structure'
    Format       As String*8 'Format name of data that follows last PCF'
                                'parameter structure'
    Flags        As Long     'Flags'
    PCFHeader    As MQCFH   'Programmable command format header'
End Type

Global MQEPH_DEFAULT As MQEPH

```

MQEPH のフィールド詳細

MQEPH 組み込み PCF ヘッダー構造は、メッセージがプログラマブルコマンドフォーマット (PCF) メッセージである場合に、そのメッセージ内に存在する追加データを記述する。*PCFHeader* フィールドでは、この構造体続く PCF パラメーターを定義します。これにより、PCF メッセージ・データの後に他のヘッダーを続けることができます。

フィールド

注: 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 502. MQEPH のフィールド		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
StrucId (構造識別子)	MQEPH_STRUC_ID	'EPH'
バージョン (構造バージョン番号)	MQEPH_VERSION_1	1
StrucLength (MQEPH 構造体の長さ+それに続く MQCFH およびパラメータ構造体の長さ)	MQEPH_STRUC_LENGTH_FIXED	68
エンコーディング (最後の PCF パラメータ構造体続くデータの数字エンコーディング)	なし	0
CodedCharSetId (最後の PCF パラメータ構造体続くデータの文字セット識別子)	MQCCSI_UNDEFINED	0
フォーマット (最後の PCF パラメータ構造体続くデータのフォーマット名)	MQFMT_NONE	ブランク
フラグ (フラグ)	MQEPH_NONE	0
PCF ヘッダー (プログラマブルコマンドフォーマット (PCF) ヘッダー)	定義された名前と値 325 ページの表 503	0

表 502. MQEPH のフィールド (続き)		
フィールド名と説明	定数の名前	定数の初期値 (存在する場合)
注: 1. 記号-は、単一のブランク文字を表します。 2. C プログラミング言語では、マクロ変数 MQEPH_DEFAULT には、表にリストされている値が含まれています。このマクロ変数を以下の方法で使用して、構造体のフィールドに初期値を設定します。 <pre>MQEPH MyEPH = {MQEPH_DEFAULT};</pre>		

StrucId (MQCHAR4)

これは、埋め込まれた PCF ヘッダー構造の構造識別子です。常に入力フィールドです。その値は MQEPH_STRUC_ID です。

値は次のものでなければなりません。

MQEPH_STRUC_ID

組み込み PCF ヘッダー構造体の ID。

C プログラミング言語では、定数 MQEPH_STRUC_ID_ARRAY も定義されている。これは MQEPH_STRUC_ID と同じ値を持ちますが、文字列ではなく文字の配列です。

Version (MQLONG)

値は次のものでなければなりません。

MQEPH_VERSION_1

組み込み PCF ヘッダー構造のバージョン番号。

以下の定数は、現行バージョンのバージョン番号を指定しています。

MQCFH_VERSION_3

組み込み PCF ヘッダー構造体の現行バージョン。

このフィールドの初期値は MQEPH_VERSION_1 です。

StrucLength (MQLONG)

これは、次のヘッダー構造体の前に置かれるデータ量です。これには以下のものが含まれます。

- MQEPH ヘッダーの長さ
- ヘッダーのあとに続くすべての PCF パラメーターの長さ
- それらのパラメーターのあとに続くブランクによる埋め込み

StrucLength は 4 の倍数でなければなりません。

構造体の固定長の部分は MQEPH_STRUC_LENGTH_FIXED によって定義されます。

このフィールドの初期値は 68 です。

Encoding (MQLONG)

これは、MQEPH 構造体とそれに関連する PCF パラメーターに続くデータの数値エンコードです。これは、MQEPH 構造体自体の文字データには適用されません。

このフィールドの初期値は 0 です。

CodedCharSetId (MQLONG)

これは、MQEPH 構造体とそれに関連する PCF パラメーターに続くデータの文字セット ID です。これは、MQEPH 構造体自体の文字データには適用されません。

このフィールドの初期値は MQCCSI_UNDEFINED です。

Format (MQCHAR8)

これは、MQEPH 構造体とそれに関連する PCF パラメーターに続くデータの形式名です。

このフィールドの初期値は MQFMT_NONE です。

フラグ (MQLONG)

以下の値を使用できます。

MQEPH_NONE

フラグは指定されていません。MQEPH_NONE は、プログラムの文書化を支援するために定義します。この定数は他の定数と組み合わせて使用するようには意図されていません。ただし、この定数の値はゼロなので、ほかの実数と組み合わせて使用しても、検出されることはありません。

MQEPH_CCSID_EMBEDDED

文字データを含むパラメーターの文字セットが、各構造の CodedCharSetId フィールド内に個々に指定されています。StrucId フィールドと Format フィールドの文字セットは、MQEPH 構造の前にあるヘッダー構造内の CodedCharSetId フィールドで定義されるか、または MQEPH がメッセージの開始点である場合には MQMD 内の CodedCharSetId フィールドで定義されます。

このフィールドの初期値は MQEPH_NONE です。

PCFHeader (MQCFH)

これはプログラマブル・コマンド・フォーマット (PCF) ヘッダーで、MQEPH 構造体の後に続く PCF パラメーターを定義します。これにより、ヘッダーの異なる PCF メッセージ・データを続けることが可能になります。

PCF ヘッダーは、初期の状態では以下の値が定義されています。

表 503. MQCFH のフィールド		
フィールド名	定数の名前	定数の値
Type	MQCFT_NONE	0
StrucLength	MQCFH_STRUC_LENGTH	36
Version	MQCFH_VERSION_3	3
StrucLength	なし	0
Command	MQCMD_NONE	0
MsgSeqNumber	なし	1
Control	MQCFC_LAST	1
CompCode	MQCC_OK	0
Reason	MQRC_NONE	0
ParameterCount	なし	0

アプリケーションは、Type を MQCFT_NONE から有効な構造体タイプに変更し、組み込み PCF ヘッダーを利用できるようにする必要があります。

MQGMO - 読み取りメッセージ・オプション

MQGMO 構造により、アプリケーションはキューからメッセージが削除される方法を制御できます。この構造体は、MQGET 呼び出しの入出力パラメーターです。

バージョン

現在の MQGMO のバージョンは MQGMO_VERSION_4 です。一部のフィールドは、特定のバージョンの MQGMO でのみ使用できます。複数の環境でアプリケーションを移植する必要がある場合は、MQGMO のバージョンがすべての環境で整合していることを確認しなければなりません。特定のバージョンの構造体にはのみ存在するフィールドは、326 ページの『MQGMO - 読み取りメッセージ・オプション』とそのフィールドの説明にその旨記載されています。

サポート対象のプログラミング言語向けに提供されるヘッダー、COPY、および INCLUDE ファイルには、環境でサポートされる最新バージョンの MQGMO が含まれますが、フィールド *Version* の初期値は MQGMO_VERSION_1 に設定されています。バージョン 1 の構造体内に存在しないフィールドを使用するには、*Version* フィールドを必要なバージョンのバージョン番号に設定します。

文字セットとエンコード

MQGMO 内のデータは、キュー・マネージャ **CodedCharSetId** 属性およびローカル・キュー・マネージャのエンコーディング (MQENC_NATIVE で指定) によって指定される文字セットでなければならない。ただし、アプリケーションが MQ MQI クライアントとして実行されている場合、構造体はクライアントの文字セットとエンコードに従っている必要があります。

フィールド

注: 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 504. MQGMO のフィールド		
フィールド名と説明	定数の名前	定数の初期値 (存在する場合)
StrucId (構造識別子)	MQGMO_STRUC_ID	'GMO'
バージョン (構造バージョン番号)	MQGMO_VERSION_1	1
MQGMO - オプションフィールド (MQGET の動作を制御するオプション)	MQGMO_NO_WAIT	0
WaitInterval (待機間隔)	なし	0
Signal1 (信号)	なし	z/OS ではヌル・ポインター、それ以外は 0。
Signal2 (信号識別子)	なし	0
ResolvedQName (解決済み宛先キュー名)	なし	ヌル・ストリングまたはブランク
注: <i>Version</i> が MQGMO_VERSION_2 より小さい場合は、残りのフィールドは無視されます。		
MatchOptions (MQGET で使用される選択基準を制御するオプション)	MQMO_MATCH_MSG_ID + MQMO_MATCH_CORREL_ID	3
GroupStatus (メッセージがグループ内にあるかどうかを示すフラグ)	MQGS_NOT_IN_GROUP	'0'

表 504. MQGMO のフィールド (続き)

フィールド名と説明	定数の名前	定数の初期値 (存在する場合)
<u>SegmentStatus</u> (取得したメッセージが論理メッセージのセグメントであるかどうかを示すフラグ)	MQSS_NOT_A_SEGMENTS	'0'
<u>セグメンテーション</u> (取得したメッセージに対してさらなるセグメンテーションが許可されるかどうかを示すフラグ)	MQSEG_INHIBITED	'0'
<u>Reserved1</u> (予約済み)	なし	'0'
注: <i>Version</i> が MQGMO_VERSION_3 より小さい場合、残りのフィールドは無視されます。		
<u>MsgToken</u> (メッセージトークン)	MQMTOK_NONE	Null
<u>ReturnedLength</u> (返されるメッセージデータのバイト数)	MQRL_UNDEFINED	-1
注: <i>Version</i> が MQGMO_VERSION_4 より小さい場合、残りのフィールドは無視されます。		
<u>Reserved2</u> (予約済み)	なし	'0'
<u>MsgHandle</u> (キューから取得されるメッセージのプロパティで埋められるメッセージへのハンドル)	MQHM_NONE	0
注: 1. 記号-は、単一のブランク文字を表します。 2. ヌル・ストリングまたはブランクの値は、C 言語ではヌル・ストリングを表し、他のプログラミング言語ではブランク文字を表します。 3. C プログラミング言語では、マクロ変数 MQGMO_DEFAULT には、表にリストされている値が含まれています。この変数を以下の方法で使用すると、構造体のフィールドに初期値を設定できます。 <pre>MQGMO MyGMO = {MQGMO_DEFAULT};</pre>		

言語ごとの宣言

MQGMO の C 宣言

```
typedef struct tagMQGMO MQGMO;
struct tagMQGMO {
    MQCHAR4    StructId;        /* Structure identifier */
    MQLONG     Version;         /* Structure version number */
    MQLONG     Options;         /* Options that control the action of */
                                /* MQGET */
    MQLONG     WaitInterval;    /* Wait interval */
    MQLONG     Signal1;         /* Signal */
    MQLONG     Signal2;         /* Signal identifier */
    MQCHAR48    ResolvedQName;  /* Resolved name of destination queue */
    /* Ver:1 */
    MQLONG     MatchOptions;    /* Options controlling selection */
                                /* criteria used for MQGET */
    MQCHAR     GroupStatus;     /* Flag indicating whether message */
                                /* retrieved is in a group */
    MQCHAR     SegmentStatus;   /* Flag indicating whether message */
                                /* retrieved is a segment of a logical */
                                /* message */
    MQCHAR     Segmentation;    /* Flag indicating whether further */
                                /* segmentation is allowed for the */
                                /* message retrieved */
    MQCHAR     Reserved1;       /* Reserved */
    /* Ver:2 */
    MQBYTE16    MsgToken;       /* Message token */
    MQLONG     ReturnedLength;  /* Length of message data returned */
}
```

```

                                (bytes) */
/* Ver:3 */
MQLONG    Reserved2;          /* Reserved */
MQHMSG     MsgHandle;          /* Message handle */
/* Ver:4 */
};

```

注:z/OSでは、*Signal1* フィールドは *PMQLONG* として宣言されます。

MQGMO の COBOL 宣言

```

**  MQGMO structure
10  MQGMO.
**  Structure identifier
15  MQGMO-STRUCID          PIC X(4).
**  Structure version number
15  MQGMO-VERSION          PIC S9(9) BINARY.
**  Options that control the action of MQGET
15  MQGMO-OPTIONS          PIC S9(9) BINARY.
**  Wait interval
15  MQGMO-WAITINTERVAL     PIC S9(9) BINARY.
**  Signal
15  MQGMO-SIGNAL1          PIC S9(9) BINARY.
**  Signal identifier
15  MQGMO-SIGNAL2          PIC S9(9) BINARY.
**  Resolved name of destination queue
15  MQGMO-RESOLVEDQNAME    PIC X(48).
**  Options controlling selection criteria used for MQGET
15  MQGMO-MATCHOPTIONS     PIC S9(9) BINARY.
**  Flag indicating whether message retrieved is in a group
15  MQGMO-GROUPSTATUS      PIC X.
**  Flag indicating whether message retrieved is a segment of a
**  logical message
15  MQGMO-SEGMENTSTATUS    PIC X.
**  Flag indicating whether further segmentation is allowed for the
**  message retrieved
15  MQGMO-SEGMENTATION     PIC X.
**  Reserved
15  MQGMO-RESERVED1        PIC X.
**  Message token
15  MQGMO-MSGTOKEN         PIC X(16).
**  Length of message data returned (bytes)
15  MQGMO-RETURNEDLENGTH   PIC S9(9) BINARY.
**  Reserved
15  MQGMO-RESERVED2        PIC S9(9) BINARY.
**  Message handle
15  MQGMO-MSGHANDLE        PIC S9(18) BINARY.

```

注:z/OSでは、*Signal1* フィールドは *POINTER* として宣言されます。

MQGMO の PL/I 宣言

```

dcl
1  MQGMO based,
3  StrucId          char(4),          /* Structure identifier */
3  Version          fixed bin(31),    /* Structure version number */
3  Options          fixed bin(31),    /* Options that control the action of
                                   MQGET */
3  WaitInterval     fixed bin(31),    /* Wait interval */
3  Signal1          fixed bin(31),    /* Signal */
3  Signal2          fixed bin(31),    /* Signal identifier */
3  ResolvedQName     char(48),        /* Resolved name of destination
                                   queue */
3  MatchOptions     fixed bin(31),    /* Options controlling selection
                                   criteria used for MQGET */
3  GroupStatus       char(1),          /* Flag indicating whether message
                                   retrieved is in a group */
3  SegmentStatus     char(1),          /* Flag indicating whether message
                                   retrieved is a segment of a logical
                                   message */
3  Segmentation      char(1),          /* Flag indicating whether further
                                   segmentation is allowed for the
                                   message retrieved */
3  Reserved1         char(1),          /* Reserved */
3  MsgToken          char(16),         /* Message token */
3  ReturnedLength    fixed bin(31);    /* Length of message data returned
                                   (bytes) */

```

```

3 Reserved2      fixed bin(31); /* Reserved */
3 MsgHandle      fixed bin(63); /* Message handle */

```

注: z/OS では、*Signal1* フィールドは pointer として宣言されます。

MQGMO の高水準アセンブラ宣言

```

MQGMO          DSECT
MQGMO_STRUCID  DS    CL4    Structure identifier
MQGMO_VERSION  DS    F      Structure version number
MQGMO_OPTIONS  DS    F      Options that control the action of
*                               MQGET
MQGMO_WAITINTERVAL DS    F      Wait interval
MQGMO_SIGNAL1  DS    F      Signal
MQGMO_SIGNAL2  DS    F      Signal identifier
MQGMO_RESOLVEDQNAME DS    CL48 Resolved name of destination queue
MQGMO_MATCHOPTIONS DS    F      Options controlling selection criteria
*                               used for MQGET
MQGMO_GROUPSTATUS DS    CL1   Flag indicating whether message
*                               retrieved is in a group
MQGMO_SEGMENTSTATUS DS    CL1   Flag indicating whether message
*                               retrieved is a segment of a logical
*                               message
MQGMO_SEGMENTATION DS    CL1   Flag indicating whether further
*                               segmentation is allowed for the message
*                               retrieved
MQGMO_RESERVED1 DS    CL1   Reserved
MQGMO_MSGTOKEN  DS    XL16   Message token
MQGMO_RETURNEDLENGTH DS    F      Length of message data returned (bytes)
MQGMO_RESERVED2 DS    F      Reserved
MQGMO_MSGHANDLE DS    D      Message handle
MQGMO_LENGTH    EQU    *-MQGMO
MQGMO_AREA      ORG    MQGMO
                DS     CL(MQGMO_LENGTH)

```

MQGMO の高水準アセンブラ宣言

```

Type MQGMO
  StrucId      As String*4    'Structure identifier'
  Version      As Long        'Structure version number'
  Options      As Long        'Options that control the action of MQGET'
  WaitInterval As Long        'Wait interval'
  Signal1      As Long        'Signal'
  Signal2      As Long        'Signal identifier'
  ResolvedQName As String*48  'Resolved name of destination queue'
  MatchOptions As Long        'Options controlling selection criteria'
                                'used for MQGET'
  GroupStatus  As String*1    'Flag indicating whether message'
                                'retrieved is in a group'
  SegmentStatus As String*1    'Flag indicating whether message'
                                'retrieved is a segment of a logical'
                                'message'
  Segmentation As String*1    'Flag indicating whether further'
                                'segmentation is allowed for the message'
                                'retrieved'
  Reserved1    As String*1    'Reserved'
  MsgToken     As MQBYTE16    'Message token'
  ReturnedLength As Long      'Length of message data returned (bytes)'
End Type

```

PROPCTL チャンネルオプション MQGMO

channel **PROPCTL** 属性を使用して、IBM MQ/IBM MQ 10.0 の旧バージョンからパートナーキューマネージャに送信されるメッセージに含まれるメッセージプロパティを制御します。

表 505. チャネル・メッセージ・プロパティ属性の設定	
PROPCTL	説明
ALL	このオプションは、前のバージョンのパートナー・キュー・マネージャーに接続されたアプリケーションが、IBM MQ 10.0 アプリケーションによってメッセージ中に設定されるプロパティを処理できる場合に使用してください。 MQRFH2 に配置される名前と値のペアに加えて、すべてのプロパティがパートナー・キュー・マネージャーに送信されます。 アプリケーション設計に関する次の 2 つの問題を考慮する必要があります。 1. パートナー・キュー・マネージャに接続されたアプリケーションは、別の IBM MQ 10.0 キュー・マネージャで生成されたヘッダー MQRFH2 を含むメッセージを処理できる必要がある。 2. パートナー・キュー・マネージャーに接続されたアプリケーションは、MQPD_SUPPORT_REQUIRED というフラグが付いた新しいメッセージ・プロパティを正しく処理する必要があります。 ALL チャネル・オプションを設定すると、そのチャネルを使用する JMS アプリケーションを、IBM MQ 10.0 とそれ以前のバージョンの間で相互運用することができます。メッセージ・プロパティを使用する新しい IBM MQ 10.0 アプリケーションは、以前のバージョンのアプリケーションが MQRFH2 ヘッダーをどのように処理するかに応じて、以前のバージョンからのアプリケーションとの相互運用が可能です。

表 505. チャネル・メッセージ・プロパティ属性の設定 (続き)	
PROPCTL	説明
COMPAT	このオプションは、以前のバージョンのパートナー・キュー・マネージャーに接続されたアプリケーションに、場合によって (常にではない) メッセージ・プロパティを送信するために使用します。メッセージ・プロパティは、次の 2 つの条件が満たされる場合にのみ送信されます。 1. 処理が必須のメッセージ・プロパティとしてマークされたプロパティが 1 つもないこと。 2. 少なくとも 1 つのメッセージ・プロパティが "reserved" フォルダーにある。注を参照。 COMPAT チャネル・オプションを設定すると、そのチャネルを使用する JMS アプリケーションを、IBM MQ 10.0 とそれ以前のバージョンの間で相互運用することができます。 チャネルは、メッセージ・プロパティを使用するすべてのアプリケーションに使用可能になるわけではありません。予約済みフォルダーを使用するアプリケーションにのみ使用可能になります。メッセージが送信されるか、またはプロパティが送信されるかに関する規則は以下のとおりです。 1. メッセージにプロパティが含まれているが、そのプロパティが "予約済み" フォルダーに関連付けられていない場合、メッセージ・プロパティは送信されません。 2. メッセージ・プロパティが "予約済み" プロパティ・フォルダーに作成されている場合、メッセージ関連付けられているすべてのメッセージ・プロパティが送信されます。ただし、以下の制限があります。 a. メッセージ・プロパティのいずれかが、サポートが必要であるとしてマークされている場合 (MQPD_SUPPORT_REQUIRED または MQPD_SUPPORT_REQUIRED_IF_LOCAL)、メッセージ全体が拒否されます。メッセージのレポート・オプションの値に応じて、メッセージが返されるか、破棄されるか、または送達不能キューに送信されます。 b. サポートが必要であるとしてマークされているメッセージ・プロパティがない場合、個々のプロパティが送信されないことがあります。メッセージ・プロパティ記述子フィールドのいずれかがデフォルト以外の値に設定されている場合、個々のプロパティは送信されません。ただし、メッセージは送信されます。デフォルト以外のプロパティ記述子フィールドの値の例は、MQPD_USER_CONTEXT です。 注: "予約済み" フォルダー名は mcd.、jms.、usr.、または mqext. で始まります。これらのフォルダーは、JMS インターフェースを使用するアプリケーション用に作成されます。IBM MQ 10.0 の場合、これらのフォルダー内に入れられた名前と値のペアは、メッセージ・プロパティとして処理されます。 MQRFH2 ヘッダーに配置された名前と値のペアに加え、メッセージ・プロパティも MQRFH2 ヘッダーで送信されます。MQRFH2 ヘッダーに配置された名前と値のペアは、メッセージが拒否されない限り、送信されます。
NONE	このオプションは、以前のバージョンのパートナー・キュー・マネージャーに接続されたアプリケーションにメッセージ・プロパティが送信されないようにするために使用します。名前と値のペアおよびメッセージ・プロパティが含まれる MQRFH2 は送信されますが、名前と値のペアだけが設定されて送信されます。 チャネル NONE オプションが設定されている場合、メッセージ JMS は JMS メッセージプロパティなしで JMSBytesMessage、または JMSTextMessage として送信されます。IBM MQ 10.0 のアプリケーションで設定されるすべてのプロパティを、以前のバージョンのアプリケーションで無視できる場合は、相互運用が可能です。

PROPCTL キューのオプション MQGMO

PROPCTL キュー属性を使用して、MQGMO メッセージ・プロパティ・オプションの設定なしで MQGET を呼び出すアプリケーションにメッセージ・プロパティを返す方法を制御します。

表 506. キュー・メッセージ・プロパティ属性の設定値

PROPCTL	説明
ALL	ALL オプションを使用することで、同じキューからメッセージを読み取る異なるアプリケーションが、そのメッセージを異なる方法で処理できるようになります。 - 変更せずに以前のバージョンからマイグレーションしたアプリケーションは、引き続き MQRFH2 を直接読み取ることができます。MQRFH2 ヘッダーにあるプロパティには直接アクセスできます。 新しいプロパティや新しいプロパティ属性を処理するためにはアプリケーションを変更する必要があります。MQRFH2 ヘッダーの数やレイアウトの変更によって、アプリケーションが影響を受けることもあります。一部のフォルダ属性が削除される可能性があります。あるいは、以前のバージョンでは無視されていたヘッダー MQRFH2 のレイアウトエラーを報告 IBM MQ する場合があります。 - 新しいまたは変更されたアプリケーションは、メッセージ・プロパティ MQI を使用してメッセージ・プロパティを照会し、MQRFH2 ヘッダーにある名前と値のペアを直接読み取ることができます。 メッセージ内のすべてのプロパティがアプリケーションに返されます。 - アプリケーションが MQCRTMH を呼び出してメッセージ・ハンドルを作成する場合、アプリケーションは MQINQMP を使用してメッセージ・プロパティを照会する必要があります。MQRFH2 からはメッセージ・プロパティが除去され、メッセージ・プロパティではない名前と値のペアが残ります。 - アプリケーションがメッセージ・ハンドルを作成しない場合には、すべてのメッセージ・プロパティおよび名前と値のペアが MQRFH2 に残ります。 ALL は、受信アプリケーションがオプション MQGMO_PROPERTIES を設定していない場合、またはに設定している場合にのみ MQGMO_PROPERTIES_AS_Q_DEF この効果を発揮します。

表 506. キュー・メッセージ・プロパティ属性の設定値 (続き)

PROPCTL	説明
互換性 (既定値)	COMPAT がデフォルト・オプションです。以前のバージョンから変更されていないアプリケーションのように、GMO_PROPERTIES_* が設定されていない場合は、COMPAT として扱われます。デフォルトが COMPAT オプションであるため、明示的に MQRFH2 を作成していなかった以前のバージョンのアプリケーションは、変更を加えなくても IBM MQ 10.0 で機能します。 このオプションは、JMS メッセージを読み取るように作成した以前のバージョンの MQI アプリケーションがある場合に使用します。 - プロパティ MQRFH2JMS はヘッダーに格納され、アプリケーションには MQRFH2 ヘッダーで返されます。フォルダ名はで始まる名前構成 mcd., jms., usr., または mqext されます。 - メッセージに JMS フォルダーが含まれていて、IBM MQ 10.0 アプリケーションがそのメッセージに新しいプロパティ・フォルダーを追加した場合、それらのプロパティも MQRFH2 に入れて返されます。したがって、新しいプロパティや新しいプロパティ属性があれば、それを処理できるようにアプリケーションを変更する必要があります。変更されていないアプリケーションは、MQRFH2 ヘッダーのレイアウトや数の変更による影響を受ける可能性があります。一部のフォルダ属性が削除されている可能性があり、あるいは以前のバージョンでは無視していたヘッダー MQRFH2 のレイアウトにエラーが見つかる IBM MQ 場合があります。 注: このシナリオでは、アプリケーションが接続するキュー・マネージャーが以前のバージョンでも IBM MQ 10.0 でも、アプリケーションの動作は同じです。チャンネルの PROPCTL 属性が COMPAT または ALL に設定されている場合、メッセージに含まれる新しいメッセージ・プロパティはすべて以前のバージョンのパートナー・キュー・マネージャーに送信されます。 - メッセージが JMS メッセージではなく、他のプロパティを含む場合、それらのプロパティは MQRFH2 ヘッダーに指定されたアプリケーションに返されません。¹ - このオプションを使用すると、MQRFH2 を明示的に作成する以前のバージョンのアプリケーションも、多くの場合に正常に動作します。MQRFH2 たとえば、メッセージのプロパティを含む JMSMQI プログラムは、引き続き正常に動作します。作成されるメッセージに JMS メッセージ・プロパティはないものの、他の何らかの MQRFH2 フォルダーが含まれている場合、そのフォルダーはアプリケーションに返されます。フォルダーがメッセージ・プロパティ・フォルダーである場合にのみ、それら特定のフォルダーは MQRFH2 から削除されます。メッセージ・プロパティ・フォルダーは、新しいフォルダー属性 content='properties' を持つか、または <u>定義済みプロパティ・フォルダー名</u> または <u>未グループ化プロパティ・フォルダー名</u> にリストされた名前のフォルダーであることによって識別されます。 - アプリケーションが MQCRTMH を呼び出してメッセージ・ハンドルを作成する場合、アプリケーションは MQINQMP を使用してメッセージ・プロパティを照会する必要があります。MQRFH2 ヘッダーにあるメッセージ・プロパティは削除されます。メッセージ・プロパティではない名前と値のペアは MQRFH2 に残ります。 - アプリケーションが MQCRTMH を呼び出してメッセージ・ハンドルを作成する場合、アプリケーションはメッセージに JMS フォルダーが含まれるかどうかにかかわらず、すべてのメッセージ・プロパティを照会することができます。 - アプリケーションがメッセージ・ハンドルを作成しない場合には、すべてのメッセージ・プロパティおよび名前と値のペアが MQRFH2 に残ります。 メッセージに新しいユーザー・プロパティ・フォルダーが含まれている場合、そのメッセージは新規または変更された IBM MQ 10.0 アプリケーションによって作成されたと推測できます。これらの新しいプロパティを受信アプリケーションが MQRFH2 内で直接処理するには、アプリケーションを ALL オプションを使用するように変更する必要があります。デフォルトの COMPAT オプションが設定されている場合、変更されていないアプリケーションは、プロパティ IBM MQ 10.0 なしで残りの処理を継続 MQRFH2 します。 アプリケーションの開発に関する参照情報 333 PROPCTL インターフェースは、MQRFH2 フォルダーを読み取る古いアプリケーションをサポートすると同時に、新規または変更されたアプリケーションがメッセージ・プロパティ

表 506. キュー・メッセージ・プロパティ属性の設定値 (続き)

PROPCTL	説明
FORCE	FORCE オプションを指定すると、すべてのメッセージ・プロパティが MQRFH2 ヘッダーに配置されます。MQRFH2 ヘッダーのすべてのメッセージ・プロパティおよび名前と値のペアはメッセージに残ります。メッセージ・プロパティは MQRFH2 から削除されず、メッセージ・ハンドルから使用可能になります。FORCE オプションを選択すると、新たにマイグレーションされたアプリケーションが MQRFH2 ヘッダーからメッセージ・プロパティを読み取れるようになります。 例えば、IBM MQ 10.0 メッセージ・プロパティを処理するようにアプリケーションを変更する一方で MQRFH2 ヘッダーを直接処理する機能も残したとします。最初に PROPCTL キュー属性を FORCE に設定しておけば、メッセージ・プロパティを使用するようにアプリケーションを切り替えるタイミングを決定できます。メッセージ・プロパティを使用できるようになったら、PROPCTL キュー属性に別の値を設定します。アプリケーションの新しい機能が予期したとおりに動作しない場合は、PROPCTL オプションの設定を FORCE に戻します。 FORCE は、受信アプリケーションがオプション MQGMO_PROPERTIES を設定していない場合、またはに設定している場合にのみ MQGMO_PROPERTIES_AS_Q_DEF この効果を発揮します。
NONE	既存のアプリケーションがメッセージを処理する際、すべてのメッセージプロパティを無視できるようにするため、また新規または変更されたアプリケーションがメッセージプロパティをクエリできるようにするため、NONE オプションを使用します。 - アプリケーションが MQCRTMH を呼び出してメッセージ・ハンドルを作成する場合、アプリケーションは MQINQMP を使用してメッセージ・プロパティを照会する必要があります。MQRFH2 からはメッセージ・プロパティが除去され、メッセージ・プロパティではない名前と値のペアが残ります。 - アプリケーションがメッセージ・ハンドルを作成しない場合には、MQRFH2 にあるメッセージ・プロパティがすべて削除されます。MQRFH2 ヘッダーの名前と値のペアはメッセージに残ります。 NONE は、受信アプリケーションがオプション MQGMO_PROPERTIES を設定していない場合、またはに設定している場合にのみ MQGMO_PROPERTIES_AS_Q_DEF この効果を発揮します。

¹ IBM MQ classes for JMS によって作成された特定のプロパティ・フォルダーが存在することは、JMS メッセージを示します。プロパティフォルダは mcd., jms., usr., または mqext.

表 506. キュー・メッセージ・プロパティ属性の設定値 (続き)

PROPCTL	説明
V6COMPAT	このオプションは、送信時と同じ形式で MQRFH2 を受け取る場合に使用します。送信アプリケーション (つまりキュー・マネージャー) が追加のメッセージ・プロパティを作成する場合、それらのメッセージはメッセージ・ハンドルで返されます。 このオプションは送信キューと受信キューの両方に、および中継の伝送キューがあればそのキューにも設定する必要があります。このオプションは、キュー名解決パスのキュー定義に設定されている他のすべての PROPCTL オプションをオーバーライドします。 例外的な状況でのみこの V6COMPAT オプションを使用してください。例えば、アプリケーションを以前のバージョンから移行する場合、このオプションは以前のバージョンの IBM MQ 10.0 動作を保持するため有用です。このオプションはメッセージ・スループットに影響することがあります。また、このオプションが送信側キュー、受信側キュー、および中継伝送キューに設定されていることを確認する必要があるため、管理は一層難しくなります。 V6COMPAT この効果は、受信アプリケーションがオプション MQGMO_PROPERTIES を設定していない場合、またはそれをに設定している MQGMO_PROPERTIES_AS_Q_DEF 場合にのみ発生します。

メッセージのプロパティと名前と値のペアに関する詳細については、を参照してください [510 ページの『NameValueData \(MQCHARn\)』](#)。

メッセージプロパティのオプション MQGMO

メッセージプロパティオプション **MQGMO** を使用して、メッセージプロパティがアプリケーションに返される方法を制御します。

表 507. MQGMO メッセージ・プロパティ・オプションの設定値

MQGMO オプション	説明
MQGMO_PROPERTIES_AS_Q_DEF	IBM MQ 同じキューから読み取るアプリケーションは、設定を行わない GMO_PROPERTIES_* 場合、メッセージのプロパティを異なる方法で受信します。IBM MQ メッセージハンドルを作成しないアプリケーションは、キュー PROPCTL 属性によって制御される。IBM MQ アプリケーションは、メッセージ・プロパティを MQRFH2 に入れて受け取るか、それともメッセージ・ハンドルを作成してメッセージ・プロパティを照会するかを選択できます。アプリケーションがメッセージ・ハンドルを作成する場合、MQRFH2 のメッセージ・プロパティは削除されます。 - 新規または変更済みの IBM MQ アプリケーションが GMO_PROPERTIES_* を設定しないかまたは MQGMO_PROPERTIES_AS_Q_DEF に設定している場合に、このアプリケーションはメッセージ・プロパティを照会することを選択できます。メッセージ・ハンドルを作成し、MQINQMP MQI 呼び出しを使用してメッセージ・プロパティを照会する場合は、MQCRTMH を設定する必要があります。 - 新規または変更されたアプリケーションがメッセージハンドルを作成しない場合、受信したメッセージのプロパティは MQRFH2 ヘッダーから直接読み取る必要があります。 - キュー属性 PROPCTL が FORCE に設定されている場合、プロパティはメッセージ・ハンドルに入れて返されません。すべてのプロパティは MQRFH2 ヘッダーに入れて返されます。 - キュー属性 PROPCTL が NONE または COMPAT に設定されている場合、メッセージ・ハンドルを作成する IBM MQ アプリケーションはすべてのメッセージ・プロパティを受け取ります。
MQGMO_PROPERTIES_IN_HANDLE	アプリケーションは強制的にメッセージ・プロパティを使用します。このオプションは、変更済みのアプリケーションがメッセージ・ハンドルの作成に失敗したかどうかを検出するために使用します。アプリケーションは、MQINQMP を呼び出すのではなく、MQRFH2 から直接メッセージ・プロパティを読み取ろうとします。
MQGMO_NO_PROPERTIES	- すべてのプロパティが削除されました。JMS プロパティなど、キュー・マネージャーが生成するプロパティは削除されます。 - メッセージ・ハンドルが作成されても、プロパティは削除されます。他の MQRFH2 フォルダーにある名前と値のペアは、メッセージ・データ内で使用可能です。
MQGMO_PROPERTIES_FORCE_MQRFH2	メッセージ・ハンドルが作成された場合でも、プロパティは MQRFH2 ヘッダーに入れて返されます。 メッセージ・ハンドルが作成された場合でも、MQINQMP はメッセージ・プロパティを返しません。プロパティを照会すると、MQRC_PROPERTY_NOT_AVAILABLE が返されます。

表 507. MQGMO メッセージ・プロパティ・オプションの設定値 (続き)

MQGMO オプション	説明
MQGMO_PROPERTIES_COMPATIBILITY	JMS クライアントからのメッセージの場合、JMS プロパティは MQRFH2 ヘッダーに入れて返されます。メッセージ・ハンドルを作成する新規または変更済みの IBM MQ アプリケーションは異なる動作をします。 - メッセージに mcd., jms., usr., または mqext フォルダーが含まれている場合は、メッセージ・プロパティ・フォルダーにあるすべてのプロパティが返されます。 - メッセージに含まれるプロパティ・フォルダーが mcd., jms., usr., または mqext フォルダーではない場合、メッセージ・プロパティは MQRFH2 に返されません。 - 新規または変更済みの IBM MQ アプリケーションがメッセージ・ハンドルを作成する場合は、MQINQMP MQI 呼び出しを使用してメッセージ・プロパティを照会します。MQRFH2 にあるメッセージ・プロパティはすべて削除されます。 - 新規または変更済みの IBM MQ アプリケーションがメッセージ・ハンドルを作成する場合は、メッセージに含まれるすべてのプロパティを照会できます。メッセージに mcd., jms., usr., または mqext フォルダーが含まれていなくても、すべてのメッセージ・プロパティを照会できます。

関連資料

PROPCTL

2471 (09A7) (RC2471): MQRC_PROPERTY_NOT_AVAILABLE

MQGMO のフィールド詳細

MQGMO の Get-message オプション構造体により、アプリケーションはキューからメッセージが削除される方法を制御できます。この構造体は、MQGET 呼び出しの入出力パラメーターです。

フィールド

注: 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 508. MQGMO のフィールド		
フィールド名と説明	定数の名前	定数の初期値 (存在する場合)
StrucId (構造識別子)	MQGMO_STRUC_ID	'GMO'
バージョン (構造バージョン番号)	MQGMO_VERSION_1	1
MQGMO - オプションフィールド (MQGET の動作を制御するオプション)	MQGMO_NO_WAIT	0
WaitInterval (待機間隔)	なし	0
Signal1 (信号)	なし	z/OS ではヌル・ポインター、それ以外は 0。
Signal2 (信号識別子)	なし	0
ResolvedQName (解決済み宛先キュー名)	なし	ヌル・ストリングまたはブランク

表 508. MQGMO のフィールド (続き)		
フィールド名と説明	定数の名前	定数の初期値 (存在する場合)
注: <i>Version</i> が MQGMO_VERSION_2 より小さい場合は、残りのフィールドは無視されます。		
MatchOptions (MQGET で使用される選択基準を制御するオプション)	MQMO_MATCH_MSG_ID + MQMO_MATCH_CORREL_ID	3
GroupStatus (メッセージがグループ内にあるかどうかを示すフラグ)	MQGS_NOT_IN_GROUP	'-'
SegmentStatus (取得したメッセージが論理メッセージのセグメントであるかどうかを示すフラグ)	MQSS_NOT_A_SEGMENT	'-'
セグメンテーション (取得したメッセージに対してさらなるセグメンテーションが許可されるかどうかを示すフラグ)	MQSEG_INHIBITED	'-'
Reserved1 (予約済み)	なし	'-'
注: <i>Version</i> が MQGMO_VERSION_3 より小さい場合、残りのフィールドは無視されます。		
MsgToken (メッセージトークン)	MQMTOK_NONE	Null
ReturnedLength (返されるメッセージデータのバイト数)	MQRL_UNDEFINED	-1
注: <i>Version</i> が MQGMO_VERSION_4 より小さい場合、残りのフィールドは無視されます。		
Reserved2 (予約済み)	なし	'-'
MsgHandle (キューから取得されるメッセージのプロパティで埋められるメッセージへのハンドル)	MQHM_NONE	0
注: 1. 記号-は、単一のブランク文字を表します。 2. ヌル・ストリングまたはブランクの値は、C 言語ではヌル・ストリングを表し、他のプログラミング言語ではブランク文字を表します。 3. C プログラミング言語では、マクロ変数 MQGMO_DEFAULT には、表にリストされている値が含まれています。この変数を以下の方法で使用すると、構造体のフィールドに初期値を設定できます。 <pre>MQGMO MyGMO = {MQGMO_DEFAULT};</pre>		

StrucId (MQCHAR4)

これは、get メッセージオプション構造体の構造体識別子です。常に入力フィールドです。その値は MQGMO_STRUC_ID です。

値は次のものでなければなりません。

MQGMO_STRUC_ID

get メッセージオプション構造体の識別子。

C プログラミング言語では、定数 MQGMO_STRUC_ID_ARRAY も定義されている。これは MQGMO_STRUC_ID と同じ値を持ちますが、文字列ではなく文字の配列です。

Version (MQLONG)

Version は構造体のバージョン番号です。

値は次のいずれかでなければなりません。

MQGMO_VERSION_1

バージョン 1 の読み取りメッセージ・オプション構造体。

このバージョンはすべての環境でサポートされます。

MQGMO_VERSION_2

バージョン 2 の読み取りメッセージ・オプション構造体。

このバージョンはすべての環境でサポートされます。

MQGMO_VERSION_3

バージョン 3 の読み取りメッセージ・オプション構造体。

このバージョンはすべての環境でサポートされます。

MQGMO_VERSION_4

バージョン 4 の読み取りメッセージ・オプション構造体。

このバージョンはすべての環境でサポートされます。

これより新しいバージョンの構造体にのみ存在するフィールドは、そのフィールドの説明にその旨記載されています。以下の定数は、現行バージョンのバージョン番号を指定しています。

MQGMO_CURRENT_VERSION

読み取りメッセージ・オプション構造体の現行バージョン。

これは常に入力フィールドです。フィールドの初期値は、MQGMO_VERSION_1 です。

Options (MQLONG)

MQGMO オプションは、MQGET のアクションを制御します。オプションは 0 個以上指定できます。複数のオプションの値を指定する必要があるときは、次のようにします。

- 値を加算する (同じ定数は 2 回以上加算しない)
- ビット単位の OR 演算を使用して値を結合する (プログラミング言語がビット演算をサポートしている場合)

無効なオプションの組み合わせについては注記されています。それ以外の組み合わせは有効です。

待機オプション

以下のオプションは、メッセージがキューに到着するまでの待機に関連するオプションです。

MQGMO_WAIT

アプリケーションは、適切なメッセージが到着するまで待機します。アプリケーションが待機する最大時間は、*WaitInterval* に指定されています。

重要: 適切なメッセージが即時に使用可能であれば、待機つまり遅延は生じません。

MQGET 要求が使用禁止になっている場合や、待機中に MQGET 要求が使用禁止になる場合は、待機が取り消されます。キューに適切なメッセージがあるかどうかに関係なく、呼び出しは MQCC_FAILED および理由コード MQRC_GET_INHIBITED で完了します。

または MQGMO_BROWSE_NEXT MQGMO_BROWSE_FIRST オプションと共に MQGMO_WAIT 使用できます。

同じ共有キューでいくつかのアプリケーションが待機している場合は、以下の規則に従い、適切なメッセージが到着したときに活動化されるアプリケーションが選択されます。

表 509. 共有キューにおける MQGET 呼び出しの活動化の規則。		
活動化されるのを待機している MQGET 呼び出しの数		結果
BROWSE オプション付き	BROWSE オプションなし ²	
なし	1 以上	BROWSE オプションなしの 1 つの MQGET 呼び出しが活動化されます。
1 以上	なし	BROWSE オプション付きのすべての MQGET 呼び出しが活動化されます。
1 以上	1 以上	BROWSE オプションなしの 1 つの MQGET 呼び出しが活動化されます。BROWSE オプション付きの活動化される MQGET 呼び出しの数は、予測不能です。

BROWSE オプションなしの複数の MQGET 呼び出しが同じキューで待機している場合は、1 つだけが活動化されます。キュー・マネージャーは、待機中の呼び出しに対して以下の順序で優先順位を与えます。

1. 特定のメッセージによってのみ満たすことができる特定の get-wait 要求。例えば、特定の MsgId または CorrelId (あるいはその両方) を持つ要求。
2. どのメッセージによっても満たすことができる一般的な get-wait 要求。

注：

- 最初のカテゴリー内では、より具体的な get - wait 要求に優先順位を追加することはできません。例えば、MsgId と CorrelId の両方を指定する要求などです。
- いずれのカテゴリーでも、どのアプリケーションが選択されるか予測はできません。特に、待機時間が長いアプリケーションから選択されるとは限りません。
- オペレーティング・システムのパス長および優先順位スケジューリングが考慮された結果、待機中のアプリケーションのうち、予期された優先順位より低いオペレーティング・システムの優先順位を持つアプリケーションがメッセージを取得する場合があります。
- 待機をしていないアプリケーションのほうが、待機中のアプリケーションより優先されてメッセージを取得することもあります。

 z/OS では、以下の点が適用されます。

- メッセージの到着を待機している間に、アプリケーションが他の作業を続行するようにしたい場合は、代わりにシグナル・オプション (MQGMO_SET_SIGNAL) を使用することを検討してください。ただし、シグナル・オプションは環境固有のものなので、異なる環境との間で移植するアプリケーションではこれを使用してはなりません。
- 複数の MQGET 呼び出しが同じメッセージを待機している場合、待機オプションとシグナル・オプションが混在していても、各待機呼び出しは同等に扱われます。MQGMO_WAIT で MQGMO_SET_SIGNAL を指定するとエラーになります。また、未解決のシグナルがあるキュー・ハンドルと共にこのオプションを指定した場合も、エラーになります。
- MQIT_MSG_TOKEN の IndexType を持つキューに対して MQGMO_WAIT または MQGMO_SET_SIGNAL を指定した場合、選択基準は許可されません。つまり、以下のようになります。
 - バージョン 1 の MQGMO を使用している場合は、MQGET 呼び出しで指定されている MQMD のフィールドの MsgId と CorrelId に、それぞれ MQMI_NONE と MQCI_NONE を設定します。
 - version-2 以降 MQGMO を使用している場合は、MatchOptions フィールドを MQMO_NONE に設定します。

² オプション MQGMO_LOCK を指定した呼び出し MQGET は、非ブラウザ呼び出しとして扱われる。

- 共有キューに対する MQGET 呼び出しにおいて、この呼び出しがブラウズ要求またはグループ・メッセージの破壊読み取りである場合、MsgId または CorrelId を使用した突き合わせがいずれも行われなければ、200 ミリ秒後にユーザーのシグナル ECB で MQEC_MSG_ARRIVED が通知されます。

適切なメッセージがキューに到着していない場合でもこれが行われ、やがて待機インターバルが満了すると MQEC_WAIT_INTERVAL_EXPIRED がキューに通知されます。MQEC_MSG_ARRIVED が通知される場合、2 番目の MQGET 呼び出しを再発行することでメッセージを取得する必要があります (入手可能な場合)。

この手法を使うと、タイムリーな方法でメッセージ到着が通知されるようになりますが、非共有キューに対する同様の呼び出しシーケンスと比べて、予期されない処理オーバーヘッドと見なすこともできます。

MQGMO_WAIT は、MQGMO_BROWSE_MSG_UNDER_CURSOR または MQGMO_MSG_UNDER_CURSOR と一緒に指定しても無視されます。エラーは発生しません。

MQGMO_NO_WAIT

適切なメッセージを入手できない場合、アプリケーションは待機しません。MQGMO_NO_WAIT は、MQGMO_WAIT の反対です。MQGMO_NO_WAIT は、プログラム文書化を支援するために定義されています。いずれも指定されていないときは、これがデフォルト値になります。

MQGMO_SET_SIGNAL

このオプションは、Signal1 フィールドおよび Signal2 フィールドとともに使用します。このオプションにより、アプリケーションは、メッセージの到着を待つ間に他の作業の処理を進めることができます。また、アプリケーションは、複数のキューに到着するメッセージを待つことができます (適切なオペレーティング・システム機能が使用可能な場合)。

注: MQGMO_SET_SIGNAL オプションは環境によって異なります。移植するアプリケーションには使用しないでください。

次の 2 つの状況においては、このオプションが指定されていない場合と同様の方法で呼び出しが完了します。

1. 現在使用可能なメッセージが、メッセージ記述子に指定されている基準を満たしている場合。
2. パラメーター・エラーやその他の同期エラーが検出された場合。

メッセージ記述子に指定されている基準を満たすメッセージがどれも現在使用可能でない場合は、メッセージの到着を待たずに制御がアプリケーションに戻されます。**CompCode** および **Reason** パラメーターは、MQCC_WARNING および MQRC_SIGNAL_REQUEST_ACCEPTED に設定されます。その他のメッセージ記述子内の出力フィールド、および MQGET 呼び出しの出力パラメーターは、設定されません。その後、適切なメッセージが到着すると、ECB を通知することによってシグナルが送達されます。

この後、呼び出し元は MQGET 呼び出しを再発行して、メッセージを取得する必要があります。アプリケーションは、オペレーティング・システムが提供する関数を使用して、このシグナルを待機することができます。

オペレーティング・システムが複数待機のメカニズムを提供している場合、そのメカニズムを利用して複数のキューのいずれかにメッセージが到着するのを待機できます。

ゼロ以外の WaitInterval が指定された場合、待機間隔が満了した後にシグナルが送信されます。キュー・マネージャーは待機をキャンセルすることもでき、その場合、シグナルが送達されます。

複数の MQGET 呼び出しが同じメッセージ用のシグナルを設定する場合があります。アプリケーションが活動化される順序は、MQGMO_WAIT で説明されている順序と同じです。

複数の MQGET 呼び出しが同じメッセージを待機している場合、各待機呼び出しは同等に扱われます。複数の呼び出しで、待機オプションとシグナル・オプションが混在していても構いません。

特定の条件下では、MQGET 呼び出しがメッセージを取得し、そのメッセージが到着した結果として生成されたシグナルが送達される場合があります。シグナルが送達される場合、アプリケーションは、使用可能なメッセージがない状態に対する準備ができていなければなりません。

1 つのキュー・ハンドルに、複数の未解決のシグナル要求があってはなりません。

このオプションと組み合わせて使用できないオプションは、以下のとおりです。

- MQGMO_UNLOCK
- MQGMO_WAIT

共有キューに対する MQGET 呼び出しにおいて、この呼び出しが browse 要求またはグループ・メッセージの破壊読み取りである場合、MsgId または CorrelId を使用した突き合わせがいずれも行われなければ、200 ミリ秒後にユーザーのシグナル ECB で MQEC_MSG_ARRIVED が通知されます。

これは、適切なメッセージがキューに到着していない場合でも、待機間隔が満了するまで、キューが MQEC_WAIT_INTERVAL_EXPIRED で通知されるまで発生します。MQEC_MSG_ARRIVED が通知される場合、2 番目の MQGET 呼び出しを再発行することでメッセージを取得する必要があります (入手可能な場合)。

この手法を使うと、タイムリーな方法でメッセージ到着が通知されるようになりますが、非共有キューに対する同様の呼び出しシーケンスと比べて、予期されない処理オーバーヘッドと見なすこともできます。

メッセージの追加頻度が低い場合には、これは効率的なメッセージ取得方法ではありません。browse のケースでこのオーバーヘッドを防ぐには、MQGET 呼び出しで、MsgId (索引なし、または MsgId による索引付けの場合) あるいは CorrelId (CorrelId による索引付けの場合) を突き合わせるように指定してください。

 このオプションは z/OS でのみサポートされます。

MQGMO_FAIL_IF QUIESCING

キュー・マネージャーが静止状態にある場合、MQGET 呼び出しを強制的に失敗させます。

 このオプションは z/OS、接続 (``CICS または``IMS``アプリケーション用) が休止状態にある場合にも、``呼び出し MQGET を強制的に失敗させます。

このオプションが MQGMO_WAIT または MQGMO_SET_SIGNAL とともに指定され、キュー・マネージャーが静止状態になった時点で待機またはシグナルが未解決である場合は、以下のようになります。

- 待機は取り消され、呼び出しは完了コード MQCC_FAILED と理由コード MQRC_Q_MGR QUIESCING または MQRC_CONNECTION QUIESCING を戻します。
- シグナルが、環境固有のシグナル完了コードと共に取り消されます。

 z/OS では、シグナルはイベント完了コード MQEC_Q_MGR QUIESCING または MQEC_CONNECTION QUIESCING で完了します。

MQGMO_FAIL_IF QUIESCING が指定されず、キュー・マネージャーまたは接続が静止状態に入った場合、待機またはシグナルは取り消されません。

同期ポイントのオプション

以下のオプションは、作業単位内での MQGET 呼び出しの参加に関連したオプションです。

MQGMO_SYNCPOINT

この要求は、通常の作業単位プロトコルの中で操作することです。メッセージには、他のアプリケーションでは使用できないものとしてマークが付けられますが、作業単位がコミットされたときにのみ、キューから削除されます。作業単位がバックアウトされると、メッセージは再び使用可能になります。

MQGMO_SYNCPOINT および MQGMO_NO_SYNCPOINT を設定解除したままにすることができます。その場合、get 要求の作業単位プロトコルへの組み込みは、キュー・マネージャーの実行環境によって決定されます。アプリケーションの実行環境によって決定されるものではありません。

-  z/OS では、get 要求は作業単位内にあります。
- z/OS 以外のすべての環境では、get 要求は作業単位内にありません。

このような違いがあるため、移植するアプリケーションでは、このオプションをデフォルトにすることはできません。MQGMO_SYNCPOINT または MQGMO_NO_SYNCPOINT を明示的に指定してください。

このオプションと組み合わせて使用できないオプションは、以下のとおりです。

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_BROWSE_NEXT
- MQGMO_LOCK
- MQGMO_NO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT
- MQGMO_UNLOCK

MQGMO_SYNCPOINT_IF_PERSISTENT

この要求は、取り出されたメッセージが持続する場合に限り、標準の作業単位プロトコル内で機能します。持続メッセージは、MQMD の Persistence フィールドに値 MQPER_PERSISTENT を持ちます。

- メッセージが持続メッセージの場合、キュー・マネージャーは、アプリケーションが MQGMO_SYNCPOINT を指定したかのように呼び出しを処理します。
- メッセージが持続メッセージでない場合、キュー・マネージャーは、アプリケーションが MQGMO_NO_SYNCPOINT を指定したかのように呼び出しを処理します。

このオプションと組み合わせて使用できないオプションは、以下のとおりです。

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_BROWSE_NEXT
- MQGMO_COMPLETE_MSG
- MQGMO_MARK_SKIP_BACKOUT
- MQGMO_NO_SYNCPOINT
- MQGMO_SYNCPOINT
- MQGMO_UNLOCK

このオプションは、次の環境でサポートされます。

-  AIX
-  IBM i
-  Linux
-  z/OS

および、これらのシステムに接続された IBM MQ MQI clients。

MQGMO_NO_SYNCPOINT

この要求は、通常の作業単位プロトコルの外部で動作することになります。ブラウズ・オプションなしのメッセージを受け取った場合、メッセージは直ちにキューから削除されます。作業単位をバックアウトすることによって、メッセージを再度使用可能にすることはできません。

このオプションは、MQGMO_BROWSE_FIRST または MQGMO_BROWSE_NEXT を指定した場合に想定されます。

MQGMO_SYNCPOINT および MQGMO_NO_SYNCPOINT を設定解除したままにすることができます。その場合、get 要求の作業単位プロトコルへの組み込みは、キュー・マネージャーの実行環境によって決定されます。アプリケーションの実行環境によって決定されるものではありません。

-  z/OS では、get 要求は作業単位内にあります。
- z/OS 以外のすべての環境では、get 要求は 作業単位内にありません。

このような違いがあるため、移植するアプリケーションでは、このオプションをデフォルトにすることはできません。MQGMO_SYNCPOINT または MQGMO_NO_SYNCPOINT のいずれかを明示的に指定してください。

このオプションと組み合わせて使用できないオプションは、以下のとおりです。

- MQGMO_MARK_SKIP_BACKOUT
- MQGMO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT

MQGMO_MARK_SKIP_BACKOUT

このオプションでマークしたメッセージをキュー上に復元せずに、作業単位をバックアウトします。

このオプションは z/OS でのみサポートされます。

このオプションを指定する場合は、MQGMO_SYNCPOINT も指定する必要があります。

MQGMO_MARK_SKIP_BACKOUT は、以下のいずれかのオプションでは無効です。

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_BROWSE_NEXT
- MQGMO_LOCK
- MQGMO_NO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT
- MQGMO_UNLOCK

注：IMS および CICS では、MQGMO_MARK_SKIP_BACKOUT でマークされたメッセージを含む作業単位をバックアウトした後に、追加の IBM MQ 呼び出しを発行しなければならない場合があります。このマークが付いたメッセージを含む新しい作業単位をコミットする前に、IBM MQ 呼び出しを実行する必要があります。呼び出しは、任意の IBM MQ 呼び出しにすることができます。

1. IMS で、IMS APAR PN60855 をまだ適用しておらず、IMS MPP または BMP アプリケーションを実行している場合。
2. CICS で、任意のアプリケーションを実行している場合。

どちらの場合でも、バックアウトされたメッセージを含む新しい作業単位をコミットする前に、任意の IBM MQ 呼び出しを実行します。

注：1 つの作業単位内では、バックアウトのスキップとしてマークできる get 要求は 1 つだけです。マークしない get 要求は複数あっても、1 つもなくとも構いません。

アプリケーションが作業単位をバックアウトした場合、MQGMO_MARK_SKIP_BACKOUT を使用してリトリブされたメッセージは、前の状態に復元されません。その他のリソース更新はバックアウトされます。メッセージは、バックアウト要求によって開始された新しい作業単位の中で取得されたかのように扱われます。メッセージは、MQGMO_MARK_SKIP_BACKOUT オプションを指定せずに検索されます。

MQGMO_MARK_SKIP_BACKOUT は、いくつかのリソースが変更された後で、作業単位が正常に完了できないことが明らかになった場合に役立ちます。このオプションを省略した場合、作業単位をバックアウトするとメッセージがキューに復元されます。そのメッセージが次に取得された時に、同じ一連のイベントが再度発生します。

しかし、MQGMO_MARK_SKIP_BACKOUT オプションを元の MQGET 呼び出しで指定した場合、作業単位をバックアウトすると、他のリソースに対する更新もバックアウトされます。メッセージは、新しい作業単位のもとで取得されたかのように扱われます。アプリケーションは適切なエラー処理を実行することができます。元のメッセージの送信者にレポート・メッセージを送信したり、送達不能キューに元のメッセージを入れたりすることができます。その後、アプリケーションは新しい作業単位をコミットすることができます。新しい作業単位をコミットすると、メッセージは元のキューから永久的に削除されます。

MQGMO_MARK_SKIP_BACKOUT は、単一の物理メッセージにマークを付けます。メッセージがメッセージ・グループに所属する場合、グループ内の他のメッセージはマークされません。同様に、マークされたメッセージが論理メッセージのセグメントである場合、論理メッセージ中の他のセグメントはマークされません。

グループ内のどのメッセージにもマークを付けることができますが、MQGMO_LOGICAL_ORDER を使用してメッセージを取得する場合は、グループ内の最初のメッセージにマークを付けると便利です。作業単位がバックアウトされると、最初の (マークされた) メッセージが新しい作業単位に移動されます。グループ内の 2 番目以降のメッセージはキューに復元されます。キューに残っているメッセージは、MQGMO_LOGICAL_ORDER を使用して別のアプリケーションで検索することはできません。グループ内の最初のメッセージは、既にキューに存在しなくなっています。ただし、作業単位をバックアウトしたアプリケーションは、MQGMO_LOGICAL_ORDER オプションを使用して、2 番目以降のメッセージを新しい作業単位に取り出すことができます。最初のメッセージは既に取得されています。

場合によっては、新しい作業単位をバックアウトしなければならないことがあります。例えば、送達不能キューがいっぱいだが、メッセージを廃棄してはならない場合などです。新しい作業単位のバックアウトにより、元のキューのメッセージが復元され、メッセージが失われるのを防ぐことができます。しかし、この状態では処理を続行できません。新しい作業単位をバックアウトした後、アプリケーションは、オペレーターまたは管理者に、リカバリー不能エラーがあることを通知してから終了する必要があります。

MQGMO_MARK_SKIP_BACKOUT は、get 要求を含む作業単位が、それをバックアウトするアプリケーションによって中断された場合にのみ機能します。トランザクションまたはシステムに障害が発生したために、取得要求を含む作業単位がバックアウトされた場合、MQGMO_MARK_SKIP_BACKOUT は無視されます。このオプションを使用して取得されたメッセージはすべて、このオプションなしで取得されたメッセージと同じように、キューで復元されます。

オプションを閲覧する

以下のオプションは、キュー上のメッセージのブラウズに関連したオプションです。

MQGMO_BROWSE_FIRST

キューが MQGMO_BROWSE オプション付きでオープンされる場合、ブラウズ・カーソルが設定され、論理的にキューの最初のメッセージの前に配置されます。その後 MQGET 呼び出しを使用するときに MQGMO_BROWSE_FIRST、MQGMO_BROWSE_NEXT、または MQGMO_BROWSE_MSG_UNDER_CURSOR オプションを指定すると、キューからメッセージを非破壊的に取得することができます。ブラウズ・カーソルは、キュー内のメッセージの中で、MQGMO_BROWSE_NEXT を指定した次の MQGET 呼び出しが適切なメッセージの検索を始める位置をマークします。

MQGMO_BROWSE_FIRST は、以下のいずれかのオプションでは無効です。

- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_BROWSE_NEXT
- MQGMO_MARK_SKIP_BACKOUT
- MQGMO_MSG_UNDER_CURSOR
- MQGMO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT
- MQGMO_UNLOCK

キューがブラウズ用にオープンされていなかった場合もエラーになります。

MQGMO_BROWSE_FIRST を指定した MQGET 呼び出しでは、以前のブラウズ・カーソルの位置は無視されます。メッセージ記述子に指定された条件を満たすキュー上の最初のメッセージが検索されます。メッセージはキューに残り、ブラウズ・カーソルはこのメッセージの上に置かれます。

この呼び出しの後、ブラウズ・カーソルは、戻されたメッセージ上に置かれます。MQGMO_BROWSE_NEXT を指定した MQGET 呼び出しが次に出される前に、キューからメッセージが削除される場合があります。その場合は、そのキューの中で削除されたメッセージが入っていた位置に (現在そこが空であっても) ブラウズ・カーソルが残ります。

この MQGMO_MSG_UNDER_CURSOR オプションを非ブラウザの MQGET 呼び出しで使用すると、キューからメッセージを削除することになります。

ブラウザ・カーソルは、たとえ同じ *Hobj* ハンドルを使用していたとしても、非ブラウザの MQGET 呼び出しによっては移動されません。また、ブラウザの MQGET 呼び出しで完了コード MQCC_FAILED または理由コード MQRC_TRUNCATED_MSG_FAILED が戻された場合も移動されません。

ブラウザされるメッセージをロックするには、このオプションとともに MQGMO_LOCK オプションを指定します。

MQGMO_BROWSE_FIRST は、MQGMO_*オプションと MQMO_*オプションの任意の有効な組み合わせで指定できます。これらのオプションは、グループ内のメッセージおよび論理メッセージのセグメントの処理を制御します。

MQGMO_LOGICAL_ORDER を指定すると、メッセージは論理順序でブラウザされます。このオプションを省略すると、メッセージは物理順序でブラウザされます。MQGMO_BROWSE_FIRST を指定すると、論理順序と物理順序を切り替えることができます。後続の MQGET 呼び出しで MQGMO_BROWSE_NEXT を使用すると、キュー・ハンドル用に MQGMO_BROWSE_FIRST を指定した最後の呼び出しと同じ順序でキューをブラウザします。

キュー・マネージャーは、MQGET 呼び出し用にグループおよびセグメント情報を 2 セット保持します。ブラウザ呼び出し用のグループおよびセグメント情報は、キューからメッセージを削除する呼び出しの情報とは別個に保持されています。MQGMO_BROWSE_FIRST を指定すると、キュー・マネージャーはブラウザ用のグループおよびセグメント情報を無視します。キュー・マネージャーは、現行のグループも現行の論理メッセージも存在していない場合と同じようにキューをスキャンします。MQGET 呼び出しが成功すると、完了コード MQCC_OK または MQCC_WARNING が戻され、ブラウザ用のグループおよびセグメント情報は、戻されたメッセージの情報に設定されます。呼び出しが失敗した場合、ブラウザ用のグループおよびセグメント情報は、呼び出しが行われる前の状態から変更されません。

MQGMO_BROWSE_NEXT

MQGET 呼び出しで指定された選択基準を満たす、キュー上の次のメッセージにブラウザ・カーソルを進めます。メッセージはアプリケーションに戻されますが、キューに残ります。

MQGMO_BROWSE_NEXT は、以下のいずれかのオプションでは無効です。

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_MARK_SKIP_BACKOUT
- MQGMO_MSG_UNDER_CURSOR
- MQGMO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT
- MQGMO_UNLOCK

キューがブラウザ用にオープンされていなかった場合もエラーになります。

MQGMO_BROWSE_NEXT は、キューがブラウザ用にオープンされた後に、それがキューをブラウザするための最初の呼び出しである場合、MQGMO_BROWSE_FIRST と同じように動作します。

MQGMO_BROWSE_NEXT を指定した MQGET 呼び出しが次に出される前に、カーソル下のメッセージがキューから削除される場合があります。そのキューの中で削除されたメッセージが入っていた位置に (現在そこが空であっても) ブラウズ・カーソルが論理的には残ります。

メッセージは、以下の 2 つの方法のいずれかによってキューに入れられます。

- 優先順位内の FIFO (MQMDS_PRIORITY)、または
- 優先順位に関係ない FIFO (MQMDS_FIFO)

MsgDeliverySequence キュー属性は、適用されるメソッドを示します (詳しくは、[830 ページの『キューの属性』](#)を参照してください)。

キューの `MsgDeliverySequence` が `MQMDS_PRIORITY` である可能性があります。メッセージは、ブラウザ・カーソルが現在示しているキューより高い優先順位のキューに到着します。この場合、`MQGMO_BROWSE_NEXT` を使用するキューの現在のスweep中に、より高い優先順位のメッセージは検出されません。これは、ブラウザ・カーソルが `MQGMO_BROWSE_FIRST` でリセットされた後、またはキューを再オープンすることによってのみ検出できます。

必要に応じて、この `MQGMO_MSG_UNDER_CURSOR` オプションを非ブラウザの `MQGET` 呼び出しで使用し、キューからメッセージを削除することができます。

ブラウザ・カーソルは、同じ `Hobj` ハンドルを使用している非ブラウザの `MQGET` 呼び出しによっては移動されません。

ブラウザされるメッセージをロックするには、このオプションとともに `MQGMO_LOCK` オプションを指定します。

`MQGMO_BROWSE_NEXT` は、`MQGMO_*`オプションと `MQMO_*`オプションの任意の有効な組み合わせで指定できます。これらのオプションは、グループ内のメッセージおよび論理メッセージのセグメントの処理を制御します。

`MQGMO_LOGICAL_ORDER` を指定すると、メッセージは論理順序でブラウザされます。このオプションを省略すると、メッセージは物理順序でブラウザされます。`MQGMO_BROWSE_FIRST` を指定すると、論理順序と物理順序を切り替えることができます。後続の `MQGET` 呼び出しで `MQGMO_BROWSE_NEXT` を使用すると、キュー・ハンドル用に `MQGMO_BROWSE_FIRST` を指定した最後の呼び出しと同じ順序でキューをブラウザします。この条件が満たされない場合、呼び出しは失敗し、理由コード `MQRC_INCONSISTENT_BROWSE` が戻ります。

注: `MQGMO_LOGICAL_ORDER` を指定していないときに、`MQGET` 呼び出しを使用して1つのメッセージ・グループの最後より先をブラウザする場合には、特に注意してください。例えば、キュー上のグループ内の最後のメッセージが、グループ内の最初のメッセージより前に来ているとします。`MQGMO_BROWSE_NEXT` を使用してグループの終わりを超えてブラウザし、`MsgSeqNumber` を1に設定して `MQMO_MATCH_MSG_SEQ_NUMBER` を指定すると、既にブラウザされているグループの最初のメッセージが戻されます。これは、即時に発生する可能性があります。あるいは、介入グループがある場合、何度かの `MQGET` 呼び出し後に発生するかもしれません。グループ内にない論理メッセージについても、同じ考慮事項が適用されます。

ブラウザ呼び出し用のグループおよびセグメント情報は、キューからメッセージを削除する呼び出しの情報とは別個に保持されています。

MQGMO_BROWSE_MSG_UNDER_CURSOR

`MQGMO` の `MatchOptions` フィールドに指定された `MQMO_*`オプションに関係なく、ブラウザ・カーソルによって指示されたメッセージを非破壊的に取り出します。

`MQGMO_BROWSE_MSG_UNDER_CURSOR` は、以下のいずれかのオプションでは無効です。

- `MQGMO_BROWSE_FIRST`
- `MQGMO_BROWSE_NEXT`
- `MQGMO_MARK_SKIP_BACKOUT`
- `MQGMO_MSG_UNDER_CURSOR`
- `MQGMO_SYNCPOINT`
- `MQGMO_SYNCPOINT_IF_PERSISTENT`
- `MQGMO_UNLOCK`

キューがブラウザ用にオープンされていなかった場合もエラーになります。

ブラウザ・カーソルによって示されるメッセージは、`MQGMO_BROWSE_FIRST` オプションまたは `MQGMO_BROWSE_NEXT` オプションのいずれかを使用して最後に取得されたメッセージです。このキューがオープンされて以降、これらの呼び出しのいずれも発行されていない場合、呼び出しは失敗します。また、ブラウザ・カーソルの下にあったメッセージが破壊的に取得されている場合も、呼び出しは失敗します。

ブラウザ・カーソルの位置は、この呼び出しでは変更されません。

この MQGMO_MSG_UNDER_CURSOR オプションを非ブラウザの MQGET 呼び出しで使用し、キューからメッセージを削除することができます。

ブラウザ・カーソルは、たとえ同じ Hobj ハンドルを使用していたとしても、非ブラウザの MQGET 呼び出しによっては移動されません。また、ブラウザの MQGET 呼び出しで完了コード MQCC_FAILED または理由コード MQRC_TRUNCATED_MSG_FAILED が戻された場合も移動されません。

MQGMO_BROWSE_MSG_UNDER_CURSOR が MQGMO_LOCK とともに指定されている場合:

- 既にロックされているメッセージがある場合、そのメッセージがカーソルの下になければなりません。ロック解除や再度ロックをしないで戻されるようにするためです。メッセージは、ロックされたままになります。
- ロックされたメッセージがなく、かつブラウザ・カーソルの下にメッセージがある場合、そのメッセージはロックされ、アプリケーションに戻されます。ブラウザ・カーソルの下にメッセージがない場合、呼び出しは失敗します。

MQGMO_LOCK を指定せずに MQGMO_BROWSE_MSG_UNDER_CURSOR を指定した場合:

- 既にロックされているメッセージがある場合、そのメッセージがカーソルの下になければなりません。メッセージはアプリケーションに戻され、その後ロック解除されます。メッセージはロック解除されているので、メッセージをもう一度ブラウザできない場合や、同じアプリケーションから破壊的に取得できない場合があります。キューからメッセージを取得する別のアプリケーションによって、破壊的に取得されている場合があるからです。
- ロックされたメッセージがなく、ブラウザ・カーソルの下にメッセージがある場合、そのメッセージがアプリケーションに戻されます。ブラウザ・カーソルの下にメッセージがない場合、呼び出しは失敗します。

MQGMO_COMPLETE_MSG が MQGMO_BROWSE_MSG_UNDER_CURSOR とともに指定されている場合、ブラウザ・カーソルは、MQMD 内の Offset フィールドがゼロであるメッセージを識別する必要があります。この条件が満たされない場合、呼び出しは失敗し、理由コード MQRC_INVALID_MSG_UNDER_CURSOR が戻ります。

ブラウザ呼び出し用のグループおよびセグメント情報は、キューからメッセージを削除する呼び出しの情報とは別個に保持されています。

MQGMO_MSG_UNDER_CURSOR

MQGMO の MatchOptions フィールドに指定されている MQMO_*オプションに関係なく、ブラウザ・カーソルが指すメッセージを検索します。メッセージはキューから削除されます。

ブラウザ・カーソルによって示されるメッセージは、MQGMO_BROWSE_FIRST オプションまたは MQGMO_BROWSE_NEXT オプションのいずれかを使用して最後に取得されたメッセージです。

MQGMO_COMPLETE_MSG が MQGMO_MSG_UNDER_CURSOR とともに指定されている場合、ブラウザ・カーソルは、MQMD 内の Offset フィールドがゼロであるメッセージを識別する必要があります。この条件が満たされない場合、呼び出しは失敗し、理由コード MQRC_INVALID_MSG_UNDER_CURSOR が戻ります。

このオプションと組み合わせて使用できないオプションは、以下のとおりです。

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_BROWSE_NEXT
- MQGMO_UNLOCK

キューがブラウザ用および入力用にオープンされていなかった場合もエラーになります。ブラウザ・カーソルが取得可能なメッセージを現在指し示していない場合は、MQGET 呼び出しでエラーが戻されます。

MQGMO_MARK_BROWSE_HANDLE

正常な MQGET によって戻されたメッセージ、または戻された *MsgToken* によって識別されるメッセージにマークが付けられます。マークは、呼び出しで使用するオブジェクト・ハンドルに固有のもので、

メッセージはキューから削除されません。

MQGMO_MARK_BROWSE_HANDLE は、以下のいずれかのオプションも指定されている場合にのみ有効です。

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_BROWSE_NEXT

MQGMO_MARK_BROWSE_HANDLE は、以下のいずれかのオプションでは無効です。

- MQGMO_ALL_MSGS_AVAILABLE
- MQGMO_ALL_SEGMENTS_AVAILABLE
- MQGMO_COMPLETE_MSG
- MQGMO_LOCK
- MQGMO_LOGICAL_ORDER
- MQGMO_UNLOCK

メッセージは、次のいずれかのイベントが発生するまでこの状態のままとなります。

- 関係するオブジェクト・ハンドルが、通常クローズまたは異常クローズされる。
- オプション MQGMO_UNMARK_BROWSE_HANDLE を指定した MQGET への呼び出しによって、このハンドルのメッセージがマーク解除される。
- メッセージが、MQCC_OK または MQCC_WARNING が戻されて完了した破壊 MQGET への呼び出しから戻される。MQGET が後でロールバックされた場合も、メッセージの状態は変更されたままです。
- メッセージの有効期限が切れる。

MQGMO_MARK_BROWSE_CO_OP

正常な MQGET によって戻されたメッセージ、または戻された *MsgToken* によって識別されるメッセージは、協働セット内のすべてのハンドルについてマーク付けされます。

協働レベル・マークは、既に設定されている可能性のある任意のハンドル・レベル・マークに追加して付けられます。

メッセージはキューから削除されません。

MQGMO_MARK_BROWSE_CO_OP は、使用されるオブジェクト・ハンドルが MQOO_CO_OP を指定した MQOPEN の呼び出しによって戻された場合にのみ有効です。以下の MQGMO オプションのいずれか 1 つも指定する必要があります。

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_BROWSE_NEXT

このオプションと組み合わせて使用できないオプションは、以下のとおりです。

- MQGMO_ALL_MSGS_AVAILABLE
- MQGMO_ALL_SEGMENTS_AVAILABLE
- MQGMO_COMPLETE_MSG
- MQGMO_LOCK
- MQGMO_LOGICAL_ORDER
- MQGMO_UNLOCK

メッセージが既にマークされており、オプション `MQGMO_UNMARKED_BROWSE_MSG` が指定されていない場合、呼び出しは `MQCC_FAILED` および理由コード `MQRC_MSG_MARKED_BROWSE_CO_OP` で失敗します。

メッセージは、次のいずれかのイベントが発生するまでこの状態のままとなります。

- 協働セットのオブジェクト・ハンドルすべてがクローズされる。
- オプション `MQGMO_UNMARK_BROWSE_CO_OP` を指定した `MQGET` の呼び出しによって協働ブラウザーのメッセージがマーク解除される。
- キュー・マネージャーによってメッセージが自動的にマーク解除される。
- メッセージが非ブラウザ `MQGET` の呼び出しから戻される。 `MQGET` が後でロールバックされた場合も、メッセージの状態は変更されたままです。
- メッセージの有効期限が切れる。

MQGMO_UNMARKED_BROWSE_MSG

`MQGMO_UNMARKED_BROWSE_MSG` を指定した `MQGET` の呼び出しは、ハンドルに対してマーク解除されたとみなされるメッセージを戻します。ハンドルに対してマークが付けられたメッセージは戻しません。キューが `MQOO_CO_OP` オプションを指定した `MQOPEN` の呼び出しによってオープンされており、メッセージが協働セットのメンバーによってマークされている場合も、メッセージを戻しません。

このオプションと組み合わせて使用できないオプションは、以下のとおりです。

- `MQGMO_ALL_MSGS_AVAILABLE`
- `MQGMO_ALL_SEGMENTS_AVAILABLE`
- `MQGMO_COMPLETE_MSG`
- `MQGMO_LOCK`
- `MQGMO_LOGICAL_ORDER`
- `MQGMO_UNLOCK`

MQGMO_UNMARK_BROWSE_CO_OP

このオプションを指定した `MQGET` 呼び出しの後では、メッセージは協働ハンドルのセット内のどのオープン・ハンドルからも、協働セット用にマーク付けされているとはみなされなくなります。その呼び出しの前にメッセージがハンドル・レベルでマーク付けされた場合は、引き続きハンドル・レベルでマーク付けされているとみなされます。

`MQGMO_UNMARK_BROWSE_CO_OP` は、`MQOO_CO_OP` オプションを指定した `MQOPEN` 呼び出しが成功して戻されたハンドルと共に使用する場合にのみ有効です。 `MQGET` は、協働ハンドル・セットからメッセージがマーク付けされているとみなされていなくても成功します。

`MQGMO_UNMARK_BROWSE_CO_OP` は非ブラウザ `MQGET` 呼び出しでは無効であり、以下のオプションと組み合わせて使用することもできません。

- `MQGMO_ALL_MSGS_AVAILABLE`
- `MQGMO_ALL_SEGMENTS_AVAILABLE`
- `MQGMO_COMPLETE_MSG`
- `MQGMO_LOCK`
- `MQGMO_LOGICAL_ORDER`
- `MQGMO_MARK_BROWSE_CO_OP`
- `MQGMO_UNLOCK`
- `MQGMO_UNMARKED_BROWSE_MSG`

MQGMO_UNMARK_BROWSE_HANDLE

このオプションを指定して `MQGET` を呼び出すと、置かれているメッセージは、このハンドルからは、マーク付けされているメッセージとみなされなくなります。

呼び出しは、このハンドルに対してメッセージがマークされていなくても成功します。

このオプションは、非ブラウザ MQGET 呼び出しでは無効であり、以下のオプションと組み合わせて使用することもできません。

- MQGMO_ALL_MSGS_AVAILABLE
- MQGMO_ALL_SEGMENTS_AVAILABLE
- MQGMO_COMPLETE_MSG
- MQGMO_LOCK
- MQGMO_LOGICAL_ORDER
- MQGMO_MARK_BROWSE_CO_OP
- MQGMO_UNLOCK
- MQGMO_UNMARKED_BROWSE_MSG

ロックオプション

以下のオプションは、キュー上のメッセージのロックに関連したオプションです。

MQGMO_LOCK

ブラウザされるメッセージをロックします。ロックされたメッセージは、このキューに対してオープンされた他のハンドルからは見えなくなります。このオプションを指定できるのは、以下のオプションのいずれか 1 つが指定されている場合に限りです。

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_NEXT
- MQGMO_BROWSE_MSG_UNDER_CURSOR

各キュー・ハンドルに対して 1 つのメッセージだけをロックできます。メッセージは、論理メッセージでも物理メッセージでも構いません。

- MQGMO_COMPLETE_MSG を指定すると、論理メッセージを構成するすべてのメッセージ・セグメントがキュー・ハンドルにロックされます。メッセージはすべてそのキュー上にあり、取得可能になっている必要があります。
- MQGMO_COMPLETE_MSG を省略すると、1 つの物理メッセージのみがキュー・ハンドルに対してロックされます。このメッセージがたまたま論理メッセージのセグメントである場合、ロックされたセグメントは、他のアプリケーションが MQGMO_COMPLETE_MSG を使用して論理メッセージを検索またはブラウザするのを防ぎます。

ロックされたメッセージは常にブラウザ・カーソルの下にあります。メッセージは、後で MQGMO_MSG_UNDER_CURSOR オプションを指定した MQGET 呼び出しを実行するとキューから削除できます。キュー・ハンドルを使用する他の MQGET 呼び出し (例えば、ロックされたメッセージのメッセージ ID を指定した呼び出し) でも、メッセージを削除できます。

呼び出しが完了コード MQCC_FAILED または MQCC_WARNING と理由コード MQRC_TRUNCATED_MSG_FAILED を戻した場合、メッセージはロックされません。

アプリケーションがキューからメッセージを削除しない場合、ロックは以下のアクションのいずれかによって解除されます。

- このハンドルに対して、MQGMO_BROWSE_FIRST または MQGMO_BROWSE_NEXT のどちらかを指定した別の MQGET 呼び出しを発行する。呼び出しが MQCC_OK または MQCC_WARNING で完了すると、ロックは解除されます。呼び出しが MQCC_FAILED で完了すると、メッセージはロックされたままになります。ただし、次の場合は例外になります。
 - MQCC_WARNING が MQRC_TRUNCATED_MSG_FAILED で返された場合、メッセージはアンロックされません。
 - MQCC_FAILED が MQRC_NO_MSG_AVAILABLE とともに返された場合、メッセージはアンロックされます。

MQGMO_LOCK も指定した場合、返されるメッセージはロックされます。MQGMO_LOCK を省略すると、呼び出し後にロックされたメッセージは表示されません。

MQGMO_WAIT を指定し、すぐに使用可能なメッセージがない場合は、待機の開始前に元のメッセージがアンロックされます。

- このハンドルに対して、MQGMO_BROWSE_MSG_UNDER_CURSOR を指定した (MQGMO_LOCK は指定しない) 別の MQGET 呼び出しを発行する。呼び出しが MQCC_OK または MQCC_WARNING で完了すると、ロックは解除されます。呼び出しが MQCC_FAILED で完了すると、メッセージはロックされたままになります。ただし、次のような例外があります。
 - MQCC_WARNING が MQRC_TRUNCATED_MSG_FAILED で返された場合、メッセージはアンロックされません。
- このハンドルに対して、MQGMO_UNLOCK を指定した別の MQGET 呼び出しを発行する。
- このハンドルに対して MQCLOSE 呼び出しを発行する。アプリケーションの終了によって MQCLOSE が暗黙的に行われる可能性があります。

付随するブラウズ・オプションを指定するのに必要な MQOO_BROWSE の他には、MQGMO_LOCK を指定するのに特別な MQOPEN オプションは必要ありません。

MQGMO_LOCK は、以下のいずれかのオプションでは無効です。

- MQGMO_MARK_SKIP_BACKOUT
- MQGMO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT
- MQGMO_UNLOCK

MQGMO_UNLOCK

ロック解除するメッセージは、MQGMO_LOCK オプションを指定した MQGET 呼び出しによって事前にロックされていなければなりません。このハンドルに対してロックされたメッセージがない場合、呼び出しは MQCC_WARNING および MQRC_NO_MSG_LOCKED で完了します。

MQGMO_UNLOCK を指定した場合、**MsgDesc**、**BufferLength**、**Buffer**、および **DataLength** パラメーターは検査も変更もされません。*Buffer* にメッセージは返されません。

MQGMO_UNLOCK を指定するために特別なオープン・オプションは必要ありません (ただし、最初にロック要求を発行するには MQOO_BROWSE が必要です)。

このオプションは、以下のオプション以外のオプションとは組み合わせて使用できません。

- MQGMO_NO_WAIT
- MQGMO_NO_SYNCPOINT

これらのオプションは、指定されているかどうかに関わらず、どちらも想定されています。

メッセージデータオプション

以下のオプションは、メッセージがキューから読み取られるときのメッセージ・データの処理に関連しています。

MQGMO_ACCEPT_TRUNCATED_MSG

メッセージ・バッファが小さく、メッセージ全体を収容できない場合に、MQGET 呼び出しによってバッファを満たすことができますようにします。MQGET はバッファに収容できるだけメッセージを入れます。警告完了コードを出して、その処理を終了します。つまり、以下のようになります。

- メッセージをブラウズすると、ブラウズ・カーソルが、戻されたメッセージに進みます。
- メッセージを削除すると、戻されたメッセージがキューから削除されます。
- 他にエラーが発生していなければ、理由コード MQRC_TRUNCATED_MSG_ACCEPTED が戻ります。

このオプションを指定しないと、バッファは、収容できる分だけのメッセージで満たされたままになります。警告完了コードが出されますが、処理は完了しません。つまり、以下のようになります。

- メッセージをブラウズしても、ブラウズ・カーソルは進みません。
- メッセージを削除しても、メッセージはキューから削除されません。
- 他のエラーが発生しない場合は、理由コード MQRC_TRUNCATED_MSG_FAILED が返されます。

MQGMO_CONVERT

このオプションは、メッセージ内のアプリケーションデータを、``呼び出し MQGET の **MsgDesc** パラメータで指定された **CodedCharSetId** **Encoding** 値と `` 値に準拠するように変換します。データは、**Buffer** パラメーターにコピーされる前に変換されます。

メッセージが書き込まれたときに指定された **Format** フィールドは、メッセージ内のデータの性質を識別するために変換プロセスによって想定されます。メッセージ・データの変換は、組み込み形式の場合はキュー・マネージャーによって行われ、他の形式の場合はユーザー作成出口によって行われます。データ変換出口の詳細については、[903 ページの『データ変換出口』](#)を参照してください。

- 変換が成功した場合、`` **MsgDesc** パラメータで指定された `` **CodedCharSetId** および **Encoding** フィールドは、`` 呼び出し MQGET からの戻り値において変更されません。
- 変換が失敗した場合のみ、メッセージ・データは変換されずに戻されます。MsgDesc の **CodedCharSetId** および **Encoding** フィールドは、変換されていないメッセージの値に設定されます。この場合、完了コードは MQCC_WARNING です。

いずれの場合も、これらのフィールドは、**Buffer** パラメーターで返されるメッセージ・データの文字セット ID とエンコードを記述します。

キュー・マネージャーが変換を実行する形式名のリストについては、[385 ページの『MQMD - メッセージ記述子』](#)で説明されている **Format** フィールドを参照してください。

グループおよびセグメントのオプション

以下のオプションは、論理メッセージのグループおよびセグメント内のメッセージの処理に関連しています。オプションについて説明する前に、重要な用語についていくつか定義しておきます。

物理メッセージ

物理メッセージは、キューに入れたりキューから削除したりできる最小の情報単位です。多くの場合、1つのMQPUT、MQPUT1、またはMQGET 呼び出しで指定または取得された情報に相当します。すべての物理メッセージには、固有のメッセージ記述子、MQMD があります。通常、物理メッセージは、メッセージ ID (MQMD の **MsgId** フィールド) の値が異なることによって区別されます。ただし、キュー・マネージャーは固有の値を強制しません。

論理メッセージ

論理メッセージは、アプリケーション情報の単一ユニットです。システムに制約がない場合には、1つの論理メッセージが1つの物理メッセージになります。論理メッセージが大きい場合、システムの制約により、1つの論理メッセージをセグメントと呼ばれる複数の物理メッセージに分割することが必要になる場合があります。

セグメント化された論理メッセージは、同じ非ヌル・グループ ID (MQMD の **GroupId** フィールド) を持つ複数の物理メッセージで構成されます。これらは同じメッセージ・シーケンス番号 (MQMD の **MsgSeqNumber** フィールド) を持っています。セグメントは、セグメント・オフセット (MQMD の **Offset** フィールド) の値が異なることによって区別されます。セグメント・オフセットは、論理メッセージ内のデータの先頭から始まる、物理メッセージ内のデータのオフセットです。各セグメントは1つの物理メッセージなので、論理メッセージ内のセグメントには通常、それぞれ固有のメッセージ ID があります。

セグメント化されていない論理メッセージにも、送信側のアプリケーションでセグメント化が許可されている場合は、ヌル以外のグループ ID があります。この場合、論理メッセージがメッセージ・グループに属していなければ、そのグループ ID を持つ物理メッセージは1つだけになります。送信側アプリケーションによってセグメント化が禁止されている論理メッセージは、論理メッセージがメッセージ・グループに属していない限り、ヌルのグループ ID MQGI_NONE を持ちます。

メッセージ・グループ

メッセージ・グループは、同じヌル以外のグループ ID を持つ1つ以上の論理メッセージのセットです。グループ内のそれぞれの論理メッセージは、メッセージ順序番号に指定された固有の値で区別されま

す。順序番号は 1 から n までの整数で、n はグループ内の論理メッセージの数です。1 つ以上の論理メッセージをセグメント化すると、グループ内の物理メッセージの数は n 個を超えます。

MQGMO_LOGICAL_ORDER

MQGMO_LOGICAL_ORDER は、キュー・ハンドルに対する連続した MQGET 呼び出しでメッセージが戻される順序を制御します。このオプションは、それぞれの呼び出しごとに指定する必要があります。

同じキュー・ハンドルに対する後続の MQGET 呼び出しで MQGMO_LOGICAL_ORDER を指定すると、グループ内のメッセージはそのメッセージ順序番号に指定された順序で戻されます。論理メッセージのセグメントはそのセグメント・オフセットで指定された順序で戻されます。この順序は、これらのメッセージやセグメントのキューでの出現順序とは異なる場合があります。

注: MQGMO_LOGICAL_ORDER を指定しても、グループに属していないメッセージやセグメントではないメッセージに悪影響はありません。そのようなメッセージは、事実上、1 つのメッセージのみで構成されるメッセージ・グループに属するものとして扱われます。グループ内のメッセージ、メッセージ・セグメント、およびグループ内にないセグメント化されていないメッセージが混在するキューからメッセージを取り出す場合は、MQGMO_LOGICAL_ORDER を指定しても安全です。

必要な順序でメッセージを戻すように、キュー・マネージャーは、連続した MQGET 呼び出し間でグループおよびセグメント情報を保持します。グループおよびセグメント情報により、キュー・ハンドルに対する現行のメッセージ・グループと現行の論理メッセージが特定されます。また、グループおよび論理メッセージ内の現行の位置、およびメッセージが 1 つの作業単位内で取得されているかどうかも特定します。キュー・マネージャーがこの情報を保持するので、アプリケーションでは、それぞれの MQGET 呼び出しの前にグループおよびセグメント情報を設定する必要はありません。具体的には、アプリケーションが MQMD に GroupId、MsgSeqNumber、および Offset の各フィールドを設定する必要がないことを意味します。ただし、アプリケーションは、各呼び出しで MQGMO_SYNCPOINT オプションまたは MQGMO_NO_SYNCPOINT オプションを正しく設定する必要があります。

キューがオープンしているときは、現行のメッセージ・グループも現行の論理メッセージも存在しません。1 つのメッセージ・グループが現行のメッセージ・グループとなるのは、MQMF_MSG_IN_GROUP フラグ付きのメッセージが MQGET 呼び出しで戻されたときです。連続した呼び出しで MQGMO_LOGICAL_ORDER を指定すると、以下のようなメッセージが戻されるまで、そのグループは現行グループのままになります。

- MQMF_SEGMENT が指定されていない MQMF_LAST_MSG_IN_GROUP (つまり、グループ内の最後の論理メッセージがセグメント化されていません)。
- MQMF_LAST_SEGMENT を指定した MQMF_LAST_MSG_IN_GROUP (つまり、戻されるメッセージは、グループ内の最後の論理メッセージの最後のセグメントです)。

このようなメッセージが戻されると、メッセージ・グループが終了し、MQGET 呼び出しが正常に完了すると現行のグループはなくなります。同様に、論理メッセージが現行の論理メッセージとなるのは、MQMF_SEGMENT フラグ付きのメッセージが MQGET 呼び出しで戻されたときです。

MQMF_LAST_SEGMENT フラグを持つメッセージが返されると、論理メッセージは終了します。

選択基準が指定されていない場合、後続の MQGET 呼び出しは、キュー上の最初のメッセージ・グループのメッセージを正しい順序で戻します。次に 2 番目のメッセージ・グループのメッセージが戻されます。これは、使用できるメッセージがなくなるまで続きます。MatchOptions フィールドに以下のオプションを 1 つ以上指定することによって、戻される特定のメッセージ・グループを選択することができます。

- MQMO_MATCH_MSG_ID
- MQMO_MATCH_CORREL_ID
- MQMO_MATCH_GROUP_ID

ただし、これらのオプションが有効なのは、現行のメッセージ・グループも現行の論理メッセージも存在しない場合だけです。詳しくは、[326 ページの『MQGMO - 読み取りメッセージ・オプション』](#)で説明されている MatchOptions フィールドを参照してください。

[355 ページの表 510](#) キューマネージャが、呼び出し MQGET で返すメッセージを検索しようとする際に参照する、OffsetMsgId フィールド、GroupIdCorrelId、MsgSeqNumber、の値を示します。

規則は、キューからメッセージを削除する場合にもキューに入っているメッセージをブラウズする場合にも適用されます。この表の「どちらも」は「はい」または「いいえ」を意味します。

LOG ORD

呼び出しで MQGMO_LOGICAL_ORDER オプションが指定されているかどうかを示します。

Cur grp

現行のメッセージ・グループが呼び出しの前に存在するかどうかを示します。

Cur log msg

現行の論理メッセージが呼び出しの前に存在するかどうかを示します。

その他の列

キュー・マネージャーが検索する値を示します。「前の」という表現は、キュー・ハンドルに対して前のメッセージのフィールドに戻された値を示します。

表 510. 論理メッセージのグループおよびセグメントに関連した MQGET オプション							
指定するオプション	呼び出しの前のグループおよび論理メッセージの状況		キュー・マネージャーが検索する値				
LOG ORD	Cur grp	Cur log msg	MsgId	CorrelId	GroupId	MsgSeqNumber	Offset
はい	いいえ	いいえ	統制活動担当者 MatchOptions	統制活動担当者 MatchOptions	統制活動担当者 MatchOptions	1	0
はい	いいえ	はい	任意のメッセージ ID	任意の相関 ID	前のグループ ID	1	前のオフセット + 前のセグメント長
はい	はい	いいえ	任意のメッセージ ID	任意の相関 ID	前のグループ ID	前の順序番号 + 1	0
はい	はい	はい	任意のメッセージ ID	任意の相関 ID	前のグループ ID	前の順序番号	前のオフセット + 前のセグメント長
いいえ	どちらも	どちらも	統制活動担当者 MatchOptions	統制活動担当者 MatchOptions	統制活動担当者 MatchOptions	統制活動担当者 MatchOptions	統制活動担当者 MatchOptions

キュー上に複数のメッセージ・グループが存在し、戻してよいものである場合、それらのグループが戻される順序は、各グループ内の最初の論理メッセージの最初のセグメントがキュー上のどの位置にあるかによって決定されます。つまり、メッセージ順序番号が 1、オフセットが 0 の物理メッセージにより、該当するグループが戻される順序が決定されます。

MQGMO_LOGICAL_ORDER オプションは、以下のように作業単位に影響します。

- 1つのグループ内の最初の論理メッセージまたはセグメントが 1つの作業単位内で取得された場合には、同じキュー・ハンドルが使用されていれば、そのグループ内の他の論理メッセージおよびセグメントはすべて 1つの作業単位内で取得する必要があります。ただし、これらは同じ作業単位内で取得する必要はありません。これにより、多くの物理メッセージからなるメッセージ・グループを、キュー・ハンドルに対する 2つ以上の連続した作業単位にまたがって分割できます。
- 1つのグループ内の最初の論理メッセージまたはセグメントが 1つの作業単位内で取得されていない場合には、同じキュー・ハンドルが使用されていれば、そのグループ内のその他の論理メッセージおよびセグメントはどれも 1つの作業単位内で取得することはできません。

これらの条件が満たされないと、MQGET 呼び出しは失敗し、理由コード MQRC_INCONSISTENT_UOW が戻ります。

が MQGMO_LOGICAL_ORDER 指定された場合、呼び出し MQGET 時に提供される MQGMO は以下より小さくあってはならず、MQMD は MQGMO_VERSION_2 以下より小さく MQMD_VERSION_2 あってはならない。この条件が満たされない場合、呼び出しは失敗し、該当する理由コード MQRC_WRONG_GMO_VERSION または MQRC_WRONG_MD_VERSION が戻ります。

キュー・ハンドルに対する連続した MQGET 呼び出しで MQGMO_LOGICAL_ORDER を指定しないと、メッセージ・グループに属しているか、また論理メッセージのセグメントであるかに関係なく、メッセージが戻されます。これにより、あるグループ内のメッセージや論理メッセージのセグメントが正しくない順序で戻されたり、他のグループ内のメッセージ、他の論理メッセージのセグメント、またはグループにも属さずセグメントでもないメッセージとが混在したりする場合があります。この場合、連続した MQGET 呼び出しで戻されるメッセージは、それらの呼び出しで指定した MQMO_* オプションによって制御されます (これらのオプションの詳細については、[326 ページの『MQGMO - 読み取りメッセージ・オプション』](#)で説明している *MatchOptions* フィールドを参照してください)。

この手法を用いると、システム障害が発生した後に、メッセージ・グループまたは論理メッセージを途中から再開することができます。システムを再始動したら、アプリケーションで GroupId、MsgSeqNumber、Offset、および MatchOptions の各フィールドに適切な値を設定してから、MQGMO_SYNCPOINT または MQGMO_NO_SYNCPOINT を設定した (ただし、MQGMO_LOGICAL_ORDER は指定しない) MQGET 呼び出しを発行できます。この呼び出しが成功した場合、キュー・マネージャーはグループおよびセグメント情報を保持し、キュー・ハンドルを使用する後続の MQGET 呼び出しで通常どおり MQGMO_LOGICAL_ORDER を指定できます。

MQGET 呼び出しのためにキュー・マネージャーが保持しているグループおよびセグメント情報は、MQPUT 呼び出しのためにキュー・マネージャーが保持しているグループおよびセグメント情報とは異なります。また、キュー・マネージャーは、以下についての情報もそれぞれ保持しています。

- キューからメッセージを削除する MQGET 呼び出し。
- キュー上のメッセージをブラウズする MQGET 呼び出し。

任意のキューハンドルに対して、アプリケーションは指定する呼び出しと指定 MQGETMQGMO_LOGICAL_ORDERMQGET しない呼び出しを混在させることができる。ただし、以下の点に注意してください。

- MQGMO_LOGICAL_ORDER を省略した場合、MQGET 呼び出しが成功するたびに、キュー・マネージャーは、保存されているグループおよびセグメント情報を、戻されたメッセージに対応する値に変更して設定します。つまり、キュー・ハンドルに対してキュー・マネージャーで保持されていた既存のグループおよびセグメント情報が、この値で置換されます。呼び出しのアクション (ブラウズまたは削除) に該当する情報だけが変更されます。
- MQGMO_LOGICAL_ORDER を省略すると、現行のメッセージ・グループまたは論理メッセージがあっても呼び出しは失敗しません。呼び出しは成功し、MQCC_WARNING 完了コードが出される場合があります。[357 ページの表 511](#) に、発生する可能性のあるいくつかのケースを示しています。これらの場合、完了コードが MQCC_OK でなければ、理由コードは以下のいずれか (該当するもの) になります。
 - MQRC_INCOMPLETE_GROUP
 - MQRC_INCOMPLETE_MSG
 - MQRC_INCONSISTENT_UOW

注: キュー・マネージャーは、キューをブラウズするとき、またはブラウズ用にオープンされたが入力用にオープンされなかったキューをクローズするときに、グループおよびセグメント情報をチェックしません。これらの場合、完了コードは常に MQCC_OK です (他のエラーは想定されません)。

表 511. グループおよびセグメント情報と整合性のない MQGET または MQCLOSE 呼び出しの結果		
現行の呼び出し	前の呼び出しが MQGMO_LOGICAL_ORDER を指定し た MQGET	前の呼び出しが MQGMO_LOGICAL_ORDER を指定し ない MQGET
MQGET で MQGMO_LOGICAL_ORDER	MQCC_FAILED	MQCC_FAILED
MQGET (MQGMO_LOGICAL_ORDER なし)	MQCC_WARNING	MQCC_OK
終了していないグループまたは論理 メッセージを指定している MQCLOSE	MQCC_WARNING	MQCC_OK

メッセージおよびセグメントを論理順序でリトリブしたいアプリケーションは、MQGMO_LOGICAL_ORDER を指定することをお勧めします。これは、使用する最も簡単なオプションであるためです。このオプションを指定すると、キュー・マネージャーがグループおよびセグメント情報を管理するので、アプリケーションでこの情報を管理する必要はなくなります。ただし、特殊アプリケーションは、MQGMO_LOGICAL_ORDER オプションによって提供される制御よりも多くの制御を必要とする場合があります。これは、そのオプションを指定しないことによって行うことができます。そのようなアプリケーションでは、MQMD の MsgId、CorrelId、GroupId、MsgSeqNumber、および Offset フィールドと、MQGMO の MatchOptions の MQMO_* オプションを、それぞれの MQGET 呼び出しの前に正しく設定する必要があります。

例えば、受信する物理メッセージがグループ内にあるか、論理メッセージのセグメント内にあるかに関係なく、それらのメッセージを転送するアプリケーションでは、MQGMO_LOGICAL_ORDER を指定してはなりません。送信側のキュー・マネージャーと受信側のキュー・マネージャーの間にパスが複数あるような複雑なネットワークの場合には、物理メッセージが正しくない順序で到達することがあります。MQGMO_LOGICAL_ORDER も、MQPUT 呼び出しで対応する MQPMO_LOGICAL_ORDER も指定しないことによって、転送アプリケーションは、論理順序で次の物理メッセージが到着するのを待たずに、到着するとすぐに各物理メッセージを検索して転送することができます。

MQGMO_LOGICAL_ORDER は、他のどの MQGMO_* オプションとも一緒に指定できます。また、該当する状況ではさまざまな MQMO_* オプションと一緒に指定できます (前のセクションを参照してください)。

-  z/OS では、このオプションは専用キューと共用キューでサポートされますが、キューの索引タイプは MQIT_GROUP_ID でなければなりません。共有キューの場合、キューのマップ先の CFSTRUCT オブジェクトは CFLEVEL(3) 以上でなければなりません。
- このオプションは、以下のプラットフォームのすべてのローカル・キューでサポートされています。

-  AIX
-  Linux
-  IBM i
-  Windows

および、これらのシステムに接続された IBM MQ MQI clients。

MQGMO_COMPLETE_MSG

完全な論理メッセージだけを MQGET 呼び出しで戻すことができます。論理メッセージがセグメント化されていると、キュー・マネージャーはそれらのセグメントの再組み立てを行い、完全な論理メッセージをアプリケーションに戻します。論理メッセージがセグメント化されていたことは、それを受け取るアプリケーションには分かりません。

注: キュー・マネージャーがメッセージ・セグメントの再組み立てを行うように指定するのは、このオプションだけです。このオプションを指定しないと、キューに入っている (および MQGET 呼び出しで指定された他の選択基準を満たしている) セグメントはそれぞれ個別にアプリケーションに戻されま

す。個々のセグメントを受け取りたくないアプリケーションは、常に `MQGMO_COMPLETE_MSG` を指定する必要があります。

このオプションを使用するには、アプリケーションが完全なメッセージを収容するのに十分な大きさのバッファを提供するか、`MQGMO_ACCEPT_TRUNCATED_MSG` オプションを指定する必要があります。

キューにセグメント化されたメッセージがあり、いくつかのセグメントが欠落している場合 (ネットワーク内で遅延し、まだ到着していないことが原因である可能性があります)、`MQGMO_COMPLETE_MSG` を指定すると、不完全な論理メッセージに属するセグメントを取得できなくなります。ただし、これらのメッセージ・セグメントは引き続き **CurrentQDepth** キュー属性の値に寄与します。これは、**CurrentQDepth** がゼロより大きい場合でも、検索可能な論理メッセージが存在しない可能性があることを意味します。

持続メッセージの場合、キュー・マネージャーがセグメントの再組み立てを実行できるのは、1つの作業単位内だけです。

- **MQGET** 呼び出しがユーザー定義の作業単位内で実行されていれば、その作業単位が使用されます。呼び出しが再組み立て処理中に失敗した場合、キュー・マネージャーは、再組み立て中に削除されたすべてのセグメントをキューに復元します。ただし、このように失敗しても、作業単位は正常にコミットされます。
- 呼び出しがユーザー定義の作業単位外で実行されていて、またユーザー定義の作業単位が存在していない場合、キュー・マネージャーは、この呼び出しの間の作業単位を作成します。呼び出しが成功すると、キュー・マネージャーは作業単位を自動的にコミットします (アプリケーションでこれを行う必要はありません)。呼び出しが失敗すると、キュー・マネージャーは作業単位をバックアウトします。
- 呼び出しがユーザー定義の作業単位外で実行されているが、ユーザー定義の作業単位が存在している場合、キュー・マネージャーは再組み立てを実行できません。再組み立てが必要でないメッセージでは、呼び出しが成功することもあります。しかし、メッセージの再組み立てが必要な場合は、呼び出しは理由コード `MQRC_UOW_NOT_AVAILABLE` で失敗します。

非持続メッセージの場合、キュー・マネージャーが再組み立てを実行するのに、作業単位が使用可能になっている必要はありません。

1つのセグメントであるそれぞれの物理メッセージには、固有のメッセージ記述子があります。単一の論理メッセージを構成するセグメントの場合、メッセージ記述子内のほとんどのフィールドは、論理メッセージ内のすべてのセグメントで同じです。通常、論理メッセージ内のセグメント間で異なるのは、**MsgId**、**Offset**、および **MsgFlags** フィールドのみです。ただし、セグメントが中間キュー・マネージャーの送達不能キューに置かれている場合、**DLQ** ハンドラーは `MQGMO_CONVERT` オプションを指定してメッセージを取得します。これにより、セグメントの文字セットまたはエンコードが変更される可能性があります。それで、この途中で **DLQ** ハンドラーがセグメントの送信に成功した場合、そのセグメントが宛先キュー・マネージャーに到着した時点で、そのセグメントの文字セットまたはエンコードは論理メッセージ内の他のセグメントと異なっていることがあります。

CodedCharSetId フィールドと **Encoding** フィールドが異なるセグメントで構成される論理メッセージを、キュー・マネージャーが単一の論理メッセージに再組み立てすることはできません。その代わりに、キュー・マネージャーは、論理メッセージの先頭にあり、同じ文字セット ID とエンコードを持つ連続したセグメントをいくつか再組み立てして戻します。このとき、**MQGET** 呼び出しは完了し、完了コード `MQCC_WARNING` と、理由コード `MQRC_INCONSISTENT_CCSDS` または `MQRC_INCONSISTENT_ENCODINGS` の該当する方が戻ります。これは、`MQGMO_CONVERT` が指定されているかどうかに関係なく発生します。残りのセグメントを取得するには、アプリケーションで `MQGMO_COMPLETE_MSG` オプションを指定せずに **MQGET** 呼び出しを再発行する必要があります。これによって、セグメントが1つずつ取得できるようになります。`MQGMO_LOGICAL_ORDER` を使用して、残りのセグメントを順番に取り出すことができます。

セグメントの書き込みを行うアプリケーションでは、メッセージ記述子内の他のフィールドに、セグメント間で異なる値を設定できます。ただし、受信側アプリケーションが `MQGMO_COMPLETE_MSG` を使用して論理メッセージを検索する場合は、これを行う利点はありません。キュー・マネージャーは、論理メッセージの再組み立て時に、最初のセグメントのメッセージ記述子からの値をメッセージ記述子

に戻します。唯一の例外は `MsgFlags` フィールドで、再組み立てされたメッセージが唯一のセグメントであることを示すためにキュー・マネージャーが設定します。

レポート・メッセージに `MQGMO_COMPLETE_MSG` が指定されている場合、キュー・マネージャーは特殊な処理を実行します。キュー・マネージャーは、キューをチェックして、論理メッセージ内のさまざまなセグメントに関連するそのレポート・タイプのすべてのレポート・メッセージがキュー上に存在しているかどうかを確認します。正しい場合は、`MQGMO_COMPLETE_MSG` を指定することにより、単一メッセージとして取り出すことができます。これを可能にするには、セグメント化をサポートするキュー・マネージャーまたは MCA によってレポート・メッセージを生成するか、発信元アプリケーションが少なくとも 100 バイトのメッセージ・データを要求する必要があります(つまり、適切な `MQRO_*_WITH_DATA` または `MQRO_*_WITH_FULL_DATA` オプションを指定する必要があります)。1 つのセグメントに入るアプリケーション・データの一部に欠落があると、戻されるレポート・メッセージでは、足りないバイト分はヌルで置き換えられます。

`MQGMO_COMPLETE_MSG` が `MQGMO_MSG_UNDER_CURSOR` または `MQGMO_BROWSE_MSG_UNDER_CURSOR` とともに指定されている場合、MQMD 内の *Offset* フィールドの値が 0 であるメッセージにブラウズ・カーソルを配置する必要があります。この条件が満たされない場合、呼び出しは失敗し、理由コード `MQRC_INVALID_MSG_UNDER_CURSOR` が戻ります。

`MQGMO_COMPLETE_MSG` は `MQGMO_ALL_SEGMENTS_AVAILABLE` を暗黙指定するため、指定する必要はありません。

`MQGMO_COMPLETE_MSG` は、`MQGMO_SYNCPOINT_IF_PERSISTENT` 以外の任意の `MQGMO_*` オプション、および `MQMO_MATCH_OFFSET` 以外の任意の `MQMO_*` オプションと一緒に指定することができます。

-  z/OS では、このオプションは専用キューおよび共用キューでサポートされますが、キューの索引タイプは `MQIT_GROUP_ID` でなければなりません。共有キューの場合、キューのマップ先である `CFSTRUCT` オブジェクトは `CFLEVEL(3)` 以上でなければなりません。

- 以下のプラットフォーム:

-  AIX
-  IBM i
-  Linux
-  Windows

および、これらのシステムに接続された IBM MQ MQI clients の場合、このオプションはすべてのローカル・キューについてサポートされます。

MQGMO_ALL_MSGS_AVAILABLE

グループ内のメッセージがすべて使用可能な場合に限り、そのグループ内のメッセージを取得できるようになります。キューにいくつかのメッセージが欠落しているメッセージ・グループが含まれている場合(ネットワーク内で遅延していて、まだ到着していないなどの理由で)、`MQGMO_ALL_MSGS_AVAILABLE` を指定すると、不完全なグループに属するメッセージを取得できなくなります。ただし、これらのメッセージは引き続き **CurrentQDepth** キュー属性の値に寄与します。これは、`CurrentQDepth` がゼロより大きい場合でも、検索可能なメッセージ・グループが存在しない可能性があることを意味します。検索可能なメッセージが他にない場合は、指定された待機間隔(ある場合)が経過した後で、理由コード `MQRC_NO_MSG_AVAILABLE` が戻されます。

`MQGMO_ALL_MSGS_AVAILABLE` の処理は、`MQGMO_LOGICAL_ORDER` も指定されているかどうかによって異なります。

- 両方のオプションを指定すると、`MQGMO_ALL_MSGS_AVAILABLE` は、現行のグループまたは論理メッセージがない場合にのみ有効になります。現行のグループまたは論理メッセージがある場合、`MQGMO_ALL_MSGS_AVAILABLE` は無視されます。これは、メッセージを論理順序で処理するときに `MQGMO_ALL_MSGS_AVAILABLE` がオンのままである可能性があることを意味します。

- `MQGMO_LOGICAL_ORDER` を指定せずに `MQGMO_ALL_MSGS_AVAILABLE` を指定すると、`MQGMO_ALL_MSGS_AVAILABLE` は常に効果を持ちます。つまり、このオプションは、グループ内の最初のメッセージがキューから削除されたときにオフにする必要があります。これによって、そのグループ内の残りのメッセージを削除できるようになります。

`MQGMO_ALL_MSGS_AVAILABLE` を指定した `MQGET` 呼び出しが正常に完了するということは、`MQGET` 呼び出しが発行された時点で、グループ内のすべてのメッセージがキュー上にあるという意味です。しかし、それでも他のアプリケーションが、グループからメッセージを削除できることに注意してください (グループは、グループ内の最初のメッセージを取得するアプリケーションに対してロックされていません)。

このオプションを省略すると、不完全なグループに属しているメッセージが取得されることがあります。

`MQGMO_ALL_MSGS_AVAILABLE` は `MQGMO_ALL_SEGMENTS_AVAILABLE` を暗黙指定するため、指定する必要はありません。

`MQGMO_ALL_MSGS_AVAILABLE` は、他の任意の `MQGMO_*` オプション、および任意の `MQMO_*` オプションと一緒に指定することができます。

-  **z/OS** z/OS では、このオプションは専用キューおよび共有キューでサポートされますが、キューの索引タイプは `MQIT_GROUP_ID` でなければなりません。共有キューの場合、キューのマップ先である `CFSTRUCT` オブジェクトは `CFLEVEL(3)` 以上でなければなりません。
- 以下のプラットフォーム:

-  **AIX** AIX
-  **IBM i** IBM i
-  **Linux** Linux
-  **Windows** Windows

および、これらのシステムに接続された IBM MQ MQI clients の場合、このオプションはすべてのローカル・キューについてサポートされます。

MQGMO_ALL_SEGMENTS_AVAILABLE

論理メッセージ内のセグメントがすべて使用可能な場合に限り、その論理メッセージ内のセグメントを取得できるようになります。キューにセグメント化されたメッセージがあり、いくつかのセグメントが欠落している場合 (ネットワーク内で遅延し、まだ到着していないことが原因である可能性があります)、`MQGMO_ALL_SEGMENTS_AVAILABLE` を指定すると、不完全な論理メッセージに属するセグメントを取得できなくなります。ただし、これらのセグメントは引き続き **CurrentQDepth** キュー属性の値に加算されます。これは、**CurrentQDepth** がゼロより大きい場合でも、検索可能な論理メッセージがない可能性があることを意味します。検索可能なメッセージが他にない場合は、指定された待機間隔 (ある場合) が経過した後で、理由コード `MQRC_NO_MSG_AVAILABLE` が戻されます。

`MQGMO_ALL_SEGMENTS_AVAILABLE` の処理は、`MQGMO_LOGICAL_ORDER` も指定されているかどうかによって異なります。

- 両方のオプションを指定した場合、`MQGMO_ALL_SEGMENTS_AVAILABLE` が有効になるのは、現行の論理メッセージが存在しない場合のみです。現行の論理メッセージがある場合、`MQGMO_ALL_SEGMENTS_AVAILABLE` は無視されます。これは、メッセージを論理順序で処理するときに `MQGMO_ALL_SEGMENTS_AVAILABLE` がオンのままである可能性があることを意味します。
- `MQGMO_LOGICAL_ORDER` を指定せずに `MQGMO_ALL_SEGMENTS_AVAILABLE` を指定すると、`MQGMO_ALL_SEGMENTS_AVAILABLE` は常に効果を持ちます。つまり、このオプションは、論理メッセージ内の最初のセグメントがキューから削除されたときにオフにする必要があります。これによって、その論理メッセージ内の残りのセグメントを削除できるようになります。

このオプションを指定しないと、論理メッセージが不完全な場合でも、メッセージ・セグメントが取得されることがあります。

MQGMO_COMPLETE_MSG と MQGMO_ALL_SEGMENTS_AVAILABLE の両方とも、いずれかのセグメントを取得する前にすべてのセグメントが使用可能になっている必要がありますが、前者は完全なメッセージを返し、後者はセグメントを1つずつ取得することを許可します。

レポート・メッセージに MQGMO_ALL_SEGMENTS_AVAILABLE が指定されている場合、キュー・マネージャーはキューを検査して、完全な論理メッセージを構成するセグメントごとに少なくとも1つのレポート・メッセージがあるかどうかを確認します。存在する場合は、MQGMO_ALL_SEGMENTS_AVAILABLE 条件が満たされます。しかし、キュー・マネージャーは、キューに入っているレポート・メッセージのタイプはチェックしないので、論理メッセージのセグメントに対応したレポート・メッセージにはさまざまなタイプのレポートが混在している可能性があります。結果として、MQGMO_ALL_SEGMENTS_AVAILABLE が成功しても、MQGMO_COMPLETE_MSG が成功することを意味するわけではありません。ある論理メッセージのセグメントについてさまざまなレポート・タイプが混在している場合には、これらのレポート・メッセージは1つずつ取得する必要があります。

MQGMO_ALL_SEGMENTS_AVAILABLE は、他の任意の MQGMO_* オプション、および任意の MQMO_* オプションと一緒に指定することができます。

- z/OS では、このオプションは専用キューおよび共有キューでサポートされますが、キューの索引タイプは MQIT_GROUP_ID でなければなりません。共有キューの場合、キューのマップ先である CFSTRUCT オブジェクトは CFLEVEL(3) 以上でなければなりません。

- 以下のプラットフォーム:

-  AIX
-  IBM i
-  Linux
-  Windows

および、これらのシステムに接続された IBM MQ MQI clients の場合、このオプションはすべてのローカル・キューについてサポートされます。

プロパティオプション

以下のオプションは、メッセージのプロパティに関連したオプションです。

MQGMO_PROPERTIES_AS_Q_DEF

メッセージ記述子 (または拡張) に含まれるプロパティを除き、メッセージのプロパティは、**PropertyControl** キュー属性の定義に従って表す必要があります。MsgHandle が指定されている場合、**PropertyControl** キュー属性の値が MQPROP_FORCE_MQRFH2 でない限り、このオプションは無視され、メッセージのプロパティは MsgHandle を介して使用可能になります。

これは、プロパティ・オプションが指定されていないときのデフォルト・アクションです。

MQGMO_PROPERTIES_IN_HANDLE

メッセージのプロパティは、MsgHandle を介して使用可能にする必要があります。メッセージ・ハンドルが提供されない場合、呼び出しは理由 MQRC_HMSG_ERROR で失敗します。

注: メッセージ・ハンドルを作成しないアプリケーションによって後でメッセージが読み取られると、キュー・マネージャーはメッセージ・プロパティを MQRFH2 構造体に入れます。予期しない MQRFH2 ヘッダーが存在すると、既存のアプリケーションの動作が中断されることがあります。

MQGMO_NO_PROPERTIES

メッセージ記述子 (または拡張) に含まれるものを除き、メッセージのプロパティは取得されません。MsgHandle が指定されている場合は、無視されます。

MQGMO_PROPERTIES_FORCE_MQRFH2

メッセージ記述子 (または拡張) に含まれるプロパティを除き、メッセージのプロパティは MQRFH2 ヘッダーを使用して表す必要があります。これにより、プロパティを取得することが予期されるも

の、メッセージ・ハンドルを使用するように変更できない、アプリケーションの以前のバージョンとの互換性が提供されます。MsgHandle が指定されている場合は無視されます。

MQGMO_PROPERTIES_COMPATIBILITY

メッセージに、接頭辞が "mcd.", "usr.", "jms.", "mqext." または を持つプロパティが含まれる場合、すべてのメッセージプロパティは MQRFH2 ヘッダーでアプリケーションに配信されます。それ以外の場合、メッセージ記述子 (または拡張) に含まれるものを除くメッセージのプロパティはすべて廃棄され、アプリケーションにアクセスできなくなります。

デフォルトの選択肢

説明されているオプションがどれも必要でない場合には、以下のオプションを使用できます。

MQGMO_NONE

この値は、他のオプションが指定されなかったことを示すために使用します。すべてのオプションはデフォルト値であるとみなされます。MQGMO_NONE は、プログラムの文書化を支援します。このオプションを他のオプションと一緒に使用することは意図されていませんが、値がゼロであるため、そのような使用を検出できません。

Options フィールドの初期値は、MQGMO_NO_WAIT に MQGMO_PROPERTIES_AS_Q_DEF を加えた値です。

WaitInterval (MQLONG)

MQGET 呼び出しで適切なメッセージ (つまり、MQGET 呼び出しの **MsgDesc** パラメーターで指定した選択基準を満たすメッセージ) が到着するまで待機するおおよその時間をミリ秒で表します。詳細は [385 ページ](#) の『MQMD - メッセージ記述子』に記載されている フィールド *MsgId* を参照してください。この時間が経過しても、適切なメッセージが到着しなかった場合、呼び出しは完了し、MQCC_FAILED と理由コード MQRC_NO_MSG_AVAILABLE が戻ります。

重要: 適切なメッセージが即時に使用可能であれば、待機つまり遅延は生じません。

z/OS では、MQGET 呼び出しが実際に待機している期間は、システムのロードおよび作業スケジュールの問題に影響され、また *WaitInterval* に指定された値と *WaitInterval* を約 100 ミリ秒超えている場合では異なることがあります。

WaitInterval は、MQGMO_WAIT または MQGMO_SET_SIGNAL オプションと組み合わせて使用します。これらのオプションがどちらも指定されていないときは、これは無視されます。これらのうちのいずれかが指定されている場合は、*WaitInterval* は、ゼロ以上であるか、または次の特別な値でなければなりません。

MQWI_UNLIMITED

無制限の待機間隔。

このフィールドの初期値は 0 です。

Signal1 (MQLONG)

これは、MQGMO_SET_SIGNAL オプションと合わせて指定する場合にだけ使用される入力フィールドで、メッセージが使用可能なときに送達するシグナルを識別します。

注: このフィールドのデータ・タイプおよび用途は、環境によって決まります。したがって、異なる環境との間で移植するアプリケーションでは、シグナルを使用してはなりません。

- z/OS では、このフィールドにはイベント制御ブロック (ECB) のアドレスを入れなければなりません。ECB は、MQGET 呼び出しを発行する前にアプリケーションでクリアする必要があります。ECB が含まれているストレージは、キューがクローズされるまでは解放してはなりません。ECB は、下で説明されているシグナル完了コードの 1 つと共に、キュー・マネージャーによって通知されます。これらの完了コードは、ECB のビット 2 から 31、つまり z/OS マッピング・マクロ IHAECB 内でユーザー完了コードとして定義されている領域に設定されます。
- それ以外のすべての環境では、これは予約フィールドです。したがって、値に意味はありません。

シグナル完了コードは、以下のとおりです。

MQEC_MSG_ARRIVED

適切なメッセージがキューに到着しました。このメッセージは呼び出し元用に予約されませんでした。第2のMQGET要求を発行する必要があります。ただし、第2の要求を出す前に別のアプリケーションがそのメッセージを取り出すことがあります。

MQEC_WAIT_INTERVAL_EXPIRED

適切なメッセージが到着しない間に、指定の *WaitInterval* が過ぎました。

MQEC_WAIT_CANCELED

不確定な理由(キュー・マネージャーの終了、キューが使用不可になっているなど)によって、待機が取り消されました。さらに診断を行う場合は要求を再発行してください。

MQEC_Q_MGR QUIESCING

キュー・マネージャーが静止状況になった(MQGET 呼び出しに MQGMO_FAIL_IF QUIESCING が指定されている)ために、待機が取り消されました。

MQEC_CONNECTION QUIESCING

接続が静止状態に入った(MQGET 呼び出しに MQGMO_FAIL_IF QUIESCING が指定されている)ために、待機が取り消されました。

フィールドの初期値は、環境により決まります。

- z/OS では、初期値はヌル・ポインターです。
- それ以外のすべての環境では、初期値は 0 です。

Signal2 (MQLONG)

これは、MQGMO_SET_SIGNAL オプションと組み合わせて指定された場合にだけ使用される入力フィールドです。これは、予約フィールドです。したがって、値に意味はありません。

このフィールドの初期値は 0 です。

ResolvedQName (MQCHAR48)

これは、メッセージが取り出されたキューのローカル名に対してキュー・マネージャーが設定した出力フィールドであり、ローカル・キュー・マネージャーに対して定義されます。次の場合には、キューをオープンするのに使用された名前とは異なります。

- 別名キューがオープンされた。この場合、別名が解決したローカル・キューの名前が戻されます。
- モデル・キューがオープンされた。この場合、動的なローカル・キューの名前が戻されます。

このフィールドの長さは MQ_Q_NAME_LENGTH によって指定されます。このフィールドの初期値は、C 言語ではヌル・ストリングであり、他のプログラミング言語では 48 桁のブランク文字です。

MatchOptions (MQLONG)

このオプションを指定すると、MQGET 呼び出しで戻すメッセージを選択するのに **MsgDesc** パラメーター内のどのフィールドを使用するかをアプリケーションで決めることができます。アプリケーションは、このフィールドに必要なオプションを設定してから、**MsgDesc** パラメーター内の対応するフィールドを、それらのフィールドに必要な値に設定します。MQMD にこれらの値が入っているメッセージのみが、MQGET 呼び出しで **MsgDesc** パラメーターを使用した場合の検索の候補になります。戻すメッセージの選択時には、対応した一致オプションが指定されていないフィールドは無視されます。MQGET 呼び出しで選択基準を指定しない(つまり、どのメッセージも受け入れ可能にする)場合は、*MatchOptions* を MQMO_NONE に設定します。

- z/OS では、使用される選択基準は、キューに使用される索引のタイプによって制限されます。詳細は、**IndexType** キュー属性を参照してください。

MQGMO_LOGICAL_ORDER を指定すると、特定のメッセージだけが次の MQGET 呼び出しで戻されることになります。

- 現行のグループまたは論理メッセージがない場合は、*MsgSeqNumber* が 1 に等しく、*Offset* が 0 に等しいメッセージのみが戻されます。この場合には、以下の一致オプションを 1 つ以上使用することにより、戻されるメッセージのうちのどれが戻されるかを選定できます。

- MQMO_MATCH_MSG_ID
- MQMO_MATCH_CORREL_ID
- MQMO_MATCH_GROUP_ID

- 現行のグループまたは論理メッセージが存在している場合は、グループ内の次のメッセージまたは論理メッセージ内の次のセグメントだけが戻されることになります。これは、MQMO_* オプションを指定しても変更できません。

上記のどちらの場合も、該当しない一致オプションを指定できますが、このとき必ず **MsgDesc** パラメーター内の関連フィールドの値と、返されるメッセージ内の対応したフィールドの値は一致していなければなりません。この条件が満たされないと、呼び出しは失敗し、理由コード MQRC_MATCH_OPTIONS_ERROR が返されます。

MQGMO_MSG_UNDER_CURSOR または MQGMO_BROWSE_MSG_UNDER_CURSOR のどちらかを指定すると、*MatchOptions* は無視されます。

メッセージ・プロパティーに基づくメッセージの取得では、一致オプションは使用されません。詳しくは、464 ページの『[SelectionString \(MQCHARV\)](#)』を参照してください。

以下の一致オプションを 1 つ以上指定できます。

MQMO_MATCH_MSG_ID

取得対象のメッセージは、MQGET 呼び出しのパラメータ **MsgDesc** 内のフィールド *MsgId* の値と一致するメッセージ識別子を持つ必要があります。これは、適用される他の一致 (例えば、相関 ID) に加えて一致している必要があります。

このオプションを省略した場合、パラメータ **MsgDescMsgId** 内のフィールドは無視され、任意のメッセージ識別子が一致します。

注: MQMI_NONE は、メッセージの MQMD 内のどのメッセージ ID とも一致する特殊なメッセージ ID です。したがって、MQMO_MATCH_MSG_ID を MQMI_NONE と組み合わせて指定しても、MQMO_MATCH_MSG_ID を指定しなくても同じことです。

MQMO_MATCH_CORREL_ID

取得対象のメッセージは、MQGET 呼び出しのパラメータ **MsgDesc** 内のフィールド *CorrelId* の値と一致する相関識別子を持つ必要があります。これは、適用される他の一致 (例えば、メッセージ ID) に加えて一致している必要があります。

このオプションを省略した場合、パラメータ **MsgDescCorrelId** 内のフィールドは無視され、任意の相関識別子が一致します。

注: MQCI_NONE は、メッセージの MQMD 内のどの相関 ID とも一致する特殊な相関 ID です。したがって、MQMO_MATCH_CORREL_ID を MQCI_NONE と組み合わせて指定しても、MQMO_MATCH_CORREL_ID を指定しなくても同じことです。

MQMO_MATCH_GROUP_ID

取得対象のメッセージは、MQGET 呼び出しのパラメータ **MsgDesc** 内のフィールド *GroupId* の値と一致するグループ識別子を持つ必要があります。これは、適用される他の一致 (例えば、相関 ID) に加えて一致している必要があります。

このオプションを省略した場合、パラメータ **MsgDescGroupId** 内のフィールドは無視され、任意のグループ識別子が一致します。

注: MQGI_NONE は、メッセージの MQMD 内のどのグループ ID とも一致する特殊なグループ ID です。したがって、MQMO_MATCH_GROUP_ID を MQGI_NONE と組み合わせて指定しても、MQMO_MATCH_GROUP_ID を指定しなくても同じことです。

MQMO_MATCH_MSG_SEQ_NUMBER

取得対象のメッセージは、MQGET 呼び出しのパラメータ **MsgDesc** 内のフィールド *MsgSeqNumber* の値と一致するメッセージシーケンス番号を持つ必要があります。これは、適用される他の一致 (例えば、グループ ID) に加えて一致している必要があります。

このオプションを省略した場合、パラメータ **MsgDesc** 内のフィールド *MsgSeqNumber* は無視され、任意のメッセージシーケンス番号が一致します。

MQMO_MATCH_OFFSET

取得対象のメッセージは、MQGET 呼び出しのパラメータ **MsgDesc** 内のフィールド *Offset* の値と一致するオフセットを持つ必要があります。これは、適用される他の一致 (例えば、メッセージ・シーケンス番号) に加えて一致している必要があります。

このオプションを省略した場合、パラメータ **MsgDesc** 内のフィールド *Offset* は無視され、任意のオフセットが一致します。

- このオプションは、z/OS ではサポートされていません。

MQMO_MATCH_MSG_TOKEN

検索するメッセージのメッセージ・トークンは、MQGET 呼び出しで指定される MQGMO 構造の *MsgToken* フィールドの値と一致している必要があります。

このオプションはすべてのローカル・キューで指定できます。 *IndexType* が MQIT_MSG_TOKEN のキュー (WLM 管理キュー) にこれを指定する場合、MQMO_MATCH_MSG_TOKEN と共に他の一致オプションを指定することができません。

MQMO_MATCH_MSG_TOKEN は、MQGMO_WAIT や MQGMO_SET_SIGNAL と共に指定することはできません。 MQIT_MSG_TOKEN の *IndexType* があるキューにメッセージが到着するまでアプリケーションを待機させる場合は、MQMO_NONE を指定します。

このオプションを省略すると、MQGMO の *MsgToken* フィールドは無視され、すべてのメッセージ・トークンが一致します。

上記で説明されたオプションをいずれも指定しない場合、以下のオプションを使用できます。

MQMO_NONE

戻されるメッセージを選択するときに一致条件を使用しません。キューに入っているメッセージがすべて取り出せるようになります (ただし、MQGMO_ALL_MSGS_AVAILABLE、MQGMO_ALL_SEGMENTS_AVAILABLE、および MQGMO_COMPLETE_MSG オプションで制御できます)。

MQMO_NONE は、プログラム・ドキュメンテーションの援助機能です。このオプションを他の MQMO_* オプションと一緒に使用することは意図されていませんが、値がゼロであるため、そのように使用しても、それを検出することはできません。

これは入力フィールドです。このフィールドの初期値は、MQMO_MATCH_MSG_ID と MQMO_MATCH_CORREL_ID を組み合わせたものです。 *Version* が MQGMO_VERSION_2 より小さい場合、このフィールドは無視されます。

注: *MatchOptions* フィールドの初期値は、初期の MQSeries® キュー・マネージャーとの互換性を維持するために定義されています。ただし、選択基準を使用せずにキューから一連のメッセージを読み取る場合、この初期値では、アプリケーションが各 MQGET 呼び出しの前におよび *CorrelId* フィールド *MsgId* を MQMI_NONE および MQCI_NONE にリセットする必要があります。 *Version* に MQGMO_VERSION_2 を、さらに *MatchOptions* に MQMO_NONE を設定しておけば、*MsgId* および *CorrelId* を各呼び出しごとに再設定しなくても済みます。

GroupStatus (MQCHAR)

このフラグは、取り出されたメッセージが 1 つのグループに属しているかどうかを示します。

以下の値がどれか 1 つ含まれています。

MQGS_NOT_IN_GROUP

メッセージは 1 つのグループに属していない。

MQGS_MSG_IN_GROUP

メッセージは1つのグループに属しているが、グループの最後にあるものではない。

MQGS_LAST_MSG_IN_GROUP

メッセージはグループの最後にあるものである。

この値は、グループに属しているメッセージが1つしかない場合にも戻されます。

これは出力フィールドです。このフィールドの初期値は、MQGS_NOT_IN_GROUP です。Version が MQGMO_VERSION_2 より小さい場合、このフィールドは無視されます。

SegmentStatus (MQCHAR)

これは、取り出されたメッセージが論理メッセージの1つのセグメントであるかどうかを示すフラグです。以下の値がどれか1つ含まれています。

MQSS_NOT_A_SEGMENT

メッセージは1つのセグメントではない。

MQSS_SEGMENT

メッセージは1つのセグメントであるが、論理メッセージの最後のセグメントではない。

MQSS_LAST_SEGMENT

メッセージは論理メッセージの最後のセグメントである。

この値は、論理メッセージを構成しているセグメントが1つしかない場合にも戻されます。

z/OS では、キュー・マネージャーはこのフィールドを常に MQSS_NOT_A_SEGMENT に設定します。

これは出力フィールドです。このフィールドの初期値は、MQSS_NOT_A_SEGMENT です。Version が MQGMO_VERSION_2 より小さい場合、このフィールドは無視されます。

Segmentation (MQCHAR)

これは、取り出されたメッセージに対して、さらにセグメント化できるかどうかを示すフラグです。以下の値がどれか1つ含まれています。

MQSEG_INHIBITED

セグメント化できない。

MQSEG_ALLOWED

セグメント化できます。

z/OS では、キュー・マネージャーはこのフィールドを常に MQSEG_INHIBITED に設定します。

これは出力フィールドです。このフィールドの初期値は、MQSEG_INHIBITED です。Version が MQGMO_VERSION_2 より小さい場合、このフィールドは無視されます。

Reserved1 (MQCHAR)

これは予約フィールドです。このフィールドの初期値は、ブランク文字です。Version が MQGMO_VERSION_2 より小さい場合、このフィールドは無視されます。

MsgToken (MQBYTE16)

MsgToken フィールド - MQGMO 構造体。このフィールドは、キュー・マネージャーがメッセージを一意的に識別するために使用されます。

これは、キューのメッセージを一意的に識別するためにキュー・マネージャーが生成するバイト・ストリングです。メッセージ・トークンは、メッセージが初めてキュー・マネージャーに入れられたときに生成され、キュー・マネージャーが再始動されない場合はそのメッセージがキュー・マネージャーから永久に削除されるまで、メッセージと一緒に保持されます。

メッセージをキューから除去すると、そのメッセージのインスタンスを識別していた *MsgToken* が有効でなくなり、再利用できなくなります。キュー・マネージャーを再始動する場合、再始動の前にキューのメッセージを識別していた *MsgToken* は、再始動後に有効でなくなることがあります。ただし、異なるメッ

セージ・インスタンスを識別するために *MsgToken* が再利用されることはありません。キュー・マネージャーによって *MsgToken* が生成されます。これは外部アプリケーションには表示されません。

バージョン 3 以降の MQGMO が提供されている MQGET の呼び出しによってメッセージが戻されると、キューのメッセージを識別する *MsgToken* がキュー・マネージャーによって MQGMO に戻されます。この例外が 1 つあります。同期点の外側のキューからメッセージを除去する場合、キュー・マネージャーが *MsgToken* を戻さないことがあります。後続の MQGET 呼び出しで戻されるメッセージを識別するためにこれを使用できないためです。*MsgToken* は、後続の MQGET 呼び出しのメッセージを参照する場合にのみアプリケーションで使用します。

MsgToken が提供され、*MatchOption* MQMO_MATCH_MSG_TOKEN が指定されて、MQGMO_MSG_UNDER_CURSOR と MQGMO_BROWSE_MSG_UNDER_CURSOR のどちらも指定されない場合には、その *MsgToken* によって識別されるメッセージだけを戻すことができます。このオプションは INDXTYPE に関係なくすべてのローカル・キューで有効です。z/OS の場合は INDXTYPE(MSGTOKEN) の使用をワークロード・マネージャー (WLM) キューだけに限定しなければなりません。

MatchOptions が他にも指定される場合は、そのすべてが検査されます。一致しない場合、MQRC_NO_MSG_AVAILABLE が戻されます。MQGMO_BROWSE_NEXT が MQMO_MATCH_MSG_TOKEN と共にコード化される場合、*MsgToken* によって識別されるメッセージは、呼び出し側のハンドルのブラウザ・カーソルを超える場合にのみ戻されます。

MQGMO_MSG_UNDER_CURSOR または MQGMO_BROWSE_MSG_UNDER_CURSOR が指定される場合、MQMO_MATCH_MSG_TOKEN は無視されます。

次の読み取りメッセージ・オプションを指定する場合、MQMO_MATCH_MSG_TOKEN は無効になります。

- MQGMO_WAIT
- MQGMO_SET_SIGNAL

MQMO_MATCH_MSG_TOKEN を指定する MQGET 呼び出しでは、バージョン 3 以降の MQGMO を呼び出しに対して提供する必要があります。そうしないと、MQRC_WRONG_GMO_VERSION が戻されます。

この時点で *MsgToken* が有効でないと、MQCC_FAILED が MQRC_NO_MSG_AVAILABLE と共に戻されます (それ以外に別のエラーがない場合)。

ReturnedLength (MQLONG)

これは出力フィールドであり、キュー・マネージャーは、MQGET 呼び出しの **Buffer** パラメーターで戻されたメッセージ・データ長 (バイト単位) に設定します。キュー・マネージャーがこの機能をサポートしていない場合は、*ReturnedLength* は値 MQRL_UNDEFINED に設定されます。

メッセージがエンコードや文字セット間で変換された場合、メッセージ・データのサイズが変わることがあります。MQGET 呼び出しからの戻り値は、次のようになります。

- *ReturnedLength* が MQRL_UNDEFINED でない場合は、戻されたメッセージ・データのバイト数が *ReturnedLength* で示されます。
- が MQRL_UNDEFINED の値を持つ *ReturnedLength* 場合、返されるメッセージデータのバイト数は通常、この *BufferLength* 小さい方の値によって決まりますが、MQGET 呼び出しが理由コード *DataLength* MQRC_TRUNCATED_MSG_ACCEPTED で完了した場合、これより少なくなる可能性があります。この場合、**Buffer** パラメーター内の重要でないバイトはヌルに設定されます。

以下のような特殊値が定義されます。

MQRL_UNDEFINED

戻されたデータの長さが定義されていない。

z/OS では、*ReturnedLength* フィールドに戻される値は常に MQRL_UNDEFINED です。

このフィールドの初期値は、MQRL_UNDEFINED です。*Version* が MQGMO_VERSION_3 より前の場合、このフィールドは無視されます。

Reserved2 (MQLONG)

これは予約フィールドです。このフィールドの初期値は、ブランク文字です。Version が MQGMO_VERSION_4 より前の場合、このフィールドは無視されます。

MsgHandle (MQHMSG)

MQGMO_PROPERTIES_AS_Q_DEF オプションが指定され、かつキュー **PropertyControl** 属性が MQPROP_FORCE_MQRFH2 に設定されていない場合、これはキューから取得されるメッセージのプロパティで埋められるメッセージへのハンドルです。このハンドルは、MQCRTMH 呼び出しによって作成されます。ハンドルに既に関連付けられているプロパティは、メッセージを取り出す前にすべてクリアされます。

以下のような値も指定することができます。

MQHM_NONE

メッセージ・ハンドルは提供されません。

MQGET 呼び出し上でメッセージ記述子は必須ではありません。有効なメッセージ・ハンドルが提供され、出力上でメッセージ・プロパティを組み込むのに使用される場合は、メッセージ・ハンドルに関連付けられているメッセージ記述子が入力フィールド用に使用されます。

メッセージ記述子を MQGET 呼び出しに指定すると、メッセージ・ハンドルに関連付けられているメッセージ記述子より常に優先します。

MQGMO_PROPERTIES_FORCE_MQRFH2 が指定されている場合、または MQGMO_PROPERTIES_AS_Q_DEF が指定され、かつキュー属性 **PropertyControl** が MQPROP_FORCE_MQRFH2 である場合、メッセージ記述子パラメータが指定されていないと、呼び出しは理由コード MQRC_MD_ERROR で失敗する。

MQGET 呼び出しから戻る際に、このメッセージ・ハンドルに関連付けられているプロパティとメッセージ記述子は更新され (メッセージ記述子が MQGET 呼び出し上で指定されている場合はメッセージ記述子も)、取り出されたメッセージの状態を反映します。その後、MQINQMP 呼び出しを使用して、メッセージのプロパティを照会できます。

拡張メッセージ記述子 (ある場合) を除いて、MQINQMP 呼び出しを使用して照会できるプロパティはメッセージ・データには含まれません。キュー上のメッセージがメッセージ・データ中のプロパティに含まれていた場合は、データがアプリケーションに戻る前にメッセージ・データから除去されます。

メッセージ・ハンドルが提供されていないか、Version が MQGMO_VERSION_4 より前の場合は、MQGET 呼び出し上で有効なメッセージ記述子を提供しなければなりません。メッセージのプロパティ (メッセージ記述子に含まれるものを除く) は、MQGMO 構造体のプロパティオプションとキュー **PropertyControl** 属性の値に従い、メッセージデータ内で返されます。

このプロパティは常に入力フィールドです。フィールドの初期値は、MQHM_NONE です。このフィールドは、MQGMO_VERSION_4Version 未満の場合に無視されます。

MQIIH - IMS 情報ヘッダー

MQIIH 構造体は、IMS ブリッジ経由で IMS に送信されるメッセージのヘッダー情報を表します。サポートされている IBM MQ プラットフォームであれば、MQIIH 構造を含むメッセージを作成して送信できますが、ブリッジ IMS を利用できるのは IBM MQ for z/OS キューマネージャのみです。したがって、メッセージが非キュー z/OS マネージャから IMS に到達するためには、メッセージがルーティングされる少なくとも 1 つの z/OS キューマネージャを、あなたのキューマネージャネットワークが包含している必要があります。

可用性

すべての IBM MQ システムおよび IBM MQ クライアント。

形式名

MQFMT_IMS

文字セットとエンコード

MQIIH 構造体およびアプリケーションメッセージデータに使用される文字セットとエンコーディングには特別な条件が適用されます：

- IMS ブリッジ・キューを所有するキュー・マネージャーに接続するアプリケーションは、キュー・マネージャーの文字セットとエンコードで記述されている MQIIH 構造体を提供する必要があります。その理由は、この場合には MQIIH 構造体のデータ変換を実行されないからです。
- 他のキュー・マネージャーに接続するアプリケーションでは、サポートされている文字セットとエンコードで記述されている MQIIH 構造体を提供することができます。その理由は、IMS ブリッジ・キューを所有するキュー・マネージャーに接続する受信メッセージ・チャンネル・エージェントが、MQIIH 構造体を変換するからです。
- MQIIH 構造体の後続くアプリケーション・メッセージ・データは MQIIH 構造体と同じ文字セットとエンコードでなければなりません。MQIIH 構造体の *CodedCharSetId* フィールドおよび *Encoding* フィールドを使用して、そのアプリケーション・メッセージ・データの文字セットとエンコードを指定しないでください。

データがキュー・マネージャーにサポートされる組み込み形式でない場合、アプリケーション・メッセージ・データを変換するために、ユーザーはデータ変換出口を提供することが必要です。

フィールド

注：次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 512. MQIIH における MQIIH のフィールド		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
<u>StrucId</u> (構造識別子)	MQIIH_STRUC_ID	'IIH'
<u>バージョン</u> (構造バージョン番号)	MQIIH_VERSION_1	1
<u>StrucLength</u> (MQIIH 構造の長さ)	MQIIH_LENGTH_1	84
<u>エンコーディング</u> (予約済み - 参照 369 ページの『文字セットとエンコード』)	なし	0
<u>CodedCharSetId</u> (予約済み - 参照 369 ページの『文字セットとエンコード』)	なし	0
<u>フォーマット</u> (MQ データ形式名 (MQIIH の後に続く))	MQFMT_NONE	ブランク
<u>フラグ</u> (フラグ)	MQIIH_NONE	0
<u>LTermOverride</u> (論理ターミナルオーバーライド)	なし	ブランク
<u>MFSMapName</u> (メッセージ形式サービスマップ名)	なし	ブランク
<u>ReplyToFormat</u> (MQ 返信メッセージのフォーマット名)	MQFMT_NONE	ブランク
<u>認証器</u> (RACF® パスワードまたはパスチケット)	MQIAUT_NONE	ブランク
<u>TranInstanceId</u> (トランザクションインスタンス識別子)	MQITII_NONE	Null
<u>TranState</u> (トランザクション状態)	MQITS_NOT_IN_CONVE RSATION	' '
<u>CommitMode</u> (コミットモード)	MQICM_COMMIT_THEN _SEND	'0'
<u>SecurityScope</u> (セキュリティスコープ)	MQISS_CHECK	'C'

表 512. MQIIH における MQIIH のフィールド (続き)

フィールド名と説明	定数の名前	定数の初期値 (存在する場合)
<u>予約済み</u> (予約済み)	なし	' '

注:

- 記号' 'は、単一のブランク文字を表します。
- C プログラミング言語では、マクロ変数 MQIIH_DEFAULT には、表にリストされている値が含まれています。この変数を以下の方法で使用すると、構造体のフィールドに初期値を設定できます。

```
MQIIH MyIIH = {MQIIH_DEFAULT};
```

言語ごとの宣言

MQIIH の C 宣言

```
typedef struct tagMQIIH MQIIH;
struct tagMQIIH {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQLONG     StrucLength;      /* Length of MQIIH structure */
    MQLONG     Encoding;         /* Reserved */
    MQLONG     CodedCharSetId;   /* Reserved */
    MQCHAR8    Format;           /* MQ format name of data that follows
                                MQIIH */
    MQLONG     Flags;            /* Flags */
    MQCHAR8    LTermOverride;    /* Logical terminal override */
    MQCHAR8    MFSMapName;       /* Message format services map name */
    MQCHAR8    ReplyToFormat;    /* MQ format name of reply message */
    MQCHAR8    Authenticator;    /* RACF password or passticket */
    MQBYTE16   TranInstanceId;   /* Transaction instance identifier */
    MQCHAR     TranState;        /* Transaction state */
    MQCHAR     CommitMode;       /* Commit mode */
    MQCHAR     SecurityScope;    /* Security scope */
    MQCHAR     Reserved;        /* Reserved */
};
```

MQIIH の COBOL 宣言

```
**      MQIIH structure
10 MQIIH.
**      Structure identifier
15 MQIIH-STRUCID          PIC X(4).
**      Structure version number
15 MQIIH-VERSION         PIC S9(9) BINARY.
**      Length of MQIIH structure
15 MQIIH-STRUCLength     PIC S9(9) BINARY.
**      Reserved
15 MQIIH-ENCODING        PIC S9(9) BINARY.
**      Reserved
15 MQIIH-CODEDCHARSETID  PIC S9(9) BINARY.
**      MQ format name of data that follows MQIIH
15 MQIIH-FORMAT          PIC X(8).
**      Flags
15 MQIIH-FLAGS           PIC S9(9) BINARY.
**      Logical terminal override
15 MQIIH-LTERMOverride   PIC X(8).
**      Message format services map name
15 MQIIH-MFSMAPNAME      PIC X(8).
**      MQ format name of reply message
15 MQIIH-REPLYTOFORMAT   PIC X(8).
**      RACF password or passticket
15 MQIIH-AUTHENTICATOR   PIC X(8).
**      Transaction instance identifier
15 MQIIH-TRANINSTANCEID  PIC X(16).
**      Transaction state
15 MQIIH-TRANSTATE       PIC X.
**      Commit mode
15 MQIIH-COMMITMODE      PIC X.
**      Security scope
```

```

15 MQIIH-SECURITYSCOPE PIC X.
**   Reserved
15 MQIIH-RESERVED      PIC X.

```

MQIIH の PL/I 宣言

```

dcl
1 MQIIH based,
3 StrucId      char(4),          /* Structure identifier */
3 Version      fixed bin(31),    /* Structure version number */
3 StrucLength  fixed bin(31),    /* Length of MQIIH structure */
3 Encoding     fixed bin(31),    /* Reserved */
3 CodedCharSetId fixed bin(31),  /* Reserved */
3 Format        char(8),          /* MQ format name of data that follows
                                   MQIIH */
3 Flags        fixed bin(31),    /* Flags */
3 LTermOverride char(8),          /* Logical terminal override */
3 MFSMapName   char(8),          /* Message format services map name */
3 ReplyToFormat char(8),          /* MQ format name of reply message */
3 Authenticator char(8),          /* RACF password or passticket */
3 TranInstanceId char(16),        /* Transaction instance identifier */
3 TranState    char(1),          /* Transaction state */
3 CommitMode    char(1),          /* Commit mode */
3 SecurityScope char(1),          /* Security scope */
3 Reserved     char(1);          /* Reserved */

```

MQIIH の高水準アセンブラ宣言

MQIIH	DSECT	
MQIIH_STRUCID	DS CL4	Structure identifier
MQIIH_VERSION	DS F	Structure version number
MQIIH_STRUCLNGTH	DS F	Length of MQIIH structure
MQIIH_ENCODING	DS F	Reserved
MQIIH_CODEDCHARSETID	DS F	Reserved
MQIIH_FORMAT	DS CL8	MQ format name of data that follows
*		MQIIH
MQIIH_FLAGS	DS F	Flags
MQIIH_LTERM_OVERRIDE	DS CL8	Logical terminal override
MQIIH_MFSMAPNAME	DS CL8	Message format services map name
MQIIH_REPLYTOFORMAT	DS CL8	MQ format name of reply message
MQIIH_AUTHENTICATOR	DS CL8	RACF password or passticket
MQIIH_TRANINSTANCEID	DS XL16	Transaction instance identifier
MQIIH_TRANSTATE	DS CL1	Transaction state
MQIIH_COMMITMODE	DS CL1	Commit mode
MQIIH_SECURITYSCOPE	DS CL1	Security scope
MQIIH_RESERVED	DS CL1	Reserved
*		
MQIIH_LENGTH	EQU *-MQIIH	
	ORG MQIIH	
MQIIH_AREA	DS CL(MQIIH_LENGTH)	

MQIIH の Visual Basic 宣言

```

Type MQIIH
StrucId      As String*4 'Structure identifier'
Version      As Long      'Structure version number'
StrucLength  As Long      'Length of MQIIH structure'
Encoding     As Long      'Reserved'
CodedCharSetId As Long    'Reserved'
Format       As String*8  'MQ format name of data that follows MQIIH'
Flags       As Long      'Flags'
LTermOverride As String*8 'Logical terminal override'
MFSMapName   As String*8  'Message format services map name'
ReplyToFormat As String*8 'MQ format name of reply message'
Authenticator As String*8 'RACF password or passticket'
TranInstanceId As MQBYTE16 'Transaction instance identifier'
TranState     As String*1  'Transaction state'
CommitMode    As String*1  'Commit mode'
SecurityScope As String*1  'Security scope'
Reserved      As String*1  'Reserved'
End Type

```

MQIIH のフィールド詳細

MQIIH IMS 情報ヘッダー構造は、ブリッジ IMS を IMS 介して送信されるメッセージのヘッダー情報を記述する。

フィールド

注: 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 513. MQIIH における MQIIH のフィールド		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
StrucId (構造識別子)	MQIIH_STRUC_ID	'IIH↵'
バージョン (構造バージョン番号)	MQIIH_VERSION_1	1
StrucLength (MQIIH 構造の長さ)	MQIIH_LENGTH_1	84
エンコーディング (予約済み - 参照 369 ページの『文字セットとエンコード』)	なし	0
CodedCharSetId (予約済み - 参照 369 ページの『文字セットとエンコード』)	なし	0
フォーマット (MQ データ形式名 (MQIIH の後に続く))	MQFMT_NONE	ブランク
フラグ (フラグ)	MQIIH_NONE	0
LTermOverride (論理ターミナルオーバーライド)	なし	ブランク
MFSTMapName (メッセージ形式サービスマップ名)	なし	ブランク
ReplyToFormat (MQ 返信メッセージのフォーマット名)	MQFMT_NONE	ブランク
認証器 (RACF パスワードまたはパスチケット)	MQIAUT_NONE	ブランク
TranInstanceId (トランザクションインスタンス識別子)	MQITII_NONE	Null
TranState (トランザクション状態)	MQITS_NOT_IN_CONVE RSATION	'↵'
CommitMode (コミットモード)	MQICM_COMMIT_THEN _SEND	'0'
SecurityScope (セキュリティスコープ)	MQISS_CHECK	'C'
予約済み (予約済み)	なし	'↵'
注: - 記号↵は、単一のブランク文字を表します。 - C プログラミング言語では、マクロ変数 MQIIH_DEFAULT には、表にリストされている値が含まれています。この変数を以下の方法で使用すると、構造体のフィールドに初期値を設定できます。 <pre>MQIIH MyIIH = {MQIIH_DEFAULT};</pre>		

StrucId (MQCHAR4)

これは情報 IMS ヘッド構造の構造識別子です。常に入力フィールドです。その値は MQIIH_STRUC_ID です。

値は次のものでなければなりません。

MQIIH_STRUC_ID

IMS 情報ヘッダー構造体の ID。

C プログラミング言語では、定数 MQIIH_STRUC_ID_ARRAY も定義されている。これは MQIIH_STRUC_ID と同じ値を持ちますが、文字列ではなく文字の配列です。

Version (MQLONG)

これは構造体のバージョン番号です。値は次のものでなければなりません。

MQIIH_VERSION_1

IMS 情報ヘッダー構造体のバージョン番号。

以下の定数は、現行バージョンのバージョン番号を指定しています。

MQIIH_CURRENT_VERSION

IMS 情報ヘッダー構造体の現行バージョン。

フィールドの初期値は、MQIIH_VERSION_1 です。

StrucLength (MQLONG)

これは MQIIH 構造体の長さです。値は次のものでなければなりません。

MQIIH_LENGTH_1

IMS 情報ヘッダー構造体の長さ。

フィールドの初期値は、MQIIH_LENGTH_1 です。

Encoding (MQLONG)

これは、予約フィールドです。したがって、値に意味はありません。このフィールドの初期値は 0 です。

MQIIH 構造に従うサポート対象構造体のエンコーディングは、MQIIH 構造体自体と同じであり、先行する MQ ヘッダーから取得される。

CodedCharSetId (MQLONG)

これは、予約フィールドです。したがって、値に意味はありません。このフィールドの初期値は 0 です。

MQIIH 構造に従うサポート対象構造体の文字セット ID は、MQIIH 構造体自体のものと同一であり、先行する MQ ヘッダーから取得される。

Format (MQCHAR8)

これは、MQIIH 構造体の後続くデータの MQ 形式名を指定します。

MQPUT または MQPUT1 呼び出しでは、アプリケーションは、このフィールドをデータに適切な値に設定する必要があります。

このフィールドの長さは MQ_FORMAT_LENGTH によって指定されます。このフィールドの初期値は MQFMT_NONE です。

フラグ (MQLONG)

フラグ値は、以下のものでなければなりません。

MQIIH_NONE

フラグなし。

MQIIH_PASS_EXPIRATION

応答メッセージには以下のものが含まれています。

- 要求メッセージと同じ満了レポート・オプション
- 要求メッセージからの残りの満了時間 (ブリッジの処理時間は未調整)

この値が設定されていないと、満了時間は 無制限 に設定されます。

MQIIH_REPLY_FORMAT_NONE

応答の MQIIH.Format フィールドを MQFMT_NONE に設定します。

MQIIH_IGNORE_PURG

OTMA 接頭部で TMAMIPRG 標識を設定し、OTMA に、TP PCB for CM0 トランザクションで PURG 呼び出しを無視するように要求します。

MQIIH_CM0_REQUEST_RESPONSE

コミット・モード 0 (CM0) のトランザクションの場合、このフラグは TMAMHRSP 標識を OTMA 接頭部に設定します。この標識を設定することによって、元の IMS アプリケーション・プログラムが IOPCB に応答せず、メッセージが別のトランザクションに切り替わることもない場合に、OTMA/IMS が DFS2082 RESPONSE MODE TRANSACTION TERMINATED WITHOUT REPLY メッセージを生成するように要求します。

フィールドの初期値は、MQIIH_NONE です。

LTermOverride (MQCHAR8)

論理端末の指定変更。IO PCB フィールド内に配置されます。これはオプションです。指定されていない場合は、TPIPE 名が使用されます。最初のバイトがブランクまたはヌルの場合は、無視されます。

このフィールドの長さは MQ_LTERM_OVERRIDE_LENGTH によって指定されます。このフィールドの初期値は 8 個のブランク文字です。

MFSMapName (MQCHAR8)

メッセージ形式サービスのマップ名。IO PCB フィールド内に配置されます。フィールドはオプションです。入力では MID を表し、出力では MOD を表します。最初のバイトがブランクまたはヌルの場合は、無視されます。

このフィールドの長さは MQ_MFS_MAP_NAME_LENGTH によって指定されます。このフィールドの初期値は 8 個のブランク文字です。

ReplyToFormat (MQCHAR8)

これは、現行メッセージに回答して送信される応答メッセージの MQ 形式名です。このフィールドの長さは MQ_FORMAT_LENGTH によって指定されます。このフィールドの初期値は MQFMT_NONE です。

MQGMO_CONVERT を使用して応答メッセージ内のデータを変換するには、MQIIH.replyToFormat=MQFMT_STRING または MQIIH.replyToFormat=MQFMT_IMS_VAR_STRING のいずれかを指定します。これらのフィールドの使用方法については、[413 ページの『Format \(MQCHAR8\)』](#)を参照してください。

デフォルト値 (MQIIH.replyToFormat=MQFMT_NONE) が要求メッセージに指定されている場合は、MQGMO_CONVERT を使用して応答メッセージを取得しても、データ変換は実行されません。

Authenticator (MQCHAR8)

これは、RACF パスワードまたはパスチケットです。このフィールドはオプションです。指定した場合は、MQMD セキュリティー・コンテキスト内のユーザー ID とともに、セキュリティ・コンテキストを提供するために IMS に送られる UTOKEN の作成に使用されます。指定しなかった場合は、ユーザー ID が検証なしで使用されます。これは、RACF スイッチの設定に左右されます。スイッチを設定するには、オーセンティケーターが必要な場合があります。

最初のバイトがブランクまたはヌルの場合は、フィールドは無視されます。以下のような特別な値を使用することができます。

MQIAUT_NONE

認証なし。

C プログラミング言語では、定数 MQIAUT_NONE_ARRAY も定義されます。これは、MQIAUT_NONE と同じ値を持っていますが、ストリングではなく文字の配列です。

このフィールドの長さは MQ_AUTHENTICATOR_LENGTH によって指定されます。フィールドの初期値は、MQIAUT_NONE です。

TranInstanceId (MQBYTE16)

これはトランザクション・インスタンス ID です。このフィールドは IMS からの出力メッセージが使用するもので、したがって、最初の入力時には無視されます。TranState に MQITS_IN_CONVERSATION を設定する場合は、IMS がメッセージを適切な会話に関連付けられるように、これを次の入力と後続のすべての入力で指定する必要があります。次の特殊値を使用することができます。

MQITII_NONE

トランザクション・インスタンス ID なし。

C プログラミング言語では、定数 MQITII_NONE_ARRAY も定義されます。これは、MQITII_NONE と同じ値を持っていますが、ストリングではなく文字の配列です。

このフィールドの長さは MQ_TRAN_INSTANCE_ID_LENGTH によって指定されます。フィールドの初期値は、MQITII_NONE です。

TranState (MQCHAR)

これは IMS 会話の状態を示します。最初の入力時には会話は存在しないので、このフィールドは最初の入力では無視されます。後続の入力では、会話が活動状態か否かを示します。出力では、IMS により設定されます。値は次のいずれかでなければなりません。

MQITS_IN_CONVERSATION

会話中。

MQITS_NOT_IN_CONVERSATION

会話中でない。

MQITS_ARCHITECTED

トランザクション状態のデータを構造化形式で戻す。

この値が使用できるのは、IMS /DISPLAY TRAN コマンドの場合に限られます。これを指定すると、トランザクション状態のデータが、文字形式ではなく IMS 構造化形式で戻されます。  詳細については、[を参照してください トランザクションプログラム IMS の作成を通じて IBM MQ。](#)

フィールドの初期値は、MQITS_NOT_IN_CONVERSATION です。

CommitMode (MQCHAR)

これは IMS コミット・モードです。コミットモード IMS の詳細については、OTMA リファレンスを参照してください。値は次のいずれかでなければなりません。

MQICM_COMMIT_THEN_SEND

コミット後に送信。

このモードは、出力の二重キューイングを暗黙指定しますが、領域占有時間は短くなります。高速パス・トランザクションおよび会話型トランザクションは、実行できません。

MQICM_SEND_THEN_COMMIT

送信後コミット。

コミット・モード MQICM_SEND_THEN_COMMIT の結果として開始された IMS トランザクションは、IMS システム定義におけるトランザクションの定義内容 (TRANSACT マクロの MSGTYPE パラメーター) に関係なく、すべて RESPONSE モードで実行されます。また、これは、トランザクション切り替えによって開始されたトランザクションにも適用されます。

フィールドの初期値は、MQICM_COMMIT_THEN_SEND です。

SecurityScope (MQCHAR)

これは、必要な IMS セキュリティー処理を示します。以下の値が定義されます。

MQISS_CHECK

チェック・セキュリティ有効範囲。ACEE は、従属領域でなく、制御領域に作成されます。

MQISS_FULL

完全セキュリティ有効範囲。キャッシュ ACEE は制御領域に作成され、非キャッシュ ACEE は従属領域に作成されます。MQISS_FULL を使用する場合は、ACEE を作成するユーザー ID から従属領域で使用するリソースにアクセスできるようにします。

このフィールドに MQISS_CHECK と MQISS_FULL のどちらも指定されていない場合は、MQISS_CHECK と見なされます。

フィールドの初期値は、MQISS_CHECK です。

Reserved (MQCHAR)

これは予約フィールドです。空白でなければならない。

MQIMPO - メッセージ・プロパティ照会オプション

MQIMPO 構造体を使用すると、アプリケーションで、メッセージのプロパティを照会する方法を制御するオプションを指定できます。この構造は、MQINQMP 呼び出しの入力パラメーターです。

可用性

すべての IBM MQ システムおよび IBM MQ クライアント。

文字セットとエンコード

MQIMPO 内のデータは、アプリケーションの文字セットおよびアプリケーションのエンコーディング (MQENC_NATIVE) でなければならない。

フィールド

注: 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 514. MQIPMO のフィールド		
フィールド名と説明	定数の名前	定数の初期値 (存在する場合)
<u>StrucId</u> (構造識別子)	MQIMPO_STRUC_ID	'IMPO'
<u>バージョン</u> (構造バージョン番号)	MQIMPO_VERSION_1	1
<u>オプション</u> (MQINQMP の動作を制御するオプション)	MQIMPO_INQ_FIRST	
<u>RequestedEncoding</u> (問い合わせたプロパティを変換するエンコーディング)	MQENC_NATIVE	
<u>RequestedCCSID</u> (問い合わせ対象プロパティの文字セット)	MQCCSI_APPL	
<u>ReturnedEncoding</u> (返り値のエンコーディング)	MQENC_NATIVE	
<u>ReturnedCCSID</u>	0	
<u>Reserved1</u> (予約フィールド)	空白文字 (4 バイトフィールド)	
<u>ReturnedName</u> (問い合わせ物件名)	MQCHARV_DEFAULT	
<u>TypeString</u> (プロパティのデータ型の文字列表現)	ヌル・ストリングまたはブランク	

表 514. MQIPMO のフィールド (続き)

フィールド名と説明	定数の名前	定数の初期値（存在する場合）
注： 1. ヌル・ストリングまたはブランクの値は、C 言語ではヌル・ストリングを表し、他のプログラミング言語ではブランク文字を表します。 2. C プログラミング言語では、マクロ変数 MQIMPO_DEFAULT には、表にリストされている値が含まれています。このマクロ変数を以下の方法で使用して、構造体のフィールドに初期値を設定します。 <pre>MQIMPO MyIMPO = {MQIMPO_DEFAULT};</pre>		

言語ごとの宣言

MQIMPO の C 宣言

```
typedef struct tagMQIMPO MQIMPO;
struct tagMQIMPO {
    MQCHAR4  StrucId;           /* Structure identifier */
    MQLONG   Version;          /* Structure version number */
    MQLONG   Options;          /* Options that control the action of
                               MQINQMP */
    MQLONG   RequestedEncoding; /* Requested encoding of Value */
    MQLONG   RequestedCCSID;    /* Requested character set identifier
                               of Value */
    MQLONG   ReturnedEncoding;  /* Returned encoding of Value */
    MQLONG   ReturnedCCSID;     /* Returned character set identifier
                               of Value */
    MQCHAR   Reserved1;        /* Reserved field */
    MQCHARV  ReturnedName;     /* Returned property name */
    MQCHAR8  TypeString;       /* Property data type as a string */
};
```

MQIMPO の COBOL 宣言

```
**  MQIMPO structure
10 MQIMPO.
**  Structure identifier
15 MQIMPO-STRUCID          PIC X(4).
**  Structure version number
15 MQIMPO-VERSION         PIC S9(9) BINARY.
**  Options that control the action of MQINQMP
15 MQIMPO-OPTIONS         PIC S9(9) BINARY.
**  Requested encoding of VALUE
15 MQIMPO-REQUESTEDENCODING PIC S9(9) BINARY.
**  Requested character set identifier of VALUE
15 MQIMPO-REQUESTEDCCSID   PIC S9(9) BINARY.
**  Returned encoding of VALUE
15 MQIMPO-RETURNEDENCODING PIC S9(9) BINARY.
**  Returned character set identifier of VALUE
15 MQIMPO-RETURNEDCCSID    PIC S9(9) BINARY.
**  Reserved field
15 MQIMPO-RESERVED1
**  Returned property name
15 MQIMPO-RETURNEDNAME.
**  Address of variable length string
20 MQIMPO-RETURNEDNAME-VSPTR  POINTER.
**  Offset of variable length string
20 MQIMPO-RETURNEDNAME-VSOFFSET PIC S9(9) BINARY.
**  CCSID of variable length string
20 MQIMPO-RETURNEDNAME-VSCCSID PIC S9(9) BINARY.
**  Property data type as string
15 MQIMPO-TYPESTRING        PIC S9(9) BINARY.
```

MQIMPO の PL/I 宣言

```
dcl
  1 MQIMPO based,
  3 StrucId          char(4),          /* Structure identifier */
```

```

3 Version          fixed bin(31), /* Structure version number */
3 Options          fixed bin(31), /* Options that control the
                                action of MQINQMP */
3 RequestedEncoding fixed bin(31), /* Requested encoding of
                                Value */
3 RequestedCCSID   fixed bin(31), /* Requested character set
                                identifier of Value */
3 ReturnedEncoding fixed bin(31), /* Returned encoding of
                                Value */
3 ReturnedCCSID    fixed bin(31), /* Returned character set
                                identifier of Value */
3 Reserved1        fixed bin(31), /* Reserved field */
3 ReturnedName,    /* Returned property name */
5 ReturnedName_VSPtr pointer, /* Address of returned
                                name */
5 5 ReturnedName_VSOffset fixed bin(31), /* Offset of returned
                                name */
5 5 ReturnedName_VSCCSID fixed bin(31), /* CCSID of returned
                                name */
3 TypeString       char(8); /* Property data type as
                                string */

```

MQIMPO の高水準アセンブラ宣言

```

MQIMPO          DSECT
MQIMPO_STRUCID  DS   CL4  Structure identifier
MQIMPO_VERSION  DS   F    Structure version number
MQIMPO_OPTIONS  DS   F    Options that control the
*               action of MQINQMP
MQIMPO_REQUESTEDENCODING DS F Requested encoding of VALUE
MQIMPO_REQUESTEDCCSID   DS F Requested character set
*               identifier of VALUE
MQIMPO_RETURNEDENCODING DS F Returned encoding of VALUE
MQIMPO_RETURNEDCCSID    DS F Returned character set
*               identifier of VALUE
MQIMPO_RESERVED1       DS F Reserved field
MQIMPO_RETURNEDNAME     DS 0F Force fullword alignment
MQIMPO_RETURNEDNAME_VSPTR DS F Address of returned name
MQIMPO_RETURNEDNAME_VSOFFSET DS F Offset of returned name
MQIMPO_RETURNEDNAME_VSLENGTH DS F Length of returned name
MQIMPO_RETURNEDNAME_VSCCSID DS F CCSID of returned name
MQIMPO_RETURNEDNAME_LENGTH EQU *-MQIMPO_RETURNEDNAME
                                ORG MQIMPO_RETURNEDNAME
MQIMPO_RETURNEDNAME_AREA DS   CL(MQIMPO_RETURNEDNAME_LENGTH)
*
MQIMPO_TYPESTRING       DS   CL8 Property data type as string
MQIMPO_LENGTH           EQU   *-MQIMPO
MQIMPO_AREA             DS   CL(MQIMPO_LENGTH)

```

MQIMPO のフィールド詳細

MQIMPO の問い合わせメッセージプロパティオプション構造体により、アプリケーションはメッセージのプロパティが問い合わせられる方法を制御するオプションを指定できます。この構造は、MQINQMP 呼び出しの入力パラメーターです。

フィールド

注: 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 515. MQIPMO のフィールド		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
StrucId (構造識別子)	MQIMPO_STRUC_ID	'IMPO'
バージョン (構造バージョン番号)	MQIMPO_VERSION_1	1
オプション (MQINQMP の動作を制御するオプション)	MQIMPO_INQ_FIRST	
RequestedEncoding (問い合わせたプロパティを変更するエンコーディング)	MQENC_NATIVE	

表 515. MQIPMO のフィールド (続き)

フィールド名と説明	定数の名前	定数の初期値 (存在する場合)
<u>RequestedCCSID</u> (問い合わせ対象プロパティの文字セット)	MQCCSI_APPL	
<u>ReturnedEncoding</u> (戻り値のエンコーディング)	MQENC_NATIVE	
<u>ReturnedCCSID</u>	0	
<u>Reserved1</u> (予約フィールド)	空白文字 (4 バイトフィールド)	
<u>ReturnedName</u> (問い合わせ物件名)	MQCHARV_DEFAULT	
<u>TypeString</u> (プロパティのデータ型の文字列表現)	ヌル・ストリングまたは ブランク	

注:

1. ヌル・ストリングまたはブランクの値は、C 言語ではヌル・ストリングを表し、他のプログラミング言語ではブランク文字を表します。
2. C プログラミング言語では、マクロ変数 MQIMPO_DEFAULT には、表にリストされている値が含まれています。このマクロ変数を以下の方法で使用して、構造体のフィールドに初期値を設定します。

```
MQIMPO MyIMPO = {MQIMPO_DEFAULT};
```

StrucId (MQCHAR4)

これは、問い合わせメッセージのプロパティオプション構造体の構造体識別子です。常に入力フィールドです。その値は MQIMPO_STRUC_ID です。

値は次のものでなければなりません。

MQIMPO_STRUC_ID

問い合わせメッセージのプロパティオプション構造体の識別子。

C プログラミング言語では、定数 MQIMPO_STRUC_ID_ARRAY も定義されている。これは MQIMPO_STRUC_ID と同じ値を持ちますが、文字列ではなく文字の配列です。

Version (MQLONG)

問い合わせメッセージプロパティオプション構造 - バージョンフィールド

これは構造体のバージョン番号です。値は次のものでなければなりません。

MQIMPO_VERSION_1

メッセージ・プロパティ照会オプション構造のバージョン番号。

以下の定数は、現行バージョンのバージョン番号を指定しています。

MQIMPO_CURRENT_VERSION

メッセージ・プロパティ照会オプション構造の現行バージョン。

これは常に入力フィールドです。このフィールドの初期値は、MQIMPO_VERSION_1 です。

Options (MQLONG)

問い合わせメッセージプロパティオプション構造 - オプションフィールド

以下のオプションは、MQINQMP のアクションを制御します。以下に説明する 1 つ以上のオプションを指定できます。複数のオプションを指定するには、値を加算する (同じ定数を複数回加算しない) か、ビット単位 OR 演算を使用して値を結合 (プログラミング言語でビット演算がサポートされる場合) します。

無効なオプションの組み合わせについては注記されています。それ以外の組み合わせは有効です。

値データ・オプション: 以下のオプションは、プロパティがメッセージから取り出される時の値データの処理と関係しています。

MQIMPO_CONVERT_VALUE

このオプションは、プロパティの値を、*RequestedCCSID* および *RequestedEncoding* の指定値に合うように変換してから、MQINQMP 呼び出しがプロパティ値を *Value* 領域に戻すように要求します。

- 変換が正常に実行されると、MQINQMP 呼び出しから戻る際に、*ReturnedCCSID* および *ReturnedEncoding* フィールドは、*RequestedCCSID* および *RequestedEncoding* と同じ値に設定されます。
- 変換は失敗したものの、MQINQMP 呼び出しがそれ以外についてはエラーなしで完了した場合、プロパティ値は変換されないまま返されます。

プロパティがストリングの場合は、*ReturnedCCSID* および *ReturnedEncoding* フィールドは、未変換ストリングの文字セットとエンコードに設定されます。

この場合、完了コードは MQCC_WARNING で、理由コードは MQRC_PROP_VALUE_NOT_CONVERTED です。プロパティ・カーソルは、返されたプロパティに進みます。

プロパティ値が変換中に拡張し、**Value** パラメーターのサイズを超える場合は、値が変換されずに戻され、完了コードは MQCC_FAILED になります。理由コードは MQRC_PROPERTY_VALUE_TOO_BIG に設定されます。

MQINQMP 呼び出しの **DataLength** パラメーターは、変換されたプロパティ値を収容するために必要なバッファのサイズをアプリケーションが判別できるようにするために、プロパティ値が変換された場合の長さを返します。プロパティ・カーソルは変更されません。

このオプションは、以下のことも要求します。

- プロパティ名にワイルドカードが含まれているかどうか。および
- *ReturnedName* フィールドが、戻される名前のアドレスまたはオフセットを使用して初期設定される。

その後、戻された名前は、*RequestedCCSID* および *RequestedEncoding* 値に合うように変換されます。

- 変換が正常に実行されると、*ReturnedName* の *VSCCSID* フィールドおよび返される名前のエンコードは、*RequestedCCSID* および *RequestedEncoding* の入力値に設定されます。
- 変換は失敗したものの、MQINQMP 呼び出しがそれ以外についてはエラーまたは警告なしで完了した場合、返される名前は未変換のままです。この場合、完了コードは MQCC_WARNING で、理由コードは MQRC_PROP_NAME_NOT_CONVERTED です。

プロパティ・カーソルは、返されたプロパティに進みます。値と名前が両方とも変換されない場合は、MQRC_PROP_VALUE_NOT_CONVERTED が戻されます。

戻される名前が変換中に拡張し、*RequestedName* の *VSBuFSIZE* フィールドのサイズを超える場合は、戻されるストリングは変換されないままになり、完了コードは MQCC_FAILED になります。理由コードは MQRC_PROPERTY_NAME_TOO_BIG に設定されます。

変換されたプロパティ値を収容するのに必要なバッファのサイズをアプリケーションが判別できるように、MQCHARV 構造の *VSLength* フィールドは、プロパティ値の変換結果の長さを返します。プロパティ・カーソルは変更されません。

MQIMPO_CONVERT_TYPE

このオプションは、プロパティの値を、現行のデータ・タイプから、MQINQMP 呼び出しの **Type** パラメーターで指定したデータ・タイプに変換するように要求します。

- 変換が正常に実行されると、MQINQMP 呼び出しから戻る際に、**Type** パラメーターは変更されません。

- 変換が失敗したが、それ以外は MQINQMP 呼び出しがエラーなしで完了した場合は、呼び出しは理由 MQRC_PROP_CONV_NOT_SUPPORTED で失敗します。プロパティ・カーソルは変更されません。

データ・タイプの変換中に値が拡張し、変換された値が **Value** パラメーターのサイズを超える場合は、値が変換されずに戻され、完了コードは MQCC_FAILED になります。理由コードは MQRC_PROPERTY_VALUE_TOO_BIG に設定されます。

MQINQMP 呼び出しの **DataLength** パラメーターは、変換されたプロパティ値を収容するために必要なバッファのサイズをアプリケーションが判別できるようにするために、プロパティ値が変換された場合の長さを返します。プロパティ・カーソルは変更されません。

MQINQMP 呼び出しの **Type** パラメーターの値が無効な場合は、呼び出しは理由 MQRC_PROPERTY_TYPE_ERROR で失敗します。

要求されたデータ・タイプ変換がサポートされていない場合は、呼び出しは理由 MQRC_PROP_CONV_NOT_SUPPORTED で失敗します。以下のデータ・タイプ変換がサポートされています。

表 516. サポートされているデータ・タイプ変換	
プロパティのデータ・タイプ	サポートされているターゲット・データ・タイプ
MQTYPE_BOOLEAN	MQTYPE_STRING、MQTYPE_INT8、MQTYPE_INT16、MQTYPE_INT32、MQTYPE_INT64
MQTYPE_BYTE_STRING	MQTYPE_STRING
MQTYPE_INT8	MQTYPE_STRING、MQTYPE_INT16、MQTYPE_INT32、MQTYPE_INT64
MQTYPE_INT16	MQTYPE_STRING、MQTYPE_INT32、MQTYPE_INT64
MQTYPE_INT32	MQTYPE_STRING、MQTYPE_INT64
MQTYPE_INT64	MQTYPE_STRING
MQTYPE_FLOAT32	MQTYPE_STRING、MQTYPE_FLOAT64
MQTYPE_FLOAT64	MQTYPE_STRING
MQTYPE_STRING	MQTYPE_BOOLEAN、MQTYPE_INT8、MQTYPE_INT16、MQTYPE_INT32、MQTYPE_INT64、MQTYPE_FLOAT32、MQTYPE_FLOAT64
MQTYPE_NULL	なし

サポートされる変換を制御する一般規則は、以下のとおりです。

- 数値のプロパティ値は、変換中にデータが失われなければ、データ・タイプ間で変換できる。
例えば、データ・タイプ MQTYPE_INT32 のプロパティの値をデータ・タイプ MQTYPE_INT64 の値に変換できますが、データ・タイプ MQTYPE_INT16 の値には変換できません。
- どのデータ・タイプのプロパティ値でも、ストリングに変換できる。
- ストリング・プロパティ値は、ストリングが変換用に正しくフォーマットされていれば、任意の他のデータ・タイプに変換できます。アプリケーションが正しくフォーマットされていないストリング・プロパティ値に変換しようとする、IBM MQ は理由コード MQRC_PROP_NUMBER_FORMAT_ERROR を返します。
- サポートされていない変換をアプリケーションが試行すると、IBM MQ は理由コード MQRC_PROP_CONV_NOT_SUPPORTED を返します。

プロパティ値のデータ・タイプを変換する場合の具体的な規則は、以下のとおりです。

- MQTYPE_BOOLEAN プロパティ値をストリングに変換する際には、値 TRUE がストリング「TRUE」に変換され、値 false はストリング「FALSE」に変換される。

- MQTYPE_BOOLEAN プロパティ値を数値データ・タイプに変換する際には、値 TRUE が 1 に変換され、値 FALSE はゼロに変換される。
- スtring・プロパティ値を MQTYPE_BOOLEAN 値に変換する際には、String「TRUE」または「1」が TRUE に変換され、String「FALSE」または「0」が FALSE に変換される。

用語「TRUE」および「FALSE」には大/小文字の区別がないことに注意してください。

その他のStringは変換できません。IBM MQ は理由コード MQRC_PROP_NUMBER_FORMAT_ERROR を戻します。

- String・プロパティ値をデータ・タイプ MQTYPE_INT8、MQTYPE_INT16、MQTYPE_INT32、または MQTYPE_INT64 の値に変換する際には、Stringは以下の形式でなければなりません。

```
[blanks][sign]digits
```

Stringの構成要素の意味は以下のとおりです。

blanks

オプションの先行ブランク文字

sign

オプションの正符号 (+) または負符号 (-) 文字。

digits

数字 (0 から 9 まで) の連続シーケンス。少なくとも 1 つの数字が存在する必要があります。

数字のシーケンスの後のStringには数字以外の文字を含めることができますが、それらの文字の最初のものに達するとすぐに変換は停止します。Stringは 10 進整数を表すと想定されます。

Stringが正しく形式設定されていない場合は、IBM MQ は理由コード MQRC_PROP_NUMBER_FORMAT_ERROR を戻します。

- String・プロパティ値をデータ・タイプ MQTYPE_FLOAT32 または MQTYPE_FLOAT64 の値に変換する際には、Stringは以下の形式でなければなりません。

```
[blanks][sign]digits[.digits][e_char[e_sign]e_digits]
```

Stringの構成要素の意味は以下のとおりです。

blanks

オプションの先行ブランク文字

sign

オプションの正符号 (+) または負符号 (-) 文字。

digits

数字 (0 から 9 まで) の連続シーケンス。少なくとも 1 つの数字が存在する必要があります。

e_char

指数文字。「E」か「e」のどちらかです。

e_sign

指数用の、オプションの正符号 (+) または負符号 (-) 文字。

e_digits

指数用の、数字 (0-9) の連続シーケンス。Stringに指数文字がある場合、1 つ以上の数字がなければなりません。

数字のシーケンスの後、または指数を表すオプションの文字の後のStringには数字以外の文字を含めることができますが、それらの文字の最初のものに達するとすぐに変換は停止します。Stringは、10 の累乗の指数を持つ 10 進浮動小数点数を表すと想定されます。

Stringが正しく形式設定されていない場合は、IBM MQ は理由コード MQRC_PROP_NUMBER_FORMAT_ERROR を戻します。

- 数値のプロパティ値をStringに変換する際には、値は、この値に関する ASCII 文字を含むStringではなく、この値の 10 進数のString表現に変換されます。例えば、整数 65 はString「A」ではなくString「65」に変換されます。

- ・バイト・ストリングのプロパティ値をストリングに変換する際には、各バイトは、そのバイトを表す2つの16進文字に変換されます。例えば、バイト配列 {0xF1, 0x12, 0x00, 0xFF} はストリング「F11200FF」に変換されます。

MQIMPO_QUERY_LENGTH

プロパティ値のタイプと長さを照会します。長さは、MQINQMP 呼び出しの **DataLength** パラメータで返されます。プロパティ値は戻されません。

ReturnedName バッファを指定すると、MQCHARV 構造の **VSLength** フィールドは、プロパティ名の長さで埋められます。プロパティ名は戻されません。

反復オプション: 以下は、ワイルドカード文字がある名前を使用した、プロパティの反復に関するオプションです。

MQIMPO_INQ_FIRST

指定された名前に一致する最初のプロパティを照会します。この呼び出しの後に、カーソルは返されるプロパティに設定されます。

これがデフォルト値です。

そのあとで、必要なら、この **MQIMPO_INQ_PROP_UNDER_CURSOR** オプションを **MQINQMP** 呼び出しで使用して、同じプロパティに対して再び照会できます。

プロパティ・カーソルは1つしかないことに注意してください。したがって、**MQINQMP** 呼び出しに指定したプロパティ名を変更すると、カーソルはリセットされます。

このオプションは次のいずれかのオプションが指定されている場合には、無効です。

MQIMPO_INQ_NEXT

MQIMPO_INQ_PROP_UNDER_CURSOR

MQIMPO_INQ_NEXT

プロパティ・カーソルからの検索を続行しながら、指定された名前に一致する次のプロパティを照会します。カーソルは、返されたプロパティに進みます。

指定された名前に関する最初の **MQINQMP** 呼び出しの場合は、指定された名前と一致する最初のプロパティが戻されます。

そのあとで、必要なら、この **MQIMPO_INQ_PROP_UNDER_CURSOR** オプションを **MQINQMP** 呼び出しで使用して、同じプロパティに対して再び照会できます。

カーソルの下のプロパティが削除されている場合は、**MQINQMP** は削除されたプロパティより後で次に一致するプロパティを戻します。

反復の進行中に、ワイルドカードと一致するプロパティが追加された場合、そのプロパティは反復の完了までに返される場合もあれば、返されない場合もあります。**MQIMPO_INQ_FIRST** を使用して反復を再始動すると、プロパティが戻されます。

反復の進行中に、ワイルドカードと一致するプロパティで削除されたものは、削除後には返されません。

このオプションは次のいずれかのオプションが指定されている場合には、無効です。

MQIMPO_INQ_FIRST

MQIMPO_INQ_PROP_UNDER_CURSOR

MQIMPO_INQ_PROP_UNDER_CURSOR

プロパティ・カーソルによって指し示されるプロパティの値を取り出します。プロパティ・カーソルによって指し示されるプロパティは、**MQIMPO_INQ_FIRST** または **MQIMPO_INQ_NEXT** オプションのいずれかによって最後に照会されたものです。

メッセージ・ハンドルを再利用する際、**MQGET** 呼び出しの **MQGMO** の **MsgHandle** フィールド中でメッセージ・ハンドルを指定する際、または **MQPUT** 呼び出しの **MQPMO** 構造体の **OriginalMsgHandle** または **NewMsgHandle** フィールド中でメッセージ・ハンドルを指定する際には、プロパティ・カーソルはリセットされます。

プロパティ・カーソルがまだ確立されていない時点でこのオプションを使用する場合や、プロパティ・カーソルによって指し示されるプロパティが削除されている場合は、呼び出しは完了コード MQCC_FAILED および理由 MQRC_PROPERTY_NOT_AVAILABLE で失敗します。

このオプションは次のいずれかのオプションが指定されている場合には、無効です。

MQIMPO_INQ_FIRST
MQIMPO_INQ_NEXT

上記のオプションがどれも必要でない場合には、以下のオプションを使用できます。

MQIMPO_NONE

この値は、他のオプションが指定されなかったことを示すために使用します。すべてのオプションはデフォルト値であるとみなされます。

MQIMPO_NONE は、プログラムの文書化を支援します。このオプションは、他のオプションと組み合わせて使用するオプションではありません。ただし、このオプションの値はゼロと等価なので、他のオプションと組み合わせて使用しても、エラーとして検出されることはありません。

これは常に入力フィールドです。このフィールドの初期値は、MQIMPO_INQ_FIRST です。

RequestedEncoding (MQLONG)

メッセージプロパティオプション構造体の照会 - RequestedEncoding フィールド

これは、MQIMPO_CONVERT_VALUE または MQIMPO_CONVERT_TYPE の指定時に、照会されるプロパティ値の変換結果のエンコードです。

このフィールドの初期値は MQENC_NATIVE です。

RequestedCCSID (MQLONG)

メッセージプロパティオプション構造体の照会 - RequestedCCSID フィールド

値が文字ストリングの場合に、照会されるプロパティ値の変換結果の文字セット。

MQIMPO_CONVERT_VALUE または MQIMPO_CONVERT_TYPE の指定時には、*ReturnedName* の変換結果のエンコードでもあります。

このフィールドの初期値は MQCCSI_APPL です。

ReturnedEncoding (MQLONG)

メッセージプロパティオプション構造体の照会 - ReturnedEncoding フィールド

出力上に戻される値のエンコードです。

MQIMPO_CONVERT_VALUE オプションを指定する場合に、変換が正常に実行されると、戻される際に、*ReturnedEncoding* フィールドの値は渡される値と同じになります。

このフィールドの初期値は MQENC_NATIVE です。

ReturnedCCSID (MQLONG)

メッセージプロパティオプション構造体の照会 - ReturnedCCSID フィールド

MQINQMP 呼び出しの **Type** パラメーターが MQTYPE_STRING の場合に、出力上に戻される値の文字セットです。

MQIMPO_CONVERT_VALUE オプションを指定する場合に、変換が正常に実行されると、戻される際に、*ReturnedCCSID* フィールドの値は渡される値と同じになります。

フィールドの初期値は、0 です。

Reserved1 (MQCHAR)

これは予約フィールドです。このフィールドの初期値は、ブランク文字 (4 バイト・フィールド) です。

ReturnedName (MQCHARV)

メッセージプロパティオプション構造体の照会 - ReturnedName フィールド

照会されるプロパティの実際の名前。

MQCHARV 構造の *VSPtr* または *VSoffset* フィールドを使用して、入力上にストリング・バッファを渡すことができます。ストリング・バッファの長さは、MQCHARV 構造の *VSBufsize* フィールドを使用して指定されます。

MQINQMP 呼び出しから戻る際に、ストリング・バッファが、照会されたプロパティの名前を完全に入れられる長さである場合は、ストリング・バッファはこの名前で完了します。MQCHARV 構造の *VSLength* フィールドは、プロパティ名の長さで埋められます。名前の変換が失敗したかどうかにかかわらず、MQCHARV 構造の *VSCCSID* フィールドは埋められ、返される名前の文字セットが示されます。

これは入出力フィールドです。このフィールドの初期値は MQCHARV_DEFAULT です。

TypeString (MQCHAR8)

メッセージプロパティオプション構造体の照会 - TypeString フィールド

プロパティのデータ・タイプのストリング表現。

MQRFH2 ヘッダー中にプロパティが指定されていて、MQRFH2 dt 属性が認識されない場合は、このフィールドを使用してプロパティのデータ・タイプを判別できます。*TypeString* はコード化文字セット 1208 (UTF-8) で戻され、認識に失敗したプロパティの dt 属性の値の先頭 8 バイトになります。

これは、常に出力フィールドです。このフィールドの初期値は、C プログラミング言語ではヌル・ストリングですが、その他のプログラミング言語では 8 桁のブランク文字です。

MQMD - メッセージ記述子

MQMD 構造体は、メッセージが送信アプリケーションと受信アプリケーションの間を移動する際に、アプリケーションデータに付随する制御情報を含みます。この構造体は、MQGET、MQPUT、および MQPUT1 呼び出しに指定する入出力パラメーターです。

可用性

すべての IBM MQ システム、およびこれらのシステムに接続された IBM MQ MQI clients。

バージョン

現在の MQMD のバージョンは MQMD_VERSION_2 です。複数の環境間で移植可能にすることを意図したアプリケーションでは、必ず対象のすべての環境で、必要なバージョンの MQMD がサポートされているようにしてください。これより新しいバージョンの構造体にはのみ存在するフィールドについては、そのフィールドの説明にその旨を記載しています。

サポートされるプログラム言語用に提供されているヘッダー・ファイル、コピー・ファイル、およびインクルード・ファイルには、環境でサポートされる最新バージョンの MQMD が含まれていますが、*Version* フィールドの初期値は MQMD_VERSION_1 に設定されています。version-1 構造体に存在しないフィールドを使用するには、アプリケーションで、*Version* フィールドを必要なバージョンのバージョン番号に設定する必要があります。

バージョン 1 の構造体の宣言は、MQMD1 という名前で使用できます。

文字セットとエンコード

MQMD 内のデータは、ローカルキューマネージャの文字セットおよびエンコーディングでなければならない。これらはキューマネージャ **CodedCharSetId** 属性および **MQENC_NATIVE** によって指定される。ただし、アプリケーションを IBM MQ MQI client として実行する場合は、構造体はクライアントの文字セットとエンコードに従っている必要があります。

送信側と受信側のキュー・マネージャで使用する文字セットまたはエンコードが違う場合、MQMD のデータは自動的に変換されます。アプリケーションで MQMD を変換する必要はありません。

MQMD の異なるバージョンの使用

version-2 の MQMD は、version-1 の MQMD を使用し、メッセージデータに MQMDE 構造体をプレフィックスとして付加することに相当する。ただし、MQMDE 構造体のすべてのフィールドにデフォルト値を設定する場合には、MQMDE を省略することができます。バージョン 1 の MQMD、および MQMDE は、次のようにして使用します。

- MQPUT 呼び出しおよび MQPUT1 呼び出しでアプリケーションからバージョン 1 の MQMD を提供するには、オプションとして、メッセージ・データの先頭に MQMDE を付けることができます。その場合は、MQMD の *Format* フィールドに MQFMT_MD_EXTENSION を設定して、MQMDE が存在することを指定します。アプリケーションが MQMDE を提供しない場合、キュー・マネージャは MQMDE の各フィールドにデフォルト値が設定されたものと見なします。

注：バージョン 2 の MQMD に存在して、バージョン 1 の MQMD に存在しないフィールドのいくつかは、MQPUT 呼び出しおよび MQPUT1 呼び出しに対する入出力フィールドです。ただし、キュー・マネージャは、MQPUT 呼び出しおよび MQPUT1 呼び出しで出力が生成されても MQMDE の該当するフィールドに値を戻しません。出力値が必要な場合には、そのアプリケーションでバージョン 2 の MQMD を使用する必要があります。

- MQGET 呼び出しでアプリケーションからバージョン 1 の MQMD を提供した場合は、MQMDE の 1 つ以上のフィールドがデフォルト以外の値の場合に限り、キュー・マネージャから返されるメッセージの先頭に MQMDE が付けられます。その場合、MQMD の *Format* フィールドに MQFMT_MD_EXTENSION という値が設定されて、MQMDE の存在が示されます。

キュー・マネージャが MQMDE の各フィールドに使用するデフォルト値は、[444 ページの表 522](#) に示す各フィールドの初期値と同じです。

メッセージが伝送キュー上にある場合、MQMD 内のフィールドの一部が特定の値に設定されます。詳細については、[609 ページの『MQXQH - 伝送キュー・ヘッダー』](#)を参照してください。

メッセージ・コンテキスト

MQMD の特定のフィールドにはメッセージ・コンテキストが含まれます。メッセージ・コンテキストには、ID コンテキストと発信元コンテキストという 2 つのタイプがあります。一般に、

- ID コンテキストは、最初にメッセージを書き込んだアプリケーションに関連したものです。
- 発信元コンテキストは、一番最近にメッセージを書き込んだアプリケーションに関連したものです。

これら 2 つのアプリケーションは同じアプリケーションのこともあります、異なるアプリケーションであるというケースもあります (例えば、メッセージが 1 つのアプリケーションから別のアプリケーションに転送された場合)。

ID コンテキストと発信元コンテキストには一般に前述のような意味合いがありますが、MQMD 内のこの 2 つのタイプのコンテキスト・フィールドの内容は、メッセージが書き込まれた時点での

MQPMO_*_CONTEXT オプションの指定によって異なります。その結果、ID コンテキストは必ずしも最初にメッセージを書き込んだアプリケーションに関連したものではなく、発信元コンテキストは必ずしも一番最近にメッセージを書き込んだアプリケーションに関連したものではありません。その内容は、アプリケーション・スイートの設計に応じて決まります。

メッセージ・チャネル・エージェント (MCA) がメッセージ・コンテキストを変更することは一切ありません。リモート・キュー・マネージャからメッセージを受け取る MCA は、MQPUT または MQPUT1 呼び出しでコンテキスト・オプション MQPMO_SET_ALL_CONTEXT を使用します。これにより、受信側 MCA で、

送信側 MCA からメッセージと一緒に伝達されてきたメッセージ・コンテキストを正確に保持することが可能になります。ただし、その結果として、起点コンテキストはメッセージの送信側の MCA および受信側の MCA のいずれにも関連付けられません。起点コンテキストは、メッセージを書き込んだ以前のアプリケーションを指します。すべての中間アプリケーションがメッセージ・コンテキストを渡している場合、起点コンテキストは発信元のアプリケーション自身を指します。

この説明の中で、コンテキスト・フィールドは、最初の説明のとおり各フィールドが使用された場合として記述されています。メッセージのコンテキストの詳細については、[メッセージのコンテキスト](#)を参照してください。

フィールド

注：次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 517. MQMD における MQMD のフィールド		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
StrucId (構造識別子)	MQMD_STRUC_ID	'MD'
バージョン (構造バージョン番号)	MQMD_VERSION_1	1
レポート (レポートメッセージのオプション)	MQRO_NONE	0
MsgType (メッセージタイプ)	MQMT_DATAGRAM	8
MQMD - 期限フィールド (メッセージの有効期間)	MQEI_UNLIMITED	-1
MQMD - フィードバックフィールド (フィードバックまたは理由コード)	MQFB_NONE	0
符号化 (メッセージデータの数値符号化)	MQENC_NATIVE	環境に依存
CodedCharSetId (メッセージデータの文字セット識別子)	MQCCSI_Q_MGR	0
フォーマット (メッセージデータのフォーマット名)	MQFMT_NONE	ブランク
優先度 (メッセージ優先度)	MQPRI_PRIORITY_AS_Q_DEF	-1
永続性 (メッセージ永続性)	MQPER_PERSISTENCE_AS_Q_DEF	2
MQMD - メッセージ識別子フィールド (MsgId)	MQMI_NONE	Null
CorrelId (相関識別子)	MQCI_NONE	Null
BackoutCount (バックアウトカウンター)	なし	0
ReplyToQ (返信キューの名前)	なし	ヌル・ストリングまたはブランク
ReplyToQMGr (応答キュー管理者の名前)	なし	ヌル・ストリングまたはブランク
UserIdentifier (ユーザー識別子)	なし	ヌル・ストリングまたはブランク
AccountingToken (会計トークン)	MQACT_NONE	Null
ApplIdentityData (身元に関する申請データ)	なし	ヌル・ストリングまたはブランク

表 517. MQMD における MQMD のフィールド (続き)

フィールド名と説明	定数の名前	定数の初期値 (存在する場合)
<u>PutApplType</u> (メッセージを表示したアプリケーションの種類)	MQAT_NO_CONTEXT	0
<u>PutApplName</u> (メッセージを表示したアプリケーション名)	なし	ヌル・ストリングまたはブランク
<u>PutDate</u> (メッセージが投稿された日付)	なし	ヌル・ストリングまたはブランク
<u>PutTime</u> (メッセージが投稿された時刻)	なし	ヌル・ストリングまたはブランク
<u>ApplOriginData</u> (原産地に関する申請データ)	なし	ヌル・ストリングまたはブランク
注: <i>Version</i> が MQMD_VERSION_2 より小さい場合は、残りのフィールドは無視されます。		
<u>GroupId</u> (グループ識別子)	MQGI_NONE	Null
<u>MsgSeqNumber</u> (グループ内の論理メッセージのシーケンス番号)	なし	1
オフセット (物理メッセージ内のデータの論理メッセージの先頭からのオフセット)	なし	0
MQMD - メッセージフラグ (MsgFlags フィールド)	MQMF_NONE	0
<u>OriginalLength</u> (原文の長さ)	MQOL_UNDEFINED	-1
注: 1. ヌル・ストリングまたはブランクの値は、C 言語ではヌル・ストリングを表し、他のプログラミング言語ではブランク文字を表します。 2. C プログラミング言語では、マクロ変数 MQMD_DEFAULT には、表にリストされている値が含まれています。この変数を以下の方法で使用すると、構造体のフィールドに初期値を設定できます。		
<pre>MQMD MyMD = {MQMD_DEFAULT};</pre>		

言語ごとの宣言

MQMD の C 宣言

```
typedef struct tagMQMD MQMD;
struct tagMQMD {
    MQCHAR4    StructId;        /* Structure identifier */
    MQLONG     Version;         /* Structure version number */
    MQLONG     Report;          /* Options for report messages */
    MQLONG     MsgType;         /* Message type */
    MQLONG     Expiry;          /* Message lifetime */
    MQLONG     Feedback;        /* Feedback or reason code */
    MQLONG     Encoding;        /* Numeric encoding of message data */
    MQLONG     CodedCharSetId;  /* Character set identifier of message
                                data */
    MQCHAR8    Format;          /* Format name of message data */
    MQLONG     Priority;         /* Message priority */
    MQLONG     Persistence;     /* Message persistence */
    MQBYTE24   MsgId;           /* Message identifier */
    MQBYTE24   CorrelId;        /* Correlation identifier */
    MQLONG     BackoutCount;     /* Backout counter */
    MQCHAR48   ReplyToQ;        /* Name of reply queue */
    MQCHAR48   ReplyToQMgr;     /* Name of reply queue manager */
    MQCHAR12   UserIdentifier;  /* User identifier */
    MQBYTE32   AccountingToken; /* Accounting token */
}
```



```

MQCHAR32  ApplIdentityData; /* Application data relating to
                               identity */
MQLONG    PutApplType;      /* Type of application that put the
                               message */
MQCHAR28  PutApplName;      /* Name of application that put the
                               message */
MQCHAR8    PutDate;         /* Date when message was put */
MQCHAR8    PutTime;         /* Time when message was put */
MQCHAR4    ApplOriginData;  /* Application data relating to origin */
MQBYTE24   GroupId;         /* Group identifier */
MQLONG     MsgSeqNumber;    /* Sequence number of logical message
                               within group */
MQLONG     Offset;          /* Offset of data in physical message
                               from start of logical message */
MQLONG     MsgFlags;        /* Message flags */
MQLONG     OriginalLength;  /* Length of original message */
};

```

MQMD の COBOL 宣言

```

**      MQMD structure
10 MQMD.
**      Structure identifier
15 MQMD-STRUCID          PIC X(4).
**      Structure version number
15 MQMD-VERSION          PIC S9(9) BINARY.
**      Options for report messages
15 MQMD-REPORT           PIC S9(9) BINARY.
**      Message type
15 MQMD-MSGTYPE          PIC S9(9) BINARY.
**      Message lifetime
15 MQMD-EXPIRY           PIC S9(9) BINARY.
**      Feedback or reason code
15 MQMD-FEEDBACK         PIC S9(9) BINARY.
**      Numeric encoding of message data
15 MQMD-ENCODING         PIC S9(9) BINARY.
**      Character set identifier of message data
15 MQMD-CODEDCHARSETID  PIC S9(9) BINARY.
**      Format name of message data
15 MQMD-FORMAT           PIC X(8).
**      Message priority
15 MQMD-PRIORITY         PIC S9(9) BINARY.
**      Message persistence
15 MQMD-PERSISTENCE      PIC S9(9) BINARY.
**      Message identifier
15 MQMD-MSGID            PIC X(24).
**      Correlation identifier
15 MQMD-CORRELID         PIC X(24).
**      Backout counter
15 MQMD-BACKOUTCOUNT    PIC S9(9) BINARY.
**      Name of reply queue
15 MQMD-REPLYTOQ         PIC X(48).
**      Name of reply queue manager
15 MQMD-REPLYTOQMGR      PIC X(48).
**      User identifier
15 MQMD-USERIDENTIFIER   PIC X(12).
**      Accounting token
15 MQMD-ACCOUNTINGTOKEN  PIC X(32).
**      Application data relating to identity
15 MQMD-APPLIDENTITYDATA PIC X(32).
**      Type of application that put the message
15 MQMD-PUTAPPLTYPE      PIC S9(9) BINARY.
**      Name of application that put the message
15 MQMD-PUTAPPLNAME      PIC X(28).
**      Date when message was put
15 MQMD-PUTDATE          PIC X(8).
**      Time when message was put
15 MQMD-PUTTIME          PIC X(8).
**      Application data relating to origin
15 MQMD-APPLORIGINDATA  PIC X(4).
**      Group identifier
15 MQMD-GROUPID          PIC X(24).
**      Sequence number of logical message within group
15 MQMD-MSGSEQNUMBER     PIC S9(9) BINARY.
**      Offset of data in physical message from start of logical message
15 MQMD-OFFSET           PIC S9(9) BINARY.
**      Message flags
15 MQMD-MSGFLAGS         PIC S9(9) BINARY.
**      Length of original message
15 MQMD-ORIGINALLENGTH   PIC S9(9) BINARY.

```

MQMD の PL/I 宣言

```

dcl
  1 MQMD based,
    3 StrucId          char(4),          /* Structure identifier */
    3 Version          fixed bin(31),    /* Structure version number */
    3 Report           fixed bin(31),    /* Options for report messages */
    3 MsgType          fixed bin(31),    /* Message type */
    3 Expiry           fixed bin(31),    /* Message lifetime */
    3 Feedback         fixed bin(31),    /* Feedback or reason code */
    3 Encoding         fixed bin(31),    /* Numeric encoding of message
                                         data */
    3 CodedCharSetId   fixed bin(31),    /* Character set identifier of
                                         message data */
    3 Format            char(8),          /* Format name of message data */
    3 Priority          fixed bin(31),    /* Message priority */
    3 Persistence      fixed bin(31),    /* Message persistence */
    3 MsgId            char(24),         /* Message identifier */
    3 CorrelId         char(24),         /* Correlation identifier */
    3 BackoutCount     fixed bin(31),    /* Backout counter */
    3 ReplyToQ         char(48),         /* Name of reply queue */
    3 ReplyToQMgr      char(48),         /* Name of reply queue manager */
    3 UserIdentifier   char(12),         /* User identifier */
    3 AccountingToken  char(32),         /* Accounting token */
    3 ApplIdentityData char(32),         /* Application data relating to
                                         identity */
    3 PutApplType      fixed bin(31),    /* Type of application that put the
                                         message */
    3 PutApplName      char(28),         /* Name of application that put the
                                         message */
    3 PutDate          char(8),          /* Date when message was put */
    3 PutTime          char(8),          /* Time when message was put */
    3 ApplOriginData   char(4),          /* Application data relating to
                                         origin */
    3 GroupId          char(24),         /* Group identifier */
    3 MsgSeqNumber     fixed bin(31),    /* Sequence number of logical
                                         message within group */
    3 Offset           fixed bin(31),    /* Offset of data in physical
                                         message from start of logical
                                         message */
    3 MsgFlags         fixed bin(31),    /* Message flags */
    3 OriginalLength   fixed bin(31);   /* Length of original message */

```

MQMD の高水準アセンブラ宣言

MQMD	DSECT	
MQMD_STRUCID	DS	CL4 Structure identifier
MQMD_VERSION	DS	F Structure version number
MQMD_REPORT	DS	F Options for report messages
MQMD_MSGTYPE	DS	F Message type
MQMD_EXPIRY	DS	F Message lifetime
MQMD_FEEDBACK	DS	F Feedback or reason code
MQMD_ENCODING	DS	F Numeric encoding of message data
MQMD_CODEDCHARSETID	DS	F Character set identifier of message data
*		
MQMD_FORMAT	DS	CL8 Format name of message data
MQMD_PRIORITY	DS	F Message priority
MQMD_PERSISTENCE	DS	F Message persistence
MQMD_MSGID	DS	XL24 Message identifier
MQMD_CORRELID	DS	XL24 Correlation identifier
MQMD_BACKOUTCOUNT	DS	F Backout counter
MQMD_REPLYTOQ	DS	CL48 Name of reply queue
MQMD_REPLYTOQMGR	DS	CL48 Name of reply queue manager
MQMD_USERIDENTIFIER	DS	CL12 User identifier
MQMD_ACCOUNTINGTOKEN	DS	XL32 Accounting token
MQMD_APPLIDENTITYDATA	DS	CL32 Application data relating to identity
MQMD_PUTAPPLTYPE	DS	F Type of application that put the message
*		
MQMD_PUTAPPLNAME	DS	CL28 Name of application that put the message
*		
MQMD_PUTDATE	DS	CL8 Date when message was put
MQMD_PUTTIME	DS	CL8 Time when message was put
MQMD_APPLORIGINDATA	DS	CL4 Application data relating to origin
MQMD_GROUPID	DS	XL24 Group identifier
MQMD_MSGSEQNUMBER	DS	F Sequence number of logical message within group
*		
MQMD_OFFSET	DS	F Offset of data in physical message from start of logical message
*		
MQMD_MSGFLAGS	DS	F Message flags
MQMD_ORIGINALLENGTH	DS	F Length of original message

```

*
MQMD_LENGTH      EQU    *-MQMD
                  ORG    MQMD
MQMD_AREA         DS     CL(MQMD_LENGTH)

```

MQMD の Visual Basic 宣言

```

Type MQMD
    StructId      As String*4 'Structure identifier'
    Version       As Long     'Structure version number'
    Report        As Long     'Options for report messages'
    MsgType       As Long     'Message type'
    Expiry        As Long     'Message lifetime'
    Feedback      As Long     'Feedback or reason code'
    Encoding      As Long     'Numeric encoding of message data'
    CodedCharSetId As Long     'Character set identifier of message'
    'data'
    Format        As String*8 'Format name of message data'
    Priority      As Long     'Message priority'
    Persistence   As Long     'Message persistence'
    MsgId        As MQBYTE24 'Message identifier'
    CorrelId     As MQBYTE24 'Correlation identifier'
    BackoutCount As Long     'Backout counter'
    ReplyToQ     As String*48 'Name of reply queue'
    ReplyToQMgr  As String*48 'Name of reply queue manager'
    UserId       As String*12 'User identifier'
    AccountingToken As MQBYTE32 'Accounting token'
    ApplIdentityData As String*32 'Application data relating to identity'
    PutApplType  As Long     'Type of application that put the'
    'message'
    PutApplName  As String*28 'Name of application that put the'
    'message'
    PutDate      As String*8 'Date when message was put'
    PutTime      As String*8 'Time when message was put'
    ApplOriginData As String*4 'Application data relating to origin'
    GroupId      As MQBYTE24 'Group identifier'
    MsgSeqNumber As Long     'Sequence number of logical message'
    'within group'
    Offset       As Long     'Offset of data in physical message'
    'from start of logical message'
    MsgFlags     As Long     'Message flags'
    OriginalLength As Long     'Length of original message'
End Type

```

MQMD のフィールド詳細

MQMD メッセージ記述子構造体は、メッセージが送信アプリケーションと受信アプリケーションの間を移動する際に、アプリケーションデータに付随する制御情報を含みます。この構造体は、MQGET、MQPUT、および MQPUT1 呼び出しに指定する入出力パラメーターです。

フィールド

注: 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 518. MQMD における MQMD のフィールド		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
StructId (構造識別子)	MQMD_STRUC_ID	'MD'
バージョン (構造バージョン番号)	MQMD_VERSION_1	1
レポート (レポートメッセージのオプション)	MQRO_NONE	0
MsgType (メッセージタイプ)	MQMT_DATAGRAM	8
有効期限 (メッセージの存続期間)	MQEI_UNLIMITED	-1
フィードバック (フィードバックまたは理由コード)	MQFB_NONE	0
符号化 (メッセージデータの数値符号化)	MQENC_NATIVE	環境に依存

表 518. MQMD における MQMD のフィールド (続き)

フィールド名と説明	定数の名前	定数の初期値 (存在する場合)
<u>CodedCharSetId</u> (メッセージデータの文字セット識別子)	MQCCSI_Q_MGR	0
<u>フォーマット</u> (メッセージデータのフォーマット名)	MQFMT_NONE	ブランク
<u>優先度</u> (メッセージ優先度)	MQPRI_PRIORITY_AS_Q_DEF	-1
<u>永続性</u> (メッセージ永続性)	MQPER_PERSISTENCE_AS_Q_DEF	2
<u>MsgId</u> (メッセージ識別子)	MQMI_NONE	Null
<u>CorrelId</u> (相関識別子)	MQCI_NONE	Null
<u>BackoutCount</u> (バックアウトカウンター)	なし	0
<u>ReplyToQ</u> (返信キューの名前)	なし	ヌル・ストリングまたはブランク
<u>ReplyToQMGr</u> (応答キュー管理者の名前)	なし	ヌル・ストリングまたはブランク
<u>UserIdentifier</u> (ユーザー識別子)	なし	ヌル・ストリングまたはブランク
<u>AccountingToken</u> (会計トークン)	MQACT_NONE	Null
<u>ApplIdentityData</u> (身元に関する申請データ)	なし	ヌル・ストリングまたはブランク
<u>PutApplType</u> (メッセージを表示したアプリケーションの種類)	MQAT_NO_CONTEXT	0
<u>PutApplName</u> (メッセージを表示したアプリケーション名)	なし	ヌル・ストリングまたはブランク
<u>PutDate</u> (メッセージが投稿された日付)	なし	ヌル・ストリングまたはブランク
<u>PutTime</u> (メッセージが投稿された時刻)	なし	ヌル・ストリングまたはブランク
<u>ApplOriginData</u> (原産地に関する申請データ)	なし	ヌル・ストリングまたはブランク
注: <i>Version</i> が MQMD_VERSION_2 より小さい場合は、残りのフィールドは無視されます。		
<u>GroupId</u> (グループ識別子)	MQGI_NONE	Null
<u>MsgSeqNumber</u> (グループ内の論理メッセージのシーケンス番号)	なし	1
<u>オフセット</u> (物理メッセージ内のデータの論理メッセージの先頭からのオフセット)	なし	0
<u>MsgFlags</u> (メッセージフラグ)	MQMF_NONE	0
<u>OriginalLength</u> (原文の長さ)	MQOL_UNDEFINED	-1

表 518. MQMD における MQMD のフィールド (続き)

フィールド名と説明	定数の名前	定数の初期値 (存在する場合)
注: 1. ヌル・ストリングまたはブランクの値は、C 言語ではヌル・ストリングを表し、他のプログラミング言語ではブランク文字を表します。 2. C プログラミング言語では、マクロ変数 MQMD_DEFAULT には、表にリストされている値が含まれています。この変数を以下の方法で使用すると、構造体のフィールドに初期値を設定できます。 <pre>MQMD MyMD = {MQMD_DEFAULT};</pre>		

StrucId (MQCHAR4)

これはメッセージ記述子構造体の構造体識別子です。常に入力フィールドです。その値は MQMD_STRUC_ID です。

値は次のものでなければなりません。

MQMD_STRUC_ID

メッセージ記述子構造体の識別子。

C プログラミング言語では、定数 MQMD_STRUC_ID_ARRAY も定義されている。これは MQMD_STRUC_ID と同じ値を持ちますが、文字列ではなく文字の配列です。

Version (MQLONG)

これは構造体のバージョン番号です。以下のいずれかでなければなりません。

MQMD_VERSION_1

バージョン 1 のメッセージ記述子の構造体。

このバージョンはすべての環境でサポートされます。

MQMD_VERSION_2

バージョン 2 のメッセージ記述子の構造体。

このバージョンは、IBM MQ V6.0 以降のすべての環境と、それらのシステムに接続された IBM MQ MQI clients でサポートされます。

注: バージョン 2 の MQMD が使用されているときには、アプリケーション・メッセージ・データの先頭に存在する可能性のある MQ ヘッダー構造体に対してキュー・マネージャーが追加の検査を実行します。詳細については、MQPUT 呼び出しの使用上の注意を参照してください。

これより新しいバージョンの構造体にのみ存在するフィールドは、そのフィールドの説明にその旨記載されています。以下の定数は、現行バージョンのバージョン番号を指定しています。

MQMD_CURRENT_VERSION

メッセージ記述子の構造体の現行バージョン。

これは常に入力フィールドです。このフィールドの初期値は MQMD_VERSION_1 です。

Report (MQLONG)

レポート・メッセージは、他のメッセージに関するメッセージであり、元のメッセージに関する予定されたイベントまたは予定されていないイベントについてアプリケーションに知らせるために使用されます。Report フィールドを使用すると、元のメッセージを送信するアプリケーションは、どのレポート・メッセージが必要であるか、アプリケーション・メッセージ・データを組み込むかどうか、また、(レポートと応答のどちらの場合も) レポート・メッセージまたは応答メッセージの中のメッセージおよび相関 ID をどのように設定するかを指定することができます。以下のタイプのレポート・メッセージのいずれかまたはすべてを要求する (またはどれも要求しない) ことができます。

- 例外
- 有効期限
- 到着確認 (COA)
- 送達時に確認 (COD)
- 肯定アクション通知 (PAN)
- 否定アクション通知 (NAN)

以下に説明する 1 つ以上のオプションを指定できます。複数のオプションを指定するには、値を加算する (同じ定数を複数回加算しない) か、ビット単位 OR 演算を使用して値を結合 (プログラミング言語でビット演算がサポートされる場合) します。

レポート・メッセージを受け取ったアプリケーションでは、MQMD 内の *Feedback* フィールドを調べることによって、レポートが生成された理由を判別できます。詳細については、*Feedback* フィールドを参照してください。

メッセージをトピックに書き込むときにレポート・オプションを使用すると、ゼロ個、1 個、または複数のレポート・メッセージを生成し、アプリケーションに送信されることがあります。これは、パブリケーション・メッセージをゼロ個、1 個、または複数のサブスクライブ・アプリケーションに送信されることがあるためです。

例外オプション

例外レポートメッセージを要求するには、リストされているオプションのいずれかを指定してください。

MQRO_EXCEPTION

メッセージ・チャンネル・エージェントは、メッセージが別のキュー・マネージャーに送信され、指定されたターゲット・キューにメッセージを送達できない場合に、このタイプのレポートを生成します。例えば、ターゲット・キューまたは中間伝送キューがいっぱいである場合や、メッセージが大きすぎてキューに入らない場合などがあります。

例外レポート・メッセージが生成されるかどうかは、元のメッセージの持続性と、元のメッセージが経由するメッセージ・チャンネルの速度 (通常または高速) によって次のように異なります。

- すべての持続性メッセージと、通常のメッセージ・チャンネルを経由する非持続性メッセージについては、送信側アプリケーションでエラー条件に対して指定されたアクションが正常に完了した場合のみ、例外レポートが生成されます。送信側アプリケーションでは、エラー条件が発生したときの元のメッセージの後処理を制御するために、次のアクションのいずれかを指定できます。
 - MQRO_DEAD_LETTER_Q (元のメッセージを送達不能キューに入れます)。
 - MQRO_DISCARD_MSG (元のメッセージを廃棄します)。

送信側アプリケーションで指定したアクションを正常に完了できなかった場合は、元のメッセージは伝送キューに残され、例外レポート・メッセージは生成されません。

- 高速メッセージ・チャンネルを経由する非持続性メッセージについては、エラー条件に対して指定されたアクションを正常に完了できない場合でも、元のメッセージが伝送キューから削除され、例外レポートが生成されます。例えば、MQRO_DEAD_LETTER_Q が指定されている場合に、送達不能キューがいっぱいであるなどの理由で元のメッセージを送達不能キューに入れられなかったときには、例外レポート・メッセージが生成され、元のメッセージが廃棄されます。

通常メッセージ・チャンネルおよび高速メッセージ・チャンネルの詳細については、[非持続メッセージ速度 \(NPMSPEED\)](#) を参照してください。

MQPUT または MQPUT1 呼び出しから戻された理由コードを用いて、元のメッセージを書き込んだアプリケーションに問題を同期的に通知できる場合、例外レポートは生成されません。

アプリケーションは、例外レポートを送信することによってメッセージを処理できないことを通知することもできます (借方記帳のトランザクションにおいて、対応する貸方科目の限度を超えてしまう場合など)。

元のメッセージのメッセージ・データは、レポート・メッセージに組み込まれません。

MQRO_EXCEPTION、MQRO_EXCEPTION_WITH_DATA、および MQRO_EXCEPTION_WITH_FULL_DATA のいずれかを指定し、複数を指定しないでください。

MQRO_EXCEPTION_WITH_DATA

これは MQRO_EXCEPTION と同じです。ただし、元のメッセージのアプリケーション・メッセージ・データの最初の 100 バイトがレポート・メッセージに組み込まれます。元のメッセージに 1 つ以上の MQ ヘッダー構造が含まれている場合、それらはアプリケーション・データの 100 バイトに加えて、レポート・メッセージに組み込まれます。

MQRO_EXCEPTION、MQRO_EXCEPTION_WITH_DATA、および MQRO_EXCEPTION_WITH_FULL_DATA のいずれかを指定し、複数を指定しないでください。

MQRO_EXCEPTION_WITH_FULL_DATA

全データの例外レポートが必要です。

これは MQRO_EXCEPTION と同じです。ただし、元のメッセージのアプリケーション・メッセージ・データのすべてがレポート・メッセージに組み込まれます。

MQRO_EXCEPTION、MQRO_EXCEPTION_WITH_DATA、および MQRO_EXCEPTION_WITH_FULL_DATA のいずれかを指定し、複数を指定しないでください。

満期オプション

有効期限レポートメッセージをリクエストするには、記載されているオプションのいずれかを指定してください。

MQRO_EXPIRATION

このタイプのレポートは、有効期限が過ぎたためにアプリケーションへの送達の前にメッセージが廃棄された場合に、キュー・マネージャーによって生成されます (*Expiry* フィールドを参照)。このオプションが設定されていないと、上記の理由でメッセージが廃棄された場合に (MQRO_EXCEPTION_* オプションが指定されていても) レポート・メッセージは生成されません。

元のメッセージのメッセージ・データは、レポート・メッセージに組み込まれません。

MQRO_EXPIRATION、MQRO_EXPIRATION_WITH_DATA、および MQRO_EXPIRATION_WITH_FULL_DATA のいずれかを指定し、複数を指定しないでください。

MQRO_EXPIRATION_WITH_DATA

これは MQRO_EXPIRATION と同じです。ただし、元のメッセージのアプリケーション・メッセージ・データの最初の 100 バイトがレポート・メッセージに組み込まれます。元のメッセージに 1 つ以上の MQ ヘッダー構造が含まれている場合、それらはアプリケーション・データの 100 バイトに加えて、レポート・メッセージに組み込まれます。

MQRO_EXPIRATION、MQRO_EXPIRATION_WITH_DATA、および MQRO_EXPIRATION_WITH_FULL_DATA のいずれかを指定し、複数を指定しないでください。

MQRO_EXPIRATION_WITH_FULL_DATA

これは MQRO_EXPIRATION と同じです。ただし、元のメッセージのアプリケーション・メッセージ・データのすべてがレポート・メッセージに組み込まれます。

MQRO_EXPIRATION、MQRO_EXPIRATION_WITH_DATA、および MQRO_EXPIRATION_WITH_FULL_DATA のいずれかを指定し、複数を指定しないでください。

到着時確認オプション

到着確認レポートメッセージを要求するには、記載されているオプションのいずれかを指定してください。

MQRO_COA

このタイプのレポートは、メッセージがターゲット・キューに入れられるときに、ターゲット・キューを所有するキュー・マネージャーによって生成されます。元のメッセージのメッセージ・データは、レポート・メッセージに組み込まれません。

メッセージが作業単位の一部として書き込まれる場合で、ターゲット・キューがローカル・キューの場合、キュー・マネージャーによって生成される COA レポート・メッセージは、作業単位がコミットされる場合にのみ検索可能になります。

メッセージ記述子にある *Format* フィールドが、MQFMT_XMIT_Q_HEADER または MQFMT_DEAD_LETTER_HEADER である場合、COA レポートは生成されません。これにより、メッセ

ージが伝送キューに書き込まれる場合、またはメッセージが配信不能で送達不能キューに書き込まれる場合に、COA レポートは生成されなくなります。

IMS ブリッジ・キューの場合、COA レポートが生成されるのは、MQ ブリッジ・キューにメッセージが書き込まれたときではなく、メッセージが IMS キューに到達したとき (IMS から確認応答を受け取ったとき) です。つまり、IMS がアクティブでない場合は、IMS が開始されて IMS キューにメッセージが入られるまで COA レポートは生成されないことになります。

MQMD.Report=MQRO_COA でメッセージを書き込むプログラムを実行するユーザーには、応答キューに対する +passid 権限が必要です。ユーザーに +passid 権限がない場合、COA レポート・メッセージは応答キューに到達しません。このレポート・メッセージを送達不能キューに入れるように試行されます。

MQRO_COA、MQRO_COA_WITH_DATA、および MQRO_COA_WITH_FULL_DATA のいずれかを指定し、複数を指定しないでください。

MQRO_COA_WITH_DATA

これは MQRO_COA と同じです。ただし、元のメッセージのアプリケーション・メッセージ・データの最初の 100 バイトがレポート・メッセージに組み込まれます。元のメッセージに 1 つ以上の MQ ヘッダー構造が含まれている場合、それらはアプリケーション・データの 100 バイトに加えて、レポート・メッセージに組み込まれます。

MQRO_COA、MQRO_COA_WITH_DATA、および MQRO_COA_WITH_FULL_DATA のいずれかを指定し、複数を指定しないでください。

MQRO_COA_WITH_FULL_DATA

これは MQRO_COA と同じです。ただし、元のメッセージのアプリケーション・メッセージ・データのすべてがレポート・メッセージに組み込まれます。

MQRO_COA、MQRO_COA_WITH_DATA、および MQRO_COA_WITH_FULL_DATA のいずれかを指定し、複数を指定しないでください。

配達確認オプション

配達確認レポートメッセージを要求するには、記載されているオプションのいずれかを指定してください。

MQRO_COD

このタイプのレポートは、アプリケーションがターゲット・キューからメッセージを取得してメッセージをキューから削除するときに、キュー・マネージャーによって生成されます。元のメッセージのメッセージ・データは、レポート・メッセージに組み込まれません。

メッセージが作業単位の一部として検索される場合、レポート・メッセージは同じ作業単位内に生成され、作業単位がコミットされるまではそのレポートが利用できなくなります。作業単位がバックアウトされる場合、レポートは送信されません。

MQGMO_MARK_SKIP_BACKOUT オプションによりメッセージが取り出される場合は、COD レポートは常に生成されるとは限りません。1 次作業単位はバックアウトされたが、2 次作業単位はコミットされた場合、メッセージはキューから削除されますが、COD レポートは生成されません。

メッセージ記述子にある *Format* フィールドが MQFMT_DEAD_LETTER_HEADER である場合、COD レポートは生成されません。これにより、メッセージが配布できなくなり、送達不能キューに書き込まれる場合は、COD レポートが生成されなくなります。

宛先キューが XCF キューの場合、MQRO_COD は無効です。

MQRO_COD、MQRO_COD_WITH_DATA、および MQRO_COD_WITH_FULL_DATA のいずれかを指定し、複数を指定しないでください。

MQRO_COD_WITH_DATA

これは MQRO_COD と同じです。ただし、元のメッセージのアプリケーション・メッセージ・データの最初の 100 バイトがレポート・メッセージに組み込まれます。元のメッセージに 1 つ以上の MQ ヘッダー構造が含まれている場合、それらはアプリケーション・データの 100 バイトに加えて、レポート・メッセージに組み込まれます。

元のメッセージについての MQGET 呼び出しで MQGMO_ACCEPT_TRUNCATED_MSG が指定され、取り出されたメッセージに切り捨てが行われた場合、レポート・メッセージ内のアプリケーション・メッセージ・データの量は次のように環境によって異なります。

- z/OS では、次のうちの最小値です。
 - 元のメッセージの長さ
 - メッセージを取り出すために使用されたバッファの長さ
 - 100 バイト
- その他の環境では、次のうちの最小値です。
 - 元のメッセージの長さ
 - 100 バイト

宛先キューが XCF キューの場合は、MQRO_COD_WITH_DATA は無効です。

MQRO_COD、MQRO_COD_WITH_DATA、および MQRO_COD_WITH_FULL_DATA のいずれかを指定し、複数を指定しないでください。

MQRO_COD_WITH_FULL_DATA

これは MQRO_COD と同じです。ただし、元のメッセージのアプリケーション・メッセージ・データのすべてがレポート・メッセージに組み込まれます。

宛先キューが XCF の場合は、MQRO_COD_WITH_FULL_DATA は無効です。

MQRO_COD、MQRO_COD_WITH_DATA、および MQRO_COD_WITH_FULL_DATA のいずれかを指定し、複数を指定しないでください。

アクション通知オプション

受信アプリケーションに肯定応答または否定応答のレポートメッセージを送信するよう要求するには、記載されているオプションのいずれか一方または両方を指定してください。

MQRO_PAN

このタイプのレポートは、メッセージを取り出し、そのメッセージに従ってアクションを実行するアプリケーションにより生成されます。これは、メッセージで要求されたアクションが正常に実行されたことを示します。レポートを生成するアプリケーションは、レポートにデータを含めるかどうかを決定します。

メッセージを取り出すアプリケーションにこの要求を送らないと、キュー・マネージャーはこのオプションに基づくアクションを実行しません。メッセージを取り出すアプリケーションは、必要に応じてレポートを生成する必要があります。

MQRO_NAN

このタイプのレポートは、メッセージを取り出し、そのメッセージに従ってアクションを実行するアプリケーションにより生成されます。これは、メッセージで要求されたアクションが正常に実行されなかったことを示します。レポートを生成するアプリケーションは、レポートにデータを含めるかどうかを決定します。例えば、要求を実行できなかった理由を示すデータを含めることができます。

メッセージを取り出すアプリケーションにこの要求を送らないと、キュー・マネージャーはこのオプションに基づくアクションを実行しません。メッセージを取り出すアプリケーションは、必要に応じてレポートを生成する必要があります。

アプリケーションは、どの条件が肯定アクションに対応し、どの条件が否定アクションに対応するかを判別する必要があります。ただし、要求の一部だけが実行された場合には、要求があれば、PAN レポートではなく、NAN レポートを生成します。どの条件も、肯定アクションと否定アクションのいずれか一方に対応付け、両方には対応させてはなりません。

メッセージ識別子オプション

レポートメッセージ（または返信メッセージ）の *MsgId* ヘッダーがどのように設定されるかを制御するために、リストされているオプションのいずれかを指定してください。

MQRO_NEW_MSG_ID

これはデフォルトのアクションで、このメッセージの結果としてレポートまたは応答が生成される場合、レポートまたは応答メッセージの新規 *MsgId* が生成されます。

MQRO_PASS_MSG_ID

このメッセージの結果としてレポートまたは応答が生成される場合、このメッセージの *MsgId* がレポートまたは応答メッセージの *MsgId* にコピーされることを示します。

パブリケーション・メッセージの *MsgId* は、パブリケーションのコピーを受け取る各サブスクライバーごとに異なるため、レポートまたは応答メッセージにコピーされる *MsgId* もそれぞれ異なります。

このオプションを指定しない場合は、MQRO_NEW_MSG_ID が使用されます。

関連識別子オプション

レポートメッセージ（または返信メッセージ）の *CorrelId* ヘッダーがどのように設定されるかを制御するために、リストされているオプションのいずれかを指定してください。

MQRO_COPY_MSG_ID_TO_CORREL_ID

これはデフォルトのアクションで、このメッセージの結果としてレポートまたは応答が生成される場合、このメッセージの *MsgId* がレポートまたは応答メッセージの *CorrelId* にコピーされることを示します。

パブリケーション・メッセージの *MsgId* は、パブリケーションのコピーを受け取る各サブスクライバーごとに異なるため、レポートまたは応答メッセージの *CorrelId* にコピーされる *MsgId* もそれぞれ異なります。

MQRO_PASS_CORREL_ID

このメッセージの結果としてレポートまたは応答が生成される場合、このメッセージの *CorrelId* がレポートまたは応答メッセージの *CorrelId* にコピーされることを示します。

パブリケーション・メッセージの *CorrelId* は、MQSO_SET_CORREL_ID オプションを使用して MQSD の SubCorrelId フィールドを MQCI_NONE に設定しなければ、サブスクライバーに固有のものとなります。そのため、レポートまたは応答メッセージの *CorrelId* にコピーされる *CorrelId* がそれぞれ異なる可能性があります。

このオプションを指定しない場合は、MQRO_COPY_MSG_ID_TO_CORREL_ID が使用されます。

要求に応答したりレポート・メッセージを生成したりするサーバーでは、MQRO_PASS_MSG_ID または MQRO_PASS_CORREL_ID オプションが元のメッセージに設定されていたかをチェックする必要があります。それらのオプションが設定されていた場合、サーバーは、オプションに対応するアクションを実行しなければなりません。どちらのオプションも設定されていない場合、サーバーは、該当するデフォルト・アクションを実行する必要があります。

処分方法

元のメッセージが宛先キューに配信できない場合の処理方法を制御するには、リストされたオプションのいずれかを指定してください。アプリケーションは処理オプションを、例外レポートの要求とは無関係に設定できます。

MQRO_DEAD_LETTER_Q

これはデフォルトのアクションで、メッセージをターゲット・キューに送達できない場合に、メッセージを送達不能キューに入れます。これは次のような場合に行われます。

- MQPUT または MQPUT1 呼び出しによって戻された理由コードを用いて、元のメッセージを書き込んだアプリケーションに問題を同期的に通知できない場合。送信側によって要求される場合は、例外レポート・メッセージが生成されます。
- 元のメッセージを書き込むアプリケーションがトピックに書き込んでいた場合。

MQRO_DISCARD_MSG

これは、メッセージをターゲット・キューに送達できない場合にメッセージを廃棄します。これは次のような場合に行われます。

- MQPUT または MQPUT1 呼び出しによって戻された理由コードを用いて、元のメッセージを書き込んだアプリケーションに問題を同期的に通知できない場合。送信側によって要求される場合は、例外レポート・メッセージが生成されます。
- 元のメッセージを書き込むアプリケーションがトピックに書き込んでいた場合。

元のメッセージを送達不能キューに入れずに、元のメッセージを送信側に戻す場合、送信側は MQRO_DISCARD_MSG を MQRO_EXCEPTION_WITH_FULL_DATA と一緒に指定する必要があります。

MQRO_PASS_DISCARD_AND_EXPIRY

このオプションをメッセージで指定した結果として、レポートまたは応答が生成される場合、レポートのメッセージ記述子は以下を継承します。

- MQRO_DISCARD_MSG (設定されている場合)。
- メッセージの残りの有効期限時間 (有効期限レポートではない場合)。有効期限レポートの場合、有効期限時間は 60 秒に設定されます。

アクティビティオプション

MQRO_ACTIVITY

この値を使用すると、キュー・マネージャー・ネットワーク上の**すべての**メッセージの経路をトレースすることができます。このレポート・オプションはどの現行ユーザー・メッセージでも指定できます。これを指定すると、ネットワーク上のメッセージの経路の計算を即時に開始することができます。

メッセージを生成するアプリケーションがアクティビティ・レポートの生成を使用可能にすることができない場合は、キュー・マネージャー管理者が提供する API 交差出口を使ってレポート作成を使用可能にすることができます。

注：

1. アクティビティ報告書を生成できるキュー・マネージャーがネットワークに少なければ少ないほど、経路は単純なものとなります。
2. 取られた経路を判別するためにアクティビティ報告書を正しい順序に配列することが困難な場合があります。
3. アクティビティ報告書で、要求先への経路を見つけることができない場合があります。
4. このレポート・オプションを設定したメッセージは、そのオプションを認識していないとしても、すべてのキュー・マネージャーによって受け入れられなければなりません。これにより、あらゆるユーザー・メッセージで (それが IBM WebSphere MQ 6.0 以降以外のキュー・マネージャーによって処理される場合でも) レポート・オプションを設定することが可能となります。
5. プロセス (キュー・マネージャーまたはユーザー・プロセスのいずれか) で、このオプションが設定されたメッセージに対してアクティビティが実行される場合、アクティビティ報告書を生成し、配置することを選択することができます。

デフォルトの選択肢

レポートオプションが不要な場合は、以下を指定してください：

MQRO_NONE

この値を使用して、他のオプションが指定されていないことを示します。MQRO_NONE は、プログラム・ドキュメンテーションの援助機能として定義されています。このオプションを他のオプションと組み合わせて使用することは意図されていませんが、値がゼロであるため、そのような使い方をしても検出できません。

一般情報

1. 必要とされるすべてのレポート・タイプは、元のメッセージを送信するアプリケーション具体的に指定されている必要があります。例えば、COA レポートは要求されているが、例外レポートは要求されないという場合は、メッセージが宛先キューに入れられるときに COA レポートが生成されますが、メッセージが宛先キューに到着したときに宛先キューがいっぱいであっても、例外レポートは生成されません。Report オプションが設定されていない場合、キュー・マネージャーまたはメッセージ・チャネル・エージェント (MCA) によって、レポート・メッセージが生成されることはありません。

一部のレポート・オプションは、ローカル・キュー・マネージャーで認識できなくても指定できます。これは、宛先 キュー・マネージャーでオプションを処理する場合に有用です。詳細については、[898 ページの『レポート・オプションおよびメッセージ・フラグ』](#)を参照してください。

レポート・メッセージを要求するときは、そのレポートの送信先となるキューの名前を *ReplyToQ* フィールドで指定しなければなりません。レポート・メッセージを受信したとき、メッセージ記述子の中の *Feedback* フィールドを調べることによって、レポートの性質を判別することができます。

2. レポート・メッセージを生成するキュー・マネージャーまたは MCA が、レポート・メッセージを応答キューに入れることができない場合 (例えば応答キューまたは伝送キューがいっぱいになっているため)、レポート・メッセージは、代わりに送達不能キューに入ります。これにも失敗した場合、あるいは送達不能キューがない場合に、取るべきアクションは、レポート・メッセージのタイプによって決まります。

- レポート・メッセージが例外レポートである場合は、その例外レポートを生成したメッセージが伝送キューに残されます。これにより、メッセージが失われていないことが分かります。
- 他のすべてのレポート・タイプについては、レポート・メッセージは廃棄され、処理は引き続き正常に行われます。処理を実行する理由は、元のメッセージがすでに無事に送達されているか (COA または COD レポート・メッセージの場合)、元のメッセージがすでに対象外になっているか (満了レポート・メッセージの場合) のいずれかです。

レポート・メッセージが無事にキュー (宛先キューまたは中間伝送キューのいずれか) に入ると、そのメッセージには特別な処理は行われなくなります (他のメッセージと全く同じ扱いになります)。

3. レポートが生成されると、*ReplyToQ* キューが開き、レポートを生成させたメッセージの MQMD 内の *UserIdentifier* の権限を使用してレポート・メッセージが書き込まれます。ただし、以下の場合は例外です。
 - 受信側の MCA によって生成される例外レポートは、レポートを生成させたメッセージを書き込むために MCA が使用したものと同一許可を使って書き込まれます。
 - キュー・マネージャーによって生成される COA レポートは、レポートを生成させたメッセージをそのキュー・マネージャーに書き込むときに使用されたものと同一権限を使って書き込まれます。例えば、受信側の MCA がその MCA のユーザー ID を使ってメッセージを書き込んだ場合、キュー・マネージャーは MCA のユーザー ID を使って COA レポートを書き込みます。

レポートを生成するアプリケーションには、通常、応答を生成する場合と同じ権限が必要です。これは、通常、元のメッセージのユーザー ID に与えられた権限と同じです。

レポートがリモート宛先に移動しなければならない場合は、他のメッセージに対する場合と同じ方式で、それを受け取るかどうかを、送信側と受信側が決めることができます。

4. データ付きのレポート・メッセージが要求される場合は、以下のことが行われます。
 - レポート・メッセージは、常に、元のメッセージの送信側が要求したデータ量で生成されます。レポート・メッセージが応答キューに対して大きすぎると、上記の処理が行われます。レポート・メッセージは、応答キューに適合するために切り捨てられることはありません。
 - 元のメッセージの *Format* が、MQFMT_XMIT_Q_HEADER である場合、レポートに組み込まれるデータには、MQXQH は含まれません。レポート・データは、元のメッセージにある MQXQH の後にあるデータの最初のバイトから始まります。これは、キューが伝送キューであるかどうかに関係なく起こります。
5. COA、COD、または満了レポート・メッセージが、応答キューで受け取られた場合は、元のメッセージが、適切に到着した、送達された、または満了したことが保証されます。しかし、上記のレポート・メッセージが 1 つ以上が要求されていて、それが受け取られない場合でも、その逆を想定することはできません。これは、以下のいずれかが起こった可能性があるためです。
 - a. リンクがダウンしたために、レポート・メッセージは中止された。
 - b. 中間伝送キューまたは応答キューに、ブロッキング条件 (キューがいっぱいである、書き込みが禁止されている、など) が存在しているため、レポート・メッセージは中止された。
 - c. レポート・メッセージが送達不能キューに入っている。

- d. キュー・マネージャーがレポート・メッセージの生成を試行したが、適切なキューにメッセージを書き込むことも送達不能キューに書き込むこともできなかったために、レポート・メッセージを生成することができなかった。
- e. 報告されるアクション(到着、送達、または満了)と、それに対応するレポート・メッセージの生成との間で、キュー・マネージャーの障害が起こった。(COD レポート・メッセージは同じ作業単位内で生成されるので、COD レポート・メッセージではアプリケーションが作業単位内で元のメッセージを取り出す場合は上記のことは起こりません。)

例外レポート・メッセージも、上記の 1、2 および 3 の理由から、同様に中止されることがあります。しかし、MCA が例外レポート・メッセージを生成できない(応答キューと送達不能キューのどちらにもレポート・メッセージを書き込むことができない)場合は、元のメッセージは送信側の伝送キューにとどまり、チャンネルがクローズされます。このような状況は、レポート・メッセージがチャンネルの送信側、受信側のどちらで生成されても起こります。

6. 元のメッセージが一時的にブロック状態になっていた(例外レポート・メッセージが生成され、元のメッセージが送達不能キューに書き込まれる結果になった)が、そのブロック状態がクリアされ、さらにアプリケーションが送達不能キューから元のメッセージを読み取って、その宛先に再び書き込んだ場合は、以下のことが起こる可能性があります。
 - ・ 例外レポート・メッセージは生成されたが、最終的に元のメッセージは、その宛先に正しく到着する。
 - ・ 元のメッセージは、後で別のブロック状態になる可能性があるため、元のメッセージ 1 つに対して複数の例外レポート・メッセージが生成される。

トピックに投稿する際にメッセージを報告する

1. メッセージをトピックに書き込むときに、レポートを生成することができます。このメッセージはトピックのすべてのサブスクライバーに送信されます。ゼロの場合もあれば、1 またはそれ以上の場合もあります。結果として多数のレポート・メッセージが生成される場合があるので、レポート・オプションを使用するように選択する場合には、これを考慮に入れる必要があります。
2. メッセージをトピックに書き込むときに、メッセージのコピーを置く宛先キューが多数存在する場合があります。そのうちのいくつかの宛先キューに問題(例えばキュー・フル)がある場合、MQPUT の実行が成功するかどうかは(メッセージの持続性に依拠して)NPMSGDLV または PMSGDLV の設定に依存します。宛先キューへのメッセージ送達が必ず成功するという設定(例えばそれが永続サブスクライバーへの持続メッセージで、PMSGDLV が ALL または ALLDUR に設定されている)の場合、成功とは、次の基準のいずれかが満たされる場合として定義されます。
 - ・ サブスクライバー・キューへの書き込みが成功する
 - ・ MQRO_DEAD_LETTER_Q を使用し、サブスクライバー・キューがメッセージを受け取ることができない場合に送達不能キューに正常に書き込まれる
 - ・ サブスクライバー・キューがメッセージを受け取ることができない場合に MQRO_DISCARD_MSG を使用する

メッセージセグメントのメッセージを報告する

1. セグメント化が許可されている(MQMF_SEGMENTATION_ALLOWED フラグの項を参照)メッセージについてレポート・メッセージを要求することができます。キュー・マネージャーがメッセージをいくつかのセグメントに分割する必要があると判断した場合には、関連する条件に応じてセグメントごとにレポート・メッセージが生成されます。アプリケーションでは、要求したレポート・メッセージのタイプごとに複数のレポート・メッセージを受信できるようにしておく必要があります。レポート・メッセージの GroupId フィールドを使用して、元のメッセージのグループ ID によって複数のレポートと関連付けます。また、Feedback フィールドを使用して、各レポート・メッセージのタイプを識別します。
2. MQGMO_LOGICAL_ORDER を使用してセグメントに関するレポート・メッセージを取り出す場合には、MQGET 呼び出しを続けて発行したときに異なるタイプのレポートが戻されることがあるので注意してください。例えば、キュー・マネージャーによってセグメントに分割されたメッセージに対して COA レポートおよび COD レポートを要求されている場合、それらのレポート・メッセージに対して MQGET 呼び出しを発行したときに COA レポート・メッセージと COD レポート・メッセージが順不同で混ざり合ってしまうことがあります。これを回避するには、MQGMO_COMPLETE_MSG オプションを使用します(MQGMO_ACCEPT_TRUNCATED_MSG を組み合わせても構いません)。MQGMO_COMPLETE_MSG を使用すると、キュー・マネージャーは同じレポート・タイプのレポート・メッセージを再度組み立てます。例えば、最初の MQGET 呼び出しで元のメッセージに関連するすべての COA メッセージの再組み

立てを実行し、2 番目の MQGET 呼び出しですべての COD メッセージの再組み立てを実行する場合などが考えられます。どちらの再組み立てが先に実行されるかは、どちらのタイプのレポート・メッセージが先にキューに入るかによります。

3. アプリケーション自体がセグメントを書き込む場合、そのアプリケーションではセグメントごとに異なるレポート・オプションを指定できます。ただし、以下の点に注意してください。
 - MQGMO_COMPLETE_MSG オプションを使用してセグメントを取り出している場合、キュー・マネージャーに格納されるのは、最初のセグメント内のレポート・オプションだけです。
 - セグメントを一度に 1 つずつ取り出している場合に、大部分のセグメントに MQRO_COD_* オプションのいずれかが指定されていても、このオプションが指定されていないセグメントが 1 つでもあれば、MQGMO_COMPLETE_MSG オプションを使用して 1 回の MQGET 呼び出しでレポート・メッセージを取り出したり、MQGMO_ALL_SEGMENTS_AVAILABLE オプションを使用してすべてのレポート・メッセージが到着した時間を検出したりすることができません。
4. MQ ネットワークのキュー・マネージャーはそれぞれ異なる機能を持つ場合があります。セグメント化をサポートしていないキュー・マネージャーまたは MCA によりセグメントに対するレポート・メッセージが生成された場合、そのキュー・マネージャーまたは MCA は、デフォルト時には、レポート・メッセージの中に必要なセグメント情報を含めません。そのため、このレポートを生成する原因となった元のメッセージを識別することが困難な場合があります。この問題を回避するには、レポート・メッセージと共にデータを要求します (MQRO_*_WITH_DATA オプションと MQRO_*_WITH_FULL_DATA オプションのうちどちらか該当する方を指定する)。ただし、MQRO_*_WITH_DATA を指定した場合には、100 バイト未満のアプリケーション・メッセージ・データがレポート・メッセージを取り出すアプリケーションに戻されることがあるので注意してください (レポート・メッセージがセグメント化をサポートしていないキュー・マネージャーまたは MCA により生成された場合)。

レポートメッセージのメッセージ記述子の内容

キューマネージャまたはメッセージチャネルエージェント (MCA) がレポートメッセージを生成する際、メッセージ記述子のフィールドを以下の値に設定し、その後通常の方法でメッセージを配置する。

表 519. レポート・メッセージがシステム生成される場合に MQMD フィールドに使用される値

MQMD のフィールド	使用される値
<i>StrucId</i>	MQMD_STRUC_ID
<i>Version</i>	MQMD_VERSION_2
<i>Report</i>	MQRO_NONE
<i>MsgType</i>	MQMT_REPORT
<i>Expiry</i>	MQEI_UNLIMITED
<i>Feedback</i>	レポートの性質に応じたもの (MQFB_COA、MQFB_COD、MQFB_EXPIRATION、または MQRC_* 値)
<i>Encoding</i>	元のメッセージ記述子からコピーされます。
<i>CodedCharSetId</i>	元のメッセージ記述子からコピーされます。
<i>Format</i>	元のメッセージ記述子からコピーされます。
<i>Priority</i>	元のメッセージ記述子からコピーされます。
<i>Persistence</i>	元のメッセージ記述子からコピーされます。
<i>MsgId</i>	元のメッセージ記述子のレポート・オプションで指定されたもの
<i>CorrelId</i>	元のメッセージ記述子のレポート・オプションで指定されたもの
<i>BackoutCount</i>	0
<i>ReplyToQ</i>	ブランク
<i>ReplyToQMgr</i>	キュー・マネージャーの名前。
<i>UserIdentifier</i>	MQPMO_PASS_IDENTITY_CONTEXT オプションで設定されたもの

表 519. レポート・メッセージがシステム生成される場合に MQMD フィールドに使用される値 (続き)

MQMD のフィールド	使用される値
<i>AccountingToken</i>	MQPMO_PASS_IDENTITY_CONTEXT オプションで設定されたもの
<i>ApplIdentityData</i>	MQPMO_PASS_IDENTITY_CONTEXT オプションで設定されたもの
<i>PutApplType</i>	MQAT_QMGR またはメッセージ・チャネル・エージェントに適切なもの
<i>PutApplName</i>	キュー・マネージャー名またはメッセージ・チャネル・エージェント名の最初の 28 バイト。IMS ブリッジで生成されたレポート・メッセージの場合、このフィールドには、メッセージに関連する IMS システムの XCF グループ名と XCF メンバー名が入っています。
<i>PutDate</i>	レポート・メッセージが送信された日付
<i>PutTime</i>	レポート・メッセージが送信された時刻
<i>ApplOriginData</i>	ブランク
<i>GroupId</i>	元のメッセージ記述子からコピーされます。
<i>MsgSeqNumber</i>	元のメッセージ記述子からコピーされます。
<i>Offset</i>	元のメッセージ記述子からコピーされます。
<i>MsgFlags</i>	元のメッセージ記述子からコピーされます。
<i>OriginalLength</i>	MQOL_UNDEFINED 以外の場合は、元のメッセージ記述子からコピーされる。MQOL_UNDEFINED の場合は、元のメッセージ・データの長さが設定される。

レポートを生成するアプリケーションでは、以下を除いて、同様の値を設定することをお勧めします。

- *ReplyToQMGR* フィールドは、ブランクに設定することができます (メッセージを書き込むときに、キュー・マネージャーが、フィールドをローカル・キュー・マネージャーの名前に変更します)。
- コンテキスト・フィールドは、応答に対して使用されるはずのオプション (通常は MQPMO_PASS_IDENTITY_CONTEXT) を用いて設定します。

レポートフィールドの分析

フィールド *Report* はサブフィールドを含みます。このため、メッセージの送信者が特定のレポートを要求したかどうかを確認する必要があるアプリケーションは、[900 ページの『レポート・フィールドの分析』](#)で説明されている手法のいずれかを使用しなければなりません。

これは、MQGET 呼び出しでは出力フィールド、MQPUT および MQPUT1 呼び出しでは入力フィールドです。このフィールドの初期値は MQRO_NONE です。

MsgType (MQLONG)

これは、メッセージ・タイプを示します。メッセージ・タイプは、以下のようにグループ化されています。

MQMT_SYSTEM_FIRST

システム定義のメッセージ・タイプに関する最低値。

MQMT_SYSTEM_LAST

システム定義のメッセージ・タイプに関する最高値。

現在、以下の値がシステム範囲内で定義されています。

MQMT_DATAGRAM

メッセージは、応答が不要なメッセージです。

MQMT_REQUEST

メッセージは、応答が必要なメッセージです。

応答の送信先キューの名前を、*ReplyToQ* フィールドに指定します。この *Report* フィールドは、応答のヘッダー *MsgId* と *CorrelId* ボディの設定方法を示します。

MQMT_REPLY

メッセージは、以前の要求メッセージ (MQMT_REQUEST) に対する応答です。メッセージは、要求メッセージの *ReplyToQ* フィールドで指示されるキューに送信される必要があります。応答の *MsgId* および *CorrelId* の設定方法を制御するには、要求の *Report* フィールドを使用します。

注: キュー・マネージャーは、要求と応答の関係を強制することはありません。これはアプリケーションが担当します。

MQMT_REPORT

メッセージは、通常は他のメッセージに関連して (例えば、有効でないデータを含む要求メッセージを受信した)、予期されていた結果または予期されていない結果を報告しています。このメッセージを元のメッセージのメッセージ記述子の *ReplyToQ* フィールドで指示されるキューに送信します。

Feedback フィールドを設定して、レポートの種類を指示します。レポート・メッセージの *MsgId* および *CorrelId* の設定方法を制御するには、元のメッセージの *Report* フィールドを使用します。

キュー・マネージャーまたはメッセージ・チャネル・エージェントによって生成されたレポート・メッセージは、上記のように設定された *Feedback* および *CorrelId* フィールドと共に、常に *ReplyToQ* キューに送信されます。

アプリケーション定義の値を使用することもできます。ただし、以下の範囲内でなければなりません。

MQMT_APPL_FIRST

アプリケーション定義のメッセージ・タイプに関する最低値。

MQMT_APPL_LAST

アプリケーション定義のメッセージ・タイプに関する最高値。

MQPUT および MQPUT1 呼び出しでは、*MsgType* 値は、システム定義の範囲またはアプリケーション定義の範囲のいずれかの範囲内になければなりません。そうではない場合は呼び出しが失敗し、理由コード MQRC_MSG_TYPE_ERROR が返されます。

これは、MQGET 呼び出しでは出力フィールド、MQPUT および MQPUT1 呼び出しでは入力フィールドです。このフィールドの初期値は MQMT_DATAGRAM です。

Expiry (MQLONG)

これは、メッセージを書き込むアプリケーションで設定される時間で、10 分の 1 秒単位で表されます。この時間が経過するまでに宛先キューからメッセージが除去されなかった場合、そのメッセージは廃棄の対象となります。

例えば、有効期限を 1 分に設定するには、. を **MQMD** 設定する必要があります。**Expiry 600** まで

この値は、メッセージが宛先キューに滞在する時間に応じて、またメッセージがリモート・キューに対する書き込みである場合は中間の伝送キューに存在した時間に応じて、減少します。また、伝送にかなりの時間がかかった場合は、その伝送の時間に応じてメッセージ・チャネル・エージェントによって値を減少させることがあります。同様に、このメッセージを別のキューに送るアプリケーションも、長時間にわたってメッセージを保持した場合には、必要に応じて値を減分することがあります。しかし、満了時間は概数として扱われるので、短い時間間隔の調節のためにこの値を減分する必要はありません。

アプリケーションが MQGET 呼び出しを使ってメッセージを取り出したとき、*Expiry* フィールドは、有効期限の残り時間を表します。

メッセージの有効期限を過ぎると、メッセージはキュー・マネージャーによって廃棄される対象となります。まだ満了していないメッセージを戻すブラウズまたは非ブラウズの MQGET 呼び出しが発行されたときに、メッセージが廃棄されます。例えば、MQGMO の *MatchOptions* フィールドが MQMO_NONE に設定された非ブラウズ MQGET 呼び出しにおいて FIFO 順のキューから読み取りを実行する場合、満了していないメッセージが出現するまで、満了しているメッセージはすべて廃棄されます。優先順位方式のキューで同じ呼び出しを発行した場合は、満了していない最初のメッセージより先にキューに到着した満了したメッセージのうち、優先順位が等しいメッセージとそれより優先順位が高いメッセージが廃棄されます。

ブラウズまたは非ブラウズのどちらの MQGET 呼び出しを使用しても満了したメッセージがアプリケーションに戻されることはないので、MQGET 呼び出しが正常に終了した後、メッセージ記述子の *Expiry* フィールドはゼロより大きい値か、特殊な値である MQEI_UNLIMITED になります。

メッセージをリモート・キューに書き込む場合、メッセージは、宛先キューに到達する前の中間伝送キューにある間に満了してしまう (そして廃棄される) 可能性もあります。

メッセージに MQRO_EXPIRATION_* レポート・オプションの 1 つが指定されている場合は、満了したメッセージが廃棄される時に、レポートが生成されます。オプションがまったく指定されていない場合、そうしたレポートは生成されません。指定時間の経過後は、このメッセージは関係がなくなったと見なされます (後のメッセージに置き換わったと考えられるため)。

同期点までに書き込まれたメッセージの有効期限間隔が開始するのは、同期点がコミットされた時点ではなく、メッセージが書き込まれた時点です。この有効期限間隔は、同期点がコミットされる前に終わる可能性もあります。その場合、メッセージはコミット操作の少し後に破棄されるため、MQGET 操作への応答としてアプリケーションに返されることはありません。

有効期限に基づいてメッセージを廃棄する他のプログラムはいずれも、要求に応じて、該当のレポート・メッセージを送らなければなりません。

注:

1. メッセージが *Expiry* 時間をゼロまたは 999 999 999 より大きい数値で書き込まれた場合、MQPUT または MQPUT1 呼び出しは失敗し、理由コード MQRC_EXPIRY_ERROR が戻ります。この場合、レポート・メッセージは生成されません。

理由コード 2013、MQRC_EXPIRY_ERROR を有効にするには、環境変数 AMQ_ENFORCE_MAX_EXPIRY_ERROR を有効にする必要があります。

次に示すのは、Linux の例です。

```
$ export AMQ_ENFORCE_MAX_EXPIRY_ERROR=True
```

次の点に注意してください。

- 重要なのは、変数をエクスポートすることです。
 - 実際の値は無視されますが、セットアップを検討する際には **True** を使用すると役立つ場合があります。
2. 有効期限を過ぎたメッセージが、すぐには廃棄されないこともあるので、有効期限を過ぎているため取り出しの対象にならないメッセージがキューに入っている可能性もあります。それにもかかわらず、これらのメッセージは、キュー・サイズのトリガーなどのあらゆる目的のために、キュー内のメッセージ数のカウントに含まれます。

サブスクライバー／コンシューマー（クライアント）がメッセージを取得しようとした際、そのメッセージが期限切れの場合、クライアントは何も受信しません。メッセージが古すぎたため破棄されたためです。さらに、クライアントはエラーメッセージを一切受け取らない。
 3. 満了レポートは、廃棄の対象となったときでなく、メッセージが廃棄される時に、要求に応じて生成されます。
 4. 満了メッセージの廃棄および満了レポートの生成 (要求がある場合) は、アプリケーションの作業単位の一部ではありません。これは、作業単位の中で動作する MQGET 呼び出しの結果、メッセージが廃棄されるようにスケジュールされていた場合でも同じです。
 5. 満了間近のメッセージが作業単位の中で MQGET 呼び出しによって取り出され、そのあとで、その作業単位がバックアウトされると、メッセージが廃棄の対象になり再び取り出すことができないようになります。
 6. 満了間近のメッセージが MQGMO_LOCK を指定した MQGET 呼び出しによってロックされている場合は、そのメッセージが廃棄の対象になり、MQGMO_MSG_UNDER_CURSOR を指定した MQGET 呼び出しで取り出せなくなることがあります。このような場合は、その後の MQGET 呼び出しが発行されると、理由コード MQRC_NO_MSG_UNDER_CURSOR が戻ります。
 7. 有効期限時刻がゼロよりも大きい要求メッセージを取り出す場合、アプリケーションは応答メッセージを送信するときに以下のいずれかのアクションを実行できます。

- 有効期限までの残り時間を、要求メッセージから応答メッセージにコピーする。
- 応答メッセージ中の有効期限を、ゼロより大きい明示的な値に設定する。
- 応答メッセージ中の有効期限を MQEI_UNLIMITED に設定する。

実行されるアクションは、アプリケーションの設計によって異なります。ただし、送達不能 (未配布メッセージ) キューにメッセージを書き込む際のデフォルト・アクションは、メッセージの有効期限の残りをそのまま引き継ぎ、それを減分し続けるという動作でなければなりません。

8. トリガー・メッセージは、常に、MQEI_UNLIMITED で生成されます。
9. MQFMT_XMIT_Q_HEADER の *Format* 名をもつメッセージ (通常は伝送キューに入っている) は、MQXQH 内に 2 番目のメッセージ記述子を持っています。したがって、それに関連付けられた 2 つの *Expiry* フィールドを持っています。この場合、以下の点に注意してください。
 - アプリケーションがリモート・キューにメッセージを書き込む場合、キュー・マネージャーは、メッセージを最初にローカル伝送キューに入れ、MQXQH 構造を用いてアプリケーション・メッセージ・データに接頭部を付加します。キュー・マネージャーは、2 つの *Expiry* フィールドを、アプリケーションによって指定された値と同じ値に設定します。

アプリケーションがローカル伝送キューにメッセージを直接書き込む場合は、メッセージ・データの先頭は既に MQXQH 構造になっており、形式名は MQFMT_XMIT_Q_HEADER でなければなりません。この場合、アプリケーションは 2 つの *Expiry* フィールドを同じ値に設定する必要はありません (キュー・マネージャーは、MQXQH 内部の *Expiry* フィールドに有効な値が含まれているかどうか、またメッセージ・データがその値を保管できるだけの長さであるかを検査します)。伝送キューに直接書き込むことのできるアプリケーションの場合は、アプリケーションは組み込みメッセージ記述子を持つ伝送キュー・ヘッダーを作成する必要があります。ただし、伝送キューに書き込まれるメッセージ記述子の満了値が、組み込みメッセージ記述子の値と矛盾する場合、満了エラー拒否が発生します。

 - MQFMT_XMIT_Q_HEADER の *Format* 名をもつメッセージがキューから (通常のキューであっても、伝送キューであっても) 取り出される場合、キュー・マネージャーは、両方の *Expiry* フィールドからキューでの待機経過時間を差し引きます。メッセージ・データに MQXQH の *Expiry* フィールドを入れる十分な長さが無い場合も、エラーは発生しません。
 - キュー・マネージャーは、別個のメッセージ記述子 (つまり、MQXQH 構造体内部に埋め込まれているメッセージ記述子以外のもの) の *Expiry* フィールドを使用して、メッセージが廃棄の対象になるかどうかをテストします。
 - 2 つの *Expiry* フィールドの初期値が異なる場合、別個のメッセージが取り出された時点で MQXQH 内の *Expiry* フィールドに指定した時間は経過していても、そのメッセージ記述子の中の *Expiry* 時間は、ゼロより大きい (したがってメッセージは廃棄の対象にならない) 可能性があります。この場合、MQXQH の *Expiry* フィールドは、ゼロに設定されます。

10. IMS ブリッジから返される応答メッセージの有効期限は、MQIIH の *Flags* フィールドで MQIIH_PASS_EXPIRATION が設定されていない限り、無制限になります。詳しくは、[Flags](#) を参照してください。

以下のような特殊値が認識されます。

MQEI_UNLIMITED

メッセージは、無制限の満了時間を指定されています。

これは、MQGET 呼び出しでは出力フィールド、MQPUT および MQPUT1 呼び出しでは入力フィールドです。このフィールドの初期値は MQEI_UNLIMITED です。

関連項目 [CAPEXPY: メッセージの有効期限を設定する](#)



期限切れのメッセージ z/OS

IBM MQ for z/OS では、期限切れのメッセージは、次の適切な MQGET 呼び出しによって廃棄されます。

ただし、そのような呼び出しがないと、満了したメッセージが廃棄されないため、一部のキューでは、多くの満了メッセージが累積されることがあります。このような状態にならないようにするために、以下の

いずれかの方法により、定期的にキューをスキャンするようにキュー・マネージャーを設定し、1つ以上のキューの満了メッセージを廃棄させます。

定期的なスキャン

EXPRYINT (満了間隔) キュー・マネージャー属性を使用して期間を指定することができます。満了間隔に達するたびに、キュー・マネージャーは、満了メッセージを廃棄するためのスキャンの対象となるキューを探します。

キュー・マネージャーは、各キュー上の満了メッセージに関する情報を維持しているので、満了メッセージについてスキャンを行う価値があるかどうかが分かります。したがって、キューの選択のみはいつでもスキャンされます。

共有キューのスキャンは、キュー共有グループ内の1つのキュー・マネージャーによってのみ行われます。通常、最初のキュー・マネージャーが再始動するか、または最初のキュー・マネージャーにEXPRYINTが設定されます。このキュー・マネージャーが終了すると、キュー共有グループ内の別のキュー・マネージャーがキュー・スキャンを引き継ぎます。キュー共有グループ内のすべてのキュー・マネージャーについて、満了間隔値を同じ値に設定します。

EXPRYINT 設定にかかわらず、キュー・マネージャーの再始動時に各キューに対して期限切れ処理が行われることに注意してください。

明示的な要求

スキャンするキューを指定して、REFRESH QMGR TYPE(EXPIRY) コマンドを出します。

Feedback (MQLONG)

Feedback フィールドは、MQMT_REPORT タイプのメッセージと共に使用され、レポートの性質を表します。また、メッセージのタイプがこのタイプの場合に限り有効です。

このフィールドには、値 MQFB_* のいずれか、または値 MQRC_* のいずれかを指定できます。フィードバック・コードは、以下のようにグループ化されています。

MQFB_NONE

フィードバックが提供されていない。

MQFB_SYSTEM_FIRST

システム生成のフィードバックの最低値。

MQFB_SYSTEM_LAST

システム生成のフィードバックの最高値。

システム生成のフィードバック・コードの範囲 MQFB_SYSTEM_FIRST から MQFB_SYSTEM_LAST には、このトピックにリストされている一般的なフィードバック・コード (MQFB_*) および、メッセージを宛先キューに書き込めないときに戻る可能性のある理由コード (MQRC_*) も含まれています。

MQFB_APPL_FIRST

アプリケーション生成のフィードバックの最低値。

MQFB_APPL_LAST

アプリケーション生成のフィードバックの最高値。

レポート・メッセージを生成するアプリケーションでは、キュー・マネージャーまたはメッセージ・チャネル・エージェントによって生成されるレポート・メッセージをシミュレートするとき以外は、システム範囲にあるフィードバック・コード (MQFB_QUIT を除く) を使用してはなりません。

MQPUT または MQPUT1 呼び出しでは、指定される値は、MQFB_NONE、あるいはシステム範囲内またはアプリケーション範囲内でなければなりません。これは、MsgType の値にかかわらずチェックされます。

一般的なフィードバック・コード:

MQFB_COA

宛先キューへの到着の確認 (MQRO_COA 参照)。

MQFB_COD

受信側アプリケーションへの配布の確認 (MQRO_COD 参照)。

MQFB_EXPIRATION

メッセージは、有効期限が切れる前に宛先キューから除去されなかったため、廃棄されました。

MQFB_PAN

アクションの正常終了通知 (MQRO_PAN 参照)。

MQFB_NAN

アクションの異常終了通知 (MQRO_NAN 参照)。

MQFB_QUIT

アプリケーションの終了。

実行中のアプリケーション・プログラムのインスタンス数を制御するために、ワークロード・スケジューリング・プログラムによってのみ使用されます。このフィードバック・コードと共に MQMT_REPORT メッセージをアプリケーション・プログラムのインスタンスに送信すると、そのインスタンスに処理を停止するよう指示したことになります。しかし、この規則の順守はアプリケーション側の問題であり、キュー・マネージャーでは強制しません。

チャネルフィードバックコード

MQFB_CHANNEL_COMPLETED

チャネルは正常に終了しました。

MQFB_CHANNEL_FAIL

チャネルは異常終了し、STOPPED 状態になります。

MQFB_CHANNEL_FAIL_RETRY

チャネルは異常終了し、RETRY 状態になります。

IMS-ブリッジフィードバックコード

予期しない IMS-OTMA センス・コードを受け取った場合に、これらのコードが使用されます。そのセンス・コード、またはセンス・コードが 0x1A の場合はそのセンス・コードに関連した理由コードが、*Feedback* に示されます。

1. MQFB_IMS_FIRST (301) から MQFB_IMS_LAST (399) の範囲のフィードバックコードについて、0x1A 以外のセンスコードが受信されました。符号は、式 (フィードバック - MQFB_IMS_FIRST+1) によって与えられる。-OTMA IMS センスコードの意味および対応する理由コードについては、『IMS』のドキュメントにある「[NAK メッセージの OTMA センスコード](#)」の項に記載されています。
2. MQFB_IMS_NACK_1A_REASON_FIRST (600) から MQFB_IMS_NACK_1A_REASON_LAST (855) の範囲の *Feedback* コードで、0x1A のセンス・コードを受け取った。センスコードに関連付けられた理由コードは、式 (フィードバック - MQFB_IMS_NACK_1A_REASON_FIRST) によって与えられる。『IMS』のドキュメントにある「[001A](#)」のメッセージを参照してください。

IMS ブリッジが生成するフィードバック・コードは次のとおりです。

MQFB_DATA_LENGTH_ZERO

セグメント長が、メッセージのアプリケーション・データにおいてゼロであった。

MQFB_DATA_LENGTH_NEGATIVE

セグメント長が、メッセージのアプリケーション・データにおいて負であった。

MQFB_DATA_LENGTH_TOO_BIG

セグメント長が、メッセージのアプリケーション・データにおいて大き過ぎた。

MQFB_BUFFER_OVERFLOW

長さフィールドのどれか 1 つの値が原因で、データがメッセージ・バッファーからオーバーフローしました。

MQFB_LENGTH_OFF_BY_ONE

長さフィールドのどれか 1 つの値が、1 バイト短すぎます。

MQFB_IIH_ERROR

MQMD の *Format* フィールドには MQFMT_IMS が指定されていますが、メッセージは有効な MQIIH 構造で始まっていません。

MQFB_NOT_AUTHORIZED_FOR_IMS

メッセージ記述子 MQMD に含まれているユーザー ID、または MQIIH 構造体の *Authenticator* フィールドに含まれているパスワードが、IMS ブリッジで実行された検証に失敗しました。その結果、メッセージは IMS に渡されませんでした。

MQFB_IMS_ERROR

予期しないエラーが IMS から戻されました。エラーについて詳しくは、IMS ブリッジが存在しているシステムの IBM MQ エラー・ログを調べてください。

MQFB_IMS_FIRST

- IMSOTMA センスコードが 0x1A でない場合、IMS 生成されるフィードバックコードは MQFB_IMS_FIRST (301) から MQFB_IMS_LAST (399) の範囲になります。IMS-OTMA センス・コード自体は、*Feedback* から MQFB_IMS_ERROR を引いた値です。

MQFB_IMS_LAST

センス・コードが 0x1A ではない場合に IMS で生成されるフィードバックの最高値。

MQFB_IMS_NACK_1A_REASON_FIRST

センス・コードが 0x1A の場合に IMS で生成されるフィードバック・コードは MQFB_IMS_NACK_1A_REASON_FIRST (600) から MQFB_IMS_NACK_1A_REASON_LAST (855) までの範囲になります。

MQFB_IMS_NACK_1A_REASON_LAST

センス・コードが 0x1A の場合に IMS で生成されるフィードバックの最高値。

CICS-ブリッジフィードバックコード

以下のフィードバックコードは、CICS bridge: によって生成されます

MQFB_CICS_APPL_ABENDED

メッセージ内で指定したアプリケーション・プログラムが異常終了しました。このフィードバック・コードは、MQDLH 構造の *Reason* フィールドのみに戻されます。

MQFB_CICS_APPL_NOT_STARTED

メッセージ内で指定したアプリケーション・プログラムに関する EXEC CICS LINK が失敗しました。このフィードバック・コードは、MQDLH 構造の *Reason* フィールドのみに戻されます。

MQFB_CICS_BRIDGE_FAILURE

通常のエラー処理を完了せずに CICS bridge が異常終了しました。

MQFB_CICS_CCSID_ERROR

文字セット ID が無効です。

MQFB_CICS_CIH_ERROR

CICS 情報ヘッダー構造体が欠落しているか、または無効です。

MQFB_CICS_COMMAREA_ERROR

CICS commarea の長さが無効です。

MQFB_CICS_CORREL_ID_ERROR

相関 ID が無効です。

MQFB_CICS_DLQ_ERROR

CICS bridge・タスクが、この要求に対する応答を送達不能キューにコピーできませんでした。要求はバックアウトされました。

MQFB_CICS_ENCODING_ERROR

エンコードが無効です。

MQFB_CICS_INTERNAL_ERROR

CICS bridge で予期しないエラーが発生しました。

このフィードバック・コードは、MQDLH 構造の *Reason* フィールドのみに戻されます。

MQFB_CICS_NOT_AUTHORIZED

ユーザー ID が許可されないか、パスワードが無効です。

このフィードバック・コードは、MQDLH 構造の *Reason* フィールドのみに戻されます。

MQFB_CICS_UOW_BACKED_OUT

以下のいずれかの理由により、作業単位がバックアウトされました。

- 同じ作業単位内の他の要求の処理中に障害が検出された。
- 作業単位の処理中に CICS の異常終了が発生した。

MQFB_CICS_UOW_ERROR

作業単位の制御フィールド *UOWControl* が無効です。

トレースルートメッセージフィードバックコード

MQFB_ACTIVITY

MQFMT_EMBEDDED_PCF 形式と共に使用することにより、アクティビティ報告書を伴うユーザー・データのオプションが可能になります。

MQFB_MAX_ACTIVITIES

メッセージが関係するアクティビティの数が最大アクティビティ限界を超えたことによりトレース・ルート・メッセージが廃棄された場合に返されます。

MQFB_NOT_FORWARDED

トレース・ルート・メッセージをサポートしていないリモート・キュー・マネージャーにトレース・ルート・メッセージが送信されそうになったために廃棄された場合に返されます。

MQFB_NOT_DELIVERED

トレース・ルート・メッセージがローカル・キューに書き込まれそうになったために廃棄された場合に返されます。

MQFB_UNSUPPORTED_FORWARDING

転送パラメーターの値が認識されず、拒否ビット・マスクになっているためにトレース・ルート・メッセージが廃棄された場合に返されます。

MQFB_UNSUPPORTED_DELIVERY

送達パラメーターの値が認識されず、拒否ビット・マスクになっているためにトレース・ルート・メッセージが廃棄された場合に返されます。

IBM MQ 理由コード

例外レポートメッセージには、理由 *Feedback* コードが含ま IBM MQ れています。代表的な理由コードは次のとおりです。

MQRC_PUT_INHIBITED

(2051, X'803') このキューでは書き込み呼び出しが使用禁止になっています。

MQRC_Q_FULL

(2053, X'805') キューには既に最大数のメッセージが入っています。

MQRC_NOT_AUTHORIZED

(2035, X'7F3') アクセスは許可されません。

MQRC_Q_SPACE_NOT_AVAILABLE

(2056, X'808') ディスク上にキューのためのスペースがありません。

MQRC_PERSISTENT_NOT_ALLOWED

(2048, X'800') キューは永続的なメッセージをサポートしていません。

MQRC_MSG_TOO_BIG_FOR_Q_MGR

(2031, X'7EF') メッセージ長がキュー・マネージャーの最大許容長より大きいです。

MQRC_MSG_TOO_BIG_FOR_Q

(2030, X'7EE') メッセージの長さが、キューの最大許容数より大きいです。

理由コードの全リストについては、以下の資料を参照してください。

- 詳細については [IBM MQ for z/OS、API 完了と理由コード](#) を参照してください。
- その他のすべてのプラットフォームについては、[API の完了コードと理由コード](#) を参照してください。

これは、MQGET 呼び出しでは出力フィールド、MQPUT および MQPUT1 呼び出しでは入力フィールドです。このフィールドの初期値は MQFB_NONE です。

Encoding (MQLONG)

ここでは、メッセージ内の数値データの数値エンコードを指定します。これは、MQMD 構造体自体の数値データには適用されません。数値エンコード方式では、2 進整数、パック 10 進整数、および浮動小数点数の表記が定義されています。

z/OS z/OS では、2 進整数に使用されるエンコードによって文字セットの表現が決まることを対応する文字セット ID が示す場合は、メッセージ本文の文字データの整数エンコードを指定するために Encoding フィールドの 2 進整数部分も使用されます。この影響を受けるのは、特定のマルチバイト文字セット (例えば UTF-16 文字セット) だけです。

MQPUT または MQPUT1 呼び出しでは、アプリケーションは、このフィールドをデータに適切な値に設定する必要があります。キュー・マネージャーは、フィールドが有効かどうかをチェックしません。以下のような特殊値が定義されます。

MQENC_NATIVE

このコード・エンコードは、アプリケーションが実行されているプログラミング言語およびマシンのデフォルトになります。

注: この定数の値は、プログラム言語と環境によって異なります。このため、アプリケーションは、実行する環境に適したヘッダー・ファイル、マクロ・ファイル、COPY ファイル、および INCLUDE ファイルを使用してコンパイルする必要があります。

メッセージを書き込むアプリケーションは、通常、MQENC_NATIVE を指定します。メッセージを取り出すアプリケーションは、このフィールドを MQENC_NATIVE という値と比較する必要があります。値が異なる場合、アプリケーション側でメッセージの中の数値データを変換することが必要な場合があります。

MQGMO_CONVERT オプションを使用して、MQGET 呼び出しの処理の一部としてキュー・マネージャーがメッセージを変換するように要求します。Encoding フィールドの構成方法については、[896 ページの『マシン・エンコード』](#)を参照してください。

MQGET 呼び出しで MQGMO_CONVERT オプションを指定する場合、このフィールドは入出力フィールドです。アプリケーションで指定する値は、メッセージ・データを必要に応じて変換した後のエンコードです。変換が成功したか不要の場合、値は変化しません。変換が失敗したときは、MQGET 呼び出しの後の値は、アプリケーションに戻された未変換メッセージのエンコードを表しています。

それ以外の場合、MQGET 呼び出しでは出力フィールド、MQPUT および MQPUT1 呼び出しでは入力フィールドです。このフィールドの初期値は MQENC_NATIVE です。

CodedCharSetId (MQLONG)

このフィールドには、メッセージ本文に含まれる文字データの文字セット ID を指定します。

注: 呼び出しのパラメーターとして指定する MQMD 構造体およびその他の MQ データ構造体の文字データは、キュー・マネージャーの文字セットでなければなりません。これは、キュー・マネージャーの **CodedCharSetId** 属性によって定義されます。この属性の詳細については、[791 ページの『キュー・マネージャーの属性』](#)を参照してください。

オプションに MQGMO_CONVERT を指定した MQGET を呼び出すときにこのフィールドが MQCCSI_Q_MGR に設定されている場合の動作は、クライアント・アプリケーションとサーバー・アプリケーションで異なります。サーバー・アプリケーションでは、文字変換に使用されるコード・ページはキュー・マネージャーの CodedCharSetId です。クライアント・アプリケーションでは、文字変換に使用されるコード・ページは現在のロケールのコード・ページです。

クライアント・アプリケーションの場合、キュー・マネージャー上のロケールではなくクライアントのロケールに基づいて、MQCCSI_Q_MGR に記入されます。この規則の例外は、IMS ブリッジ・キューにメッセージを書き込む場合です。MQMD の CodedCharSetId フィールドには、キュー・マネージャーの CCSID が返されます。

次の特殊値を使用してはなりません。

MQCCSI_APPL

これを使用すると、MQMD の CodedCharSetId フィールドに間違った値が入るため、MQGMO_CONVERT オプションを指定した MQGET 呼び出しを使用してメッセージを受け取る際に、戻りコード MQRC_SOURCE_CCSD_ERROR (または、z/OS の場合は MQRC_FORMAT_ERROR) のエラーになります。

次の特殊値を使用することができます。

MQCCSI_Q_MGR

メッセージ内の文字データは、キュー・マネージャーの文字セットになります。

MQPUT 呼び出しおよび MQPUT1 呼び出しの場合、キュー・マネージャーは、メッセージと一緒に送信される MQMD にあるこの値を、キュー・マネージャーの正しい文字セット ID に変更します。結果として、値 MQCCSI_Q_MGR は MQGET 呼び出しにより戻されることはありません。

MQCCSI_DEFAULT

データの CodedCharSetId 型は、MQCFH 構造体の直前にあるヘッダー構造体の *String* フィールド CodedCharSetId によって定義される。または、MQCFH がメッセージの先頭にある場合、MQMD のフィールド CodedCharSetId によって定義される。

MQCCSI_INHERIT

メッセージ内の文字データは、この構造体と同じ文字セットです。つまり、キュー・マネージャーの文字セットです。(MQMD の場合のみ、MQCCSI_INHERIT は MQCCSI_Q_MGR と同じ意味になります。)

メッセージと共に送信される MQMD 内のこの値は、キュー・マネージャーにより、実際の MQMD 文字セット ID に変更されます。エラーが発生しない限り、値 MQCCSI_INHERIT が MQGET 呼び出しによって返されることはありません。

MQMD の PutApplType フィールドの値が MQAT_BROKER である場合は、MQCCSI_INHERIT を使用しないでください。

MQCCSI_EMBEDDED

メッセージ内の文字データの文字セットは、メッセージ・データそのものに ID が含まれている文字セットになります。メッセージ・データ内には、データの異なった部分に適用される、任意の数の文字セット ID を入れることができます。この値は、文字セットを混用したデータを含む PCF メッセージ (MQFMT_ADMIN, MQFMT_EVENT、または MQFMT_PCF 形式) の場合に使用する必要があります。PCF メッセージに含まれる MQCFST、MQCFSL、および MQCFSTF の各構造体には、MQCCSI_DEFAULT ではない、明示的な文字セット ID が指定されていなければなりません。

MQFMT_EMBEDDED_PCF 形式のメッセージに文字セットを混用したデータを入れる場合には、MQCCSI_EMBEDDED を使用しないでください。その代わりに、MQEPH 構造体の Flags フィールドに MQEPH_CCSDID_EMBEDDED を設定します。これは、先行する構造体に MQCCSI_EMBEDDED を設定するのと同様です。PCF メッセージに含まれる MQCFST、MQCFSL、および MQCFSTF の各構造体には、MQCCSI_DEFAULT ではない、明示的な文字セット ID が指定されていなければなりません。MQEPH 構造体の詳細については、[320 ページの『MQEPH - 組み込み PCF ヘッダー』](#)を参照してください。

この値は、MQPUT 呼び出しおよび MQPUT1 呼び出しに対してのみ指定してください。この値を MQGET 呼び出しに指定すると、メッセージが変換されなくなります。

MQPUT 呼び出しおよび MQPUT1 呼び出しの場合、キュー・マネージャーは、メッセージと一緒に送信される MQMD にある値 MQCCSI_Q_MGR および MQCCSI_INHERIT を前述のように変更しますが、MQPUT 呼び出しまたは MQPUT1 呼び出しに指定された MQMD は変更しません。指定された値について、それ以外の検査は行われません。

メッセージを取得するアプリケーションでは、このフィールドの値をアプリケーションが想定している値と比較して、値が異なっている場合には、メッセージ内の文字データをアプリケーションで変換する必要があります。

 MQMD の `Encoding encoding z/OS` フィールドが、メッセージ本体の文字データの整数エンコーディングを指定するために使用されるのは、MQMD の `encoding CodedCharSetId` フィールドが、文字セットの表現が二進整数に使用されるエンコーディングに依存することを示す場合である。

 において マルチプラットフォーム、文字データのバイト順序は、キューマネージャが実行されているプラットフォームのネイティブ整数エンコーディングと同じであると想定されます。この影響を受けるのは、特定のマルチバイト文字セット (例えば UTF-16 文字セット) だけです。

MQGET 呼び出しで MQGMO_CONVERT オプションを指定する場合、このフィールドは入出力フィールドです。アプリケーションから指定する値は、必要な場合にメッセージ・データの変換先となるコード化文字セット ID です。変換に成功した場合、または変換が不要の場合は、この値は変更されません (ただし、値 MQCCSI_Q_MGR または MQCCSI_INHERIT は実際の値に変換されます)。変換に失敗した場合、MQGET 呼

び出しの後の値は、アプリケーションに返された未変換のメッセージのコード化文字セット ID を表しています。

それ以外の場合、MQGET 呼び出しでは出力フィールド、MQPUT および MQPUT1 呼び出しでは入力フィールドです。このフィールドの初期値は MQCCSI_Q_MGR です。

Format (MQCHAR8)

これは、メッセージの送信側がメッセージの中のデータの性質を受信側に示すために使用する名前です。この名前には、キュー・マネージャーの文字セットにある文字であれば何でも指定することができますが、名前には次の制限があります。

- A から Z までの英大文字
- 0 から 9 までの数字

上記以外の文字が使用されている場合は、送信側および受信側のキュー・マネージャーの各文字セットの間で名前を変換できないこともあります。

名前に、フィールドの長さまでブランクを埋め込みます。あるいは、ヌル文字を使用して、フィールドの端に達する前に名前を終了させます。ヌル文字およびそれに続く文字は、ブランクと見なされます。名前の前または途中にはブランクを指定しないでください。MQGET 呼び出しの場合は、キュー・マネージャーは、フィールドの長さまでブランクを埋め込んだ名前を戻します。

キュー・マネージャーは、名前が上述の推奨事項に準じているかどうかは検査しません。

大文字、小文字、および大/小文字混合の MQ で始まる名前は、キュー・マネージャーで定義されたことを意味します。ユーザーは、自分用の形式では、これらの文字で始まる名前を使用しないでください。キュー・マネージャーの組み込み形式は次のとおりです。

MQFMT_NONE

データの性質は未定義です。キューからメッセージを取り出すときに、MQGMO_CONVERT オプションを使用してデータを変換できません。

MQGET 呼び出しで MQGMO_CONVERT を指定し、メッセージ中のデータの文字セットやエンコード方式が **MsgDesc** パラメーターによって指定されているものと異なる場合、次のような完了コードおよび理由コードと共にメッセージが返されます (その他のエラーは発生しないことを想定した場合です)。

- メッセージの先頭に MQFMT_NONE データがある場合は、完了コード MQCC_WARNING と理由コード MQRC_FORMAT_ERROR が返されます。
- MQFMT_NONE データがメッセージの末尾にある場合 (つまり、1 つまたは複数の MQ ヘッダー構造が先行している場合) は、完了コード MQCC_OK と理由コード MQRC_NONE が返されます。この場合、MQ ヘッダー構造体は、要求されている文字セットとエンコード方式に変換されます。

C 言語の場合、定数 MQFMT_NONE_ARRAY も定義されます。これは、MQFMT_NONE と同じ値ですが、ストリングではなく文字の配列です。

MQFMT_ADMIN

メッセージは、プログラマブル・コマンド・フォーマット (PCF) のコマンド・サーバー要求または応答メッセージです。MQGMO_CONVERT オプションを MQGET 呼び出しに指定している場合は、この形式のメッセージを変換することができます。プログラマブル・コマンド・フォーマット・メッセージの使用の詳細については、[プログラマブル・コマンド・フォーマットの使用](#)を参照してください。

C 言語の場合、定数 MQFMT_ADMIN_ARRAY も定義されます。これは、MQFMT_ADMIN と同じ値ですが、ストリングではなく文字の配列です。

MQFMT_CICS

メッセージ・データは、CICS 情報ヘッダー MQCIH で始まり、アプリケーション・データが後に続きます。アプリケーション・データの形式名は、MQCIH 構造の Format フィールドで指定されています。

 z/OS では、MQGET 呼び出しで MQGMO_CONVERT オプションを指定して、MQFMT_CICS 形式のメッセージを変換します。

C 言語の場合、定数 MQFMT_CICS_ARRAY も定義されます。これは、MQFMT_CICS と同じ値ですが、ストリングではなく文字の配列です。

MQFMT_COMMAND_1

このメッセージは、オブジェクト・カウント、完了コード、および理由コードが入っている MQSC コマンド・サーバー応答メッセージです。MQGMO_CONVERT オプションを MQGET 呼び出しに指定している場合は、この形式のメッセージを変換することができます。

C 言語の場合、定数 MQFMT_COMMAND_1_ARRAY も定義されます。これは、MQFMT_COMMAND_1 と同じ値ですが、ストリングではなく文字の配列です。

MQFMT_COMMAND_2

メッセージは、要求されたオブジェクトの情報を含む MQSC コマンド・サーバー応答メッセージです。MQGMO_CONVERT オプションを MQGET 呼び出しに指定している場合は、この形式のメッセージを変換することができます。

C 言語の場合、定数 MQFMT_COMMAND_2_ARRAY も定義されます。これは、MQFMT_COMMAND_2 と同じ値ですが、ストリングではなく文字の配列です。

MQFMT_DEAD_LETTER_HEADER

メッセージ・データは送達不能ヘッダー MQDLH で始まります。元のメッセージからのデータは、MQDLH 構造体のすぐ後に続きます。元のメッセージ・データの形式名は、MQDLH 構造体の *Format* フィールドで指定されています。この構造体の詳細については、306 ページの『MQDLH - 送達不能ヘッダー』を参照してください。MQGMO_CONVERT オプションを MQGET 呼び出しに指定している場合は、この形式のメッセージを変換することができます。

MQFMT_DEAD_LETTER_HEADER の *Format* を持つメッセージについては、COA および COD レポートは生成されません。

C 言語の場合、定数 MQFMT_DEAD_LETTER_HEADER_ARRAY も定義されます。これは、MQFMT_DEAD_LETTER_HEADER と同じ値ですが、ストリングではなく文字の配列です。

MQFMT_DIST_HEADER

メッセージ・データは、配布リスト・ヘッダー MQDH で始まります。このデータの中には、MQOR レコードおよび MQPMR レコードの配列などがあります。配布リスト・ヘッダーには追加データが続く場合があります。補足データ (存在する場合) の形式は、MQDH 構造の *Format* フィールドで指定されています。この構造の詳細については、「299 ページの『MQDH - 配布ヘッダー』」を参照してください。MQGMO_CONVERT オプションが MQGET 呼び出しで指定されている場合に、形式が MQFMT_DIST_HEADER のメッセージを変換できます。

この形式は、次の環境でサポートされます。

-  AIX
-  IBM i
-  Linux
-  Windows

および、これらのシステムに接続された IBM MQ MQI clients。

C 言語の場合、定数 MQFMT_DIST_HEADER_ARRAY も定義されます。これは、MQFMT_DIST_HEADER と同じ値ですが、ストリングではなく文字の配列です。

MQFMT_EMBEDDED_PCF

PCF コマンド値が MQCMD_TRACE_ROUTE に設定された場合のトレース・ルート・メッセージの形式。この形式を使用すると、先行する PCF パラメーターをアプリケーションが処理できれば、ユーザー・データをトレース・ルート・メッセージと共に送信することができます。

PCF ヘッダーは、最初のヘッダーである必要があります。そうでない場合、メッセージはトレース・ルート・メッセージとして扱われません。つまり、メッセージをグループ化できず、トレース・ルート・メッセージをセグメント化できないことを意味します。トレース・ルート・メッセージがグループとして送信されると、メッセージは理由コード MQRC_MSG_NOT_ALLOWED_IN_GROUP で拒否されます。

MQFMT_ADMIN もトレース・ルート・メッセージの形式として使用できますが、この場合にはトレース・ルート・メッセージと共にユーザー・データを送信することができないことに注意してください。

MQFMT_EVENT

メッセージは、発生したイベントを報告する MQ イベント・メッセージです。イベント・メッセージは、プログラマブル・コマンドと同じ構造を持っています。この構造についての詳細は、[PCF コマンド・メッセージ](#)を参照してください。また、イベントについては、[イベント・モニター](#)を参照してください。

MQGMO_CONVERT オプションを MQGET 呼び出しに指定している場合は、バージョン 1 のイベント・メッセージはすべての環境で変換することができます。バージョン 2 のイベント・メッセージは、z/OS でのみ変換できます。

C 言語の場合、定数 MQFMT_EVENT_ARRAY も定義されます。これは、MQFMT_EVENT と同じ値ですが、ストリングではなく文字の配列です。

MQFMT_IMS

メッセージ・データは IMS 情報ヘッダー MQIIH で始まり、その後にアプリケーション・データが続きます。アプリケーション・データの形式名は、MQIIH 構造の Format フィールドで指定されています。

MQGMO_CONVERT を指定して MQGET を使用する場合に、MQIIH 構造がどのように扱われるのかについて詳しくは、373 ページの『Format (MQCHAR8)』および 374 ページの『ReplyToFormat (MQCHAR8)』を参照してください。

C 言語の場合、定数 MQFMT_IMS_ARRAY も定義されます。これは、MQFMT_IMS と同じ値ですが、ストリングではなく文字の配列です。

MQFMT_IMS_VAR_STRING

メッセージは IMS 可変長ストリングで、形式は 11zzccc です。各部分の詳細は次のとおりです。

11

IMS 可変長ストリング項目の合計長を指定する長さ 2 バイトのフィールドです。合計長は、「11 (2 バイト) + zz (2 バイト) + 文字ストリングの長さ」となります。11 は、Encoding フィールドに指定されたエンコードで表された 2 バイトの 2 進整数を示します。

zz

IMS にとって有効なフラグを含む 2 バイト・フィールドです。zz は 2 つの MQBYTE フィールドから成るバイト・ストリングを示し、送信しても送信側と受信側の間でその内容が変わることはありません (つまり、zz は変換の影響を受けません)。

ccc

可変長文字ストリングを示します。長さは 11-4 文字です。ccc は、CodedCharSetId フィールドで指定された文字セットで表されます。

z/OS では、一連の IMS 変数ストリング (各ストリングの形式は 11zzccc) を結合してメッセージ・データを構成することができます。連続する IMS 変数ストリング間で、スキップされるバイトがあってはなりません。これは、最初のストリングの長さが奇数である場合、2 番目のストリングの位置合わせがうまくいかないということです。つまり、2 の倍数になっている境界では開始しません。基本データ・タイプの位置合わせを必要とするマシンで、このようなストリングを組み立てる際には注意してください。

MQGET 呼び出しで MQGMO_CONVERT オプションを使用して、MQFMT_IMS_VAR_STRING 形式のメッセージを変換します。

C 言語の場合、定数 MQFMT_IMS_VAR_STRING_ARRAY も定義されます。これは、MQFMT_IMS_VAR_STRING と同じ値ですが、ストリングではなく文字の配列です。

MQFMT_MD_EXTENSION

メッセージ・データは、拡張メッセージ記述子 MQMDE で始まります。このデータの後に他のデータ (通常はアプリケーション・メッセージ・データ) が続くこともあります。MQMDE に続くデータの形式名、文字セット、およびエンコード方式は、MQMDE の Format、CodedCharSetId、および Encoding フィールドで指定されています。この構造体の詳細については、439 ページの『MQMDE - 拡張メッセージ記述子』を参照してください。MQGMO_CONVERT オプションを MQGET 呼び出しに指定している場合は、この形式のメッセージを変換することができます。

C 言語の場合、定数 MQFMT_MD_EXTENSION_ARRAY も定義されます。これは、MQFMT_MD_EXTENSION と同じ値ですが、ストリングではなく文字の配列です。

MQFMT_PCF

このメッセージは、プログラマブル・コマンド・フォーマット (PCF) メッセージの構造体に適合するユーザー定義のメッセージです。MQGMO_CONVERT オプションを MQGET 呼び出しに指定している場合は、この形式のメッセージを変換することができます。プログラマブル・コマンド・フォーマット・メッセージの使用の詳細については、[プログラマブル・コマンド・フォーマットの使用](#)を参照してください。

C 言語の場合、定数 MQFMT_PCF_ARRAY も定義されます。これは、MQFMT_PCF と同じ値ですが、ストリングではなく文字の配列です。

MQFMT_REF_MSG_HEADER

メッセージ・データは、参照メッセージ・ヘッダー MQRMH で始まります。このデータの後に他のデータが続くこともあります。データの形式名、文字セット、およびエンコード方式は、MQRMH の Format、CodedCharSetId、および Encoding フィールドで指定されています。この構造体の詳細については、525 ページの『MQRMH - 参照メッセージ・ヘッダー』を参照してください。

MQGMO_CONVERT オプションを MQGET 呼び出しに指定している場合は、この形式のメッセージを変換することができます。

この形式は、次の環境でサポートされます。

-  AIX
-  IBM i
-  Linux
-  Windows

および、これらのシステムに接続された IBM MQ MQI clients。

C 言語の場合、定数 MQFMT_REF_MSG_HEADER_ARRAY も定義されます。これは、MQFMT_REF_MSG_HEADER と同じ値ですが、ストリングではなく文字の配列です。

MQFMT_RF_HEADER

メッセージ・データは、規則およびフォーマット・ヘッダー MQRFH で始まります。このデータの後に他のデータが続くこともあります。データの形式名、文字セット、およびエンコード (存在する場合) は、MQRFH の Format フィールド、CodedCharSetId フィールド、および Encoding フィールドでそれぞれ指定されています。MQGMO_CONVERT オプションを MQGET 呼び出しに指定している場合は、この形式のメッセージを変換することができます。

C 言語の場合、定数 MQFMT_RF_HEADER_ARRAY も定義されます。これは、MQFMT_RF_HEADER と同じ値ですが、ストリングではなく文字の配列です。

MQFMT_RF_HEADER_2

メッセージ・データは、バージョン 2 の規則およびフォーマット・ヘッダー MQRFH2 で始まります。このデータの後に他のデータが続くこともあります。オプションのデータがある場合は、その形式名、文字セット、およびエンコード方式は、MQRFH2 の Format、CodedCharSetId、および Encoding フィールドで指定されています。MQGMO_CONVERT オプションを MQGET 呼び出しに指定している場合は、この形式のメッセージを変換することができます。

C 言語の場合、定数 MQFMT_RF_HEADER_2_ARRAY も定義されます。これは、MQFMT_RF_HEADER_2 と同じ値ですが、ストリングではなく文字の配列です。

MQFMT_STRING

アプリケーション・メッセージ・データは SBCS ストリング (1 バイト文字セット)、または DBCS ストリング (2 バイト文字セット) のいずれかにすることができます。MQGMO_CONVERT オプションを MQGET 呼び出しに指定している場合は、この形式のメッセージを変換することができます。

C 言語の場合、定数 MQFMT_STRING_ARRAY も定義されます。これは、MQFMT_STRING と同じ値ですが、ストリングではなく文字の配列です。

MQFMT_TRIGGER

このメッセージは、MQTM 構造によって記述されるトリガー・メッセージです。この構造の詳細については、587 ページの『MQTM - トリガー・メッセージ』を参照してください。MQGMO_CONVERT オプションを MQGET 呼び出しに指定している場合は、この形式のメッセージを変換することができます。

C 言語の場合、定数 MQFMT_TRIGGER_ARRAY も定義されます。これは、MQFMT_TRIGGER と同じ値ですが、ストリングではなく文字の配列です。

MQFMT_WORK_INFO_HEADER

メッセージ・データは、作業情報ヘッダー MQWIH で始まり、その後にアプリケーション・データが続きます。アプリケーション・データの形式名は、MQWIH 構造の Format フィールドで指定されています。

 z/OS では、MQGET 呼び出しで MQGMO_CONVERT オプションを指定して、MQFMT_WORK_INFO_HEADER 形式のメッセージのユーザー・データを変換します。ただし、MQWIH 構造自体は、常にキュー・マネージャーの文字セットおよびエンコードで戻されます (つまり、MQGMO_CONVERT オプションが指定されるかどうかに関係なく、MQWIH 構造は変換されます)。

C 言語の場合、定数 MQFMT_WORK_INFO_HEADER_ARRAY も定義されます。これは、MQFMT_WORK_INFO_HEADER と同じ値ですが、ストリングではなく文字の配列です。

MQFMT_XMIT_Q_HEADER

メッセージ・データは伝送キュー・ヘッダー MQXQH で始まります。元のメッセージからのデータは、MQXQH 構造体のすぐ後に続きます。元のメッセージ・データの形式名は、MQMD 構造の Format フィールドで指定されており、伝送キュー・ヘッダー MQXQH の一部です。この構造体の詳細については、609 ページの『MQXQH - 伝送キュー・ヘッダー』を参照してください。

MQFMT_XMIT_Q_HEADER の Format を持つメッセージについては、COA および COD レポートは生成されません。

C 言語の場合、定数 MQFMT_XMIT_Q_HEADER_ARRAY も定義されます。これは、MQFMT_XMIT_Q_HEADER と同じ値ですが、ストリングではなく文字の配列です。

これは、MQGET 呼び出しでは出力フィールド、MQPUT および MQPUT1 呼び出しでは入力フィールドです。このフィールドの長さは MQ_FORMAT_LENGTH によって指定されます。このフィールドの初期値は MQFMT_NONE です。

Priority (MQLONG)

MQPUT および MQPUT1 呼び出しでは、値はゼロ以上でなければなりません。ゼロは、最低の優先順位です。以下のような特殊値も使用することができます。

MQPRI_PRIORITY_AS_Q_DEF

- キューがクラスタキューの場合、メッセージの優先度は、メッセージが配置されるキューの特定のインスタンスを所有する宛先キューマネージャで定義された属性 **DefPriority** から取得されます。

クラスター・キューの複数インスタンスがあり、それらがこの属性で異なる場合、いずれかの値が選出されます。どの値が使用されるかは予測できません。したがって、この属性はすべてのインスタンスで同じ値に設定する必要があります。そうしない場合、エラー・メッセージ AMQ9407 がキュー・マネージャー・ログに発行されます。 [How are destination object attributes resolved for aliases, remote and cluster queues?](#) も参照してください。

DefPriority の値は、メッセージが宛先キューに置かれるときに **Priority** フィールドにコピーされます。その後、**DefPriority** が変更されても、既にキューに置かれているメッセージは影響を受けません。

- キューがクラスター・キューではない場合、メッセージの優先順位は、ローカル のキュー・マネージャで定義された **DefPriority** 属性から取られます。これは、宛先キュー・マネージャがリモートの場合も同じです。

キュー名の解決パスに複数の定義がある場合、デフォルトの優先順位は、パスの最初の定義にあるこの属性の値から取られます。次のタイプがあります。

- 別名キュー
- ローカル・キュー
- リモート・キューのローカル定義
- キュー・マネージャーの別名
- 伝送キュー (例えば、*DefXmitQName* キューなど)

DefPriority の値は、メッセージが書き込まれるときに *Priority* フィールドにコピーされます。その後、*DefPriority* が変更されても、既書き込まれているメッセージは影響を受けません。

MQGET 呼び出しの戻り値は、常にゼロ以上です。値 *MQPRI_PRIORITY_AS_Q_DEF* は、返されません。

メッセージが、ローカル・キュー・マネージャーでサポートされている最大の優先順位 (この最大値は **MaxPriority** キュー・マネージャー属性で指定される) より高い優先順位で書き込まれた場合、メッセージはキュー・マネージャーで受け入れられますが、キュー・マネージャーの最大優先順位でキューに入れます。MQPUT または MQPUT1 呼び出しは、MQCC_WARNING および理由コード MQRC_PRIORITY_EXCEEDS_MAXIMUM と共に完了します。ただし、その *Priority* フィールドはメッセージを配置したアプリケーションが指定した値を保持します。

z/OS では、MsgSeqNumber が 1 のメッセージが、メッセージ・デリバリー・シーケンス MQMDS_PRIORITY および索引タイプ MQIT_GROUP_ID のキューに入れられた場合、キューは、そのメッセージを別の優先順位で処理します。メッセージが優先順位 0 または 1 のキューに入れられた場合は、優先順位 2 が指定されているかのように処理されます。これは、効率的なグループの完全性テストを可能にするために、このタイプのキューに置かれたメッセージの順序が最適化されるためです。メッセージ・デリバリー・シーケンス MQMDS_PRIORITY および索引タイプ MQIT_GROUP_ID に関する詳細については、[MsgDeliverySequence 属性](#)を参照してください。

メッセージに応答する際、アプリケーションは、応答メッセージに対して要求メッセージの優先順位を使用する必要があります。他の状況では、MQPRI_PRIORITY_AS_Q_DEF を指定すると、アプリケーションを変更することなく優先順位の調整を行うことができます。

これは、MQGET 呼び出しでは出力フィールド、MQPUT および MQPUT1 呼び出しでは入力フィールドです。このフィールドの初期値は MQPRI_PRIORITY_AS_Q_DEF です。

Persistence (MQLONG)

このフィールドは、システム障害が発生してキュー・マネージャーを再始動してもメッセージが残るかどうを示します。MQPUT および MQPUT1 呼び出しでは、値は次のいずれかでなければなりません。

MQPER_PERSISTENT

メッセージは、システム障害およびキュー・マネージャーの再始動後も存続します。メッセージが書き込まれ、メッセージが書き込まれた作業単位がコミットされると (メッセージが作業単位の一部として書き込まれる場合)、メッセージは補助ストレージに保存されます。これは、メッセージがキューから除去され、それが取得された作業単位がコミットされる (メッセージが作業単位の一部として取得された場合) まで存続します。

持続メッセージがリモート・キューに送信されると、蓄積交換機構が、宛先への経路に沿って各キュー・マネージャーにメッセージを保持します。これは、次のキュー・マネージャーにメッセージが到達したことが分かるまで続きます。

持続メッセージを以下に書き込むことはできません。

- 一時動的キュー
- CFLEVEL(2) 以下で CFSTRUCT オブジェクトにマップする共有キュー、または CFSTRUCT オブジェクトが RECOVER(NO) として定義されている共有キュー。

持続メッセージは、永続動的キュー、および事前定義のキューに書き込むことができます。

MQPER_NOT_PERSISTENT

通常、メッセージは、システムの障害およびキュー・マネージャーの再始動の後には存続しません。これは、キュー・マネージャーの再始動にメッセージの完全なコピーが補助ストレージで見つかった場合でも適用されます。

NPMCLASS (HIGH) キューの場合、非持続メッセージは、キュー・マネージャーの通常のシャットダウンおよび再始動の後も存続します。

注: ネイティブ HA 構成では、キューマネージャインスタンスのリーダー変更は、設定にかかわらず、**NPMCLASS** 常に非永続メッセージの破棄を伴う。

共有キューの場合、非持続メッセージは、キュー・マネージャー再開後もキュー共有グループ内で存続しますが、共有キューにメッセージを保管するためのカップリング・ファシリティに障害が発生すると、存続しません。

MQPER_PERSISTENCE_AS_Q_DEF

- キューがクラスター・キューの場合、メッセージの持続性は、メッセージが入れられるキューの特定のインスタンスを所有する宛先キュー・マネージャーで定義された **DefPersistence** 属性から取られます。

クラスター・キューの複数インスタンスがあり、それらがこの属性で異なる場合、いずれかの値が選出されます。どの値が使用されるかは予測できません。したがって、この属性はすべてのインスタンスで同じ値に設定する必要があります。そうしない場合、エラー・メッセージ AMQ9407 がキュー・マネージャー・ログに発行されます。 [How are destination object attributes resolved for aliases, remote and cluster queues?](#) も参照してください。

DefPersistence の値は、メッセージが宛先キューに置かれるときに **Persistence** フィールドにコピーされます。その後、**DefPersistence** が変更されても、既にキューに置かれているメッセージは影響を受けません。

- キューがクラスター・キューでない場合、メッセージの持続性は、宛先キュー・マネージャーがリモートであっても、ローカル・キュー・マネージャーで定義された **DefPersistence** 属性から取られます。

キュー名の解決パスに複数の定義がある場合、デフォルトの持続性は、パスの最初の定義にあるこの属性の値から取られます。次のタイプがあります。

- 別名キュー
- ローカル・キュー
- リモート・キューのローカル定義
- キュー・マネージャーの別名
- 伝送キュー (例えば、**DefXmitQName** キューなど)

DefPersistence の値は、メッセージが書き込まれるときに **Persistence** フィールドにコピーされます。その後、**DefPersistence** が変更されても、既に書き込まれているメッセージは影響を受けません。

持続メッセージと非持続メッセージが同一キューにあっても構いません。

メッセージに応答する際、アプリケーションは、応答メッセージに対して要求メッセージの持続性を使用する必要があります。

MQGET 呼び出しの場合、戻り値は MQPER_PERSISTENT または MQPER_NOT_PERSISTENT のどちらかです。

これは、MQGET 呼び出しでは出力フィールド、MQPUT および MQPUT1 呼び出しでは入力フィールドです。このフィールドの初期値は MQPER_PERSISTENCE_AS_Q_DEF です。

MsgId (MQBYTE24)

あるメッセージを他のメッセージと区別するために使用されるバイト・ストリングです。2つのメッセージが同じメッセージ ID を持つことは、キュー・マネージャーによって禁止されませんが、通常は避けてください。メッセージ ID は、メッセージの永続的なプロパティであり、キュー・マネージャーを再始動しても存続します。メッセージ ID は、文字ストリングではなくバイト・ストリングなので、あるキュー・マネージャーから別のキュー・マネージャーへメッセージが流れても、文字セット間の変換は行われません。

MQPUT および MQPUT1 呼び出しにおいて、アプリケーションが MQMI_NONE または MQPMO_NEW_MSG_ID を指定した場合、キューマネージャはメッセージが置かれる際に一意のメッセージ識別子(*MsgId*)を生成し、メッセージと共に送信されるメッセージ記述子に配置する。また、キュー・マネージャは、送信側のアプリケーションに属するメッセージ記述子の中にこのメッセージ ID を返します。アプリケーションは、この値を使用して、特定のメッセージに関する情報を記録し、アプリケーションの他の部分からの照会に応答することができます。

注: キューマネージャによって生成 *MsgId* される A は、4 バイトの製品識別子(ASCII または EBCDICAMQ- のいずれかで、CSQ- または、ここで - は空白文字を表す) で構成され、その後に製品固有の実装による一意の文字列が続く。IBM MQ の場合は、これには、キュー・マネージャ名の最初の 12 文字と、システム・クロックからとられた値が入っています。したがって、メッセージ ID を固有値にするためには、相互通信可能なすべてのキュー・マネージャの名前の最初の 12 文字が異なっている必要があります。また、固有のストリングを生成する機能は、システム・クロックが逆方向に変更されないことを前提としています。キュー・マネージャが生成するメッセージ ID とアプリケーションが生成するメッセージ ID が重複する可能性をなくすために、アプリケーションは、ID を生成するに当たって、その最初の文字が ASCII または EBCDIC の A から I (X'41' から X'49' および X'C1' から X'C9') にならないようにする必要があります。ただし、アプリケーションでこれらの範囲の先頭文字をもつ ID を生成しないように回避措置がとられることはありません。

メッセージをトピックに書き込む場合、キュー・マネージャは、パブリッシュされるメッセージごとに必要に応じて固有のメッセージ ID を生成します。アプリケーションで MQPMO_NEW_MSG_ID を指定すると、キュー・マネージャは出力で戻す固有メッセージ ID を生成します。アプリケーションによって MQMI_NONE が指定されている場合、MQMD 内の *MsgId* フィールドの値は、呼び出しからの戻り時に変更されません。

保存パブリケーションについて詳しくは、[478 ページの『Options \(MQLONG\)』](#)の MQPMO_RETAIN の説明を参照してください。

メッセージを配布リストに書き込む場合は、キュー・マネージャが必要に応じて固有のメッセージ ID を生成しますが、MQMI_NONE または MQPMO_NEW_MSG_ID を指定したとしても、呼び出しからの戻り時に MQMD の *MsgId* フィールドの値が変更されることはありません。キュー・マネージャが生成するメッセージ ID をアプリケーションが認識する必要がある場合は、アプリケーションは *MsgId* フィールドのある MQPMR レコードを提供する必要があります。

送信側のアプリケーションは、メッセージ ID に対して、MQMI_NONE 以外の値を指定することもできます。こうすると、キュー・マネージャは、固有のメッセージ ID の生成を停止します。メッセージを転送するアプリケーションでは、これを使用して元のメッセージのメッセージ ID を伝搬することができます。

キュー・マネージャは、以下の目的以外に、このフィールドを使用することはありません。

- 要求があれば、上述のように固有値を生成する。
- メッセージの取得要求を発行したアプリケーションに値を配布する。
- このメッセージについて生成されたレポート・メッセージがあれば、(*Report* オプションに応じて) そのレポート・メッセージの *CorrelId* フィールドに値をコピーする。

キュー・マネージャまたはメッセージ・チャネル・エージェントでは、レポート・メッセージを生成したときに、元のメッセージの *Report* フィールドで指定されている方法、つまり MQRO_NEW_MSG_ID と MQRO_PASS_MSG_ID のいずれかに従って、*MsgId* フィールドを設定します。レポート・メッセージを生成するアプリケーションも、これと同じことを行う必要があります。

MQGET 呼び出しでは、*MsgId* は、キューから特定のメッセージを取り出すために使用できる 5 つのフィールドの 1 つです。通常、MQGET 呼び出しでは、キューにある次のメッセージを戻しますが、5 つの選択基準の 1 つ以上を任意の組み合わせで指定することによって、特定のメッセージを取得できます。これらのフィールドは以下のとおりです。

- *MsgId*
- *CorrelId*
- *GroupId*
- *MsgSeqNumber*

- *Offset*

アプリケーションでは、これらのフィールドの1つ以上を必要な値に設定した上で、MQGMO の *MatchOptions* フィールドに対応する MQMO_* 一致オプションを設定して、これらのフィールドを選択基準として使用します。これらのフィールドに値が指定されたメッセージだけが取り出しの対象になります。(アプリケーション側で変更しない場合は) *MatchOptions* フィールドのデフォルトは、メッセージ ID と相関 ID の両方を突き合わせます。

▶ **z/OS** z/OS では、使用可能な選択基準は、キューに使用される索引のタイプによって制限されます。詳細は、**IndexType** キュー属性を参照してください。

通常は、選択基準を満たす最初のメッセージがキューから戻されます。ただし、MQGMO_BROWSE_NEXT を指定した場合は、選択基準を満たす次のメッセージが戻されます。このメッセージの走査は、現行カーソル位置の後のメッセージから開始されます。

注：選択基準を満たすメッセージがキューから順次スキャンされるので、選択基準を指定しなかった場合よりも、取り出し時間は遅くなります。特に、条件を満たすメッセージを見付けるまでに多数のメッセージをスキャンする必要がある場合は、この傾向が強くなります。以下はこの例外です。

- ▶ **Multi** *CorrelId* 索引により真の順次スキャンの実行が不要になる 64 ビット Multiplatforms 上の *CorrelId* による MQGET 呼び出し。
- ▶ **z/OS** MQGET 呼び出しを によって *IndexType* 実行 z/OS。

どちらの場合も、取り出しパフォーマンスは改善されます。

グループ内のメッセージおよび論理メッセージのセグメントに関する MQGET オプションの表を参照し、様々な状況における選択基準の使用方法について詳細を確認してください。

メッセージ ID として MQMI_NONE を指定すると、MQMO_MATCH_MSG_ID を指定しなかった場合と同じ結果になります。つまり、すべてのメッセージ ID が一致することになります。

MQGMO_MSG_UNDER_CURSOR オプションが、MQGET 呼び出しの **GetMsgOpts** パラメーターで指定されている場合、このフィールドは無視されます。

MQGET 呼び出しから戻ったとき、*MsgId* フィールドは、戻されたメッセージ (それがあある場合) のメッセージ ID に設定されます。

以下のような特別な値を使用することができます。

MQMI_NONE

メッセージ ID が指定されていません。

値は、フィールドの長さについては 2 進ゼロです。

C 言語の場合、定数 MQMI_NONE_ARRAY も定義されます。これは、MQMI_NONE と同じ値ですが、ストリングではなく文字の配列です。

これは、MQGET、MQPUT、および MQPUT1 呼び出しの入出力フィールドです。このフィールドの長さは MQ_MSG_ID_LENGTH によって指定されます。このフィールドの初期値は MQMI_NONE です。

CorrelId (MQBYTE24)

CorrelId フィールドは、メッセージ・ヘッダー内のプロパティーで、特定のメッセージまたは特定のメッセージ・グループを識別するために使用できます。

これは、バイト・ストリングで、1つのメッセージを別のメッセージと関連付けたり、メッセージをアプリケーションが実行している他の作業と関連付けたりするために、アプリケーションによって使用できます。相関 ID はメッセージの永続的なプロパティーであり、キュー・マネージャーを再始動しても保持されます。相関 ID は文字ストリングではなくバイト・ストリングであるため、あるキュー・マネージャーから別のキュー・マネージャーにメッセージが転送され、文字セットが異なっても、相関 ID は変換されません。

MQPUT 呼び出しおよび MQPUT1 呼び出しの場合、アプリケーションは任意の値を指定できます。キュー・マネージャーはこの値をメッセージと一緒に送信し、メッセージの取得要求を出したアプリケーションに配信します。

アプリケーションで MQPMO_NEW_CORREL_ID を指定すると、キュー・マネージャーによって固有の相関 ID が生成されてメッセージと一緒に送信され、さらに MQPUT 呼び出しまたは MQPUT1 呼び出しの出力として送信側アプリケーションにその ID が返されます。

キュー・マネージャーによって生成される相関 ID は、3 バイトの製品 ID (ASCII または EBCDIC の AMQ または CSQ) の後に、予約済みの 1 バイトと、製品固有の実装の固有ストリングを続けた値です。IBM MQ では、この製品固有の実装ストリングには、キュー・マネージャー名の先頭 12 文字と、システム・クロックから導出された値が含まれます。したがって、メッセージ ID を固有にするには、相互に通信可能なすべてのキュー・マネージャーについて名前の先頭 12 文字が異なっている必要があります。また、固有のストリングを生成する機能は、システム・クロックが逆方向に変更されないことを前提としています。キュー・マネージャーが生成するメッセージ ID とアプリケーションが生成するメッセージ ID が重複する可能性をなくすために、アプリケーションは、ID を生成するに当たって、その最初の文字が ASCII または EBCDIC の A から I (X'41' から X'49' および X'C1' から X'C9') にならないようにする必要があります。ただし、アプリケーションでこれらの範囲の先頭文字をもつ ID を生成しないように回避措置がとられることはありません。

生成された相関 ID は、メッセージが保存される場合にはそのメッセージとともに保持され、MQSUB 呼び出しで渡される MQSD の SubCorrelId フィールドに MQCI_NONE を指定するサブスクライバーに対してパブリケーションとしてメッセージを送信するときの相関 ID として使用されます。保存パブリケーションの詳細については、[MQPMO オプション](#)を参照してください。

キュー・マネージャーまたはメッセージ・チャネル・エージェントがレポート・メッセージを生成するとき、CorrelId フィールドは、元のメッセージの Report フィールドに指定された値 (MQRO_COPY_MSG_ID_TO_CORREL_ID または MQRO_PASS_CORREL_ID) に応じて設定されます。レポート・メッセージを生成するアプリケーションも、これと同じことを行う必要があります。

MQGET 呼び出しの場合、CorrelId は、キューから取得する特定のメッセージを選択するために使用できる 5 つのフィールドの 1 つです。このフィールドに値を指定する方法の詳細については、[MsgId](#) フィールドの説明を参照してください。

相関 ID として MQCI_NONE を指定することは、MQMO_MATCH_CORREL_ID を指定しないことと同じ結果になり、任意の相関 ID が合致します。

MQGET 呼び出しの **GetMsgOpts** パラメーターに MQGMO_MSG_UNDER_CURSOR オプションを指定した場合、このフィールドは無視されます。

MQGET 呼び出しから戻った時点で、CorrelId フィールドには、返されたメッセージ (それがあある場合) の相関 ID が設定されます。

以下のような特別な値を使用することができます。

MQCI_NONE

相関 ID は指定されません。

値は、フィールドの長さについては 2 進ゼロです。

C 言語の場合、定数 MQCI_NONE_ARRAY も定義されます。これは、MQCI_NONE と同じ値ですが、ストリングではなく文字の配列です。

MQCI_NEW_SESSION

メッセージは、新しいセッションの先頭です。

この値は、新規セッションの開始、つまり、メッセージの新規シーケンスの始まりとして CICS bridge により認識されます。

C 言語の場合、定数 MQCI_NEW_SESSION_ARRAY も定義されます。これは MQCI_NEW_SESSION と同じ値ですが、ストリングではなく文字の配列です。

MQGET 呼び出しの場合、これは入出力フィールドです。MQPUT 呼び出しおよび MQPUT1 呼び出しの場合、MQPMO_NEW_CORREL_ID が指定されていないときは入力フィールド、MQPMO_NEW_CORREL_ID が

指定されているときは出力フィールドです。このフィールドの長さは MQ_CORREL_ID_LENGTH によって指定されます。このフィールドの初期値は MQCI_NONE です。

注:

階層内のパブリケーションの相関 ID を受け渡すことはできません。そのフィールドはキュー・マネージャーによって使用されます。

BackoutCount (MQLONG)

これは、メッセージが、作業単位の一部として MQGET 呼び出しから事前に戻されて、その後バックアウトされた回数のカウントです。これは、アプリケーションがメッセージ内容に基づいて処理エラーを検出する上で助けになります。カウントには、MQGMO_BROWSE_* オプションのいずれかを指定する MQGET 呼び出しは含まれません。

カウントの正確度は、**HardenGetBackout** キュー属性の影響を受けます。[830 ページの『キューの属性』](#)を参照してください。

 z/OS では、値 255 は、メッセージが 255 回以上バックアウトされたことを意味します。255 より大きい値が戻されることはありません。

これは、MQGET 呼び出しの出力フィールドです。MQPUT および MQPUT1 呼び出しでは、無視されます。このフィールドの初期値は 0 です。

ReplyToQ (MQCHAR48)

これは、メッセージの読み取り要求を発行したアプリケーションが、MQMT_REPLY および MQMT_REPORT メッセージを送信する宛先のメッセージ・キューの名前です。*ReplyToQMgr* によって識別されたキュー・マネージャーで定義されているキューの、ローカル名です。このキューをモデル・キューにはなりません。ただし、送信側のキュー・マネージャーは、メッセージが書き込まれたときにこれを確認しません。

MQPUT および MQPUT1 呼び出しで、*MsgType* フィールドの値が MQMT_REQUEST の場合、またはレポートが *Report* フィールドによって要求されている場合、このフィールドがブランクであってはなりません。ただし、メッセージ・タイプにかかわらず、指定された値 (または置き換えられた値) が、メッセージの読み取り要求を発行したアプリケーションに渡されます。

ReplyToQMgr フィールドがブランクの場合、ローカル・キュー・マネージャーは、独自のキュー定義内で *ReplyToQ* 名を探索します。この名前を持つリモート・キューのローカル定義が存在している場合は、伝送されたメッセージの中の *ReplyToQ* 値が、リモート・キューの定義内の **RemoteQName** 属性の値によって置換されます。受信側のアプリケーションがメッセージの MQGET 呼び出しを出すと、メッセージ記述子にこの値が返されます。リモート・キューのローカル定義が存在しない場合は、*ReplyToQ* は変わりません。

名前を指定する場合は、末尾にブランクを含めることができます。最初のヌル文字およびその後続く文字は、ブランクとして扱われます。それ以外の場合、名前がキューの命名規則を満たしているかどうかの検査は行われません。伝送されるメッセージの中で *ReplyToQ* が置き換えられる場合は、伝送される名前についても同様です。必要に応じて名前が指定されたかどうかのみ検査されます。

応答先キューが要求されていない場合は、*ReplyToQ* フィールドにブランクまたはヌル・ストリング (C プログラミング言語) を設定するか、1 個以上のブランクとそれに続くヌル文字を設定します。このフィールドを初期化していない状態のまま残さないでください。

MQGET 呼び出しの場合、キュー・マネージャーは、常にフィールドの長さまでブランクを埋め込んだ名前を返します。

レポートメッセージを必要とするメッセージが配信できず、かつレポートメッセージも指定されたキューに配信できない場合、元のメッセージとレポートメッセージの両方がデッドレター (未配信メッセージ) キューに送られます (で説明 [791 ページの『キュー・マネージャーの属性』](#) されている属性

DeadLetterQName 参照)。

これは、MQGET 呼び出しでは出力フィールド、MQPUT および MQPUT1 呼び出しでは入力フィールドです。このフィールドの長さは MQ_Q_NAME_LENGTH によって指定されます。このフィールドの初期値は、C 言語ではヌル・ストリングであり、他のプログラミング言語では 48 桁の空白文字です。

ReplyToQMGr (MQCHAR48)

これは、応答メッセージまたはレポート・メッセージを送信する宛先のキュー・マネージャーの名前です。*ReplyToQ* は、このキュー・マネージャーで定義されるキューのローカル名です。

ReplyToQMGr フィールドが空白の場合、ローカル・キュー・マネージャーは、独自のキュー定義内で *ReplyToQ* 名を探索します。この名前を持つリモート・キューのローカル定義が存在している場合は、伝送されたメッセージの中の *ReplyToQMGr* 値が、リモート・キューの定義内の **RemoteQMGrName** 属性の値によって置換されます。受信側のアプリケーションがメッセージの MQGET 呼び出しを出すと、メッセージ記述子にこの値が返されます。リモート・キューのローカル定義が存在しない場合は、メッセージと共に伝送される *ReplyToQMGr* がローカル・キュー・マネージャーの名前になります。

名前を指定する場合は、末尾に空白を含めることができます。最初のヌル文字およびその後続く文字は、空白として扱われます。それ以外の場合、名前がキュー・マネージャーの命名規則を満たしているかどうか、またはこの名前が送信側のキュー・マネージャーに知られているかどうかの検査は行われません。伝送されるメッセージの中で *ReplyToQMGr* が置き換えられる場合は、伝送される名前についても同様です。

応答先キューが要求されていない場合は、*ReplyToQMGr* フィールドに空白またはヌル・ストリング (C プログラミング言語) を設定するか、1 個以上の空白とそれに続くヌル文字を設定します。このフィールドを初期化していない状態のまま残さないでください。

MQGET 呼び出しの場合、キュー・マネージャーは、常にフィールドの長さまで空白を埋め込んだ名前を返します。

これは、MQGET 呼び出しでは出力フィールド、MQPUT および MQPUT1 呼び出しでは入力フィールドです。このフィールドの長さは MQ_Q_MGR_NAME_LENGTH で指定します。このフィールドの初期値は、C 言語ではヌル・ストリングであり、他のプログラミング言語では 48 桁の空白文字です。

UserIdentifier (MQCHAR12)

これは、メッセージの ID コンテキストの一部です。メッセージ・コンテキストの詳細については、[385 ページの『MQMD - メッセージ記述子』](#)および[メッセージのコンテキスト](#)を参照してください。

UserIdentifier には、メッセージを発信したアプリケーションのユーザー ID を指定します。キュー・マネージャーはこの情報を文字データとして扱いますが、そのフォーマットの定義はしません。ユーザー識別子が環境からどのように決定されるかについての詳細は、「[認証と認可: オペレーティングシステム \(アプリケーションプロセス\) のユーザー ID](#)」を参照してください。

 キューマネージャが z/OS、`MQ\_GET\_USER\_AUTHORITY` または `MQPUT1` `MQ\_GET\_USER\_AUTHORITY\_MOM` `MQPUT` 呼び出しに対してこの情報を生成する際、`MQ\_GET\_USER\_AUTHORITY` または `MQ\_GET\_USER\_AUTHORITY\_MOM` `MQOPEN` `MQPUT1` 呼び出しの `USER\_AUTHORITY\_PARAMETER` **ObjDesc** パラメータ **AlternateUserId** から `USER\_AUTHORITY` を使用します。ただし、`MQOO\_ALTERNATE\_USER\_AUTHORITY` または `MQPMO\_ALTERNATE\_USER\_AUTHORITY` オプションが指定されている場合に限りです。それらのオプションを指定しなかった場合、キュー・マネージャーは、環境から判断したユーザー ID を使用します。

 通常 IBM MQ for Multiplatforms、フィールド **UserIdentifier** はキューマネージャによって生成される出力フィールドであり、メッセージを送信する接続に関連付けられたユーザー ID で設定されます。このユーザー ID は、使用されている認証メカニズムとプラットフォームによって決定されます。ただし、または **MQPUT1** `MQPUT` の呼び出しでは、このフィールドを入力/出力フィールドにすることができます。

- このフィールドは、**PutMsgOpts** パラメータで **MQPMO_SET_IDENTITY_CONTEXT** または **MQPMO_SET_ALL_CONTEXT** を指定した場合に入力/出力フィールドとなります。フィールド内でヌル

文字より後の情報はすべて破棄されます。キュー・マネージャーは、ヌル文字とそれ以降の文字をブランクに変換します。

- ユーザー ID フィールドをキューマネージャに生成させる代わりに指定したい場合は、MQPMO_SET_IDENTITY_CONTEXT または MQPMO_SET_ALL_CONTEXT を指定し、その後、フィールド **UserIdentifier** にユーザー ID を指定してください。
- MQPMO_SET_IDENTITY_CONTEXT または MQPMO_SET_ALL_CONTEXT を指定しない場合、この **UserIdentifier** フィールドは入力時に無視され、出力専用フィールドとなります。

IDPWOS 認証において、12 文字を超える ID が **UserIdentifier** フィールドに格納される方法は、プラットフォーム IBM MQ およびバージョンによって異なります：

-  のために IBM MQ 9.4.0 Long Term Support、そして IBM MQ 9.4.1 および 9.4.2 Continuous Delivery、のみ Windows クライアントまたはサーバー アプリケーションは、12 文字を超える接続ユーザー ID を採用できます。この場合、ユーザー ID は最初の 12 文字を保存することで示されます。
-  から IBM MQ 9.4.3 すべてのクライアントまたはサーバー アプリケーション Continuous Delivery プラットフォームは、12 文字を超える接続ユーザー ID を採用できます。の上 Windows、最初の 12 文字を保存することでユーザー ID が示されます。の上 AIX、Linux そして IBM MQ Appliance、また設定する必要があります `AllowLongUID=Yes` キューマネージャーのプロパティ。このプロパティが設定されている場合、ユーザー ID は先頭 11 文字に+記号を付加して表示されます。

コール **MQPUT1** またはコールの正常な完了後、このフィールドにはメッセージがキューに格納された場合に、メッセージと共に送信されたユーザー ID が含まれます。これはメッセージが保持される場合にメッセージと共に保持される値 **UserIdentifier** ですが、メッセージが購読者へのパブリケーションとして送信される際には使用されません。購読者は自身宛ての **UserIdentifier** 全パブリケーションにおいて **UserIdentifier** この値を上書きする値を提供するためです。メッセージがコンテキストを持っていない場合、フィールドは完全にブランクになります。保持されたパブリケーションの詳細については、MQPMO オプションの MQPMO_RETAIN を参照してください。

この **MQGET** 呼び出しにおいて、フィールド **UserIdentifier** は出力フィールドです。このフィールドの長さによっては与えられる **MQ_USER_ID_LENGTH**。このフィールドの初期値は、C 言語ではヌル・ストリングですが、その他のプログラミング言語では 12 個のブランク文字です。

メッセージ受信後、後続の `open MQOPEN` または `MQPUT1 ObjDesc open` 呼び出しの `open` パラメータの **AlternateUserID** フィールドで **UserIdentifier** を使用し、アプリケーションではなく `open` を実行 **UserIdentifier** するユーザーに対する認証チェックを実行する。以下に例を示します。

1. ユーザー A からメッセージが送信されました。
2. メッセージはアプリケーション B によってキューから取り出される。
3. アプリケーション B は、ユーザー A の権限のもとでさらなるメッセージング活動を実行したい。

これを実現するため、アプリケーション B は受信メッセージの MQMD からフィールド **UserIdentifier** をコピーし、新規に開くハンドルの MQOD のフィールド **AlternateUserID** に書き込む。アプリケーション B も、これを許可する特定の権限を持っている必要があることに注意してください。

AccountingToken (MQBYTE32)

これは、会計トークンで、メッセージの ID コンテキストの一部です。メッセージ・コンテキストの詳細については、385 ページの『MQMD - メッセージ記述子』およびメッセージのコンテキストを参照してください。

AccountingToken を使用することで、アプリケーションはメッセージの結果として実行された作業に適切な課金を行うことができます。キュー・マネージャーはこの情報をビット・ストリングとして処理し、その内容は検査しません。

キュー・マネージャーは、この情報を次のように生成します。

- このフィールドの最初のバイトを、フィールドの後続バイトにある会計情報の長さに設定します。長さは 0 から 30 までの範囲であり、2 進整数として最初のバイトに保管されます。
- 2 番目以降のバイト (長さフィールドに指定) を、環境に応じた会計情報に設定します。

–  **z/OS** z/OS では、会計情報は以下に設定されます。

- z/OS バッチでは、JES JOB カードからの会計情報、または EXEC カード内の JES ACCT ステートメントからの会計情報 (コンマの区切り文字は 'X'FF' に変更されます)。この情報は、必要に応じて 31 バイトに切り捨てられます。
- TSO の場合は、ユーザーのアカウント番号。
- CICS の場合は、LU 6.2 作業単位 ID (UEPUOWDS) (26 バイト)。
- IMS の場合は、16 文字の IMS リカバリー・トークンと連結した 8 文字の PSB 名。

–  **IBM i** IBM i では、会計情報は、ジョブの会計コードに設定されます。

–  **Linux**  **AIX** AIX and Linux では、会計情報は ASCII 文字による数値のユーザー ID に設定されます。

–  **Windows** Windows では、会計情報は圧縮形式の Windows セキュリティー ID (SID) に設定されます。SID は、*UserIdentifier* フィールドに格納されたユーザー ID を一意に識別します。*AccountingToken* フィールドに SID が格納される場合、(SID の 3 番目以降のバイトにある) 6 バイトの ID 権限は省略されます。例えば、Windows SID の長さが 28 バイトの場合、*AccountingToken* フィールドには SID 情報のうちの 22 バイトが格納されます。

- 会計フィールドの最後のバイト (バイト 32) は、会計トークン・タイプに設定されます (この場合、MQACTT_NT_SECURITY_ID、x'0b')。

MQACTT_CICS_LUOW_ID

CICS LUOW ID。

 **Windows** **MQACTT_NT_SECURITY_ID**

Windows セキュリティー ID。

 **IBM i** **MQACTT_OS400_ACCOUNT_TOKEN**

IBM i 会計トークン。

 **UNIX** **MQACTT_UNIX_NUMERIC_ID**

UNIX の数値 ID。

MQACTT_USER

ユーザー定義の会計トークン。

MQACTT_UNKNOWN

不明な会計トークン・タイプ。

会計トークン・タイプは以下の環境でのみ、明示的な値に設定されます。

–  **AIX** AIX

–  **IBM i** IBM i

–  **Linux** Linux

–  **Windows** Windows

および、これらのシステムに接続された IBM MQ MQI clients。これ以外の環境では、会計トークン・タイプは値 MQACTT_UNKNOWN に設定されます。これらの環境では、*PutApplType* フィールドを使用して、受け取った会計トークンのタイプを推測します。

- その他のすべてのバイトを 2 進ゼロに設定します。

MQPUT 呼び出しおよび MQPUT1 呼び出しの場合、**PutMsgOpts** パラメーターに

MQPMO_SET_IDENTITY_CONTEXT または MQPMO_SET_ALL_CONTEXT が指定されているときには、これは入出力フィールドです。MQPMO_SET_IDENTITY_CONTEXT または MQPMO_SET_ALL_CONTEXT のどち

らも指定されていないと、入力は無視され、出力専用フィールドになります。メッセージのコンテキストの詳細については、[メッセージのコンテキスト](#)を参照してください。

MQPUT または MQPUT1 呼び出しが正常に完了すると、メッセージがキューに書き込まれた場合、このフィールドには、そのメッセージと共に送信された `AccountingToken` が入ります。これは、保存される場合にメッセージと共に保持される `AccountingToken` の値になります (保存パブリケーションについて詳しくは、[478 ページの『Options \(MQLONG\)』](#)の `MQPMO_RETAIN` の説明を参照してください)。しかし、メッセージがサブスクライバーにパブリケーションとして送信される場合には `AccountingToken` として使用されません。送信されたすべてのパブリケーションの `AccountingToken` をオーバーライドするために値を提供するためです。メッセージにコンテキストがない場合、フィールドは完全に 2 進ゼロになります。

これは、MQGET 呼び出しの出力フィールドです。

このフィールドは、キュー・マネージャーの文字セットに基づいた変換の対象ではありません。フィールドは、文字のストリングではなく、ビットのストリングとして扱われます。

キュー・マネージャーは、フィールドにある情報については、何も処理しません。アプリケーションは、会計の目的で情報を使用する場合に、情報を解釈する必要があります。

`AccountingToken` フィールドには、次の特殊値を使用することができます。

MQACT_NONE

アカウンティング・トークンが指定されていません。

値は、フィールドの長さについては 2 進ゼロです。

C 言語の場合、定数 `MQACT_NONE_ARRAY` も定義されます。これは、`MQACT_NONE` と同じ値ですが、ストリングではなく文字の配列です。

このフィールドの長さは `MQ_ACCOUNTING_TOKEN_LENGTH` によって指定されます。このフィールドの初期値は `MQACT_NONE` です。

ApplIdentityData (MQCHAR32)

これは、メッセージの **ID コンテキスト** の一部です。メッセージ・コンテキストの詳細については、[385 ページの『MQMD - メッセージ記述子』](#)および[メッセージのコンテキスト](#)を参照してください。

ApplIdentityData は、アプリケーション・スイートによって定義される情報で、メッセージやその発信元に関する追加の情報を提供するために使用することができます。キュー・マネージャーはこの情報を文字データとして扱いますが、そのフォーマットの定義はしません。キュー・マネージャーは、この情報を生成するときに、全体をブランクにします。

MQPUT 呼び出しおよび MQPUT1 呼び出しの場合、**PutMsgOpts** パラメーターに

`MQPMO_SET_IDENTITY_CONTEXT` または `MQPMO_SET_ALL_CONTEXT` が指定されているときには、これは入出力フィールドです。ヌル文字がある場合は、ヌル文字およびその後続く文字は、キュー・マネージャーによってブランクに変換されます。`MQPMO_SET_IDENTITY_CONTEXT` または `MQPMO_SET_ALL_CONTEXT` のどちらも指定されていないと、入力は無視され、出力専用フィールドになります。メッセージのコンテキストの詳細については、[メッセージのコンテキスト](#)を参照してください。

MQPUT または MQPUT1 呼び出しが正常に完了すると、メッセージがキューに書き込まれた場合、このフィールドには、そのメッセージと共に送信された *ApplIdentityData* が入ります。これは、保存される場合にメッセージと共に保持される *ApplIdentityData* の値になります (保存パブリケーションについて詳しくは、`MQPMO_RETAIN` の説明を参照してください)。しかし、メッセージがサブスクライバーにパブリケーションとして送信される場合には *ApplIdentityData* として使用されません。送信されたすべてのパブリケーションの *ApplIdentityData* をオーバーライドするために値を提供するためです。メッセージがコンテキストを持っていない場合、フィールドは完全にブランクになります。

これは、MQGET 呼び出しの出力フィールドです。このフィールドの長さは `MQ_APPL_IDENTITY_DATA_LENGTH` によって指定されます。このフィールドの初期値は、C 言語ではヌル・ストリングですが、その他のプログラミング言語では 32 桁のブランク文字です。

PutApplType (MQLONG)

これは、メッセージを書き込むアプリケーションのタイプで、メッセージの**起点コンテキスト**の一部です。メッセージ・コンテキストの詳細については、[385 ページの『MQMD - メッセージ記述子』およびメッセージのコンテキスト](#)を参照してください。

PutApplType には、以下のいずれかの標準タイプを指定できます。独自のタイプを定義することもできます。その場合、使用できるのは MQAT_USER_FIRST から MQAT_USER_LAST の範囲の値だけです。

MQAT_AIX

AIX アプリケーション (MQAT_UNIX と同じ値)。

MQAT_AMQP

AMQP プロトコル・アプリケーション

MQAT_BROKER

ブローカー。

MQAT_CICS

CICS トランザクション。

MQAT_CICS_BRIDGE

CICS bridge。

MQAT_CICS_VSE

CICS/VSE トランザクション。

MQAT_DOS

PC DOS 上の IBM MQ MQI client アプリケーション。

MQAT_DQM

分散キュー・マネージャー・エージェント。

MQAT_GUARDIAN

Tandem Guardian アプリケーション (MQAT_NSK と同じ値)。

MQAT_IMS

IMS アプリケーション。

MQAT_IMS_BRIDGE

IMSブリッジ。

MQAT_JAVA

Java。

MQAT_MVS

MVS または TSO アプリケーション (MQAT_ZOS と同じ値)。

MQAT_NOTES_AGENT

Lotus Notes エージェント・アプリケーション。

MQAT_OS390

OS/390® アプリケーション (MQAT_ZOS と同じ値)。

MQAT_OS400

IBM i アプリケーション。

MQAT_QMGR

キュー・マネージャー。

MQAT_UNIX

UNIX アプリケーション。

MQAT_VOS

Stratus VOS アプリケーション。

MQAT_WINDOWS

16 ビットの Windows アプリケーション。

MQAT_WINDOWS_NT

32 ビットの Windows アプリケーション。

MQAT_WLM

z/OS ワークロード・マネージャー・アプリケーション。

MQAT_XCF

XCF。

MQAT_ZOS

z/OS アプリケーション。

MQAT_DEFAULT

デフォルトのアプリケーション・タイプ。

この値は、アプリケーションが実行中のプラットフォームの、デフォルト・アプリケーション・タイプです。

注：この定数の値は環境によります。そのため、アプリケーションを実行する予定のプラットフォームに適切なヘッダー・ファイル、組み込みファイル、または COPY ファイルを使用してアプリケーションをコンパイルしてください。

MQAT_UNKNOWN

この値は、他のコンテキスト情報は存在しているのに、アプリケーション・タイプが不明であることを示すために使用します。

MQAT_USER_FIRST

ユーザー定義のアプリケーション・タイプの最低値。

MQAT_USER_LAST

ユーザー定義のアプリケーション・タイプの最高値。

以下のような特殊値が出されることもあります。

MQAT_NO_CONTEXT

この値は、メッセージがコンテキストなしで書き込まれたときに、キュー・マネージャーによって設定されます (すなわち、MQPMO_NO_CONTEXT コンテキスト・オプションが指定されます)。

メッセージを取り出す際、メッセージにコンテキストがあるかどうかを判別するために、この値について *PutApplType* をテストすることができます (他のコンテキスト・フィールドのいずれかがブランクでないときは、アプリケーションで MQPMO_SET_ALL_CONTEXT を使用して *PutApplType* を MQAT_NO_CONTEXT に設定しないように推奨されています)。

アプリケーションの書き込みの結果として、キュー・マネージャーがこの情報を生成するとき、フィールドは、環境により求められる値に設定されます。IBM i では、MQAT_OS400 に設定されます。キュー・マネージャーが IBM i で MQAT_CICS を使用することはありません。

MQPUT および MQPUT1 呼び出しでは、**PutMsgOpts** パラメーターに MQPMO_SET_ALL_CONTEXT が指定されていれば、これは入出力フィールドです。MQPMO_SET_ALL_CONTEXT が指定されていない場合、このフィールドは入力で無視され、出力専用フィールドになります。

これは、MQGET 呼び出しの出力フィールドです。このフィールドの初期値は MQAT_NO_CONTEXT です。

PutApplName (MQCHAR28)

これは、メッセージを書き込むアプリケーションの名前で、メッセージの起点コンテキストの一部です。内容はプラットフォーム間で異なり、リリース間で異なることもあります。

メッセージ・コンテキストの詳細については、[385 ページの『MQMD - メッセージ記述子』](#)および[メッセージのコンテキスト](#)を参照してください。

アプリケーション名を他のプログラミング言語で指定できます。詳細については、[サポートされているプログラミング言語でのアプリケーション名の指定](#)を参照してください。

PutApplName の形式は、*PutApplType* の値によって決まり、リリース間で変わることがあります。変更はまれですが、環境が変わると発生します。

キュー・マネージャーがこのフィールドを設定する場合 (つまり MQPMO_SET_ALL_CONTEXT 以外のすべてのオプションの場合)、環境で決定されている値に設定します。

- ▶ **z/OS** z/OS では、キュー・マネージャーは次のものを使用します。
 - z/OS バッチの場合は、JES JOB カードからの 8 文字のジョブ名
 - TSO の場合は、7 文字の TSO ユーザー ID
 - CICS の場合は、8 文字のアプリケーション ID とそれに続く 4 文字のトランザクション ID
 - IMS の場合は、8 文字の IMS システム ID とそれに続く 8 文字の PSB 名
 - XCF の場合は、8 文字の XCF グループ名とそれに続く 16 文字の XCF メンバー名
 - キュー・マネージャーが生成したメッセージの場合は、キュー・マネージャー名の最初の 28 文字
 - CICS のない分散キューイングの場合は、チャンネル・イニシエーターの 8 文字のジョブ名とそれに続く送達不能キューに書き込まれた 8 文字のモジュール名、およびそれに続く 8 文字のタスク ID。

それぞれの名前について、フィールドの残り部分にスペースがあれば右側にブランクが埋め込まれます。複数の名前がある場合、名前と名前の間に区切り文字はありません。
- ▶ **Windows** Windows システムでは、キュー・マネージャーは次の名前を使用します。
 - CICS アプリケーションの場合は、CICS トランザクション名
 - CICS アプリケーション以外の場合は、実行可能なジョブの完全修飾名の右端から 28 文字
- ▶ **IBM i** IBM i では、キュー・マネージャーは完全修飾ジョブ名を使用します。
- ▶ **Linux** ▶ **AIX** AIX and Linux では、キュー・マネージャーは次の名前を使用します。
 - CICS アプリケーションの場合は、CICS トランザクション名
 - CICS 以外のアプリケーションでは、MQ はプロセス名をオペレーティング・システムに問い合わせます。これはプログラム・ファイル名 (絶対パスなし) として戻されます。その後、MQ はこのプロセス名を MQMD.PutApplName フィールドに次のように書き込みます。

▶ **AIX** **AIX**

名前が 28 バイト以下である場合、その名前が挿入されて右側にスペースが埋め込まれます。
 名前が 28 バイトより大きい場合、名前の左端から 28 バイトが挿入されます。

▶ **Linux** **Linux**

名前が 15 バイト以下である場合、その名前が挿入されて右側にスペースが埋め込まれます。
 名前が 15 バイトより大きい場合、名前の左端から 15 バイトが挿入され、右側にスペースが埋め込まれます。

例えば /opt/mqm/samp/bin/amqspout QNAME QMNAME を実行すると、PutApplName は 'amqspout' となります。この MQCHAR28 フィールドには 21 個のスペース文字が埋め込まれています。PutApplName には /opt/mqm/samp/bin を含む絶対パスが含まれていないことに注意してください。

MQPUT および MQPUT1 呼び出しでは、**PutMsgOpts** パラメーターに MQPMO_SET_ALL_CONTEXT が指定されていれば、これは入出力フィールドです。フィールド内でヌル文字より後の情報はすべて破棄されます。ヌル文字およびその後に続く文字は、キュー・マネージャーによってブランクに変換されます。MQPMO_SET_ALL_CONTEXT が指定されていない場合、このフィールドは入力で無視され、出力専用フィールドになります。

PutDate (MQCHAR8)

これは、メッセージが書き込まれたときの日付で、メッセージの**起点コンテキスト**の一部です。メッセージ・コンテキストの詳細については、[385 ページの『MQMD - メッセージ記述子』](#)および**メッセージのコンテキスト**を参照してください。

キュー・マネージャーがこのフィールドを生成する際に使用する日付の形式は、以下のとおりです。

- YYYYMMDD

文字は、以下のものを表します。

YYYY

年 (4 桁の数字)

MM

月 (01 から 12 まで)

DD

日 (01 から 31 まで)

PutDate および *PutTime* フィールドには、グリニッジ標準時 (GMT) が使用されます。GMT に正確に合わせたシステム・クロックに従います。

メッセージが作業単位の一部として書き込まれた場合は、作業単位がコミットされた日付ではなく、メッセージが書き込まれた日付です。

MQPUT および MQPUT1 呼び出しでは、**PutMsgOpts** パラメーターに MQPMO_SET_ALL_CONTEXT が指定されていれば、これは入出力フィールドです。フィールドの内容は、キュー・マネージャーによって検査されませんが、フィールド内のヌル文字のあとにある情報はいずれも廃棄されます。キュー・マネージャーは、ヌル文字とそれ以降の文字をブランクに変換します。MQPMO_SET_ALL_CONTEXT が指定されていない場合、このフィールドは入力で無視され、出力専用フィールドになります。

これは、MQGET 呼び出しの出力フィールドです。このフィールドの長さは MQ_PUT_DATE_LENGTH によって指定されます。このフィールドの初期値は、C 言語ではヌル・ストリングですが、その他のプログラミング言語では 8 個のブランク文字です。

PutTime (MQCHAR8)

これは、メッセージが書き込まれたときの時刻で、メッセージの**起点コンテキスト**の一部です。メッセージ・コンテキストの詳細については、[385 ページの『MQMD - メッセージ記述子』](#)および[メッセージのコンテキスト](#)を参照してください。

キュー・マネージャーがこのフィールドを生成する際に使用する時刻の形式は、以下のとおりです。

- HHMMSSSTH

文字は、順に以下のものを表します。

HH

時間 (00 から 23 まで)

MM

分 (00 から 59 まで)

SS

秒 (00 から 59 まで。下記の注を参照)

T

10 分の 1 秒 (0 から 9 まで)

H

100 分の 1 秒 (0 から 9 まで)

注: システム・クロックが非常に正確な標準時間に同期している場合は、ごくまれですが、*PutTime* の秒数として 60 または 61 が返されることがあります。これは、グローバル時間標準にうるう秒が挿入されたときに発生します。

PutDate および *PutTime* フィールドには、グリニッジ標準時 (GMT) が使用されます。GMT に正確に合わせたシステム・クロックに従います。

メッセージが作業単位の一部として書き込まれた場合は、作業単位がコミットされた時刻ではなく、メッセージが書き込まれた時刻です。

MQPUT および MQPUT1 呼び出しでは、**PutMsgOpts** パラメーターに MQPMO_SET_ALL_CONTEXT が指定されていれば、これは入出力フィールドです。キュー・マネージャーはフィールドの内容を検査しませんが、フィールド内のヌル文字のあとにある情報はいずれも廃棄されます。キュー・マネージャーは、ヌル文字とそれ以降の文字をブランクに変換します。MQPMO_SET_ALL_CONTEXT が指定されていない場合、このフィールドは入力で無視され、出力専用フィールドになります。

これは、MQGET 呼び出しの出力フィールドです。このフィールドの長さは MQ_PUT_TIME_LENGTH によって指定されます。このフィールドの初期値は、C 言語ではヌル・ストリングですが、その他のプログラミング言語では 8 個のブランク文字です。

ApplOriginData (MQCHAR4)

これは、メッセージの起点コンテキストの一部です。メッセージ・コンテキストの詳細については、[385 ページの『MQMD - メッセージ記述子』](#)および[メッセージのコンテキスト](#)を参照してください。

ApplOriginData は、アプリケーション・スイートにより定義される情報で、メッセージの発信元についての追加情報を提供するのに使用できます。例えば、ID データが信頼できるかどうかを示すために、適切なユーザー権限で実行されているアプリケーションにより設定が可能です。

キュー・マネージャーはこの情報を文字データとして扱いますが、そのフォーマットの定義はしません。キュー・マネージャーは、この情報を生成するときに、全体をブランクにします。

MQPUT および MQPUT1 呼び出しでは、**PutMsgOpts** パラメーターに MQPMO_SET_ALL_CONTEXT が指定されていれば、これは入出力フィールドです。フィールド内でヌル文字より後の情報はすべて破棄されます。キュー・マネージャーは、ヌル文字とそれ以降の文字をブランクに変換します。MQPMO_SET_ALL_CONTEXT が指定されていない場合、このフィールドは入力で無視され、出力専用フィールドになります。

これは、MQGET 呼び出しの出力フィールドです。このフィールドの長さは MQ_APPL_ORIGIN_DATA_LENGTH によって指定されます。このフィールドの初期値は、C 言語ではヌル・ストリングですが、その他のプログラミング言語では 4 つのブランク文字です。

このメッセージがパブリッシュされると、ApplOriginData は設定されていますが、受信するサブスクリプションでこれはブランクです。

GroupId (MQBYTE24)

物理メッセージが属する特定のメッセージ・グループまたは論理メッセージを識別するために使用されるバイト・ストリングです。GroupId フィールドは、メッセージのセグメント化が許可されている場合にも使用されます。いずれの場合も、GroupId フィールドには非ヌル値が設定され、MsgFlags フィールドには以下に示すフラグのうち 1 つ以上が設定されます。

- MQMF_MSG_IN_GROUP
- MQMF_LAST_MSG_IN_GROUP
- MQMF_SEGMENT
- MQMF_LAST_SEGMENT
- MQMF_SEGMENTATION_ALLOWED

上記のフラグが設定されなかった場合、GroupId の値は特殊なヌル値である MQGI_NONE となります。

以下の場合には、アプリケーションが MQPUT または MQGET 呼び出しでこのフィールドを設定する必要はありません。

- MQPUT 呼び出しで、MQPMO_LOGICAL_ORDER が指定されている場合。
- MQGET 呼び出しで、MQMO_MATCH_GROUP_ID が指定されていない場合。

以下、これらの呼び出しをレポート・メッセージ以外のメッセージに使用する場合の推奨方法について説明します。ただし、アプリケーションがさらに制御を要求する場合、または呼び出しが MQPUT1 の場合、アプリケーションは、GroupId に適切な値が設定されていることを確認する必要があります。

メッセージ・グループおよびメッセージ・セグメントは、グループ ID が重複していない場合にのみ正しく処理できます。そのため、アプリケーションごとに固有のグループ ID を生成しないでください。アプリケーションでは次のいずれかの処理を行ってください。

- MQPMO_LOGICAL_ORDER が指定されている場合、キュー・マネージャーは、グループに含まれている最初のメッセージまたは論理メッセージのセグメントである最初のメッセージに対して固有のグループ

ID を自動的に生成し、残りのメッセージにそのグループ ID を使用します。そのため、アプリケーションが特別なアクションを取る必要はありません。これが、推奨されている手順です。

- MQPMO_LOGICAL_ORDER が指定されていない場合、アプリケーションは、キュー・マネージャーにグループ ID を生成するように要求する必要があります。そのためには、グループに含まれているメッセージまたは論理メッセージのセグメントに対して発行する最初の MQPUT 呼び出しまたは MQPUT1 呼び出しで、*GroupId* を MQGI_NONE に設定します。次に、その呼び出しの出力時にキュー・マネージャーから返されるグループ ID を、グループ内の残りのメッセージまたは論理メッセージのセグメントに使用する必要があります。メッセージ・グループの中にセグメント分割されたメッセージがある場合は、そのグループのすべてのセグメントおよびメッセージに、同じグループ ID を使用する必要があります。

MQPMO_LOGICAL_ORDER が指定されていない場合、グループに含まれているメッセージおよび論理メッセージのセグメントは任意の順序 (逆順など) で書き込むことができますが、グループ ID は、これらのメッセージのいずれかに対して発行される最初の MQPUT または MQPUT1 呼び出しにより割り当てられる必要があります。

MQPUT および MQPUT1 呼び出しの入力時にキュー・マネージャーが使用する値については、[キューでの物理順序](#)で説明されています。MQPUT 呼び出しおよび MQPUT1 呼び出しの出力時に、オープンされたオブジェクトが単一キューであり、配布リストではない場合、キュー・マネージャーはこのフィールドに、メッセージと共に送信された値を設定します。オープンされたオブジェクトが配布リストである場合、このフィールドの値は変わりません。後者の場合、生成されたグループ ID をアプリケーションが認識する必要がある場合、アプリケーションは、*GroupId* フィールドのある MQPMR レコードを提供する必要があります。

MQGET 呼び出しの入力時にキュー・マネージャーが使用する値については、[355 ページの表 510](#) で説明します。MQGET 呼び出しの出力時に、キュー・マネージャーは、このフィールドに、取り出されたメッセージの値を設定します。

以下のような特殊値が定義されます。

MQGI_NONE

グループ ID は指定されません。

値は、フィールドの長さについては 2 進ゼロです。この値は、グループに含まれていないメッセージ (論理メッセージのセグメントではない) で、かつセグメント化が許可されていないメッセージに使用されます。

C 言語の場合、定数 MQGI_NONE_ARRAY も定義されます。これは、MQGI_NONE と同じ値ですが、ストリングではなく文字の配列です。

フィールドの長さは、MQ_GROUP_ID_LENGTH で指定します。このフィールドの初期値は MQGI_NONE です。Version が MQMD_VERSION_2 より小さい場合、このフィールドは無視されます。

MsgSeqNumber (MQLONG)

グループ内の論理メッセージの順序番号です。

順序番号は 1 から始まり、グループに新しい論理メッセージが追加されるたびに 1 つずつ大きくなり、最大で 999 999 999 です。グループ内にない物理メッセージは、シーケンス番号 1 を持ちます。

以下の場合、アプリケーションが MQPUT または MQGET 呼び出しでこのフィールドを設定する必要はありません。

- MQPUT 呼び出しで、MQPMO_LOGICAL_ORDER が指定されている場合。
- MQGET 呼び出しで、MQMO_MATCH_MSG_SEQ_NUMBER が指定されていない場合。

以下、これらの呼び出しをレポート・メッセージ以外のメッセージに使用する場合の推奨方法について説明します。ただし、アプリケーションがさらに制御を要求する場合、または呼び出しが MQPUT1 の場合、アプリケーションは、*MsgSeqNumber* に適切な値が設定されていることを確認する必要があります。

MQPUT および MQPUT1 呼び出しの入力時にキュー・マネージャーが使用する値については、[キューでの物理順序](#)で説明されています。MQPUT および MQPUT1 呼び出しの出力時に、キュー・マネージャーは、このフィールドにメッセージと共に送信された値を設定します。

MQGET 呼び出しの入力時にキュー・マネージャーが使用する値については、[355 ページの表 510](#) で説明します。MQGET 呼び出しの出力時に、キュー・マネージャーは、このフィールドに、取り出されたメッセージの値を設定します。

このフィールドの初期値は 1 です。Version が MQMD_VERSION_2 より小さい場合、このフィールドは無視されます。

Offset (MQLONG)

物理メッセージのデータの (そのデータを一部として含んでいる) 論理メッセージの先頭からのオフセットをバイト単位で指定します。このようなデータをセグメントといいます。オフセットは、0 から 999 999 999 までの範囲です。論理メッセージのセグメントでない物理メッセージのオフセット値は、0 です。

以下の場合には、アプリケーションが MQPUT または MQGET 呼び出しでこのフィールドを設定する必要はありません。

- MQPUT 呼び出しで、MQPMO_LOGICAL_ORDER が指定されている場合。
- MQGET 呼び出しで、MQMO_MATCH_OFFSET が指定されていない場合。

以下、これらの呼び出しをレポート・メッセージ以外のメッセージに使用する場合の推奨方法について説明します。ただし、アプリケーションがそれらの設定を行わなかった場合、または呼び出しが MQPUT1 の場合、アプリケーションは、*Offset* に適切な値が設定されていることを確認する必要があります。

MQPUT および MQPUT1 呼び出しの入力時にキュー・マネージャーが使用する値については、[キューでの物理順序](#)で説明されています。MQPUT および MQPUT1 呼び出しの出力時に、キュー・マネージャーは、このフィールドにメッセージと共に送信された値を設定します。

論理メッセージの 1 つのセグメントに関するレポート・メッセージの場合、*OriginalLength* フィールド (MQOL_UNDEFINED ではない場合) は、キュー・マネージャーが保存するセグメント情報の中のオフセットを更新するために使用されます。

MQGET 呼び出しの入力時にキュー・マネージャーが使用する値については、[355 ページの表 510](#) で説明します。MQGET 呼び出しの出力時に、キュー・マネージャーは、このフィールドに、取り出されたメッセージの値を設定します。

フィールドの初期値は、0 です。Version が MQMD_VERSION_2 より小さい場合、このフィールドは無視されます。

MsgFlags (MQLONG)

メッセージの属性を指定したり、メッセージの処理を制御したりするフラグです。それらは以下のカテゴリーに分類されます：

- セグメント化フラグ
- 状況フラグ

セグメンテーションフラグ

メッセージがキューに対して大きすぎる場合、そのメッセージをキューに追加しようとする試みは通常失敗する。セグメント化とは、キュー・マネージャーまたはアプリケーションがメッセージをセグメントといういくつかの小さな単位に分割して、各セグメントを別個の物理メッセージとしてキューに入れるための手法を指します。メッセージを取り出すアプリケーションは、セグメントを 1 つずつ取り出すか、またはセグメントの再組み立てを実行して 1 つのメッセージにするようキュー・マネージャーに要求することができます。後者の場合、メッセージは MQGET 呼び出しで戻されます。後者の方法を行うには、MQGET 呼び出しで MQGMO_COMPLETE_MSG オプションを指定して、メッセージ全体を格納できるだけの長さのバッファを提供します。(MQGMO_COMPLETE_MSG オプションの詳細については、[326 ページの『MQGMO - 読み取りメッセージ・オプション』](#)を参照してください。) メッセージのセグメント化は、送信側のキュー・マネージャー、中間キュー・マネージャー、または宛先キュー・マネージャーで実行できます。

以下のいずれかのオプションを指定すると、メッセージのセグメント化を制御できます。

MQMF_SEGMENTATION_INHIBITED

このオプションを指定した場合、キュー・マネージャーはメッセージをセグメントに分割することはありません。すでにセグメントになっているメッセージに対して、このオプション指定すると、そのセグメントはさらに小さいセグメントに分割されることはありません。

このフラグの値は2進ゼロです。これはデフォルトです。

MQMF_SEGMENTATION_ALLOWED

このオプションを指定した場合、キュー・マネージャーはメッセージをセグメントに分割することができます。すでにセグメントになっているメッセージにこのオプションを指定した場合は、そのセグメントをさらにいくつかの小さなセグメントに分割することができます。

MQMF_SEGMENTATION_ALLOWED は、MQMF_SEGMENT または MQMF_LAST_SEGMENT が設定されていなくても設定できます。

- z/OS では、キュー・マネージャーはメッセージのセグメント化をサポートしません。メッセージがキューに対して大きすぎる場合は、MQPUT または MQPUT1 呼び出しが失敗し、理由コード MQRC_MSG_TOO_BIG_FOR_Q が出力されます。ただし、MQMF_SEGMENTATION_ALLOWED オプションを引き続き指定することができ、メッセージをリモート・キュー・マネージャーでセグメント化することができます。

キュー・マネージャーは、メッセージをセグメント化するときに、各セグメントとともに送信される MQMD のコピーの MQMF_SEGMENT フラグをオンにします。ただし、アプリケーションが MQPUT または MQPUT1 呼び出しで提供する MQMD 内のこれらのフラグの設定は変更しません。論理メッセージ内の最後のセグメントの場合には、キュー・マネージャーは、セグメントとともに送信される MQMD 内の MQMF_LAST_SEGMENT フラグもオンにします。

注: MQMF_SEGMENTATION_ALLOWED を指定して、MQPMO_LOGICAL_ORDER を指定せずにメッセージを書き込むときには注意が必要です。メッセージが次の条件に該当する場合、

- セグメントではない
- グループに属していない
- 転送されない

この場合、アプリケーションでは、キュー・マネージャーがメッセージごとに固有のグループ ID を生成するように、それぞれの MQPUT 呼び出しまたは MQPUT1 呼び出しの発行前に *GroupId* フィールドを MQGI_NONE にリセットしなければなりません。この操作を行わないと、互いに関連のないメッセージに同じグループ ID が割り当てられ、それ以降の処理で問題が発生することがあります。*GroupId* フィールドをリセットするタイミングについては、*GroupId* フィールドおよび MQPMO_LOGICAL_ORDER オプションの説明を参照してください。

キュー・マネージャーは、セグメント (必要な場合はセグメントにヘッダー・データを加えたもの) がキューに収まるように、必要に応じてメッセージをいくつかのセグメントに分割します。ただし、キュー・マネージャーが生成する 1 つのセグメントのサイズには下限があり、この下限より小さくできるのは、あるメッセージから作成された最後のセグメントだけです (アプリケーションが生成するセグメントのサイズの下限は 1 バイトです)。キュー・マネージャーが生成する各セグメントは、長さが均等にならないことがあります。キュー・マネージャーは、メッセージを次のように処理します。

- ユーザー定義形式は、16 バイトの倍数の境界で分割されます。キュー・マネージャーでは (最後のセグメントを除いて) 16 バイトより小さいセグメントは生成されません。
- 組み込み形式の場合は、MQFMT_STRING 形式を除き、存在するデータの性質に合った箇所で分割されます。ただし、キュー・マネージャーは IBM MQ ヘッダー構造の途中ではメッセージを分割しません。つまり、MQ ヘッダー構造体が 1 つ含まれているセグメントは、それ以上分割されず、結果として、そのメッセージの最小セグメント・サイズは 16 バイトより大きくなります。

キュー・マネージャーが生成する 2 番目以降のセグメントは、次のいずれかで開始されます。

- MQ ヘッダー構造
- アプリケーション・メッセージ・データの先頭
- アプリケーション・メッセージ・データの途中
- MQFMT_STRING 形式の場合は、存在するデータの性質 (SBCS、DBCS、またはこれらが混在した SBCS/DBCS) に関係なく分割される。ストリングが DBCS または SBCS/DBCS である場合は、文字セットの

変換ができないセグメントになることがあります。キュー・マネージャーは、(最後のセグメントを除いて) MQFMT_STRING メッセージを 16 バイトより小さいセグメントに分割することはありません。

- 各セグメントの先頭のデータを正しく記述するため、キュー・マネージャーは、各セグメントの MQMD の *Format* フィールド、*CodedCharSetId* フィールド、および *Encoding* フィールドを設定する。形式名は、組み込み形式名またはユーザー定義の形式名になります。
- *Offset* の値がゼロより大きいセグメントについて、MQMD の *Report* フィールドを変更する。各レポート・タイプについて、レポート・オプションが MQRO_*_WITH_DATA であるのに、セグメント内にユーザー・データ (IBM MQ ヘッダー構造が存在する場合にその後に続くデータ) の最初の 100 バイトのどれも含めることができない場合は、レポート・オプションを MQRO_* に変更する。

キュー・マネージャーは上記の規則に従いますが、それ以外でも予期せずにメッセージを分割することがあります。そのため、メッセージがどこで分割されるかについては推測しないでください。

持続メッセージの場合、キュー・マネージャーは作業単位の範囲内でのみセグメント化を実行できます。ただし、次のような点に注意してください。

- MQPUT または MQPUT1 呼び出しがユーザー定義の作業単位内で実行されていれば、その作業単位が使用されます。セグメント分割の処理中に呼び出しが失敗した場合、キュー・マネージャーはその呼び出しによりキューに入れられたセグメントをすべて削除します。ただし、このように失敗しても、作業単位は正常にコミットされます。
- 呼び出しがユーザー定義の作業単位以外で実行されていて、またユーザー定義の作業単位が存在していない場合、キュー・マネージャーは、この呼び出しの間だけの作業単位を作成します。呼び出しが成功すると、キュー・マネージャーは作業単位を自動的にコミットします。呼び出しが失敗すると、キュー・マネージャーは作業単位をバックアウトします。
- 呼び出しがユーザー定義の作業単位以外で実行されていて、さらにユーザー定義の作業単位も存在している場合、キュー・マネージャーはセグメント化を実行できません。メッセージをセグメントに分割する必要がない場合には、呼び出しは成功します。しかし、メッセージをセグメントに分割する必要がある場合には、呼び出しは失敗し、理由コード MQRC_UOW_NOT_AVAILABLE が戻ります。

非持続メッセージの場合、キュー・マネージャーはセグメント化を実行するために作業単位を使用できるようにする必要はありません。

セグメント化される可能性のあるメッセージのデータを変換する場合には特に注意が必要です。

- MQGET 呼び出しで受信側のアプリケーションがデータを変換し、かつ MQGMO_COMPLETE_MSG オプションを指定した場合は、データ変換出口で変換を行うために、データ変換出口に完全なメッセージが渡されます。メッセージがセグメントに分割されたことがデータ変換出口で認識されます。
- 受信側のアプリケーションが 1 回につき 1 つのセグメントを取り出す場合は、データ変換出口を呼び出して 1 回につき 1 つのセグメントを変換します。したがって、データ変換出口は各セグメント内のデータを他のセグメント内のデータと関係なく変換することが必要となります。

メッセージのデータの性質上、16 バイト境界の任意の位置でデータをセグメントに分割するとデータ変換出口で変換できないセグメントになるような場合、または形式が MQFMT_STRING で文字セットが DBCS または SBCS/DBCS である場合は、送信側のアプリケーションがセグメントを作成して書き込み、MQMF_SEGMENTATION_INHIBITED を指定してこれらのセグメントがさらにいくつかのセグメントに分割されないようにする必要があります。これにより、送信側のアプリケーションでは、データ変換出口が各セグメントを正常に変換できるだけの十分な情報が各セグメントに含まれるようにすることができます。

- 送信側のメッセージ・チャンネル・エージェント (MCA) に対して送信側での変換を指定した場合、MCA は論理メッセージのセグメントでないメッセージのみを変換し、論理メッセージのセグメントであるメッセージは変換しません。

このフラグは、MQPUT 呼び出しおよび MQPUT1 呼び出しでは入力フラグであり、MQGET 呼び出しでは出力フラグです。後者の呼び出しの場合、キュー・マネージャーはこのフラグの値を MQGMO の *Segmentation* フィールドにも書き出します。

フラグの初期値は、MQMF_SEGMENTATION_INHIBITED です。

ステータスフラグ

これらは、物理メッセージがメッセージグループに属するか、論理メッセージのセグメントであるか、その両方であるか、あるいはどちらでもないかを示すフラグです。MQPUT 呼び出し、または MQPUT1 呼び出しでは、以下のオプションのうち 1 つ以上を指定できます。MQGET 呼び出しでは以下のオプションのうち 1 つ以上が戻されます。

MQMF_MSG_IN_GROUP

メッセージは特定のグループのメンバーである。

MQMF_LAST_MSG_IN_GROUP

メッセージはグループ内の最後の論理メッセージです。

このフラグを設定すると、キュー・マネージャーはメッセージで送信される MQMD のコピーの MQMF_MSG_IN_GROUP をオンにしますが、MQPUT または MQPUT1 呼び出しでアプリケーションで提供される MQMD のフラグの設定は変更しません。

このフラグは 1 つの論理メッセージのみで構成されるグループにも有効です。この場合にも、MQMF_LAST_MSG_IN_GROUP は設定されますが、*MsgSeqNumber* フィールドの値は 1 になります。

MQMF_SEGMENT

メッセージは論理メッセージのセグメントです。

MQMF_LAST_SEGMENT を指定せずに MQMF_SEGMENT を指定する場合、セグメント内のアプリケーションメッセージデータの長さ（存在する可能性 IBM MQ のあるヘッダー構造体の長さは除く）は、少なくとも 1 以上でなければならない。長さがゼロの場合、MQPUT または MQPUT1 呼び出しは、理由コード MQRC_SEGMENT_LENGTH_ZERO で失敗します。

z/OS では、索引タイプが MQIT_GROUP_ID であるキューにメッセージが入れられる場合、このオプションはサポートされません。

MQMF_LAST_SEGMENT

メッセージは論理メッセージの最後のセグメントです。

このフラグを設定すると、キュー・マネージャーはメッセージで送信される MQMD のコピーの MQMF_SEGMENT をオンにしますが、MQPUT または MQPUT1 呼び出しでアプリケーションで提供される MQMD のフラグの設定は変更しません。

論理メッセージは 1 つのセグメントだけで構成することができます。この場合にも、このフラグは設定されますが、*Offset* フィールドの値はゼロになります。

MQMF_LAST_SEGMENT を指定した場合、セグメント内のアプリケーション・メッセージ・データの長さ（ヘッダー構造が存在する場合にはその長さを除く）はゼロにすることができます。

z/OS では、索引タイプが MQIT_GROUP_ID であるキューにメッセージが入れられる場合、このオプションはサポートされません。

アプリケーションでは、メッセージの書き込み時にこれらのフラグが正しく設定されるようにする必要があります。MQPMO_LOGICAL_ORDER を指定した場合、または前に行ったキュー・ハンドルに対する MQPUT 呼び出しで MQPMO_LOGICAL_ORDER を指定した場合は、このフラグの設定値はキュー・マネージャーがキュー・ハンドル用に保存するグループ情報およびセグメント情報が矛盾してはなりません。MQPMO_LOGICAL_ORDER を指定した場合、キュー・ハンドルに対して連続した MQPUT 呼び出しを行うときには以下の条件があります。

- 現行のグループまたは論理メッセージがない場合は、上記のすべてのフラグ（およびこれらのフラグを組み合わせたもの）が有効である。
- MQMF_MSG_IN_GROUP を指定した場合には、MQMF_LAST_MSG_IN_GROUP を指定するまで MQMF_MSG_IN_GROUP をオンにしておくこと。この条件が満たされないと、呼び出しは失敗し、理由コード MQRC_INCOMPLETE_GROUP が戻ります。
- MQMF_SEGMENT を指定した場合には、MQMF_LAST_SEGMENT を指定するまで MQMF_SEGMENT をオンにしておくこと。この条件が満たされないと、呼び出しは失敗し、理由コード MQRC_INCOMPLETE_MSG が戻ります。
- MQMF_MSG_IN_GROUP を指定しないで MQMF_SEGMENT を指定した場合には、MQMF_LAST_SEGMENT を指定するまで MQMF_MSG_IN_GROUP をオフにしておくこと。この条件が満たされないと、呼び出しは失敗し、理由コード MQRC_INCOMPLETE_MSG が戻ります。

キューでの物理順序に、これらのフラグの有効な組み合わせと各種のフィールドに使用する値が示されています。

これらのフラグは、MQPUT 呼び出し、および MQPUT1 呼び出しでは入力フラグであり、MQGET 呼び出しでは出力フラグです。後者の呼び出しの場合、キュー・マネージャーはフラグの値を MQGMO の *GroupStatus* フィールドおよび *SegmentStatus* フィールドにも書き出します。

パブリッシュ/サブスクライブでは、グループ化されたメッセージやセグメント化されたメッセージは使用できません。

デフォルトフラグ

以下の指定により、メッセージがデフォルト属性を持つことを示すことができます：

MQMF_NONE

メッセージ・フラグはありません (デフォルトのメッセージ属性)。

これはセグメンテーションを禁止し、メッセージがグループに属していないこと、およびメッセージが論理メッセージのセグメントではないことを示します。MQMF_NONE は、プログラム・ドキュメンテーションの援助機能として定義されています。このフラグを他の目的で使用することは意図されていませんが、値がゼロであるため、そのように使用しても検出されることはありません。

MsgFlags フィールドはいくつかのサブフィールドに分かれています。詳細については、[898 ページの『レポート・オプションおよびメッセージ・フラグ』](#)を参照してください。

このフィールドの初期値は MQMF_NONE です。Version が MQMD_VERSION_2 より小さい場合、このフィールドは無視されます。

OriginalLength (MQLONG)

このフィールドが関係するのは、セグメントであるレポート・メッセージのみです。これが指定するのは、レポート・メッセージが関係するメッセージ・セグメントの長さであり、セグメントが形成する論理メッセージの長さ、またはレポート・メッセージ内のデータの長さではありません。

注：セグメントであるメッセージのレポート・メッセージを生成する場合、キュー・マネージャーとメッセージ・チャンネル・エージェントは、元のメッセージの *GroupId*、*MsgSeqNumber*、*Offset*、および *MsgFlags* の各フィールドをレポート・メッセージの MQMD にコピーします。その結果、そのレポート・メッセージも 1 つのセグメントです。レポート・メッセージを生成するアプリケーションは、同じことを行い、*OriginalLength* フィールドを正しく設定する必要があります。

以下のような特殊値が定義されます。

MQOL_UNDEFINED

元のメッセージの長さが定義されていません。

OriginalLength は MQPUT 呼び出しおよび MQPUT1 呼び出しの入力フィールドですが、アプリケーションが提供する値は特定の状況でのみ受け入れられます。

- 書き込んでいるメッセージがセグメントであると同時にレポート・メッセージでもある場合、キュー・マネージャーは指定された値を受け入れます。値は次のものでなければなりません。
 - セグメントが最後のセグメントではない場合は、ゼロより大きい値
 - セグメントが最後のセグメントである場合は、ゼロ以上の値
 - メッセージに含まれるデータの長さ以上の値

これらの条件を満たさない場合、呼び出しは失敗し、理由コード MQRC_ORIGINAL_LENGTH_ERROR が返されます。

- 書き込んでいるメッセージがセグメントであるが、レポート・メッセージではない場合、キュー・マネージャーはこのフィールドを無視して、アプリケーション・メッセージ・データの長さを使用します。
- それ以外の場合、キュー・マネージャーはこのフィールドを無視して、MQOL_UNDEFINED を使用します。

これは、MQGET 呼び出し用の出力フィールドです。

このフィールドの初期値は MQOL_UNDEFINED です。Version が MQMD_VERSION_2 より小さい場合、このフィールドは無視されます。

MQMDE - 拡張メッセージ記述子

MQMDE 構造は、アプリケーションメッセージデータに先行して発生することがあるデータを記述する。この構造体には、バージョン 2 の MQMD には存在するが、バージョン 1 の MQMD には存在しない MQMD フィールドがあります。

可用性

すべての IBM MQ システム、およびこれらのシステムに接続された IBM MQ MQI clients。

形式名

MQFMT_MD_EXTENSION

文字セットとエンコード

MQMDE 内のデータは、ローカルキューマネージャの文字セットおよびエンコーディングでなければならない。これらはキューマネージャ **CodedCharSetId** 属性および C プログラミング言語では MQENC_NATIVE によって指定される。

MQMDE の文字セットおよびエンコードは、以下の構造体の *CodedCharSetId* および *Encoding* フィールドに設定する必要があります。

- MQMD (MQMDE 構造体がメッセージ・データの開始点にある場合)
- MQMDE 構造体に先行するヘッダー構造体 (その他のすべての場合)

MQMDE がキュー・マネージャの文字セットとエンコードにない場合、MQMDE は無効にはなりません。つまり、MQMDE はメッセージ・データとして扱われます。

注: Windows の場合、Micro Focus COBOL でコンパイルされたアプリケーションでは、キュー・マネージャのエンコードとは異なる値である MQENC_NATIVE を使用します。MQPUT 呼び出し、MQPUT1 呼び出し、および MQGET 呼び出しでの MQMD 構造体の数値フィールドは、Micro Focus COBOL のエンコードで記述する必要がありますが、MQMDE 構造体の数値フィールドはキュー・マネージャのエンコードで記述する必要があります。後者は C プログラミング言語の MQENC_NATIVE で与えられ、その値は 546 になります。

使用法

version-2 の MQMD を使用するアプリケーションは、MQMDE 構造体に遭遇することはありません。しかし、特殊な用途のアプリケーションおよびバージョン 1 の MQMD を引き続き使用するアプリケーションでは、ある特定の場合に MQMDE 構造体が検出されることがあります。MQMDE 構造体が存在するのは以下のような場合です。

- MQPUT 呼び出しおよび MQPUT1 呼び出しで指定された場合
- MQGET 呼び出しから戻された場合
- 伝送キューのメッセージ内

MQPUT および MQPUT1 呼び出しで指定された MQMDE

MQPUT 呼び出しおよび MQPUT1 呼び出しでアプリケーションからバージョン 1 の MQMD を提供する場合には、オプションとして、メッセージ・データの先頭に MQMDE を付けることができます。その場合は、MQMD の *Format* フィールドに MQFMT_MD_EXTENSION を設定して、MQMDE が存在することを指定します。アプリケーションが MQMDE を提供しない場合、キュー・マネージャは MQMDE の各フィールドにデフォルト値が設定されたものと見なします。キュー・マネージャが使用するデフォルト値は、この構造体の初期値と同じです ([441 ページの表 521](#) を参照)。

アプリケーションが version-2 MQMD を提供し、アプリケーションメッセージデータに MQMDE をプレフィックスとして付加する場合、構造体は以下の表に示すように処理される。

表 520. MQPUT または MQPUT1 で MQMDE が指定された場合のキュー・マネージャーのアクション (MQMDE)			
MQMD Version	バージョン 2 のフィールドの値	MQMDE 内の対応するフィールドの値	キュー・マネージャーによるアクション
1	-	有効	MQMDE を参照する
2	デフォルト	有効	MQMDE を参照する
2	デフォルト値以外	有効	MQMDE をメッセージ・データとして扱う
1 または 2	任意	無効	呼び出しは失敗し、該当する理由コードが戻る
1 または 2	任意	MQMDE が無効な文字セットまたはエンコードで記述されているか、サポートされていないバージョンである	MQMDE をメッセージ・データとして扱う
注: z/OS では、アプリケーションが MQMDE を使用してバージョン 1 の MQMD を指定する場合、キュー・マネージャーは、キューの <i>IndexType</i> が MQIT_GROUP_ID である場合のみ MQMDE の妥当性検査を行います。			

ただし、特殊な場合が 1 つあります。アプリケーションがバージョン 2 の MQMD を使用してセグメントである (MQMF_SEGMENT フラグまたは MQMF_LAST_SEGMENT フラグが設定された) メッセージを書き込むときに、MQMD での形式名が MQFMT_DEAD_LETTER_HEADER である場合、キュー・マネージャーは MQMDE 構造体を生成して、MQDLH 構造体とその後に続くデータの間に挿入します。キュー・マネージャーがメッセージと共に保存する MQMD 内のバージョン 2 のフィールドにはデフォルト値が設定されません。

バージョン 2 の MQMD には存在するが、バージョン 1 の MQMD には存在しないフィールドの中には、MQPUT および MQPUT1 で入出力フィールドになるものもあります。ただし、キュー・マネージャーは、MQPUT 呼び出しおよび MQPUT1 呼び出しで出力が生成されても MQMDE の該当するフィールドに値を戻しません。出力値が必要な場合には、そのアプリケーションでバージョン 2 の MQMD を使用する必要があります。

MQGET 呼び出しで MQMDE が戻される場合

MQGET 呼び出しでアプリケーションからバージョン 1 の MQMD を提供した場合は、MQMDE の 1 つ以上のフィールドがデフォルト以外の値の場合に限り、キュー・マネージャーから返されるメッセージの先頭に MQMDE が付けられます。キュー・マネージャーは、MQMD 内の *Format* フィールドを値 MQFMT_MD_EXTENSION に設定し、MQMDE が存在することを示します。

アプリケーションが **Buffer** パラメーターの先頭に MQMDE を設定しても、その MQMDE は無視されます。MQGET 呼び出しからの戻り時には、MQMDE はメッセージの MQMDE に置換されている (MQMDE が必要な場合) か、またはアプリケーション・メッセージ・データで上書きされています (MQMDE が不要な場合)。

MQGET 呼び出しが MQMDE を戻す場合、MQMDE のデータは通常、キュー・マネージャーの文字セットおよびエンコードで記述されます。ただし以下のような場合は、MQMDE が別の文字セットおよびエンコードの形になっていることがあります。

- MQMDE が MQPUT 呼び出しまたは MQPUT1 呼び出しのデータとして扱われた (これが生じる状況については、[440 ページの表 520](#) を参照)。
- TCP 接続で接続されたリモート・キュー・マネージャーからメッセージを受け取ったが、受信側のメッセージ・チャンネル・エージェント (MCA) が正しくセットアップされていなかった。

注: Windows の場合、Micro Focus COBOL でコンパイルされたアプリケーションではキュー・マネージャーのエンコード方式とは異なる値である MQENC_NATIVE を使用します (上記参照)。

伝送キューのメッセージ内に MQMDE がある場合

伝送キュー内のメッセージには、接頭部として MQXQH 構造体が付いています。この構造体の中には、バージョン 1 の MQMD が格納されています。MQMDE も存在する場合があります、MQXQH 構造体とアプリケーションメッセージデータの間に配置されるが、通常は MQMDE 内の 1 つ以上のフィールドがデフォルト値以外の場合にのみ存在する。

他の MQ ヘッダー構造体も MQXQH 構造体とアプリケーション・メッセージ・データの間に存在する場合があります。例えば、送達不能ヘッダー MQDLH があり、メッセージがセグメントでない場合は、次のような順序で配置されます。

- MQXQH (バージョン 1 の MQMD を格納)
- MQMDE
- MQDLH
- アプリケーション・メッセージ・データ

フィールド

注: 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 521. MQMDE のフィールド		
フィールド名と説明	定数の名前	定数の初期値 (存在する場合)
<u>StrucId</u> (構造識別子)	MQMDE_STRUC_ID	'MDE'
<u>バージョン</u> (構造バージョン番号)	MQMDE_VERSION_2	2
<u>StrucLength</u> (MQMDE 構造の長さ)	MQMDE_LENGTH_2	72
<u>エンコーディング</u> (MQMDE に続くデータの数字エンコーディング)	MQENC_NATIVE	環境に依存
<u>CodedCharSetId</u> (MQMDE に続くデータの文字セット識別子)	MQCCSI_UNDEFINED	0
<u>フォーマット</u> (MQMDE に続くデータのフォーマット名)	MQFMT_NONE	ブランク
<u>フラグ</u> (一般的なフラグ)	MQMDEF_NONE	0
<u>GroupId</u> (グループ識別子)	MQGI_NONE	Null
<u>MsgSeqNumber</u> (グループ内の論理メッセージのシーケンス番号)	なし	1
<u>オフセット</u> (物理メッセージ内のデータの論理メッセージの先頭からのオフセット)	なし	0
<u>MsgFlags</u> (メッセージフラグ)	MQMF_NONE	0
<u>OriginalLength</u> (原文の長さ)	MQOL_UNDEFINED	-1

表 521. MQMDE のフィールド (続き)

フィールド名と説明	定数の名前	定数の初期値 (存在する場合)
注: - 記号-は、単一のブランク文字を表します。 - C プログラミング言語では、マクロ変数 MQMDE_DEFAULT には、表にリストされている値が含まれています。この変数を以下の方法で使用すると、構造体のフィールドに初期値を設定できます。 <pre>MQMDE MyMDE = {MQMDE_DEFAULT};</pre>		

言語ごとの宣言

MQMDE の C 宣言

```
typedef struct tagMQMDE MQMDE;
struct tagMQMDE {
    MQCHAR4    StrucId;        /* Structure identifier */
    MQLONG     Version;        /* Structure version number */
    MQLONG     StrucLength;    /* Length of MQMDE structure */
    MQLONG     Encoding;       /* Numeric encoding of data that follows
                               MQMDE */
    MQLONG     CodedCharSetId; /* Character-set identifier of data that
                               follows MQMDE */
    MQCHAR8     Format;        /* Format name of data that follows
                               MQMDE */
    MQLONG     Flags;          /* General flags */
    MQBYTE24    GroupId;       /* Group identifier */
    MQLONG     MsgSeqNumber;    /* Sequence number of logical message
                               within group */
    MQLONG     Offset;         /* Offset of data in physical message from
                               start of logical message */
    MQLONG     MsgFlags;       /* Message flags */
    MQLONG     OriginalLength; /* Length of original message */
};
```

MQMDE の COBOL 宣言

```
**      MQMDE structure
10 MQMDE.
**      Structure identifier
15 MQMDE-STRUCID          PIC X(4).
**      Structure version number
15 MQMDE-VERSION          PIC S9(9) BINARY.
**      Length of MQMDE structure
15 MQMDE-STRUCLength     PIC S9(9) BINARY.
**      Numeric encoding of data that follows MQMDE
15 MQMDE-ENCODING        PIC S9(9) BINARY.
**      Character-set identifier of data that follows MQMDE
15 MQMDE-CODEDCHARSETID PIC S9(9) BINARY.
**      Format name of data that follows MQMDE
15 MQMDE-FORMAT          PIC X(8).
**      General flags
15 MQMDE-FLAGS           PIC S9(9) BINARY.
**      Group identifier
15 MQMDE-GROUPID         PIC X(24).
**      Sequence number of logical message within group
15 MQMDE-MSGSEQNUMBER    PIC S9(9) BINARY.
**      Offset of data in physical message from start of logical message
15 MQMDE-OFFSET          PIC S9(9) BINARY.
**      Message flags
15 MQMDE-MSGFLAGS        PIC S9(9) BINARY.
**      Length of original message
15 MQMDE-ORIGINALLENGTH PIC S9(9) BINARY.
```

MQMDE の PL/I 宣言

```
dcl
1 MQMDE based,
3 StrucId          char(4),          /* Structure identifier */
```

```

3 Version          fixed bin(31), /* Structure version number */
3 StrucLength      fixed bin(31), /* Length of MQMDE structure */
3 Encoding         fixed bin(31), /* Numeric encoding of data that
                                follows MQMDE */
3 CodedCharSetId   fixed bin(31), /* Character-set identifier of data
                                that follows MQMDE */
3 Format           char(8),        /* Format name of data that follows
                                MQMDE */
3 Flags           fixed bin(31), /* General flags */
3 GroupId         char(24),        /* Group identifier */
3 MsgSeqNumber     fixed bin(31), /* Sequence number of logical message
                                within group */
3 Offset          fixed bin(31), /* Offset of data in physical message
                                from start of logical message */
3 MsgFlags        fixed bin(31), /* Message flags */
3 OriginalLength   fixed bin(31); /* Length of original message */

```

MQMDE の高水準アセンブラ宣言

MQMDE	DSECT		
MQMDE_STRUCID	DS	CL4	Structure identifier
MQMDE_VERSION	DS	F	Structure version number
MQMDE_STRUCLNGTH	DS	F	Length of MQMDE structure
MQMDE_ENCODING	DS	F	Numeric encoding of data that follows MQMDE
*MQMDE_CODEDCHARSETID	DS	F	Character-set identifier of data that follows MQMDE
*MQMDE_FORMAT	DS	CL8	Format name of data that follows MQMDE
MQMDE_FLAGS	DS	F	General flags
MQMDE_GROUPID	DS	XL24	Group identifier
MQMDE_MSGSEQNUMBER	DS	F	Sequence number of logical message within group
*MQMDE_OFFSET	DS	F	Offset of data in physical message from start of logical message
*MQMDE_MSGFLAGS	DS	F	Message flags
MQMDE_ORIGINALLENGTH	DS	F	Length of original message
*MQMDE_LENGTH	EQU	*-MQMDE	
	ORG	MQMDE	
MQMDE_AREA	DS	CL(MQMDE_LENGTH)	

MQMDE の Visual Basic 宣言

```

Type MQMDE
    StrucId          As String*4 'Structure identifier'
    Version          As Long      'Structure version number'
    StrucLength      As Long      'Length of MQMDE structure'
    Encoding         As Long      'Numeric encoding of data that follows'
                                'MQMDE'
    CodedCharSetId   As Long      'Character-set identifier of data that'
                                'follows MQMDE'
    Format           As String*8  'Format name of data that follows MQMDE'
    Flags           As Long      'General flags'
    GroupId         As MQBYTE24  'Group identifier'
    MsgSeqNumber     As Long      'Sequence number of logical message within'
                                'group'
    Offset          As Long      'Offset of data in physical message from'
                                'start of logical message'
    MsgFlags        As Long      'Message flags'
    OriginalLength   As Long      'Length of original message'
End Type

```

MQMDE のフィールド詳細

MQMDE メッセージ記述子拡張構造は、アプリケーションメッセージデータの前に発生することがあるデータを記述する。この構造体には、バージョン 2 の MQMD には存在するが、バージョン 1 の MQMD には存在しない MQMD フィールドがあります。

フィールド

注: 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 522. MQMDE のフィールド

フィールド名と説明	定数の名前	定数の初期値（存在する場合）
<u>StrucId</u> (構造識別子)	MQMDE_STRUC_ID	'MDE↵'
<u>バージョン</u> (構造バージョン番号)	MQMDE_VERSION_2	2
<u>StrucLength</u> (MQMDE 構造の長さ)	MQMDE_LENGTH_2	72
<u>エンコーディング</u> (MQMDE に続くデータの数字エンコーディング)	MQENC_NATIVE	環境に依存
<u>CodedCharSetId</u> (MQMDE に続くデータの文字セット識別子)	MQCCSI_UNDEFINED	0
<u>フォーマット</u> (MQMDE に続くデータのフォーマット名)	MQFMT_NONE	ブランク
<u>フラグ</u> (一般的なフラグ)	MQMDEF_NONE	0
<u>GroupId</u> (グループ識別子)	MQGI_NONE	Null
<u>MsgSeqNumber</u> (グループ内の論理メッセージのシーケンス番号)	なし	1
<u>オフセット</u> (物理メッセージ内のデータの論理メッセージの先頭からのオフセット)	なし	0
<u>MsgFlags</u> (メッセージフラグ)	MQMF_NONE	0
<u>OriginalLength</u> (原文の長さ)	MQOL_UNDEFINED	-1
注： 1. 記号↵は、単一のブランク文字を表します。 2. C プログラミング言語では、マクロ変数 MQMDE_DEFAULT には、表にリストされている値が含まれています。この変数を以下の方法で使用すると、構造体のフィールドに初期値を設定できます。 <pre>MQMDE MyMDE = {MQMDE_DEFAULT};</pre>		

StrucId (MQCHAR4)

これはメッセージ記述子拡張構造体の構造体識別子です。常に入力フィールドです。その値は MQMDE_STRUC_ID です。

値は次のものでなければなりません。

MQMDE_STRUC_ID

メッセージ記述子拡張構造体の識別子。

C プログラミング言語では、定数 MQMDE_STRUC_ID_ARRAY も定義されている。これは MQMDE_STRUC_ID と同じ値を持ちますが、文字列ではなく文字の配列です。

Version (MQLONG)

これは構造体のバージョン番号です。値は以下のものでなければなりません。

MQMDE_VERSION_2

バージョン 2 の拡張メッセージ記述子の構造体。

以下の定数は、現行バージョンのバージョン番号を指定しています。

MQMDE_CURRENT_VERSION

拡張メッセージ記述子の構造体の現行バージョン。

フィールドの初期値は、MQMDE_VERSION_2 です。

StrucLength (MQLONG)

これは、MQMDE 構造体の長さです。以下の値が定義されています。

MQMDE_LENGTH_2

バージョン 2 の拡張メッセージ記述子の構造体の長さ。

フィールドの初期値は、MQMDE_LENGTH_2 です。

Encoding (MQLONG)

これは、MQMDE 構造体の後続くデータの数値エンコードを指定します。MQMDE 構造体自体の数値データには適用されません。

MQPUT または MQPUT1 呼び出しでは、アプリケーションは、このフィールドをデータに適切な値に設定する必要があります。キュー・マネージャーは、フィールドが有効かどうかをチェックしません。データエンコーディングの詳細については、[385 ページの『MQMD - メッセージ記述子』](#)に記載されている [エンコーディングフィールド](#) を参照してください。

このフィールドの初期値は MQENC_NATIVE です。

CodedCharSetId (MQLONG)

ここでは、MQMDE 構造体の後続くデータの文字セット ID を指定します。これは、MQMDE 構造体自体の文字データには適用されません。

MQPUT または MQPUT1 呼び出しでは、アプリケーションは、このフィールドをデータに適切な値に設定する必要があります。キュー・マネージャーは、このフィールドが有効かどうかをチェックしません。以下のような特別な値を使用することができます。

MQCCSI_INHERIT

この構造体の後続くデータの文字データは、この構造体に設定されているのと同じ文字セットになります。

キュー・マネージャーは、メッセージで送信される構造体の中のこの値を、構造体の実際の文字セット ID に変更します。エラーが発生しない限り、値 MQCCSI_INHERIT が MQGET 呼び出しによって返されることはありません。

MQCCSI_INHERIT は、MQMD の *PutApplType* フィールドの値が MQAT_BROKER である場合は使用できません。

この値は、次の環境でサポートされます。

-  AIX
-  IBM i
-  Linux
-  Windows

および、これらのシステムに接続された IBM MQ クライアント。

このフィールドの初期値は MQCCSI_UNDEFINED です。

Format (MQCHAR8)

ここでは、MQMDE 構造体の後続くデータの形式名を指定します。

MQPUT または MQPUT1 呼び出しでは、アプリケーションは、このフィールドをデータに適切な値に設定する必要があります。キュー・マネージャーは、このフィールドが有効かどうかをチェックしません。フォーマット名に関する詳細については、[フォーマットフィールド](#) [385 ページの『MQMD - メッセージ記述子』](#)の説明を参照してください。

このフィールドの初期値は MQFMT_NONE です。

フラグ (MQLONG)

以下のフラグを指定できます。

MQMDEF_NONE
フラグなし。

フィールドの初期値は、MQMDEF_NONE です。

GroupId (MQBYTE24)

に記載されている 385 ページの『MQMD - メッセージ記述子』フィールド [GroupId](#) を参照してください。
このフィールドの初期値は MQGI_NONE です。

MsgSeqNumber (MQLONG)

に記載されている 385 ページの『MQMD - メッセージ記述子』フィールド [MsgSeqNumber](#) を参照してください。
このフィールドの初期値は 1 です。

Offset (MQLONG)

オフセットフィールドについては、に記載されている 385 ページの『MQMD - メッセージ記述子』内容を参照してください。
このフィールドの初期値は 0 です。

MsgFlags (MQLONG)

に記載されている 385 ページの『MQMD - メッセージ記述子』フィールド [MsgFlags](#) を参照してください。
このフィールドの初期値は MQMF_NONE です。

OriginalLength (MQLONG)

に記載されている 385 ページの『MQMD - メッセージ記述子』フィールド [OriginalLength](#) を参照してください。
このフィールドの初期値は MQOL_UNDEFINED です。

MQMHBO - メッセージ・ハンドルからバッファーへの変換オプション

アプリケーションでは、MQMHBO 構造体を使用することによって、メッセージ・ハンドルからバッファーを生成する方法を制御するためのオプションを指定することができます。この構造体は、MQMHBUF 呼び出しの入力パラメーターです。

文字セットとエンコード

MQMHBO 内のデータは、アプリケーションの文字セットおよびアプリケーションのエンコーディング (MQENC_NATIVE) でなければならない。

フィールド

注: 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 523. MQMHBO のフィールド		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
StrucId (構造識別子)	MQMHBO_STRUC_ID	'MHBO'
バージョン (構造バージョン番号)	MQMHBO_VERSION_1	1

表 523. MQMHBO のフィールド (続き)

フィールド名と説明	定数の名前	定数の初期値 (存在する場合)
<u>オプション</u> (MQMHBUF の動作を制御するオプション)	MQMHBO_PROPERTIES_I N_MQRFH2	

注:

1. ヌル・ストリングまたはブランクの値は、C 言語ではヌル・ストリングを表し、他のプログラミング言語ではブランク文字を表します。
2. C プログラミング言語では、マクロ変数 MQMHBO_DEFAULT には、表にリストされている値が含まれています。このマクロ変数を以下の方法で使用して、構造体のフィールドに初期値を設定します。

```
MQMHBO MyMHBO = {MQMHBO_DEFAULT};
```

言語ごとの宣言**MQMHBO の C 宣言**

```
typedef struct tagMQMHBO MQMHBO;
struct tagMQMHBO {
    MQCHAR4  StrucId;          /* Structure identifier */
    MQLONG   Version;         /* Structure version number */
    MQLONG   Options;         /* Options that control the action of
                               MQMHBUF */
};
```

MQMHBO の COBOL 宣言

```
** MQMHBO structure
10 MQMHBO.
**   Structure identifier
15 MQMHBO-STRUCID          PIC X(4).
**   Structure version number
15 MQMHBO-VERSION         PIC S9(9) BINARY.
**   Options that control the action of MQMHBUF
15 MQMHBO-OPTIONS        PIC S9(9) BINARY.
```

MQMHBO の PL/I 宣言

```
Dcl
1 MQMHBO based,
3 StrucId      char(4),          /* Structure identifier */
3 Version      fixed bin(31),    /* Structure version number */
3 Options      fixed bin(31),    /* Options that control the action
                                of MQMHBUF */
```

MQMHBO の高水準アセンブラ宣言

```
MQMHBO          DSECT
MQMHBO_STRUCID   DS    CL4  Structure identifier
MQMHBO_VERSION   DS    F    Structure version number
MQMHBO_OPTIONS   DS    F    Options that control the
*                  action of MQMHBUF
MQMHBO_LENGTH    EQU    *-MQMHBO
MQMHBO_AREA      DS    CL(MQMHBO_LENGTH)
```

MQMHBO のフィールド詳細

MQMHBO メッセージハンドルからバッファへのオプション構造体により、アプリケーションはメッセージハンドルからバッファが生成される方法を制御するオプションを指定できます。この構造体は、MQMHBUF 呼び出しの入力パラメーターです。

フィールド

注: 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 524. MQMHBO のフィールド		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
StrucId (構造識別子)	MQMHBO_STRUC_ID	'MHBO'
バージョン (構造バージョン番号)	MQMHBO_VERSION_1	1
オプション (MQMHBUF の動作を制御するオプション)	MQMHBO_PROPERTIES_IN_MQRFH2	
注: - ヌル・ストリングまたはブランクの値は、C 言語ではヌル・ストリングを表し、他のプログラミング言語ではブランク文字を表します。 - C プログラミング言語では、マクロ変数 MQMHBO_DEFAULT には、表にリストされている値が含まれています。このマクロ変数を以下の方法で使用して、構造体のフィールドに初期値を設定します。 <pre>MQMHBO MyMHBO = {MQMHBO_DEFAULT};</pre>		

StrucId (MQCHAR4)

これは、メッセージハンドルからバッファオプション構造体への構造体識別子です。常に入力フィールドです。その値は MQMHBO_STRUC_ID です。

値は次のものでなければなりません。

MQMHBO_STRUC_ID

メッセージ・ハンドルからバッファへの変換オプション構造の ID。

C プログラミング言語では、定数 MQMHBO_STRUC_ID_ARRAY も定義されている。これは MQMHBO_STRUC_ID と同じ値を持ちますが、文字列ではなく文字の配列です。

Version (MQLONG)

メッセージハンドルからバッファオプション構造体への変換 - バージョンフィールド これは構造体のバージョン番号です。値は次のものでなければなりません。

MQMHBO_VERSION_1

メッセージ・ハンドルからバッファへの変換オプション構造のバージョン番号。

以下の定数は、現行バージョンのバージョン番号を指定しています。

MQMHBO_CURRENT_VERSION

メッセージ・ハンドルからバッファへの変換オプション構造の現行バージョン。

これは常に入力フィールドです。このフィールドの初期値は、MQMHBO_VERSION_1 です。

Options (MQLONG)

メッセージハンドルからバッファオプション構造体への操作 - オプションフィールド これらのオプションは、MQMHBUF のアクションを制御します。

以下のオプションを指定しなければなりません。

MQMHBO_PROPERTIES_IN_MQRFH2

プロパティをメッセージ・ハンドルからバッファへに変換する際に、MQRFH2 形式に変換します。

オプションで、以下のオプションも指定できます。複数のオプションを指定するには、値を加算する (同じ定数を複数回加算しない) か、ビット単位 OR 演算を使用して値を結合 (プログラミング言語でビット演算がサポートされる場合) します。

MQMHBO_DELETE_PROPERTIES

バッファに追加されるプロパティが、メッセージ・ハンドルから削除される。呼び出しが失敗すると、プロパティは削除されません。

これは常に入力フィールドです。このフィールドの初期値は、MQMHBO_PROPERTIES_IN_MQRFH2 です。

MQOD - オブジェクト記述子

MQOD 構造体は、オブジェクトを名前で指定するために使用されます。この構造体は、MQOPEN および MQPUT1 呼び出しの入出力パラメーターです。

次のタイプのオブジェクトが有効です。

- キューまたは配布リスト
- 名前リスト
- プロセス定義
- キュー・マネージャー
- トピック

可用性

すべての IBM MQ システム、およびそれらに IBM MQ MQI clients 接続されているシステム。

バージョン

現在の MQOD のバージョンは MQOD_VERSION_4 です。複数の環境間で移植するアプリケーションでは、MQOD の必須バージョンが、関係するすべての環境で必ずサポートされていなければなりません。これより新しいバージョンの構造体にのみ存在するフィールドについては、そのフィールドの説明にその旨を記載しています。

サポートされるプログラム言語用に提供されているヘッダー・ファイル、コピー・ファイル、およびインクルード・ファイルには、環境でサポートされる最新バージョンの MQOD が含まれていますが、*Version* フィールドの初期値は MQOD_VERSION_1 に設定されています。version-1 構造体に存在しないフィールドを使用するには、アプリケーションで、*Version* フィールドを必要なバージョンのバージョン番号に設定する必要があります。

配布リストをオープンするには、*Version* が MQOD_VERSION_2 以上でなければなりません。

文字セットとエンコード

MQOD 内のデータは、キュー・マネージャ **CodedCharSetId** 属性によって指定される文字セットおよび MQENC_NATIVE によって指定されるローカル・キュー・マネージャのエンコーディングでなければなりません。ただし、アプリケーションが MQ MQI クライアントとして実行されている場合、構造体はクライアントの文字セットとエンコードに従っている必要があります。

フィールド

注: 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

フィールド名と説明	定数の名前	定数の初期値 (存在する場合)
<u>StrucId</u> (構造識別子)	MQOD_STRUC_ID	'OD---'
<u>バージョン</u> (構造バージョン番号)	MQOD_VERSION_1	1

フィールド名と説明	定数の名前	定数の初期値（存在する場合）
<u>ObjectType</u> (オブジェクトタイプ)	MQOT_Q	1
<u>ObjectName</u> (オブジェクト名)	なし	ヌル・ストリングまたは ブランク
<u>ObjectQMgrName</u> (オブジェクトキューマネージャ名)	なし	ヌル・ストリングまたは ブランク
<u>DynamicQName</u> (動的キュー名)	なし	'CSQ.*' ...の場合 z/OS ; 'AMQ.*' それ以外の 場合は
<u>AlternateUserId</u> (代替ユーザー識別子)	なし	ヌル・ストリングまたは ブランク
注: <i>Version</i> が MQOD_VERSION_2 より小さい場合は、残りのフィールドは無視されます。		
<u>RecsPresent</u> (存在するオブジェクトレコードの数)	なし	0
<u>KnownDestCount</u> (正常に開かれたローカルキューの数)	なし	0
<u>UnknownDestCount</u> (正常に開かれたリモートキューの数)	なし	0
<u>InvalidDestCount</u> (開けられなかったキューの数)	なし	0
<u>ObjectRecOffset</u> (MQOD の先頭からの最初のオブジェクトレコードのオフセット)	なし	0
<u>ResponseRecOffset</u> (MQOD の先頭からの最初の応答レコードのオフセット)	なし	0
<u>ObjectRecPtr</u> (最初のオブジェクトレコードのアドレス)	なし	ヌル・ポインターまたは ヌル・バイト
<u>ResponseRecPtr</u> (最初の応答レコードのアドレス)	なし	ヌル・ポインターまたは ヌル・バイト
注: <i>Version</i> が MQOD_VERSION_3 より小さい場合、残りのフィールドは無視されます。		
<u>AlternateSecurityId</u> (代替セキュリティ識別子)	MQSID_NONE	Null
<u>ResolvedQName</u> (解決済みキュー名)	なし	ヌル・ストリングまたは ブランク
<u>ResolvedQMgrName</u> (解決済みキューマネージャ名)	なし	ヌル・ストリングまたは ブランク
注: <i>Version</i> が MQOD_VERSION_4 より小さい場合は、残りのフィールドは無視されます。		
<u>ObjectString</u> (長いオブジェクト名)	MQCHARV_DEFAULT	MQCHARV で定義されて いるとおり
<u>SelectionString</u> (選択文字列)	MQCHARV_DEFAULT	MQCHARV で定義されて いるとおり
<u>ResObjectString</u> (解決済み長いオブジェクト名)	MQCHARV_DEFAULT	MQCHARV で定義されて いるとおり
<u>ResolvedType</u> (解決済みオブジェクトタイプ)	MQOT_NONE	0

フィールド名と説明	定数の名前	定数の初期値（存在する場合）
注： - 記号-は、単一のブランク文字を表します。 - ヌル・ストリングまたはブランクの値は、C 言語ではヌル・ストリングを表し、他のプログラミング言語ではブランク文字を表します。 - C プログラミング言語では、マクロ変数 MQOD_DEFAULT には、表にリストされている値が含まれています。この変数を以下の方法で使用すると、構造体のフィールドに初期値を設定できます。 <pre>MQOD MyOD = {MQOD_DEFAULT};</pre>		

言語ごとの宣言

MQOD の C 宣言

```
typedef struct tagMQOD MQOD;
struct tagMQOD {
    MQCHAR4      StructId;          /* Structure identifier */
    MQLONG       Version;           /* Structure version number */
    MQLONG       ObjectType;        /* Object type */
    MQCHAR48     ObjectName;        /* Object name */
    MQCHAR48     ObjectQMgrName;    /* Object queue manager name */
    MQCHAR48     DynamicQName;      /* Dynamic queue name */
    MQCHAR12     AlternateUserId;   /* Alternate user identifier */
    /* Ver:1 */
    MQLONG       RecsPresent;        /* Number of object records present */
    MQLONG       KnownDestCount;    /* Number of local queues opened
                                     successfully */
    MQLONG       UnknownDestCount;  /* Number of remote queues opened
                                     successfully */
    MQLONG       InvalidDestCount;  /* Number of queues that failed to
                                     open */
    MQLONG       ObjectRecOffset;   /* Offset of first object record from
                                     start of MQOD */
    MQLONG       ResponseRecOffset; /* Offset of first response record
                                     from start of MQOD */
    MQPTR        ObjectRecPtr;      /* Address of first object record */
    MQPTR        ResponseRecPtr;    /* Address of first response record */
    /* Ver:2 */
    MQBYTE40     AlternateSecurityId; /* Alternate security identifier */
    MQCHAR48     ResolvedQName;      /* Resolved queue name */
    MQCHAR48     ResolvedQMgrName;   /* Resolved queue manager name */
    /* Ver:3 */
    MQCHARV      ObjectString;       /* Object Long name */
    MQCHARV      SelectionString;    /* Message Selector */
    MQCHARV      ResObjectString;    /* Resolved Long object name */
    MQLONG       ResolvedType        /* Alias queue resolved
                                     object type */
    /* Ver:4 */
};
```

MQOD の COBOL 宣言

```
** MQOD structure
10  MQOD.
** Structure identifier
15  MQOD-STRUCID                PIC X(4).
** Structure version number
15  MQOD-VERSION                PIC S9(9) BINARY.
** Object type
15  MQOD-OBJECTTYPE            PIC S9(9) BINARY.
** Object name
15  MQOD-OBJECTNAME            PIC X(48).
** Object queue manager name
15  MQOD-OBJECTQMGRNAME        PIC X(48).
** Dynamic queue name
15  MQOD-DYNAMICQNAME          PIC X(48).
** Alternate user identifier
15  MQOD-ALTERNATEUSERID        PIC X(12).
** Number of object records present
```

```

15 MQOD-RECSPRESENT          PIC S9(9) BINARY.
** Number of local queues opened successfully
15 MQOD-KNOWNDESTCOUNT      PIC S9(9) BINARY.
** Number of remote queues opened successfully
15 MQOD-UNKNOWNDESTCOUNT    PIC S9(9) BINARY.
** Number of queues that failed to open
15 MQOD-INVALIDDESTCOUNT     PIC S9(9) BINARY.
** Offset of first object record from start of MQOD
15 MQOD-OBJECTRECOFFSET       PIC S9(9) BINARY.
** Offset of first response record from start of MQOD
15 MQOD-RESPONSERECOFFSET     PIC S9(9) BINARY.
** Address of first object record
15 MQOD-OBJECTRECPtr         POINTER.
** Address of first response record
15 MQOD-RESPONSERECPtr       POINTER.
** Alternate security identifier
15 MQOD-ALTERNATESECURITYID    PIC X(40).
** Resolved queue name
15 MQOD-RESOLVEDQNAME         PIC X(48).
** Resolved queue manager name
15 MQOD-RESOLVEDQMGRNAME      PIC X(48).
** Object Long name
15 MQOD-OBJECTSTRING.
** Address of variable length string
20 MQOD-OBJECTSTRING-VSPTR    POINTER.
** Offset of variable length string
20 MQOD-OBJECTSTRING-VSOFFSET PIC S9(9) BINARY.
** size of buffer
20 MQOD-OBJECTSTRING-VSBUFSIZE PIC S9(9) BINARY.
** Length of variable length string
20 MQOD-OBJECTSTRING-VSLENGTH PIC S9(9) BINARY.
** CCSID of variable length string
20 MQOD-OBJECTSTRING-VSCCSID  PIC S9(9) BINARY.
** Message Selector
15 MQOD-SELECTIONSTRING.
** Address of variable length string
20 MQOD-SELECTIONSTRING-VSPTR POINTER.
** Offset of variable length string
20 MQOD-SELECTIONSTRING-VSOFFSET PIC S9(9) BINARY.
** size of buffer
20 MQOD-SELECTIONSTRING-VSBUFSIZE PIC S9(9) BINARY.
** Length of variable length string
20 MQOD-SELECTIONSTRING-VSLENGTH PIC S9(9) BINARY.
** CCSID of variable length string
20 MQOD-SELECTIONSTRING-VSCCSID  PIC S9(9) BINARY.
** Resolved Long object name
15 MQOD-RESOBJECTSTRING.
** Address of variable length string
20 MQOD-RESOBJECTSTRING-VSPTR  POINTER.
** Offset of variable length string
20 MQOD-RESOBJECTSTRING-VSOFFSET PIC S9(9) BINARY.
** size of buffer
20 MQOD-RESOBJECTSTRING-VSBUFSIZE PIC S9(9) BINARY.
** Length of variable length string
20 MQOD-RESOBJECTSTRING-VSLENGTH PIC S9(9) BINARY.
** CCSID of variable length string
20 MQOD-RESOBJECTSTRING-VSCCSID  PIC S9(9) BINARY.
** Alias queue resolved object type
15 MQOD-RESOLVEDTYPE          PIC S9(9) BINARY.

```

MQOD の PL/I 宣言

```

dcl
1 MQOD based,
3 StrucId          char(4),          /* Structure identifier */
3 Version          fixed bin(31),    /* Structure version number */
3 ObjectType       fixed bin(31),    /* Object type */
3 ObjectName       char(48),         /* Object name */
3 ObjectQMGrName   char(48),         /* Object queue manager name */
3 DynamicQName     char(48),         /* Dynamic queue name */
3 AlternateUserId  char(12),         /* Alternate user identifier */
3 RecsPresent      fixed bin(31),    /* Number of object records
                                     present */
3 KnownDestCount   fixed bin(31),    /* Number of local queues opened
                                     successfully */
3 UnknownDestCount fixed bin(31),    /* Number of remote queues opened
                                     successfully */
3 InvalidDestCount fixed bin(31),    /* Number of queues that failed to
                                     open */
3 ObjectRecOffset  fixed bin(31),    /* Offset of first object record

```



```

3 ResponseRecOffset    fixed bin(31), /* from start of MQOD */
                        /* Offset of first response record
                        from start of MQOD */
3 ObjectRecPtr         pointer, /* Address of first object record */
3 ResponseRecPtr       pointer, /* Address of first response
                                record */
3 AlternateSecurityId  char(40), /* Alternate security identifier */
3 ResolvedQName        char(48), /* Resolved queue name */
3 ResolvedQMgrName     char(48), /* Resolved queue manager name */
3 ObjectString,        /* Object Long name */
5 VSPtr               pointer, /* Address of variable length string */
5 VSOFFSET            fixed bin(31), /* Offset of variable length string */
5 VSBUFSize           fixed bin(31), /* size of buffer */
5 VSLength            fixed bin(31), /* Length of variable length string */
5 VSCCSID             fixed bin(31), /* CCSID of variable length string */
3 SelectionString,    /* Message Selection */
5 VSPtr               pointer, /* Address of variable length string */
5 VSOFFSET            fixed bin(31), /* Offset of variable length string */
5 VSBUFSize           fixed bin(31), /* size of buffer */
5 VSLength            fixed bin(31), /* Length of variable length string */
5 VSCCSID             fixed bin(31), /* CCSID of variable length string */
3 ResObjectString,    /* Resolved Long object name */
5 VSPtr               pointer, /* Address of variable length string */
5 VSOFFSET            fixed bin(31), /* Offset of variable length string */
5 VSBUFSize           fixed bin(31), /* size of buffer */
5 VSLength            fixed bin(31), /* Length of variable length string */
5 VSCCSID             fixed bin(31), /* CCSID of variable length string */
3 ResolvedType        fixed bin(31); /* Alias queue resolved object type */

```

MQOD の高水準アセンブラ宣言

MQOD	DSECT	
MQOD_STRUCTID	DS	CL4 Structure identifier
MQOD_VERSION	DS	F Structure version number
MQOD_OBJECTTYPE	DS	F Object type
MQOD_OBJECTNAME	DS	CL48 Object name
MQOD_OBJECTQMGRNAME	DS	CL48 Object queue manager name
MQOD_DYNAMICQNAME	DS	CL48 Dynamic queue name
MQOD_ALTERNATEUSERID	DS	CL12 Alternate user identifier
MQOD_RECSPRESENT	DS	F Number of object records present
MQOD_KNOWNDESTCOUNT	DS	F Number of local queues opened
*		successfully
MQOD_UNKNOWNDDESTCOUNT	DS	F Number of remote queues opened
*		successfully
MQOD_INVALIDDESTCOUNT	DS	F Number of queues that failed to
*		open
MQOD_OBJECTRECOFFSET	DS	F Offset of first object record from
*		start of MQOD
MQOD_RESPONSERECOFFSET	DS	F Offset of first response record
*		from start of MQOD
MQOD_OBJECTRECPTTR	DS	F Address of first object record
MQOD_RESPONSERECPTTR	DS	F Address of first response record
MQOD_ALTERNATESECURITYID	DS	XL40 Alternate security identifier
MQOD_RESOLVEDQNAME	DS	CL48 Resolved queue name
MQOD_RESOLVEDQMGRNAME	DS	CL48 Resolved queue manager name
MQOD_OBJECTSTRING	DS	F Object Long name
MQOD_OBJECTSTRING_VSPTR	DS	F Address of variable length string
MQOD_OBJECTSTRING_VSOFFSET	DS	F Offset of variable length string
MQOD_OBJECTSTRING_VSBUFSize	DS	F size of buffer
MQOD_OBJECTSTRING_VSLength	DS	F Length of variable length string
MQOD_OBJECTSTRING_VSCCSID	DS	F CCSID of variable length string
MQOD_OBJECTSTRING_LENGTH	EQU	*- MQOD_OBJECTSTRING
	ORG	MQOD_OBJECTSTRING
MQOD_OBJECTSTRING_AREA	DS	CL(MQOD_OBJECTSTRING_LENGTH)
*		
MQOD_SELECTIONSTRING	DS	F Message Selector
MQOD_SELECTIONSTRING_VSPTR	DS	F Address of variable length string
MQOD_SELECTIONSTRING_VSOFFSET	DS	F Offset of variable length string
MQOD_SELECTIONSTRING_VSBUFSize	DS	F size of buffer
MQOD_SELECTIONSTRING_VSLength	DS	F Length of variable length string
MQOD_SELECTIONSTRING_VSCCSID	DS	F CCSID of variable length string
MQOD_SELECTIONSTRING_LENGTH	EQU	*- MQOD_SELECTIONSTRING
	ORG	MQOD_SELECTIONSTRING
MQOD_SELECTIONSTRING_AREA	DS	CL(MQOD_SELECTIONSTRING_LENGTH)
*		
MQOD_RESOBJECTSTRING	DS	F Resolved Long object name
MQOD_RESOBJECTSTRING_VSPTR	DS	F Address of variable length string
MQOD_RESOBJECTSTRING_VSOFFSET	DS	F Offset of variable length string
MQOD_RESOBJECTSTRING_VSBUFSize	DS	F size of buffer
MQOD_RESOBJECTSTRING_VSLength	DS	F Length of variable length string

```

MQOD_RESOBJECTSTRING_VSCCSID DS F CCSID of variable length string
MQOD_RESOBJECTSTRING_LENGTH EQU *- MQOD_RESOBJECTSTRING
                                ORG MQOD_RESOBJECTSTRING
MQOD_RESOBJECTSTRING_AREA DS CL(MQOD_RESOBJECTSTRING_LENGTH)
MQOD_RESOLVEDTYPE DS F Alias queue object resolved type
*
MQOD_LENGTH EQU *-MQOD
                                ORG MQOD
MQOD_AREA DS CL(MQOD_LENGTH)

```

MQOD の Visual Basic 宣言

```

Type MQOD
  StrucId As String*4 'Structure identifier'
  Version As Long 'Structure version number'
  ObjectType As Long 'Object type'
  ObjectName As String*48 'Object name'
  ObjectQMgrName As String*48 'Object queue manager name'
  DynamicQName As String*48 'Dynamic queue name'
  AlternateUserId As String*12 'Alternate user identifier'
  RecsPresent As Long 'Number of object records present'
  KnownDestCount As Long 'Number of local queues opened'
                                'successfully'
  UnknownDestCount As Long 'Number of remote queues opened'
                                'successfully'
  InvalidDestCount As Long 'Number of queues that failed to'
                                'open'
  ObjectRecOffset As Long 'Offset of first object record from'
                                'start of MQOD'
  ResponseRecOffset As Long 'Offset of first response record'
                                'from start of MQOD'
  ObjectRecPtr As MQPTR 'Address of first object record'
  ResponseRecPtr As MQPTR 'Address of first response record'
  AlternateSecurityId As MQBYTE40 'Alternate security identifier'
  ResolvedQName As String*48 'Resolved queue name'
  ResolvedQMgrName As String*48 'Resolved queue manager name'
End Type

```

MQOD のフィールド詳細

MQOD オブジェクト記述子構造体は、名前によってオブジェクトを指定するために使用されます。この構造体は、MQOPEN および MQPUT1 呼び出しの入出力パラメーターです。

フィールド

注：次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

フィールド名と説明	定数の名前	定数の初期値（存在する場合）
<u>StrucId</u> (構造識別子)	MQOD_STRUC_ID	'OD-1'
<u>バージョン</u> (構造バージョン番号)	MQOD_VERSION_1	1
<u>ObjectType</u> (オブジェクトタイプ)	MQOT_Q	1
<u>ObjectName</u> (オブジェクト名)	なし	ヌル・ストリングまたは ブランク
<u>ObjectQMgrName</u> (オブジェクトキューマネージャ名)	なし	ヌル・ストリングまたは ブランク
<u>DynamicQName</u> (動的キュー名)	なし	'CSQ.*' ...の場合 z/OS ; 'AMQ.*' それ以外の 場合は
<u>AlternateUserId</u> (代替ユーザー識別子)	なし	ヌル・ストリングまたは ブランク
注：Version が MQOD_VERSION_2 より小さい場合は、残りのフィールドは無視されます。		

フィールド名と説明	定数の名前	定数の初期値（存在する場合）
<u>RecsPresent</u> (存在するオブジェクトレコードの数)	なし	0
<u>KnownDestCount</u> (正常に開かれたローカルキューの数)	なし	0
<u>UnknownDestCount</u> (正常に開かれたリモートキューの数)	なし	0
<u>InvalidDestCount</u> (開けられなかったキューの数)	なし	0
<u>ObjectRecOffset</u> (MQOD の先頭からの最初のオブジェクトレコードのオフセット)	なし	0
<u>ResponseRecOffset</u> (MQOD の先頭からの最初の応答レコードのオフセット)	なし	0
<u>ObjectRecPtr</u> (最初のオブジェクトレコードのアドレス)	なし	ヌル・ポインターまたはヌル・バイト
<u>ResponseRecPtr</u> (最初の応答レコードのアドレス)	なし	ヌル・ポインターまたはヌル・バイト
注: <i>Version</i> が MQOD_VERSION_3 より小さい場合、残りのフィールドは無視されます。		
<u>AlternateSecurityId</u> (代替セキュリティ識別子)	MQSID_NONE	Null
<u>ResolvedQName</u> (解決済みキュー名)	なし	ヌル・ストリングまたはブランク
<u>ResolvedQMgrName</u> (解決済みキューマネージャ名)	なし	ヌル・ストリングまたはブランク
注: <i>Version</i> が MQOD_VERSION_4 より小さい場合は、残りのフィールドは無視されます。		
<u>ObjectString</u> (長いオブジェクト名)	MQCHARV_DEFAULT	MQCHARV で定義されているとおり
<u>SelectionString</u> (選択文字列)	MQCHARV_DEFAULT	MQCHARV で定義されているとおり
<u>ResObjectString</u> (解決済み長いオブジェクト名)	MQCHARV_DEFAULT	MQCHARV で定義されているとおり
<u>ResolvedType</u> (解決済みオブジェクトタイプ)	MQOT_NONE	0
注: 1. 記号-は、単一のブランク文字を表します。 2. ヌル・ストリングまたはブランクの値は、C 言語ではヌル・ストリングを表し、他のプログラミング言語ではブランク文字を表します。 3. C プログラミング言語では、マクロ変数 MQOD_DEFAULT には、表にリストされている値が含まれています。この変数を以下の方法で使用すると、構造体のフィールドに初期値を設定できます。 <pre>MQOD MyOD = {MQOD_DEFAULT};</pre>		

StrucId (MQCHAR4)

これはオブジェクト記述子構造体の構造体識別子です。常に入力フィールドです。その値は MQOD_STRUC_ID です。

値は次のものでなければなりません。

MQOD_STRUC_ID

オブジェクト記述子構造体の識別子。

C プログラミング言語では、定数 MQOD_STRUC_ID_ARRAY も定義されている。これは MQOD_STRUC_ID と同じ値を持ちますが、文字列ではなく文字の配列です。

Version (MQLONG)

これは構造体のバージョン番号で、値は以下のいずれかでなければなりません。

MQOD_VERSION_1

バージョン 1 のオブジェクト記述子構造体。

MQOD_VERSION_2

バージョン 2 のオブジェクト記述子構造体。

MQOD_VERSION_3

バージョン 3 のオブジェクト記述子構造体。

MQOD_VERSION_4

バージョン 4 のオブジェクト記述子構造体。

すべての IBM MQ V7.0 環境ですべてのバージョンがサポートされます。

これより新しいバージョンの構造体にのみ存在するフィールドは、そのフィールドの説明にその旨記載されています。以下の定数は、現行バージョンのバージョン番号を指定しています。

MQOD_CURRENT_VERSION

現行バージョンのオブジェクト記述子構造体。

これは常に入力フィールドです。フィールドの初期値は、MQOD_VERSION_1 です。

ObjectType (MQLONG)

オブジェクト記述子で名前が付けられるオブジェクトのタイプ。指定可能な値は以下のとおりです。

MQOT_CLNTCONN_CHANNEL

クライアント接続チャネル。オブジェクトの名前は *ObjectName* フィールドにあります。

MQOT_Q

列。オブジェクトの名前は *ObjectName* フィールドにあります。

MQOT_NAMELIST

名前リスト。オブジェクトの名前は *ObjectName* フィールドにあります。

MQOT_PROCESS

プロセス定義。オブジェクトの名前は *ObjectName* フィールドにあります。

MQOT_Q_MGR

キュー・マネージャー。オブジェクトの名前は *ObjectName* フィールドにあります。

MQOT_TOPIC

トピック。トピック名のフルネームは、*ObjectName* および *ObjectString* の 2 種類のフィールドからビルドできます。

これら 2 つのフィールドの使用法の詳細については、[「トピック文字列の結合」](#) を参照してください。

これは常に入力フィールドです。フィールドの初期値は、MQOT_Q です。

ObjectName (MQCHAR48)

これは、によって識別されるキュー・マネージャ上で定義 *ObjectQMGrName* されたオブジェクトのローカル名です。この名前には、以下に示す文字を使用できます。

- 英大文字 (A から Z まで)

- 英小文字 (a から z まで)
- 数字 (0 から 9 まで)
- ピリオド (.)、スラッシュ (/)、下線 (_)、パーセント (%)

名前の先頭をブランクにしたり、名前にブランクを埋め込んだりすることはできませんが、名前の後にブランクを入れることはできます。ヌル文字を使用して、名前の中における有効なデータの末尾を示します。ヌル文字とそれに続く文字はすべてブランクとして扱われます。以下に示す制約事項は、それぞれ明記している環境に適用されます。

- EBCDIC カタカナを使用するシステムでは、小文字を使用できません。
- On z/OS:
 - 先頭または末尾に下線がある名前は使用しないでください。これらの名前は、操作パネルや制御パネルで処理できません。
 - パーセント文字は、RACF では特別な意味があります。RACF を外部セキュリティ・マネージャーとして使用する場合、名前にはパーセントを含めないでください。パーセントが含まれていると、RACF 総称プロファイルを使用したときに、それらの名前はどのセキュリティ検査にも組み込まれません。
- IBM i で英小文字、スラッシュ、パーセントの各文字が含まれている名前をコマンドに指定する場合は、それを引用符で囲む必要があります。構造体内のフィールドまたは呼び出しのパラメーターとして指定する名前には、引用符を使用してはなりません。

トピック名のフルネームは、*ObjectName* および *ObjectString* の 2 種類のフィールドからビルドできます。これらの 2 つのフィールドの使用の詳細については、「[トピック文字列の結合](#)」を参照してください。

以下に示す点は、記されているオブジェクトのタイプに適用されます。

- モデルキューの名前がである *ObjectName* 場合、キューマネージャはモデルキューの属性を持つ動的キューを作成し、フィールド *ObjectName* に作成されたキューの名前を返します。モデル・キューは MQOPEN 呼び出しでのみ指定されます。したがって MQPUT1 呼び出しでは無効です。
- *ObjectName* が、TARGTYPE(TOPIC) を持つ別名キューの名前である場合、セキュリティ検査はまず名前付きの別名キューに対して行われます。これは別名キューが使用されときの通常の動作です。セキュリティ検査が正常に完了すると、MQOPEN 呼び出しは続行され、MQOT_TOPIC に対して MQOPEN 呼び出しのように動作します。これには管理トピック・オブジェクトに対してセキュリティ検査を実行することも含まれます。
- *ObjectName* および *ObjectQMgrName* が、ローカル・キュー・マネージャーが所属するキュー共有グループが所有する共有キューを識別する場合、そのローカル・キュー・マネージャーで同じ名前のキュー定義を使用することはできません。そのような定義(ローカル・キュー、別名キュー、リモート・キュー、またはモデル・キュー)がある場合、呼び出しは理由コード MQRC_OBJECT_NOT_UNIQUE を伴って失敗します。
- オープンされているオブジェクトが配布リストの場合(すなわち、*RecsPresent* があり、ゼロより大きい場合)、*ObjectName* はブランクまたはヌル・ストリングでなければなりません。この条件を満たさないと、この呼び出しは失敗し、理由コード MQRC_OBJECT_NAME_ERROR が戻ります。
- *ObjectType* が MQOT_Q_MGR の場合は、特別な規則が適用されます。つまり、最初のヌル文字またはフィールドの終わりまで、名前全体がブランクでなければなりません。

ObjectName がモデル・キューの名前である場合は、MQOPEN 呼び出しの入出力フィールドです。それ以外の場合は、入力専用フィールドです。このフィールドの長さは MQ_Q_NAME_LENGTH によって指定されます。このフィールドの初期値は、C 言語ではヌル・ストリングであり、他のプログラミング言語では 48 桁のブランク文字です。

ObjectQMgrName (MQCHAR48)

これは、オブジェクト *ObjectName* が定義されているキューマネージャの名前です。この名前でも有効な文字は、*ObjectName* の場合と同じです(456 ページの『[ObjectName \(MQCHAR48\)](#)』を参照)。最初のヌル文字またはフィールドの終わりまで名前をすべてブランクにすると、アプリケーションが接続されているキュー・マネージャー(ローカル・キュー・マネージャー)を指定したと見なされます。

以下に示す点は、記されているオブジェクトのタイプに適用されます。

- *ObjectType* が MQOT_TOPIC、MQOT_NAMELIST、MQOT_PROCESS、または MQOT_Q_MGR である場合は、*ObjectQMgrName* はブランクか、またはローカル・キュー・マネージャーの名前でなければなりません。
- *ObjectName* がモデル・キューの名前である場合、キュー・マネージャーは、モデル・キューの属性をもつ動的キューを作成し、キューが作成されたキュー・マネージャーの名前を *ObjectQMgrName* フィールドに戻します。これは、ローカル・キュー・マネージャーの名前です。モデル・キューは MQOPEN 呼び出しでのみ指定されます。したがって MQPUT1 呼び出しでは無効です。
- *ObjectName* がクラスター・キューの名前であり、*ObjectQMgrName* がブランクである場合、MQOPEN 呼び出しが戻したキュー・ハンドルを使用して送信されるメッセージの宛先は、次のようにキュー・マネージャーによって (または、クラスター・ワークロード出口がインストールされている場合はそれによって) 選択されます。
 - MQOO_BIND_ON_OPEN が指定された場合、キュー・マネージャーは MQOPEN 呼び出しの処理時にクラスター・キューの特定のインスタンスを選択し、そのキュー・ハンドルを使用して送信されるすべてのメッセージは、そのインスタンスへ送信されます。
 - MQOO_BIND_NOT_FIXED が指定された場合、キュー・マネージャーはキュー・ハンドルを使用する連続した MQPUT 呼び出しのそれぞれにおいて、その宛先キューの (クラスター内の別のキュー・マネージャー上にある) 別のインスタンスを選択する場合があります。

アプリケーションからクラスター・キューの特定の インスタンス (つまり、クラスターの特定のキュー・マネージャー上にあるキュー・インスタンス) へメッセージを送信する必要がある場合は、アプリケーションで *ObjectQMgrName* フィールドにそのキュー・マネージャーの名前を指定しなければなりません。これにより、ローカル・キュー・マネージャーは指定された宛先キュー・マネージャーへメッセージを送信することを強制されます。

- 共有キューの名前がである *ObjectName* 場合は、ローカル・キュー・マネージャーが属するキュー共有グループが所有するもので、*ObjectQMgrName* には、キュー共有グループ名、ローカル・キュー・マネージャー名、または空白を指定することができます。これらの値のいずれかが指定されると、同じキューにメッセージが配置されます。

キュー共有グループは、z/OS でのみサポートされています。

- *ObjectName* が、リモート・キュー共有グループ (つまり、ローカル・キュー・マネージャーが所属しないキュー共有グループ) が所有する共有キューの名前である場合、*ObjectQMgrName* はそのキュー共有グループの名前でなければなりません。そのグループに属するキュー・マネージャーの名前を使用することもできますが、その特定のキュー・マネージャーが、キュー共有グループにメッセージが届いたときに使用不可能である場合、メッセージの到着が遅れる可能性があります。
- オープンされているオブジェクトが配布リストの場合 (すなわち、*RecsPresent* がゼロより大きい場合)、*ObjectQMgrName* はブランクまたはヌル・ストリングでなければなりません。この条件を満たさないと、この呼び出しは失敗し、理由コード MQRC_OBJECT_Q_MGR_NAME_ERROR が戻ります。

ObjectName がモデル・キューの名前である場合は、MQOPEN 呼び出しの入出力フィールドです。それ以外の場合は、入力専用フィールドです。このフィールドの長さは MQ_Q_MGR_NAME_LENGTH で指定します。このフィールドの初期値は、C 言語ではヌル・ストリングであり、他のプログラミング言語では 48 桁のブランク文字です。

DynamicQName (MQCHAR48)

これは、MQOPEN 呼び出しによって作成される動的キューの名前です。これが関係してくるのは *ObjectName* にモデル・キューが指定されている場合だけであり、それ以外の場合はすべて *DynamicQName* は無視されます。

この名前で有効な文字は、*ObjectName* の場合と同じです。ただし、アスタリスクも有効です。*ObjectName* がモデル・キューの名前である場合は、ブランクである名前 (あるいは、最初のヌル文字の前にブランクしかない名前) は無効です。

名前の最後の空白でない文字がアスタリスク (*) の場合、キューマネージャはアスタリスクを、ローカルキューマネージャ内で生成されるキュー名が一意であることを保証する文字列に置換します。これを保証

できるだけ文字数を確保するためには、アスタリスクの位置がカラム 1 から 33 までの範囲でなければなりません。アスタリスクの後に、ブランクまたはヌル文字以外の文字があってはなりません。

名前がキュー・マネージャーによって生成された文字だけで構成される場合は、最初の文字にアスタリスクを指定できます。

z/OS では、先頭文字にアスタリスクを置いた名前を使用しないでください。これは、フルネームが自動的に生成されたキューに対してはセキュリティチェックが実行されないためです。

これは入力フィールドです。このフィールドの長さは MQ_Q_NAME_LENGTH によって指定されます。フィールドの初期値は、環境により決まります。

- z/OS では、値は 'CSQ.*' です。
- 他のプラットフォームでは、値は 'AMQ.*' です。

この値は C ではヌル終了ストリングであり、他のプログラミング言語ではブランクが埋め込まれたストリングです。

AlternateUserId (MQCHAR12)

MQOO_ALTERNATE_USER_AUTHORITY を MQOPEN 呼び出しで指定した場合、または MQPMO_ALTERNATE_USER_AUTHORITY を MQPUT1 呼び出しで指定した場合、このフィールドには、代替ユーザー ID が入っています。代替ユーザー ID は、アプリケーションが現在実行されているユーザー ID の代わりに、オープンの特権を検査するために使用されるものです。ただし、検査によっては、現行のユーザー ID を使って実行されます (例えば、コンテキストの検査など)。

MQOO_ALTERNATE_USER_AUTHORITY または MQPMO_ALTERNATE_USER_AUTHORITY が指定されていて、このフィールドが最初のヌル文字またはフィールドの終わりまで全体が完全にブランクになっている場合、オープンが成功するのは、指定されたオプションでこのオブジェクトをオープンするのにユーザー許可が必要でない場合だけです。

MQOO_ALTERNATE_USER_AUTHORITY あるいは MQPMO_ALTERNATE_USER_AUTHORITY のどちらも指定されていない場合は、このフィールドは無視されます。

以下の環境では、次のような違いがあります。

- z/OS では、オープンのための権限を検査するために、*AlternateUserId* の最初の 8 文字だけが使用されます。ただし、現行のユーザー ID に、この特定の代替ユーザー ID を指定する権限があることが必要です。この検査には、代替ユーザー ID の 12 文字がすべて使用されます。ユーザー ID に指定できる文字は、外部セキュリティ管理プログラムにより許可されています。

キューに *AlternateUserId* が指定されている場合は、その後メッセージが書き込まれるときに、キュー・マネージャーがその値を使用できます。MQPUT または MQPUT1 呼び出しで指定された MQPMO_*_CONTEXT オプションによりキューマネージャが識別コンテキスト情報を生成する場合、キューマネージャはメッセージの MQMD 内のフィールド *UserIdentifierAlternateUserId* に、現在のユーザー識別子の代わりにを配置する。

- 他の環境では、*AlternateUserId* は開こうとしているオブジェクトに対するアクセス制御チェックにのみ使用される。オブジェクトがキューである場合、そのキューハンドルを使用して送信された *AlternateUserId* メッセージの MQMD 内のフィールド *UserIdentifier* の内容には影響しません。

これは入力フィールドです。このフィールドの長さは MQ_USER_ID_LENGTH によって指定されます。このフィールドの初期値は、C 言語ではヌル・ストリングですが、その他のプログラミング言語では 12 個のブランク文字です。

RecsPresent (MQLONG)

これは、アプリケーションが提供した MQOR オブジェクト・レコードの数です。この数がゼロより大きい場合は配布リストがオープンされており、*RecsPresent* がリスト中の宛先キューの数になっていることを示しています。配布リストに含めることができるのは 1 つの宛先のみです。

RecsPresent の値はゼロ未満であってはなりません。また、この値がゼロより大きい場合、*ObjectType* は MQOT_Q でなければなりません。これらの条件を満たさないと、その呼び出しは失敗し、理由コード MQRC_RECS_PRESENT_ERROR が戻ります。

z/OS では、このフィールドをゼロにする必要があります。

これは入力フィールドです。このフィールドの初期値は 0 です。このフィールドは、*Version* が MQOD_VERSION_2 より前の場合には無視されます。

KnownDestCount (MQLONG)

これは配布リスト中のキューの数で、ローカル・キューに解決し、オープンに成功したキューの数です。この数にはリモート・キューに解決するキューの数は含まれません。ローカル伝送キューを使用して最初にメッセージを格納する場合でも同様です。このフィールドは、配布リストにはない 1 つのキューをオープンするときも設定されます。

これは出力フィールドです。このフィールドの初期値は 0 です。このフィールドは、MQOD_VERSION_1*Version* 未満の場合に無視されます。

UnknownDestCount (MQLONG)

これは配布リスト中のキューの数で、リモート・キューに解決し、オープンに成功したキューの数です。このフィールドは、配布リストにはない 1 つのキューをオープンするときも設定されます。

これは出力フィールドです。このフィールドの初期値は 0 です。このフィールドは、MQOD_VERSION_1*Version* 未満の場合に無視されます。

InvalidDestCount (MQLONG)

これは配布リスト中のキューの数で、オープンに失敗したキューの数です。このフィールドは、配布リストにはない 1 つのキューをオープンするときも設定されます。

注：このフィールドは、MQOPEN 呼び出しまたは MQPUT1 呼び出しの **CompCode** パラメーターが MQCC_OK または MQCC_WARNING の場合に限り設定されます。**CompCode** パラメーター MQCC_FAILED の場合は、設定されません。

これは出力フィールドです。このフィールドの初期値は 0 です。このフィールドは、MQOD_VERSION_1*Version* 未満の場合に無視されます。

ObjectRecOffset (MQLONG)

これは、MQOD 構造体の先頭からの MQOR オブジェクト・レコードのオフセットをバイト数で表したものです。オフセットの値は、正負どちらの値にもなります。*ObjectRecOffset* は、配布リストがオープン中の場合にのみ使用されます。*RecsPresent* がゼロの場合、このフィールドは無視されます。

配布リストがオープン中の場合、1 つ以上の MQOR オブジェクト・レコードは、配布リスト中の宛先キューの名前を指定するために提供されなければなりません。これは次の 2 つのうちいずれかの方法で行うことができます。

- オフセット・フィールド *ObjectRecOffset* を使用する。

この場合、アプリケーションは (必要なだけ多くの配列エレメントのある) MQOR レコードの配列で始まる MQOD を含む、独自の構造体を宣言する必要があります。さらに *ObjectRecOffset* を、MQOD の先頭からその配列で最初のエレメントのオフセットに設定する必要があります。このオフセットが正しいこと、および値が MQLONG 内に収まることを確認する必要があります (最も制限の多いプログラミング言語は COBOL で、有効範囲は -999 999 999 から +999 999 999 です)。

ポインターのデータ・タイプをサポートしていないプログラミング言語や、他の環境に移植できない方式のポインター・データ・タイプをインプリメントしているプログラミング言語 (COBOL プログラミング言語など) の場合には、*ObjectRecOffset* を使用してください。

- ポインター・フィールド *ObjectRecPtr* を使用する。

この場合は、アプリケーションで MQOD 構造体とは別に MQOR 構造体の配列を宣言でき、*ObjectRecPtr* に配列のアドレスを設定できます。

他の環境へ移植できる形式のポインター・データ・タイプをサポートするプログラミング言語 (例えば C プログラミング言語など) には、*ObjectRecPtr* を使用してください。

どの方法を選ぶにしても、*ObjectRecOffset* および *ObjectRecPtr* のいずれかを使用します。両方ともゼロの場合、または両方ともゼロでない場合、呼び出しは失敗し、理由コード MQRC_OBJECT_RECORDS_ERROR が戻ります。

これは入力フィールドです。このフィールドの初期値は 0 です。このフィールドは、*Version* が MQOD_VERSION_2 より前の場合には無視されます。

ResponseRecOffset (MQLONG)

これは、MQOD 構造体の先頭から最初の MQRR 応答レコードのオフセットをバイト数で表したものです。オフセットの値は、正負どちらの値にもなります。*ResponseRecOffset* は、配布リストがオープン中の場合にのみ使用されます。*RecsPresent* がゼロの場合、このフィールドは無視されます。

配布リストがオープン中の場合、1 つ以上の MQRR 応答レコードの配列を提供することができます。これは、オープンに失敗したキューを特定するため (この場合 MQRR の *CompCode* フィールドに入ります)、および失敗した理由をそれぞれ特定するためです (この場合、MQRR の *Reason* フィールドに入ります)。データは、応答レコードの配列に、キューの名前がオブジェクト・レコードの配列に発生したのと同じ順番で戻ります。キュー・マネージャーは、呼び出しの結果が混在したときだけ応答レコードを設定します。つまり、オープンに成功したキューもあれば失敗したキューもある場合や、全部失敗したが理由が異なる場合などです。呼び出しから理由コード MQRC_MULTIPLE_REASONS が出るのはこの場合です。すべてのキューに同じ理由コードが該当する場合は、その理由コードが MQOPEN または MQPUT1 呼び出しの **Reason** パラメーター内に戻され、応答レコードは設定されません。応答レコードはオプションですが、指定する場合はこれらの *RecsPresent* が必要です。

応答レコードは、オブジェクトレコードと同様の方法で提供できます。つまり、`` 内のオフセット *ResponseRecPtr* を指定するか、`` 内のアドレス *ResponseRecOffset* を指定します。詳細な手順については、`` を [ObjectRecOffset](#) 参照してください。ただし、と *ResponseRecOffset* のうち 1 *ResponseRecPtr* つだけを使用できます。両方が 0 以外の場合、呼び出しは理由コード MQRC_RESPONSE_RECORDS_ERROR で失敗します。

MQPUT1 呼び出しの場合、これらの応答レコードはエラーについての情報を返すのに使用されます。このエラーは、キューがオープンされる場合や、メッセージが配布リスト中のキューに送られる場合に発生します。あるキューに対するオープン操作から出る完了コードおよび理由コードは、そのキューに対する PUT 操作から出るコードに置き換えられます。ただし、これは前者から戻った完了コードが MQCC_OK または MQCC_WARNING であった場合に限られます。

これは入力フィールドです。このフィールドの初期値は 0 です。このフィールドは、*Version* が MQOD_VERSION_2 より前の場合には無視されます。

ObjectRecPtr (MQPTR)

これは、最初の MQOR オブジェクト・レコードのアドレスです。*ObjectRecPtr* は、配布リストがオープン中の場合にのみ使用されます。*RecsPresent* がゼロの場合、このフィールドは無視されます。

オブジェクトレコードの指定には *ObjectRecOffset*、または *ObjectRecPtr* のいずれかを使用できますが、両方同時に使用することはできません。フィールド *ObjectRecOffset* の説明については [ObjectRecOffset](#) 参照してください。*ObjectRecPtr* を使用しない場合、ヌル・ポインターまたはヌル・バイトを設定します。

これは入力フィールドです。このフィールドの初期値は、ポインターをサポートするプログラミング言語のヌル・ポインターです。それ以外の場合は、すべてヌルのバイトのストリングです。このフィールドは、*Version* が MQOD_VERSION_2 より前の場合には無視されます。

注: プログラミング言語がそのポインターのデータ・タイプをサポートしないプラットフォームでは、このフィールドは初期値がすべてヌルのバイト・ストリングである、適当な長さのバイト・ストリングとして宣言されます。

ResponseRecPtr (MQPTR)

これは、最初の MQRR 応答レコードのアドレスです。 *ResponseRecPtr* は、配布リストがオープン中の場合にのみ使用されます。 *RecsPresent* がゼロの場合、このフィールドは無視されます。

応答レコードを指定するには *ResponseRecOffset*、または *ResponseRecPtr* のいずれかを使用してください。両方を使用しないでください。詳細はを参照してください [ResponseRecOffset](#)。
ResponseRecPtr を使用しない場合、ヌル・ポインターまたはヌル・バイトを設定します。

これは入力フィールドです。このフィールドの初期値は、ポインターをサポートするプログラミング言語のヌル・ポインターです。それ以外の場合は、すべてヌルのバイトのストリングです。このフィールドは、*Version* が MQOD_VERSION_2 より前の場合には無視されます。

注: プログラミング言語がそのポインターのデータ・タイプをサポートしないプラットフォームでは、このフィールドは初期値がすべてヌルのバイト・ストリングである、適当な長さのバイト・ストリングとして宣言されます。

AlternateSecurityId (MQBYTE40)

これは、適切な許可検査を実行できるようにするために、*AlternateUserId* と共に許可サービスに渡されるセキュリティ ID です。 *AlternateSecurityId* は、次の場合のみ使用されます。

- MQOPEN 呼び出しで MQOO_ALTERNATE_USER_AUTHORITY が指定されている
- MQPUT1 呼び出しで MQPMO_ALTERNATE_USER_AUTHORITY が指定されている

フィールド *AlternateUserId* は、最初のヌル文字またはフィールドの終わりまで完全に空白ではない。

では Windows、を供給するために *AlternateSecurityId* 使用でき、*AlternateUserId* Windows これはを一意に識別するセキュリティ識別子 (SID) である。ユーザーの SID は、API `LookupAccountName()` Windows 呼び出しを使用してシステム Windows から取得できます。

z/OS では、このフィールドは無視されます。

AlternateSecurityId フィールドは、以下の構造体を持っています。

- 最初のバイトは、後続の有効データの長さを示す 2 進整数です。値には、このバイト自体は含まれません。セキュリティ ID がいない場合、長さはゼロになります。
- 2 番目のバイトは、存在するセキュリティ ID のタイプを示します。可能な値は次のとおりです。

MQSIDT_NT_SECURITY_ID

Windows セキュリティ ID。

MQSIDT_NONE

セキュリティ ID なし。

- 3 番目のバイトから、最初のバイトで定義された長さまでは、セキュリティ ID 自体が含まれています。
- フィールドの残りのバイトは、2 進ゼロに設定されます。

次の特殊値を使用することができます。

MQSID_NONE

セキュリティ ID が指定されていない。

値は、フィールドの長さについては 2 進ゼロです。

C プログラミング言語では、定数 `MQSID_NONE_ARRAY` も定義されます。これは `MQSID_NONE` と同じ値ですが、ストリングではなく文字の配列です。

これは入力フィールドです。このフィールドの長さは `MQ_SECURITY_ID_LENGTH` によって指定されます。このフィールドの初期値は、`MQSID_NONE` です。 *Version* が MQOD_VERSION_3 より小さい場合、このフィールドは無視されます。

ResolvedQName (MQCHAR48)

これは、ローカル・キュー・マネージャーが名前を解決した後の宛先キューの名前です。戻される名前は、*ResolvedQMgrName* によって識別されるキュー・マネージャー上に存在するキューの名前です。

非ブランク値は、オブジェクトがブラウズ、入力、または出力 (あるいはこれらの組み合わせ) を目的としてオープンされた単一のキューである場合にだけ戻されます。オープンされたオブジェクトが以下のいずれかである場合、*ResolvedQName* はブランクに設定されます。

- キューでない
- キューだが、オープンの目的がブラウズ、入力、および出力のいずれでもない
- 配布リスト
- トピック・オブジェクトを参照する別名キュー (代わりに [ResObjectString](#) を参照してください)。
- トピック・オブジェクトを解決する別名キュー

これは出力フィールドです。このフィールドの長さは `MQ_Q_NAME_LENGTH` によって指定されます。このフィールドの初期値は、C 言語ではヌル・ストリングであり、他のプログラミング言語では 48 桁のブランク文字です。*Version* が `MQOD_VERSION_3` より小さい場合、このフィールドは無視されます。

ResolvedQMgrName (MQCHAR48)

これは、ローカル・キュー・マネージャーによって名前が解決された後の宛先キュー・マネージャーの名前です。戻される名前は、*ResolvedQName* によって識別されるキューを所有するキュー・マネージャーの名前です。*ResolvedQMgrName* は、ローカル・キュー・マネージャーの名前にすることができます。

ResolvedQName が、ローカル・キュー・マネージャーが属するキュー共有グループが所有する共有キューである場合、*ResolvedQMgrName* はそのキュー共有グループの名前です。キューが他のキュー共有グループによって所有されている場合、*ResolvedQName* は、キュー共有グループの名前またはキュー共有グループのメンバーであるキュー・マネージャーの名前にすることができます (返される値の性質は、ローカル・キュー・マネージャーに存在するキュー定義によって異なります)。

非ブランク値は、オブジェクトがブラウズ、入力、または出力 (あるいはこれらの組み合わせ) を目的としてオープンされた単一のキューである場合にだけ戻されます。オープンされたオブジェクトが以下のいずれかである場合、*ResolvedQMgrName* はブランクに設定されます。

- キューでない
- キューだが、オープンの目的がブラウズ、入力、および出力のいずれでもない
- `MQOO_BIND_NOT_FIXED` が指定されたクラスター・キュー (または **DefBind** キュー属性の値が `MQBND_BIND_NOT_FIXED` の場合は `MQOO_BIND_AS_Q_DEF` が有効なクラスター・キュー)
- 配布リスト

これは出力フィールドです。このフィールドの長さは `MQ_Q_NAME_LENGTH` によって指定されます。このフィールドの初期値は、C 言語ではヌル・ストリングであり、他のプログラミング言語では 48 桁のブランク文字です。*Version* が `MQOD_VERSION_3` より小さい場合、このフィールドは無視されます。

ObjectString (MQCHARV)

ObjectString フィールドは長いオブジェクト名を指定します。

これは、使用される長いオブジェクト名を指定します。このフィールドは、*ObjectType* の特定の値でのみ参照され、それ以外のすべての値では無視されます。このフィールドが使用されることを示す値について詳しくは、*ObjectType* の説明を参照してください。

`MQCHARV` 構造体の使い方に関する説明に従うと *ObjectString* の指定が正しくない場合、または最大長を超える場合、呼び出しは失敗し、理由コード `MQRC_OBJECT_STRING_ERROR` が戻されます。

これは入力フィールドです。この構造体のフィールドの初期値は、`MQCHARV` 構造体のものと同じです。

トピック名のフルネームは、*ObjectName* および *ObjectString* の 2 種類のフィールドからビルドできます。これらの 2 つのフィールドの使用方法的詳細については、「[トピック文字列の結合](#)」を参照してください。

SelectionString (MQCHARV)

このストリングを使用して、キューからメッセージを取り出す際に使用される選択基準を提供します。

以下の場合には、*SelectionString* は指定しないでください。

- *ObjectType* が MQOT_Q でない場合
- オープン対象のキューを、MQOO_BROWSE または MQOO_INPUT_* オプションの 1 つを使用してオープンしようとしていない場合

これらの場合に *SelectionString* を提供すると、この呼び出しは失敗し、理由コード MQRC_SELECTOR_INVALID_FOR_TYPE が戻ります。

構造 240 ページの『MQCHARV - 可変長ストリング』体の使用方法の説明に従ってが正しく指定されていない *SelectionString* 場合、または最大長を超えた場合、呼び出しは理由コード MQRC_SELECTION_STRING_ERROR で失敗します。*SelectionString* の最大長は MQ_SELECTOR_LENGTH です。

SelectionString の使用法については、[セレクトター](#)で説明しています。

ResObjectString (MQCHARV)

ResObjectString フィールドは、*ObjectName* フィールドで指定される名前をキュー・マネージャーが解決した後の長いオブジェクト名です。

このフィールドは、トピック、およびトピック・オブジェクトを参照するキュー別名についてのみ戻されます。

ObjectString に長いオブジェクト名が指定され、*ObjectName* に何も指定されない場合、このフィールドに返される値は、*ObjectString* で指定されたものと同じになります。

このフィールドを省略する (つまり、ResObjectString.VSBufSize がゼロの) 場合は、*ResObjectString* は戻されませんが、長さが ResObjectString.VSLength 中に戻されます。

バッファ長 (ResObjectStrng.VSBufSize で指定される) が *ResObjectString* の全長未満の場合は、ストリングは切り捨てられ、提供されているバッファに収まる数だけ右端文字が戻されます。

MQCHARV 構造体の使い方に関する説明に従うと *ResObjectString* の指定が正しくない場合、または最大長を超える場合、呼び出しは失敗し、理由コード MQRC_RES_OBJECT_STRING_ERROR が戻されます。

ResolvedType (MQLONG)

開いている解決済み (ベース) オブジェクトのタイプ。

指定できる値は以下のとおりです。

MQOT_Q

解決済みオブジェクトはキューです。直接キューが開かれているとき、またはキューを位置指定している別名キューが開かれているときは、この値が適用されます。

MQOT_TOPIC

解決済みオブジェクトはトピックです。直接トピックが開かれているとき、またはトピックを位置指定している別名キューが開かれているときは、この値が適用されます。

MQOT_NONE

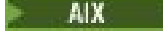
解決済みタイプはキューでもトピックでもありません。

MQOR - オブジェクト・レコード

MQOR 構造を使用して、単一の宛先キューのキュー名とキューマネージャ名を指定します。MQOR は、MQOPEN 呼び出しおよび MQPUT1 呼び出しのための入力構造体です。

可用性

MQOR 構造は、以下のプラットフォームで利用可能です：

-  AIX
-  IBM i
-  Linux
-  Windows

および、これらのシステムに接続された IBM MQ MQI clients。

文字セットとエンコード

MQOR 内のデータは、キュー・マネージャ **CodedCharSetId** 属性によって指定される文字セットおよびローカル・キュー・マネージャのエンコーディング (MQENC_NATIVE によって指定される) でなければなりません。ただし、アプリケーションが MQ MQI クライアントとして実行されている場合、構造体はクライアントの文字セットとエンコードに従っている必要があります。

使用法

MQOPEN 呼び出し時にこれらの構造体の配列を指定することで、キューのリストを開くことができます。このリストは配布リストと呼ばれます。各メッセージの書き込みは、その MQOPEN 呼び出しによって戻ったキュー・ハンドルを使用して、リスト中の各キューに置かれます。これは、そのキューのオープンが成功することが前提となります。

フィールド

注：次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 525. MQOR における MQOR のフィールド		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
<u>ObjectName</u> (オブジェクト名)	なし	ヌル・ストリングまたはブランク
<u>ObjectQMgrName</u> (オブジェクトキューマネージャ名)	なし	ヌル・ストリングまたはブランク
注： 1. ヌル・ストリングまたはブランクの値は、C 言語ではヌル・ストリングを表し、他のプログラミング言語ではブランク文字を表します。 2. C プログラミング言語では、マクロ変数 MQOR_DEFAULT には、表にリストされている値が含まれています。この変数を以下の方法で使用すると、構造体のフィールドに初期値を設定できます。 <pre>MQOR MyOR = {MQOR_DEFAULT};</pre>		

言語ごとの宣言

MQOR の C 宣言

```
typedef struct tagMQOR MQOR;
struct tagMQOR {
    MQCHAR48  ObjectName;          /* Object name */
    MQCHAR48  ObjectQMgrName;     /* Object queue manager name */
};
```

MQOR の COBOL 宣言

```
**      MQOR structure
10 MQOR.
**      Object name
15 MQOR-OBJECTNAME      PIC X(48).
**      Object queue manager name
15 MQOR-OBJECTQMGRNAME PIC X(48).
```

MQOR の PL/I 宣言

```
dcl
1 MQOR based,
3 ObjectName      char(48), /* Object name */
3 ObjectQMgrName char(48); /* Object queue manager name */
```

MQOR の Visual Basic 宣言

```
Type MQOR
    ObjectName      As String*48 'Object name'
    ObjectQMgrName As String*48 'Object queue manager name'
End Type
```

MQOR のフィールド詳細

MQOR オブジェクトレコード構造を使用して、単一の宛先キューのキュー名とキューマネージャ名を指定します。MQOR は、MQOPEN 呼び出しおよび MQPUT1 呼び出しのための入力構造体です。

フィールド

注: 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 526. MQOR における MQOR のフィールド		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
ObjectName (オブジェクト名)	なし	ヌル・ストリングまたはブランク
ObjectQMgrName (オブジェクトキューマネージャ名)	なし	ヌル・ストリングまたはブランク
注:		
1. ヌル・ストリングまたはブランクの値は、C 言語ではヌル・ストリングを表し、他のプログラミング言語ではブランク文字を表します。		
2. C プログラミング言語では、マクロ変数 MQOR_DEFAULT には、表にリストされている値が含まれています。この変数を以下の方法で使用すると、構造体のフィールドに初期値を設定できます。		
MQOR MyOR = {MQOR_DEFAULT};		

ObjectName (MQCHAR48)

これは ObjectName MQOD 構造体のフィールドと同じですが、以下の点が異なります：

- キューの名前でなければならない。
- モデル・キューの名前であってはならない。

これは常に入力フィールドです。このフィールドの初期値は、C 言語ではヌル・ストリングであり、他のプログラミング言語では 48 桁のブランク文字です。

ObjectQMgrName (MQCHAR48)

これは [MQOD ObjectQMgrName](#) 構造体のフィールドと同じです。

これは常に入力フィールドです。このフィールドの初期値は、C 言語ではヌル・ストリングであり、他のプログラミング言語では 48 桁のブランク文字です。

MQPD - プロパティ記述子

この **MQPD** 構造体は、プロパティの属性を定義するために使用されます。この構造は、MQSETMP 呼び出しの入出力パラメーターおよび MQINQMP 呼び出しの出力パラメーターです。

可用性

この **MQPD** 構造は、以下のプラットフォームで利用可能です：

-  AIX
-  IBM i
-  Linux
-  Windows
-  z/OS

および、IBM MQ MQI clients。

文字セットとエンコード

入力データは、アプリケーションの文字セットおよびアプリケーション **MQPD** のエンコーディング (**MQENC_NATIVE**) でなければならない。

フィールド

注：次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 527. MQPD のフィールド		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
StrucId (構造識別子)	MQPD_STRUC_ID	'PD'
バージョン (構造バージョン番号)	MQPD_VERSION_1	1
オプション (オプション)	MQPD_NONE	0
サポート (メッセージプロパティに必要なサポート)	MQPD_SUPPORT_OPTIONAL	0
コンテキスト (プロパティが属するメッセージコンテキスト)	MQPD_NO_CONTEXT	0
CopyOptions (コピーオプションがどのプロパティに属するか)	MQCOPY_DEFAULT	0

表 527. MQPD のフィールド (続き)

フィールド名と説明	定数の名前	定数の初期値（存在する場合）
注： 1. C プログラミング言語では、マクロ変数 MQPD_DEFAULT に表にリストされている値が設定されています。この変数を以下の方法で使用すると、構造体のフィールドに初期値を設定できます。 <pre>MQPD MyPD = {MQPD_DEFAULT};</pre>		

言語ごとの宣言

MQPD の C 宣言

```
typedef struct tagMQPD MQPD;
struct tagMQPD {
    MQCHAR4  StrucId;      /* Structure identifier */
    MQLONG   Version;     /* Structure version number */
    MQLONG   Options;     /* Options that control the action of
                           MQSETMP and MQINQMP */
    MQLONG   Support;     /* Property support option */
    MQLONG   Context;     /* Property context */
    MQLONG   CopyOptions; /* Property copy options */
};
```

MQPD の COBOL 宣言

```
** MQPD structure
10 MQPD.
**   Structure identifier
15 MQPD-STRUCID PIC X(4).
**   Structure version number
15 MQPD-VERSION PIC S9(9) BINARY.
**   Options that control the action of MQSETMP and
**   MQINQMP
15 MQPD-OPTIONS PIC S9(9) BINARY.
**   Property support option
15 MQPD-SUPPORT PIC S9(9) BINARY.
**   Property context
15 MQPD-CONTEXT PIC S9(9) BINARY.
**   Property copy options
15 MQPD-COPYOPTIONS PIC S9(9) BINARY.
```

MQPD の PL/I 宣言

```
dcl
1 MQPD based,
3 StrucId      char(4),      /* Structure identifier */
3 Version      fixed bin(31), /* Structure version number */
3 Options      fixed bin(31), /* Options that control the action
                             of MQSETMP and MQINQMP */
3 Support      fixed bin(31), /* Property support option */
3 Context      fixed bin(31), /* Property context */
3 CopyOptions  fixed bin(31); /* Property copy options */
```

MQPD の高水準アセンブラ宣言

MQPD	DSECT	
MQPD_STRUCID	DS CL4	Structure identifier
MQPD_VERSION	DS F	Structure version number
MQPD_OPTIONS	DS F	Options that control the action of MQSETMP and MQINQMP
*		
MQPD_SUPPORT	DS F	Property support option
MQPD_CONTEXT	DS F	Property context
MQPD_COPYOPTIONS	DS F	Property copy options
MQPD_LENGTH	EQU *-MQPD	
MQPD_AREA	DS CL(MQPD_LENGTH)	

MQPD のフィールド詳細

プロパティ **MQPD** 記述子構造体は、プロパティの属性を定義するために使用されます。この構造は、MQSETMP 呼び出しの入出力パラメーターおよび MQINQMP 呼び出しの出力パラメーターです。

フィールド

注：次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 528. MQPD のフィールド		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
StrucId (構造識別子)	MQPD_STRUC_ID	'PD'
バージョン (構造バージョン番号)	MQPD_VERSION_1	1
オプション (オプション)	MQPD_NONE	0
サポート (メッセージプロパティに必要なサポート)	MQPD_SUPPORT_OPTIONAL	0
コンテキスト (プロパティが属するメッセージコンテキスト)	MQPD_NO_CONTEXT	0
CopyOptions (コピーオプションがどのプロパティに属するか)	MQCOPY_DEFAULT	0
注： 1. C プログラミング言語では、マクロ変数 MQPD_DEFAULT に表にリストされている値が設定されています。この変数を以下の方法で使用すると、構造体のフィールドに初期値を設定できます。 <pre>MQPD MyPD = {MQPD_DEFAULT};</pre>		

StrucId (MQCHAR4)

これはプロパティ記述子構造体の構造識別子です。常に入力フィールドです。その値は MQPD_STRUC_ID です。

値は次のものでなければなりません。

MQPD_STRUC_ID

プロパティ記述子構造体の識別子。

C プログラミング言語の場合、定数 **MQPD_STRUC_ID_ARRAY** も定義されます。これはと同じ値 **MQPD_STRUC_ID** を持ちますが、文字列ではなく文字の配列です。

Version (MQLONG)

これは構造体のバージョン番号です。値は以下のものでなければなりません。

MQPD_VERSION_1

バージョン 1 のプロパティ記述子構造体。

以下の定数は、現行バージョンのバージョン番号を指定しています。

MQPD_CURRENT_VERSION

プロパティ記述子構造体の現行バージョン。

これは常に入力フィールドです。このフィールドの初期値は **MQPD_VERSION_1** です。

Options (MQLONG)

値は次のものでなければなりません。

MQPD_NONE

指定されるオプションはありません。

これは常に入力フィールドです。フィールドの初期値は、MQPD_NONE です。

Support (MQLONG)

このフィールドは、メッセージ・プロパティーが含まれるメッセージをキューに書き込むために、キュー・マネージャーでこのプロパティーについてどのレベルのサポートが必要かを記述します。これは、IBM MQ 定義のプロパティーにのみ適用され、他のすべてのプロパティーに対するサポートはオプションです。

このフィールドは、IBM MQ 定義のプロパティーがキュー・マネージャーにより認知された時点で、正しい値に自動的に設定されます。プロパティーが認識されない場合は、MQPD_SUPPORT_OPTIONAL が割り当てられます。IBM MQ 定義のプロパティーが含まれているメッセージをキュー・マネージャーが受け取り、誤ったプロパティーであると認識した場合、キュー・マネージャーは *Support* フィールドの値を訂正します。

MQCMHO_NO_VALIDATION オプションが設定されたメッセージ・ハンドルで MQSETMP 呼び出しを使用して IBM MQ 定義のプロパティーを設定すると、*Support* は入力フィールドになります。これにより、アプリケーションは、接続しているキュー・マネージャーではサポートされない IBM MQ 定義のプロパティーに正しい値を設定する (メッセージの処理は別のキュー・マネージャーで行う) ことができます。

IBM MQ 定義のプロパティーではないプロパティーには、常に値 MQPD_SUPPORT_OPTIONAL が割り当てられます。

MQINQMP 呼び出しにより以下のいずれかの値が戻されます。または、MQCMHO_NO_VALIDATION オプションが設定されたメッセージ・ハンドルに MQSETMP 呼び出しを使用する場合には、いずれかの値を指定できます。

MQPD_SUPPORT_OPTIONAL

プロパティーはサポートされていなくても、キュー・マネージャーに受け入れられます。メッセージ・プロパティーをサポートしていないキュー・マネージャーにメッセージをフローするために、このプロパティーは破棄される場合があります。この値は、IBM MQ 定義ではないプロパティーにも割り当てられます。

MQPD_SUPPORT_REQUIRED

プロパティーに対するサポートは必須です。メッセージは、IBM MQ 定義のプロパティーをサポートしないキュー・マネージャーによりリジェクトされます。MQPUT 呼び出しまたは MQPUT1 呼び出しは、完了コード MQCC_FAILED および理由コード MQRC_UNSUPPORTED_PROPERTY で失敗します。

MQPD_SUPPORT_REQUIRED_IF_LOCAL

メッセージの宛先がローカル・キューになっている場合、メッセージは、IBM MQ 定義のプロパティーをサポートしないキュー・マネージャーによりリジェクトされます。MQPUT 呼び出しまたは MQPUT1 呼び出しは、完了コード MQCC_FAILED および理由コード MQRC_UNSUPPORTED_PROPERTY で失敗します。

メッセージの宛先がリモート・キュー・マネージャーである場合、MQPUT 呼び出しまたは MQPUT1 呼び出しは成功します。

メッセージ・ハンドルが MQCMHO_NO_VALIDATION オプションを設定して作成された場合、これは MQINQMP 呼び出しの出力フィールドおよび MQSETMP 呼び出しの入力フィールドになります。このフィールドの初期値は、MQPD_SUPPORT_OPTIONAL です。

Context (MQLONG)

ここでは、プロパティーが属しているメッセージ・コンテキストについて説明します。

IBM MQ 定義のプロパティーが含まれているメッセージをキュー・マネージャーが受け取り、誤ったプロパティーであると認識した場合、キュー・マネージャーは *Context* フィールドの値を訂正します。

次のようなオプションを指定できます。

MQPD_USER_CONTEXT

プロパティは user コンテキストに関連付けられます。

MQSETMP 呼び出しを使用してユーザー・コンテキストと関連付けたプロパティを設定するのに、特別な権限は必要ありません。

上記で説明されたオプションが必要ない場合、以下のオプションを使用できます。

MQPD_NO_CONTEXT

プロパティはメッセージ・コンテキストに関連付けられません。

認識されない値は MQRC_PD_ERROR の *Reason* コードで拒否されます。

これは、MQSETMP 呼び出しの入出力フィールドおよび MQINQMP 呼び出しからの出力フィールドです。このフィールドの初期値は、MQPD_NO_CONTEXT です。

CopyOptions (MQLONG)

これは、プロパティのコピー先となるメッセージ・タイプについて説明します。これは、認識された IBM MQ 定義プロパティのための出力専用フィールドです。IBM MQ によって適切な値が設定されます。

キュー・マネージャーが不正と認識した IBM MQ 定義のプロパティを含むメッセージを、キュー・マネージャーが受信した場合、キュー・マネージャーは、*CopyOptions* フィールドの値を訂正します。

以下に説明する 1 つ以上のオプションを指定できます。複数のオプションを指定するには、値を加算する (同じ定数を複数回加算しない) か、ビット単位 OR 演算を使用して値を結合 (プログラミング言語でビット演算がサポートされる場合) します。

MQCOPY_FORWARD

このプロパティは、転送されるメッセージにコピーされます。

MQCOPY_PUBLISH

このプロパティは、メッセージのパブリッシュ中にサブスクライバーが受信したメッセージにコピーされます。

MQCOPY_REPLY

このプロパティは応答メッセージにコピーされます。

MQCOPY_REPORT

このプロパティはレポート・メッセージにコピーされます。

MQCOPY_ALL

このプロパティはすべてのタイプの後続メッセージにコピーされます。

デフォルト・オプション: コピー・オプションのデフォルト・セットを提供するには、以下のオプションを指定できます。

MQCOPY_DEFAULT

このプロパティは、転送中のメッセージ、レポート・メッセージ、またはメッセージのパブリッシュ中にサブスクライバーが受信したメッセージにコピーされます。

これは、オプション MQCOPY_FORWARD、MQCOPY_REPORT、および MQCOPY_PUBLISH を組み合わせて指定するのと同じことになります。

上記で説明されたオプションを指定しない場合、以下のオプションを使用します。

MQCOPY_NONE

この値は、その他のコピー・オプションが指定されないことを示すために使用します。プログラム上では、このプロパティと後続のメッセージの間には関連はありません。これは、メッセージ記述子プロパティの場合は常に返されます。

これは、MQSETMP 呼び出しの入出力フィールドおよび MQINQMP 呼び出しからの出力フィールドです。このフィールドの初期値は、MQCOPY_DEFAULT です。

MQPMO - メッセージ書き出しオプション

MQPMO 構造により、アプリケーションはメッセージがキューに配置される方法やトピックに公開される方法を制御するオプションを指定できます。この構造体は、MQPUT および MQPUT1 呼び出しの入出力パラメーターです。

バージョン

現在の MQPMO のバージョンは MQPMO_VERSION_3 です。いくつかのフィールドは、特定のバージョンの MQPMO でのみ使用可能です。複数の環境でアプリケーションを移植する必要がある場合は、MQPMO のバージョンがすべての環境で整合していることを確認しなければなりません。特定のバージョンの構造体のみ存在するフィールドは、このトピックおよびフィールドの説明においてその旨が明記されています。

サポートされるプログラム言語用に提供されているヘッダー・ファイル、コピー・ファイル、およびインクルード・ファイルには、環境でサポートされる最新バージョンの MQPMO が含まれていますが、Version フィールドの初期値は MQPMO_VERSION_1 に設定されています。version-1 構造体に存在しないフィールドを使用するには、アプリケーションで、Version フィールドを必要なバージョンのバージョン番号に設定する必要があります。

文字セットとエンコード

MQPMO 内のデータは、キューマネージャ **CodedCharSetId** 属性によって指定される文字セットおよび MQENC_NATIVE によって指定されるローカルキューマネージャのエンコーディングでなければならない。ただし、アプリケーションが MQ MQI クライアントとして実行されている場合、構造体はクライアントの文字セットとエンコードに従っている必要があります。

フィールド

注: 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 529. MQPMO のフィールド		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
StrucId (構造識別子)	MQPMO_STRUC_ID	'PMO'
バージョン (構造バージョン番号)	MQPMO_VERSION_1	1
オプション (MQPUT および MQPUT1 の動作を制御するオプション)	MQPMO_NONE	0
タイムアウト (予約済み)	なし	-1
コンテキスト (入力キューのオブジェクトハンドル)	なし	0
KnownDestCount (ローカルキューに正常に送信されたメッセージの数)	なし	0
UnknownDestCount (リモートキューに正常に送信されたメッセージの数)	なし	0
InvalidDestCount (送信できなかったメッセージの数)	なし	0
ResolvedQName (解決済み宛先キュー名)	なし	ヌル・ストリングまたはブランク
ResolvedQMgrName (解決済み宛先キューマネージャ名)	なし	ヌル・ストリングまたはブランク
注: Version が MQPMO_VERSION_2 より小さい場合は、残りのフィールドは無視されます。		

表 529. MQPMO のフィールド (続き)

フィールド名と説明	定数の名前	定数の初期値 (存在する場合)
<u>RecsPresent</u> (存在するプットメッセージレコードまたは応答レコードの数)	なし	0
<u>PutMsgRecFields</u> (MQPMR フィールドの存在を示すフラグ)	MQPMRF_NONE	0
<u>PutMsgRecOffset</u> (MQPMO の先頭からの最初の put メッセージレコードのオフセット)	なし	0
<u>ResponseRecOffset</u> (MQPMO の先頭からの最初の応答レコードのオフセット)	なし	0
<u>PutMsgRecPtr</u> (最初のメッセージ記録のアドレス)	なし	ヌル・ポインターまたはヌル・バイト
<u>ResponseRecPtr</u> (最初の応答レコードのアドレス)	なし	ヌル・ポインターまたはヌル・バイト
注: <i>Version</i> が MQPMO_VERSION_3 より小さい場合、残りのフィールドは無視されます。		
<u>OriginalMsgHandle</u> (元のメッセージハンドル)	MQHM_NONE	0
<u>NewMsgHandle</u> (新しいメッセージハンドル)	MQHM_NONE	0
アクション (実行されるプットの種類と、 <code>OriginalMsgHandle</code> フィールドで指定された元のメッセージと <code>NewMsgHandle</code> フィールドで指定された新しいメッセージとの関係)	MQACTP_NEW	0
<u>PubLevel</u> (出版物が対象とする購読レベル)	なし	9
注: 1. 記号-は、単一のブランク文字を表します。 2. ヌル・ストリングまたはブランクの値は、C 言語ではヌル・ストリングを表し、他のプログラミング言語ではブランク文字を表します。 3. C プログラミング言語では、マクロ変数 MQPMO_DEFAULT には、表にリストされている値が含まれています。このマクロ変数を以下の方法で使用して、構造体のフィールドに初期値を設定します。 <pre>MQPMO MyPMO = {MQPMO_DEFAULT};</pre>		

言語ごとの宣言

MQPMO の C 宣言

```
typedef struct tagMQPMO MQPMO;
struct tagMQPMO {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQLONG     Options;          /* Options that control the action of
                                MQPUT and MQPUT1 */
    MQLONG     Timeout;          /* Reserved */
    MQHOBJ     Context;          /* Object handle of input queue */
    MQLONG     KnownDestCount;    /* Number of messages sent
                                successfully to local queues */
    MQLONG     UnknownDestCount; /* Number of messages sent
                                successfully to remote queues */
    MQLONG     InvalidDestCount; /* Number of messages that could not
                                be sent */
    MQCHAR48   ResolvedQName;    /* Resolved name of destination
```

```

        MQCHAR48  ResolvedQMGrName;    /* queue */
        /* Resolved name of destination queue
        manager */
/* Ver:1 */
        MQLONG    RecsPresent;          /* Number of put message records or
        response records present */
        MQLONG    PutMsgRecFields;      /* Flags indicating which MQPMR fields
        are present */
        MQLONG    PutMsgRecOffset;      /* Offset of first put message record
        from start of MQPMO */
        MQLONG    ResponseRecOffset;    /* Offset of first response record
        from start of MQPMO */
        MQPTR     PutMsgRecPtr;         /* Address of first put message
        record */
        MQPTR     ResponseRecPtr;       /* Address of first response record */
/* Ver:2 */
        MQHMSG    OriginalMsgHandle;    /* Original message handle */
        MQHMSG    NewMsgHandle;         /* New message handle */
        MQLONG    Action;               /* The action being performed */
        MQLONG    PubLevel;             /* Subscription level */
/* Ver:3 */
};

```

MQPMO の COBOL 宣言

```

**      MQPMO structure
10 MQPMO.
**      Structure identifier
15 MQPMO-STRUCID      PIC X(4).
**      Structure version number
15 MQPMO-VERSION     PIC S9(9) BINARY.
**      Options that control the action of MQPUT and MQPUT1
15 MQPMO-OPTIONS     PIC S9(9) BINARY.
**      Reserved
15 MQPMO-TIMEOUT     PIC S9(9) BINARY.
**      Object handle of input queue
15 MQPMO-CONTEXT     PIC S9(9) BINARY.
**      Number of messages sent successfully to local queues
15 MQPMO-KNOWNDESTCOUNT PIC S9(9) BINARY.
**      Number of messages sent successfully to remote queues
15 MQPMO-UNKNOWNDESTCOUNT PIC S9(9) BINARY.
**      Number of messages that could not be sent
15 MQPMO-INVALIDDESTCOUNT PIC S9(9) BINARY.
**      Resolved name of destination queue
15 MQPMO-RESOLVEDQNAME PIC X(48).
**      Resolved name of destination queue manager
15 MQPMO-RESOLVEDQMGRNAME PIC X(48).
**      Number of put message records or response records present
15 MQPMO-RECSPRESENT PIC S9(9) BINARY.
**      Flags indicating which MQPMR fields are present
15 MQPMO-PUTMSGRECFIELDS PIC S9(9) BINARY.
**      Offset of first put message record from start of MQPMO
15 MQPMO-PUTMSGRECOFFSET PIC S9(9) BINARY.
**      Offset of first response record from start of MQPMO
15 MQPMO-RESPONSERECOFFSET PIC S9(9) BINARY.
**      Address of first put message record
15 MQPMO-PUTMSGRECPTTR POINTER.
**      Address of first response record
15 MQPMO-RESPONSERECPTTR POINTER.
**      Original message handle
15 MQPMO-ORIGINALMSGHANDLE PIC S9(18) BINARY.
**      New message handle
15 MQPMO-NEWMSGHANDLE PIC S9(18) BINARY.
**      The action being performed
15 MQPMO-ACTION      PIC S9(9) BINARY.
**      Publish level
15 MQPMO-PUBLEVEL    PIC S9(9) BINARY.

```

MQPMO の PL/I 宣言

```

dcl
  1 MQPMO based,
    3 StrucId          char(4),          /* Structure identifier */
    3 Version          fixed bin(31),    /* Structure version number */
    3 Options          fixed bin(31),    /* Options that control the action
    of MQPUT and MQPUT1 */
    3 Timeout          fixed bin(31),    /* Reserved */
    3 Context          fixed bin(31),    /* Object handle of input queue */
    3 KnownDestCount   fixed bin(31),    /* Number of messages sent

```

3	UnknownDestCount	fixed bin(31),	/* Number of messages sent successfully to local queues */
3	InvalidDestCount	fixed bin(31),	/* Number of messages sent successfully to remote queues */
3	ResolvedQName	char(48),	/* Number of messages that could not be sent */
3	ResolvedQMgrName	char(48),	/* Resolved name of destination queue */
3	RecsPresent	fixed bin(31),	/* Resolved name of destination queue manager */
3	PutMsgRecFields	fixed bin(31),	/* Number of put message records or response records present */
3	PutMsgRecOffset	fixed bin(31),	/* Flags indicating which MQPMR fields are present */
3	ResponseRecOffset	fixed bin(31),	/* Offset of first put message record from start of MQPMO */
3	PutMsgRecPtr	pointer,	/* Offset of first response record from start of MQPMO */
3	ResponseRecPtr	pointer,	/* Address of first put message record */
3	OriginalMsgHandle	fixed bin(63),	/* Address of first response record */
3	NewMsgHandle	fixed bin(63);	/* Original message handle */
3	Action	fixed bin(31);	/* New message handle */
3	PubLevel	fixed bin(31);	/* The action being performed */
			/* Publish level */

MQPMO の高水準アセンブラ宣言

MQPMO	DSECT		
MQPMO_STRUCID	DS	CL4	Structure identifier
MQPMO_VERSION	DS	F	Structure version number
MQPMO_OPTIONS	DS	F	Options that control the action of MQPUT and MQPUT1
*			
MQPMO_TIMEOUT	DS	F	Reserved
MQPMO_CONTEXT	DS	F	Object handle of input queue
MQPMO_KNOWNDESTCOUNT	DS	F	Number of messages sent successfully to local queues
*			
MQPMO_UNKNOWNDSTCOUNT	DS	F	Number of messages sent successfully to remote queues
*			
MQPMO_INVALIDDESTCOUNT	DS	F	Number of messages that could not be sent
*			
MQPMO_RESOLVEDQNAME	DS	CL48	Resolved name of destination queue
MQPMO_RESOLVEDQMGRNAME	DS	CL48	Resolved name of destination queue manager
*			
MQPMO_RECSPRESENT	DS	F	Number of put message records or response records present
*			
MQPMO_PUTMSGRECFIELDS	DS	F	Flags indicating which MQPMR fields are present
*			
MQPMO_PUTMSGRECOFFSET	DS	F	Offset of first put message record from start of MQPMO
*			
MQPMO_RESPONSERECOFFSET	DS	F	Offset of first response record from start of MQPMO
*			
MQPMO_PUTMSGRECPtr	DS	F	Address of first put message record
*			
MQPMO_RESPONSERECPtr	DS	F	Address of first response record
MQPMO_ORIGINALMSGHANDLE	DS	D	Original message handle
MQPMO_NEWMSGHANDLE	DS	D	New message handle
MQPMO_ACTION	DS	F	The action being performed
MQPMO_PUBLEVEL	DS	F	Publish level
*			
MQPMO_LENGTH	EQU		*-MQPMO
	ORG		MQPMO
MQPMO_AREA	DS		CL(MQPMO_LENGTH)

MQPMO の Visual Basic 宣言

Type MQPMO		
StrucId	As String*4	'Structure identifier'
Version	As Long	'Structure version number'
Options	As Long	'Options that control the action of 'MQPUT and MQPUT1'
Timeout	As Long	'Reserved'
Context	As Long	'Object handle of input queue'
KnownDestCount	As Long	'Number of messages sent successfully 'to local queues'
UnknownDestCount	As Long	'Number of messages sent successfully 'to remote queues'
InvalidDestCount	As Long	'Number of messages that could not be 'sent'

```

ResolvedQName      As String*48 'Resolved name of destination queue'
ResolvedQMgrName    As String*48 'Resolved name of destination queue'
                        'manager'
RecsPresent         As Long      'Number of put message records or'
                        'response records present'
PutMsgRecFields     As Long      'Flags indicating which MQPMR fields'
                        'are present'
PutMsgRecOffset     As Long      'Offset of first put message record'
                        'from start of MQPMO'
ResponseRecOffset   As Long      'Offset of first response record from'
                        'start of MQPMO'
PutMsgRecPtr        As MQPTR     'Address of first put message record'
ResponseRecPtr      As MQPTR     'Address of first response record'
End Type

```

MQPMO のフィールド詳細

MQPMO の Put メッセージオプション構造体により、アプリケーションはメッセージがキューに配置される方法、またはトピックに公開される方法を制御するオプションを指定できます。この構造体は、MQPUT および MQPUT1 呼び出しの入出力パラメーターです。

フィールド

注: 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 530. MQPMO のフィールド		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
<u>StrucId</u> (構造識別子)	MQPMO_STRUC_ID	'PMO'
<u>バージョン</u> (構造バージョン番号)	MQPMO_VERSION_1	1
<u>オプション</u> (MQPUT および MQPUT1 の動作を制御するオプション)	MQPMO_NONE	0
<u>タイムアウト</u> (予約済み)	なし	-1
<u>コンテキスト</u> (入力キューのオブジェクトハンドル)	なし	0
<u>KnownDestCount</u> (ローカルキューに正常に送信されたメッセージの数)	なし	0
<u>UnknownDestCount</u> (リモートキューに正常に送信されたメッセージの数)	なし	0
<u>InvalidDestCount</u> (送信できなかったメッセージの数)	なし	0
<u>ResolvedQName</u> (解決済み宛先キュー名)	なし	ヌル・ストリングまたはブランク
<u>ResolvedQMgrName</u> (解決済み宛先キューマネージャ名)	なし	ヌル・ストリングまたはブランク
注: <i>Version</i> が MQPMO_VERSION_2 より小さい場合は、残りのフィールドは無視されます。		
<u>RecsPresent</u> (存在するプットメッセージレコードまたは応答レコードの数)	なし	0
<u>PutMsgRecFields</u> (MQPMR フィールドの存在を示すフラグ)	MQPMRF_NONE	0
<u>PutMsgRecOffset</u> (MQPMO の先頭からの最初の put メッセージレコードのオフセット)	なし	0

表 530. MQPMO のフィールド (続き)		
フィールド名と説明	定数の名前	定数の初期値 (存在する場合)
<u>ResponseRecOffset</u> (MQPMO の先頭からの最初の応答レコードのオフセット)	なし	0
<u>PutMsgRecPtr</u> (最初のメッセージ記録のアドレス)	なし	ヌル・ポインターまたはヌル・バイト
<u>ResponseRecPtr</u> (最初の応答レコードのアドレス)	なし	ヌル・ポインターまたはヌル・バイト
注: <i>Version</i> が MQPMO_VERSION_3 より小さい場合、残りのフィールドは無視されます。		
<u>OriginalMsgHandle</u> (元のメッセージハンドル)	MQHM_NONE	0
<u>NewMsgHandle</u> (新しいメッセージハンドル)	MQHM_NONE	0
アクション (実行されるプットの種類と、 <code>OriginalMsgHandle</code> フィールドで指定された元のメッセージと <code>NewMsgHandle</code> フィールドで指定された新しいメッセージとの関係)	MQACTP_NEW	0
<u>PubLevel</u> (出版物が対象とする購読レベル)	なし	9
注: 1. 記号-は、単一のブランク文字を表します。 2. ヌル・ストリングまたはブランクの値は、C 言語ではヌル・ストリングを表し、他のプログラミング言語ではブランク文字を表します。 3. C プログラミング言語では、マクロ変数 MQPMO_DEFAULT には、表にリストされている値が含まれています。このマクロ変数を以下の方法で使用して、構造体のフィールドに初期値を設定します。 <pre>MQPMO MyPMO = {MQPMO_DEFAULT};</pre>		

StrucId (MQCHAR4)

これは、put メッセージオプション構造体の構造体識別子です。常に入力フィールドです。その値は MQPMO_STRUC_ID です。

値は次のものでなければなりません。

MQPMO_STRUC_ID

put メッセージオプション構造体の識別子。

C プログラミング言語では、定数 MQPMO_STRUC_ID_ARRAY も定義されている。これは MQPMO_STRUC_ID と同じ値を持ちますが、文字列ではなく文字の配列です。

Version (MQLONG)

構造体のバージョン番号。

値は次のいずれかでなければなりません。

MQPMO_VERSION_1

バージョン 1 の書き込みメッセージ・オプション構造体。

このバージョンはすべての環境でサポートされます。

MQPMO_VERSION_2

バージョン 2 の書き込みメッセージ・オプション構造体。

このバージョンは、次の環境でサポートされます。

-  AIX
-  IBM i
-  Linux
-  Windows

および、これらのシステムに接続された IBM MQ MQI clients。

MQPMO_VERSION_3

バージョン 3 の書き込みメッセージ・オプション構造体。

このバージョンはすべての環境でサポートされます。

これより新しいバージョンの構造体にのみ存在するフィールドは、そのフィールドの説明にその旨記載されています。以下の定数は、現行バージョンのバージョン番号を指定しています。

MQPMO_CURRENT_VERSION

書き込みメッセージ・オプション構造体の現行バージョン。

これは常に入力フィールドです。このフィールドの初期値は、MQPMO_VERSION_1 です。

Options (MQLONG)

このオプション・フィールドは **MQPUT** および **MQPUT1** 呼び出しの操作を制御します。

スコープオプション

MQPMO オプションのいずれかを指定する (または何も指定しない) ことができます。複数のオプションを指定するには、値を加算する (同じ定数を複数回加算しない) か、ビット単位 OR 演算を使用して値を結合 (プログラミング言語でビット演算がサポートされる場合) します。有効でない組み合わせについては、注記されています。それ以外の組み合わせは有効です。

以下のオプションは、送られるパブリケーションの有効範囲を制御します。

MQPMO_SCOPE_QMGR

パブリケーションは、このキュー・マネージャーにサブスクライブしたサブスクライバーにのみ送られます。パブリケーションは、このキュー・マネージャーにサブスクリプションを作成したリモート・パブリッシュ/サブスクライブ・キュー・マネージャーには転送されません。これは、PUBSCOPE トピック属性を使用して設定された動作より優先されます。

注: 設定されていない場合、パブリケーションの有効範囲は PUBSCOPE トピック属性によって決定されます。

出版オプション

以下のオプションは、メッセージをトピックに公開する方法を制御します。

MQPMO_SUPPRESS_REPLYTO

このパブリケーションの MQMD の *ReplyToQ* および *ReplyToQMgr* フィールドに指定された情報は、サブスクライバーに渡されません。このオプションを、*ReplyToQ* を必要とするレポート・オプションと一緒に使用すると、呼び出しは MQRC_MISSING_REPLY_TO_Q で失敗します。

MQPMO_RETAIN

送信されたパブリケーションがキュー・マネージャーによって保存されます。この保存では、サブスクライバーはこのパブリケーションが公開された後、MQSUBRQ 呼び出しを使用することにより、そのコピーを要求することができます。また、パブリケーションが作成された後で、サブスクリプションを作成するアプリケーションにこのパブリケーションを送信することも可能になります (オプション MQSO_NEW_PUBLICATIONS_ONLY を使用してパブリケーションを送信しないように選択した場合を除く)。保存されたパブリケーションがアプリケーションに送られると、そのパブリケーションの MQIsRetained メッセージ・プロパティによって示されます。

トピック・ツリーの各ノードに保存できるパブリケーションは1つだけです。それで、他のアプリケーションによって公開されたこのトピックの保存パブリケーションが既に存在する場合は、このパブリケーションによって置き換えられます。そのため、同じトピックに関するメッセージを保存するパブリッシャーを複数持つことは避けたほうがよいでしょう。

保存パブリケーションがサブスクライバーによって要求される場合、使用されるサブスクリプションのトピックにワイルドカードが含まれていることがあります。その場合、(トピック・ツリーのさまざまなノードの)いくつかの保存パブリケーションがマッチングする可能性があり、複数のパブリケーションが要求側のアプリケーションに送られる場合があります。詳細については、[788 ページの『MQSUBRQ - サブスクリプション要求』](#) 呼び出しの説明を参照してください。

保存パブリケーションとサブスクリプション・レベルの相互作用については、[パブリケーションの代行受信](#)を参照してください。

このオプションを使用してもパブリケーションを保持できない場合、メッセージはパブリッシュされず、呼び出しは MQRC_PUT_NOT_RETAINED で失敗します。

MQPMO_NOT_OWN_SUBS

アプリケーションが所有するサブスクリプションにパブリケーションを送信しないことを、キュー・マネージャーに指示します。接続ハンドルが同じ場合、サブスクリプションは同じアプリケーションが所有するものと見なされます。

MQPMO_WARN_IF_NO_SUBS_MATCHED

パブリケーションに一致するサブスクリプションがない場合は、完了コード (CompCode)

MQCC_WARNING と理由コード MQRC_NO_SUBS_MATCHED を戻します。

PUT 操作によって MQRC_NO_SUBS_MATCHED が戻された場合、そのパブリケーションはどのサブスクリプションにも送達されていません。ただし、その PUT 操作で MQPMO_RETAIN オプションが指定されている場合、メッセージは保持され、その後で定義された一致サブスクリプションに送達されます。

以下のいずれかの条件が満たされる場合、トピックのサブスクリプションはパブリケーションに一致します。

- メッセージがそのサブスクリプション・キューに送達されている。
- メッセージがそのサブスクリプション・キューに送達されるはずであったが、キューに問題があってそのキューにメッセージを入れることができないため、送達不能キューに入れられたか、または廃棄された。
- ルーティング出口が、そのサブスクリプションへのメッセージ送達を抑制するように定義されている。

以下のいずれかの条件が満たされる場合、トピックのサブスクリプションはパブリケーションに一致しません。

- サブスクリプションの選択ストリングがパブリケーションに一致していない。
- サブスクリプションで MQSO_PUBLICATION_ON_REQUEST オプションが指定されている。
- PUT 操作で MQPMO_NOT_OWN_SUBS オプションが指定され、サブスクリプションがパブリッシャーの ID と一致しているため、パブリケーションが送達されない。

同期ポイントのオプション

以下のオプションは、作業単位内の、MQPUT または MQPUT1 呼び出しの関与に関連します。

MQPMO_SYNCPOINT

この要求は、通常の作業単位プロトコルの中で操作することです。メッセージは、作業単位がコミットされるまで、作業単位の外側には表示されません。作業単位がバックアウトされると、メッセージは除去されます。

MQPMO_SYNCPOINT と MQPMO_NO_SYNCPOINT をどちらも指定しない場合、書き込み要求が作業単位プロトコルに組み込まれるかどうかは、アプリケーションを実行している環境ではなく、キュー・マネージャーを実行している環境によって決まります。z/OS では、書き込み要求は作業単位の中に含まれます。他のすべての環境では、書き込み要求は作業単位に組み込まれません。

各環境ではこのような違いがあるため、移植するアプリケーションではこのオプションをデフォルト値にせず、MQPMO_SYNCPOINT または MQPMO_NO_SYNCPOINT のいずれかを明示的に指定する必要があります。

MQPMO_SYNCPOINT と MQPMO_NO_SYNCPOINT は同時に指定しないでください。

MQPMO_NO_SYNCPOINT

この要求は、通常の作業単位プロトコルの外部で動作することになります。メッセージは即時に使用可能になり、作業単位をバックアウトしても削除できません。

MQPMO_NO_SYNCPOINT と MQPMO_SYNCPOINT をどちらも指定しない場合、書き込み要求が作業単位プロトコルに組み込まれるかどうかは、アプリケーションを実行している環境ではなく、キュー・マネージャーを実行している環境によって決まります。z/OS では、書き込み要求は作業単位の中に含まれます。他のすべての環境では、書き込み要求は作業単位に組み込まれません。

各環境ではこのような違いがあるため、移植するアプリケーションではこのオプションをデフォルト値にせず、MQPMO_SYNCPOINT または MQPMO_NO_SYNCPOINT のいずれかを明示的に指定する必要があります。

MQPMO_NO_SYNCPOINT と MQPMO_SYNCPOINT は同時に指定しないでください。

メッセージ識別子および関連識別子のオプション

以下のオプションは、キュー・マネージャーが新しいメッセージ ID、または関連 ID を生成するように要求します。

MQPMO_NEW_MSG_ID

キュー・マネージャーは、MQMD 内の *MsgId* フィールドの内容を新しいメッセージ ID に置き換えます。このメッセージ ID はメッセージと共に送信され、MQPUT 呼び出しまたは MQPUT1 呼び出しからの出力時にアプリケーションに戻ります。

MQPMO_NEW_MSG_ID オプションは、メッセージが配布リストに書き込まれるときにも指定できます。詳細については、MQPMR 構造体の *MsgId* フィールドの説明を参照してください。

このオプションを使用すると、各 MQPUT または MQPUT1 呼び出しの前に *MsgId* フィールドを MQMI_NONE にリセットする必要がなくなります。

MQPMO_NEW_CORREL_ID

キュー・マネージャーは、MQMD 内の *CorrelId* フィールドの内容を新しい関連 ID に置き換えます。この関連 ID はメッセージと共に送信され、MQPUT 呼び出しまたは MQPUT1 呼び出しからの出力時にアプリケーションに戻ります。

MQPMO_NEW_CORREL_ID オプションは、メッセージが配布リストに書き込まれているときにも指定できます。詳細については、MQPMR 構造体の *CorrelId* フィールドの説明を参照してください。

MQPMO_NEW_CORREL_ID は、アプリケーションに固有の関連 ID が必要な状況で役に立ちます。

グループおよびセグメントのオプション

以下のオプションは、論理メッセージのグループおよびセグメント内のメッセージの処理に関連しています。オプションを理解する上で助けとなる定義を以下に示します。



重要: パブリッシュ/サブスクライブでは、セグメント化またはグループ化されたメッセージを使用できません。

物理メッセージ

物理メッセージは、キューに入れたりキューから除去できる最小単位の情報です。多くの場合、1つの MQPUT、MQPUT1、または MQGET 呼び出しで指定された情報や取り出された情報に相当します。すべての物理メッセージには、固有のメッセージ記述子 (MQMD) があります。通常、物理メッセージは、メッセージ ID (MQMD の *MsgId* フィールド) の異なる値によって区別されます。ただし、これはキュー・マネージャーによって強制されるものではありません。

論理メッセージ

論理メッセージは、1 単位のアプリケーション情報です (z/OS 以外のプラットフォームの場合のみ)。システムに制約がない場合には、1つの論理メッセージが1つの物理メッセージになります。ただし、

論理メッセージが非常に大きい場合、システムの制約により、1つの論理メッセージをセグメントと呼ばれる複数の物理メッセージに分割することが必要になる場合があります。

セグメント化された論理メッセージは、同じ非ヌル・グループ ID (MQMD の *GroupId* フィールド) および同じメッセージ・シーケンス番号 (MQMD の *MsgSeqNumber* フィールド) を持つ複数の物理メッセージで構成されます。セグメントは、セグメント・オフセット (MQMD の *Offset* フィールド) の固有の値によって区別されます。この値は、論理メッセージ内のデータの先頭からの物理メッセージ内のデータのオフセットを示します。各セグメントは1つの物理メッセージなので、論理メッセージのセグメントにはそれぞれ、固有のメッセージ ID があります。

セグメント化されていない論理メッセージにも、送信側のアプリケーションでセグメント化が許可されている場合は、NULL 以外のグループ ID があります。ただし、この場合、論理メッセージが1つのメッセージ・グループに属していないと、1つの物理メッセージしかグループ ID を持ちません。論理メッセージが1つのメッセージ・グループに属していない限り、送信側のアプリケーションによってセグメント化が禁止されている論理メッセージのグループ ID はヌルとなります (MQGI_NONE)。

メッセージ・グループ

メッセージ・グループは、同じヌル以外のグループ ID を持つ1つ以上の論理メッセージのセットです。グループ内のそれぞれの論理メッセージは、メッセージ順序番号に指定された固有の値で区別されます。指定される値は1から n までの整数で、 n はグループ内の論理メッセージの数です。1つ以上の論理メッセージがセグメント化されている場合、グループ内の物理メッセージの数は n 個を超えます。

MQPMO_LOGICAL_ORDER

このオプションは、キュー・マネージャーに、アプリケーションがグループ内のメッセージと論理メッセージのセグメントを書き込む方法を指示します。このオプションは、MQPUT 呼び出しでのみ指定できます。MQPUT1 呼び出しでは無効です。

が MQPMO_LOGICAL_ORDER 指定された場合、アプリケーションは連続した MQPUT 呼び出しを使用し、以下の処理を行うことを示します：

1. 各論理メッセージ内のセグメントを、0 からセグメント・オフセットの小さい順に間を空けずに書き込む。
2. 論理メッセージ内のセグメントをすべて書き込んでから、その次の論理メッセージのセグメントを書き込む。
3. 各メッセージ・グループ内の論理メッセージを、1 からメッセージ順序番号の小さい順に間を空けずに書き込む。IBM MQ はメッセージ・シーケンス番号を自動的にインクリメントします。
4. メッセージ・グループ内の論理メッセージをすべて書き込んでから、その次のメッセージ・グループの論理メッセージを書き込む。

MQPMO_LOGICAL_ORDER の詳細については、[論理的な順序付けと物理的な順序付け](#)を参照してください。

コンテキストオプション

以下のオプションは、メッセージ・コンテキストの処理を制御します。

MQPMO_NO_CONTEXT

コンテキストが存在しないことを示すために、識別コンテキストと起点コンテキストの両方が設定されます。つまり、MQMD のコンテキスト・フィールドは次のように設定されます。

- 文字フィールドの場合はブランク
- バイト・フィールドの場合はヌル
- 数値フィールドの場合はゼロ

MQPMO_DEFAULT_CONTEXT

識別および発信元の両方のデフォルトのコンテキスト情報がメッセージに関連付けられます。キュー・マネージャーは、メッセージ記述子のコンテキスト・フィールドを以下のように設定します。

表 531. MQMD フィールドのデフォルトのコンテキスト情報の値

MQMD のフィールド	使用される値
<i>UserIdentifier</i>	環境から決定できる場合はその値。それ以外のときは、ブランクに設定される。
<i>AccountingToken</i>	環境から判別できる場合は判別されます。判別できない場合は MQACT_NONE に設定されます。
<i>ApplIdentityData</i>	ブランクに設定されます。
<i>PutApplType</i>	環境から決定される。
<i>PutApplName</i>	環境から決定できる場合はその値。それ以外のときは、ブランクに設定される。
<i>PutDate</i>	メッセージが書き込まれる日付に設定される。
<i>PutTime</i>	メッセージが書き込まれる時刻に設定される。
<i>ApplOriginData</i>	ブランクに設定されます。

メッセージのコンテキストの詳細については、[メッセージのコンテキスト](#)を参照してください。

これらは、コンテキスト・オプションを指定しない場合のデフォルトの値、そして処置です。

MQPMO_PASS_IDENTITY_CONTEXT

メッセージには、識別コンテキストに関連付けられているコンテキスト情報が含まれます。識別コンテキストは、*Context* フィールドで指定されたキュー・ハンドルから取得されます。発信元コンテキスト情報は、MQPMO_DEFAULT_CONTEXT の場合と同様にキュー・マネージャーによって生成されます (値については、上記の表を参照)。メッセージのコンテキストの詳細については、[メッセージのコンテキスト](#)を参照してください。

MQPUT 呼び出しでは、キューが MQOO_PASS_IDENTITY_CONTEXT オプション (あるいは、それを暗黙指定するオプション) を指定してオープンされている必要があります。MQPUT1 呼び出しでは、MQOO_PASS_IDENTITY_CONTEXT オプションを指定した MQOPEN 呼び出しの場合と同じ許可検査が行われます。

MQPMO_PASS_ALL_CONTEXT

メッセージには、識別コンテキストに関連付けられているコンテキスト情報が含まれます。コンテキストは、*Context* フィールドに指定されたキュー・ハンドルから取得されます。メッセージ・コンテキストの詳細については、[コンテキスト情報の制御](#)を参照してください。

MQPUT 呼び出しでは、キューが MQOO_PASS_ALL_CONTEXT オプション (あるいは、暗黙指定するオプションで) を指定してオープンされている必要があります。MQPUT1 呼び出しでは、MQOO_PASS_ALL_CONTEXT オプションを指定した MQOPEN 呼び出しの場合と同じ許可検査が行われます。

MQPMO_SET_IDENTITY_CONTEXT

メッセージには、識別コンテキストに関連付けられているコンテキスト情報が含まれます。アプリケーションでは、MQMD 構造体の識別コンテキストを指定します。発信元コンテキスト情報は、MQPMO_DEFAULT_CONTEXT の場合と同様にキュー・マネージャーによって生成されます (値については、上記の表を参照)。メッセージのコンテキストの詳細については、[メッセージのコンテキスト](#)を参照してください。

MQPUT 呼び出しでは、キューを MQOO_SET_IDENTITY_CONTEXT オプション (あるいは、それを暗黙指定するオプション) を指定してオープンされている必要があります。MQPUT1 呼び出しでは、MQOO_SET_IDENTITY_CONTEXT オプションを指定した MQOPEN 呼び出しの場合と同じ許可検査が行われます。

MQPMO_SET_ALL_CONTEXT

メッセージには、識別コンテキストに関連付けられているコンテキスト情報が含まれます。アプリケーションでは、MQMD 構造体の識別コンテキスト、起点コンテキスト、およびユーザー・コンテキスト

トを指定します。メッセージのコンテキストの詳細については、[メッセージのコンテキスト](#)を参照してください。

MQPUT 呼び出しでは、キューを MQOO_SET_ALL_CONTEXT オプションでオープンする必要があります。MQPUT1 呼び出しでは、MQOO_SET_ALL_CONTEXT オプションによる MQOPEN 呼び出しの場合と同じ許可検査が行われます。

指定できるのは、MQPMO_*_CONTEXT コンテキスト・オプションのうちの 1 つだけです。何も指定しない場合は MQPMO_DEFAULT_CONTEXT が想定されます。

プロパティオプション

以下のオプションは、メッセージのプロパティに関連したオプションです。

MQPMO_MD_FOR_OUTPUT_ONLY

メッセージ記述子パラメーターは、書き込まれたメッセージのメッセージ記述子を戻す出力でのみ使用されます。**MQPMO** 構造体の *NewMsgHandle* または *OriginalMsgHandle* (あるいはその両方) のフィールドに関連付けられたメッセージ記述子フィールドは、入力のために使用する必要があります。

有効なメッセージ・ハンドルが提供されないと、呼び出しは失敗し、理由コード **MQRC_MD_ERROR** が戻されます。

回答選択肢

以下のオプションは、MQPUT または MQPUT1 呼び出しに対して戻される応答を制御します。これらのオプションのうち 1 つのみ指定できます。MQPMO_ASYNC_RESPONSE と MQPMO_SYNC_RESPONSE がどちらも指定されていない場合、MQPMO_RESPONSE_AS_Q_DEF または MQPMO_RESPONSE_AS_TOPIC_DEF が想定されます。

MQPMO_ASYNC_RESPONSE

MQPMO_ASYNC_RESPONSE オプションは、アプリケーションがキュー・マネージャーによる呼び出しの完了を待たずに MQPUT または MQPUT1 操作を完了することを要求します。このオプションを使用すると、メッセージング・パフォーマンスが改善される可能性があります。クライアント・バインディングを使用するアプリケーションの場合は特にそうです。アプリケーションは、MQSTAT verb を使って、前の非同期呼び出し中にエラーが発生したかどうかを定期的に検査することができます。

このオプションの場合、MQMD の以下のフィールドだけに値が入れられることが保証されます。

- ApplIdentityData
- PutApplType
- PutApplName
- ApplOriginData

さらに、MQPMO_NEW_MSG_ID または MQPMO_NEW_CORREL_ID のいずれか、あるいはその両方がオプションとして指定されると、戻される MsgId と CorrelId も完了されます (MQPMO_NEW_MSG_ID はブランクの MsgId フィールドを指定することによって暗黙的に指定することができます)。

これより前に指定されたフィールドだけが完了されます。通常、MQMD または MQPMO 構造体で戻されるその他の情報は未定義となっています。

MQPUT1 で非同期書き込み応答を要求した場合に、MQOD 構造体で戻される ResolvedQName および ResolvedQMgrName は未定義です。

MQPUT または MQPUT1 で非同期書き込み応答を要求するときに出力される CompCode および MQCC_OK および MQRC_NONE の Reason は、必ずしもメッセージがキューに正常に書き込まれたことを意味するものではありません。非同期書き込み応答を使用する MQI アプリケーションを作成するときに、メッセージがキューに書き込まれたことの確認を必要とする場合は、PUT 操作からの CompCode コードおよび Reason コードの両方を検査する必要があります。また、MQSTAT を使用して非同期エラー情報を照会することも必要です。

各 MQPUT または MQPUT1 呼び出しの成功または失敗はすぐには戻されないかもしれませんが、非同期呼び出しの下で発生した最初のエラーは、後で MQSTAT を呼び出すことによって判別できます。

非同期書き込み応答を使って同期点の下で持続メッセージを送達できず、トランザクションをコミットしようとする、そのコミットは失敗し、トランザクションがバックアウトされます。その際、完了コ

ード MQCC_FAILED、理由コード MQRC_BACKED_OUT が戻されます。アプリケーションは MQSTAT を呼び出すことにより、直前の MQPUT または MQPUT1 の失敗の原因を判別することができます。

同期ポイント下の非永続メッセージが非同期プット応答による配信に失敗し、トランザクションのコミットを試みた場合、コミットは成功するが非永続メッセージは失われる。この状況を回避するには、アプリケーションは MQCMIT を呼び出す前に、MQSTAT を呼び出して、以前に MQPUT または MQPUT1 の失敗があったかどうかを判断する必要があります。失敗が発生した場合、アプリケーションは MQBACK を呼び出すべきである。

MQPMO_SYNC_RESPONSE

この書き込み応答タイプを指定すると、MQPUT または MQPUT1 操作を常に同期的に発行することができます。PUT 操作が成功すると、MQMD および MQPMO のすべてのフィールドが完了します。

このオプションは、キューまたはトピック・オブジェクトで定義されるデフォルトの書き込み応答値に関係なく、確実に同期応答が行われるようにします。

MQPMO_RESPONSE_AS_Q_DEF

この値が MQPUT 呼び出しで指定されると、使用される書き込み応答タイプは、アプリケーションで最初にオープンされたときにキューで指定された DEFPRESP 値から取られます。

- ・キューがクラスター・キューであり、この値が MQPUT 呼び出しに指定されている場合、使用される書き込み応答タイプは、メッセージが置かれているキューの特定のインスタンスを所有する宛先キュー・マネージャーで定義された **DEFPRESP** 属性から取得されます。

クラスター・キューの複数インスタンスがあり、それらがこの属性で異なる場合、いずれかの値が選出されます。どの値が使用されるかは予測できません。したがって、この属性はすべてのインスタンスで同じ値に設定する必要があります。そうしない場合、エラー・メッセージ AMQ9407 がキュー・マネージャー・ログに発行されます。 [How are destination object attributes resolved for aliases, remote and cluster queues?](#) も参照してください。

- ・キューがクラスター・キューではなく、この値が MQPUT 呼び出しに指定されている場合、使用される書き込み応答タイプは、宛先キュー・マネージャーがリモートであっても、ローカル・キュー・マネージャーで定義された **DEFPRESP** 属性から取得されます。

IBM WebSphere MQ 7.0 より前のレベルのキュー・マネージャーに接続したクライアント・アプリケーションは、MQPMO_SYNC_RESPONSE が指定された場合と同じように動作します。

このオプションを MQPUT1 呼び出しで指定した場合、要求がサーバーに送信される前には、DEFPRESP 属性の値が分かりません。デフォルトでは、MQPUT1 呼び出しが MQPMO_SYNCPOINT を使用する場合には MQPMO_ASYNC_RESPONSE と同じように動作し、MQPMO_NO_SYNCPOINT を使用する場合には MQPMO_SYNC_RESPONSE と同じように動作します。ただし、クライアント構成ファイルで Put1DefaultAlwaysSync プロパティを設定すると、このデフォルトの動作を指定変更できます ([クライアント構成ファイルの CHANNELS スタンザを参照](#))。

MQPMO_RESPONSE_AS_TOPIC_DEF

MQPMO_RESPONSE_AS_TOPIC_DEF は、トピック・オブジェクトについて使用する MQPMO_RESPONSE_AS_Q_DEF と同義です。

その他の選択肢

以下のオプションは、許可検査、キュー・マネージャーが静止しているときに発生するイベント、キューおよびキュー・マネージャーの名前の解決を制御します。

MQPMO_ALTERNATE_USER_AUTHORITY

MQPMO_ALTERNATE_USER_AUTHORITY は、MQPUT1 呼び出しの **ObjDesc** パラメーターの *AlternateUserId* フィールドに、キューにメッセージを書き込む権限の妥当性検査に使用されるユーザー ID が入っていることを示します。この呼び出しが成功するのは、アプリケーションの実行に使用されているユーザー ID が許可されているかどうかに関係なく、指定されたオプションを使用してキューをオープンする権限が *AlternateUserId* にある場合のみです。(ただし、これは、指定されたコンテキスト・オプションには適用されず、検査は常に、アプリケーションが実行されているユーザー ID に対して行われます。)

このオプションは、MQPUT1 呼び出しの場合にのみ有効です。

MQPMO_FAIL_IF_QUIESCING

このオプションを指定すると、キュー・マネージャーが静止状態にある場合、MQPUT または MQPUT1 呼び出しが強制的に失敗します。

また、z/OS では、(CICS または IMS アプリケーションの) 接続が静止状態になっている場合、このオプションは MQPUT または MQPUT1 呼び出しを強制的に失敗させます。

この呼び出しは失敗し、完了コード MQCC_FAILED と理由コード MQRC_Q_MGR_QUIESCING または MQRC_CONNECTION_QUIESCING が戻ります。

MQPMO_RESOLVE_LOCAL_Q

このオプションを使用して、MQPMO 構造体の *ResolvedQName* にメッセージが書き込まれるローカル・キューの名前を入れ、*ResolvedQMgrName* にローカル・キューをホストするローカル・キュー・マネージャーの名前を入れます。MQPMO_RESOLVE_LOCAL_Q の詳細については、[MQOO_RESOLVE_LOCAL_Q](#) のトピックを参照してください。

キューへの書き込み許可が与えられている場合、MQPUT 呼び出しでこのフラグを指定するのに必要な権限があります。特殊権限は必要ありません。

デフォルトの選択肢

説明されているオプションを必要としない場合、以下のオプションを使用します。

MQPMO_NONE

この値は、他のオプションが指定されなかったことを示すために使用します。すべてのオプションはデフォルト値であるとみなされます。MQPMO_NONE は、プログラム・ドキュメンテーションの援助機能として定義されています。このオプションを他のオプションと一緒に使用することは意図されていませんが、値がゼロであるため、そのように使用しても検出することはできません。

MQPMO_NONE は入力フィールドです。Options フィールドの初期値は MQPMO_NONE です。

Timeout (MQLONG)

これは、予約フィールドです。したがって、値に意味はありません。このフィールドの初期値は-1 です。

Context (MQHOBj)

MQPMO_PASS_IDENTITY_CONTEXT または MQPMO_PASS_ALL_CONTEXT が指定される場合、このフィールドには書き込まれるメッセージに関連するコンテキスト情報が得られる入力キュー・ハンドルが入っていないかなければなりません。

MQPMO_PASS_IDENTITY_CONTEXT も MQPMO_PASS_ALL_CONTEXT も指定されていない場合、フィールドは無視されます。

これは入力フィールドです。このフィールドの初期値は 0 です。

KnownDestCount (MQLONG)

これは、現在の MQPUT 呼び出しまたは MQPUT1 呼び出しがローカル・キューである配布リスト中のキューへの送信に成功したメッセージの数です。この数にはリモート・キューを解決するキューへ送信されたメッセージの数は含まれません。ローカル伝送キューを使用して最初にメッセージを格納する場合でも同様です。このフィールドは、配布リストにはない単一のキューにメッセージを書き込むときも設定されます。

これは出力フィールドです。このフィールドの初期値は 0 です。このフィールドは、MQPMO_VERSION_1Version 未満の場合に設定されません。

 配布リストがサポートされていないため、このフィールドは z/OS では未定義です。

UnknownDestCount (MQLONG)

これは、現在の MQPUT 呼び出しまたは MQPUT1 呼び出しがリモート・キューを解決する配布リスト中のキューへの送信に成功したメッセージの数です。キュー・マネージャーが配布リストの形式中に一時的に

保存したメッセージは、これらの配布リストが含む個々の宛先の数として数えられます。このフィールドは、配布リストにはない単一のキューにメッセージを書き込むときも設定されます。

これは出力フィールドです。このフィールドの初期値は0です。このフィールドは、MQPMO_VERSION_1Version 未満の場合に設定されません。

▶ **z/OS** 配布リストがサポートされていないため、このフィールドはz/OSでは未定義です。

InvalidDestCount (MQLONG)

これは配布リスト中のキューに送信できなかったメッセージの数です。この数にはオープンに失敗したキューの数、およびオープンには成功したがPUT操作には失敗したキューの数も含まれています。このフィールドは、配布リストにはない単一のキューにメッセージを書き込むときも設定されます。

注：このフィールドは、MQPUT呼び出しまたはMQPUT1呼び出しの**CompCode**パラメーターがMQCC_OKまたはMQCC_WARNINGの場合に設定されます。**CompCode**パラメーターがMQCC_FAILEDの場合に設定されることもありますが、アプリケーション・コード内ではこれを利用しないでください。

これは出力フィールドです。このフィールドの初期値は0です。このフィールドは、MQPMO_VERSION_1Version 未満の場合に設定されません。

▶ **z/OS** 配布リストがサポートされていないため、このフィールドはz/OSでは未定義です。

ResolvedQName (MQCHAR48)

これは、ローカル・キューが名前の解決を実行した後の宛先キュー・マネージャーの名前です。戻される名前は、*ResolvedQMgrName*によって識別されるキュー・マネージャー上に存在するキューの名前です。

オブジェクトが単一キューである場合にのみ、非ブランク値が返されます。オブジェクトが配布リストまたはトピックである場合、返される値は未定義です。

これは出力フィールドです。このフィールドの長さはMQ_Q_NAME_LENGTHによって指定されます。このフィールドの初期値は、C言語ではヌル・ストリングであり、他のプログラミング言語では48桁のブランク文字です。

ResolvedQMgrName (MQCHAR48)

これは、ローカル・キュー・マネージャーが名前の解決を実行した後の宛先キュー・マネージャーの名前です。戻される名前は、*ResolvedQName*によって識別されるキューを所有するキュー・マネージャーの名前であり、ローカル・キュー・マネージャーの名前にすることができます。

*ResolvedQName*が、ローカル・キュー・マネージャーが属するキュー共有グループが所有する共有キューである場合、*ResolvedQMgrName*はそのキュー共有グループの名前です。キューが他のキュー共有グループによって所有されている場合、*ResolvedQName*は、キュー共有グループの名前またはキュー共有グループのメンバーであるキュー・マネージャーの名前にすることができます(返される値の性質は、ローカル・キュー・マネージャーに存在するキュー定義によって異なります)。

オブジェクトが単一キューである場合にのみ、非ブランク値が返されます。オブジェクトが配布リストまたはトピックである場合、返される値は未定義です。

これは出力フィールドです。このフィールドの長さはMQ_Q_MGR_NAME_LENGTHで指定します。このフィールドの初期値は、C言語ではヌル・ストリングであり、他のプログラミング言語では48桁のブランク文字です。

RecsPresent (MQLONG)

これは、アプリケーションが提供したMQPMR書き込みメッセージ・レコードまたはMQRR応答レコードの数です。この数はメッセージが配布リストに書き込まれる場合に限り、ゼロを超えることができます。書き込みメッセージ・レコードおよび応答レコードはオプションです。アプリケーションはレコードを提供する必要はありませんが、どちらか一方のタイプのレコードを提供することができます。ただし、アプ

リケーションが両方のタイプのレコードを提供する場合は、各タイプの *RecsPresent* レコードを提供する必要があります。

RecsPresent の値は、配布リスト内の宛先の数と同じである必要はありません。提供されるレコードの数が多すぎると、超過分は使用されません。逆に提供されるレコードの数が少なすぎると、書き込みメッセージ・レコードのない宛先のメッセージ・プロパティとしてデフォルトの値が使用されます (*PutMsgRecOffset* の項を参照してください)。

RecsPresent がゼロより小さい場合、またはゼロより大きいメッセージが配布リストに書き込まれなかった場合、その呼び出しは失敗し、理由コード *MQRC_RECS_PRESENT_ERROR* が戻ります。

これは入力フィールドです。このフィールドの初期値は 0 です。Version が *MQPMO_VERSION_2* より小さい場合、このフィールドは無視されます。

PutMsgRecFields (MQLONG)

このフィールドには、アプリケーションが提供する書き込みメッセージ・レコードにどの *MQPMR* フィールドがあるかを表示するように設定するフラグが入っています。 *PutMsgRecFields* は、メッセージが配布リストに書き込まれている間のみ使用します。 *RecsPresent* がゼロの場合、または *PutMsgRecOffset* と *PutMsgRecPtr* の両方がゼロの場合、このフィールドは無視されます。

書き込みメッセージ・レコードにあるフィールドについては、キュー・マネージャーは宛先ごとに対応する書き込みメッセージ・レコードのフィールドにある値を使用します。書き込みメッセージ・レコードにないフィールドについては、キュー・マネージャーは *MQMD* 構造体にある値を使用します。

以下のフラグを 1 つ以上使用して、書き込みメッセージ・レコードにどのフィールドがあるのか表示できます。

MQPMRF_MSG_ID

メッセージ ID フィールドがある。

MQPMRF_CORREL_ID

相関 ID フィールドがある。

MQPMRF_GROUP_ID

グループ ID フィールドがある。

MQPMRF_FEEDBACK

フィードバック・フィールドがある。

MQPMRF_ACCOUNTING_TOKEN

会計トークン・フィールドがある。

このフラグを指定する場合は、*Options* フィールドに *MQPMO_SET_IDENTITY_CONTEXT* または *MQPMO_SET_ALL_CONTEXT* を指定する必要があります。この条件を満たさないと、その呼び出しは失敗し、理由コード *MQRC_PMO_RECORD_FLAGS_ERROR* が戻ります。

MQPMR フィールドがない場合は、以下を指定できます。

MQPMRF_NONE

書き込みメッセージ・レコードのフィールドがない。

この値を指定する場合は、*RecsPresent* をゼロにするか、*PutMsgRecOffset* と *PutMsgRecPtr* の両方をゼロにする必要があります。

MQPMRF_NONE は、プログラムの文書化を支援するために定義します。この定数は他の定数と組み合わせて使用するようには意図されていません。ただし、この定数の値はゼロなので、ほかの実数と組み合わせて使用しても、検出されることはありません。

PutMsgRecFields に無効なフラグが入っている場合、または書き込みメッセージ・レコードが提供されたが *PutMsgRecFields* の値が *MQPMRF_NONE* である場合、その呼び出しは失敗し、理由コード *MQRC_PMO_RECORD_FLAGS_ERROR* が戻ります。

これは入力フィールドです。フィールドの初期値は、*MQPMRF_NONE* です。Version が *MQPMO_VERSION_2* より小さい場合、このフィールドは無視されます。

PutMsgRecOffset (MQLONG)

これは、MQPMO 構造体の先頭から最初の MQPMR 書き込みメッセージ・レコードのオフセットをバイト数で表したものです。オフセットの値は、正負どちらの値にもなります。*PutMsgRecOffset* は、メッセージが配布リストに書き込まれている間だけ使用されます。*RecsPresent* がゼロの場合、このフィールドは無視されます。

メッセージが配布リストに書き込まれている間、1つまたは複数の MQPMR 書き込みメッセージ・レコードの配列は、そのメッセージに特定の特性を、宛先に応じて個別に指定するために提供されます。ここで言う特性とは、以下のとおりです。

- メッセージ ID
- 相関 ID
- グループ ID
- フィードバック値
- アカウンティング・トークン

これらプロパティをすべて指定する必要はありませんが、選択するサブセットにかかわらず、フィールドは正しい順序で指定してください。詳細については、MQPMR 構造体を参照してください。

配布リストをオープンすると、通常は MQOD で指定されたオブジェクト・レコードと同じ数のメッセージ・レコードがなければなりません。したがって、各書き込みメッセージ・レコードは、対応するオブジェクト・レコードで識別されたキューにメッセージ・プロパティを供給します。オープンに失敗した配布リストのキューには、割り当てられた書き込みメッセージ・レコードが配列の適切な位置にまだ残っています。ただし、この場合メッセージ・プロパティは無視されます。

書き込みメッセージ・レコードの数は、オブジェクト・レコードの数とは異なる可能性があります。メッセージ・レコードがオブジェクト・レコードよりも少ない場合は、書き込みメッセージ・レコードのない宛先のメッセージ・プロパティは、メッセージ記述子 MQMD の対応するフィールドから得られます。書き込みメッセージ・レコードがオブジェクト・レコードよりも多い場合は、超過分は使用されません(それでも、超過分へのアクセスは可能です)。書き込みメッセージ・レコードはオプションですが、指定する場合は、メッセージ・レコードの *RecsPresent* が必要です。

PutMsgRecOffset にオフセットを指定するか、または *PutMsgRecPtr* にアドレスを指定することによって、書き込みメッセージ・レコードを MQOD のオブジェクト・レコードと同様の方法で提供します。これを行う方法について詳しくは、449 ページの『MQOD - オブジェクト記述子』で説明している *ObjectRecOffset* フィールドを参照してください。

PutMsgRecOffset と *PutMsgRecPtr* の両方を使用することはできません。両方ともゼロでない場合、呼び出しは失敗し、理由コード MQRC_PUT_MSG_RECORDS_ERROR が戻ります。

これは入力フィールドです。このフィールドの初期値は 0 です。*Version* が MQPMO_VERSION_2 より小さい場合、このフィールドは無視されます。

ResponseRecOffset (MQLONG)

これは、MQPMO 構造体の先頭から最初の MQRR 応答レコードのオフセットをバイト数で表したものです。オフセットの値は、正負どちらの値にもなります。*ResponseRecOffset* は、メッセージが配布リストに書き込まれている間だけ使用されます。*RecsPresent* がゼロの場合、このフィールドは無視されます。

メッセージを配布リストに書き込むとき、1つ以上の MQRR 応答レコードから成る配列を設定できます。これにより、メッセージの送信が失敗したキュー (MQRR の *CompCode* フィールド) を特定し、それぞれのキューで送信が失敗した理由 (MQRR の *Reason* フィールド) を調べることができます。メッセージの送信が失敗する原因としては、キューのオープン失敗や PUT 操作の失敗などが考えられます。キュー・マネージャーが応答コードを設定するのは、呼び出しの結果が一定でない場合だけです。結果が一定でない場合とは、送信できたメッセージと送信できなかったメッセージが混在している場合や、どのメッセージの送信も失敗したけれども、失敗した理由がそれぞれ異なる場合などです。後者の場合は、呼び出しの結果として理由コード MQRC_MULTIPLE_REASONS が戻ります。すべてのキューに同じ理由コードが適用される場合は、その理由コードが MQPUT または MQPUT1 呼び出しの **Reason** パラメーターに返され、応答レコードは設定されません。

配布リストをオープンすると、通常は MQOD で指定されたオブジェクト・レコードと同じ数の応答レコードがあります。したがって各応答レコードは、対応するオブジェクト・レコードで識別されたキューへ書き込むために、必要に応じて完了コードおよび理由コードを設定します。オープンに失敗した配布リストのキューには、割り当てられた応答レコードが配列の適切な位置にまだ残っています。ただし、この場合応答レコードには PUT 操作ではなくオープン操作で発生する完了コードおよび理由コードが設定されます。

応答レコードの数は、オブジェクト・レコードの数とは異なる可能性があります。応答レコードの数がオブジェクト・レコードの数より少ない場合、アプリケーションは PUT 操作に失敗したすべての宛先、または失敗の理由を識別することができない可能性があります。応答レコードがオブジェクト・レコードよりも多い場合は、超過分は使用されません。ただし、依然として超過分へのアクセスは可能です。応答レコードはオプションですが、指定する場合は、応答レコードの *RecsPresent* が必要です。

応答レコードは、MQOD のオブジェクトレコードと同様の方法で提供します。つまり、*ResponseRecOffset* でオフセットを指定するか、*ResponseRecPtr* でアドレスを指定します。詳細な方法については、で説明されている *ObjectRecOffset* フィールド 449 ページの『MQOD - オブジェクト記述子』を参照してください。ただし、*ResponseRecOffset* または *ResponseRecPtr* のどちらか一方のみを使用します。どちらもゼロでない場合、呼び出しは失敗し、理由コード MQRC_RESPONSE_RECORDS_ERROR が戻ります。

MQPUT1 呼び出しの場合は、このフィールドをゼロにする必要があります。これは必要があれば、応答情報がオブジェクト記述子 MQOD で指定された応答レコードに戻るからです。

これは入力フィールドです。このフィールドの初期値は 0 です。*Version* が MQPMO_VERSION_2 より小さい場合、このフィールドは無視されます。

PutMsgRecPtr (MQPTR)

これは、最初の MQPMR 書き込みメッセージのアドレスです。*PutMsgRecPtr* は、メッセージが配布リストに書き込まれている間のみ使用します。*RecsPresent* がゼロの場合、このフィールドは無視されます。

put メッセージレコードを指定するには、または *PutMsgRecPtr* のいずれかを使用できます *PutMsgRecOffset* が、両方を使用することはできません。詳細はを参照してください *PutMsgRecOffset*。 *PutMsgRecPtr* を使用しない場合、ヌル・ポインターまたはヌル・バイトを設定します。

これは入力フィールドです。このフィールドの初期値は、ポインターをサポートするプログラミング言語のヌル・ポインターです。それ以外の場合は、すべてヌルのバイトのストリングです。*Version* が MQPMO_VERSION_2 より小さい場合、このフィールドは無視されます。

注：プログラミング言語がそのポインターのデータ・タイプをサポートしないプラットフォームでは、このフィールドは初期値がすべてヌルのバイト・ストリングである、適当な長さのバイト・ストリングとして宣言されます。

ResponseRecPtr (MQPTR)

これは、最初の MQRR 応答レコードのアドレスです。*ResponseRecPtr* は、メッセージが配布リストに書き込まれている間だけ使用されます。*RecsPresent* がゼロの場合、このフィールドは無視されます。

応答レコードを指定するには *ResponseRecOffset*、または *ResponseRecPtr* のいずれかを使用してください。両方を使用しないでください。詳細はを参照してください *ResponseRecOffset*。 *ResponseRecPtr* を使用しない場合、ヌル・ポインターまたはヌル・バイトを設定します。

MQPUT1 呼び出しの場合は、このフィールドをヌル・ポインターまたはヌル・バイトにする必要があります。これは必要があれば、応答情報がオブジェクト記述子 MQOD で指定された応答レコードに戻るからです。

これは入力フィールドです。このフィールドの初期値は、ポインターをサポートするプログラミング言語のヌル・ポインターです。それ以外の場合は、すべてヌルのバイトのストリングです。*Version* が MQPMO_VERSION_2 より小さい場合、このフィールドは無視されます。

注: プログラミング言語がそのポインターのデータ・タイプをサポートしないプラットフォームでは、このフィールドは初期値がすべてヌルのバイト・ストリングである、適当な長さのバイト・ストリングとして宣言されます。

OriginalMsgHandle (MQHMSG)

これはメッセージに対するオプションのハンドルです。以前にキューから取り出された可能性もあります。このハンドルは、*Action* フィールドの値を対象として使用します。[NewMsgHandle](#) も参照してください。

元のメッセージ・ハンドルの内容は、**MQPUT** または **MQPUT1** 呼び出しによって変更されません。

これは入力フィールドです。このフィールドの初期値は **MQHM_NONE** です。Version が **MQPMO_VERSION_3** より前の場合、このフィールドは無視されます。

NewMsgHandle (MQHMSG)

これは書き込まれるメッセージに対するオプションのハンドルで、*Action* フィールドの値を対象としています。これはメッセージのプロパティを定義し、指定されている場合には *OriginalMsgHandle* の値を指定変更します。

MQPUT または **MQPUT1** 呼び出しから戻るときには、ハンドルの内容は実際に書き込まれたメッセージを反映します。

これは入力フィールドです。このフィールドの初期値は **MQHM_NONE** です。Version が **MQPMO_VERSION_3** より前の場合、このフィールドは無視されます。

Action (MQLONG)

このフィールドは、実行される書き込みのタイプと、*OriginalMsgHandle* フィールドで指定されている元のメッセージと *NewMsgHandle* フィールドで指定されている新しいメッセージの間の関係を指定します。メッセージのプロパティは、*Action* の指定値に従ってキュー・マネージャーによって選択されます。

MQPUT または **MQPUT1** 呼び出しで *MsgDesc* パラメーターを使用してメッセージ記述子の内容を指定するよう選択できます。あるいは、*MsgDesc* パラメーターを指定しなかったり、**MQPMO** 構造体の *Options* フィールド中に **MQPMO_MD_FOR_OUTPUT_ONLY** を組み込んで出力専用であると指定したりできます。

MsgDesc パラメーターを指定しなかったり、出力専用と指定したりすると、新しいメッセージのメッセージ記述子は、このトピックで記述されている規則に従って、**MQPMO** のメッセージ・ハンドル・フィールドからデータを追加されます。

[コンテキスト情報の制御](#)で説明されているコンテキストの設定や引き渡しのアクティビティは、メッセージ記述子の構成後に有効になります。

正しくないアクション値が指定されると、呼び出しが失敗し、理由コード **MQRC_ACTION_ERROR** が戻されます。

以下のアクションのいずれかを指定できます。

MQACTP_NEW

新しいメッセージが書き込まれ、プログラムによって前のメッセージとの関係は指定されません。メッセージ記述子は以下のように構成されます。

- **MQPUT** または **MQPUT1** 呼び出しで *MsgDesc* が提供され、**MQPMO.Options** 中に **MQPMO_MD_FOR_OUTPUT_ONLY** がない場合は、変更されずにメッセージ記述子として使用されます。
- *MsgDesc* を指定しないか、**MQPMO.Options** 中に **MQPMO_MD_FOR_OUTPUT_ONLY** がある場合は、キュー・マネージャーは *OriginalMsgHandle* と *NewMsgHandle* のプロパティの組み合わせを使用してメッセージ記述子を生成します。新しいメッセージ・ハンドル上で明示的に設定されたどのメッセージ記述子フィールドも、元のメッセージ・ハンドルのメッセージ記述子フィールドより優先されます。

メッセージ・データは **MQPUT** または **MQPUT1** Buffer パラメーターから取られます。

MQACTP_FORWARD

以前に取り出されたメッセージが転送されます。元のメッセージ・ハンドルは、以前に取り出されたメッセージを指定します。

新しいメッセージ・ハンドルは、元のメッセージ・ハンドル中のプロパティー (メッセージ記述子中のものも含む) に対する変更を指定します。

メッセージ記述子は以下のように構成されます。

- MQPUT または MQPUT1 呼び出しで MsgDesc が提供され、MQPMO.Options 中に MQPMO_MD_FOR_OUTPUT_ONLY がいない場合は、変更されずにメッセージ記述子として使用されます。
- MsgDesc を指定しないか、MQPMO.Options 中に MQPMO_MD_FOR_OUTPUT_ONLY がある場合は、キュー・マネージャーは OriginalMsgHandle と NewMsgHandle のプロパティーの組み合わせを使用してメッセージ記述子を生成します。新しいメッセージ・ハンドル上で明示的に設定されたどのメッセージ記述子フィールドも、元のメッセージ・ハンドルのメッセージ記述子フィールドより優先されます。
- MQPMO.Options 中に MQPMO_NEW_MSG_ID または MQPMO_NEW_CORREL_ID が指定されている場合は参照されます。

メッセージ・プロパティーは以下のように構成されます。

- MQPD.CopyOptions 中に MQCOPY_FORWARD がある元のメッセージ・ハンドルからのすべてのプロパティー
- 新しいメッセージ・ハンドルからのすべてのプロパティー。元のメッセージ・ハンドル中のプロパティーと同じ名前の、新しいメッセージ・ハンドル中のプロパティーごとに、新しいメッセージ・ハンドルから値が取られます。この規則の例外となる特殊な場合が 1 つだけあり、それは新しいメッセージ・ハンドル中のプロパティーが、元のメッセージ・ハンドル中のプロパティーと同じ名前であるにも関わらず、プロパティーの値がヌルの場合です。この場合、プロパティーはメッセージから除去されます。

転送されるメッセージ・データは MQPUT または MQPUT1 Buffer パラメーターから取られます。

MQACTP_REPLY

以前に取り出されたメッセージに対して応答が行われます。元のメッセージ・ハンドルは、以前に取り出されたメッセージを指定します。

新しいメッセージ・ハンドルは、元のメッセージ・ハンドル中のプロパティー (メッセージ記述子中のものも含む) に対する変更を指定します。

メッセージ記述子は以下のように構成されます。

- MQPUT または MQPUT1 呼び出しで MsgDesc が提供され、MQPMO.Options 中に MQPMO_MD_FOR_OUTPUT_ONLY がいない場合は、変更されずにメッセージ記述子として使用されます。
- MsgDesc を指定しないか、MQPMO.Options 中に MQPMO_MD_FOR_OUTPUT_ONLY がある場合は、初期メッセージ記述子フィールドは以下のように選択されます。

表 532. 応答メッセージ・ハンドルの変換	
MQMD のフィールド	使用される値
レポート	MQRO_PASS_DISCARD_AND_EXPIRY の場合 および MQRO_DISCARD_MSG が設定されています。 MQRO_DISCARD_MSG それ以外の場合: MQRO_NONE
MsgType	MQMT_REPLY

表 532. 応答メッセージ・ハンドルの変換 (続き)	
MQMD のフィールド	使用される値
Expiry	MQRO_PASS_DISCARD_AND_EXPIRY の場合 が設定されます。 入力メッセージからコピーされる それ以外の場合: MQEI_UNLIMITED
Feedback	MQFB_NONE
MsgId	MQPMO_NEW_MSG_ID が設定されている場合: 新しいメッセージ ID が生成される MQRO_PASS_MSG_ID が設定されている場合: 入力メッセージからコピーされる それ以外の場合: MQMI_NONE
CorrelId	MQPMO_NEW_CORREL_ID が設定されている場合: 新しい相関 ID が生成される MQRO_COPY_MSG_ID_TO_CORREL_ID が設定されて いる場合: 次の項目の MsgId フィールドからコピーされます。 入力メッセージ MQRO_PASS_CORREL_ID が設定されている場合: の CorrelId フィールドからコピーされます 入力メッセージ それ以外の場合: MQCI_NONE
BackoutCount	0
ReplyToQ	ブランク
ReplyToQMgr	ブランク
GroupId	MQGI_NONE
MsgSeqNumber	1
オフセット	0
MsgFlags	MQMF_NONE
OriginalLength	MQOL_UNDEFINED

- 変換後、メッセージ記述子は新しいメッセージ・ハンドルによって変更されます。- 新しいメッセージ・ハンドル中でプロパティーとして明示的に設定されたどのメッセージ記述子フィールドも、上記のメッセージ記述子フィールドより優先されます。

メッセージ・プロパティーは以下のように構成されます。

- MQPD.CopyOptions 中に MQCOPY_REPLY がある元のメッセージ・ハンドルからのすべてのプロパティー
- 新しいメッセージ・ハンドルからのすべてのプロパティー。元のメッセージ・ハンドル中のプロパティーと同じ名前の、新しいメッセージ・ハンドル中のプロパティーごとに、新しいメッセージ・ハンドルから値が取られます。この規則の例外となる特殊な場合が 1 つだけあり、それは新しいメッセージ・ハンドル中のプロパティーが、元のメッセージ・ハンドル中のプロパティーと同じ名前であるにも関わらず、プロパティーの値がヌルの場合です。この場合、プロパティーはメッセージから除去されます。

転送されるメッセージ・データは MQPUT/MQPUT1 Buffer パラメーターから取られます。

MQACTP_REPORT

以前に取り出されたメッセージの結果として、レポートが生成されます。元のメッセージ・ハンドルは、レポート生成の原因となるメッセージを指定します。

新しいメッセージ・ハンドルは、元のメッセージ・ハンドル中のプロパティ (メッセージ記述子中のものも含む) に対する変更を指定します。

メッセージ記述子は以下のように構成されます。

- MQPUT または MQPUT1 呼び出しで MsgDesc が提供され、MQPMO.Options 中に MQPMO_MD_FOR_OUTPUT_ONLY がいない場合は、変更されずにメッセージ記述子として使用されます。
- MsgDesc を指定しないか、MQPMO.Options 中に MQPMO_MD_FOR_OUTPUT_ONLY がある場合は、初期メッセージ記述子フィールドは以下のように選択されます。

表 533. レポート・メッセージ・ハンドルの変換	
MQMD のフィールド	使用される値
レポート	MQRO_PASS_DISCARD_AND_EXPIRY の場合、および MQRO_DISCARD_MSG が設定されます: MQRO_DISCARD_MSG それ以外の場合: MQRO_NONE
MsgType	MQMT_REPORT
Expiry	MQRO_PASS_DISCARD_AND_EXPIRY の場合 が設定されます。 入力メッセージからコピーされる それ以外の場合: MQEI_UNLIMITED
MsgId	MQPMO_NEW_MSG_ID が設定されている場合: 新しいメッセージ ID が生成される MQRO_PASS_MSG_ID が設定されている場合: 入力メッセージからコピーされる それ以外の場合: MQMI_NONE
CorrelId	MQPMO_NEW_CORREL_ID が設定されている場合: 新しい相関 ID が生成される MQRO_COPY_MSG_ID_TO_CORREL_ID が設定されて いる場合: 次の項目の MsgId フィールドからコピーされます。 入力メッセージ MQRO_PASS_CORREL_ID が設定されている場合: の CorrelId フィールドからコピーされます 入力メッセージ それ以外の場合: MQCI_NONE
BackoutCount	0
ReplyToQ	ブランク
ReplyToQMgr	ブランク

表 533. レポート・メッセージ・ハンドルの変換 (続き)	
MQMD のフィールド	使用される値
OriginalLength	BufferLength に設定される

- 変換後、メッセージ記述子は新しいメッセージ・ハンドルによって変更されます。- 新しいメッセージ・ハンドル中でプロパティーとして明示的に設定されたどのメッセージ記述子フィールドも、上記のメッセージ記述子フィールドより優先されます。

メッセージ・プロパティーは以下のように構成されます。

- MQPD.CopyOptions 中に MQCOPY_REPORT がある元のメッセージ・ハンドルからのすべてのプロパティー
- 新しいメッセージ・ハンドルからのすべてのプロパティー。元のメッセージ・ハンドル中のプロパティーと同じ名前の、新しいメッセージ・ハンドル中のプロパティーごとに、新しいメッセージ・ハンドルから値が取られます。この規則の例外となる特殊な場合が 1 つだけあり、それは新しいメッセージ・ハンドル中のプロパティーが、元のメッセージ・ハンドル中のプロパティーと同じ名前であるにも関わらず、プロパティーの値がヌルの場合です。この場合、プロパティーはメッセージから除去されます。

結果の MQMD 内の Feedback フィールドは、生成されるレポートを表します。MQFB_NONE の Feedback 値により、MQPUT または MQPUT1 呼び出しが、理由コード MQRC_FEEDBACK_ERROR で失敗します。

レポート・メッセージのユーザー・データを選択するには、IBM MQ は結果の MQMD 中の Report フィールドと Feedback フィールド、および MQPUT 呼び出しか MQPUT1 呼び出しの Buffer パラメーターと BufferLength パラメーターを参照します。

- Feedback が MQFB_COA、MQFB_COD、または MQFB_EXPIRATION の場合は、Report の値が検査されます。
- 以下のいずれかの場合が当てはまると、BufferLength の長さに関する Buffer からのメッセージ・データ全体が使用されます。
 - Feedback が MQFB_EXPIRATION で、Report に MQRO_EXPIRATION_WITH_FULL_DATA が含まれている
 - Feedback が MQFB_COD で、Report に MQRO_COD_WITH_FULL_DATA が含まれている
 - Feedback が MQFB_COA で、Report に MQRO_COA_WITH_FULL_DATA が含まれている
- 以下のいずれかの場合が当てはまると、Buffer からのメッセージの先頭 100 バイト (または、100 未満の場合は BufferLength) が使用されます。
 - Feedback が MQFB_EXPIRATION で、Report に MQRO_EXPIRATION_WITH_DATA が含まれている
 - Feedback が MQFB_COD で、Report に MQRO_COD_WITH_DATA が含まれている
 - Feedback が MQFB_COA で、Report に MQRO_COA_WITH_DATA が含まれている
- Feedback が MQFB_EXPIRATION、MQFB_COD、または MQFB_COA で、この Feedback 値に関係のある *_WITH_FULL_DATA または *_WITH_DATA オプションが Report に含まれていない場合は、メッセージにユーザー・データが組み込まれません。
- Feedback の値が上記のリストと違う場合は、Buffer と BufferLength は通常どおりに使用されます。

以下の表には、前述のリストで記述されているユーザー・データの導出も示されています。

表 534. ユーザー・データのソース			
	MQFB_COA	MQFB_COD	MQFB_EXPIRATION
MQRO_EXPIRATION_WITH_FULL_DATA	なし	なし	Buffer(Bufferlength)
MQRO_COD_WITH_FULL_DATA	なし	Buffer(Bufferlength)	なし

表 534. ユーザー・データのソース (続き)

	MQFB_COA	MQFB_COD	MQFB_EXPIRATION
MQRO_COA_WITH_FULL_DATA	Buffer(Bufferlength)	なし	なし
MQRO_EXPIRATION_WITH_DATA	なし	なし	Buffer(First 100 bytes)
MQRO_COD_WITH_DATA	なし	Buffer(First 100 bytes)	なし
MQRO_COA_WITH_DATA	Buffer(First 100 bytes)	なし	なし

PubLevel (MQLONG)

このフィールドの初期値は9です。このパブリケーションのターゲットになるサブスクリプションのレベル。最高の SubLevel がこの値以下のサブスクリプションのみが、このパブリケーションを受信します。この値はゼロから9までの範囲内でなければなりません。ゼロは最低レベルです。しかし、パブリケーションが保持されると PubLevel 1 でリパブリッシュされるため、パブリケーションは高いレベルのサブスクライバーで使用することはできなくなります。

詳細については、[インターセプト・パブリケーション](#)を参照してください。

MQPMR - 書き込みメッセージ・レコード

配布リストにメッセージを配置する際、単一の宛先に対して様々なメッセージプロパティを指定するには、MQPMR 構造を使用します。MQPMR は、MQPUT 呼び出しおよび MQPUT1 呼び出しの入出力構造体です。

可用性

MQPMR 構造は、以下のプラットフォームで利用可能です：

-  AIX
-  IBM i
-  Linux
-  Windows

および、これらのシステムに接続された IBM MQ クライアント。

文字セットとエンコード

MQPMR 内のデータは、キュー・マネージャ **CodedCharSetId** 属性によって指定される文字セットおよびローカル・キュー・マネージャのエンコーディング (MQENC_NATIVE によって指定される) でなければならない。しかし、アプリケーションが MQ クライアントとして実行している場合、構造体はクライアントの文字セット内およびエンコード内になければなりません。

使用法

MQPUT または MQPUT1 呼び出しでこれらの構造体の配列を提供することで、配布リスト内の各宛先キューに対して異なる値を指定できます。入力だけのフィールドもあれば、入出力とも可能なフィールドもあります。

注：この構造体は、固定の配置がないという点で通常とは異なっています。この構造体のフィールドはオプションであり、各フィールドの有無は MQPMO のフラグ *PutMsgRecFields* フィールドで示されます。ここにあるフィールドは **必ず、次の順序で指定する必要があります**。

- *MsgId*
- *CorrelId*
- *GroupId*
- *Feedback*
- *AccountingToken*

ここにはないフィールドはレコードのスペースをとりません。

MQPMR には固定したレイアウトはないので、サポートされるプログラミング言語用のヘッダー・ファイル、COPY ファイル、および INCLUDE ファイルで提供される定義は提供されていません。したがってアプリケーション・プログラマーは、そのアプリケーションに必要なフィールドを含む宣言を作成し、*PutMsgRecFields* のフラグを立てて、そこにあるフィールドを表示する必要があります。

フィールド

この構造体には初期値が定義されていません。サポートされているプログラミング言語のヘッダーファイル、COPY および INCLUDE ファイルに構造体の宣言が提供されていないためです。すべてのフィールドが必須の場合に構造体を宣言する方法の例を示します。

表 535. MQPMR のフィールド	
フィールド名	フィールドの説明
<u>MsgId</u>	メッセージ ID
<u>CorrelId</u>	相関 ID
<u>GroupId</u>	グループ ID
<u>フィードバック</u>	フィードバックまたは理由コード
<u>AccountingToken</u>	アカウントティング・トークン

言語ごとの宣言

MQPMR の C 宣言

```
typedef struct tagMQPMR MQPMR;
struct tagMQPMR {
    MQBYTE24  MsgId;           /* Message identifier */
    MQBYTE24  CorrelId;        /* Correlation identifier */
    MQBYTE24  GroupId;         /* Group identifier */
    MQLONG    Feedback;        /* Feedback or reason code */
    MQBYTE32  AccountingToken; /* Accounting token */
};
```

MQPMR の COBOL 宣言

```
**      MQPMR structure
10 MQPMR.
**      Message identifier
15 MQPMR-MSGID          PIC X(24).
**      Correlation identifier
15 MQPMR-CORRELID       PIC X(24).
**      Group identifier
15 MQPMR-GROUPID        PIC X(24).
**      Feedback or reason code
15 MQPMR-FEEDBACK       PIC S9(9) BINARY.
**      Accounting token
15 MQPMR-ACCOUNTINGTOKEN PIC X(32).
```

MQPMR の PL/I 宣言

```
dcl
  1 MQPMR based,
  3 MsgId          char(24),          /* Message identifier */
```

```

3 CorrelId      char(24),      /* Correlation identifier */
3 GroupId      char(24),      /* Group identifier */
3 Feedback     fixed bin(31), /* Feedback or reason code */
3 AccountingToken char(32);    /* Accounting token */

```

MQPMR の Visual Basic 宣言

```

Type MQPMR
    MsgId      As MQBYTE24 'Message identifier'
    CorrelId   As MQBYTE24 'Correlation identifier'
    GroupId    As MQBYTE24 'Group identifier'
    Feedback   As Long      'Feedback or reason code'
    AccountingToken As MQBYTE32 'Accounting token'
End Type

```

MQPMR のフィールド詳細

配布リストにメッセージを送信する際、単一の宛先に対する様々なメッセージプロパティを指定するには、MQPMR Put-message レコード構造を使用します。MQPMR は、MQPUT 呼び出しおよび MQPUT1 呼び出しの入出力構造体です。

フィールド

この構造体には初期値が定義されていません。サポートされているプログラミング言語のヘッダーファイル、COPY および INCLUDE ファイルに構造体の宣言が提供されていないためです。すべてのフィールドが必須の場合に構造体を宣言する方法の例を示します。

表 536. MQPMR のフィールド	
フィールド名	フィールドの説明
MsgId	メッセージ ID
CorrelId	相関 ID
GroupId	グループ ID
フィードバック	フィードバックまたは理由コード
AccountingToken	アカウンティング・トークン

MsgId (MQBYTE24)

これは、キューに送るメッセージに使用されるメッセージ ID です。このキューの名前は、MQOPEN 呼び出しまたは MQPUT1 呼び出しで与えられた MQOR 構造体の配列内の対応する要素に指定されていた名前です。メッセージ ID は、1 つのキューに書き込むために MQMD 内の *MsgId* フィールドと同様に処理されます。

このフィールドが MQPMR レコードにない場合、または宛先と比較して MQPMR レコードが少ない場合、MQMD 内の値は *MsgId* フィールドが入っている MQPMR レコードのない宛先に使用されます。その値が MQMI_NONE の場合、宛先ごとにメッセージ ID が生成されます (つまり、2 つの宛先には同じメッセージ ID はありません)。

MQPMO_NEW_MSG_ID が指定されると、MQPMR レコードの有無にかかわらず、配布リスト中のすべての宛先に新しいメッセージ ID が生成されます。これは、MQPMO_NEW_CORREL_ID が処理される方法とは異なります (*CorrelId* フィールドを参照)。

これは入出力フィールドです。

CorrelId (MQBYTE24)

これは、キューに送るメッセージに使用される相関 ID です。このキューの名前は、MQOPEN 呼び出しまたは MQPUT1 呼び出しで与えられた MQOR 構造体の配列内の対応する要素に指定されていた名前です。メッセージ ID は、1 つのキューに書き込むために MQMD 内の *CorrelId* フィールドと同様に処理されます。

このフィールドが MQPMR レコードにない場合、または宛先と比較して MQPMR レコードが少ない場合、MQMD 内の値は *CorrelId* フィールドが入っている MQPMR レコードのない宛先に使用されます。

MQPMO_NEW_CORREL_ID が指定されると、MQPMR レコードの有無にかかわらず、配布リスト中のすべての宛先に新しい単一の 相関 ID が生成され、使用されます。これは、MQPMO_NEW_MSG_ID が処理される方法とは異なります (*MsgId* フィールドを参照)。

これは入出力フィールドです。

GroupId (MQBYTE24)

GroupId は、キューに送るメッセージに使用されるグループ ID です。このキューの名前は、MQOPEN 呼び出しまたは MQPUT1 呼び出しで与えられた MQOR 構造体の配列内の対応する要素に指定されていた名前です。メッセージ ID は、1 つのキューに書き込むために MQMD 内の *GroupId* フィールドと同様に処理されます。

このフィールドが MQPMR レコードにない場合、または宛先と比較して MQPMR レコードが少ない場合、MQMD 内の値は *GroupId* フィールドが入っている MQPMR レコードのない宛先に使用されます。この値はキューでの物理順序で説明されているように処理されますが、以下の点が異なります。

- GroupId は、QMName とタイム・スタンプから作成されます。そのため、GroupId を固有にするためには、キュー・マネージャーの名前も固有にしてください。また、キュー・マネージャーのマシンで時刻を設定し直さないでください。
- 新しいグループ ID が使用された場合、キュー・マネージャーは宛先ごとに異なるグループ ID を生成します (つまり、2 つの宛先に同じグループ ID はありません)。
- フィールド内の値が使用された場合、呼び出しは失敗し、理由コード MQRC_GROUP_ID_ERROR が戻ります。

これは入出力フィールドです。

Feedback (MQLONG)

これは、キューに送信するメッセージに使用されるフィードバック・コードです。送信対象のキューは、MQOPEN 呼び出しまたは MQPUT1 呼び出しに提供された MQOR 構造体の配列内の対応する要素によって指定された名前を持っているものです。メッセージ ID は、1 つのキューに書き込むために MQMD 内の *Feedback* フィールドと同様に処理されます。

このフィールドがない場合、MQMD 内の値が使用されます。

これは入力フィールドです。

AccountingToken (MQBYTE32)

これはキューに送るメッセージに使用される会計トークンです。このキューの名前は、MQOPEN 呼び出しまたは MQPUT1 呼び出しで与えられた MQOR 構造体の配列内の対応する要素に指定されていた名前です。メッセージ ID は、1 つのキューに書き込むために MQMD 内の *AccountingToken* フィールドと同様に処理されます。このフィールドの内容については 385 ページの『MQMD - メッセージ記述子』、*AccountingToken* の記述を参照してください。

このフィールドがない場合、MQMD 内の値が使用されます。

これは入力フィールドです。

MQRFH - 規則およびフォーマット・ヘッダー

MQRFH 構造体は、規則と書式ヘッダーのレイアウトを定義します。このヘッダーは、名前と値のペアの形式でストリング・データを送信するために使用します。

可用性

すべての IBM MQ システム、およびこれらのシステムに接続された IBM MQ MQI clients。

形式名

MQFMT_RF_HEADER

文字セットとエンコード

MQRFH 構造体のフィールド（を含む *NameValueString*）は、MQRFH に先行するヘッダー構造体の *CodedCharSetId*、*Encoding* および *StrucLength* フィールド、または MQRFH がアプリケーションメッセージデータの先頭にある場合の MQMD 構造体の対応するフィールドによって指定される文字セットおよびエンコーディングで記述される。

文字セットは、キュー名に有効な文字用の 1 バイト文字を持つ文字セットでなければなりません。

フィールド

注：次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 537. MQRFH における MQRFH のフィールド		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
<u>StrucId</u> (構造識別子)	MQRFH_STRUC_ID	'RFH↵'
<u>バージョン</u> (構造バージョン番号)	MQRFH_VERSION_1	1
<u>StrucLength</u> (MQRFH 構造体のバイト単位の長さ)	MQRFH_STRUC_LENGTH_FIXED	32
<u>エンコーディング</u> (続くデータの数字による <i>NameValueString</i> 符号化)	MQENC_NATIVE	環境に依存
<u>CodedCharSetId</u> (続くデータの文字 <i>NameValueString</i> セット識別子を指定する)	MQCCSI_UNDEFINED	0
<u>フォーマット</u> (続くデータのフォーマット名 <i>NameValueString</i>)	MQFMT_NONE	ブランク
<u>フラグ</u> (フラグ)	MQRFH_NONE	0
<u>NameValueString</u> (名前と値のペアを含む可変長文字列)	なし	なし
注： 1. 記号↵は、単一のブランク文字を表します。 2. C プログラミング言語では、マクロ変数 MQRFH_DEFAULT には、表にリストされている値が含まれています。この変数を以下の方法で使用すると、構造体のフィールドに初期値を設定できます。 <pre>MQRFH MyRFH = {MQRFH_DEFAULT};</pre>		

言語ごとの宣言

MQRFH の C 宣言

```
typedef struct tagMQRFH MQRFH;
struct tagMQRFH {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQLONG     StrucLength;      /* Total length of MQRFH including
                                NameValueString */
    MQLONG     Encoding;         /* Numeric encoding of data that follows
                                NameValueString */
    MQLONG     CodedCharSetId;   /* Character set identifier of data that
                                follows NameValueString */
}
```

```

MQCHAR8  Format;          /* Format name of data that follows
                           NameValueString */
MQLONG   Flags;          /* Flags */
};

```

MQRFH の COBOL 宣言

```

** MQRFH structure
10 MQRFH.
** Structure identifier
15 MQRFH-STRUCID      PIC X(4).
** Structure version number
15 MQRFH-VERSION      PIC S9(9) BINARY.
** Total length of MQRFH including NAMEVALUESTRING
15 MQRFH-STRUCLENGTH  PIC S9(9) BINARY.
** Numeric encoding of data that follows NAMEVALUESTRING
15 MQRFH-ENCODING     PIC S9(9) BINARY.
** Character set identifier of data that follows NAMEVALUESTRING
15 MQRFH-CODEDCHARSETID PIC S9(9) BINARY.
** Format name of data that follows NAMEVALUESTRING
15 MQRFH-FORMAT       PIC X(8).
** Flags
15 MQRFH-FLAGS        PIC S9(9) BINARY.

```

MQRFH の PL/I 宣言

```

dcl
1 MQRFH based,
3 StrucId      char(4),          /* Structure identifier */
3 Version      fixed bin(31), /* Structure version number */
3 StrucLength  fixed bin(31), /* Total length of MQRFH including
                               NameValueString */
3 Encoding     fixed bin(31), /* Numeric encoding of data that
                               follows NameValueString */
3 CodedCharSetId fixed bin(31), /* Character set identifier of data
                               that follows NameValueString */
3 Format        char(8),          /* Format name of data that follows
                               NameValueString */
3 Flags        fixed bin(31); /* Flags */

```

MQRFH の高水準アセンブラ宣言

```

MQRFH          DSECT
MQRFH_STRUCID  DS   CL4  Structure identifier
MQRFH_VERSION  DS   F    Structure version number
MQRFH_STRUCLENGTH DS   F    Total length of MQRFH including
*                               NAMEVALUESTRING
MQRFH_ENCODING DS   F    Numeric encoding of data that follows
*                               NAMEVALUESTRING
MQRFH_CODEDCHARSETID DS   F    Character set identifier of data that
*                               follows NAMEVALUESTRING
MQRFH_FORMAT   DS   CL8  Format name of data that follows
*                               NAMEVALUESTRING
MQRFH_FLAGS    DS   F    Flags
*
MQRFH_LENGTH   EQU  *-MQRFH
ORG MQRFH
MQRFH_AREA     DS   CL(MQRFH_LENGTH)

```

MQRFH の Visual Basic 宣言

```

Type MQRFH
StrucId      As String*4 'Structure identifier'
Version      As Long     'Structure version number'
StrucLength  As Long     'Total length of MQRFH including
                          'NameValueString'
Encoding     As Long     'Numeric encoding of data that follows
                          'NameValueString'
CodedCharSetId As Long   'Character set identifier of data that
                          'follows NameValueString'
Format       As String*8 'Format name of data that follows
                          'NameValueString'
Flags        As Long     'Flags'
End Type

```


MQRFH のフィールド詳細

MQRFH ルールおよび書式設定ヘッダー構造は、ルールおよび書式設定ヘッダーのレイアウトを定義します。このヘッダーは、名前と値のペアの形式でストリング・データを送信するために使用します。

フィールド

注：次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 538. MQRFH のフィールド		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
StrucId (構造識別子)	MQRFH_STRUC_ID	'RFH↵'
バージョン (構造バージョン番号)	MQRFH_VERSION_1	1
StrucLength (MQRFH 構造体のバイト単位の長さ)	MQRFH_STRUC_LENGTH_FIXED	32
エンコーディング (続くデータの数字による NameValueString 符号化)	MQENC_NATIVE	環境に依存
CodedCharSetId (続くデータの文字 NameValueString セット識別子を指定する)	MQCCSI_UNDEFINED	0
フォーマット (続くデータのフォーマット名 NameValueString)	MQFMT_NONE	ブランク
フラグ (フラグ)	MQRFH_NONE	0
NameValueString (名前と値のペアを含む可変長文字列)	なし	なし
注： 1. 記号↵は、単一のブランク文字を表します。 2. C プログラミング言語では、マクロ変数 MQRFH_DEFAULT には、表にリストされている値が含まれています。この変数を以下の方法で使用すると、構造体のフィールドに初期値を設定できます。 <pre>MQRFH MyRFH = {MQRFH_DEFAULT};</pre>		

StrucId (MQCHAR4)

これはルールおよび書式設定ヘッダー構造の構造識別子です。常に入力フィールドです。その値は MQRFH_STRUC_ID です。

値は次のものでなければなりません。

MQRFH_STRUC_ID

ルールおよび書式ヘッダー構造の識別子。

C プログラミング言語では、定数 MQRFH_STRUC_ID_ARRAY も定義されている。これは MQRFH_STRUC_ID と同じ値を持ちますが、文字列ではなく文字の配列です。

Version (MQLONG)

これは構造体のバージョン番号です。値は以下のものでなければなりません。

MQRFH_VERSION_1

バージョン 1 の規則および書式ヘッダー構造体。

このフィールドの初期値は、MQRFH_VERSION_1 です。

StrucLength (MQLONG)

これは、構造体の末尾にある *NameValueString* フィールドを含む、MQRFH 構造体の長さ (バイト単位) です。長さには、*NameValueString* フィールドに続くユーザー・データは含まれません。

一部の環境でユーザー・データを変換する際の問題を回避するために、*StrucLength* は 4 の倍数でなければなりません。

次の定数は、構造体の固定部分の長さ、つまり *NameValueString* フィールドを除いた長さを示します。

MQRFH_STRUC_LENGTH_FIXED

MQRFH 構造体の長さ。

このフィールドの初期値は、MQRFH_STRUC_LENGTH_FIXED です。

Encoding (MQLONG)

これは、*NameValueString* のあとに続くデータの数値エンコードを指定します。MQRFH 構造体自体の数値データには適用されません。

MQPUT または MQPUT1 呼び出しでは、アプリケーションは、このフィールドをデータに適切な値に設定する必要があります。

このフィールドの初期値は MQENC_NATIVE です。

CodedCharSetId (MQLONG)

これは、*NameValueString* のあとに続くデータの文字セット ID を指定します。MQRFH 構造体自体の文字データには適用されません。

MQPUT または MQPUT1 呼び出しでは、アプリケーションは、このフィールドをデータに適切な値に設定する必要があります。以下のような特別な値を使用することができます。

MQCCSI_INHERIT

この構造体の後に続くデータの文字データは、この構造体に設定されているのと同じ文字セットになります。

キュー・マネージャーは、メッセージで送信される構造体の中のこの値を、構造体の実際の文字セット ID に変更します。エラーが発生しない限り、値 MQCCSI_INHERIT が MQGET 呼び出しによって返されることはありません。

MQCCSI_INHERIT は、MQMD の *PutApplType* フィールドの値が MQAT_BROKER である場合は使用できません。

このフィールドの初期値は MQCCSI_UNDEFINED です。

Format (MQCHAR8)

これは、*NameValueString* に続くデータの形式名を指定します。

MQPUT または MQPUT1 呼び出しでは、アプリケーションは、このフィールドをデータに適切な値に設定する必要があります。このフィールドのコーディングの規則は、MQMD の *Format* フィールドの場合と同じです。

このフィールドの初期値は MQFMT_NONE です。

フラグ (MQLONG)

以下のように指定できます。

MQRFH_NONE

フラグなし。

このフィールドの初期値は、MQRFH_NONE です。

NameValueString (MQCHARn)

これは、以下の書式で名前と値の組を含む可変長文字ストリングです。

```
name1 value1 name2 value2 name3 value3 ...
```

各名前および値は、1 つ以上の空白文字で、隣接する名前または値から分離されていなければなりません。これらの空白は意味を持ちません。名前または値に有効な空白を含めるには、その名前または値を二重引用符で囲みます。開き二重引用符とそれに対応する閉じ二重引用符の間にあるすべての文字は、有効なものとして扱われます。次の例では、名前は FAMOUS_WORDS で、値は Hello World です。

```
FAMOUS_WORDS "Hello World"
```

名前または値には、ヌル文字以外の任意の文字を含めることができます (ヌル文字は *NameValueString* の区切り文字として機能します)。しかし、インターオペラビリティを維持するためにアプリケーションは名前を次の文字に制限することができます。

- 先頭文字: 大文字または小文字の英字 (A から Z まで、または a から z まで)、または下線。
- 後続文字: 大文字または小文字の英字、10 進数字 (0 から 9 まで)、下線、ハイフン、またはドット。

名前または値に 1 つ以上の二重引用符を含める場合は、その名前または値を二重引用符で囲み、ストリング内のそれぞれの二重引用符を二重にする必要があります。

```
Famous_Words "The program displayed ""Hello World"""
```

名前と値では大文字小文字は区別されます。つまり、小文字は大文字と同じであるとは見なされません。例えば、FAMOUS_WORDS と Famous_Words は 2 つの異なる名前です。

NameValueString のバイト単位での長さは、*StrucLength* から MQRFH_STRUC_LENGTH_FIXED を引いた長さと等しくなります。いくつかの環境でのユーザー・データ変換時の問題を回避するために、この長さを 4 の倍数にします。*NameValueString* は、この長さまで空白を埋め込むか、またはストリング内の最後の有効な文字に続いてヌル文字を入れて終了します。ヌル文字とそれに続くバイトは、指定された *NameValueString* の長さまで、無視されます。

注: このフィールドの長さは固定されていないので、このフィールドは、サポートされるプログラミング言語に提供される構造体の宣言から省略されます。

MQRFH2 - 規則および書式ヘッダー 2

MQRFH2 ヘッダーは MQRFH ヘッダーに基づきますが、Unicode ストリングを変換せずに転送したり、数値データ・タイプを格納したりすることができます。MQRFH2 構造体は、バージョン 2 の規則と書式ヘッダーのフォーマットを定義します。このヘッダーは、XML に似た構文でエンコードされたデータを送信するために使用します。メッセージには、一連の複数の MQRFH2 構造体を入れることができ、その中の最後の MQRFH2 構造体には、オプションでユーザー・データを続けることができます。

可用性

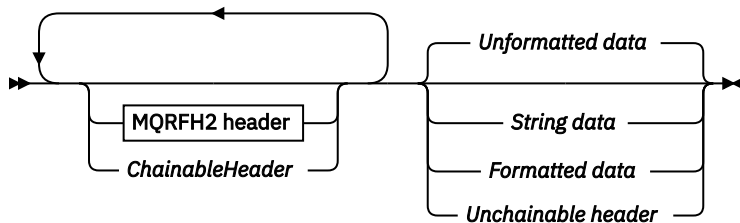
すべての IBM MQ システム、およびこれらのシステムに接続された IBM MQ MQI clients。

形式名

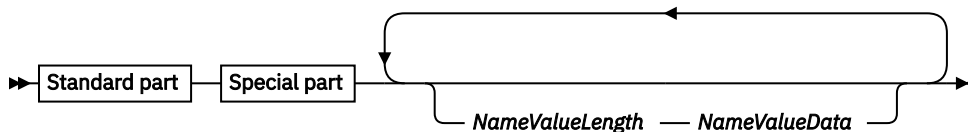
MQFMT_RF_HEADER_2

Syntax

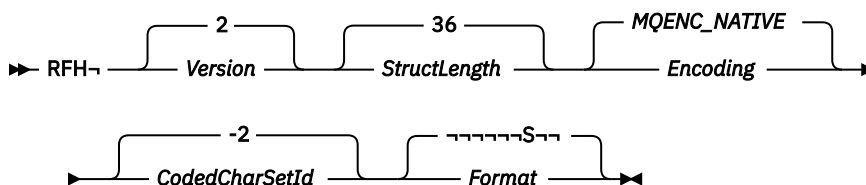
IBM MQ Message



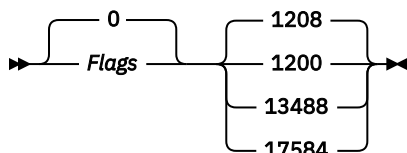
MQRFH2 header



Standard part



Special part



文字セットとエンコード

MQRFH2 構造体で使用する文字セットおよびエンコードには、次の特別な規則が適用されます。

- **NameValueData** 以外のフィールドは、文字セットおよび文字のエンコード方式に従っています。これらは MQRFH2 の前に来るヘッダー構造体の **CodedCharSetId** フィールドおよび **Encoding** フィールドによって指定されます。また、MQRFH2 がアプリケーション・メッセージ・データの先頭にある場合、MQMD 構造体の上記 2 つのフィールドによって指定されます。

文字セットは、キュー名に有効な文字用の 1 バイト文字を持つ文字セットでなければなりません。

MQGET 呼び出しで MQGMO_CONVERT が指定される場合、キュー・マネージャーは、**NameValueData** 以外の MQRFH2 フィールドを、要求された文字セットおよびエンコードに変換します。

- **NameValueData** は、フィールド **NameValueCCSID** によって指定された文字セットに含まれています。**NameValueCCSID** については、リストに挙げられた Unicode 文字セットだけが有効です。詳細については、**NameValueCCSID** の説明を参照してください。

エンコードによって表記が変わる文字セットもあります。**NameValueCCSID** がこれらの文字セットの 1 つである場合、**NameValueData** は MQRFH2 のその他のフィールドと同じエンコードでなければなりません。

MQGMO_CONVERT が MQGET 呼び出しで指定されると、キュー・マネージャーは **NameValueData** を要求されるエンコード方式に変換します。ただし、文字セットは変更しません。

フィールド

注: 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 539. フィールド名 (MQRFH2) とフィールド名 (MQRFH2)		
フィールド名	定数の名前	定数の値
StrucId (構造識別子)	MQRFH_STRUC_ID	'RFH-'
バージョン (構造バージョン番号)	MQRFH_VERSION_2	2
StrucLength (MQRFH2 構造体のバイト単位の長さ)	MQRFH_STRUC_LENGTH_FIXED_2	36
エンコーディング (最後の NameValueData フィールドに続くデータの数字エンコーディング)	MQENC_NATIVE	環境に依存
CodedCharSetId (最後の NameValueData フィールドに続くデータの文字セット識別子)	MQCCSI_INHERIT	-2
フォーマット (最後の NameValueData フィールドに続くデータのフォーマット名)	MQFMT_NONE	ブランク
フラグ (フラグ)	MQRFH_NONE	0
NameValueCCSID (フィールド NameValueData 内のデータの符号化文字セット識別子)	なし	1208
NameValueLength (フィールド NameValueData 内のデータのバイト数)	なし	None
NameValueData (メッセージプロパティの名前-値ペア)	なし	なし
注: 1. 記号-は、単一のブランク文字を表します。 2. C プログラミング言語では、マクロ変数 MQRFH2_DEFAULT には、表にリストされている値が含まれます。このマクロ変数を以下の方法で使用して、構造体のフィールドに初期値を設定します。		
<pre>MQRFH2 MyRFH2 = {MQRFH2_DEFAULT};</pre>		

言語ごとの宣言

MQRFH2 の C 宣言

```
typedef struct tagMQRFH2 MQRFH2;
struct tagMQRFH2 {
    MQCHAR4 StrucId;          /* Structure identifier */
    MQLONG Version;          /* Structure version number */
};
```

```

MQLONG   StrucLength;      /* Total length of MQRFH2 including all
                             NameValueLength and NameValueData
                             fields */
MQLONG   Encoding;        /* Numeric encoding of data that follows
                             last NameValueData field */
MQLONG   CodedCharSetId;   /* Character set identifier of data that
                             follows last NameValueData field */
MQCHAR8   Format;          /* Format name of data that follows last
                             NameValueData field */
MQLONG   Flags;           /* Flags */
MQLONG   NameValueCCSID;   /* Character set identifier of
                             NameValueData */
};

```

MQRFH2 の COBOL 宣言

```

**  MQRFH2 structure
10  MQRFH2.
**    Structure identifier
15  MQRFH2-STRUCID          PIC X(4).
**    Structure version number
15  MQRFH2-VERSION         PIC S9(9) BINARY.
**    Total length of MQRFH2 including all NAMEVALUELENGTH and
**    NAMEVALUEDATA fields
15  MQRFH2-STRUCLNGTH      PIC S9(9) BINARY.
**    Numeric encoding of data that follows last NAMEVALUEDATA field
15  MQRFH2-ENCODING        PIC S9(9) BINARY.
**    Character set identifier of data that follows last NAMEVALUEDATA
**    field
15  MQRFH2-CODEDCHARSETID  PIC S9(9) BINARY.
**    Format name of data that follows last NAMEVALUEDATA field
15  MQRFH2-FORMAT          PIC X(8).
**    Flags
15  MQRFH2-FLAGS           PIC S9(9) BINARY.
**    Character set identifier of NAMEVALUEDATA
15  MQRFH2-NAMEVALUECCSID  PIC S9(9) BINARY.

```

MQRFH2 の PL/I 宣言

```

dcl
  1 MQRFH2 based,
    3 StrucId      char(4),      /* Structure identifier */
    3 Version      fixed bin(31), /* Structure version number */
    3 StrucLength  fixed bin(31), /* Total length of MQRFH2 including
                                   all NameValueLength and
                                   NameValueData fields */
    3 Encoding     fixed bin(31), /* Numeric encoding of data that
                                   follows last NameValueData field */
    3 CodedCharSetId fixed bin(31), /* Character set identifier of data
                                   that follows last NameValueData
                                   field */
    3 Format        char(8),      /* Format name of data that follows
                                   last NameValueData field */
    3 Flags        fixed bin(31), /* Flags */
    3 NameValueCCSID fixed bin(31); /* Character set identifier of
                                   NameValueData */

```

MQRFH2 の高水準アセンブラ宣言

```

MQRFH          DSECT
MQRFH_STRUCID  DS    CL4  Structure identifier
MQRFH_VERSION  DS    F    Structure version number
MQRFH_STRUCLNGTH DS    F    Total length of MQRFH2 including all
*                NAMEVALUELENGTH and NAMEVALUEDATA fields
MQRFH_ENCODING DS    F    Numeric encoding of data that follows
*                last NAMEVALUEDATA field
MQRFH_CODEDCHARSETID DS    F    Character set identifier of data that
*                follows last NAMEVALUEDATA field
MQRFH_FORMAT   DS    CL8  Format name of data that follows last
*                NAMEVALUEDATA field
MQRFH_FLAGS    DS    F    Flags
MQRFH_NAMEVALUECCSID DS    F    Character set identifier of
*                NAMEVALUEDATA
*
MQRFH_LENGTH   EQU    *-MQRFH
MQRFH_AREA     DS      CL(MQRFH_LENGTH)

```

MQRFH2 の Visual Basic 宣言

```
Type MQRFH2
    StrucId      As String*4 'Structure identifier'
    Version      As Long     'Structure version number'
    StrucLength   As Long     'Total length of MQRFH2 including all'
                                'NameValueLength and NameValueData fields'
    Encoding      As Long     'Numeric encoding of data that follows'
                                'last NameValueData field'
    CodedCharSetId As Long     'Character set identifier of data that'
                                'follows last NameValueData field'
    Format        As String*8 'Format name of data that follows last'
                                'NameValueData field'
    Flags         As Long     'Flags'
    NameValueCCSID As Long     'Character set identifier of NameValueData'
End Type
```

MQRFH2 のフィールド詳細

ルールと書式設定ヘッダー MQRFH2 2 ヘッダーはヘッダー MQRFH に基づいているが、翻訳なしで Unicode 文字列を転送でき、数値データ型を保持できる。

フィールド

注: 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 540. MQRFH2 のフィールド		
フィールド名	定数の名前	定数の値
<u>StrucId</u> (構造識別子)	MQRFH_STRUC_ID	'RFH→'
<u>バージョン</u> (構造バージョン番号)	MQRFH_VERSION_2	2
<u>StrucLength</u> (MQRFH2 構造体のバイト単位の長さ)	MQRFH_STRUC_LENGTH_FIXED_2	36
<u>エンコーディング</u> (最後の <i>NameValueData</i> フィールドに続くデータの数字エンコーディング)	MQENC_NATIVE	環境に依存
<u>CodedCharSetId</u> (最後の <i>NameValueData</i> フィールドに続くデータの文字セット識別子)	MQCCSI_INHERIT	-2
<u>フォーマット</u> (最後の <i>NameValueData</i> フィールドに続くデータのフォーマット名)	MQFMT_NONE	ブランク
<u>フラグ</u> (フラグ)	MQRFH_NONE	0
<u>NameValueCCSID</u> (フィールド <i>NameValueData</i> 内のデータの符号化文字セット識別子)	なし	1208
<u>NameValueLength</u> (フィールド <i>NameValueData</i> 内のデータのバイト数)	なし	None

表 540. MQRFH2 のフィールド (続き)		
フィールド名	定数の名前	定数の値
NameValueData (メッセージプロパティの名前-値ペア)	なし	なし
注: - 記号- は、単一のブランク文字を表します。 - C プログラミング言語では、マクロ変数 MQRFH2_DEFAULT には、表にリストされている値が含まれます。このマクロ変数を以下の方法で使用して、構造体のフィールドに初期値を設定します。 <pre>MQRFH2 MyRFH2 = {MQRFH2_DEFAULT};</pre>		

StrucId (MQCHAR4)

これはルールと書式設定ヘッダー 2 構造の構造識別子です。常に入力フィールドです。その値は MQRFH2_STRUC_ID です。

値は次のものでなければなりません。

MQRFH2_STRUC_ID

ルールおよび書式設定ヘッダー 2 構造の識別子。

C プログラミング言語では、定数 MQRFH2_STRUC_ID_ARRAY も定義されている。これは MQRFH2_STRUC_ID と同じ値を持ちますが、文字列ではなく文字の配列です。

Version (MQLONG)

これは構造体のバージョン番号です。値は以下のものでなければなりません。

MQRFH_VERSION_2

バージョン 2 の規則および書式ヘッダー構造体。

このフィールドの初期値は、MQRFH_VERSION_2 です。

StrucLength (MQLONG)

これは、構造体の最後にある、NameValueLength および NameValueData フィールドを含む、MQRFH2 構造体の長さ (バイト数) です。構造体の末尾に複数の と NameValueData フィールド NameValueLength のペアが存在することは有効であり、その順序は次の通りである：

```
length1, data1, length2, data2, ...
```

StrucLength は、構造体の最後にある、最後の NameValueData フィールドに続く場合のあるユーザー・データを含みません。

いくつかの環境でのユーザー・データ変換に関する問題を回避するために、StrucLength を 4 の倍数にする必要があります。

以下の定数は、構造体の固定部分の長さを示します。つまり、`` NameValueLength および NameValueData `` フィールドを除いた長さを指します：

MQRFH_STRUC_LENGTH_FIXED_2

MQRFH2 構造体の固定部分の長さ。

フィールドの初期値は、MQRFH_STRUC_LENGTH_FIXED_2 です。

Encoding (MQLONG)

これは、*NameValueData* フィールドのあとに続くデータの数値エンコードを指定します。MQRFH2 構造体自体の数値データには適用されません。

MQPUT または MQPUT1 呼び出しでは、アプリケーションは、このフィールドをデータに適切な値に設定する必要があります。

このフィールドの初期値は MQENC_NATIVE です。

CodedCharSetId (MQLONG)

これは、*NameValueData* フィールドのあとに続くデータの文字セット ID を指定します。MQRFH2 構造体自体の文字データには適用されません。

MQPUT または MQPUT1 呼び出しでは、アプリケーションは、このフィールドをデータに適切な値に設定する必要があります。以下のような特別な値を使用することができます。

MQCCSI_INHERIT

この構造体の後に続くデータの文字データは、この構造体に設定されているのと同じ文字セットになります。

キュー・マネージャーは、メッセージで送信される構造体の中のこの値を、構造体の実際の文字セット ID に変更します。エラーが発生しない限り、値 MQCCSI_INHERIT が MQGET 呼び出しによって返されることはありません。

MQCCSI_INHERIT は、MQMD の *PutApplType* フィールドの値が MQAT_BROKER である場合は使用できません。

このフィールドの初期値は、MQCCSI_INHERIT です。

Format (MQCHAR8)

これは、最後の *NameValueData* フィールドの後に続くデータの形式名を指定します。

MQPUT または MQPUT1 呼び出しでは、アプリケーションは、このフィールドをデータに適切な値に設定する必要があります。このフィールドのコーディングの規則は、MQMD の *Format* フィールドの場合と同じです。

このフィールドの初期値は MQFMT_NONE です。

フラグ (MQLONG)

このフィールドの初期値は MQRFH_NONE です。MQRFH_NONE の指定は必須です。

MQRFH_NONE

フラグなし。

MQRFH_INTERNAL

MQRFH2 ヘッダーには、内部的に設定されたプロパティが含まれます。

MQRFH_INTERNAL は、キュー・マネージャーにより使用されます。

先頭の 16 ビットである MQRFH_FLAGS_RESTRICTED_MASK は、キュー・マネージャーが設定するフラグ用に予約されています。ユーザーが設定するフラグは、末尾の 16 ビットで定義されます。

NameValueCCSID (MQLONG)

これは、*NameValueData* フィールド内のデータのコード化文字セット ID を指定します。これは、MQRFH2 構造体の他のストリングの文字セットとは異なります。さらに、構造体の最後で、最後の *NameValueData* フィールドに続くデータの文字セットがある場合、それとも異なることがあります。

NameValueCCSID には、次の値のどれか 1 つが必ず含まれています。

CCSID	意味
1200	UTF-16 (サポートされる最新バージョンの Unicode)
13488	UTF-16 (Unicode バージョン 2.0 サブセット)
17584	UTF-16 (Unicode バージョン 3.0 サブセット) (ユーロ記号を含む)
1208	UTF-8 (サポートされる最新バージョンの Unicode)

UTF-16 文字セットの場合、*NameValueData* のエンコード (バイト順序) は、MQRFH2 構造体の他のフィールドのエンコードと同じでなければなりません。

Unicode 基本多言語面 (Basic Multilingual Plane) の範囲を超える文字 (U+FFFF より上の文字)、UTF-16 で代理コード・ポイント (X'D800' から X'DFFF') で表された文字、および UTF-8 での 4 バイトはサポートされていません。

注: *NameValueCCSID* に上記のリストのどの値もなく、MQRFH2 構造体が MQGET 呼び出しで変換を必要とする場合、呼び出しは理由コード MQRC_SOURCE_CCSID_ERROR を伴って完了し、メッセージは未変換のままメッセージが戻されます。

このフィールドの初期値は 1208 です。

NameValueLength (MQLONG)

対応する *NameValueData* フィールドの長さ。

これは *NameValueData* フィールド内のデータの長さ (バイト数) を指定します。 *NameValueLength* は、4 の倍数でなければなりません。

注: *NameValueLength* フィールドと *NameValueData* フィールドはオプションですが、これらのフィールドが存在する場合は、ペアとして隣接している必要があります。フィールドの対は、次のように必要に応じて何回でも繰り返すことができます。

```
length1 data1 length2 data2 length3 data3
```

これらのフィールドはオプションのため、サポートされている種々のプログラム言語用に提供されている構造体の宣言からは省略されています。

NameValueData (MQCHARn)

NameValueData メッセージプロパティの名前と値のペアを含むフォルダを格納する可変長フィールドです。

フォルダーは、XML 形式の構文を使ってエンコードされるデータを含む可変長文字ストリングです。この文字ストリングの長さ (バイト) は、*NameValueData* フィールドの前にある *NameValueLength* フィールドによって指定されます。長さは 4 の倍数でなければなりません。

NameValueLength フィールドと *NameValueData* フィールドはオプションですが、これらのフィールドが存在する場合は、ペアとして隣接している必要があります。フィールドの対は、次のように必要に応じて何回でも繰り返すことができます。

```
length1 data1 length2 data2 length3 data3
```

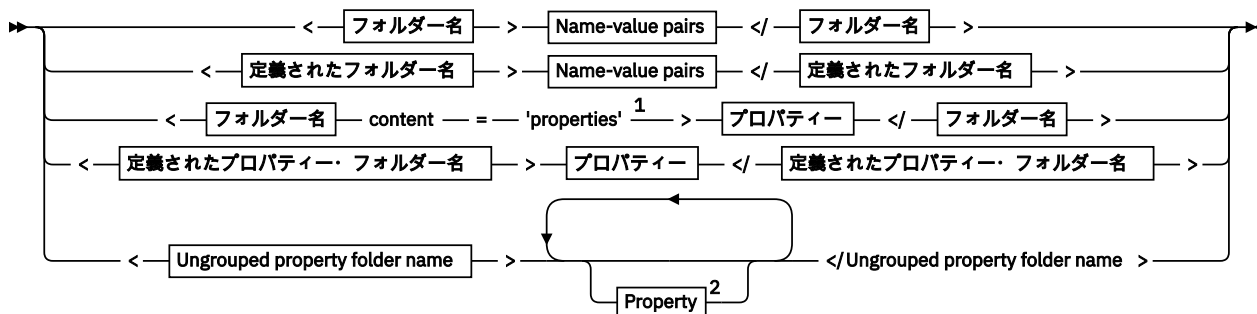
NameValueData は、MQGET 呼び出しで指定された文字セットに変換されません。MQGMO_CONVERT オプションを有効にしてメッセージを取得した場合でも、*NameValueData* は元の文字セットのままです。ただし、*NameValueData* は、MQGET 呼び出しで指定されたエンコード方式には変換されます。

注:

- これらのフィールドはオプションのため、サポートされている種々のプログラム言語用に提供されている構造体の宣言からは省略されています。

- 用語 "defined" と "reserved" は、構文図で使用されます。"Defined" は、IBM MQ が名前を使用することを意味します。"予約済み" は、名前が将来的に IBM MQ によって使用されるために予約されていることを意味します。

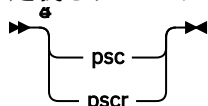
NameValueData 構文



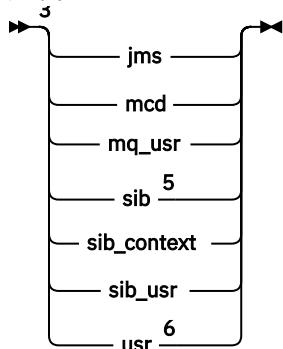
フォルダー名



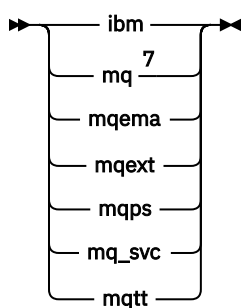
定義されたフォルダー名



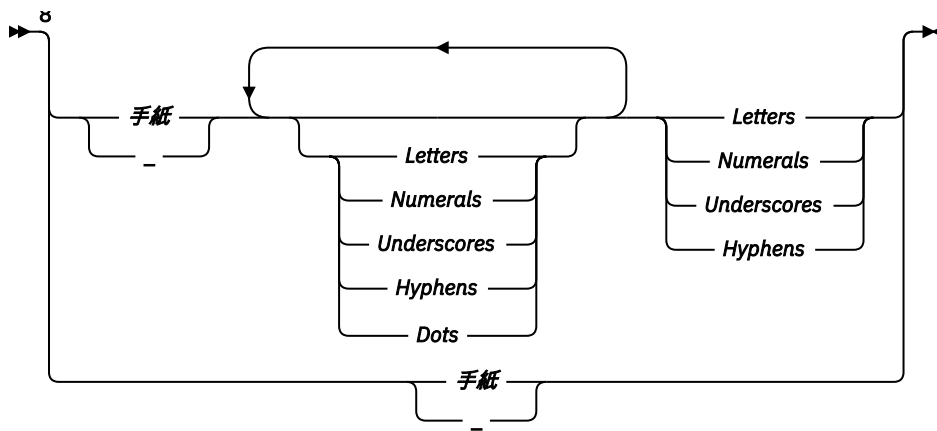
定義されたプロパティ・フォルダー名



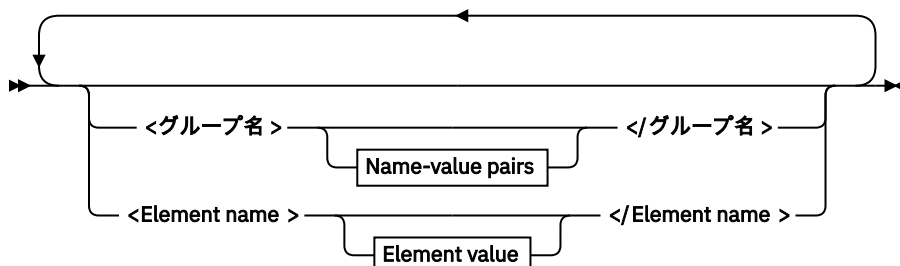
Ungrouped property folder name



名前



Name-value pairs



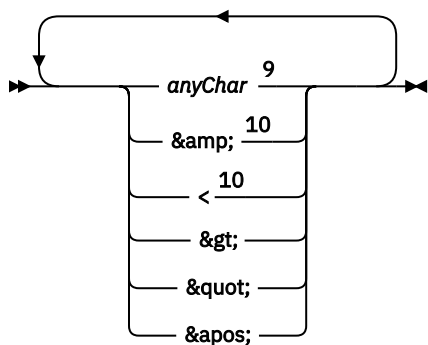
グループ名



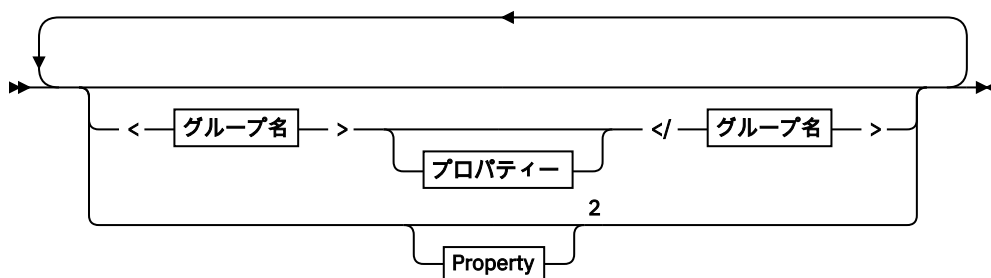
Element name



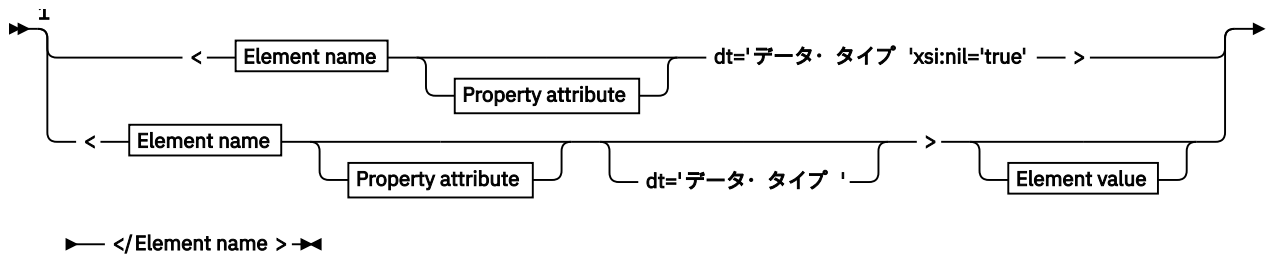
Element value



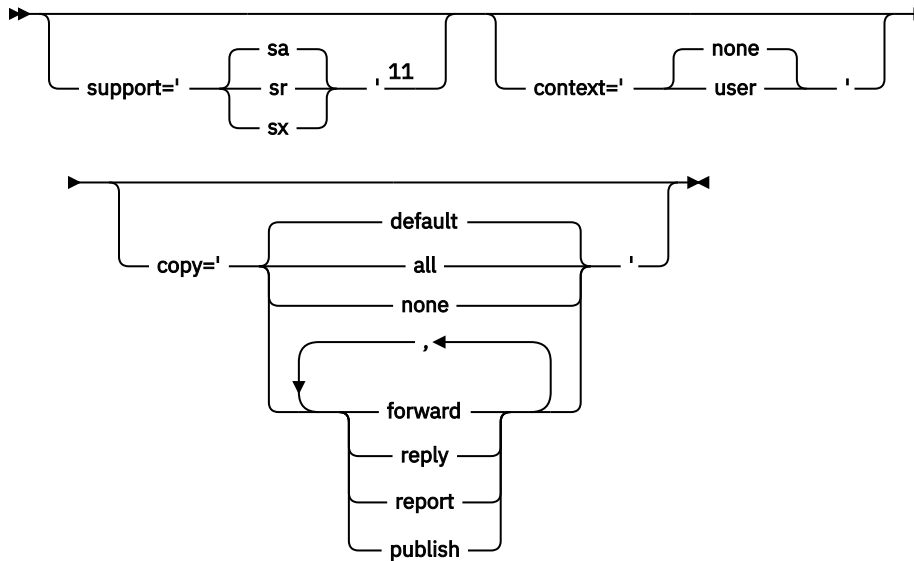
プロパティ



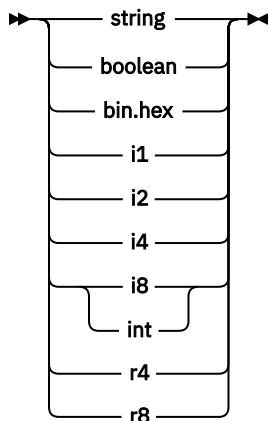
Property



Property attribute



データ・タイプ



注:

- ¹ 二重引用符または単一引用符が有効です。
- ² 無効なプロパティ名を使用しないでください。詳細は「[無効なプロパティ名](#)」を参照してください。予約済みプロパティ名は、定義された目的でのみ使用してください。詳細は「[定義済みプロパティ名](#)」を参照してください。
- ³ 名前は小文字でなければなりません。
- ⁴ 1つの psc および pscr フォルダーだけがサポートされます。
- ⁵ WebSphere Application Server サービス Integration Bus は、後続の MQRFH2 ヘッダー内の sib、sib_context、および sib_usr フォルダーを無視し、最初の MQRFH2 ヘッダー内のプロパティのみが有効です。
- ⁶ MQRFH2 には、複数の usr フォルダーが存在してはなりません。usr フォルダー内のプロパティは複数であってはなりません。

⁷ 最初の mq フォルダーにあるプロパティだけが有効です。フォルダーが UTF-8 の場合、1 バイトの UTF-8 文字のみがサポートされます。空白文字は Unicode U+0020 のみです。

⁸ 有効な文字は W3C XML 仕様で定義されており、基本的に Unicode カテゴリー Ll, Lu, Lo, Lt, Nl, Mc, Mn, Lm, および Nd で構成されます。[Unicode 文字のカテゴリ](#)を参照してください。

⁹ すべての文字が有効です。先頭および末尾の空白は要素の値の一部とみなされます。

¹⁰ 無効な文字を使用しないでください。詳細は「[無効な文字](#)」を参照してください。無効文字の代わりに、エスケープ・シーケンスを使用できます。

¹¹ support プロパティ属性は、mq フォルダーでのみ有効です。

フォルダ名

NameValueData には、単一のフォルダーが含まれます。複数のフォルダーを作成するには、複数の NameValueData フィールドを作成します。メッセージ内の単一の MQRFH2 ヘッダーに複数の NameValueData フィールドを作成できます。あるいは、複数のチェーニングされた MQRFH2 ヘッダーを作成し、それぞれに複数の NameValueData フィールドを含めることもできます。

MQRFH2 ヘッダーの順序、および NameValueData フィールドの順序は、フォルダーの論理的内容に影響を与えることはありません。1 つのメッセージに同じフォルダーが複数存在する場合、フォルダーは全体として解析されます。同じプロパティが同じフォルダーの複数インスタンスに含まれている場合、それはリストとして解析されます。

MQRFH2 の正しい構文解析は、フォルダーをメッセージに物理的に保管する代替方法の影響を受けません。

次の 4 つフォルダーはこの規則に従いません。mq、sib、sib_context、および sib_usr フォルダーの最初のインスタンスのみが構文解析されます。

チェーニングされた MQRFH2 ヘッダーの結合コンテンツで同じプロパティが複数回出現する場合は、プロパティの最初のインスタンスのみが構文解析されます。プロパティが API 呼び出し (MQSETMP など) を使用してアプリケーションから設定され、直接 MQRFH2 に追加された場合は、API 呼び出しのほうが優先されます。

フォルダー名は、名前と値のペアまたはグループを格納するフォルダーの名前です。グループおよび名前と値のペアは、フォルダー・ツリーの同じレベルで混用することができます。[514 ページの図 1](#)を参照してください。グループ名と要素名を組み合わせないでください。参照 [514 ページの図 2](#)

```
<group1><nvp1>value</nvp1></group1><group2><nvp2>value</nvp2></group2>
<group3><nvp1>value</nvp1></group3><nvp3>value</nvp3>
```

図 1. グループおよび名前と値のペアの正しい使い方

```
<group1><nvp1> value </nvp1> value </group1>
```

図 2. グループおよび名前と値のペアの間違った使い方

無効なフォルダ名または予約済みのフォルダ名を使用しないでください。詳細は「[無効なパス名](#)」および「[予約済みのフォルダ名またはプロパティフォルダ名](#)」を参照してください。定義済みフォルダ名は、その定義された目的のみに使用してください。詳細は「[定義済みフォルダ名](#)」を参照してください。

フォルダ名タグに content='properties' 属性を追加すると、そのフォルダはプロパティフォルダになります。詳細は[514 ページの図 3](#)を参照してください。

```
<myFolder></myFolder>
<myPropertyFolder content='properties'></myPropertyFolder>
```

図 3. フォルダーとプロパティ・フォルダーの例

フォルダー名では大/小文字が区別されます。フォルダー名とプロパティ・フォルダー名は同じ名前空間を共有します。それぞれ異なる名前であればなりません。Folder1 in は in Folder2 とは異なる [515 ページの図 4](#) 名前であれば [515 ページの図 5](#) なりません。

```
< Folder1 ><NVP1> value </NVP1></ Folder1 >
```

図 4. Folder1 名前空間

```
< Folder2 content='properties'>< Property1 > value </ Property1 ></ Folder2 >
```

図 5. Folder2 名前空間

異なるフォルダーに入っているグループ、プロパティ、および名前と値のペアは、それぞれ異なる名前空間を持っています。Property1 in は in Property1 とは [515 ページの図 5](#) [515 ページの図 6](#) 異なる性質である。

```
<Folder3 content='properties'>< Property1 > value </ Property1 ></Folder3>
```

図 6. Folder3 名前空間

プロパティ・フォルダーは、次の 2 つの重要な面で非プロパティ・フォルダーと異なります。

1. プロパティ・フォルダーにはプロパティが格納され、非プロパティ・フォルダーには名前と値のペアが格納されます。この 2 つのフォルダーには構文上少しの差があります。
2. メッセージのプロパティにアクセスするには、プロパティ MQI または JMS メッセージ・プロパティなどの定義されたインターフェースを使用します。インターフェースにより、MQRFH2 内のプロパティ・フォルダーが整形形式であることが保証されます。整形形式のプロパティ・フォルダーは、異なるプラットフォームおよび異なるリリースのキュー・マネージャー間で相互運用が可能です。

メッセージ・プロパティ MQI は、MQRFH2 の読み取りと書き込みを行うための堅固な方法であり、MQRFH2 を正しく構文解析する際の問題を回避します。

定義済みフォルダ名

定義されたフォルダー名は、IBM MQ またはその他の製品が使用するために予約済みのフォルダーの名前です。同じ名前のフォルダーを作成したり、フォルダーに独自の名前と値のペアを追加したりしないでください。定義されたフォルダーは、psc と pscr です。

psc および pscr は、キューに入れられたパブリッシュ/サブスクライブで使用されます。

MQMF_SEGMENT または MQMF_SEGMENTATION_ALLOWED で書き込まれたセグメント化メッセージには、定義されたフォルダー名を持つ MQRFH2 を含めることはできません。MQPUT は理由コード 2443、MQRC_SEGMENTATION_NOT_ALLOWED で失敗します。

定義済みプロパティフォルダ名

定義されたプロパティ・フォルダー名は、IBM MQ またはその他の製品が使用するためのプロパティ・フォルダーの名前です。フォルダーとその内容の名前については、[プロパティ・フォルダー](#)を参照してください。定義済みプロパティフォルダ名は、IBM MQ によって予約されているすべてのフォルダ名のサブセットです。詳細は「[予約済みフォルダ名またはプロパティフォルダ名](#)」を参照してください。

定義されたプロパティ・フォルダー内のすべての要素はプロパティです。定義されたプロパティ・フォルダーに保管されているエレメントには、content='properties' 属性があってはなりません。

プロパティを追加できるのは、定義されたプロパティ・フォルダー `usr`、`mq_usr`、および `sib_usr` に対してだけです。 `mq` や `sib` などの他のプロパティ・フォルダーでは、IBM MQ が認識できないプロパティは無視または破棄されます。

定義されたプロパティ・フォルダーそれぞれの記述には、アプリケーション・プログラムが利用できるものとして IBM MQ によって定義されたプロパティがリストされます。プロパティの中には、JMS プロパティを設定または取得して間接的にアクセスできるものもあれば、MQSETMP および MQINQMP MQI 呼び出しを使用して直接アクセスできるものもあります。

定義されたプロパティ・フォルダーには、IBM MQ によって予約済みであっても、アプリケーションからはアクセスしないというプロパティも入ります。予約済みのプロパティ名はリストされていません。 `usr`、`mq_usr`、および `sib_usr` の各プロパティ・フォルダーに、予約済みのプロパティはありません。ただし、無効なプロパティ名を持つプロパティを作成しないでください。詳細は「[無効なプロパティ名](#)」を参照してください。

プロパティフォルダ

jms

`jms` には、JMS ヘッダー・フィールドと、MQMD で完全には表現できない JMSX プロパティが含まれています。 `jms` フォルダは常に、JMS MQRFH2 に存在します。

表 541. <code>jms</code> のプロパティ名、同義語、データ・タイプ、およびフォルダー			
プロパティ 同義語	プロパティ 名	デー タ・タ イプ	フォルダー
JMSDestin ation	<code>jms.Dst</code>	strin g	<code><jms><Dst> destination </Dst></jms></code>
JMSExpira tion	<code>jms.Exp</code>	i8	<code><jms><Exp> expiration </Exp></jms></code>
JMSCorrel ation	<code>jms.Cid</code>	strin g	<code><jms><Cid> correlationId </Cid></jms></code>
JMSDelive ry	<code>jms.Dlv</code>	i4	<code><jms><Dlv> delivery </Dlv></jms></code>
JMSPriori ty	<code>jms.Pri</code>	i4	<code><jms><Pri> priority </Pri></jms></code>
JMSReplyT o	<code>jms.Rto</code>	strin g	<code><jms><Rto> replyToURI </Rto></jms></code>
JMSTimeSt amp	<code>jms.Tms</code>	i8	<code><jms><Tms> timestamp </Tms></jms></code>
JMSXGroup ID	<code>jms.Gid</code>	strin g	<code><jms><Gid> groupId </Gid></jms></code>
JMSXGroup Seq	<code>jms.Seq</code>	i4	<code><jms><Seq> messageSequenceNo </Seq></jms></code>

独自のプロパティを `jms` フォルダに追加しないでください。

mcd

`mcd` には、メッセージの形式を記述するプロパティが入ります。例えば、メッセージ・サービス・ドメインの `Msd` プロパティは、JMS メッセージを `JMSTextMessage`、`JMSBytesMessage`、`JMSStreamMessage`、`JMSMapMessage`、`JMSObjectMessage`、または `Null` として識別します。

`mcd` フォルダは常に、MQRFH2 が入っている JMS メッセージ内に存在します。

これは常に、IBM Integration Bus から送信された MQRFH2 が入っている含むメッセージ内に存在します。そして、メッセージのドメイン、形式、タイプ、およびメッセージ・セットを記述します。

表 542. mcd のプロパティ名、同義語、データ型、およびフォルダー			
プロパティ 同義語	プロパティ 名	デー タ・タ イプ	フォルダー
	mcd.Msd	string	<mcd><Msd>messageDomain</Msd></mcd>
	mcd.Set	string	<mcd><Set>messageDomain</Set></mcd>
	mcd.Type	string	<mcd><Type>messageDomain</Type></mcd>
	mcd.Fmt	string	<mcd><Fmt>messageDomain</Fmt></mcd>

独自のプロパティを mcd フォルダーに追加しないでください。

mq_usr

mq_usr には、JMS ユーザー定義プロパティとして公開されないアプリケーション定義プロパティが含まれます。JMS の要件を満たしていないプロパティを、このフォルダーに置くことができます。

mq_usr フォルダー内にプロパティを作成できます。mq_usr で作成するプロパティは、content='properties' 属性を使用して新規フォルダーで作成するプロパティと似ています。

sib

sib には、WebSphere Application Server サービス統合バス (WAS/SIB) システム・メッセージ・プロパティが含まれています。sib プロパティは、サポートされているタイプではないため、JMS プロパティとして IBM MQJMS アプリケーションに公開されません。例えば、一部の sib プロパティはバイト配列であるため、JMS プロパティとして公開できません。一部の sib プロパティは、JMS_IBM_*プロパティとして WAS/SIB アプリケーションに公開されます。これには、フォワード・ルーティング・パス・プロパティとリバース・ルーティング・パス・プロパティが含まれます。

独自のプロパティを sib フォルダーに追加しないでください。

sib_context

sib_context には、WAS/SIB ユーザー・アプリケーションに公開されていない、または JMS プロパティとして公開されていない WAS/SIB システム・メッセージ・プロパティが含まれています。

sib_context には、Web サービスに使用されるセキュリティー・プロパティとトランザクション・プロパティが含まれています。

独自のプロパティを sib_context フォルダーに追加しないでください。

sib_usr

sib_usr には、サポートされていないタイプであるために JMS ユーザー・プロパティとして公開されない WAS/SIB ユーザー・メッセージ・プロパティが含まれています。sib_usr は、SIBMessage インターフェースで WAS/SIB アプリケーションに公開されます。[サービス統合の開発](#)を参照してください。

sib_usr プロパティのタイプは bin.hex でなければならず、値は正しい形式でなければなりません。IBM MQ アプリケーションがフォルダーに bin.hex 型の要素を間違った形式で書き込むと、アプリケーションは IOException を受け取ります。プロパティのデータ・タイプが bin.hex でない場合、アプリケーションは ClassCastException を受け取ります。

このフォルダーを使用して、JMS ユーザー・プロパティを WAS/SIB で使用可能にしないでください。代わりに、usr フォルダーを使用してください。

sib_usr フォルダー内にプロパティを作成できます。

usr

usr には、メッセージに関連付けられているアプリケーション定義の JMS プロパティが入ります。usr フォルダーは、アプリケーションがアプリケーション定義プロパティを設定した場合のみ存在します。

usr は、デフォルトのプロパティ・フォルダーです。フォルダー名を指定せずにプロパティを設定すると、usr フォルダーに配置されます。

表 543. usr のプロパティ名、同義語、データ・タイプ、およびフォルダー			
プロパティ同義語	プロパティ名	データ・タイプ	フォルダー
	usr.contentType	string	<usr><contentType>text/xml; charset=utf-8</contentType></usr>
	usr.endpointURL	string	<usr><endpointURL> URI </endpointURL></usr>
	usr.targetService	string	<usr><targetService> serviceName </targetService></usr>
	usr.soapAction	string	<usr><soapAction> name </soapAction></usr>
	usr.transportVersion	string	<usr><transportVersion> version </transportVersion></usr>

usr フォルダー内にプロパティを作成できます。

MQMF_SEGMENT または MQMF_SEGMENTATION_ALLOWED で書き込まれたセグメント化メッセージには、定義されたプロパティ・フォルダー名を持つ MQRFH2 を含めることはできません。MQPUT は理由コード 2443、MQRC_SEGMENTATION_NOT_ALLOWED で失敗します。

グループ化されていないプロパティフォルダー名

ibm

ibm には、IBM MQ によってのみ使用されるプロパティが含まれます。

表 544. ibm のプロパティ名、同義語、データ型、およびフォルダー			
プロパティ同義語	プロパティ名	データ・タイプ	フォルダー
	ibm.rfp	string	<ibm><rfp>fingerprint</rfp></ibm>

独自のプロパティを ibm フォルダーに追加しないでください。

mq

mq には、IBM MQ によってのみ使用されるプロパティが含まれます。

mq フォルダーのプロパティには、次に挙げる制約事項が当てはまります。

- メッセージ内で有効な最初の mq フォルダーにあるプロパティのみが MQ の処理の対象となり、メッセージにある他の mq フォルダーのプロパティは無視されます。
- 1 バイトの UTF-8 文字だけがこのフォルダー内で許可されています。このフォルダーにマルチバイトの文字が一つでもあると、構文解析は失敗し、メッセージは拒否されます。
- このフォルダーでエスケープ・ストリングを使用しないでください。エスケープ・ストリングは、エレメントの実際の値として扱われてしまいます。
- フォルダー内では、Unicode 文字 U+0020 のみが空白文字として扱われます。その他の文字はすべて有効として扱われ、フォルダー解析の失敗やメッセージの拒否の原因となる場合があります。

mq フォルダーの構文解析が失敗した場合、またはフォルダーがこれらの制限に従わない場合、メッセージは理由コード 2527、MQRC_RFH_RESTRICTED_FORMAT_ERR で拒否されます。

独自のプロパティを mq フォルダーに追加しないでください。

mqema

mqema には、WebSphere Application Server によってのみ使用されるプロパティが含まれます。フォルダーは mqext に置き換えられました。

独自のプロパティを mqema フォルダーに追加しないでください。

mqext

mqext には、以下のタイプのプロパティが入ります。

- WebSphere Application Server 専用のプロパティ。
- メッセージの遅延送達に関連するプロパティ。

このフォルダーが存在するのは、アプリケーションで IBM 定義のプロパティを 1 つ以上設定したか、遅延送達を使用した場合です。

表 545. mqext のプロパティ名、同義語、データ型、およびフォルダー

プロパティ同義語	プロパティ名	データ・タイプ	フォルダー
JMSArmCorrelator	mqext.Arm	string	<mqext><Arm>armCorrelator</Arm></mqext>
JMSRMCorrelator	mqext.Wrm	string	<mqext><Wrm>wrmCorrelator</Wrm></mqext>
JMSDeliveryTime	mqext.Dlt	i8	<mqext><Dlt>DeliveryTime</Dlt></mqext>
JMSDeliveryDelay	mqext.Dly	i8	<mqext><Dly>DeliveryTime</Dly></mqext>

独自のプロパティを mqext フォルダーに追加しないでください。

mmps

mmps には、IBM MQ パブリッシュ/サブスクライブによってのみ使用されるプロパティが入ります。このフォルダーは、統合されたパブリッシュ/サブスクライブ・プロパティを少なくとも 1 つアプリケーションが設定した場合にのみ存在します。

表 546. mmps のプロパティ名、同義語、データ型、およびフォルダー

プロパティ同義語	プロパティ名	データ・タイプ	フォルダー
MQTopicString	mmps.Top	string	<mmps><Top>topicString</Top></mmps>

表 546. *mqs* のプロパティ名、同義語、データ型、およびフォルダー (続き)

プロパティ 同義語	プロパティ 名	デー タ・タイ プ	フォルダー
MQSubUserData	mqs.Sud	string	<mqs><Sud> <i>subscriberUserData</i> ...</Sud></mqs>
MQIsRetained	mqs.Ret	boolean	<mqs><Ret> <i>isRetained</i> </Ret></mqs>
MQPubOptions	mqs.Pub	i8	<mqs><Pub> <i>publicationOptions</i> </Pub></mqs>
MQPubLevel	mqs.Pbl	i8	<mqs><Pbl> <i>publicationLevel</i> </Pbl></mqs>
MQPubTime	mqs.Pts	string	<mqs><Pts> <i>publicationTime</i> </Pts></mqs>
MQPubSeqNum	mqs.Seq	i8	<mqs><Seq> <i>publicationSequenceNumber</i> </Seq></mqs>
MQPubStrInData	mqs.Sid	string	<mqs><Sid> <i>publicationData</i> </Sid></mqs>
MQPubFormat	mqs.Pfmt	i8	<mqs><Pfmt> <i>messageFormat</i> </Pfmt></mqs>

独自のプロパティを *mqs* フォルダーに追加しないでください。

mq_svc

mq_svc には、SupportPac MA93 によって使用されるプロパティが含まれています。

独自のプロパティを *mq_svc* フォルダーに追加しないでください。

mqtt

mqtt には、IBM MQ Telemetry によって使用されるプロパティが含まれます。

表 547. *mqtt* のプロパティ名、同義語、データ・タイプ、およびフォルダー

プロパティ 同義語	プロパティ 名	デー タ・タイ プ	フォルダー
	mqtt.clientId	string	<mqtt><clientId> <i>topicString</i> </clientId></mqtt>
	mqtt.qos	i4	<mqtt><qos> <i>qualityOfService</i> </qos></mqtt>
	mqtt.msgid	string	<mqtt><msgid> <i>messageIdentifier</i> </msgid></mqtt>

独自のプロパティを *mqtt* フォルダーに追加しないでください。

MQMF_SEGMENT または MQMF_SEGMENTATION_ALLOWED のいずれかを使用して書き込まれたセグメント化されたメッセージに、グループ化されていないプロパティ・フォルダー名を持つ MQRFH2 を含めることはできません。MQPUT は理由コード 2443、MQR_SEGMENTATION_NOT_ALLOWED で失敗します。

名前-値ペア

構文図では、"Name-value pairs" は通常のフォルダーの内容を説明します。通常のフォルダーには、グループと要素が含まれます。要素は名前と値のペアです。グループには、要素と、他のグループが含まれます。

ツリーの用語で言うと、要素はリーフ・ノードであり、グループは内部ノードです。内部ノードと、ルート・ノードであるフォルダーは、内部ノードとリーフ・ノードを混在させて含むことができます。1つのノードが内部ノードであると同時にリーフ・ノードであることはできません。[514 ページの図 2](#)を参照してください。

プロパティ

構文図では、"Properties" はプロパティ・フォルダーの内容を説明しています。プロパティ・フォルダーには、グループとプロパティが含まれます。プロパティは、名前と値のペアで、オプションとしてデータ・タイプ属性を伴います。グループには、プロパティと他のグループが含まれます。

ツリーの用語で言うと、プロパティはリーフ・ノードであり、グループは内部ノードです。内部ノードと、ルート・ノードであるプロパティ・フォルダーは、内部ノードとリーフ・ノードを混在させて含むことができます。1つのノードが内部ノードであると同時にリーフ・ノードであることはできません。[514 ページの図 2](#)を参照してください。

不動産

メッセージ・プロパティは、プロパティ・フォルダーにある名前と値のペアです。データ型属性とプロパティ属性を任意で含めることができます。例については、以下のコードを参照してください。データ・タイプ属性を省略した場合、プロパティの型は **string** です。

```
<pf><p1 dt='i8' > value </p1></pf>
```

メッセージ・プロパティの名前は、XML のような<>構文をドットに置き換えた絶対パス名になります。例えば、myPropertyFolder1.myGroup1.myGroup2.myProperty1 は次のように文字 **NameValueData** 列にマッピングされる。ストリングは読みやすいようにフォーマットしてあります。

```
<myPropertyFolder1>
  <myGroup1>
    <myGroup2>
      <myProperty1>value</myProperty1>
    </myGroup2>
  </myGroup1>
</myPropertyFolder1>
```

プロパティ・フォルダーには、複数のプロパティを含めることができます。例えば、プロパティは **Changes** のプロパティフォルダにマッピングされ [521 ページの図 7](#)、新しいクラスタでセールスキューを分離し、ゲートウェイクラスタの伝送キューを分離します。

```
myPropertyFolder1.myProperty4
myPropertyFolder1.myGroup1.myGroup2.myProperty1
myPropertyFolder1.myGroup1.myGroup2.myProperty2
myPropertyFolder1.myGroup1.myProperty3
```

図 7. 同じルート名を持つ複数のプロパティ

```
<myPropertyFolder1>
  <myProperty4>value</myProperty4>
  <myGroup1>
    <myGroup2>
      <myProperty1>value</myProperty1>
      <myProperty2>value</myProperty2>
    </myGroup2>
    <myProperty3>value</myProperty3>
  </myGroup1>
</myPropertyFolder1>
```

図 8. 複数プロパティ名のマッピング

名前

名前の先頭は文字 または下線 でなければなりません。コロンを含んではならず、末尾がピリオドであってもなりません。文字、数字、下線、ハイフン、ドットを含むことができます。有効な文字は W3C XML 仕様で定義されており、基本的に Unicode カテゴリー L1, Lu, Lo, Lt, Nl, Mc, Mn, Lm, および Nd で構成されます。[Unicode 文字のカテゴリ](#)を参照してください。

プロパティまたは名前と値のペアの完全なパスは、無効なパス名で説明されている規則に違反してはならない。パスは 4095 バイトに制限されており、Unicode 互換文字を含んではなりません。また、ストリング XML で始まってはなりません。

グループ名

グループ名は、名前と同じ構文規則を持ちます。グループ名はオプションです。プロパティおよび名前と値のペアは、フォルダーのルートに置くことができます。グループは、プロパティや名前と値のペアを編成するのに役立つ場合に使用します。

要素名

要素名は、名前と同じ構文規則を持ちます。

要素値

エレメント値には、< *Element name* >タグと< /*Element name* >タグの間にあるすべての空白文字が含まれます。値には、<と&の 2 文字を使用しないでください。<および&に置き換えます。

プロパティ属性

プロパティ属性は、プロパティ記述子フィールドをマップします。マッピングは次のとおりです。

サポート

sa (デフォルト)

MQPD_SUPPORT_OPTIONAL

sr

MQPD_SUPPORT_REQUIRED

sx

MQPD_SUPPORT_REQUIRED_IF_LOCAL

Context

なし (デフォルト)

MQPD_NO_CONTEXT

ユーザー

MQPD_USER_CONTEXT

CopyOptions

forward

MQPD_COPY_FORWARD

reply

MQPD_COPY_REPLY

レポート

MQPD_COPY_REPORT

publish

MQPD_COPY_PUBLISH

すべて

MQPD_COPY_ALL

all を他のオプションと組み合わせて使用することはしないでください。

default

MQPD_COPY_DEFAULT

default を他のオプションと組み合わせて使用することはしないでください。default は、forward + report + publish と同じです。

なし

MQPD_COPY_NONE

none を他のオプションと組み合わせて使用することはしないでください。

Support プロパティ属性は、mq フォルダーのプロパティにのみ適用されます。

Context および CopyOptions プロパティ属性は、すべてのプロパティ・フォルダーに適用されます。

データ型

MQRFH2 データ・タイプは、以下のようにメッセージ・プロパティ・タイプにマップされます。

表 548. データ・タイプ・マッピング	
MQRFH2 データ・タイプ	メッセージ・プロパティ・タイプ
bin.hex	MQBYTE[]
boolean	MQBOOL
i1	MQINT8
i2	MQINT16
i4	MQINT32
i8	MQINT64
r4	MQFLOAT32
r8	MQFLOAT64
string	MQCHAR[]

データ・タイプのない要素の型は string であると想定されます。

ヌル値は、エレメント属性 `xsi:nil='true'` によって示されます。NULL 以外の値に属性 `xsi:nil='false'` を使用しないでください。例えば、次のプロパティはヌル値を持っています。

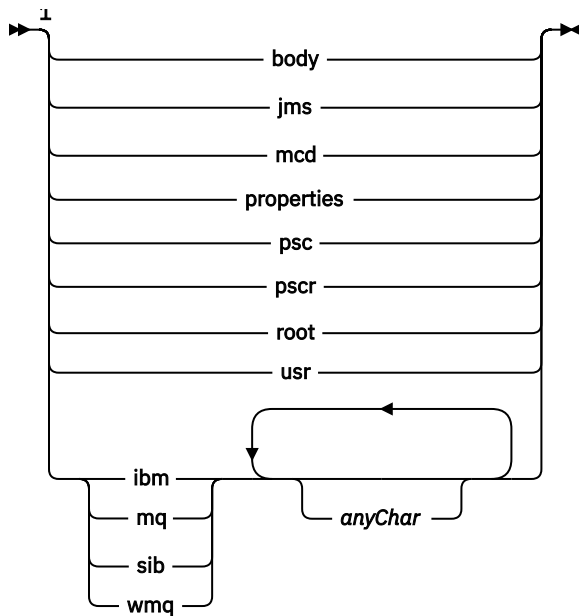
```
<NullProperty  
xsi:nil='true'></NullProperty>
```

バイトや文字ストリングのプロパティに空の値があってもかまいません。空の値は、長さゼロのエレメント値を持つ MQRFH2 エレメントによって表されます。例えば、次のプロパティは空の値を持っています。

```
<EmptyProperty></EmptyProperty>
```

予約済みフォルダまたはプロパティフォルダ名

フォルダーまたはプロパティ・フォルダーの名前が、次のいずれかのストリングで始まらないように制限してください。これらのプレフィックスは、IBM が作成するフォルダーまたはプロパティ名のために予約済みです。

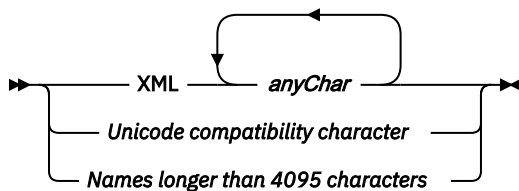


注:

¹ 予約済みのフォルダーまたはプロパティ名には、大文字/小文字のすべての組み合わせが含まれません。

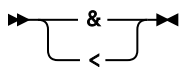
無効なパス名

名前と値のペアまたはプロパティの完全パスが、次のストリングのいずれも含まないように制限してください。



Invalid characters

リテラル"&"および"<"の代わりに、常にエスケープ・シーケンス&および<を使用してください。

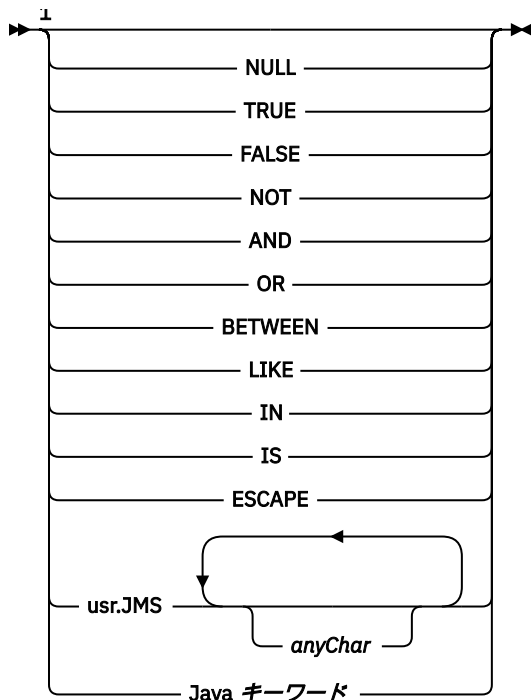


定義されたプロパティ名

定義されたプロパティ名とは、IBM MQ またはその他の製品によって定義されたプロパティの名前のことで、IBM MQ およびユーザー・アプリケーションによって使用されます。定義されたプロパティは、定義されたプロパティ・フォルダー の中にのみ存在します。定義されたプロパティ名は、プロパティ・フォルダーの記述の中で記述されます。[プロパティ・フォルダー](#)を参照してください。

無効なプロパティ名

次の規則に当てはまるプロパティ名は作成しないでください。この規則は、プロパティ要素名だけでなく、プロパティ名を指定するフル・プロパティ・パスに適用されます。



注:

¹ 無効なプロパティ名には、大文字と小文字のすべての組み合わせが含まれます。

無効な属性

プロパティフォルダーおよびプロパティには、サポートされている プロパティ属性 と データ型 のみを含めることができます。

サポートされない XML 類似の属性 (例えば引用符付きストリング値を含む名前) がプロパティ・フォルダーやプロパティに含まれていると、除去される可能性があります。

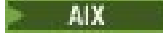
非プロパティ・フォルダーや非プロパティ要素に含まれる XML 類似の属性は、MQRFH2 ヘッダーに残されます。

MQRMH - 参照メッセージ・ヘッダー

MQRMH 構造体は参照メッセージ・ヘッダーの形式を定義します。このヘッダーは、1つのキュー・マネージャーから別のキュー・マネージャーに、非常に大量のデータ (バルク・データと呼ばれる) を送信するために、ユーザー作成のメッセージ・チャンネル出口と一緒に使用します。通常のメッセージングとは異なり、バルク・データはキューに格納されません。キューに格納されるのはバルク・データへの参照のみです。これにより、ごく少数の非常に大きなメッセージによってリソースが枯渇する可能性 IBM MQ が低減されます。

可用性

MQRMH 構造は、以下のプラットフォームで利用可能です：

-  AIX
-  IBM i
-  Linux
-  Windows

および、これらのシステムに接続された IBM MQ クライアント。

形式名

MQFMT_REF_MSG_HEADER

文字セットとエンコード

MQRMH 内の文字データおよびオフセットフィールドが参照する文字列は、ローカルキューマネージャの文字セットでなければならない。これはキューマネージャ **CodedCharSetId** 属性によって指定される。MQRMH の数値データは、ネイティブ・マシン・エンコードでなければなりません。これは、C プログラミング言語用の MQENC_NATIVE の値で指定します。

MQRMH の文字セットおよびエンコードは、以下の構造体の *CodedCharSetId* および *Encoding* フィールドに設定する必要があります。

- MQMD (MQRMH 構造体がメッセージ・データの開始点にある場合)
- MQRMH 構造体に先行するヘッダー構造体 (その他のすべての場合)

使用法

アプリケーションは、MQRMH で構成されるメッセージを送信するが、バルクデータは省略する。メッセージ・チャンネル・エージェント (MCA) が伝送キューのメッセージを読み込むと、ユーザー定義のメッセージ出口が呼び出されて参照メッセージ・ヘッダーを処理します。この出口は、参照メッセージに MQRMH 構造体によって識別されたバルク・データを追加できます。その後、MCA はそのチャンネルを通じて次のキュー・マネージャーにメッセージを送ります。

受信側には、参照メッセージ待機中のメッセージ出口が必要です。参照メッセージを受け取ると、出口はメッセージ中の MQRMH に続くバルク・データからオブジェクトを作成します。次にバルク・データを除いて参照メッセージを渡します。参照メッセージは、(バルク・データなしで) キューから参照メッセージを読み取るアプリケーションによってあとから取り出すことができます。

通常 MQRMH 構造体は、メッセージ内にあるものがすべてです。ただし、メッセージが伝送キューにある場合は、1 つ以上の追加ヘッダーが MQRMH 構造体の前に付きます。

参照メッセージは、配布リストに送ることもできます。この場合、メッセージが伝送キューにあると、MQDH 構造体およびその関連レコードは MQRMH 構造体の前に付きます。

注: 参照メッセージはセグメント化メッセージとして送信しないでください。メッセージ出口が正常に処理できないためです。

データ変換

データ変換の目的で、MQRMH 構造の変換には、ソース環境データ、ソースオブジェクト名、宛先環境データ、および宛先オブジェクト名の変換が含まれます。構造体の先頭の *StrucLength* の他のバイトは、データ変換終了後に廃棄されるか、未定義の値が入ります。バルク・データは、以下の記述がすべて該当する場合に変換されます。

- バルク・データがデータ変換実行時にメッセージの中にある。
- MQRMH 内の *Format* フィールドの値が MQFMT_NONE 以外である。
- ユーザーが作成したデータ変換出口が指定された形式名で存在する。

ただし、メッセージがキューにあるときには通常、メッセージ内にバルク・データは存在せず、その結果、MQGMO_CONVERT オプションでバルク・データが変換される点に注意してください。

フィールド

注: 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 549. MQRMH における MQRMH のフィールド

フィールド名と説明	定数の名前	定数の初期値（存在する場合）
StrucId (構造識別子)	MQRMH_STRUC_ID	'RMH↵'
バージョン (構造バージョン番号)	MQRMH_VERSION_1	1
StrucLength (MQRMH の全長。固定フィールド末尾の文字列を含むが、本体データは含まない)	なし	0
エンコーディング (バルクデータの数字による符号化)	MQENC_NATIVE	環境に依存
CodedCharSetId (バルクデータの文字セット識別子)	MQCCSI_UNDEFINED	0
フォーマット (一括データのフォーマット名)	MQFMT_NONE	ブランク
フラグ (参照メッセージフラグ)	MQRMHF_NOT_LAST	0
ObjectType (オブジェクトタイプ)	なし	ブランク
ObjectInstanceId (オブジェクトインスタンス識別子)	MQOII_NONE	Null
SrcEnvLength (ソース環境データの長さ)	なし	0
SrcEnvOffset (ソース環境データのオフセット)	なし	0
SrcNameLength (ソースオブジェクト名の長さ)	なし	0
SrcNameOffset (ソースオブジェクト名のオフセット)	なし	0
DestEnvLength (宛先環境データの長さ)	なし	0
DestEnvOffset (宛先環境データのオフセット)	なし	0
DestNameLength (目的オブジェクト名の長さ)	なし	0
DestNameOffset (宛先オブジェクト名のオフセット)	なし	0
DataLogicalLength (バルクデータの長さ)	なし	0
DataLogicalOffset (バルクデータのオフセットが低い)	なし	0
DataLogicalOffset2 (バルクデータの高オフセット)	なし	0
注： 1. 記号↵は、単一のブランク文字を表します。 2. C プログラミング言語では、マクロ変数 MQRMH_DEFAULT には、表にリストされている値が含まれています。このマクロ変数を以下の方法で使用して、構造体のフィールドに初期値を設定します。 <pre>MQRMH MyRMH = {MQRMH_DEFAULT};</pre>		

言語ごとの宣言

MQRMH の C 宣言

```
typedef struct tagMQRMH MQRMH;
struct tagMQRMH {
```

```

MQCHAR4  StrucId;           /* Structure identifier */
MQLONG   Version;          /* Structure version number */
MQLONG   StrucLength;      /* Total length of MQRMH, including
                           strings at end of fixed fields, but
                           not the bulk data */

MQLONG   Encoding;        /* Numeric encoding of bulk data */
MQLONG   CodedCharSetId;  /* Character set identifier of bulk
                           data */

MQCHAR8   Format;         /* Format name of bulk data */
MQLONG   Flags;           /* Reference message flags */
MQCHAR8   ObjectType;     /* Object type */
MQBYTE24  ObjectInstanceId; /* Object instance identifier */
MQLONG   SrcEnvLength;    /* Length of source environment data */
MQLONG   SrcEnvOffset;    /* Offset of source environment data */
MQLONG   SrcNameLength;   /* Length of source object name */
MQLONG   SrcNameOffset;   /* Offset of source object name */
MQLONG   DestEnvLength;   /* Length of destination environment
                           data */
MQLONG   DestEnvOffset;   /* Offset of destination environment
                           data */

MQLONG   DestNameLength;  /* Length of destination object name */
MQLONG   DestNameOffset;  /* Offset of destination object name */
MQLONG   DataLogicalLength; /* Length of bulk data */
MQLONG   DataLogicalOffset; /* Low offset of bulk data */
MQLONG   DataLogicalOffset2; /* High offset of bulk data */
};

```

MQRMH の COBOL 宣言

```

**  MQRMH structure
10 MQRMH.
**  Structure identifier
15 MQRMH-STRUCID          PIC X(4).
**  Structure version number
15 MQRMH-VERSION         PIC S9(9) BINARY.
**  Total length of MQRMH, including strings at end of fixed fields,
**  but not the bulk data
15 MQRMH-STRUCLength     PIC S9(9) BINARY.
**  Numeric encoding of bulk data
15 MQRMH-ENCODING        PIC S9(9) BINARY.
**  Character set identifier of bulk data
15 MQRMH-CODEDCHARSETID  PIC S9(9) BINARY.
**  Format name of bulk data
15 MQRMH-FORMAT          PIC X(8).
**  Reference message flags
15 MQRMH-FLAGS           PIC S9(9) BINARY.
**  Object type
15 MQRMH-OBJECTTYPE      PIC X(8).
**  Object instance identifier
15 MQRMH-OBJECTINSTANCEID PIC X(24).
**  Length of source environment data
15 MQRMH-SRCENVLENGTH    PIC S9(9) BINARY.
**  Offset of source environment data
15 MQRMH-SRCENVOFFSET    PIC S9(9) BINARY.
**  Length of source object name
15 MQRMH-SRCNAMELENGTH   PIC S9(9) BINARY.
**  Offset of source object name
15 MQRMH-SRCNAMEOFFSET   PIC S9(9) BINARY.
**  Length of destination environment data
15 MQRMH-DESTENVLENGTH   PIC S9(9) BINARY.
**  Offset of destination environment data
15 MQRMH-DESTENVOFFSET   PIC S9(9) BINARY.
**  Length of destination object name
15 MQRMH-DESTNAMELENGTH  PIC S9(9) BINARY.
**  Offset of destination object name
15 MQRMH-DESTNAMEOFFSET  PIC S9(9) BINARY.
**  Length of bulk data
15 MQRMH-DATALOGICALLength PIC S9(9) BINARY.
**  Low offset of bulk data
15 MQRMH-DATALOGICALOFFSET PIC S9(9) BINARY.
**  High offset of bulk data
15 MQRMH-DATALOGICALOFFSET2 PIC S9(9) BINARY.

```

MQRMH の PL/I 宣言

```

dcl
  1 MQRMH based,
    3 StrucId          char(4),          /* Structure identifier */
    3 Version          fixed bin(31),    /* Structure version number */

```

```

3 StrucLength      fixed bin(31), /* Total length of MQRMH,
                        including strings at end of
                        fixed fields, but not the bulk
                        data */
3 Encoding         fixed bin(31), /* Numeric encoding of bulk
                        data */
3 CodedCharSetId   fixed bin(31), /* Character set identifier of
                        bulk data */
3 Format           char(8),        /* Format name of bulk data */
3 Flags           fixed bin(31), /* Reference message flags */
3 ObjectType       char(8),        /* Object type */
3 ObjectInstanceId char(24),       /* Object instance identifier */
3 SrcEnvLength     fixed bin(31), /* Length of source environment
                        data */
3 SrcEnvOffset     fixed bin(31), /* Offset of source environment
                        data */
3 SrcNameLength    fixed bin(31), /* Length of source object name */
3 SrcNameOffset    fixed bin(31), /* Offset of source object name */
3 DestEnvLength    fixed bin(31), /* Length of destination
                        environment data */
3 DestEnvOffset    fixed bin(31), /* Offset of destination
                        environment data */
3 DestNameLength   fixed bin(31), /* Length of destination object
                        name */
3 DestNameOffset   fixed bin(31), /* Offset of destination object
                        name */
3 DataLogicalLength fixed bin(31), /* Length of bulk data */
3 DataLogicalOffset fixed bin(31), /* Low offset of bulk data */
3 DataLogicalOffset2 fixed bin(31); /* High offset of bulk data */

```

MQRMH の高水準アセンブラ宣言

```

MQRMH              DSECT
MQRMH_STRUCID      DS    CL4    Structure identifier
MQRMH_VERSION      DS    F      Structure version number
MQRMH_STRUCLNGTH   DS    F      Total length of MQRMH, including
*                  strings at end of fixed fields, but
*                  not the bulk data
MQRMH_ENCODING     DS    F      Numeric encoding of bulk data
MQRMH_CODEDCHARSETID DS    F    Character set identifier of bulk
*                  data
MQRMH_FORMAT       DS    CL8    Format name of bulk data
MQRMH_FLAGS        DS    F      Reference message flags
MQRMH_OBJECTTYPE   DS    CL8    Object type
MQRMH_OBJECTINSTANCEID DS    XL24 Object instance identifier
MQRMH_SRCENVLENGTH DS    F      Length of source environment data
MQRMH_SRCENVOFFSET DS    F      Offset of source environment data
MQRMH_SRCNAMELENGTH DS    F      Length of source object name
MQRMH_SRCNAMEOFFSET DS    F      Offset of source object name
MQRMH_DESTENVLENGTH DS    F      Length of destination environment
*                  data
MQRMH_DESTENVOFFSET DS    F      Offset of destination environment
*                  data
MQRMH_DESTNAMELENGTH DS    F      Length of destination object name
MQRMH_DESTNAMEOFFSET DS    F      Offset of destination object name
MQRMH_DATALOGICALLLENGTH DS    F    Length of bulk data
MQRMH_DATALOGICALOFFSET DS    F    Low offset of bulk data
MQRMH_DATALOGICALOFFSET2 DS    F    High offset of bulk data
*
MQRMH_LENGTH       EQU    *-MQRMH
                    ORG    MQRMH
MQRMH_AREA         DS    CL(MQRMH_LENGTH)

```

MQRMH の Visual Basic 宣言

```

Type MQRMH
    StrucId      As String*4 'Structure identifier'
    Version      As Long     'Structure version number'
    StrucLength  As Long     'Total length of MQRMH, including'
                        'strings at end of fixed fields, but'
                        'not the bulk data'
    Encoding     As Long     'Numeric encoding of bulk data'
    CodedCharSetId As Long   'Character set identifier of bulk data'
    Format       As String*8 'Format name of bulk data'
    Flags        As Long     'Reference message flags'
    ObjectType    As String*8 'Object type'
    ObjectInstanceId As MQBYTE24 'Object instance identifier'
    SrcEnvLength  As Long     'Length of source environment data'
    SrcEnvOffset  As Long     'Offset of source environment data'

```

```

SrcNameLength      As Long      'Length of source object name'
SrcNameOffset      As Long      'Offset of source object name'
DestEnvLength      As Long      'Length of destination environment'
                        'data'
DestEnvOffset      As Long      'Offset of destination environment'
                        'data'
DestNameLength     As Long      'Length of destination object name'
DestNameOffset     As Long      'Offset of destination object name'
DataLogicalLength  As Long      'Length of bulk data'
DataLogicalOffset  As Long      'Low offset of bulk data'
DataLogicalOffset2 As Long      'High offset of bulk data'
End Type

```

MQRMH のフィールド詳細

MQRMH 参照メッセージヘッダー構造は、参照メッセージヘッダーの形式を定義します。このヘッダーは、1つのキュー・マネージャーから別のキュー・マネージャーに、非常に大量のデータ (バルク・データと呼ばれる) を送信するために、ユーザー作成のメッセージ・チャンネル出口と一緒に使用します。通常のメッセージングとは異なり、バルク・データはキューに格納されません。キューに格納されるのはバルク・データへの参照のみです。これにより、ごく少数の非常に大きなメッセージによってリソースが枯渇する可能性 IBM MQ が低減されます。

フィールド

注: 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 550. MQRMH における MQRMH のフィールド		
フィールド名と説明	定数の名前	定数の初期値 (存在する場合)
StrucId (構造識別子)	MQRMH_STRUC_ID	'RMH'
バージョン (構造バージョン番号)	MQRMH_VERSION_1	1
StrucLength (MQRMH の全長。固定フィールド末尾の文字列を含むが、本体データは含まない)	なし	0
エンコーディング (バルクデータの数字による符号化)	MQENC_NATIVE	環境に依存
CodedCharSetId (バルクデータの文字セット識別子)	MQCCSI_UNDEFINED	0
フォーマット (一括データのフォーマット名)	MQFMT_NONE	ブランク
フラグ (参照メッセージフラグ)	MQRMHF_NOT_LAST	0
ObjectType (オブジェクトタイプ)	なし	ブランク
ObjectInstanceId (オブジェクトインスタンス識別子)	MQOII_NONE	Null
SrcEnvLength (ソース環境データの長さ)	なし	0
SrcEnvOffset (ソース環境データのオフセット)	なし	0
SrcNameLength (ソースオブジェクト名の長さ)	なし	0
SrcNameOffset (ソースオブジェクト名のオフセット)	なし	0
DestEnvLength (宛先環境データの長さ)	なし	0
DestEnvOffset (宛先環境データのオフセット)	なし	0
DestNameLength (目的オブジェクト名の長さ)	なし	0

表 550. MQRMH における MQRMH のフィールド (続き)

フィールド名と説明	定数の名前	定数の初期値 (存在する場合)
<u>DestNameOffset</u> (宛先オブジェクト名のオフセット)	なし	0
<u>DataLogicalLength</u> (バルクデータの長さ)	なし	0
<u>DataLogicalOffset</u> (バルクデータのオフセットが低い)	なし	0
<u>DataLogicalOffset2</u> (バルクデータの高オフセット)	なし	0

注:

1. 記号-は、単一のブランク文字を表します。
2. C プログラミング言語では、マクロ変数 MQRMH_DEFAULT には、表にリストされている値が含まれています。このマクロ変数を以下の方法で使用して、構造体のフィールドに初期値を設定します。

```
MQRMH MyRMH = {MQRMH_DEFAULT};
```

StrucId (MQCHAR4)

これは参照メッセージヘッダー構造の構造識別子です。常に入力フィールドです。その値は MQRMH_STRUC_ID です。

値は次のものでなければなりません。

MQRMH_STRUC_ID

参照メッセージヘッダー構造体の識別子。

C プログラミング言語では、定数 MQRMH_STRUC_ID_ARRAY も定義されている。これは MQRMH_STRUC_ID と同じ値を持ちますが、文字列ではなく文字の配列です。

Version (MQLONG)

構造体のバージョン番号。値は次のものでなければなりません。

MQRMH_VERSION_1

バージョン 1 の参照メッセージ・ヘッダー構造体の ID。

以下の定数は、現行バージョンのバージョン番号を指定しています。

MQRMH_CURRENT_VERSION

参照メッセージ・ヘッダー構造体の現行バージョン。

フィールドの初期値は、MQRMH_VERSION_1 です。

StrucLength (MQLONG)

MQRMH の全長。固定フィールドの端のストリングを含むが、バルク・データは含まない。

フィールドの初期値は、0 です。

Encoding (MQLONG)

これは、バルク・データの数値エンコードを指定します。MQRMH 構造体自体の数値データには適用されません。

MQPUT または MQPUT1 呼び出しでは、アプリケーションは、このフィールドをデータに適切な値に設定する必要があります。

このフィールドの初期値は MQENC_NATIVE です。

CodedCharSetId (MQLONG)

これは、バルク・データの文字セット ID を指定します。MQRMH 構造体自体の文字データには適用されません。

MQPUT または MQPUT1 呼び出しでは、アプリケーションは、このフィールドをデータに適切な値に設定する必要があります。以下のような特別な値を使用することができます。

MQCCSI_INHERIT

この構造体の後続くデータの文字データは、この構造体に設定されているのと同じ文字セットになります。

キュー・マネージャーは、メッセージで送信される構造体の中のこの値を、構造体の実際の文字セット ID に変更します。エラーが発生しない限り、値 MQCCSI_INHERIT が MQGET 呼び出しによって返されることはありません。

MQMD の PutApplType フィールドの値が MQAT_BROKER である場合は、MQCCSI_INHERIT を使用しないでください。

この値は、次の環境でサポートされます。

-  AIX
-  IBM i
-  Linux
-  Windows

および、これらのシステムに接続された IBM MQ クライアント。

このフィールドの初期値は MQCCSI_UNDEFINED です。

Format (MQCHAR8)

これは、バルク・データの形式名を指定します。

MQPUT または MQPUT1 呼び出しでは、アプリケーションは、このフィールドをデータに適切な値に設定する必要があります。このフィールドのコーディングの規則は、MQMD の *Format* フィールドの場合と同じです。

このフィールドの初期値は MQFMT_NONE です。

フラグ (MQLONG)

これらは参照メッセージ・フラグです。以下のフラグが定義されます。

MQRMHF_LAST

このフラグを指定すると、参照メッセージには参照先オブジェクトの最後の部分が表示されます。

MQRMHF_NOT_LAST

参照メッセージにはオブジェクトの最後の部分は表示されない。MQRMHF_NOT_LAST は、プログラムを文書化する上で役に立ちます。このオプションを他のオプションと組み合わせて使用することは意図されていませんが、値がゼロであるため、そのような使い方をしても検出できません。

フィールドの初期値は、MQRMHF_NOT_LAST です。

ObjectType (MQCHAR8)

これは、メッセージ出口がサポートする参照メッセージの種類を認識するために使用する名前です。名前は *Format* フィールドと同じ規則に従う必要があります。詳細は「[フォーマット](#)」を参照してください。

このフィールドの初期値は 8 ブランクです。

ObjectInstanceId (MQBYTE24)

このフィールドはオブジェクトに固有のインスタンスを識別するのに使用します。このフィールドが必要でない場合は、以下の値に設定してください。

MQOII_NONE

指定されたオブジェクトのインスタンス ID がありません。値は、フィールドの長さについては 2 進ゼロです。

C プログラミング言語では、定数 MQOII_NONE_ARRAY も定義されます。これは、MQOII_NONE の値と同じです。ストリングではなく文字の配列です。

このフィールドの長さは MQ_OBJECT_INSTANCE_ID_LENGTH によって指定されます。フィールドの初期値は、MQOII_NONE です。

SrcEnvLength (MQLONG)

ソース環境データの長さ。このフィールドがゼロの場合、ソース環境データは存在せず、*SrcEnvOffset* は無視されます。

このフィールドの初期値は 0 です。

SrcEnvOffset (MQLONG)

このフィールドは MQRMH 構造体の先頭からの発信元環境データのオフセットを指定します。発信元環境データは、参照メッセージの作成者が既知のデータであれば、その作成者が指定できます。例えば Windows では、発信元環境データはバルク・データを含むオブジェクトのディレクトリー・パスにすることが可能です。ただし、作成者が発信元環境データを認識していない場合、ユーザーのメッセージ出口は必要な環境情報を調べる必要があります。

発信元環境データの長さは *SrcEnvLength* に指定します。この長さがゼロの場合、発信元環境データはなく、*SrcEnvOffset* は無視されます。発信元環境データがある場合は、構造体の先頭から *StrucLength* バイト以内に完全なデータとして存在する必要があります。

アプリケーションでは、環境データが構造体中の最後の固定フィールドの直後から始まることを前提にしてはなりません。また、環境データが *SrcNameOffset* フィールド、*DestEnvOffset* フィールド、および *DestNameOffset* フィールドによって指定されたデータと連続していることも前提にしてはなりません。

このフィールドの初期値は 0 です。

SrcNameLength (MQLONG)

ソース・オブジェクト名の長さ。このフィールドがゼロの場合、ソース・オブジェクト名はなく、*SrcNameOffset* は無視されます。

このフィールドの初期値は 0 です。

SrcNameOffset (MQLONG)

このフィールドは MQRMH 構造体の先頭からの送信側オブジェクト名のオフセットを指定します。送信側オブジェクト名は、参照メッセージの作成者が既知のデータであれば、その作成者が指定できます。ただし、作成者が発信元オブジェクト名を認識していない場合、ユーザーのメッセージ出口は、アクセスされるオブジェクトを識別する必要があります。

ソース・オブジェクト名の長さは *SrcNameLength* によって指定されます。この長さがゼロの場合、ソース・オブジェクト名はなく、*SrcNameOffset* は無視されます。ソース・オブジェクト名が存在する場合は、構造体の先頭から *StrucLength* バイト以内に完全に存在する必要があります。

アプリケーションは、ソース・オブジェクト名が *SrcEnvOffset*、*DestEnvOffset*、および *DestNameOffset* フィールドによってアドレス指定されたデータのいずれかに隣接していると想定してはなりません。

このフィールドの初期値は 0 です。

DestEnvLength (MQLONG)

これは宛先環境データの長さです。このフィールドがゼロの場合、宛先環境データはなく、*DestEnvOffset* は無視されます。

DestEnvOffset (MQLONG)

このフィールドは MQRMH 構造体の先頭からの宛先環境データのオフセットを指定します。宛先環境データは、参照メッセージの作成者が認識していれば、その作成者が指定できます。例えば Windows では、宛先環境データはバルク・データが格納されるオブジェクトのディレクトリー・パスにできます。ただし、作成者が宛先環境データを認識していない場合は、必要な環境情報を調べるのはユーザーのメッセージ出口の役目です。

宛先環境データの長さは *DestEnvLength* に指定します。この長さがゼロの場合、宛先環境データはなく、*DestEnvOffset* は無視されます。宛先環境データがある場合は、構造体の先頭から *StrucLength* バイト以内に完全なデータとして存在する必要があります。

アプリケーションでは、宛先環境データが *SrcEnvOffset* フィールド、*SrcNameOffset* フィールド、および *DestNameOffset* フィールドによって指定されたデータと連続することを前提にしてはなりません。

このフィールドの初期値は 0 です。

DestNameLength (MQLONG)

宛先オブジェクト名の長さ。このフィールドがゼロの場合、宛先オブジェクト名はなく、*DestNameOffset* は無視されます。

DestNameOffset (MQLONG)

このフィールドは MQRMH 構造体の先頭からの宛先オブジェクト名のオフセットを指定します。宛先オブジェクト名は、参照メッセージの作成者が既知のデータであれば、その作成者が指定できます。ただし、作成者が宛先オブジェクト名を認識していない場合は、作成または変更されるオブジェクトを識別するのはユーザーのメッセージ出口の役目です。

宛先オブジェクト名の長さは、*DestNameLength* によって指定されます。この長さがゼロの場合、宛先オブジェクト名はなく、*DestNameOffset* は無視されます。宛先オブジェクト名が存在する場合は、構造体の先頭から *StrucLength* バイト以内に完全に存在する必要があります。

アプリケーションは、宛先オブジェクト名が *SrcEnvOffset*、*SrcNameOffset*、および *DestEnvOffset* フィールドによってアドレス指定されたデータのいずれかに隣接していると想定してはなりません。

このフィールドの初期値は 0 です。

DataLogicalLength (MQLONG)

DataLogicalLength フィールドは、MQRMH 構造体によって参照されるバルク・データの長さを指定します。

バルク・データが実際にメッセージ内に存在する場合、データは MQRMH 構造体の先頭から *StrucLength* バイトのオフセットから始まります。メッセージ全体の長さから *StrucLength* を引いた値が、存在するバルク・データの長さになります。

メッセージにデータがある場合、*DataLogicalLength* は関連するデータの量を指定します。通常、*DataLogicalLength* は、メッセージに示されているデータの長さと同じ値になります。

MQRMH 構造体がオブジェクト内の残りのデータ (指定された論理オフセットから始まる) を表す場合、バルク・データがメッセージ内に実際に存在しなければ、*DataLogicalLength* に値ゼロを使用することができます。

データがない場合は、MQRMH の端はメッセージの端に一致します。

このフィールドの初期値は 0 です。

DataLogicalOffset (MQLONG)

このフィールドは、バルク・データがその一部を形成するオブジェクトの先頭からのバルク・データのオフセットを指定します。オブジェクトの先頭からのバルク・データのオフセットを、論理オフセットと呼びます。これは、MQRMH 構造体の先頭からのバルク・データの物理オフセットではありません。物理オフセットは *StrucLength* に指定します。

論理オフセットは、参照メッセージを使用して大きいオブジェクトを送信できるように 2 つに分割されます。実際の論理オフセットは次の 2 つのフィールドの合計によって得られます。

- *DataLogicalOffset* フィールド。このフィールドは、論理オフセットを 1 000 000 000 で割ったときの余りを表します。したがって、この値の範囲は 0 以上 999 999 999 以下です。
- *DataLogicalOffset2* フィールド。このフィールドは、論理オフセットを 1 000 000 000 で割ったときの商を表します。したがって、この値は論理オフセット中にある 1 000 000 000 の倍数です。この倍数の範囲は 0 以上 999 999 999 以下です。

このフィールドの初期値は 0 です。

DataLogicalOffset2 (MQLONG)

このフィールドは、バルク・データがその一部を形成するオブジェクトの先頭からのバルク・データの高オフセットを指定します。この値の範囲は 0 以上 999 999 999 以下です。詳細は、*DataLogicalOffset* を参照してください。

このフィールドの初期値は 0 です。

MQRR - 応答レコード

宛先が配布リストである場合、単一の宛先キューに対するオープンまたはプット操作の結果として生成される完了コードと理由コードを受信するには、MQRR 構造体を使用します。MQRR は MQOPEN 呼び出し、MQPUT 呼び出し、および MQPUT1 呼び出しのための出力構造体です。

可用性

MQRR 構造体は、以下のプラットフォームで利用可能です：

-  AIX
-  IBM i
-  Linux
-  Windows

および、これらのシステムに接続された IBM MQ クライアント。

文字セットとエンコード

MQRR 内のデータは、キューマネージャ **CodedCharSetId** 属性およびローカルキューマネージャのエンコーディング (MQENC_NATIVE で指定) によって与えられる文字セットでなければならない。ただし、アプリケーションが MQ MQI クライアントとして実行されている場合、構造体はクライアントの文字セットとエンコードに従っている必要があります。

使用法

MQOPEN および MQPUT 呼び出し、あるいは MQPUT1 呼び出しにおいてこれらの構造体の配列を提供することで、呼び出しの結果が混合している場合（つまり、リスト内の特定のキューでは呼び出しが成功する

が、他のキューでは失敗する場合)に、配布リスト内の全キューの完了コードと理由コードを特定できます。その呼び出しからの理由コード MQRC_MULTIPLE_REASONS は、(アプリケーションによって与えられた場合) キュー・マネージャーが応答レコードを設定したことを示します。

フィールド

注: 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 551. MQRR のフィールド		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
<u>CompCode</u> (キューの完了コード)	MQCC_OK	0
<u>理由</u> (キューの理由コード)	MQRC_NONE	0
注: 1. C プログラミング言語では、マクロ変数 MQRR_DEFAULT には、表にリストされている値が含まれています。このマクロ変数を以下の方法で使用して、構造体のフィールドに初期値を設定します。 <pre>MQRR MyRR = {MQRR_DEFAULT};</pre>		

言語ごとの宣言

MQRR 用 C 宣言

```
typedef struct tagMQRR MQRR;
struct tagMQRR {
    MQLONG  CompCode; /* Completion code for queue */
    MQLONG  Reason;   /* Reason code for queue */
};
```

COBOL MQRR に関する宣言

```
**      MQRR structure
10 MQRR.
**      Completion code for queue
15 MQRR-COMPCODE PIC S9(9) BINARY.
**      Reason code for queue
15 MQRR-REASON   PIC S9(9) BINARY.
```

PL/I MQRR に関する宣言

```
dcl
  1 MQRR based,
    3 CompCode fixed bin(31), /* Completion code for queue */
    3 Reason   fixed bin(31); /* Reason code for queue */
```

MQRR 用の Visual Basic 宣言

```
Type MQRR
    CompCode As Long 'Completion code for queue'
    Reason   As Long 'Reason code for queue'
End Type
```

MQRR のフィールド詳細

宛先が配布リストである場合、単一の宛先キューに対するオープンまたはプット操作の結果として生成される完了コードおよび理由コードを受信するには、MQRR 応答レコード構造を使用します。MQRR は MQOPEN 呼び出し、MQPUT 呼び出し、および MQPUT1 呼び出しのための出力構造体です。

フィールド

注: 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 552. MQRR のフィールド		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
<u>CompCode</u> (キューの完了コード)	MQCC_OK	0
<u>理由</u> (キューの理由コード)	MQRC_NONE	0
注: 1. C プログラミング言語では、マクロ変数 MQRR_DEFAULT には、表にリストされている値が含まれています。このマクロ変数を以下の方法で使用して、構造体のフィールドに初期値を設定します。 <pre>MQRR MyRR = {MQRR_DEFAULT};</pre>		

CompCode (MQLONG)

これは、キューに対するオープン操作または PUT 操作の結果生じる完了コードです。このキューの名前は、MQOPEN 呼び出しまたは MQPUT1 呼び出しで与えられた MQOR 構造体の配列内の対応する要素に指定されていた名前です。

これは、常に出力フィールドです。このフィールドの初期値は、MQCC_OK です。

理由 (MQLONG)

これは、キューに対するオープン操作または PUT 操作の結果生じる理由コードです。このキューの名前は、MQOPEN 呼び出しまたは MQPUT1 呼び出しで与えられた MQOR 構造体の配列内の対応する要素に指定されていた名前です。

これは、常に出力フィールドです。このフィールドの初期値は MQRC_NONE です。

MQSCO - SSL/TLS 構成オプション

MQSCO 構造体は、MQCD 構造体内の TLS フィールドと連動して、チャネルプロトコルが TCP/IP である場合に、TLS を使用したクライアント接続を制御する設定オプションを、プロセス IBM MQ MQI client として実行されているアプリケーションが指定できるようにします。この構造体は、MQCONN 呼び出しの入力パラメーターです。

可用性

MQSCO 構造は、以下のオペレーティングシステムで利用可能です：

-  AIX
-  IBM i
-  Linux
-  Windows

クライアント・チャネルのチャネル・プロトコルが TCP/IP でない場合、MQSCO 構造体は無視されます。

バージョン

サポート対象のプログラミング言語向けに提供されるヘッダー、COPY、および INCLUDE ファイルには、最新のバージョンの MQSCO が含まれていますが、フィールド Version の初期値は MQSCO_VERSION_1

に設定されています。version-1 構造体に存在しないフィールドを使用するには、アプリケーションは必要なバージョン番号を Version フィールドに設定する必要があります。

文字セットとエンコード

MQSCO 内のデータは、キュー・マネージャ **CodedCharSetId** 属性で指定された文字セットでなければならず、ローカル・キュー・マネージャのエンコーディングは MQENC_NATIVE で指定される。

フィールド

注: 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 553. MQSCO のフィールド		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
StrucId (構造識別子)	MQSCO_STRUC_ID	'SCO~'
バージョン (構造バージョン番号)	MQSCO_CURRENT_VERSION	1
KeyRepository (キーリポジトリの場所)	なし	ヌル・ストリングまたはブランク
CryptoHardware (暗号ハードウェアの詳細)	なし	ヌル・ストリングまたはブランク
AuthInfoRecCount (存在する MQAIR レコードの数)	なし	0
AuthInfoRecOffset (MQSCO の先頭からの最初の MQAIR レコードのオフセット)	なし	0
AuthInfoRecPtr (最初の MQAIR レコードのアドレス)	なし	ヌル・ポインターまたはヌル・バイト
注: Version が MQSCO_VERSION_2 より小さい場合、以下の 2 つのフィールドは無視されます。		
KeyResetCount (TLS 秘密鍵リセット回数)	MQSCO_RESET_COUNT_DEFAULT	0
547 ページの『FipsRequired (MQLONG)』 (認定暗号 FIPS アルゴリズムを使用する IBM MQ)	MQSSL_FIPS_NO	0
注: Version が MQSCO_VERSION_3 より小さい場合、以下の 2 つのフィールドは無視されます。		
EncryptionPolicySuiteB (Suite B 暗号アルゴリズムのみを使用する)	MQ_SUITE_B_NONE, MQ_SUITE_B_NOT_AVAILABLE, MQ_SUITE_B_NOT_AVAILABLE, MQ_SUITE_B_NOT_AVAILABLE	1, 0, 0, 0
注: Version が MQSCO_VERSION_4 より小さい場合、以下の 2 つのフィールドは無視されます。		
CertificateValPolicy (証明書検証ポリシー)	MQ_CERT_VAL_POLICY_DEFAULT	0
注: Version が MQSCO_VERSION_5 より小さい場合、以下の 2 つのフィールドは無視されます。		

表 553. MQSCO のフィールド (続き)

フィールド名と説明	定数の名前	定数の初期値 (存在する場合)
<u>CertificateLabel</u> (使用中の証明書ラベルの詳細)	なし	ヌル・ストリングまたはブランク
注: 以下の 3 つのフィールドは、がより小さい <code>VersionMQSCO_VERSION_6</code> 場合に無視される。		
<u>KeyRepoPasswordPtr</u> (TLS キーリポジトリのパスワード)	なし	ヌル・ポインターまたはヌル・バイト
<u>KeyRepoPasswordOffset</u> (TLS キーリポジトリパスワードのオフセット)	なし	0
<u>KeyRepoPasswordLength</u> (TLS キーリポジトリのパスワードの長さ)	なし	0
注: 以下の 5 つのフィールドは、がより小さい <code>Version</code> 場合に無視 <code>MQSCO_VERSION_7</code> されます。		
V 30.0.0 <u>HTTPSKeyStorePtr</u> (HTTPSKeyStore のアドレス文字列)	なし	NULL ポインター
V 30.0.0 <u>HTTPSKeyStoreOffset</u> (HTTPSKeyStore 文字列のオフセット) MQSCO の先頭からの位置	なし	0
V 30.0.0 <u>HTTPSKeyStoreLength</u> (HTTPSKeyStore 文字列の長さ)	なし	0
V 30.0.0 <u>HTTSCertValidation</u> (必要な証明書検証レベル) <code>MQ_HTTPSCERTVAL_DEFAULT</code> (ファイル <code>mqclient.ini</code> に設定が存在する場合、その設定が使用されることを意味する)	<code>MQ_HTTPSCERTVAL_DEFAULT</code> <code>MQ_HTTPSCERTVAL_ANY</code> <code>MQ_HTTPSCERTVAL_NONE</code> <code>MQ_HTTPSCERTVAL_HOSTNAMECN</code>	0 1 2 3
V 30.0.0 <u>HTTSCertRevocation</u> (証明書失効チェックの必要レベル) <code>MQ_HTTPSCERTREV_DEFAULT</code> (ファイル <code>mqclient.ini</code> に設定が存在する場合、その設定が使用されることを意味する)	<code>MQ_HTTPSCERTREV_DEFAULT</code> <code>MQ_HTTPSCERTREV_REQUIRED</code> <code>MQ_HTTPSCERTREV_DISABLED</code> <code>MQ_HTTPSCERTREV_OPTIONAL</code>	0 1 2 3

注:

1. 記号↵は、単一のブランク文字を表します。
2. C プログラミング言語では、マクロ変数 `MQSCO_DEFAULT` には、表にリストされている値が含まれます。このマクロ変数を以下の方法で使用して、構造体のフィールドに初期値を設定します。

```
MQSCO MySCO = {MQSCO_DEFAULT};
```

言語ごとの宣言

MQSCO の C 宣言

```
typedef struct tagMQSCO MQSCO;
struct tagMQSCO {
    MQCHAR4      StrucId;           /* Structure identifier */
    MQLONG       Version;          /* Structure version number */
    MQCHAR256    KeyRepository;    /* Location of TLS key */
    MQCHAR256    CryptoHardware;   /* repository */
    MQCHAR256    CryptoHardware;   /* Cryptographic hardware */
    MQLONG       AuthInfoRecCount; /* configuration string */
    MQLONG       AuthInfoRecOffset; /* Number of MQAIR records */
    PMQAIR       AuthInfoRecPtr;   /* present */
    MQLONG       AuthInfoRecOffset; /* Offset of first MQAIR */
    MQLONG       AuthInfoRecOffset; /* record from start of */
    MQLONG       AuthInfoRecOffset; /* MQSCO structure */
    PMQAIR       AuthInfoRecPtr;   /* Address of first MQAIR */
    MQLONG       KeyResetCount;    /* record */
    MQLONG       KeyResetCount;    /* Number of unencrypted */
    MQLONG       KeyResetCount;    /* bytes sent/received */
    MQLONG       KeyResetCount;    /* before secret key is */
    MQLONG       KeyResetCount;    /* reset */
    MQLONG       FipsRequired;     /* Using FIPS certified */
    MQLONG       EncryptionPolicySuiteB[4]; /* algorithms */
    MQLONG       EncryptionPolicySuiteB[4]; /* Use only Suite B */
    MQLONG       CertificateValPolicy; /* cryptographic algorithms */
    MQLONG       CertificateValPolicy; /* Certificate validation */
    MQLONG       CertificateValPolicy; /* policy */
    MQCHAR64     CertificateLabel; /* Certificate label */
    MQPTR        KeyRepoPasswordPtr; /* Address of key */
    MQLONG       KeyRepoPasswordOffset; /* repository password */
    MQLONG       KeyRepoPasswordOffset; /* Offset of key repository */
    MQLONG       KeyRepoPasswordLength; /* password */
    MQLONG       KeyRepoPasswordLength; /* Length of key repository */
    MQLONG       KeyRepoPasswordLength; /* password */
    MQLONG       HTTPSCertValidation; /* HTTPS certificate */
    MQLONG       HTTPSCertRevocation; /* validation level */
    MQLONG       HTTPSCertRevocation; /* HTTPS certificate */
    MQLONG       HTTPSCertRevocation; /* revocation level */
    MQPTR        HTTPSKeyStorePtr; /* Address of HTTPS */
    MQLONG       HTTPSKeyStoreOffset; /* Keystore */
    MQLONG       HTTPSKeyStoreLength; /* Offset of HTTPS Keystore */
    MQLONG       HTTPSKeyStoreLength; /* Length of HTTPS keystore */
};
```

MQSCO の COBOL 宣言

```
** MQSCO structure
10 MQSCO.
** Structure identifier
15 MQSCO-STRUCID PIC X(4).
** Structure version number
15 MQSCO-VERSION PIC S9(9) BINARY.
** Location of TLS key repository
15 MQSCO-KEYREPOSITORY PIC X(256).
** Cryptographic hardware configuration string
15 MQSCO-CRYPTOHARDWARE PIC X(256).
** Number of MQAIR records present
15 MQSCO-AUTHINFORECCOUNT PIC S9(9) BINARY.
** Offset of first MQAIR record from start of MQSCO structure
15 MQSCO-AUTHINFORECOFFSET PIC S9(9) BINARY.
** Address of first MQAIR record
15 MQSCO-AUTHINFORECPTTR POINTER.
** Version 1 **
** Number of unencrypted bytes sent/received before secret key is
** reset
15 MQSCO-KEYRESETCOUNT PIC S9(9) BINARY.
** Using FIPS certified algorithms
15 MQSCO-FIPSREQUIRED PIC S9(9) BINARY.
** Version 2 **
** Use only Suite B cryptographic algorithms
```



```

15 MQSCO-ENCRYPTIONPOLICYSUITEB PIC S9(9) BINARY OCCURS 4.
** Version 3 **
** Certificate validation policy setting
15 MQSCO-CERTIFICATEVALPOLICY PIC S9(9) BINARY.
** Version 4 **
** SSL/TLS certificate label
15 MQSCO-CERTIFICATELABEL PIC X(64).
** Version 5 **
** Add padding to ensure that pointers start on correct
** boundaries
15 FILLER PIC S9(9) BINARY VALUE 0.
** Address of key repository password
15 MQSCO-KEYREPOPASSWORDPTR POINTER.
** Offset of key repository password
15 MQSCO-KEYREPOPASSWORDOFFSET PIC S9(9) BINARY.
** Length of key repository password
15 MQSCO-KEYREPOPASSWORDLENGTH PIC S9(9) BINARY.
** Version 6 ** V 10.0.0 ** HTTPS certificate validation level
15 MQSCO-HTTPSCERTVALIDATION PIC S9(9) BINARY VALUE 0.
** HTTPS certificate revocation level
15 MQSCO-HTTPSCERTREVOCATION PIC S9(9) BINARY VALUE 0.
** Address of HTTPS Keystore
15 MQSCO-HTTPSKEYSTOREPTR POINTER VALUE NULL.
** Offset of HTTPS Keystore
15 MQSCO-HTTPSKEYSTOREOFFSET PIC S9(9) BINARY VALUE 0.
** Length of HTTPS keystore
15 MQSCO-HTTPSKEYSTORELENGTH PIC S9(9) BINARY VALUE 0.
** Ver:7 **

```

MQSCO の PL/I 宣言

```

dcl
1 MQSCO based,
3 StrucId char(4), /* Structure identifier */
3 Version fixed bin(31), /* Structure version number */
3 KeyRepository char(256), /* Location of TLS key
repository */
3 CryptoHardware char(256), /* Cryptographic hardware
configuration string */
3 AuthInfoRecCount fixed bin(31), /* Number of MQAIR records
present */
3 AuthInfoRecOffset fixed bin(31), /* Offset of first MQAIR record
from start of MQSCO structure */
3 AuthInfoRecPtr pointer, /* Address of first MQAIR record */
3 KeyResetCount fixed bin(31), /* Key reset count */
/* Version 1 */
3 FipsRequired fixed bin(31), /* FIPS required */
/* Version 2 */
3 EncryptionPolicySuiteB (4) fixed bin(31), /* Suite B encryption policy */
/* Version 3 */
3 CertificateValPolicy fixed bin(31), /* Certificate validation policy */
/* Version 4 */
3 CertificateLabel char(64), /* SSL/TLS certificate label */
/* Version 5 */
3 KeyRepoPasswordPtr pointer, /* Address of key repository
password */
3 KeyRepoPasswordOffset fixed bin(31), /* Offset of key repository
password */
3 KeyRepoPasswordLength fixed bin(31); /* Length of key repository
password */
/* Version 6 */ V 10.0.0
3 HTTPSCertValidation fixed bin(31), /* HTTPS certificate */
validation level */
3 HTTPSCertRevocation fixed bin(31), /* HTTPS certificate */
revocation level */
3 HTTPSKeyStorePtr pointer, /* Address of HTTPS Keystore */
3 HTTPSKeyStoreOffset fixed bin(31), /* Offset of HTTPS Keystore */
3 HTTPSKeyStoreLength fixed bin(31); /* Length of HTTPS keystore */
/* Version 7 */

```

MQSCO の Visual Basic 宣言

Type MQSCO		
StrucId	As String*4	'Structure identifier'
Version	As Long	'Structure version number'
KeyRepository	As String*256	'Location of TLS key repository'
CryptoHardware	As String*256	'Cryptographic hardware configuration'
		'string'
AuthInfoRecCount	As Long	'Number of MQAIR records present'

AuthInfoRecOffset	As Long	'Offset of first MQAIR record from'
AuthInfoRecPtr	As MQPTR	'start of MQSCO structure'
KeyResetCount	As Long	'Address of first MQAIR record'
is reset'		'Number of unencrypted bytes sent/received before secret key
'Version 1'		
FipsRequired	As Long	'Mandatory FIPS CipherSpecs?'
'Version 2'		
End Type		

関連資料

265 ページの『MQCNO - 接続オプション』

MQCNO 構造により、アプリケーションはキューマネージャへの接続に関連するオプションを指定できます。この構造体は、MQCONN 呼び出しの入出力パラメーターです。

MQSCO のフィールド詳細

MQSCO 構造体（SSL / TLS）の設定オプションと、MQCD 構造体内の TLS フィールドを組み合わせることで、チャネルプロトコルが TCP/IP である場合、TLS を使用したクライアント接続を制御する設定オプションを、として IBM MQ MQI client 実行されるアプリケーションが指定できるようになります。この構造体は、MQCONN 呼び出しの入力パラメーターです。

フィールド

注：次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 554. MQSCO のフィールド		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
StrucId (構造識別子)	MQSCO_STRUC_ID	'SCO-'
バージョン (構造バージョン番号)	MQSCO_CURRENT_VERSION	1
KeyRepository (キーリポジトリの場所)	なし	ヌル・ストリングまたはブランク
CryptoHardware (暗号ハードウェアの詳細)	なし	ヌル・ストリングまたはブランク
AuthInfoRecCount (存在する MQAIR レコードの数)	なし	0
AuthInfoRecOffset (MQSCO の先頭からの最初の MQAIR レコードのオフセット)	なし	0
AuthInfoRecPtr (最初の MQAIR レコードのアドレス)	なし	ヌル・ポインターまたはヌル・バイト
注：Version が MQSCO_VERSION_2 より小さい場合、以下の 2 つのフィールドは無視されます。		
KeyResetCount (TLS 秘密鍵リセット回数)	MQSCO_RESET_COUNT_DEFAULT	0
547 ページの『FipsRequired (MQLONG)』(認定暗号 FIPS アルゴリズムを使用する IBM MQ)	MQSSL_FIPS_NO	0
注：Version が MQSCO_VERSION_3 より小さい場合、以下の 2 つのフィールドは無視されます。		

表 554. MQSCO のフィールド (続き)		
フィールド名と説明	定数の名前	定数の初期値 (存在する場合)
<u>EncryptionPolicySuiteB</u> (Suite B 暗号アルゴリズムのみを使用する)	MQ_SUITE_B_NONE, MQ_SUITE_B_NOT_AVAILABLE, MQ_SUITE_B_NOT_AVAILABLE, MQ_SUITE_B_NOT_AVAILABLE	1, 0, 0, 0
注: <i>Version</i> が MQSCO_VERSION_4 より小さい場合、以下の 2 つのフィールドは無視されます。		
<u>CertificateValPolicy</u> (証明書検証ポリシー)	MQ_CERT_VAL_POLICY_DEFAULT	0
注: <i>Version</i> が MQSCO_VERSION_5 より小さい場合、以下の 2 つのフィールドは無視されます。		
<u>CertificateLabel</u> (使用中の証明書ラベルの詳細)	なし	ヌル・ストリングまたは ブランク
注: 以下の 3 つのフィールドは、がより小さい <i>Version</i> MQSCO_VERSION_6 場合に無視される。		
<u>KeyRepoPasswordPtr</u> (TLS キーリポジトリのパスワード)	なし	ヌル・ポインターまたは ヌル・バイト
<u>KeyRepoPasswordOffset</u> (TLS キーリポジトリパスワードのオフセット)	なし	0
<u>KeyRepoPasswordLength</u> (TLS キーリポジトリのパスワードの長さ)	なし	0
注: 以下の 5 つのフィールドは、がより小さい <i>Version</i> 場合に無視 MQSCO_VERSION_7 されます。		
V 30.0.0 <u>HTTPSKeyStorePtr</u> (HTTPSKeyStore のアドレス文字列)	なし	NULL ポインター
V 30.0.0 <u>HTTPSKeyStoreOffset</u> (HTTPSKeyStore 文字列のオフセット) MQSCO の先頭からの位置	なし	0
V 30.0.0 <u>HTTPSKeyStoreLength</u> (HTTPSKeyStore 文字列の長さ)	なし	0
V 30.0.0 <u>HTTPSCertValidation</u> (必要な証明書検証レベル) MQ_HTTPSCERTVAL_DEFAULT (ファイル <code>mqclient.ini</code> に設定が存在する場合、その設定が使用されることを意味する)	MQ_HTTPSCERTVAL_DEFAULT MQ_HTTPSCERTVAL_ANY MQ_HTTPSCERTVAL_NONE MQ_HTTPSCERTVAL_HOSTNAMECN	0 1 2 3

表 554. MQSCO のフィールド (続き)

フィールド名と説明	定数の名前	定数の初期値 (存在する場合)
V 10.0.0 HTTPSCertRevocation (証明書失効チェックの必要レベル) MQ_HTTPSCERTREV_DEFAULT (ファイル mqclient.ini に設定が存在する場合、その設定が使用されることを意味する)	MQ_HTTPSCERTREV_DEFAULT MQ_HTTPSCERTREV_REQUIRED MQ_HTTPSCERTREV_DISABLED MQ_HTTPSCERTREV_OPTIONAL	0 1 2 3

注:

1. 記号-は、単一のブランク文字を表します。
2. C プログラミング言語では、マクロ変数 MQSCO_DEFAULT には、表にリストされている値が含まれます。このマクロ変数を以下の方法で使用して、構造体のフィールドに初期値を設定します。

```
MQSCO MySCO = {MQSCO_DEFAULT};
```

StrucId (MQCHAR4)

これは、SSL / TLS 設定オプション構造体の構造体識別子です。常に入力フィールドです。その値は MQSCO_STRUC_ID です。

値は次のものでなければなりません。

MQSCO_STRUC_ID

SSL / TLS 設定オプション構造体の識別子。

C プログラミング言語では、定数 MQSCO_STRUC_ID_ARRAY も定義されている。これは MQSCO_STRUC_ID と同じ値を持ちますが、文字列ではなく文字の配列です。

Version (MQLONG)

これは構造体のバージョン番号です。値は次の文字列のいずれかでなければなりません:

MQSCO_VERSION_1

バージョン 1 の TLS 構成オプション構造体。

MQSCO_VERSION_2

バージョン 2 の TLS 構成オプション構造体。

MQSCO_VERSION_3

バージョン 3 の TLS 構成オプション構造体。

MQSCO_VERSION_4

バージョン 4 の TLS 構成オプション構造体。

MQSCO_VERSION_5

バージョン 5 の TLS 構成オプション構造体。

MQSCO_VERSION_6

Version-6 TLS 設定オプション構造体。

V 10.0.0 MQSCO_VERSION_7

Version-7 TLS 設定オプション構造体。

これより新しいバージョンの構造体にのみ存在するフィールドは、そのフィールドの説明にその旨記載されています。以下の定数は、現行バージョンのバージョン番号を指定しています。

MQSCO_CURRENT_VERSION

TLS 構成オプション構造体の現行バージョン。

これは常に入力フィールドです。このフィールドの初期値は MQSCO_VERSION_1 です。

KeyRepository (MQCHAR256)

このフィールドは、AIX, Linux, and Windows IBM i システム上で実行 IBM MQ MQI clients する場合にのみ関連します。このフィールドは、鍵および証明書が保管される鍵データベース・ファイルの場所を指定します。ファイルの拡張子が指定されていない場合、自動的に という拡張子が .kdb 追加されます。

各キーデータベースファイルには、関連付けられたパスワード保管ファイルを持つことができます。隠しファイルには、キーデータベースへのプログラムによるアクセスを許可するために使用される暗号化されたパスワードが格納されています。パスワード・スタッシュ・ファイルは、キー・データベースと同じディレクトリに存在し、鍵データベースと同じファイル語幹がなければならず、最後にサフィックス .sth を付けなければなりません。

例えば、キーデータベースファイルがである場合、パスワード保管 /xxx/yyy/key.kdb ファイルはでなければならない。ここで xxx および yyy/xxx/yyy/key.sth はディレクトリ名を表す。

キーデータベースのパスワードは、MQSCO 構造体の ``KeyRepoPasswordPtr または KeyRepoPasswordOffset`` フィールドを使用して指定することもできます。

値がフィールドの長さより短い場合、値をヌル文字で終了するか、フィールドの長さまでブランクを埋め込みます。値は検査されません。キー・リポジトリのアクセス中にエラーがあった場合、呼び出しは失敗し、理由コード MQRC_KEY_REPOSITORY_ERROR が戻ります。

IBM MQ MQI client から TLS 接続を実行するには、KeyRepository に有効なキー・データベース・ファイル名を設定します。

これは入力フィールドです。このフィールドの長さは MQ_SSL_KEY_REPOSITORY_LENGTH によって指定されます。初期値は、C 言語ではヌル・ストリングですが、その他のプログラミング言語ではブランク文字です。

CryptoHardware (MQCHAR256)

このフィールドは、クライアント・システムに接続される暗号ハードウェアの構成の詳細を提供するフィールドです。

フィールドは、次のフォーマットのストリングに設定するか、ブランクまたはヌルのままにします。

```
GSK_PKCS11=the PKCS #11 driver path and file name;the PKCS #11 token label;the PKCS #11 token password;symmetric cipher setting;
```

PKCS #11 インターフェース (例えば IBM 4960 または IBM 4764) に準拠する暗号ハードウェアを使用するには、PKCS #11 ドライバー・パス、PKCS #11 トークン・ラベル、および PKCS #11 トークン・パスワード・ストリングを、それぞれ最後にセミコロンを付けて指定する必要があります。

PKCS #11 ドライバー・パスは、PKCS #11 カードに対するサポートを提供する共有ライブラリーの絶対パスです。PKCS #11 ドライバー・ファイル名は共有ライブラリーの名前です。PKCS #11 パスおよびファイル名に必要な値の例を以下に示します。

```
/usr/lib/pkcs11/PKCS11_API.so
```

PKCS #11 トークンのラベルは、ハードウェアに設定したラベルと一致している必要があります。

暗号ハードウェア構成が不要な場合には、このフィールドをブランクまたはヌルにします。

値がフィールドの長さより短い場合、値をヌル文字で終了するか、フィールドの長さまでブランクを埋め込みます。値が無効か、または暗号ハードウェアの構成に使用したときに失敗した場合、呼び出しは失敗し、理由コード MQRC_CRYPTOHARDWARE_ERROR が戻ります。

これは入力フィールドです。このフィールドの長さは `MQ_SSL_CRYPTO_HARDWARE_LENGTH` によって指定されます。初期値は、C 言語ではヌル・ストリングですが、その他のプログラミング言語ではブランク文字です。

AuthInfoRecCount (MQLONG)

これは、`AuthInfoRecPtr` または `AuthInfoRecOffset` フィールドによってアドレス指定された認証情報 (MQAIR) レコードの数です。詳しくは、212 ページの『MQAIR - 認証情報レコード』を参照してください。値はゼロ以上でなければなりません。この値が無効の場合、呼び出しは失敗し、理由コード `MQRC_AUTH_INFO_REC_COUNT_ERROR` が戻ります。

これは入力フィールドです。このフィールドの初期値は 0 です。

AuthInfoRecOffset (MQLONG)

これは、MQSCO 構造体の先頭からの最初の認証情報レコードのオフセットをバイト数で表したものです。オフセットの値は、正負どちらの値にもなります。`AuthInfoRecCount` がゼロの場合、このフィールドは無視されます。

`AuthInfoRecOffset` または `AuthInfoRecPtr` のいずれかを使用して MQAIR レコードを指定できますが、両方を使用することはできません。詳しくは、`AuthInfoRecPtr` フィールドの説明を参照してください。

これは入力フィールドです。このフィールドの初期値は 0 です。

AuthInfoRecPtr (PMQAIR)

これは、最初の認証情報レコードのアドレスです。`AuthInfoRecCount` がゼロの場合、このフィールドは無視されます。

MQAIR レコードの配列を提供するには、次の 2 つの方法があります。

- `AuthInfoRecPtr` ポインター・フィールドを使用する

この場合、アプリケーションは MQSCO 構造体とは別個に MQAIR レコードの配列を宣言でき、その配列のアドレスに `AuthInfoRecPtr` を設定できます。

他の環境へ移植できる形式のポインター・データ・タイプをサポートするプログラミング言語 (C プログラミング言語など) の場合は、`AuthInfoRecPtr` の使用を検討してください。

- `AuthInfoRecOffset` オフセット・フィールドを使用する

この場合、アプリケーションは MQSCO と後に続く MQAIR レコードを含む、複合構造体を宣言する必要があります。さらに `AuthInfoRecOffset` を、MQSCO 構造体の先頭からのその配列の最初のレコードのオフセットに設定する必要があります。この値が正しいこと、および値が MQLONG 内に収まることを確認してください (最も制限の大きいプログラミング言語は COBOL で、有効範囲は -999 999 999 から +999 999 999 です)。

ポインターのデータ・タイプをサポートしていないプログラミング言語や、他の環境に移植できない形式のポインター・データ・タイプを実装しているプログラミング言語 (COBOL プログラミング言語など) の場合には、`AuthInfoRecOffset` の使用を検討してください。

どの方法を選んでも、`AuthInfoRecPtr` または `AuthInfoRecOffset` のいずれか一方のみを使用できます。両方ともゼロでない場合、呼び出しは失敗し、理由コード `MQRC_AUTH_INFO_REC_ERROR` が戻ります。

これは入力フィールドです。このフィールドの初期値は、ポインターをサポートするプログラミング言語のヌル・ポインターです。それ以外の場合は、すべてヌルのバイトのストリングです。

注: プログラミング言語がポインターのデータ・タイプをサポートしていないプラットフォームでは、このフィールドは適切な長さのバイト・ストリングとして宣言されます。

KeyResetCount (MQLONG)

これは、秘密鍵が再交渉される前に、TLS の会話内で送受信された非暗号化バイト数の合計を表します。

このバイト数には、MCA によって送信される制御情報が含まれます。

TLS 秘密鍵のリセット・カウントを 1 バイトから 32 KB の範囲で指定すると、TLS チャネルは 32 KB の秘密鍵リセット・カウントを使用します。これは、TLS 秘密鍵リセット値が小さい場合に生じる、過剰な鍵リセットによる処理コストを避けるためです。

これは入力フィールドです。この値は、0 から 999 999 999 までの範囲の数値です。デフォルト値は 0 です。共通鍵が再折衝されないことを指定する場合には、値 0 を使用してください。

FipsRequired (MQLONG)

IBM MQ 暗号ハードウェアで構成可能であり、使用される暗号モジュールは当該ハードウェア製品が提供するものである。これらは使用中の暗号ハードウェア製品に応じて特定のレベルで認証 FIPS 可能である。

このフィールドを使用して、暗号化機能が提供 IBM MQ ソフトウェアで提供される場合、認証済みアルゴリズムのみ FIPS が使用されることを指定します。

注：

-  WindowsAIX、Linux、およびにおいて、IBM MQ は GSKit 9 IBM Crypto for C (ICC) 暗号モジュールを通じてへの準拠を実現 FIPS 140-3 しています。この FIPS 140-3 モジュールに関連する NIST 認証は、<https://csrc.nist.gov/projects/cryptographic-module-validation-program/certificate/4755> で確認できます。
 -  Linux s390xFIPS 現在、への対応は行っていません。IBM MQ このプラットフォームをご利用のお客様でサポートが必要な FIPS 場合は、以前のバージョンのままご利用ください。このプラットフォーム上で暗号モジュールが IBM Crypto for C (ICC) 認証を取得次第、今後の FixPack でサポートが FIPS 140-3 有効化されます。
 -  このソフトウェアに含まれる暗号 FIPS 140-3 モジュールは、2024 IBM Semeru Runtime 年 8 月に NIST の承認を受けた。IBM Semeru Runtime 11+IBM MQ 10.0.0 TLS を使用したおよび IBM MQ classes for Java のクライアント接続の IBM MQ classes for JMS 処理機能を、FIPS 140-3 およびに Java 8 追加しました。<https://csrc.nist.gov/projects/cryptographic-module-validation-program/certificate/4755> この FIPS 140-3 モジュールに関連する NIST 認証は、以下で確認できます。
 - Containers では IBM MQ、3.2.0 およびキューマネージャ IBM MQ Operator 一のコンテナイメージ 9.4.0.0 以降は、UBI 9 をベースとしています。FIPS 140-3 Containers における IBM MQ コンプライアンスは、現在審査中です。
-  が有効になっている FIPS 場合、コンテナ制御 IBM MQ プロセスでは、認定 FIPS 140-3 OpenSSL モジュールが使用されます。NIST 認証の詳細については、以下の URL をご覧ください：<https://access.redhat.com/compliance/fips>。IBM MQ コンテナイメージ内で実行されるキューマネージャは、ベースイメージのプラットフォームバージョンと同じ FIPS IBM MQ 認証レベルを持っています。

TLS IBM MQ 暗号化の実装がインストールされると、いくつかの FIPS 認証済みモジュールを提供する実装も同時にインストールされます。

値は次のいずれかです。

MQSSL_FIPS_NO

これがデフォルト値です。この値に設定した場合、次のことが保証されます。

- 特定のプラットフォームでサポートされるすべての CipherSpec を使用できます。
- 暗号ハードウェアを使用せずに実行する場合、CipherSpecs はプラットフォーム上で認定暗号技術を使用して実行されます FIPS 140-2。IBM MQ

FIPS 認定の CipherSpec のリストについては、[CipherSpec の有効化](#)に記載の表を参照してください。

MQSSL_FIPS_YES

この値に設定した場合、暗号ハードウェアを使って暗号化を実行していない限り、次のことが保証されます。

- このクライアント接続に適用される CipherSpec では、認証済み暗号 FIPS アルゴリズムのみを使用できます。
- インバウンドおよびアウトバウンド TLS チャンネル接続は、特定の CipherSpec が使用されている場合にのみ成功します。

詳しくは、[CipherSpec の有効化](#)を参照してください。

注：可能な場合、FIPS 専用 CipherSpecs が設定されていると、MQI クライアントは、FIPS 以外の CipherSpec を指定した接続を MQRC_SSL_INITIALIZATION_ERROR で拒否します。IBM MQ すべてのそのような接続を拒否することを保証するのではなく、設定 IBM MQ が準拠 FIPS しているかどうかを判断するのはあなたの責任です。

EncryptionPolicySuiteB (MQLONG)

このフィールドは、スイート B 準拠の暗号方式が使用されるかどうかと、使用される強度レベルを指定します。値は以下のいずれかです (複数可)。

- MQ_SUITE_B_NONE
スイート B 準拠の暗号方式は使用されません。
- MQ_SUITE_B_128_BIT
128 ビットの強度の Suite B セキュリティーを使用します。
- MQ_SUITE_B_192_BIT
192 ビットの強度の Suite B セキュリティーを使用します。

注：MQ_SUITE_B_NONE を他の値と併用すると、このフィールドは無効になります。

CertificateValPolicy (MQLONG)

このフィールドは、どのタイプの証明書妥当性検査ポリシーを使用するかを指定します。

このフィールドは、以下のいずれかの値に設定できます。

MQ_CERT_VAL_POLICY_ANY

セキュア・ソケット・ライブラリーでサポートされる各証明書妥当性検査ポリシーを適用します。ポリシーのうちのいずれかにおいて証明書チェーンが有効と見なされる場合、その証明書チェーンを受け入れます。

MQ_CERT_VAL_POLICY_RFC5280

RFC5280 準拠の証明書妥当性検査ポリシーのみ適用します。この設定は、ANY 設定よりも厳密に妥当性検査しますが、一部の旧式のデジタル証明書を拒否します。

MQ_CERT_VAL_POLICY_NONE

証明書検証ポリシーを適用しない。この設定はクライアントアプリケーション専用であり、TLS サーバー証明書を信頼チェーンの検証なしに受け入れます。

このフィールドの初期値は MQ_CERT_VAL_POLICY_ANY です。

CertificateLabel (MQCHAR64)

このフィールドは、使用される証明書ラベルの詳細を示します。

IBM MQ は、CertificateLabel フィールドのデフォルト値をブランクとして初期化します。

これは実行時にデフォルト値とて解釈され、後方互換です。

例えば 5.0 未満の MQSCO バージョンを指定するか、ブランクのデフォルト値を CertificateLabel フィールドに使用した場合、既存のデフォルト値 `ibmwebspheremquser_id` が使用されます。

KeyRepoPasswordPtr (MQPTR)

これは TLS キーリポジトリのパスフレーズのアドレスです。

これは入力フィールドです。このフィールドの初期値は、ポインターをサポートするプログラミング言語のヌル・ポインターです。それ以外の場合は、すべてヌルのバイトのストリングです。このフィールドは、MQSCO_VERSION_6Version 未満の場合に無視されます。

このフィールドは、AIX, Linux, and WindowsIBM i システム上で実行 IBM MQ MQI clients する場合にのみ関連します。

キーリポジトリのパスフレーズは、プレーンテキストの文字列として指定することも、`gpassphrase` **runmqicred** ユーティリティを使用して暗号化されたパスフレーズとして指定することもできます。

暗号化されたパスフレーズを提供する場合は、同じクライアントアプリケーションが提供する MQCSP 構造体内で、そのパスフレーズの暗号化に使用された初期鍵を指定してください。

このフィールドで指定されたキーリポジトリのパスフレーズは、MQKEYRPWD 環境変数またはクライアント設定ファイルの SSL スタンス内のプロパティ *SSLKeyRepositoryPassword* で指定されたキーリポジトリのパスフレーズを上書きします。

キーリポジトリのパスフレーズを指定するには、`KeyRepoPasswordPtr -p` または `-r KeyRepoPasswordOffset` のいずれかを使用できますが、両方同時に使用することはできません。

KeyRepoPasswordOffset (MQLONG)

これは、TLS キーリポジトリのパスフレーズが MQSCO 構造体の先頭から始まる位置のオフセット（バイト単位）です。オフセットの値は、正負どちらの値にもなります。

キーリポジトリのパスフレーズを指定するには、`KeyRepoPasswordPtr -p` または `-r KeyRepoPasswordOffset` のいずれかを使用できますが、両方同時に使用することはできません。詳細については、フィールド *KeyRepoPasswordPtr* の説明を参照してください。

これは入力フィールドです。このフィールドの初期値は 0 です。このフィールドは、MQSCO_VERSION_6Version 未満の場合に無視されます。

KeyRepoPasswordLength (MQLONG)

これは TLS キーリポジトリのパスフレーズの長さです。

キーリポジトリのパスフレーズの最大長は 128 文字です IBM i。キーリポジトリのパスフレーズが許可される最大長を超える場合、接続は MQRC_KEY_REPOSITORY_ERROR で失敗します。

これは入力フィールドです。このフィールドの初期値は 0 です。このフィールドは、MQSCO_VERSION_6Version 未満の場合に無視されます。

HTTPSKeyStorePtr

V 10.0.0

これは、HTTPSKeyStore へのパスを含む文字列のアドレスです。HTTPSKeyStore は、HTTPS サーバーによって提示された証明書に対する信頼を確立する際に、キューマネージャが使用する PKCS #12 キーリポジトリです。

これは入力フィールドです。このフィールドの初期値は、ポインターをサポートするプログラミング言語のヌル・ポインターです。それ以外の場合は、すべてヌルのバイトのストリングです。このフィールドは、MQSCO_VERSION_7Version 未満の場合に無視されます。

LinuxAIX このフィールドは、および Windows のオペレーティングシステム上で実行されている MQI クライアントに IBM MQ のみ適用されます。提供されるパスの最大長は、クライアントオペレーティングシステムのものです。

キーリポジトリのパスフレーズを指定するには、`HTTPSKeyStorePtr -p` または `-r HTTPSKeyStoreOffset` のいずれかを使用できますが、両方同時に使用することはできません。いずれのフィールドも指定しない場合、このクライアントからセキュアな HTTP を必要とする機能は無効化されます。両方のフィールドを指定した場合、呼び出しは理由コード MQRC_HTTPS_KEYSTORE_ERROR で失敗します。

HTTPSKeyStoreOffset

V 10.0.0

これは、HTTPSKeyStore パスの MQSCO 構造体の先頭からのオフセット（バイト単位）です。オフセットの値は、正負どちらの値にもなります。

HTTPSKeyStore のパスを指定するには *HTTPSKeyStorePtr*、`-P` または *-D HTTPSKeyStoreOffset* のいずれか一方を使用できますが、両方同時に使用することはできません。いずれのフィールドも指定しない場合、このクライアントからセキュアな HTTP を必要とする機能は無効化されます。両方のフィールドを指定した場合、呼び出しは理由コード MQRC_HTTPS_KEYSTORE_ERROR で失敗します。

これは入力フィールドです。このフィールドの初期値は 0 です。このフィールドは、MQSCO_VERSION_7Version 未満の場合に無視されます。

HTTPSKeyStoreLength

V 10.0.0

これは HTTPSKeyStore パスの長さです。

これは入力フィールドです。このフィールドの初期値は 0 です。このフィールドは、MQSCO_VERSION_7Version 未満の場合に無視されます。

HTTPSCertValidation

V 10.0.0

HTTPS 接続に必要な証明書検証のレベルを指定します。このフィールドには以下のいずれかの値を設定できます：

MQ_HTTPSCERTVAL_DEFAULT

検証ポリシーは、指定されている場合、ファイル *mqclient.ini* 内の HTTPSCertValidation 属性から取得されます。そのような設定が存在しない場合、動作は MQ_HTTPSCERTVAL_ANY と同じです。

MQ_HTTPSCERTVAL_ANY

ピアの証明書が有効であり、信頼された認証機関によって署名されていることのみを確認する。

MQ_HTTPSCERTVAL_HOSTNAMECN

ピアの証明書が有効であり、信頼された認証機関によって署名されていること、および証明書の識別名が接続先の HTTP サーバーのものと一致していることを確認してください。

MQ_HTTPSCERTVAL_NONE

ピアの証明書を確認しないでください。セッションは TLS で暗号化されていますが、クライアントが意図したホストと通信している保証はありません。このオプションは慎重に使用してください。このオプションを使用すると、例えば、クライアントを悪意のある CCDT ファイルにリダイレクトする中間者攻撃を可能にする可能性があります。

このフィールドは、MQSCO_VERSION_7Version 未満の場合に無視されます。

HTTPSCertRevocation

V 10.0.0

HTTPS 接続に必要な証明書失効チェックのレベルを指定します。このフィールドには以下のいずれかの値を設定できます：

MQ_HTTPSCERTREV_DEFAULT

検証ポリシーは、ファイル *mqclient.ini* 内の HTTPSCertRevocation 属性によって決定されます（指定されている場合）。そのような設定が存在しない場合、動作は MQ_HTTPSCERTREV_REQUIRED と同様です。

MQ_HTTPSCERTREV_REQUIRED

証明書失効サーバー情報が提供されている場合は、その情報を確認し、証明書が失効していないことを検証してください。このチェックが失敗した場合（例：失効サービスが利用できない場合）、HTTPS 接続は失敗します。

MQ_HTTPSCERTREV_OPTIONAL

証明書失効チェックを実行するが、失効サービスに到達できない場合、接続を許可する。

MQ_HTTPSCERTREV_DISABLED

証明書失効チェックを実行しないでください。

このフィールドは、MQSCO_VERSION_7Version 未満の場合に無視されます。

MQSD - サブスクリプション記述子

MQSD 構造体を使用して、作成されるサブスクリプションに関する詳細情報を指定します。この構造体は、MQSUB 呼び出しの入出力パラメーターです。詳細については、[MQSUB の使用上の注意](#)を参照してください。

可用性

MQSD 構造体は以下のプラットフォームで利用可能です：

-  AIX
-  IBM i
-  Linux
-  Windows
-  z/OS

および、これらのシステムに接続された IBM MQ MQI clients。

バージョン

MQSD の現在のバージョンは MQSD_VERSION_1 です。

文字セットとエンコード

MQSD 内のデータは、キューマネージャ **CodedCharSetId** 属性によって指定される文字セットおよびローカルキューマネージャのエンコーディング (MQENC_NATIVE で指定) でなければならない。ただし、アプリケーションが MQ MQI クライアントとして実行されている場合、構造体はクライアントの文字セットとエンコードに従っている必要があります。

管理対象サブスクリプション

アプリケーションに、そのサブスクリプションと一致するパブリケーションの宛先として特定のキューを使用する固有の必要がない場合は、管理対象サブスクリプション機能を利用できます。アプリケーションが管理対象サブスクリプションの使用を選ぶと、キュー・マネージャーは、MQSUB 呼び出しからの出力としてオブジェクト・ハンドルを提供して、パブリッシュされるメッセージが送信される宛先をサブスクライバーに通知します。詳細については、[Hobj \(MQHOBJ\) - 入出力](#)を参照してください。

またキュー・マネージャーは、以下の状態で、サブスクリプションの除去時に管理対象宛先から未取り出しのメッセージのクリーンアップに着手します。

- サブスクリプションの除去時 - MQCLOSE と MQCO_REMOVE_SUB の併用による - 管理対象 Hobj のクローズも行われます。
- 非永続サブスクリプション (MQSO_NON_DURABLE) を使用するアプリケーションに対する接続が失われる際に、暗黙的手段で
- サブスクリプションの除去時に満了していることにより - サブスクリプションが満了していて、管理対象 Hobj がクローズされるため。

管理対象サブスクリプションと非永続サブスクリプションを併用して、このクリーンアップを行えるようにすることにより、クローズされた非永続サブスクリプションに関するメッセージによりキュー・マネー

ジャー中のスペースが塞がれないようにする必要があります。永続サブスクリプションも管理対象宛先を使用できます。

フィールド

注：次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

フィールド名と説明	定数の名前	定数の初期値（存在する場合）
StrucId (構造識別子)	MQSD_STRUC_ID	'SD--'
バージョン (構造バージョン番号)	MQSD_VERSION_1	1
オプション (オプション)	MQSO_NON_DURABLE	0
ObjectName (オブジェクト名)	なし	ヌル・ストリングまたはブランク
AlternateUserId (代替ユーザー ID)	なし	ヌル・ストリングまたはブランク
AlternateSecurityId (代替セキュリティ ID)	MQSID_NONE	Null
SubExpiry (購読期限切れ)	MQEI_UNLIMITED	-1
ObjectString (オブジェクト文字列)	なし	MQCHARV 用に定義される名前と値
SubName (サブスクリプション名)	なし	MQCHARV 用に定義される名前と値
SubUserData (サブスクリプションユーザーデータ)	なし	MQCHARV 用に定義される名前と値
SubCorrelId (サブスクリプション相関 ID)	MQCI_NONE	Null
PubPriority (出版優先権)	MQPRI_PRIORITY_AS_Q_DEF	-3
PubAccountingToken (出版会計トークン)	MQACT_NONE	Null
PubAppIdentityData (出版物申請識別データ)	なし	ヌル・ストリングまたはブランク
SelectionString (選択基準を提供する文字列)	なし	MQCHARV 用に定義される名前と値
SubLevel (サブスクリプションレベル)	なし	1
ResObjectString (長いオブジェクト名)	なし	MQCHARV 用に定義される名前と値

注：

1. 記号-は、単一のブランク文字を表します。
2. ヌル・ストリングまたはブランクの値は、C 言語ではヌル・ストリングを表し、他のプログラミング言語ではブランク文字を表します。
3. C プログラミング言語では、マクロ変数 MQSD_DEFAULT には、表にリストされている値が含まれています。この変数を以下の方法で使用すると、構造体のフィールドに初期値を設定できます。

```
MQSD MySD = {MQSD_DEFAULT};
```

言語ごとの宣言

MQSD の C 宣言

```
typedef struct tagMQSD MQSD;
struct tagMQSD {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQLONG     Options;          /* Options associated with subscribing */
    MQCHAR48    ObjectName;      /* Object name */
    MQCHAR12    AlternateUserId; /* Alternate user identifier */
    MQBYTE40    AlternateSecurityId; /* Alternate security identifier */
    MQLONG     SubExpiry;        /* Expiry of Subscription */
    MQCHARV     ObjectString;    /* Object Long name */
    MQCHARV     SubName;         /* Subscription name */
    MQCHARV     SubUserData;     /* Subscription User data */
    MQBYTE24    SubCorrelId;     /* Correlation Id related to this subscription */
    MQLONG     PubPriority;      /* Priority set in publications */
    MQBYTE32    PubAccountingToken; /* Accounting Token set in publications */
    MQCHAR32    PubApplIdentityData; /* Appl Identity Data set in publications */
    MQCHARV     SelectionString; /* Message selector structure */
    MQLONG     SubLevel;        /* Subscription level */
    MQCHARV     ResObjectString; /* Resolved Long object name */
    /* Ver:1 */
};
```

MQSD の COBOL 宣言

```
** Address of variable length string
20  MQSD-OBJECTSTRING-VSPTR          POINTER.
** Offset of variable length string
20  MQSD-OBJECTSTRING-VSOFFSET       PIC S9(9) BINARY.
** size of buffer
20  MQSD-OBJECTSTRING-VSBUFSIZE      PIC S9(9) BINARY.
** Length of variable length string
20  MQSD-OBJECTSTRING-VSLENGTH       PIC S9(9) BINARY.
** CCSID of variable length string
20  MQSD-OBJECTSTRING-VSCCSID        PIC S9(9) BINARY.
** Subscription name
15  MQSD-SUBNAME.
** Address of variable length string
20  MQSD-SUBNAME-VSPTR              POINTER.
** Offset of variable length string
20  MQSD-SUBNAME-VSOFFSET           PIC S9(9) BINARY.
** size of buffer
20  MQSD-SUBNAME-VSBUFSIZE          PIC S9(9) BINARY.
** Length of variable length string
20  MQSD-SUBNAME-VSLENGTH           PIC S9(9) BINARY.
** CCSID of variable length string
20  MQSD-SUBNAME-VSCCSID            PIC S9(9) BINARY.
** Subscription User data
15  MQSD-SUBUSERDATA.
** Address of variable length string
20  MQSD-SUBUSERDATA-VSPTR          POINTER.
** Offset of variable length string
20  MQSD-SUBUSERDATA-VSOFFSET       PIC S9(9) BINARY.
** size of buffer
20  MQSD-SUBUSERDATA-VSBUFSIZE      PIC S9(9) BINARY.
** Length of variable length string
20  MQSD-SUBUSERDATA-VSLENGTH       PIC S9(9) BINARY.
** CCSID of variable length string
20  MQSD-SUBUSERDATA-VSCCSID        PIC S9(9) BINARY.
** Correlation Id related to this subscription
15  MQSD-SUBCORRELID                PIC X(24).
** Priority set in publications
15  MQSD-PUBPRIORITY                PIC S9(9) BINARY.
** Accounting Token set in publications
15  MQSD-PUBACCOUNTINGTOKEN          PIC X(32).
** Appl Identity Data set in publications
15  MQSD-PUBAPPLIDENTITYDATA         PIC X(32).
** Message Selector
15  MQSD-SELECTIONSTRING.
** Address of variable length string
20  MQSD-SELECTIONSTRING-VSPTR      POINTER.
** Offset of variable length string
20  MQSD-SELECTIONSTRING-VSOFFSET   PIC S9(9) BINARY.
** size of buffer
20  MQSD-SELECTIONSTRING-VSBUFSIZE  PIC S9(9) BINARY.
** Length of variable length string
```

```

20 MQSD-SELECTIONSTRING-VSLENGTH      PIC S9(9) BINARY.
** CCSID of variable length string
20 MQSD-SELECTIONSTRING-VSCCSID       PIC S9(9) BINARY.
** Selection criteria
20 MQSD-SELECTIONSTRING-SUBLEVEL      PIC S9(9) BINARY.
** Long object name
20 MQSD-SELECTIONSTRING-RESOBJSTRING  PIC S9(9) BINARY.

```

MQSD の PL/I 宣言

```

dcl
1 MQSD based,
3 StrucId      char(4),      /* Structure identifier */
3 Version      fixed bin(31), /* Structure version number */
3 Options      fixed bin(31), /* Options associated with subscribing */
3 ObjectName   char(48),     /* Object name */
3 AlternateUserId char(12),   /* Alternate user identifier */
3 AlternateSecurityId char(40), /* Alternate security identifier */
3 SubExpiry    fixed bin(31), /* Expiry of Subscription */
3 ObjectString, /* Object Long name */
5 VSPtr        pointer,      /* Address of variable length string */
5 VSOFFSET      fixed bin(31), /* Offset of variable length string */
5 VSBUFSize     fixed bin(31), /* size of buffer */
5 VSLength      fixed bin(31), /* Length of variable length string */
5 VSCCSID       fixed bin(31); /* CCSID of variable length string */
3 SubName,      /* Subscription name */
5 VSPtr        pointer,      /* Address of variable length string */
5 VSOFFSET      fixed bin(31), /* Offset of variable length string */
5 VSBUFSize     fixed bin(31), /* size of buffer */
5 VSLength      fixed bin(31), /* Length of variable length string */
5 VSCCSID       fixed bin(31); /* CCSID of variable length string */
3 SubUserData, /* Subscription User data */
5 VSPtr        pointer,      /* Address of variable length string */
5 VSOFFSET      fixed bin(31), /* Offset of variable length string */
5 VSBUFSize     fixed bin(31), /* size of buffer */
5 VSLength      fixed bin(31), /* Length of variable length string */
5 VSCCSID       fixed bin(31); /* CCSID of variable length string */
3 SubCorrelId   char(24),     /* Correlation Id related to this subscription */
3 PubPriority    fixed bin(31), /* Priority set in publications */
3 PubAccountingToken char(32), /* Accounting Token set in publications */
3 PubApplIdentityData char(32), /* Appl Identity Data set in publications */
3 SelectionString, /* Message Selection */
5 VSPtr        pointer,      /* Address of variable length string */
5 VSOFFSET      fixed bin(31), /* Offset of variable length string */
5 VSBUFSize     fixed bin(31), /* size of buffer */
5 VSLength      fixed bin(31), /* Length of variable length string */
5 VSCCSID       fixed bin(31); /* CCSID of variable length string */
3 SubLevel      fixed bin(31), /* Subscription level */
3 ResObjectString, /* Resolved Long object name */
5 VSPtr        pointer,      /* Address of variable length string */
5 VSOFFSET      fixed bin(31), /* Offset of variable length string */
5 VSBUFSize     fixed bin(31), /* size of buffer */
5 VSLength      fixed bin(31), /* Length of variable length string */
5 VSCCSID       fixed bin(31); /* CCSID of variable length string */

```

MQSD の高水準アセンブラ宣言

```

MQSD          DSECT
MQSD_STRUCID  DS CL4 Structure identifier
MQSD_VERSION  DS F Structure version number
MQSD-OPTIONS  DS F Options associated with subscribing
MQSD_OBJECTNAME DS CL48 Object name
MQSD_ALTERNATEUSERID DS CL12 Alternate user identifier
MQSD_ALTERNATESECURITYID DS CL40 Alternate security identifier
MQSD_SUBEXPIRY DS F Expiry of Subscription
MQSD_OBJECTSTRING DS 0F Object Long name
MQSD_OBJECTSTRING_VSPTR DS F Address of variable length string
MQSD_OBJECTSTRING_VSOFFSET DS F Offset of variable length string
MQSD_OBJECTSTRING_VSBUFSize DS F size of buffer
MQSD_OBJECTSTRING_VSLENGTH DS F Length of variable length string
MQSD_OBJECTSTRING_VSCCSID DS F CCSID of variable length string
MQSD_OBJECTSTRING_LENGTH EQU *-MQSD_OBJECTSTRING
ORG MQSD_OBJECTSTRING
MQSD_OBJECTSTRING_AREA DS CL(MQSD_OBJECTSTRING_LENGTH)
*
MQSD_SUBNAME DS 0F Subscription name
MQSD_SUBNAME_VSPTR DS F Address of variable length string
MQSD_SUBNAME_VSOFFSET DS F Offset of variable length string
MQSD_SUBNAME_VSBUFSize DS F size of buffer

```

```

MQSD_SUBNAME_VSLENGTH      DS  F   Length of variable length string
MQSD_SUBNAME_VSCCSID       DS  F   CCSID of variable length string
MQSD_SUBNAME_LENGTH        EQU  *-MQSD_SUBNAME
ORG MQSD_SUBNAME
MQSD_SUBNAME_AREA          DS  CL(MQSD_SUBNAME_LENGTH)
*
MQSD_SUBUSERDATA           DS  0F   Subscription User data
MQSD_SUBUSERDATA_VSPTR     DS  F   Address of variable length string
MQSD_SUBUSERDATA_VSOFFSET  DS  F   Offset of variable length string
MQSD_SUBUSERDATA_VSBUFSIZE DS  F   size of buffer
MQSD_SUBUSERDATA_VSLENGTH  DS  F   Length of variable length string
MQSD_SUBUSERDATA_VSCCSID   DS  F   CCSID of variable length string
MQSD_SUBUSERDATA_LENGTH    EQU  *-MQSD_SUBUSERDATA
ORG MQSD_SUBUSERDATA
MQSD_SUBUSERDATA_AREA      DS  CL(MQSD_SUBUSERDATA_LENGTH)
*
MQSD_SUBCORRELID           DS  CL24 Correlation Id related to this subscription
MQSD_PUBPRIORITY           DS  F   Priority set in publications
MQSD_PUBACCOUNTINGTOKEN    DS  CL32 Accounting Token set in publications
MQSD_PUBAPPLIDENTITYDATA   DS  CL32 Appl Identity Data set in publications
*
MQSD_SELECTIONSTRING       DS  F   Message Selector
MQSD_SELECTIONSTRING_VSPTR DS  F   Address of variable length string
MQSD_SELECTIONSTRING_VSOFFSET DS  F   Offset of variable length string
MQSD_SELECTIONSTRING_VSBUFSIZE DS  F   size of buffer
MQSD_SELECTIONSTRING_VSLENGTH DS  F   Length of variable length string
MQSD_SELECTIONSTRING_VSCCSID DS  F   CCSID of variable length string
MQSD_SELECTIONSTRING_LENGTH EQU  *-MQSD_SELECTIONSTRING
ORG MQSD_SELECTIONSTRING
MQSD_SELECTIONSTRING_AREA  DS  CL(MQSD_SELECTIONSTRING_LENGTH)
*
MQSD-SUBLEVEL              DS  F   Subscription level
*
MQSD_RESOBJECTSTRING        DS  F   Resolved Long object name
MQSD_RESOBJECTSTRING_VSPTR  DS  F   Address of variable length string
MQSD_RESOBJECTSTRING_VSOFFSET DS  F   Offset of variable length string
MQSD_RESOBJECTSTRING_VSBUFSIZE DS  F   size of buffer
MQSD_RESOBJECTSTRING_VSLENGTH DS  F   Length of variable length string
MQSD_RESOBJECTSTRING_VSCCSID DS  F   CCSID of variable length string
MQSD_RESOBJECTSTRING_LENGTH EQU  *-MQSD_RESOBJECTSTRING
ORG MQSD_RESOBJECTSTRING
MQSD_RESOBJECTSTRING_AREA  DS  CL(MQSD_RESOBJECTSTRING_LENGTH)
*
MQSD_LENGTH                EQU  *-MQSD
ORG MQSD
MQSD_AREA                  DS  CL(MQSD_LENGTH)

```

MQSD のフィールド詳細

MQSD サブスクリプション記述子構造体は、作成されるサブスクリプションの詳細を指定するために使用されます。この構造体は、MQSUB 呼び出しの入出力パラメーターです。

フィールド

注: 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

フィールド名と説明	定数の名前	定数の初期値（存在する場合）
StrucId (構造識別子)	MQSD_STRUC_ID	'SD-1'
バージョン (構造バージョン番号)	MQSD_VERSION_1	1
オプション (オプション)	MQSO_NON_DURABLE	0
ObjectName (オブジェクト名)	なし	ヌル・ストリングまたはブランク
AlternateUserId (代替ユーザー ID)	なし	ヌル・ストリングまたはブランク
AlternateSecurityId (代替セキュリティ ID)	MQSID_NONE	Null
SubExpiry (購読期限切れ)	MQEI_UNLIMITED	-1

フィールド名と説明	定数の名前	定数の初期値（存在する場合）
<u>ObjectString</u> (オブジェクト文字列)	なし	MQCHARV 用に定義される名前と値
<u>SubName</u> (サブスクリプション名)	なし	MQCHARV 用に定義される名前と値
<u>SubUserData</u> (サブスクリプションユーザーデータ)	なし	MQCHARV 用に定義される名前と値
<u>SubCorrelId</u> (サブスクリプション相関 ID)	MQCI_NONE	Null
<u>PubPriority</u> (出版優先権)	MQPRI_PRIORITY_AS_Q_DEF	-3
<u>PubAccountingToken</u> (出版会計トークン)	MQACT_NONE	Null
<u>PubAppIdentityData</u> (出版物申請識別データ)	なし	ヌル・ストリングまたはブランク
<u>SelectionString</u> (選択基準を提供する文字列)	なし	MQCHARV 用に定義される名前と値
<u>SubLevel</u> (サブスクリプションレベル)	なし	1
<u>ResObjectString</u> (長いオブジェクト名)	なし	MQCHARV 用に定義される名前と値
注： 1. 記号-は、単一のブランク文字を表します。 2. ヌル・ストリングまたはブランクの値は、C 言語ではヌル・ストリングを表し、他のプログラミング言語ではブランク文字を表します。 3. C プログラミング言語では、マクロ変数 MQSD_DEFAULT には、表にリストされている値が含まれています。この変数を以下の方法で使用すると、構造体のフィールドに初期値を設定できます。 <pre>MQSD MySD = {MQSD_DEFAULT};</pre>		

StrucId (MQCHAR4)

これはサブスクリプション記述子構造体の構造体識別子です。常に入力フィールドです。その値は MQSD_STRUC_ID です。

値は次のものでなければなりません。

MQSD_STRUC_ID

サブスクリプション記述子構造体の識別子。

C プログラミング言語では、定数 MQSD_STRUC_ID_ARRAY も定義されている。これは MQSD_STRUC_ID と同じ値を持ちますが、文字列ではなく文字の配列です。

Version (MQLONG)

これは構造体のバージョン番号です。値は以下のものでなければなりません。

MQSD_VERSION_1

バージョン 1 のサブスクリプション記述子の構造体。

以下の定数は、現行バージョンのバージョン番号を指定しています。

MQSD_CURRENT_VERSION

サブスクリプション記述子の構造体の現行バージョン。

これは常に入力フィールドです。フィールドの初期値は、MQSD_VERSION_1 です。

Options (MQLONG)

これは、MQSUB 呼び出しのアクションを制御するためのオプションを提供します。

以下のオプションのうち少なくとも 1 つを指定する必要があります。

- MQSO_ALTER
- MQSO_RESUME
- MQSO_CREATE

複数のオプションを指定するには、値を加算する (同じ定数を複数回加算しない) か、ビット単位 OR 演算を使用して値を結合 (プログラミング言語でビット演算がサポートされる場合) します。

無効な組み合わせについては、それぞれこのトピックで説明を行います。それ以外の他の組み合わせは有効です。

アクセスまたは作成オプション: アクセスおよび作成オプションは、サブスクリプションを作成するか、または既存のサブスクリプションを返すか、または変更するかを制御します。これらのオプションのうち少なくとも 1 つを指定する必要があります。

表 555. アクセスおよび作成オプションの有効な組み合わせ	
オプションの組み合わせ	注
MQSO_CREATE	サブスクリプションがない場合に作成します。サブスクリプションが既存の場合は、この組み合わせは失敗します。
MQSO_RESUME	既存のサブスクリプションを再開します。サブスクリプションがない場合は、この組み合わせは失敗します。
MQSO_CREATE + MQSO_RESUME	サブスクリプションがない場合は作成し、ある場合は一致するものを再開します。この組み合わせは、複数回実行されるアプリケーションで使用する場合に便利です。
MQSO_ALTER (注釈を参照)	フィールドを MQSD 中の指定内容と一致するよう変更して、既存のサブスクリプションを再開します。サブスクリプションがない場合は、この組み合わせは失敗します。
MQSO_CREATE + MQSO_ALTER (注釈を参照)	サブスクリプションがない場合は作成し、ある場合はフィールドを MQSD 中の指定内容と一致するよう変更して、一致するものを再開します。この組み合わせは、サブスクリプションが特定の状態にあることを確認してから続行する必要があるアプリケーションで使用する場合に便利です。

注記:

MQSO_ALTER を指定するオプションで MQSO_RESUME も指定できますが、この組み合わせは MQSO_ALTER を単独で指定する場合と比べて追加の効果はありません。MQSO_ALTER は MQSO_RESUME を暗黙指定します。MQSUB を呼び出してサブスクリプションを変更することは、そのサブスクリプションを再開することも意味しているからです。しかし、その逆は真ではありません。サブスクリプションを再開することは変更を暗黙に示しません。

MQSO_CREATE

指定されたトピックに関する新しいサブスクリプションを作成します。同じ SubName を使用するサブスクリプションが既存の場合、呼び出しは MQRC_SUB_ALREADY_EXISTS で失敗します。

MQSO_CREATE オプションを MQSO_RESUME と結合すると、この失敗を避けることができます。
SubName は常に必要ではありません。詳しくは、このフィールドの説明を参照してください。

MQSO_CREATE と MQSO_RESUME を結合すると、指定された *SubName* の既存のサブスクリプションがあれば、そのサブスクリプションに対するハンドルを戻します。既存のサブスクリプションがない場合は、MQSD で提供されているすべてのフィールドを使用して新しく作成します。

MQSO_CREATE を MQSO_ALTER と結合しても同様の効果が得られます。

MQSO_RESUME

SubName の指定内容と一致する既存のサブスクリプションに対するハンドルを戻します。一致するサブスクリプションの属性に対する変更は加えられず、出力の MQSD 構造体中に返されます。MQSD フィールド *StrucId*、*Version*、*Options*、*AlternateUserId*、*AlternateSecurityId*、および *SubName* のみ使用されます。

フルネームが一致するサブスクリプションが存在しない場合は、呼び出しは理由コード MQRC_NO_SUBSCRIPTION で失敗します。MQSO_CREATE オプションを MQSO_RESUME と結合すると、この失敗を避けることができます。

サブスクリプションのユーザー ID は、サブスクリプションを作成したユーザー ID か、またはその後別のユーザー ID によって変更が加えられている場合は、最近正常に変更を加えたユーザー ID です。
AlternateUserId を使用し、そのユーザーに代替ユーザー ID の使用が許可されている場合は、その下でサブスクリプションを作成したユーザー ID の代わりに、代替ユーザー ID がサブスクリプションを作成したユーザー ID として記録されます。

MQSO_ANY_USERID オプションを指定せずに作成した一致サブスクリプションが存在する場合に、サブスクリプションのユーザー ID がサブスクリプションに対するハンドルを要求するアプリケーションのユーザー ID と違う場合は、呼び出しは理由コード MQRC_IDENTITY_MISMATCH で失敗します。

一致するサブスクリプションが存在し、現在使用中の場合は、呼び出しは MQRC_SUBSCRIPTION_IN_USE で失敗します。

SubName で名前指定されたサブスクリプションが、アプリケーションからの再開や変更が無効なサブスクリプションである場合は、呼び出しは MQRC_INVALID_SUBSCRIPTION で失敗します。

MQSO_RESUME は MQSO_ALTER によって暗黙指定されるので、これらのオプションを結合する必要はありません。しかし、2つのオプションを結合してもエラーにはなりません。

MQSO_ALTER

SubName の指名内容と一致する既存のサブスクリプションに対するハンドルとサブスクリプションのフルネームを戻します。MQSD 中の指定内容と違うサブスクリプションの属性は、その属性に関する変更が禁止されているのでなければ、サブスクリプション中で変更されます。詳細は、各属性の説明に注記されており、要約は下の表にあります。変更できない属性に変更を加えようとしたら、MQSO_IMMUTABLE オプションを設定されたサブスクリプションに変更を加えようとしたらすると、呼び出しは以下の表に示されている理由コードで失敗します。

フルネームが一致するサブスクリプションが存在しない場合は、呼び出しは理由コード MQRC_NO_SUBSCRIPTION で失敗します。MQSO_CREATE オプションを MQSO_ALTER と結合すると、この失敗を避けることができます。

MQSO_CREATE と MQSO_ALTER を結合すると、指定された *SubName* の既存のサブスクリプションがあれば、そのサブスクリプションに対するハンドルを戻します。既存のサブスクリプションがない場合は、MQSD で提供されているすべてのフィールドを使用して新しく作成します。

サブスクリプションのユーザー ID とは、サブスクリプションを作成したユーザー ID です。または、この ID が後で別のユーザー ID に変更された場合、もっとも最近、変更成功したユーザー ID となります。
AlternateUserId を使用する場合で、そのユーザーで変更ユーザー ID の使用が許可されている場合、サブスクリプションを作成したユーザー ID の代わりに 変更ユーザー ID が、サブスクリプションを作成したユーザー ID として記録されます。

オプション MQSO_ANY_USERID を指定せずに作成した一致サブスクリプションが存在する場合に、サブスクリプションのユーザー ID がサブスクリプションに対するハンドルを要求するアプリケーションのユーザー ID と違う場合は、呼び出しは理由コード MQRC_IDENTITY_MISMATCH で失敗します。

一致するサブスクリプションが存在し、現在使用中の場合は、呼び出しは MQRC_SUBSCRIPTION_IN_USE で失敗します。

SubName で名前指定されたサブスクリプションが、アプリケーションからの再開や変更が無効なサブスクリプションである場合は、呼び出しは MQRC_INVALID_SUBSCRIPTION で失敗します。

以下の表は、MQSD および MQSUB 中の属性値を変更する MQSO ALTER の機能を示しています。

表 556. 変更可能な MQSD および MQSUB 中の属性			
データ・タイプ記述子または関数呼び出し	フィールド名	MQSO ALTER を使用してこの属性を変更できるか	理由コード
MQSD	耐久性オプション	いいえ	MQRC_DURABILITY_NOT_ALTERABLE
MQSD	宛先オプション	はい	なし
MQSD	登録オプション	可 (注釈 559 ページの『1』を参照)	MQSO_GROUP_SUB を変更しようとしている場合は MQRC_GROUPING_NOT_ALTERABLE
MQSD	パブリケーション・オプション	可 (注釈 559 ページの『2』を参照)	なし
MQSD	ワイルドカード・オプション	いいえ	MQRC_TOPIC_NOT_ALTERABLE
MQSD	その他のオプション	不可 (注釈 559 ページの『3』を参照)	なし
MQSD	ObjectName	いいえ	MQRC_TOPIC_NOT_ALTERABLE
MQSD	AlternateUserId	不可 (注釈 559 ページの『4』を参照)	なし
MQSD	AlternateSecurityId	不可 (注釈 559 ページの『4』を参照)	なし
MQSD	SubExpiry	はい	なし
MQSD	ObjectString	いいえ	MQRC_TOPIC_NOT_ALTERABLE
MQSD	SubName	不可 (注釈 559 ページの『5』を参照)	なし
MQSD	SubUserData	はい	なし
MQSD	SubCorrelId	可 (注釈 559 ページの『6』を参照)	グループ化されたサブスクリプション中の場合は MQRC_GROUPING_NOT_ALTERABLE
MQSD	PubPriority	はい	なし
MQSD	PubAccountingToken	はい	なし
MQSD	PubApplIdentityData	はい	なし
MQSD	SubLevel	いいえ	MQRC_SUBLEVEL_NOT_ALTERABLE
MQSUB	Hobj	可 (注釈 559 ページの『6』を参照)	グループ化されたサブスクリプション中の場合は MQRC_GROUPING_NOT_ALTERABLE

注：

1. MQSO_GROUP_SUB を変更できません。
2. MQSO_NEW_PUBLICATIONS_ONLY はサブスクリプションの一部ではないので変更できません。
3. これらのオプションはサブスクリプションの一部ではありません。
4. この属性はサブスクリプションの一部ではありません。
5. この属性は、変更されるサブスクリプションの ID です。
6. グループ化されたサブスクリプション (MQSO_GROUP_SUB) の一部の場合を除いて変更可能です。

耐久性オプション

以下のオプションは、サブスクリプションの耐久性を制御します。これらのオプションのうち 1 つのみ指定できます。MQSO ALTER オプションを使用して既存のサブスクリプションを変更しようとしている場合は、サブスクリプションの耐久性を変更できません。MQSO_RESUME を使用する MQSUB 呼び出しからの戻り時に、適切な耐久性オプションが設定されます。

MQSO_DURABLE

MQCO_REMOVE_SUB オプションを指定した MQCLOSE を使用して明示的に除去されるまでは、このトピックへのサブスクリプションが残されることを要求します。このサブスクリプションが明示的に除去されない場合、このアプリケーションのキュー・マネージャーに対する接続がクローズされた後でも残されます。

永続サブスクリプションを許可しないと定義されているトピックに対して永続サブスクリプションが要求されると、呼び出しは MQRC_DURABILITY_NOT_ALLOWED で失敗します。

MQSO_NON_DURABLE

このトピックに対するサブスクリプションが明示的に除去されていない場合は、アプリケーションのキュー・マネージャーに対する接続がクローズされる際に除去されるよう要求します。

MQSO_NON_DURABLE は、MQSO_DURABLE オプションの反対で、プログラムの文書化を支援するために定義します。いずれも指定されていないときは、これがデフォルト値になります。

目的地オプション

以下のオプションは、購読されているトピックのパブリケーションが送信される宛先を制御します。MQSO_ALTER オプションを使用して既存のサブスクリプションを変更する場合は、サブスクリプションのパブリケーションに使用される宛先を変更できます。MQSO_RESUME を使用する MQSUB 呼び出しから戻る際に、該当する場合はこのオプションが設定されます。

MQSO_MANAGED

パブリケーションが送信される宛先がキュー・マネージャーによって管理されるように要求します。

Hobj 中に戻されるオブジェクト・ハンドルは、キュー・マネージャー管理キューを表し、以後 MQGET、MQCB、MQINQ、または MQCLOSE 呼び出しと併用されます。

MQSO_MANAGED が指定されていない場合は、前の MQSUB 呼び出しから戻されたオブジェクト・ハンドルを Hobj パラメーター中に提供できません。

MQSO_NO_MULTICAST

パブリケーションが送信される宛先がマルチキャスト・グループ・アドレスでないことを要求します。このオプションは、MQSO_MANAGED オプションと結合する場合のみ有効です。キューに対するハンドルが Hobj パラメーター内で提供される場合、このサブスクリプションに対してマルチキャストは使用できず、オプションが無効になります。

MCAST(ONLY) 設定の使用により、トピックがマルチキャストのサブスクリプションのみを許可するように定義されている場合、呼び出しは理由コード MQRC_MULTICAST_REQUIRED により失敗します。

スコープオプション

以下のオプションは、作成されるサブスクリプションの範囲を制御します。MQSO_ALTER オプションを使用して既存のサブスクリプションを変更する場合は、このサブスクリプション有効範囲オプションを変更できません。MQSO_RESUME を使用して MQSUB 呼び出しから戻る際に、該当する有効範囲オプションが設定されます。

MQSO_SCOPE_QMGR

このサブスクリプションは、ローカル・キュー・マネージャー上のみに作成されます。プロキシ・サブスクリプションは、ネットワーク内の他のキュー・マネージャーに配布されません。このキュー・マネージャーでパブリッシュされるパブリケーションのみ、このサブスクリプターに送信されます。これは、SUBSCOPE トピック属性を使用して設定された動作を指定変更します。

注：設定しない場合は、サブスクリプションの有効範囲は SUBSCOPE トピック属性によって判別されます。

登録オプション

以下のオプションは、このサブスクリプションに対してキューマネージャに登録される詳細を制御します。MQSO_ALTER オプションを使用して既存のサブスクリプションを変更する場合は、これらの登録オプションを変更できます。MQSO_RESUME を使用する MQSUB 呼び出しから戻る際に、該当する登録オプションが設定されます。

MQSO_GROUP_SUB

このサブスクリプションは、同じキューを使用し、同じ関連 ID を指定する、同じ SubLevel の他のサブスクリプションとグループ化されます。そのため、トピックに対するパブリケーションにおいて、使用されているトピック・ストリングの集合がオーバーラップしているために複数のパブリケーション・メッセージがサブスクリプションのグループに提供される場合に、1つのメッセージのみキューに配布されるようにします。このオプションを使用しない場合は、一致する固有の各サブスクリプション (SubName によって識別される) にパブリケーションのコピーが提供されるので、多数のサブスクリプションによって共有されるキューにパブリケーションの複数のコピーが入れられることがあります。

グループ内で最も有意なサブスクリプションにのみ、パブリケーションのコピーが提供されます。最も有意なサブスクリプションは、ワイルドカードのある位置までトピック名のフルネームに基づきます。グループ中でワイルドカードの体系を混合して使用する場合は、ワイルドカードの位置のみ重要になります。同じキューを共有するサブスクリプションのグループ内で異なるワイルドカード方式を組み合わせないことをお勧めします。

新しくグループ化されたサブスクリプションを作成する際にも、固有の SubName がなければなりません。グループ中の既存のサブスクリプションのトピック名のフルネームと一致する場合は、呼び出しは MQRC_DUPLICATE_GROUP_SUB で失敗します。

グループ中のほとんどの有効なサブスクリプションで MQSO_NOT_OWN_PUBS も指定されており、これが同じアプリケーションからのパブリケーションである場合は、キューにパブリケーションが配布されません。

このオプションを指定してサブスクリプションに変更が加えられた場合は、グループ化を暗黙指定するフィールド、MQSUB 呼び出し上の Hobj (キューおよびキュー・マネージャー名を表す)、および SubCorrelId を変更できません。これらを変更しようとする、呼び出しは MQRC_GROUPING_NOT_ALTERABLE で失敗します。

このオプションは、SubCorrelId が MQCI_NONE に設定されていない MQSO_SET_CORREL_ID と結合しなければならず、MQSO_MANAGED とは結合できません。

MQSO_ANY_USERID

MQSO_ANY_USERID を指定すると、サブスクライバーの ID は単一のユーザー ID に制限されなくなります。そのため、ユーザーは適切な権限を持っていれば、サブスクリプションの変更や再開を行うことができます。一度に 1 人のユーザーだけがサブスクリプションを持つことができます。現在別のアプリケーションが使用中であるサブスクリプションの使用を再開しようとする、呼び出しは MQRC_SUBSCRIPTION_IN_USE で失敗します。

このオプションを既存のサブスクリプションに追加するには、MQSUB 呼び出し (MQSO_ALTER を使用) は元のサブスクリプション自体と同じユーザー ID からのものでなければなりません。

MQSO_ANY_USERID が設定された既存のサブスクリプションを MQSUB 呼び出しが参照する際に、元のサブスクリプションとユーザー ID が違う場合は、この呼び出しが成功するのは、トピックにサブスクライブする権限が新しいユーザー ID にある場合に限られます。正常終了すると、以後このサブスクライバーに対するパブリケーションは、パブリケーション・メッセージ中に新しいユーザー ID が設定されて、サブスクライバーのキューに書き込まれます。

MQSO_ANY_USERID と MQSO_FIXED_USERID を両方とも指定しないでください。どちらも使用しない場合のデフォルトは、MQSO_FIXED_USERID です。

MQSO_FIXED_USERID

MQSO_FIXED_USERID を指定すると、サブスクリプションを最後に変更するユーザー ID のみサブスクリプションの変更や再開を行うことができます。サブスクリプションが変更されていない場合は、サブスクリプションを作成したユーザー ID です。

MQSUB verb が、MQSO_ANY_USERID が設定された既存のサブスクリプションを参照し、MQSO_ALTER を使用して、オプション MQSO_FIXED_USERID を使用するようにこのサブスクリプションを変更すると、このサブスクリプションのユーザー ID はこの新しいユーザー ID で固定されます。このトピックにサブスクライブする権限が新しいユーザー ID にある場合にのみ、呼び出しは成功します。

サブスクリプションの所有者として記録されていないユーザー ID が MQSO_FIXED_USERID サブスクリプションの再開または変更を試行すると、呼び出しは失敗し、MQRC_IDENTITY_MISMATCH が発行

されます。サブスクリプションの所有者になっているユーザー ID は、DISPLAY SBSTATUS コマンドを使用して表示できます。

MQSO_ANY_USERID と MQSO_FIXED_USERID を両方とも指定しないでください。どちらも使用しない場合のデフォルトは、MQSO_FIXED_USERID です。

公開オプション

以下のオプションは、この購読者への出版物の送信方法を制御します。MQSO_ALTER オプションを使用して既存のサブスクリプションを変更する場合は、これらのパブリケーション・オプションを変更できます。

MQSO_NOT_OWN_PUBS

アプリケーションが独自のパブリケーションを参照しないことを、ブローカーに指示します。接続ハンドルが同じ場合、パブリケーションは同じアプリケーションから発信されたと思なされます。

MQSO_RESUME を使用する MQSUB 呼び出しから戻る際に、該当する場合はこのオプションが設定されます。

MQSO_NEW_PUBLICATIONS_ONLY

このサブスクリプションの作成時に、現在保存されているパブリケーションは送信されません。新しいパブリケーションのみ送信されます。このオプションは、MQSO_CREATE の指定時のみ適用されます。以後のサブスクリプションに対する変更により、パブリケーションのフローは変更されないで、トピック上に保存されているパブリケーションは新しいパブリケーションとしてサブスクライバーにすでに送信されていることとなります。

MQSO_CREATE を指定せずにこのオプションを指定すると、呼び出しは MQRC_OPTIONS_ERROR で失敗します。MQSO_RESUME を使用する MQSUB 呼び出しから戻る際に、このオプションを使用してサブスクリプションが作成された場合でもこのオプションは設定されません。

このオプションを使用しないと、以前に保存されたメッセージは、提供された宛先キューに送信されます。エラー MQRC_RETAINED_MSG_Q_ERROR または MQRC_RETAINED_NOT_DELIVERED のためにこのアクションが失敗すると、サブスクリプションの作成は失敗します。

MQSO_PUBLICATIONS_ON_REQUEST

このオプションを設定すると、特に必要な時点でサブスクライバーが情報を要求することを示します。キュー・マネージャーは非送信請求メッセージをサブスクライバーに送信しません。前の MQSUB 呼び出しから Hsub ハンドルを使用して MQSUBRQ 呼び出しを行うたびに、保存パブリケーション(トピック中でワイルドカードが指定されている場合は複数のパブリケーションの可能性あり)がサブスクライバーに送信されます。このオプションを使用して MQSUB 呼び出しを行っても、パブリケーションは送信されません。MQSO_RESUME を使用する MQSUB 呼び出しから戻る際に、該当する場合はこのオプションが設定されます。

このオプションは、1 より大きな値の SubLevel と組み合わせて使用すると無効になります。

先読みオプション

以下のオプションは、非永続メッセージがアプリケーションからの要求に先立ってアプリケーションに送信されるかどうかを制御します。

MQSO_READ_AHEAD_AS_Q_DEF

MQSUB 呼び出しで管理対象ハンドルが使用される場合、メッセージをアプリケーションが要求する前にアプリケーションに送信するかどうかは、サブスクライブ先のトピックに関連付けられたモデル・キューのデフォルトの先読み属性で決まります。

これがデフォルト値です。

MQSO_NO_READ_AHEAD

MQSUB 呼び出しで管理対象ハンドルが使用される場合、メッセージはアプリケーションが要求する前にはアプリケーションに送信されません。

MQSO_READ_AHEAD

MQSUB 呼び出しで管理対象ハンドルが使用される場合、メッセージはアプリケーションが要求する前にアプリケーションに送信される可能性があります。

注：以下の注意事項は、先読みオプションに適用されます。

1. これらのオプションは、1つだけ指定できます。MQOO_READ_AHEAD と MQOO_NO_READ_AHEAD の両方を指定すると、理由コード MQRC_OPTIONS_ERROR が戻されます。これらのオプションは、MQSO_MANAGED を指定した場合のみ適用できます。
2. これらは、既に開かれているキューが渡される場合は MQSUB には適用されません。要求時に先読みが有効になっていない可能性があります。最初の MQGET 呼び出しで使用された MQGET オプションによって、先読みが有効にならなくなる場合があります。また、先読みがサポートされていないキュー・マネージャーにクライアントが接続しているときは、先読みが無効になります。アプリケーションが IBM MQ クライアントとして実行されていない場合、これらのオプションは無視されます。

ワイルドカードオプション

以下のオプションは、MQSD の ObjectString フィールドで指定された文字列におけるワイルドカードの解釈方法を制御します。これらのオプションのうち1つのみ指定できます。MQSO_ALTER オプションを使用して既存のサブスクリプションを変更する場合は、これらのワイルドカード・オプションを変更できません。MQSO_RESUME を使用する MQSUB 呼び出しから戻る際に、該当するワイルドカード・オプションが設定されます。

MQSO_WILDCARD_CHAR

ワイルドカードは、トピック・ストリング中の文字のみに対して作動します。

以下の表に、MQSO_WILDCARD_CHAR で定義される振る舞いを示します。

表 557. ワイルドカードの解釈	
特殊文字	動作
スラッシュ (/)	特に意味はなく、単に1つの文字
アスタリスク (*)	ワイルドカード、ゼロ以上の文字
疑問符 (?)	ワイルドカード、1文字
パーセント記号 (%)	ストリング内で (*), (?), または (%) を特殊文字として解釈するのではなく文字として使用できるようにするためのエスケープ文字。例えば (%*), (%?), または (%) のように使用します。

例えば、以下のトピック上でパブリッシュするとします。

```
/level0/level1/level2/level3/level4
```

このトピックは、以下のトピックを使用するサブスクライバーと一致します。

```
*
/*
/ level0/level1/level2/level3/*
/ level0/level1/*/level3/level4
/ level0/level1/le?el2/level3/level4
```

注：パブリッシュ/サブスクライブに関する MQRFH1 形式のメッセージを使用している場合、このワイルドカードの使用法により、IBM MQ V6 および WebSphere MB V6 で提供される意味が正確に提供されます。この方法は、新しく作成するアプリケーションには使用しないことをお勧めします。以前にこのバージョンに対して実行していたもので、MQSO_WILDCARD_TOPIC で説明されているデフォルトのワイルドカードの動作を使用するように変更されていないアプリケーションについてのみ使用してください。

MQSO_WILDCARD_TOPIC

ワイルドカードは、トピック・ストリング中のトピック・エレメントのみに対して作動します。デフォルトを選択していない場合は、これがデフォルトの動作になります。

以下の表には、MQSO_WILDCARD_TOPIC で必要な動作が示されています。

表 558. ワイルドカードの解釈	
特殊文字	動作
(/)	トピック・レベルの分離文字
番号記号 (#)	ワイルドカード: 複数のトピック・レベル
正符号 (+)	ワイルドカード: 単一のトピック・レベル

注:

(+) と (#) は、1つのトピック・レベル内で他の文字 (これらの文字自体を含む) と混在していると、ワイルドカードとして扱われません。以下のストリングでは、(#) と (+) の文字は通常の文字として扱われます。

```
level0/level1/#+/level3/level#
```

例えば、以下のトピック上でパブリッシュするとします。

```
/level0/level1/level2/level3/level4
```

このトピックは、以下のトピックを使用するサブスクライバーと一致します。

```
#
/#
/ level0/level1/level2/level3/#
/ level0/level1/+/level3/level4
```

その他の選択肢

以下のオプションは、サブスクリプションではなく API 呼び出しの方法自体を制御します。MQSO_RESUME を使用する MQSUB 呼び出しから戻る際に、これらのオプションは変更されません。詳しくは、566 ページの『[AlternateUserId \(MQCHAR12\)](#)』を参照してください。

MQSO_ALTERNATE_USER_AUTHORITY

AlternateUserId フィールドは、この MQSUB 呼び出しの妥当性検査に使用するユーザー ID を格納します。指定されたアクセス・オプションでオブジェクトを開く許可が AlternateUserId に与えられる場合にのみ、呼び出しが成功します。アプリケーションを実行しているユーザー ID にそのような許可が与えられているかどうかは関係ありません。

MQSO_SET_CORREL_ID

サブスクリプションは、SubCorrelId フィールド中に提供される相関 ID を使用します。このオプションを指定しないと、サブスクリプション時にキュー・マネージャーによって相関 ID が自動的に作成され、SubCorrelId フィールドでアプリケーションに戻されます。詳細については、詳細情報 [SubCorrelId](#) をご覧ください。

このオプションは、MQSO_MANAGED と結合できません。

MQSO_SET_IDENTITY_CONTEXT

サブスクリプションは、PubAccountingToken フィールドおよび PubApplIdentityData フィールドに指定される会計トークンおよびアプリケーション ID データを使用することになります。

このオプションを指定すると、MQOO_SET_IDENTITY_CONTEXT を指定した MQOPEN 呼び出しを使用して宛先キューがアクセスされた場合と同じ許可検査が実行されます。ただし、MQSO_MANAGED オプションも使用する場合は例外で、この場合は宛先キューに関する許可検査は行われません。

このオプションを指定しないと、以下のように、このサブスクライバーに送信されるパブリケーションにデフォルトのコンテキスト情報が関連付けられます。

表 559. このサブスクライバーに送信されるパブリケーションのデフォルトのコンテキスト情報	
MQMD のフィールド	使用される値
<i>UserIdentifier</i>	サブスクリプションが作成されたときにサブスクリプションに関連付けられたユーザー ID。
<i>AccountingToken</i>	環境から決定できる場合はその値。決定できないときは、MQACT_NONE に設定される。
<i>ApplIdentityData</i>	ブランクに設定される。

このオプションは、MQSO_CREATE および MQSO_ALTER と併用する場合のみ有効です。MQSO_RESUME と併用する場合は、*PubAccountingToken* および *PubApplIdentityData* フィールドは無視されるので、このオプションは無効になります。

以前にサブスクリプションが ID コンテキスト情報を提供した場所でこのオプションを使用しないでサブスクリプションを変更すると、変更されたサブスクリプションに関するデフォルトのコンテキスト情報が生成されます。

サブスクリプションで、さまざまなユーザー ID がオプション MQSO_ANY_USERID を指定してそのサブスクリプションを使用することを許可している場合、別のユーザー ID がそのサブスクリプションを再開すると、現在のそのサブスクリプションの所有者となるその新しいユーザー ID に関するデフォルトの ID コンテキストが生成され、送達されるそれ以降のパブリケーションにはその新しい ID コンテキストが含まれるようになります。

MQSO_FAIL_IF_QUIESCING

MQSUB 呼び出しは、キュー・マネージャーが静止状態になっている場合は失敗します。z/OS では、CICS または IMS アプリケーションについてこのオプションを指定すると、接続が静止状態になっている場合には MQSUB 呼び出しを強制的に失敗させます。

ObjectName (MQCHAR48)

これは、ローカル・キュー・マネージャーに定義されたトピック・オブジェクトの名前です。

この名前には、以下に示す文字を使用できます。

- 英大文字 (A から Z まで)
- 英小文字 (a から z まで)
- 数字 (0 から 9 まで)
- ピリオド (.)、スラッシュ (/)、下線 (_)、パーセント (%)

名前の先頭をブランクにしたり、名前にブランクを埋め込んだりすることはできませんが、名前の後にブランクを入れることはできます。ヌル文字を使用して、名前の中における有効なデータの末尾を示します。ヌル文字とそれに続く文字はすべてブランクとして扱われます。以下に示す制約事項は、それぞれ明記している環境に適用されます。

- EBCDIC カタカナを使用するシステムでは、小文字を使用できません。
- On z/OS:
 - 先頭または末尾に下線がある名前は使用しないでください。これらの名前は、操作パネルや制御パネルで処理できません。
 - パーセント文字は、RACF では特別な意味があります。RACF を外部セキュリティー・マネージャーとして使用する場合、名前にはパーセントを含めないでください。パーセントが含まれていると、RACF 総称プロファイルを使用したときに、それらの名前はどのセキュリティー検査にも組み込まれません。
- IBM i で英小文字、スラッシュ、パーセントの各文字が含まれている名前をコマンドに指定する場合は、それを引用符で囲む必要があります。構造体内のフィールドまたは呼び出しのパラメーターとして指定する名前には、引用符を使用してはなりません。

ObjectName は、トピック名のフルネームを形成するのに使用します。

トピック名のフルネームは、*ObjectName* および *ObjectString* の 2 種類のフィールドからビルドできます。これらの 2 つのフィールドの使用の詳細については、「[トピック文字列の結合](#)」を参照してください。

ObjectName フィールドによって識別されるオブジェクトを検索できない場合は、*ObjectString* で指定された文字列がある場合でも、呼び出しは理由コード MQRC_UNKNOWN_OBJECT_NAME で失敗します。

MQSO_RESUME オプションを使用する MQSUB 呼び出しから戻る際に、このフィールドは変更されません。

このフィールドの長さは MQ_TOPIC_NAME_LENGTH によって指定されます。このフィールドの初期値は、C 言語ではヌル・文字列であり、他のプログラミング言語では 48 桁の空白文字です。

MQSO_ALTER オプションを使用して既存のサブスクリプションを変更する場合は、サブスクライブされるトピック・オブジェクトの名前を変更できません。このフィールドと *ObjectString* フィールドは省略できます。これらのフィールドが提供される場合は、その解決結果が同じトピック名のフルネームにならなければなりません。そうでない場合、呼び出しは MQRC_TOPIC_NOT_ALTERABLE で失敗します。

AlternateUserId (MQCHAR12)

MQSO_ALTERNATE_USER_AUTHORITY を指定する場合、このフィールドには、代替ユーザー ID が入っています。代替ユーザー ID は、アプリケーションが現在実行されているユーザー ID の代わりに、サブスクリプションや宛先キューに対する出力 (MQSUB 呼び出しの **Hobj** パラメーターで指定) の許可を検査するために使用されるものです。

正常に実行されると、アプリケーションが現在実行されているユーザー ID の代わりに、このフィールドで指定されたユーザー ID がサブスクリプション所有ユーザー ID として記録されます。

MQSO_ALTERNATE_USER_AUTHORITY が指定されていて、このフィールドが最初のヌル文字またはフィールドの終わりまで全体が完全に空白になっている場合、サブスクリプションが成功するのは、指定されたオプションまたは出力の宛先キューでこのトピックにサブスクライブするのにユーザー許可が必要でない場合だけです。

MQSO_ALTERNATE_USER_AUTHORITY が指定されていない場合は、このフィールドは無視されます。

以下の環境では、次のような違いがあります。

- z/OS では、サブスクリプションのための許可を検査するために、AlternateUserId の最初の 8 文字だけが使用されます。ただし、現行のユーザー ID に、この特定の代替ユーザー ID を指定する権限があることが必要です。この検査には、代替ユーザー ID の 12 文字がすべて使用されます。ユーザー ID に指定できる文字は、外部セキュリティ管理プログラムにより許可されています。

MQSO_RESUME を使用する MQSUB 呼び出しから戻る際に、このフィールドは変更されません。

これは入力フィールドです。このフィールドの長さは MQ_USER_ID_LENGTH によって指定されます。このフィールドの初期値は、C 言語ではヌル・文字列ですが、その他のプログラミング言語では 12 個の空白文字です。

AlternateSecurityId (MQBYTE40)

これは、AlternateUserId と共に許可サービスに渡されて、適切な許可検査を実行できるようにするセキュリティ ID です。

AlternateSecurityId が使用されるのは、MQSO_ALTERNATE_USER_AUTHORITY が指定されており、AlternateUserId フィールドが最初のヌル文字かフィールドの終わりまですべて空白でない場合のみです。

MQSO_RESUME を使用する MQSUB 呼び出しから戻る際に、このフィールドは変更されません。

詳しくは、MQOD データ・タイプの [462 ページ](#)の『AlternateSecurityId (MQBYTE40)』の説明を参照してください。

SubExpiry (MQLONG)

これは、サブスクリプションの満了後の時間で、10 分の 1 秒単位で表されます。この間隔が渡された後は、パブリケーションはこのサブスクリプションと突き合わせられません。サブスクリプションが満了すると即時に、パブリケーションはキューに送信されなくなります。ただし、既にキューに入っているパブリケーションが影響を受けることはありません。*SubExpiry* は、パブリケーションの満了期限には影響を与えません。

以下のような特殊値が認識されます。

MQEI_UNLIMITED

サブスクリプションの満了に期限はありません。

MQSO_ALTER オプションを使用して既存のサブスクリプションを変更する場合は、サブスクリプションの満了を変更できます。

MQSO_RESUME オプションを使用する MQSUB 呼び出しから戻る際、このフィールドはサブスクリプションの有効期限までの残り時間ではなく元の有効期限に設定されます。

ObjectString (MQCHARV)

これは使用される長いオブジェクト名です。

ObjectString は、トピック名のフルネームを形成するのに使用します。

トピック名のフルネームは、*ObjectName* および *ObjectString* の 2 種類のフィールドからビルドできます。これらの 2 つのフィールドの使用の詳細については、[「トピック文字列の結合」](#) を参照してください。

ObjectString の最大長は 10240 です。

MQCHARV 構造体の使い方に関する説明に従うと *ObjectString* の指定が正しくない場合、または最大長を超える場合、呼び出しは失敗し、理由コード MQRC_OBJECT_STRING_ERROR が戻されます。

これは入力フィールドです。この構造体のフィールドの初期値は、MQCHARV 構造体のものと同じです。

ワイルドカードが *ObjectString* 中にある場合は、MQSD の Options フィールドで指定されたワイルドカード・オプションを使用してこれらのワイルドカードの解釈を制御できます。

MQSO_RESUME オプションを使用する MQSUB 呼び出しから戻る際に、このフィールドは変更されません。バッファが提供される場合、使用されるトピック名のフルネームが *ResObjectString* フィールドに戻されます。

MQSO_ALTER オプションを使用して既存のサブスクリプションを変更する場合は、サブスクリプションされるトピック・オブジェクトの長い名前を変更できません。このフィールドと *ObjectName* フィールドは省略できます。これらのフィールドが提供される場合は、その解決結果が同じトピック名のフルネームにならなければなりません。そうでない場合、呼び出しは MQRC_TOPIC_NOT_ALTERABLE で失敗します。

SubName (MQCHARV)

これはサブスクリプション名を指定します。このフィールドは、Options でオプション MQSO_DURABLE が指定されているものの、提供内容が MQSO_NON_DURABLE に関するキュー・マネージャーでも使用される場合のみ必須です。

指定する場合は、*SubName* はサブスクリプションの識別に使用する方式なので、キュー・マネージャー中で固有でなければなりません。

SubName の最大長は 10240 です。

このフィールドは 2 つの目的を果たします。MQSO_DURABLE サブスクリプションの場合、このフィールドを使用してサブスクリプションを識別し、サブスクリプションに対するハンドルをクローズした場合 (MQCO_KEEP_SUB オプションを使用) か、キュー・マネージャーから切断された場合に、サブスクリプションの作成後に再開できるようにします。この処理は MQSO_RESUME オプションを指定した MQSUB 呼び出しを使用して行われます。また、サブスクリプションの管理ビューにおいて、DISPLAY SBSTATUS の SUBID フィールドにも表示されます。

MQCHARV 構造体の使用方法の説明に従い、が **SubName** 誤って指定された場合、必要なときに（つまり **SubName**）が省略される。**VSLength** がゼロである場合）、またはこれが最大長を超える場合、呼び出しは失敗し、理由コード **MQRC_SUB_NAME_ERROR** が戻ります。

これは入力フィールドです。この構造体のフィールドの初期値は、**MQCHARV** 構造体のものと同じです。

MQSO_ALTER オプションを使用して既存のサブスクリプションを変更する場合、サブスクリプション名は、参照されているサブスクリプションの検索に使用する識別フィールドなので、変更できません。

MQSO_RESUME オプションを指定した **MQSUB** 呼び出しからの出力上で変更されません。

SubUserData (MQCHARV)

これはサブスクリプション・ユーザー・データを指定します。このフィールドでサブスクリプションについて提供されるデータは、このサブスクリプションへ送信される各パブリケーションの **MQSubUserData** メッセージ・プロパティとして含まれます。

SubUserData の最大長は 10240 です。

SubUserData の指定が (**MQCHARV** 構造の使用法の説明に従って) 誤っている場合は、または最大長を超える場合は、呼び出しが理由コード **MQRC_SUB_USER_DATA_ERROR** で失敗します。

これは入力フィールドです。この構造体のフィールドの初期値は、**MQCHARV** 構造体のものと同じです。

MQSO_ALTER オプションを使用して既存のサブスクリプションを変更する場合は、サブスクリプション・ユーザー・データを変更できます。

バッファが提供されている場合は、**MQSO_RESUME** オプションを使用する **MQSUB** 呼び出しからの出力上にこの可変長フィールドが戻され、**VSBuflen** 中には正のバッファ長があります。呼び出し上でバッファが提供されていない場合は、**MQCHARV** の **VSLength** フィールドにサブスクリプション・ユーザー・データの長さのみ戻されます。提供されているバッファがフィールドを戻すのに必要なスペースより小さい場合は、提供されているバッファに **VSBuflen** のバイト数のみ戻されます。

SubCorrelId (MQBYTE24)

このフィールドには、このサブスクリプションと突き合わせるすべてのパブリケーションに共通する相関 ID が含まれます。



重要: 相関 ID は、階層内ではなく、パブリッシュ/サブスクライブ・クラスター内のキュー・マネージャー間でのみ受け渡し可能です。

このサブスクリプションと突き合わせるために送信されるすべてのパブリケーションには、メッセージ記述子中にこの相関 ID が含まれます。複数のサブスクリプションが同じキューからパブリケーションを読み取る場合は、相関 ID で **MQGET** を使用すると、特定のサブスクリプションに関するパブリケーションのみ入手できます。この相関 ID はキュー・マネージャーまたはユーザーのいずれかによって生成されます。

オプション **MQSO_SET_CORREL_ID** を指定しない場合は、相関 ID はキュー・マネージャーによって生成され、このフィールドは、このサブスクリプションに関してパブリッシュされる各メッセージ中で設定される相関 ID を含む出力フィールドになります。生成される相関 ID は、4 バイトの製品 ID (ASCII と EBCDIC のどちらでも **AMQX** または **CSQM**) と、その後ろに組み込まれる、製品によって異なる固有のストリングから構成されます。

オプション **MQSO_SET_CORREL_ID** を指定する場合は、相関 ID はユーザーによって生成され、このフィールドは、このサブスクリプションに関する各パブリケーション中で設定される相関 ID を含む入力フィールドになります。この場合、フィールドに **MQCI_NONE** が含まれていると、このサブスクリプションに関してパブリッシュされる各メッセージ中で設定される相関 ID は、元のメッセージ書き込みによって作成された相関 ID です。

オプション **MQSO_GROUP_SUB** が指定されており、指定された相関 ID が、同じキューおよびオーバーラップ・トピック・ストリングを使用する既存のグループ化されたサブスクリプションと同じである場合、グループ内で最も有意なサブスクリプションのみがパブリケーションのコピーと共に提供されます。

このフィールドの長さは **MQ_CORREL_ID_LENGTH** によって指定されます。このフィールドの初期値は **MQCI_NONE** です。

MQSO_ALTER オプションを使用して既存のサブスクリプションを変更する場合、このフィールドが入力フィールドであれば、サブスクリプション関連 ID を変更できます。ただし、サブスクリプションがグループ化されている、つまりオプション MQSO_GROUP_SUB を使用して作成されたサブスクリプションである場合は例外で、この場合はサブスクリプション関連 ID を変更できません。

MQSO_RESUME を使用する MQSUB 呼び出しから戻る際に、このフィールドはサブスクリプションの現行の関連 ID に設定されます。

PubPriority (MQLONG)

この値は、このサブスクリプションと突き合わせるすべてのパブリケーション・メッセージのメッセージ記述子 (MQMD) の *Priority* フィールドにあります。MQMD 中の *Priority* フィールドについては詳しくは、417 ページの『[Priority \(MQLONG\)](#)』を参照してください。

値はゼロ以上でなければなりません。ゼロは、最低優先順位です。以下のような特殊値も使用できます。

MQPRI_PRIORITY_AS_Q_DEF

MQSUB 呼び出し中の *Hobj* フィールド中でサブスクリプション・キューが提供されており、これが管理対象ハンドルでない場合は、メッセージの優先順位はこのキューの **DefPriority** 属性から取られます。キューがクラスター・キューであるか、キュー名の解決パスに定義が 2 つ以上ある場合には、417 ページの『[Priority \(MQLONG\)](#)』の説明どおりにパブリケーション・メッセージがキューに書き込まれる際に優先順位が判別されます。

MQSUB 呼び出しで管理対象ハンドルを使用した場合、メッセージの優先順位は、サブスクライブするトピックに関連付けられたモデル・キューの **DefPriority** 属性から取得されます。

MQPRI_PRIORITY_AS_PUBLISHED

メッセージの優先順位は、元のパブリケーションの優先順位です。これはフィールドの初期値です。

MQSO_ALTER オプションを使用して既存のサブスクリプションを変更する場合は、以後のパブリケーション・メッセージの *Priority* を変更できます。

MQSO_RESUME を使用する MQSUB 呼び出しから戻る際に、このフィールドはサブスクリプションの現在使用中の優先順位に設定されます。

PubAccountingToken (MQBYTE32)

この値は、このサブスクリプションと突き合わせるすべてのパブリケーション・メッセージのメッセージ記述子 (MQMD) の *AccountingToken* フィールドにあります。*AccountingToken* は、メッセージ ID コンテキストの一部です。メッセージのコンテキストの詳細については、[メッセージのコンテキスト](#)を参照してください。MQMD のフィールド *AccountingToken* に関する詳細については、以下を参照してください。425 ページの『[AccountingToken \(MQBYTE32\)](#)』

以下の特殊値を *PubAccountingToken* フィールドで使用できます。

MQACT_NONE

アカウンティング・トークンが指定されていません。

値は、フィールドの長さについては 2 進ゼロです。

C 言語の場合、定数 MQACT_NONE_ARRAY も定義されます。これは、MQACT_NONE と同じ値ですが、ストリングではなく文字の配列です。

オプション MQSO_SET_IDENTITY_CONTEXT を指定しない場合は、会計トークンはキュー・マネージャーによって生成され、このフィールドは、このサブスクリプションに関してパブリッシュされる各メッセージ中で設定される *AccountingToken* を含む出力フィールドになります。

オプション MQSO_SET_IDENTITY_CONTEXT を指定する場合は、会計トークンはユーザーによって生成され、このフィールドは、このサブスクリプションに関する各パブリケーション中で設定される *AccountingToken* を含む入力フィールドになります。

このフィールドの長さは MQ_ACCOUNTING_TOKEN_LENGTH によって指定されます。このフィールドの初期値は MQACT_NONE です。

MQSO ALTER オプションを使用して既存のサブスクリプションを変更する場合は、以後のパブリケーション・メッセージ中の *AccountingToken* の値を変更できます。

MQSO RESUME を使用する MQSUB 呼び出しから戻る際に、このフィールドはサブスクリプションの現在使用中の *AccountingToken* に設定されます。

PubApplIdentityData (MQCHAR32)

この値は、このサブスクリプションと一致するすべてのパブリケーション・メッセージのメッセージ記述子 (MQMD) の *ApplIdentityData* フィールドの値です。 *ApplIdentityData* は、メッセージ ID コンテキストの一部です。メッセージのコンテキストの詳細については、[メッセージのコンテキストを参照してください](#)。MQMD のフィールド *ApplIdentityData* に関する詳細については、以下を参照してください。427 ページの『[ApplIdentityData \(MQCHAR32\)](#)』

オプション MQSO_SET_IDENTITY_CONTEXT を指定しない場合は、このサブスクリプションに関してパブリッシュされる各メッセージに設定される *ApplIdentityData* は、デフォルトのコンテキスト情報としてブランクになります。

オプション MQSO_SET_IDENTITY_CONTEXT を指定する場合は、*PubApplIdentityData* はユーザーによって生成され、このフィールドは、このサブスクリプションに関する各パブリケーション中で設定される *ApplIdentityData* を含む入力フィールドになります。

このフィールドの長さは MQ_APPL_IDENTITY_DATA_LENGTH によって指定されます。このフィールドの初期値は、C 言語ではヌル・ストリングですが、その他のプログラミング言語では 32 桁のブランク文字です。

MQSO ALTER オプションを使用して既存のサブスクリプションを変更する場合は、以後のパブリケーション・メッセージの *ApplIdentityData* を変更できます。

MQSO RESUME を使用する MQSUB 呼び出しから戻る際に、このフィールドはサブスクリプションの現在使用中の *ApplIdentityData* に設定されます。

SelectionString (MQCHARV)

これは、一つのトピックからメッセージをサブスクライブする時に使用される選択基準を提供するためのストリングです。

この可変長フィールドは、MQSO_RESUME オプションを使用している MQSUB 呼び出しからの出力に戻れます。バッファが指定されていれば、VSBufSize に正のバッファ長もあります。呼び出しにバッファの指定がない場合は、選択ストリングの長さだけが、MQCHARV の VSLength フィールドに戻れます。フィールドを返すのに必要なスペースよりも提供されたバッファが小さい場合、VSBufSize バイトのみがそのバッファに戻されます。

構造 240 ページの『[MQCHARV - 可変長ストリング](#)』体の使用方法の説明に従ってが正しく指定されていない *SelectionString* 場合、または最大長を超えた場合、呼び出しは理由コード MQRC_SELECTION_STRING_ERROR で失敗します。

SelectionString の使用法については、[セレクトター](#)で説明しています。

SubLevel (MQLONG)

これはサブスクリプションに関連付けられているレベルです。パブリケーションがこのサブスクリプションに配布されるのは、このサブスクリプションが属するサブスクリプション・セットで使用される SubLevel の最高値が、パブリケーション時に使用される PubLevel 値以下である場合のみです。しかし、パブリケーションが保持されると PubLevel 1 でリパブリッシュされるため、パブリケーションは高いレベルのサブスクライバーで使用することはできなくなります。

値は 0 から 9 の範囲でなければなりません。ゼロが最低レベルです。

このフィールドの初期値は 1 です。

詳細については、[インターセプト・パブリケーション](#)を参照してください。

MQSO_ALTER オプションを使用して既存のサブスクリプションを変更する場合は、SubLevel を変更できません。

1 より大きな値の SubLevel と MQSO_PUBLICATIONS_ON_REQUEST オプションを組み合わせで指定することはできません。

MQSO_RESUME を使用する MQSUB 呼び出しから戻る際に、このフィールドはサブスクリプションの現在使用中のレベルに設定されます。

ResObjectString (MQCHARV)

これは、*ObjectName* で指定される名前をキュー・マネージャーが解決した後の長いオブジェクト名です。

ObjectString に長いオブジェクト名が指定され、*ObjectName* に何も指定されない場合、このフィールドに戻される値は、*ObjectString* で指定されたものと同じになります。

このフィールドが省略されている (つまり *ResObjectString.VSBufSize* がゼロである) 場合、*ResObjectString* は戻されませんが、長さが *ResObjectString.VSLength* に戻されます。この長さが *ResObjectString* の全長より短い場合は、切り捨てられ、指定された長さに入るだけの文字が右端から戻されます。

[MQCHARV](#) 構造体の使い方に関する説明に従うと *ResObjectString* の指定が正しくない場合、または最大長を超える場合、呼び出しは失敗し、理由コード MQRC_RES_OBJECT_STRING_ERROR が戻されます。

MQSMPO - メッセージ・プロパティ設定オプション

MQSMPO 構造体を使用すると、アプリケーションで、メッセージのプロパティを設定する方法を制御するオプションを指定できます。この構造は、**MQSETMP** 呼び出しの入力パラメーターです。

可用性

すべての IBM MQ システムおよび IBM MQ クライアント。

文字セットとエンコード

入力データは、アプリケーションの文字セットおよびアプリケーション **MQSMPO** のエンコーディング (**MQENC_NATIVE**) でなければならない。

フィールド

注: 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 560. MQSMPO のフィールド		
フィールド名と説明	定数の名前	定数の初期値 (存在する場合)
StrucId (構造識別子)	MQSMPO_STRUC_ID	' SMP0 '
バージョン (構造バージョン番号)	MQSMPO_VERSION_1	1
オプション (オプション)	MQSMPO_NONE	0
ValueEncoding (プロパティ値エンコーディング)	MQENC_NATIVE	環境に依存
ValueCCSID (プロパティ値の文字セット)	MQCCSI_APPL	-3

表 560. MQSMPO のフィールド (続き)

フィールド名と説明	定数の名前	定数の初期値（存在する場合）
注: 1. ヌル・ストリングまたはブランクの値は、C 言語ではヌル・ストリングを表し、他のプログラミング言語ではブランク文字を表します。 2. C プログラミング言語では、マクロ変数 MQSMPO_DEFAULT には、表にリストされている値が含まれています。この変数を以下の方法で使用すると、構造体のフィールドに初期値を設定できます。 <pre>MQSMPO MySMPO = {MQSMPO_DEFAULT};</pre>		

言語ごとの宣言

MQSMPO の C 宣言

```
typedef struct tagMQSMPO MQSMPO;
struct tagMQSMPO {
    MQCHAR4    StrucId;          /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQLONG     Options;          /* Options that control the action of MQSETMP */
    MQLONG     ValueEncoding;    /* Encoding of Value */
    MQLONG     ValueCCSID;       /* Character set identifier of Value */
};
```

MQSMPO の COBOL 宣言

```
**      MQSMPO structure
10 MQSMPO.
**      Structure identifier
15 MQSMPO-STRUCID      PIC X(4).
**      Structure version number
15 MQSMPO-VERSION      PIC S9(9) BINARY.
**      Options that control the action of MQSETMP
15 MQSMPO-OPTIONS      PIC S9(9) BINARY.
**      Encoding of VALUE
15 MQSMPO-VALUEENCODING PIC S9(9) BINARY.
**      Character set identifier of VALUE
15 MQSMPO-VALUECCSID   PIC S9(9) BINARY.
```

MQSMPO の PL/I 宣言

```
dcl
1 MQSMPO based,
3 StrucId      char(4),          /* Structure identifier */
3 Version      fixed bin(31),    /* Structure version number */
3 Options      fixed bin(31),    /* Options that control the action of MQSETMP */
3 ValueEncoding fixed bin(31),    /* Encoding of Value */
3 ValueCCSID   fixed bin(31),    /* Character set identifier of Value */
```

MQSMPO の高水準アセンブラ宣言

```
MQSMPO          DSECT
MQSMPO_STRUCID  DS    CL4      Structure identifier
MQSMPO_VERSION  DS    F        Structure version number
MQSMPO_OPTIONS  DS    F        Options that control the action of
*                               MQSETMP
MQSMPO_VALUEENCODING DS    F    Encoding of VALUE
MQSMPO_VALUECCSID DS    F      Character set identifier of VALUE
MQSMPO_LENGTH   EQU    *-MQSMPO
MQSMPO_AREA     DS    CL(MQSMPO_LENGTH)
```


MQSMPO のフィールド詳細

メッセージのプロパティ設定 **MQSMPO** オプション構造体により、アプリケーションはメッセージのプロパティ設定方法を制御するオプションを指定できます。この構造は、**MQSETMP** 呼び出しの入力パラメーターです。

フィールド

注: 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 561. MQSMPO のフィールド		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
StrucId (構造識別子)	MQSMPO_STRUC_ID	'SMP0'
バージョン (構造バージョン番号)	MQSMPO_VERSION_1	1
オプション (オプション)	MQSMPO_NONE	0
ValueEncoding (プロパティ値エンコーディング)	MQENC_NATIVE	環境に依存
ValueCCSID (プロパティ値の文字セット)	MQCCSI_APPL	-3
注: - ヌル・ストリングまたはブランクの値は、C 言語ではヌル・ストリングを表し、他のプログラミング言語ではブランク文字を表します。 - C プログラミング言語では、マクロ変数 MQSMPO_DEFAULT には、表にリストされている値が含まれています。この変数を以下の方法で使用すると、構造体のフィールドに初期値を設定できます。 <pre>MQSMPO MySMP0 = {MQSMPO_DEFAULT};</pre>		

StrucId (MQCHAR4)

これは、メッセージプロパティオプション構造体の構造体識別子です。常に入力フィールドです。その値は **MQSMPO_STRUC_ID** です。

値は次のものでなければなりません。

MQSMPO_STRUC_ID

メッセージプロパティオプション構造体の識別子。

C プログラミング言語の場合、定数 **MQSMPO_STRUC_ID_ARRAY** も定義されます。これはと同じ値 **MQSMPO_STRUC_ID** を持ちますが、文字列ではなく文字の配列です。

Version (MQLONG)

これは構造体のバージョン番号です。値は以下のものでなければなりません。

MQSMPO_VERSION_1

バージョン 1 のメッセージ・プロパティ設定オプション構造。

以下の定数は、現行バージョンのバージョン番号を指定しています。

MQSMPO_CURRENT_VERSION

メッセージ・プロパティ設定オプション構造の現行バージョン。

これは常に入力フィールドです。このフィールドの初期値は **MQSMPO_VERSION_1** です。

Options (MQLONG)

場所の選択肢

以下は、プロパティ・カーソルと比較したプロパティの相対位置に関するオプションです。

MQSMPO_SET_FIRST

指定した名前と一致する最初のプロパティの値を設定します。これが存在しない場合には、階層がこれと一致する他のすべてのプロパティの後に、新しいプロパティを追加します。

MQSMPO_SET_PROP_UNDER_CURSOR

プロパティ・カーソルによって指し示されるプロパティの値を設定します。プロパティ・カーソルによって指し示されるプロパティは、MQIMPO_INQ_FIRST または MQIMPO_INQ_NEXT オプションのいずれかによって最後に照会されたものです。

MQGET 呼び出しでメッセージ・ハンドルを再利用する場合や、MQPUT 呼び出しの MQGMO または MQPMO 構造体の *MsgHandle* フィールドにメッセージ・ハンドルが指定された場合には、プロパティ・カーソルはリセットされます。

プロパティ・カーソルがまだ確立されていない時点でこのオプションを使用する場合や、プロパティ・カーソルによって指し示されるプロパティが削除されている場合は、呼び出しは完了コード MQCC_FAILED および理由コード MQRC_PROPERTY_NOT_AVAILABLE で失敗します。

MQSMPO_SET_PROP_BEFORE_CURSOR

プロパティ・カーソルによって指し示されるプロパティの前に新しいプロパティを設定します。プロパティ・カーソルによって指し示されるプロパティは、MQIMPO_INQ_FIRST または MQIMPO_INQ_NEXT オプションのいずれかによって最後に照会されたものです。

MQGET 呼び出しでメッセージ・ハンドルを再利用する場合や、MQPUT 呼び出しの MQGMO または MQPMO 構造体の *MsgHandle* フィールドにメッセージ・ハンドルが指定された場合には、プロパティ・カーソルはリセットされます。

プロパティ・カーソルがまだ確立されていない時点でこのオプションを使用する場合や、プロパティ・カーソルによって指し示されるプロパティが削除されている場合は、呼び出しは完了コード MQCC_FAILED および理由コード MQRC_PROPERTY_NOT_AVAILABLE で失敗します。

MQSMPO_SET_PROP_AFTER_CURSOR

プロパティ・カーソルによって指し示されるプロパティの後に新しいプロパティを設定します。プロパティ・カーソルによって指し示されるプロパティは、MQIMPO_INQ_FIRST または MQIMPO_INQ_NEXT オプションのいずれかによって最後に照会されたものです。

MQGET 呼び出しでメッセージ・ハンドルを再利用する場合や、MQPUT 呼び出しの MQGMO または MQPMO 構造体の *MsgHandle* フィールドにメッセージ・ハンドルが指定された場合には、プロパティ・カーソルはリセットされます。

プロパティ・カーソルがまだ確立されていない時点でこのオプションを使用する場合や、プロパティ・カーソルによって指し示されるプロパティが削除されている場合は、呼び出しは完了コード MQCC_FAILED および理由コード MQRC_PROPERTY_NOT_AVAILABLE で失敗します。

MQSMPO_APPEND_PROPERTY

階層が一致する他のすべてのプロパティの後に新規プロパティを追加します。指定された名前と一致するプロパティが1つ以上存在する場合は、そのプロパティのリストの末尾の後に新規プロパティが追加されます。

このオプションを使用すると、同じ名前プロパティのリストを作成できます。

説明されているオプションを必要としない場合、以下のオプションを使用します。

MQSMPO_NONE

指定されるオプションはありません。

これは常に入力フィールドです。このフィールドの初期値は、MQSMPO_SET_FIRST です。

ValueEncoding (MQLONG)

値が数値の場合に設定されるプロパティ値のエンコード。

これは常に入力フィールドです。このフィールドの初期値は MQENC_NATIVE です。

ValueCCSID (MQLONG)

値が文字ストリングの場合に設定されるプロパティ値の文字セット。

これは常に入力フィールドです。このフィールドの初期値は **MQCCSI_APPL** です。

MQSRO - サブスクリプション要求オプション

MQSRO 構造体を使用して、サブスクリプションの要求方法を制御するオプションをアプリケーションで指定できます。この構造は、MQSUBRQ 呼び出しの入出力パラメーターです。

可用性

MQSRO 構造は、以下のプラットフォームで利用可能です：

- ▶ **AIX** AIX
- ▶ **IBM i** IBM i
- ▶ **Linux** Linux
- ▶ **Windows** Windows
- ▶ **z/OS** z/OS

および、これらのシステムに接続された IBM MQ MQI clients。

バージョン

現在の MQSRO のバージョンは MQSRO_VERSION_1 です。

文字セットとエンコード

MQSRO 内のデータは、キュー・マネージャ **CodedCharSetId** 属性によって指定される文字セットおよび MQENC_NATIVE によって指定されるローカル・キュー・マネージャのエンコーディングでなければならない。ただし、アプリケーションが MQ MQI クライアントとして実行されている場合、構造体はクライアントの文字セットとエンコードに従っている必要があります。

フィールド

注：次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

フィールド名と説明	定数の名前	定数の初期値（存在する場合）
<u>StrucId</u> (構造識別子)	MQSRO_STRUC_ID	'SRO↵'
<u>バージョン</u> (構造バージョン番号)	MQSRO_VERSION_1	1
<u>オプション</u> (オプション)	MQSRO_NONE	0
<u>NumPubs</u> (出版物の数)	なし	0
注： 1. 記号↵は、単一のブランク文字を表します。 2. C プログラミング言語では、マクロ変数 MQSRO_DEFAULT には、表にリストされている値が含まれています。この変数を以下の方法で使用すると、構造体のフィールドに初期値を設定できます。 <pre>MQSRO MySRO = {MQSRO_DEFAULT};</pre>		

言語ごとの宣言

MQSRO の C 宣言

```
typedef struct tagMQSRO MQSRO;
struct tagMQSRO {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQLONG     Options;          /* Options that control the action of MQSUBRQ */
    MQLONG     NumPubs;          /* Number of publications sent */
    /* Ver:1 */
};
```

MQSRO の COBOL 宣言

```
** MQSRO structure
10  MQSRO.
** Structure identifier
15  MQSRO-STRUCID          PIC X(4).
** Structure version number
15  MQSRO-VERSION         PIC S9(9) BINARY.
** Options that control the action of MQSUBRQ
15  MQSRO-OPTIONS         PIC S9(9) BINARY.
** Number of publications sent
15  MQSRO-NUMPUBS         PIC S9(9) BINARY.
```

MQSRO の PL/I 宣言

```
dcl
  1 MQSRO based,
    3 StrucId      char(4),      /* Structure identifier */
    3 Version      fixed bin(31), /* Structure version number */
    3 Options      fixed bin(31), /* Options that control the action of MQSUBRQ */
    3 NumPubs      fixed bin(31); /* Number of publications sent */
```

MQSRO の高水準アセンブラー宣言

```
MQSRO          DSECT
MQSRO_STRUCID  DS    CL4    Structure identifier
MQSRO_VERSION  DS    F      Structure version number
MQSRO_OPTIONS  DS    F      Options that control the action of MQSUBRQ
MQSRO_NUMPUBS  DS    F      Number of publications sent
*
MQSRO_LENGTH   EQU    *-MQSRO
ORG            MQSRO
MQSRO_AREA     DS      CL(MQSRO_LENGTH)
```

MQSRO のフィールド詳細

MQSRO サブスクリプション要求オプション構造により、アプリケーションはサブスクリプション要求の実行方法を制御するオプションを指定できます。この構造は、MQSUBRQ 呼び出しの入出力パラメーターです。

フィールド

注: 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

フィールド名と説明	定数の名前	定数の初期値（存在する場合）
<u>StrucId</u> (構造識別子)	MQSRO_STRUC_ID	'SRO~'
<u>バージョン</u> (構造バージョン番号)	MQSRO_VERSION_1	1
<u>オプション</u> (オプション)	MQSRO_NONE	0
<u>NumPubs</u> (出版物の数)	なし	0

フィールド名と説明	定数の名前	定数の初期値（存在する場合）
注： - 記号-は、単一のブランク文字を表します。 - C プログラミング言語では、マクロ変数 MQSRO_DEFAULT には、表にリストされている値が含まれています。この変数を以下の方法で使用すると、構造体のフィールドに初期値を設定できます。 <pre>MQSRO MySRO = {MQSRO_DEFAULT};</pre>		

StrucId (MQCHAR4)

これは、サブスクリプション要求オプション構造体の構造体識別子です。常に入力フィールドです。その値は MQSRO_STRUC_ID です。

値は次のものでなければなりません。

MQSRO_STRUC_ID

サブスクリプション要求オプション構造体の識別子。

C プログラミング言語では、定数 MQSRO_STRUC_ID_ARRAY も定義されている。これは MQSRO_STRUC_ID と同じ値を持ちますが、文字列ではなく文字の配列です。

Version (MQLONG)

これは構造体のバージョン番号です。値は以下のものでなければなりません。

MQSRO_VERSION_1

バージョン 1 のサブスクリプション要求オプションの構造。

以下の定数は、現行バージョンのバージョン番号を指定しています。

MQSRO_CURRENT_VERSION

サブスクリプション要求オプションの構造の現行バージョン。

これは常に入力フィールドです。フィールドの初期値は、MQSRO_VERSION_1 です。

Options (MQLONG)

以下のオプションを 1 つ指定する必要があります。オプションは、1 つだけ指定することができます。

MQSRO_FAIL_IF QUIESCING

MQSUBRQ 呼び出しは、キュー・マネージャーが静止状態にあるときに失敗します。z/OS では、CICS または IMS アプリケーションについてこのオプションを指定すると、接続が静止状態になっている場合には MQSUBRQ 呼び出しを強制的に失敗させます。

デフォルト・オプション: 上記で説明されたオプションが必要でない場合、以下のオプションを使用しなければなりません。

MQSRO_NONE

この値は、他のオプションが指定されなかったことを示すために使用します。すべてのオプションはデフォルト値であるとみなされます。

MQSRO_NONE は、プログラムの文書化に役立ちます。このオプションは、他のオプションと組み合わせて使用するオプションではありませんが、このオプションの値はゼロと等価なので、他のオプションと組み合わせて使用しても、エラーとして検出されることはありません。

NumPubs (MQLONG)

これはアプリケーションに戻される出力フィールドで、この呼び出しの結果としてサブスクリプション・キューに送信されるパブリケーションの数を示します。この呼び出しの結果としてこの数のパブリケーシ

ョンが送信されていますが、これだけ多くのメッセージをアプリケーションが取得できるという保証はありません。非持続メッセージの場合は特にそうです。

サブスクライブされるトピックにワイルドカードが含まれていた場合は、パブリケーションが複数ある可能性があります。Hsub で表されているサブスクリプションの作成時にトピック・ストリング中にワイルドカードがない場合は、この呼び出しの結果として 1 つ以上のパブリケーションが送信されます。

MQSTS - 状況報告構造体

MQSTS 構造体は、MQSTAT コマンドからの出力パラメータです。MQSTAT コマンドは、ステータス情報を取得するために使用されます。この情報は MQSTS 構造体に戻されます。

文字セットとエンコード

MQSTS 内の文字データは、ローカルキューマネージャの文字セットで保持されます。これはキューマネージャ CodedCharSetId 属性によって指定されます。MQSTS の数値データは、ネイティブ・マシン・エンコードになっています。これは、Encoding によって与えられます。

フィールド

注: 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 562. MQSTS のフィールド		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
StrucId (構造識別子)	MQSTS_STRUC_ID	'STAT-'
バージョン (構造バージョン番号)	MQSTS_VERSION_1	1
CompCode (最初のエラーの完了コード)	MQCC_OK	0
理由 (最初のエラーの理由コード)	MQRC_NONE	0
PutSuccessCount (成功した非同期 put 呼び出しの数)	なし	0
PutWarningCount (警告が発生した非同期 put 呼び出しの数)	なし	0
PutFailureCount (失敗した非同期 put 呼び出しの数)	なし	0
ObjectType (故障したオブジェクトの種類)	MQOT_Q	1
ObjectName (故障したオブジェクトの名前)	なし	ヌル・ストリングまたはブランク
ObjectQMgrName (障害が発生したオブジェクトを所有するキューマネージャの名前)	なし	ヌル・ストリングまたはブランク
ResolvedObjectName (解決済み宛先キュー名)	なし	ヌル・ストリングまたはブランク
ResolvedQMgrName (解決済み宛先キューマネージャ名)	なし	ヌル・ストリングまたはブランク
注: Version が MQSTS_VERSION_2 より前のものである場合、以下のフィールドは無視されます。		
ObjectString (失敗したオブジェクトの長いオブジェクト名)	MQCHARV_DEFAULT	{NULL,0,0,0,-3}

表 562. MQSTS のフィールド (続き)

フィールド名と説明	定数の名前	定数の初期値（存在する場合）
SubName (失敗したサブスクリプションのサブスクリプション名)	MQCHARV_DEFAULT	{NULL,0,0,0,-3}
OpenOptions (障害に関連するオプションを開く)	なし	0
サブオプション (障害に関連するサブスクリプションオプション)	なし	0
注： - 記号-は、単一のブランク文字を表します。 - ヌル・ストリングまたはブランクの値は、C 言語ではヌル・ストリングを表し、他のプログラミング言語ではブランク文字を表します。 - C プログラミング言語では、マクロ変数 MQSTS_DEFAULT に表にリストされている値が設定されています。この変数を以下の方法で使用すると、構造体のフィールドに初期値を設定できます。 <pre>MQSTS MySTS = {MQSTS_DEFAULT};</pre>		

言語ごとの宣言

MQSTS の C 宣言

```
typedef struct tagMQSTS MQSTS;
struct tagMQSTS {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQLONG     CompCode;         /* Completion Code of first error */
    MQLONG     Reason;           /* Reason Code of first error */
    MQLONG     PutSuccessCount;   /* Number of Async calls succeeded */
    MQLONG     PutWarningCount;   /* Number of Async calls had warnings */
    MQLONG     PutFailureCount;   /* Number of Async calls had failures */
    MQLONG     ObjectType;       /* Failing object type */
    MQCHAR48    ObjectName;       /* Failing object name */
    MQCHAR48    ObjectQMgrName;   /* Failing object queue manager name */
    MQCHAR48    ResolvedObjectName; /* Resolved name of destination queue */
    MQCHAR48    ResolvedQMgrName; /* Resolved name of destination qmgr */
/* Ver:1 */
    MQCHARV     ObjectString;     /* Failing object long name */
    MQCHARV     SubName;         /* Failing subscription name */
    MQLONG     OpenOptions;      /* Failing open options */
    MQLONG     SubOptions;       /* Failing subscription options */
/* Ver:2 */
};
```

MQSTS の COBOL 宣言

```
** MQSTS structure
10  MQSTS.
** Structure identifier
15  MQSTS-STRUCID PIC X(4).
** Structure version number
15  MQSTS-VERSION PIC S9(9) BINARY.
** Completion Code of first error
15  MQSTS-COMPCODE PIC S9(9) BINARY.
** Reason Code of first error
15  MQSTS-REASON PIC S9(9) BINARY.
** Number of Async put calls succeeded
15  MQSTS-PUTSUCCESSCOUNT PIC S9(9) BINARY.
** Number of Async put calls had warnings
15  MQSTS-PUTWARNINGCOUNT PIC S9(9) BINARY.
** Number of Async put calls had failures
15  MQSTS-PUTFAILURECOUNT PIC S9(9) BINARY.
** Failing object type
15  MQSTS-OBJECTTYPE PIC S9(9) BINARY.
** Failing object name
```

```

15 MQSTS-OBJECTNAME PIC X(48).
** Failing object queue manager
15 MQSTS-OBJECTQMGRNAME PIC X(48).
** Resolved name of destination queue
15 MQSTS-RESOLVEDOBJECTNAME PIC X(48).
** Resolved name of destination qmgr
15 MQSTS-RESOLVEDQMGRNAME PIC X(48).
** Ver:1 **
** Failing object long name
15 MQSTS-OBJECTSTRING.
** Address of variable length string
20 MQSTS-OBJECTSTRING-VSPTR POINTER.
** Offset of variable length string
20 MQSTS-OBJECTSTRING-VSOFFSET PIC S9(9) BINARY.
** Size of buffer
20 MQSTS-OBJECTSTRING-VSBUFSIZE PIC S9(9) BINARY.
** Length of variable length string
20 MQSTS-OBJECTSTRING-VSLENGTH PIC S9(9) BINARY.
** CCSID of variable length string
20 MQSTS-OBJECTSTRING-VSCCSID PIC S9(9) BINARY.
** Failing subscription name
15 MQSTS-SUBNAME.
** Address of variable length string
20 MQSTS-SUBNAME-VSPTR POINTER.
** Offset of variable length string
20 MQSTS-SUBNAME-VSOFFSET PIC S9(9) BINARY.
** Size of buffer
20 MQSTS-SUBNAME-VSBUFSIZE PIC S9(9) BINARY.
** Length of variable length string
20 MQSTS-SUBNAME-VSLENGTH PIC S9(9) BINARY.
** CCSID of variable length string
20 MQSTS-SUBNAME-VSCCSID PIC S9(9) BINARY.
** Failing open options
15 MQSTS-OPENOPTIONS PIC S9(9) BINARY.
** Failing subscription options
15 MQSTS-SUBOPTIONS PIC S9(9) BINARY.
** Ver:2 **

```

MQSTS の PL/I 宣言

```

dcl
1 MQSTS based,
3 StructId          char(4),          /* Structure identifier */
3 Version           fixed bin(31),    /* Structure version number */
3 CompCode          fixed bin(31),    /* Completion code */
3 Reason            fixed bin(31),    /* Reason code */
3 PutSuccessCount   fixed bin(31),    /* Put success count */
3 PutWarningCount   fixed bin(31),    /* Put warning count */
3 PutFailureCount   fixed bin(31),    /* Put failure count */
3 ObjectType        fixed bin(31),    /* Object type */
3 ObjectName        char(48),         /* Object name */
3 ObjectQmgrName    char(48),         /* Object queue manager */
3 ResolvedObjectName char(48),        /* Resolved Object name */
3 ResolvedQmgrName  char(48);        /* Resolved Object queue manager */
/* Ver:1 */
3 ObjectString,     /* Failing object long name */
5 VSPtr pointer,    /* Address of variable length string */
5 VSOffset fixed bin(31), /* Offset of variable length string */
5 VSBufSize fixed bin(31), /* Size of buffer */
5 VSLength fixed bin(31), /* Length of variable length string */
5 VSCCSID fixed bin(31); /* CCSID of variable length string */
3 SubName,          /* Failing subscription name */
5 VSPtr pointer,    /* Address of variable length string */
5 VSOffset fixed bin(31), /* Offset of variable length string */
5 VSBufSize fixed bin(31), /* Size of buffer */
5 VSLength fixed bin(31), /* Length of variable length string */
5 VSCCSID fixed bin(31); /* CCSID of variable length string */
3 OpenOptions fixed bin(31), /* Failing open options */
3 SubOptions fixed bin(31); /* Failing subscription options */
/* Ver:2 */

```

MQSTS の高水準アセンブラ宣言

MQSTS	DSECT		
MQSTS_STRUCID	DS	CL4	Structure identifier
MQSTS_VERSION	DS	F	Structure version number
MQSTS_COMPCODE	DS	F	Completion code
MQSTS_REASON	DS	F	Reason code
MQSTS_PUTSUCCESSCOUNT	DS	F	Success count


```

MQSTS_PUTWARNINGCOUNT    DS    F    Warning count
MQSTS_PUTFAILURECOUNT    DS    F    Failure count
MQSTS_OBJTYPE              DS    F    Object type
MQSTS_OBJNAME              DS    CL48  Object name
MQSTS_OBJQMGR              DS    CL48  Object queue manager
MQSTS_ROBJNAME             DS    CL48  Resolved object name
MQSTS_ROBJQMGR             DS    CL48  Resolved object queue manager
MQSTS_OBJECTSTRING         DS    0F    Force fullword alignment
MQSTS_OBJECTSTRING_VSPTR   DS    A    Address of variable length string
MQSTS_OBJECTSTRING_VSOFFSET DS    F    Offset of variable length string
MQSTS_OBJECTSTRING_VSBUFFSIZE DS    F    Size of buffer
MQSTS_OBJECTSTRING_VSLNGTH DS    F    Length of variable length string
MQSTS_OBJECTSTRING_VSCCSID DS    F    CCSID of variable length string
MQSTS_OBJECTSTRING_LENGTH EQU    *-MQSTS_OBJECTSTRING
                                ORG    MQSTS_OBJECTSTRING
MQSTS_OBJECTSTRING_AREA    DS    CL(MQSTS_OBJECTSTRING_LENGTH)
*
MQSTS_SUBNAME              DS    0F    Force fullword alignment
MQSTS_SUBNAME_VSPTR        DS    A    Address of variable length string
MQSTS_SUBNAME_VSOFFSET     DS    F    Offset of variable length string
MQSTS_SUBNAME_VSBUFFSIZE   DS    F    Size of buffer
MQSTS_SUBNAME_VSLNGTH      DS    F    Length of variable length string
MQSTS_SUBNAME_VSCCSID      DS    F    CCSID of variable length string
MQSTS_SUBNAME_LENGTH       EQ    *-MQSTS_SUBNAME
                                ORG    MQSTS_SUBNAME
MQSTS_SUBNAME_AREA         DS    CL(MQSTS_SUBNAME_LENGTH)
*
MQSTS_OPENOPTIONS          DS    F    Failing open options
MQSTS_SUBOPTIONS           DS    F    Failing subscription option
MQSTS_LENGTH               EQU    *-MQSTS
                                ORG    MQSTS
MQSTS_AREA                 DS    CL(MQSTS_LENGTH)

```

関連資料

777 ページの『MQSTAT - 状況情報の取り出し』

MQSTAT 呼び出しを使用して、状況情報を取り出します。戻される状況情報のタイプは、呼び出しで指定されている Type 値によって判別されます。

MQSTS のフィールド詳細

MQSTS ステータス報告構造は、MQSTAT コマンドからの出力パラメータです。MQSTAT コマンドは、ステータス情報を取得するために使用されます。この情報は MQSTS 構造体に戻されます。

フィールド

注: 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 563. MQSTS のフィールド		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
StrucId (構造識別子)	MQSTS_STRUC_ID	'STAT-'
バージョン (構造バージョン番号)	MQSTS_VERSION_1	1
CompCode (最初のエラーの完了コード)	MQCC_OK	0
理由 (最初のエラーの理由コード)	MQRC_NONE	0
PutSuccessCount (成功した非同期 put 呼び出しの数)	なし	0
PutWarningCount (警告が発生した非同期 put 呼び出しの数)	なし	0
PutFailureCount (失敗した非同期 put 呼び出しの数)	なし	0
ObjectType (故障したオブジェクトの種類)	MQOT_Q	1

表 563. MQSTS のフィールド (続き)		
フィールド名と説明	定数の名前	定数の初期値 (存在する場合)
<u>ObjectName</u> (故障したオブジェクトの名前)	なし	ヌル・ストリングまたはブランク
<u>ObjectQMgrName</u> (障害が発生したオブジェクトを所有するキューマネージャの名前)	なし	ヌル・ストリングまたはブランク
<u>ResolvedObjectName</u> (解決済み宛先キュー名)	なし	ヌル・ストリングまたはブランク
<u>ResolvedQMgrName</u> (解決済み宛先キューマネージャ名)	なし	ヌル・ストリングまたはブランク
注: Version が MQSTS_VERSION_2 より前のものである場合、以下のフィールドは無視されます。		
<u>ObjectString</u> (失敗したオブジェクトの長いオブジェクト名)	MQCHARV_DEFAULT	{NULL,0,0,0,-3}
<u>SubName</u> (失敗したサブスクリプションのサブスクリプション名)	MQCHARV_DEFAULT	{NULL,0,0,0,-3}
<u>OpenOptions</u> (障害に関連するオプションを開く)	なし	0
<u>サブオプション</u> (障害に関連するサブスクリプションオプション)	なし	0
注: 1. 記号-は、単一のブランク文字を表します。 2. ヌル・ストリングまたはブランクの値は、C 言語ではヌル・ストリングを表し、他のプログラミング言語ではブランク文字を表します。 3. C プログラミング言語では、マクロ変数 MQSTS_DEFAULT に表にリストされている値が設定されています。この変数を以下の方法で使用すると、構造体のフィールドに初期値を設定できます。 <pre>MQSTS MySTS = {MQSTS_DEFAULT};</pre>		

StrucId (MQCHAR4)

これはステータス報告構造体の構造体識別子です。常に入力フィールドです。その値は MQSTS_STRUC_ID です。

値は次のものでなければなりません。

MQSTS_STRUC_ID

ステータス報告構造体の識別子。

C プログラミング言語の場合、定数 MQSTS_STRUC_ID_ARRAY も定義されます。これはと同じ値 MQSTS_STRUC_ID を持ちますが、文字列ではなく文字の配列です。

Version (MQLONG)

構造体のバージョン番号。

値は、以下のいずれかでなければなりません。

MQSTS_VERSION_1

バージョン 1 の状況報告構造体。

MQSTS_VERSION_2

バージョン 2 の状況報告構造体。

以下の定数は、現行バージョンのバージョン番号を指定しています。

MQSTS_CURRENT_VERSION

状況報告構造体の現行バージョン。現行バージョンは MQSTS_VERSION_2 です。

Version は、常に入力フィールドです。初期値は、MQSTS_VERSION_1 です。

CompCode (MQLONG)

報告対象の操作の完了コード。

CompCode の解釈は、MQSTAT **Type** パラメーターの値に依存します。

MQSTAT_TYPE_ASYNC_ERROR

これは、ObjectName で指定されたオブジェクトに対する以前の非同期 PUT 操作によって生成された完了コードです。

MQSTAT_TYPE_RECONNECTION

これは、再接続中であるか、または再接続に失敗した場合に、その再接続の開始の原因となった完了コードです。

接続が現在確立されている場合、値は MQCC_OK になります。

MQSTAT_TYPE_RECONNECTION_ERROR

これは、再接続に失敗した場合に、その再接続の失敗の原因となった完了コードです。

接続が現在確立されているかまたは再接続の途中である場合、値は MQCC_OK になります。

CompCode は、常に出力フィールドです。初期値は、MQCC_OK です。

理由 (MQLONG)

報告対象の操作の理由コード。

Reason の解釈は、MQSTAT **Type** パラメーターの値に依存します。

MQSTAT_TYPE_ASYNC_ERROR

これは、ObjectName で指定されたオブジェクトに対する前の非同期 PUT 操作によって生成された理由コードです。

MQSTAT_TYPE_RECONNECTION

接続が再接続されたり、再接続に失敗したりした場合、この理由コードが再接続開始の原因となっています。

接続が現在確立されている場合、値は MQRC_NONE になります。

MQSTAT_TYPE_RECONNECTION_ERROR

接続が再接続に失敗した場合、この理由コードが再接続失敗の原因となっています。

接続が現在確立されているかまたは再接続の途中である場合、値は MQRC_NONE になります。

Reason は出力フィールドです。初期値は、MQRC_NONE です。

PutSuccessCount (MQLONG)

成功した非同期 PUT 操作の数。

PutSuccessCount の値は、MQSTAT **Type** パラメーターの値に依存します。

MQSTAT_TYPE_ASYNC_ERROR

MQSTS 構造体で指定されたオブジェクトに対して行われた非同期 PUT 操作のうち、MQCC_OK で完了したものの数。

MQSTAT_TYPE_RECONNECTION

ゼロ。

MQSTAT_TYPE_RECONNECTION_ERROR

ゼロ。

PutSuccessCount は出力フィールドです。初期値はゼロです。

PutWarningCount (MQLONG)

警告で終了した非同期 PUT 操作の数。

PutWarningCount の値は、MQSTAT **Type** パラメーターの値に依存します。

MQSTAT_TYPE_ASYNC_ERROR

MQSTS 構造体で指定されたオブジェクトに対して行われた非同期 PUT 操作のうち、MQCC_WARNING で完了したものの数。

MQSTAT_TYPE_RECONNECTION

ゼロ。

MQSTAT_TYPE_RECONNECTION_ERROR

ゼロ。

PutWarningCount は出力フィールドです。初期値はゼロです。

PutFailureCount (MQLONG)

失敗した非同期 PUT 操作の数。

PutFailureCount の値は、MQSTAT **Type** パラメーターの値に依存します。

MQSTAT_TYPE_ASYNC_ERROR

MQSTS 構造体で指定されたオブジェクトに対して行われた非同期 PUT 操作のうち、MQCC_FAILED で完了したものの数。

MQSTAT_TYPE_RECONNECTION

ゼロ。

MQSTAT_TYPE_RECONNECTION_ERROR

ゼロ。

PutFailureCount は出力フィールドです。初期値はゼロです。

ObjectType (MQLONG)

報告対象の *ObjectName* で指定されたオブジェクトのタイプ。

ObjectType に指定できる値を、[103 ページの『MQOT_*\(オブジェクト・タイプおよび拡張オブジェクト・タイプ\)』](#)にリストしています。

ObjectType は出力フィールドです。初期値は、MQOT_Q です。

ObjectName (MQCHAR48)

報告対象のオブジェクトの名前。

ObjectName の解釈は、MQSTAT **Type** パラメーターの値に依存します。

MQSTAT_TYPE_ASYNC_ERROR

これは、PUT 操作で使用されるキューまたはトピックの名前です。失敗した場合は、MQSTS 構造体の *CompCode* および *Reason* フィールドで報告されます。

MQSTAT_TYPE_RECONNECTION

接続が再接続されている場合、これはその接続と関連付けられたキュー・マネージャーの名前です。

MQSTAT_TYPE_RECONNECTION_ERROR

接続が再接続に失敗した場合、これは再接続が失敗する原因となったオブジェクトの名前です。失敗の理由は、MQSTS 構造体の *CompCode* および *Reason* フィールドで報告されます。

ObjectName は出力フィールドです。初期値は、C 言語ではヌル・ストリングであり、他のプログラミング言語では 48 桁のブランク文字です。

ObjectQmgrName (MQCHAR48)

報告対象のキュー・マネージャーの名前。

ObjectQMgrName の解釈は、MQSTAT **Type** パラメーターの値に依存します。

MQSTAT_TYPE_ASYNC_ERROR

これは、オブジェクト *ObjectName* が定義されているキューマネージャの名前です。最初のヌル文字またはフィールドの終わりまで名前をすべてブランクにすると、アプリケーションが接続されているキュー・マネージャー (ローカル・キュー・マネージャー) を指定したと見なされます。

MQSTAT_TYPE_RECONNECTION



ObjectQMgrName フィールドには、再接続が要求されているキュー・マネージャーの名前が含まれます。キュー・マネージャーが指定されていない場合は空白になります。可能な場合には、クライアントはその名前のキュー・マネージャーへの再接続を試行します。



ブランク。

MQSTAT_TYPE_RECONNECTION_ERROR

接続が再接続に失敗した場合、これは再接続が失敗する原因となったオブジェクトの名前です。失敗の理由は、MQSTS 構造体の *CompCode* および *Reason* フィールドで報告されます。

ObjectQMgrName は出力フィールドです。値は、C 言語ではヌル・ストリングであり、他のプログラミング言語では 48 桁のブランク文字です。

ResolvedObjectName (MQCHAR48)

ローカル・キュー・マネージャーが名前を解決した後の、*ObjectName* で指定されるオブジェクトの名前。

ResolvedObjectName の解釈は、MQSTAT **Type** パラメーターの値に依存します。

MQSTAT_TYPE_ASYNC_ERROR

ResolvedObjectName は、ローカル・キュー・マネージャーが名前を解決した後の、*ObjectName* で指定されるオブジェクトの名前です。返される名前は、によって識別されるキュー・マネージャ上に *ResolvedQMgrName* 存在するオブジェクトの名前です。

MQSTAT_TYPE_RECONNECTION

ブランク。

MQSTAT_TYPE_RECONNECTION_ERROR

ブランク。

ResolvedObjectName は出力フィールドです。初期値は、C 言語ではヌル・ストリングであり、他のプログラミング言語では 48 桁のブランク文字です。

ResolvedQmgrName (MQCHAR48)

ローカル・キュー・マネージャーが名前を解決した後の宛先キュー・マネージャーの名前。

ResolvedQMgrName の解釈は、MQSTAT **Type** パラメーターの値に依存します。

MQSTAT_TYPE_ASYNC_ERROR

ResolvedQMgrName は、ローカル・キュー・マネージャーが名前を解決した後の宛先キュー・マネージャーの名前です。戻される名前は、*ResolvedObjectName* によって識別されたオブジェクトを所有する、キュー・マネージャーの名前です。*ResolvedQMgrName* は、ローカル・キュー・マネージャーの名前にすることができます。

MQSTAT_TYPE_RECONNECTION

ブランク。

MQSTAT_TYPE_RECONNECTION_ERROR

ブランク。

ResolvedQMgrName は、常に出力フィールドです。初期値は、C 言語ではヌル・ストリングであり、他のプログラミング言語では 48 桁のブランク文字です。

ObjectString (MQCHARV)

報告対象の失敗オブジェクトの長いオブジェクト名。現行では、MQSTS のバージョン 2 以上のみ。

ObjectString の解釈は、MQSTAT **Type** パラメーターの値に依存します。

MQSTAT_TYPE_ASYNC_ERROR

これは、失敗した MQPUT 操作で使用されたキューまたはトピックの長いオブジェクト名です。

MQSTAT_TYPE_RECONNECTION

長さゼロのストリング

MQSTAT_TYPE_RECONNECTION_ERROR

これは、再接続が失敗する原因となったオブジェクトの長いオブジェクト名です。

ObjectString は出力フィールドです。その初期値は、長さゼロのストリングです。

SubName (MQCHARV)

失敗しているサブスクリプションの名前。現行では、MQSTS のバージョン 2 以上のみ。

SubName の解釈は、MQSTAT **Type** パラメーターの値に依存します。

MQSTAT_TYPE_ASYNC_ERROR

長さゼロのストリング。

MQSTAT_TYPE_RECONNECTION

長さゼロのストリング。

MQSTAT_TYPE_RECONNECTION_ERROR

再接続が失敗する原因となったサブスクリプションの名前。サブスクリプション名が使用可能でない場合、または失敗がサブスクリプションに関連していない場合には、これは長さゼロのストリングです。

SubName は出力フィールドです。その初期値は、長さゼロのストリングです。

OpenOptions (MQLONG)

報告対象のオブジェクトを開くために使用される OpenOptions。現行では、MQSTS のバージョン 2 以上のみ。

OpenOptions の値は、MQSTAT **Type** パラメーターの値に依存します。

MQSTAT_TYPE_ASYNC_ERROR

ゼロ。

MQSTAT_TYPE_RECONNECTION

ゼロ。

MQSTAT_TYPE_RECONNECTION_ERROR

障害の発生時に使用されていた OpenOptions。失敗の理由は、MQSTS 構造体の *CompCode* および *Reason* フィールドで報告されます。

OpenOptions は出力フィールドです。初期値はゼロです。

SubOptions (MQLONG)

失敗したサブスクリプションを開くために使用された SubOptions。現行では、MQSTS のバージョン 2 以上のみ。

SubOptions の解釈は、MQSTAT **Type** パラメーターの値に依存します。

MQSTAT_TYPE_ASYNC_ERROR

ゼロ。

MQSTAT_TYPE_RECONNECTION

ゼロ。

MQSTAT_TYPE_RECONNECTION_ERROR

障害の発生時に使用されていた SubOptions。障害がトピックへのサブスクリプションに関連していない場合、返される値はゼロです。

SubOptions は出力フィールドです。初期値はゼロです。

MQTM - トリガー・メッセージ

MQTM 構造体は、キュー・トリガー・イベントが発生した時に、キュー・マネージャーによりトリガー・モニター・アプリケーションに送信されるトリガー・メッセージ内のデータについて記述します。この構造体は、IBM MQ トリガー・モニター・インターフェース (TMI) の一部です。TMI は、IBM MQ フレームワーク・インターフェースに含まれています。

形式名

MQFMT_TRIGGER。

文字セットとエンコード

MQTM 内の文字データは、MQTM を生成するキューマネージャの文字セットで表されます。MQTM の数値データは、MQTM を生成するキュー・マネージャのマシン・エンコードにあります。

MQTM の文字セットとエンコーディングは、以下の `` および *Encoding* `` *CodedCharSetId* フィールドによって指定される：

- MQMD (MQTM 構造体がメッセージ・データの開始点にある場合)
- MQTM 構造体に先行するヘッダー構造体 (その他のすべての場合)

使用法

トリガー監視アプリケーションは、トリガーメッセージ内の情報の一部またはすべてを、トリガー監視アプリケーションが起動するアプリケーションに渡す必要がある場合があります。開始済みアプリケーションに必要な情報には、*QName*、*TriggerData*、および *UserData* があります。トリガー・モニター・アプリケーションでは、起動したアプリケーションに MQTM 構造体を直接渡すだけでなく、MQTMC2 構造体を渡すこともできます。どちらを渡すかは、起動したアプリケーション側の環境および条件で許可されるもので決まります。MQTMC2 の詳細については、595 ページの『MQTMC2 - トリガー・メッセージ 2 (文字フォーマット)』を参照してください。

-  **z/OS** z/OS では、CKTI トランザクションを使用して開始される MQAT_CICS アプリケーションの場合、トリガー・メッセージ構造体 MQTM 全体が開始されるトランザクションで使用できます。EXEC CICS RETRIEVE コマンドを使用して、情報を取り出します。
-  **IBM i** IBM i では、IBM MQ が提供するトリガー・モニター・アプリケーションが、MQTMC2 構造体を開始済みアプリケーションに渡します。

トリガーの使用については、[トリガーによる IBM MQ アプリケーションの開始](#)を参照してください。

フィールド

注：次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 564. MQTM における MQTM のフィールド		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
<u>StrucId</u> (構造識別子)	MQTM_STRUC_ID	'TM-1'
<u>バージョン</u> (構造バージョン番号)	MQTM_VERSION_1	1
<u>QName</u> (トリガーされたキューの名前)	なし	ヌル・ストリングまたはブランク
<u>ProcessName</u> (プロセスオブジェクト名)	なし	ヌル・ストリングまたはブランク
<u>TriggerData</u> (トリガーデータ)	なし	ヌル・ストリングまたはブランク
<u>ApplType</u> (アプリケーションタイプ)	なし	0
<u>ApplId</u> (アプリケーション識別子)	なし	ヌル・ストリングまたはブランク
<u>EnvData</u> (環境データ)	なし	ヌル・ストリングまたはブランク
<u>UserData</u> (ユーザーデータ)	なし	ヌル・ストリングまたはブランク

表 564. MQTM における MQTM のフィールド (続き)

フィールド名と説明	定数の名前	定数の初期値 (存在する場合)
注: - 記号-は、単一のブランク文字を表します。 - ヌル・ストリングまたはブランクの値は、C 言語ではヌル・ストリングを表し、他のプログラミング言語ではブランク文字を表します。 - C プログラミング言語では、マクロ変数 MQTM_DEFAULT には、表にリストされている値が含まれています。このマクロ変数を以下の方法で使用して、構造体のフィールドに初期値を設定します。 <pre>MQTM MyTM = {MQTM_DEFAULT};</pre>		

言語ごとの宣言

MQTM の C 宣言

```
typedef struct tagMQTM MQTM;
struct tagMQTM {
    MQCHAR4    StrucId;        /* Structure identifier */
    MQLONG     Version;        /* Structure version number */
    MQCHAR48   QName;          /* Name of triggered queue */
    MQCHAR48   ProcessName;    /* Name of process object */
    MQCHAR64   TriggerData;    /* Trigger data */
    MQLONG     ApplType;       /* Application type */
    MQCHAR256  ApplId;         /* Application identifier */
    MQCHAR128  EnvData;        /* Environment data */
    MQCHAR128  UserData;       /* User data */
};
```

MQTM の COBOL 宣言

```
**      MQTM structure
10 MQTM.
**      Structure identifier
15 MQTM-STRUCID      PIC X(4).
**      Structure version number
15 MQTM-VERSION     PIC S9(9) BINARY.
**      Name of triggered queue
15 MQTM-QNAME       PIC X(48).
**      Name of process object
15 MQTM-PROCESSNAME PIC X(48).
**      Trigger data
15 MQTM-TRIGGERDATA PIC X(64).
**      Application type
15 MQTM-APPLTYPE     PIC S9(9) BINARY.
**      Application identifier
15 MQTM-APPLID       PIC X(256).
**      Environment data
15 MQTM-ENVDATA      PIC X(128).
**      User data
15 MQTM-USERDATA     PIC X(128).
```

MQTM の PL/I 宣言

```
dcl
1 MQTM based,
3 StrucId      char(4),          /* Structure identifier */
3 Version      fixed bin(31),    /* Structure version number */
3 QName        char(48),         /* Name of triggered queue */
3 ProcessName  char(48),         /* Name of process object */
3 TriggerData  char(64),         /* Trigger data */
3 ApplType     fixed bin(31),    /* Application type */
3 ApplId       char(256),        /* Application identifier */
3 EnvData      char(128),        /* Environment data */
3 UserData     char(128);        /* User data */
```

MQTM の高水準アセンブラ宣言

```

MQTM                DSECT
MQTM_STRUCID        DS    CL4      Structure identifier
MQTM_VERSION        DS    F        Structure version number
MQTM_QNAME          DS    CL48     Name of triggered queue
MQTM_PROCESSNAME    DS    CL48     Name of process object
MQTM_TRIGGERDATA    DS    CL64     Trigger data
MQTM_APPLTYPE       DS    F        Application type
MQTM_APPLID         DS    CL256    Application identifier
MQTM_ENVDATA        DS    CL128    Environment data
MQTM_USERDATA       DS    CL128    User data
*
MQTM_LENGTH         EQU    *-MQTM
MQTM_AREA           DS    CL(MQTM_LENGTH)

```

MQTM の Visual Basic 宣言

```

Type MQTM
    StrucId      As String*4      'Structure identifier'
    Version      As Long          'Structure version number'
    QName        As String*48     'Name of triggered queue'
    ProcessName  As String*48     'Name of process object'
    TriggerData  As String*64     'Trigger data'
    ApplType     As Long          'Application type'
    ApplId       As String*256    'Application identifier'
    EnvData      As String*128    'Environment data'
    UserData     As String*128    'User data'
End Type

```

トリガー・メッセージの MQMD

表 565. キュー・マネージャーによって生成されるトリガー・メッセージの MQMD のフィールドの設定

MQMD のフィールド	使用される値
<i>StrucId</i>	MQMD_STRUC_ID
<i>Version</i>	MQMD_VERSION_1
<i>Report</i>	MQRO_NONE
<i>MsgType</i>	MQMT_DATAGRAM
<i>Expiry</i>	MQEI_UNLIMITED
<i>Feedback</i>	MQFB_NONE
<i>Encoding</i>	MQENC_NATIVE
<i>CodedCharSetId</i>	キュー・マネージャーの CodedCharSetId 属性
<i>Format</i>	MQFMT_TRIGGER
<i>Priority</i>	開始キューの DefPriority 属性
<i>Persistence</i>	MQPER_NOT_PERSISTENT
<i>MsgId</i>	固有の値
<i>CorrelId</i>	MQCI_NONE
<i>BackoutCount</i>	0
<i>ReplyToQ</i>	ブランク
<i>ReplyToQMgr</i>	キュー・マネージャーの名前。
<i>UserIdentifier</i>	ブランク
<i>AccountingToken</i>	MQACT_NONE
<i>ApplIdentityData</i>	ブランク

表 565. キュー・マネージャーによって生成されるトリガー・メッセージの MQMD のフィールドの設定 (続き)

MQMD のフィールド	使用される値
<i>PutApplType</i>	MQAT_QMGR またはメッセージ・チャネル・エージェントに適切なもの
<i>PutAppName</i>	キュー・マネージャー名の最初の 28 バイト
<i>PutDate</i>	トリガー・メッセージが送信された日付
<i>PutTime</i>	トリガー・メッセージが送信された時刻
<i>ApplOriginData</i>	ブランク

トリガー・メッセージを生成するアプリケーションでは、以下のものを除いて、同様の値を設定することをお勧めします。

- *Priority* フィールドは、MQPRI_PRIORITY_AS_Q_DEF に設定することができます (キュー・マネージャーが、メッセージを書き込むときに、このフィールドを開始キューのデフォルト優先順位に変更します)。
- *ReplyToQMGr* フィールドは、ブランクに設定することができます (メッセージを書き込むときに、キュー・マネージャーが、このフィールドをローカル・キュー・マネージャーの名前に変更します)。
- コンテキスト・フィールドは、アプリケーションに対して適切に設定しなければなりません。

MQTM のフィールド詳細

MQTM トリガーメッセージ構造は、キューに対してトリガーイベントが発生した際に、キュー管理者がトリガー監視アプリケーションに送信するトリガーメッセージ内のデータを記述する。

フィールド

注: 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 566. MQTM における MQTM のフィールド		
フィールド名と説明	定数の名前	定数の初期値 (存在する場合)
<u>StrucId</u> (構造識別子)	MQTM_STRUC_ID	'TM-'
<u>バージョン</u> (構造バージョン番号)	MQTM_VERSION_1	1
<u>QName</u> (トリガーされたキューの名前)	なし	ヌル・ストリングまたはブランク
<u>ProcessName</u> (プロセスオブジェクト名)	なし	ヌル・ストリングまたはブランク
<u>TriggerData</u> (トリガーデータ)	なし	ヌル・ストリングまたはブランク
<u>ApplType</u> (アプリケーションタイプ)	なし	0
<u>AppId</u> (アプリケーション識別子)	なし	ヌル・ストリングまたはブランク
<u>EnvData</u> (環境データ)	なし	ヌル・ストリングまたはブランク
<u>UserData</u> (ユーザーデータ)	なし	ヌル・ストリングまたはブランク

表 566. MQTM における MQTM のフィールド (続き)

フィールド名と説明	定数の名前	定数の初期値 (存在する場合)
注: 1. 記号-は、単一のブランク文字を表します。 2. ヌル・ストリングまたはブランクの値は、C 言語ではヌル・ストリングを表し、他のプログラミング言語ではブランク文字を表します。 3. C プログラミング言語では、マクロ変数 MQTM_DEFAULT には、表にリストされている値が含まれています。このマクロ変数を以下の方法で使用して、構造体のフィールドに初期値を設定します。 <pre>MQTM MyTM = {MQTM_DEFAULT};</pre>		

StrucId (MQCHAR4)

これはトリガーメッセージ構造の構造識別子です。常に入力フィールドです。その値は MQTM_STRUC_ID です。

値は次のものでなければなりません。

MQTM_STRUC_ID

トリガーメッセージ構造体の識別子。

C プログラミング言語では、定数 MQTM_STRUC_ID_ARRAY も定義されている。これは MQTM_STRUC_ID と同じ値を持ちますが、文字列ではなく文字の配列です。

Version (MQLONG)

これは構造体のバージョン番号です。値は次のものでなければなりません。

MQTM_VERSION_1

トリガー・メッセージ構造体のバージョン番号。

以下の定数は、現行バージョンのバージョン番号を指定しています。

MQTM_CURRENT_VERSION

トリガー・メッセージ構造体の現行バージョン。

フィールドの初期値は、MQTM_VERSION_1 です。

QName (MQCHAR48)

これは、トリガー・イベントが発生したキューの名前であり、トリガー・モニター・アプリケーションによって開始されたアプリケーションで使用されます。キュー・マネージャーは、起動されたキューの **QName** 属性の値でこのフィールドを初期設定します。この属性の詳細については、[830 ページの『キューの属性』](#)を参照してください。

定義済みフィールド長より短い名前は、右側がブランクで埋め込まれます。ヌル文字で終了することはありません。

このフィールドの長さは MQ_Q_NAME_LENGTH によって指定されます。このフィールドの初期値は、C 言語ではヌル・ストリングであり、他のプログラミング言語では 48 桁のブランク文字です。

ProcessName (MQCHAR48)

これは、起動されたキューに指定されるキュー・マネージャーのプロセス・オブジェクトの名前であり、トリガー・メッセージを受け取るトリガー・モニター・アプリケーションで使用します。キュー・マネージャーは、**QName** フィールドによって識別されるキューの **ProcessName** 属性の値でこのフィールドを初期設定します。この属性の詳細については、[830 ページの『キューの属性』](#)を参照してください。

定義済みフィールド長より短い名前は、常に右側がブランクで埋め込まれます。ヌル文字で終了することはありません。

このフィールドの長さは MQ_PROCESS_NAME_LENGTH によって指定されます。このフィールドの初期値は、C 言語ではヌル・ストリングであり、他のプログラミング言語では 48 桁のブランク文字です。

TriggerData (MQCHAR64)

これは、トリガー・メッセージを受け取るトリガー・モニター・アプリケーションで使用する自由形式のデータです。キューマネージャはこのフィールドを、フィールド *QName* で識別されるキューの属性 **TriggerData** 値で初期化する。この属性の詳細については 830 ページの『[キューの属性](#)』を参照のこと。このデータの内容は、キュー・マネージャーにとっては意味のないものです。

z/OS では、CKTI トランザクションを使用して始動される CICS アプリケーションの場合、この情報は使用されません。

このフィールドの長さは MQ_TRIGGER_DATA_LENGTH によって指定されます。このフィールドの初期値は、C 言語ではヌル・ストリングですが、その他のプログラミング言語では 64 桁のブランク文字です。

ApplType (MQLONG)

これは、開始するプログラムの性質を識別するもので、トリガー・メッセージを受け取るトリガー・モニター・アプリケーションで使用されます。キューマネージャはこのフィールドを、フィールド *ProcessName* で識別されるプロセスオブジェクトの属性 **ApplType** の値で初期化する。この属性の詳細については 866 ページの『[プロセス定義の属性](#)』を参照のこと。このデータの内容は、キュー・マネージャーにとっては意味のないものです。

ApplType は、以下の標準値のいずれかにすることができます。ユーザー定義のタイプも使用できますが、MQAT_USER_FIRST から MQAT_USER_LAST の範囲の値に限定する必要があります。

MQAT_AIX

AIX アプリケーション (MQAT_UNIX と同じ値)。

MQAT_BATCH

バッチ・アプリケーション

MQAT_BROKER

ブローカー・アプリケーション

MQAT_CICS

CICS トランザクション。

MQAT_CICS_BRIDGE

CICS bridge アプリケーション。

MQAT_CICS_VSE

CICS/VSE トランザクション。

MQAT_DOS

PC DOS 上の IBM MQ MQI client アプリケーション。

MQAT_IMS

IMS アプリケーション。

MQAT_IMS_BRIDGE

IMSブリッジ・アプリケーション。

MQAT_JAVA

Java アプリケーション。

MQAT_MVS

MVS または TSO アプリケーション (MQAT_ZOS と同じ値)。

MQAT_NOTES_AGENT

Lotus Notes エージェント・アプリケーション。

MQAT_OS390

OS/390 アプリケーション (MQAT_ZOS と同じ値)。

MQAT_OS400

IBM i アプリケーション。

MQAT_RRS_BATCH

RRS バッチ・アプリケーション。

MQAT_UNIX

UNIX アプリケーション。

MQAT_UNKNOWN

不明なアプリケーション・タイプ。

MQAT_USER

ユーザー定義のアプリケーション・タイプ。

MQAT_VOS

Stratus VOS アプリケーション。

MQAT_WINDOWS

16 ビットの Windows アプリケーション。

MQAT_WINDOWS_NT

32 ビットの Windows アプリケーション。

MQAT_WLM

z/OS ワークロード・マネージャー・アプリケーション。

MQAT_XCF

XCF。

MQAT_ZOS

z/OS アプリケーション。

MQAT_USER_FIRST

ユーザー定義のアプリケーション・タイプの最低値。

MQAT_USER_LAST

ユーザー定義のアプリケーション・タイプの最高値。

このフィールドの初期値は 0 です。

ApplId (MQCHAR256)

これは、開始されるアプリケーションを識別する文字ストリングであり、トリガー・メッセージを受け取るトリガー・モニター・アプリケーションで使用されます。キューマネージャはこのフィールドを、フィールド *ProcessName* で識別されるプロセスオブジェクトの属性 **ApplId** の値で初期化する。この属性の詳細については [866 ページの『プロセス定義の属性』](#) を参照のこと。このデータの内容は、キュー・マネージャにとっては意味のないものです。

ApplId の意味は、トリガー・モニター・アプリケーションによって決まります。IBM MQ によって提供されるトリガー・モニターでは、*ApplId* を実行可能プログラムの名前にする必要があります。以下の注意事項は、指定している特定の環境に適用されます。

- z/OS では、*ApplId* は次のようになります。
 - CICS トリガー・モニター・トランザクション CKTI を使用して始動されるアプリケーションの場合、CICS トランザクション ID です。
 - IMS トリガー・モニター CSQQTRMN を使用して始動されるアプリケーションの場合、IMS トランザクション ID です。
- Windows システムの場合、プログラム名の前にドライブとディレクトリー・パスを付けることができます。
- IBM i の場合、プログラム名の前にライブラリー名と / 文字を付けることができます。
- AIX and Linux の場合、プログラム名の前にディレクトリー・パスを付けることができます。

このフィールドの長さは MQ_PROCESS_APPL_ID_LENGTH によって指定されます。このフィールドの初期値は、C 言語ではヌル・ストリングですが、その他のプログラミング言語では 256 桁の空白文字です。

EnvData (MQCHAR128)

これは、開始されるアプリケーションに関連する環境関連情報が入っている文字ストリングであり、トリガー・メッセージを受け取るトリガー・モニター・アプリケーションで使用されます。キューマネージャはこのフィールドを、フィールド *ProcessName* で識別されるプロセスオブジェクトの属性 **EnvData** の値で初期化する。この属性の詳細については 866 ページの『プロセス定義の属性』を参照のこと。このデータの内容は、キュー・マネージャーにとっては意味のないものです。

z/OS では、CKTI トランザクションを使用して始動される CICS アプリケーション、または CSQQTRMN トランザクションを使用して始動される IMS アプリケーションの場合、この情報は使用されません。

このフィールドの長さは MQ_PROCESS_ENV_DATA_LENGTH によって指定されます。このフィールドの初期値は、C 言語ではヌル・ストリングですが、その他のプログラミング言語では 128 桁の空白文字です。

UserData (MQCHAR128)

これは、開始されるアプリケーションに関連するユーザー情報が入っている文字ストリングであり、トリガー・メッセージを受け取るトリガー・モニター・アプリケーションで使用されます。キューマネージャはこのフィールドを、フィールド *ProcessName* で識別されるプロセスオブジェクトの属性 **UserData** の値で初期化する。この属性の詳細については 866 ページの『プロセス定義の属性』を参照のこと。このデータの内容は、キュー・マネージャーにとっては意味のないものです。

Microsoft Windows では、プロセス定義が **runmqtrm** に渡される場合、文字ストリングに二重引用符を含めてはなりません。

フィールドの長さは、MQ_PROCESS_USER_DATA_LENGTH で指定します。このフィールドの初期値は、C 言語ではヌル・ストリングですが、その他のプログラミング言語では 128 桁の空白文字です。

MQTMC2 - トリガー・メッセージ 2 (文字フォーマット)

トリガー監視アプリケーションが開始キューからトリガーメッセージ (MQTM) を取得する際、トリガー監視はトリガーメッセージ内の情報の一部またはすべてを、トリガー監視が起動するアプリケーションに渡す必要がある場合があります。

開始済みアプリケーションに必要な情報には、*QName*、*TriggerData*、および *UserData* があります。トリガー・モニター・アプリケーションでは、起動したアプリケーションに MQTM 構造体を直接渡すだけでなく、MQTMC2 構造体を渡すこともできます。どちらを渡すかは、起動したアプリケーション側の環境および条件で許可されるもので決まります。

この構造体は、IBM MQ トリガー・モニター・インターフェース (TMI) の一部です。TMI は、IBM MQ フレームワーク・インターフェースに含まれています。

文字セットとエンコード

MQTMC2 内の文字データは、ローカルキューマネージャの文字セットで保持されます。これはキューマネージャ **CodedCharSetId** 属性によって指定されます。

使用法

MQTMC2 構造は、MQTM 構造の形式と非常に似ています。相違点は、MQTM 内の非文字フィールドが、MQTMC2 では、同じ長さの文字フィールドに変更される、構造体の終わりにキュー・マネージャー名が追加されることです。

- ▶ **z/OS** z/OS では、CSQQTRMN アプリケーションを使用して始動される MQAT_IMS アプリケーションの場合は、MQTMC2 構造体が開始済みアプリケーションで使用可能になります。

-  IBM i では、IBM MQ が提供するトリガー・モニター・アプリケーションが、MQTMC2 構造体を開始済みアプリケーションに渡します。

フィールド

注：次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 567. MQTMC2 のフィールド		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
StrucId (構造識別子)	MQTMC_STRUC_ID	'TMC'
バージョン (構造バージョン番号)	MQTMC_VERSION_2	'2'
QName (トリガーされたキューの名前)	なし	ヌル・ストリングまたはブランク
ProcessName (プロセスオブジェクト名)	なし	ヌル・ストリングまたはブランク
TriggerData (トリガーデータ)	なし	ヌル・ストリングまたはブランク
ApplType (アプリケーションの種類)	なし	ブランク
ApplId (アプリケーション識別子)	なし	ヌル・ストリングまたはブランク
EnvData (環境データ)	なし	ヌル・ストリングまたはブランク
UserData (ユーザーデータ)	なし	ヌル・ストリングまたはブランク
QMgrName (キューマネージャー名)	なし	ヌル・ストリングまたはブランク
注： - 記号は、単一のブランク文字を表します。 - ヌル・ストリングまたはブランクの値は、C 言語ではヌル・ストリングを表し、他のプログラミング言語ではブランク文字を表します。 - C プログラミング言語では、マクロ変数 MQTMC2_DEFAULT は、上記の値を含みます。このマクロ変数を以下の方法で使用して、構造体のフィールドに初期値を設定します。 <pre>MQTMC2 MyTMC = {MQTMC2_DEFAULT};</pre>		

言語ごとの宣言

MQTMC2 の C 宣言

```
typedef struct tagMQTMC2 MQTMC2;
struct tagMQTMC2 {
    MQCHAR4    StrucId;        /* Structure identifier */
    MQCHAR4    Version;        /* Structure version number */
    MQCHAR48    QName;         /* Name of triggered queue */
    MQCHAR48    ProcessName;    /* Name of process object */
    MQCHAR64    TriggerData;    /* Trigger data */
    MQCHAR4     ApplType;       /* Application type */
    MQCHAR256    ApplId;        /* Application identifier */
    MQCHAR128    EnvData;       /* Environment data */
    MQCHAR128    UserData;      /* User data */
}
```



```
MQCHAR48 QMgrName;      /* Queue manager name */
};
```

MQTMC2 の COBOL 宣言

```
** MQTMC2 structure
10 MQTMC2.
** Structure identifier
15 MQTMC2-STRUCID PIC X(4).
** Structure version number
15 MQTMC2-VERSION PIC X(4).
** Name of triggered queue
15 MQTMC2-QNAME PIC X(48).
** Name of process object
15 MQTMC2-PROCESSNAME PIC X(48).
** Trigger data
15 MQTMC2-TRIGGERDATA PIC X(64).
** Application type
15 MQTMC2-APPLTYPE PIC X(4).
** Application identifier
15 MQTMC2-APPLID PIC X(256).
** Environment data
15 MQTMC2-ENVDATA PIC X(128).
** User data
15 MQTMC2-USERDATA PIC X(128).
** Queue manager name
15 MQTMC2-QMGRNAME PIC X(48).
```

MQTMC2 の PL/I 宣言

```
dcl
1 MQTMC2 based,
3 StrucId char(4), /* Structure identifier */
3 Version char(4), /* Structure version number */
3 QName char(48), /* Name of triggered queue */
3 ProcessName char(48), /* Name of process object */
3 TriggerData char(64), /* Trigger data */
3 ApplType char(4), /* Application type */
3 ApplId char(256), /* Application identifier */
3 EnvData char(128), /* Environment data */
3 UserData char(128), /* User data */
3 QMgrName char(48); /* Queue manager name */
```

MQTMC2 の高水準アセンブラ宣言

```
MQTMC2          DSECT
MQTMC2_STRUCID   DS CL4      Structure identifier
MQTMC2_VERSION   DS CL4      Structure version number
MQTMC2_QNAME     DS CL48     Name of triggered queue
MQTMC2_PROCESSNAME DS CL48   Name of process object
MQTMC2_TRIGGERDATA DS CL64   Trigger data
MQTMC2_APPLTYPE  DS CL4      Application type
MQTMC2_APPLID    DS CL256    Application identifier
MQTMC2_ENVDATA   DS CL128    Environment data
MQTMC2_USERDATA  DS CL128    User data
MQTMC2_QMGRNAME  DS CL48     Queue manager name
*
MQTMC2_LENGTH    EQU *-MQTMC2
ORG MQTMC2
MQTMC2_AREA      DS CL(MQTMC2_LENGTH)
```

MQTMC2 の Visual Basic 宣言

```
Type MQTMC2
StrucId As String*4 'Structure identifier'
Version As String*4 'Structure version number'
QName As String*48 'Name of triggered queue'
ProcessName As String*48 'Name of process object'
TriggerData As String*64 'Trigger data'
ApplType As String*4 'Application type'
ApplId As String*256 'Application identifier'
EnvData As String*128 'Environment data'
UserData As String*128 'User data'
QMgrName As String*48 'Queue manager name'
End Type
```

MQTMC2 のフィールド詳細

トリガー監視アプリケーションが開始キューからトリガーメッセージ (MQTM) を取得する際、トリガー監視はトリガーメッセージ内の情報の一部またはすべてを、トリガー監視が起動するアプリケーションに渡す必要がある場合があります。

フィールド

注: 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 568. MQTMC2 のフィールド		
フィールド名と説明	定数の名前	定数の初期値 (存在する場合)
StrucId (構造識別子)	MQTMC_STRUC_ID	'TMC'
バージョン (構造バージョン番号)	MQTMC_VERSION_2	'2'
QName (トリガーされたキューの名前)	なし	ヌル・ストリングまたはブランク
ProcessName (プロセスオブジェクト名)	なし	ヌル・ストリングまたはブランク
TriggerData (トリガーデータ)	なし	ヌル・ストリングまたはブランク
ApplType (アプリケーションの種類)	なし	ブランク
ApplId (アプリケーション識別子)	なし	ヌル・ストリングまたはブランク
EnvData (環境データ)	なし	ヌル・ストリングまたはブランク
UserData (ユーザーデータ)	なし	ヌル・ストリングまたはブランク
QMgrName (キューマネージャー名)	なし	ヌル・ストリングまたはブランク
注: 1. 記号は、単一のブランク文字を表します。 2. ヌル・ストリングまたはブランクの値は、C 言語ではヌル・ストリングを表し、他のプログラミング言語ではブランク文字を表します。 3. C プログラミング言語では、マクロ変数 MQTMC2_DEFAULT は、上記の値を含みます。このマクロ変数を以下の方法で使用して、構造体のフィールドに初期値を設定します。 <pre>MQTMC2 MyTMC = {MQTMC2_DEFAULT};</pre>		

StrucId (MQCHAR4)

これはトリガーメッセージ 2 (文字形式) 構造の構造識別子です。常に入力フィールドです。その値は MQTMC2_STRUC_ID です。

値は次のものでなければなりません。

MQTMC2_STRUC_ID

トリガーメッセージ識別子 (文字形式) 構造体。

C プログラミング言語では、定数 MQTMC2_STRUC_ID_ARRAY も定義されている。これは MQTMC2_STRUC_ID と同じ値を持ちますが、文字列ではなく文字の配列です。

Version (MQCHAR4)

構造体のバージョン番号。

値は次のものでなければなりません。

MQTMC_VERSION_2

バージョン 2 トリガー・メッセージ (文字形式) 構造体。

C プログラミング言語では、定数 MQTMC_VERSION_2_ARRAY も定義されます。これは、MQTMC_VERSION_2 と同じ値ですが、ストリングではなく文字の配列です。

以下の定数は、現行バージョンのバージョン番号を指定しています。

MQTMC_CURRENT_VERSION

トリガー・メッセージ (文字形式) 構造体の現行バージョン。

QName (MQCHAR48)

起動されたキューの名前。

MQTM 構造体の [*QName*](#) フィールドを参照してください。

ProcessName (MQCHAR48)

プロセス・オブジェクトの名前。

MQTM 構造体の [*ProcessName*](#) フィールドを参照してください。

TriggerData (MQCHAR64)

トリガー・データです。

MQTM 構造体の [*TriggerData*](#) フィールドを参照してください。

ApplType (MQCHAR4)

アプリケーション・タイプ。

元のトリガー・メッセージの MQTM 構造体の [*ApplType*](#) フィールドの値にかかわらず、このフィールドには常にブランクが入っています。

ApplId (MQCHAR256)

アプリケーション ID。

MQTM 構造体の [*ApplId*](#) フィールドを参照してください。

EnvData (MQCHAR128)

環境データ。

MQTM 構造体の [*EnvData*](#) フィールドを参照してください。

UserData (MQCHAR128)

ユーザー・データ。

MQTM 構造体の [*UserData*](#) フィールドを参照してください。

QMgrName (MQCHAR48)

キュー・マネージャー名。

これは、トリガー・イベントが発生したキュー・マネージャーの名前です。

MQWIH - 作業情報ヘッダー

メッセージがワークロード z/OS・マネージャ (WLM) によって処理される場合、そのメッセージは MQWIH 構造体で開始されなければなりません。この構造は、WLM によって処理されるメッセージの先頭に存在しなければならぬ情報を記述する。

可用性

すべての IBM MQ システム、およびこれらのシステムに IBM MQ 接続されているクライアント。

形式名

MQFMT_WORK_INFO_HEADER.

文字セットとエンコード

MQWIH 構造体のフィールドは、MQWIH に先行するヘッダー構造体の `` *CodedCharSetId* および *Encoding* `` フィールド、または MQWIH がアプリケーションメッセージデータの先頭にある場合は MQMD 構造体のそれらのフィールドによって指定される文字セットおよびエンコーディングで記述される。

文字セットは、キュー名に有効な文字用の 1 バイト文字を持つ文字セットでなければなりません。

使用法

IBM MQ がサポートされるどのプラットフォームでも、MQWIH 構造体を含むメッセージを作成して送信できますが、WLM とやり取りできるのは IBM MQ for z/OS キュー・マネージャーのみです。したがって、z/OS 以外のキュー・マネージャーから WLM にメッセージが到達するためには、メッセージのルーティングで経由できる z/OS キュー・マネージャーが少なくとも 1 つ、キュー・マネージャー・ネットワークに含まれていなければなりません。

フィールド

注: 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 569. MQWIH のフィールド		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
<u>StrucId</u> (構造識別子)	MQWIH_STRUC_ID	'WIH~'
<u>バージョン</u> (構造バージョン番号)	MQWIH_VERSION_1	1
<u>StrucLength</u> (MQWIH 構造の長さ)	MQWIH_LENGTH_1	120
<u>エンコーディング</u> (MQWIH に続くデータの数字エンコーディング)	なし	0
<u>CodedCharSetId</u> (MQWIH に続くデータの文字セット識別子)	MQCCSI_UNDEFINED	0
<u>フォーマット</u> (MQWIH に続くデータのフォーマット名)	MQFMT_NONE	ブランク
<u>フラグ</u> (フラグ)	MQWIH_NONE	0
<u>ServiceName</u> (サービス名)	なし	ブランク
<u>ServiceStep</u> (サービスステップ名)	なし	ブランク

表 569. MQWIH のフィールド (続き)		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
MsgToken (メッセージトークン)	MQMTOK_NONE	Null
予約済み（予約済み）	なし	ブランク
注： - 記号-は、単一のブランク文字を表します。 - C プログラミング言語では、マクロ変数 MQWIH_DEFAULT には、表にリストされている値が含まれています。このマクロ変数を以下の方法で使用して、構造体のフィールドに初期値を設定します。 <pre>MQWIH MyWIH = {MQWIH_DEFAULT};</pre>		

言語ごとの宣言

MQWIH の C 宣言

```
typedef struct tagMQWIH MQWIH;
struct tagMQWIH {
    MQCHAR4    StrucId;        /* Structure identifier */
    MQLONG     Version;        /* Structure version number */
    MQLONG     StrucLength;    /* Length of MQWIH structure */
    MQLONG     Encoding;       /* Numeric encoding of data that follows
                               MQWIH */
    MQLONG     CodedCharSetId; /* Character-set identifier of data that
                               follows MQWIH */
    MQCHAR8     Format;         /* Format name of data that follows
                               MQWIH */
    MQLONG     Flags;          /* Flags */
    MQCHAR32    ServiceName;    /* Service name */
    MQCHAR8     ServiceStep;    /* Service step name */
    MQBYTE16    MsgToken;       /* Message token */
    MQCHAR32    Reserved;       /* Reserved */
};
```

MQWIH の COBOL 宣言

```
**      MQWIH structure
10 MQWIH.
**      Structure identifier
15 MQWIH-STRUCID          PIC X(4).
**      Structure version number
15 MQWIH-VERSION          PIC S9(9) BINARY.
**      Length of MQWIH structure
15 MQWIH-STRUCLength      PIC S9(9) BINARY.
**      Numeric encoding of data that follows MQWIH
15 MQWIH-ENCODING         PIC S9(9) BINARY.
**      Character-set identifier of data that follows MQWIH
15 MQWIH-CODEDCHARSETID   PIC S9(9) BINARY.
**      Format name of data that follows MQWIH
15 MQWIH-FORMAT           PIC X(8).
**      Flags
15 MQWIH-FLAGS            PIC S9(9) BINARY.
**      Service name
15 MQWIH-SERVICENAME       PIC X(32).
**      Service step name
15 MQWIH-SERVICESTEP      PIC X(8).
**      Message token
15 MQWIH-MSGTOKEN          PIC X(16).
**      Reserved
15 MQWIH-RESERVED          PIC X(32).
```

MQWIH の PL/I 宣言

```
dcl
  1 MQWIH based,
  3 StrucId          char(4),          /* Structure identifier */
```

```

3 Version          fixed bin(31), /* Structure version number */
3 StrucLength      fixed bin(31), /* Length of MQWIH structure */
3 Encoding         fixed bin(31), /* Numeric encoding of data that
                                follows MQWIH */
3 CodedCharSetId   fixed bin(31), /* Character-set identifier of data
                                that follows MQWIH */
3 Format           char(8),        /* Format name of data that follows
                                MQWIH */
3 Flags           fixed bin(31), /* Flags */
3 ServiceName      char(32),      /* Service name */
3 ServiceStep      char(8),       /* Service step name */
3 MsgToken         char(16),      /* Message token */
3 Reserved         char(32);      /* Reserved */

```

MQWIH の高水準アセンブラ宣言

```

MQWIH          DSECT
MQWIH_STRUCID   DS    CL4    Structure identifier
MQWIH_VERSION   DS    F      Structure version number
MQWIH_STRUCLNGTH DS    F      Length of MQWIH structure
MQWIH_ENCODING  DS    F      Numeric encoding of data that follows
*              MQWIH
MQWIH_CODEDCHARSETID DS    F  Character-set identifier of data that
*              follows MQWIH
MQWIH_FORMAT     DS    CL8    Format name of data that follows MQWIH
MQWIH_FLAGS      DS    F      Flags
MQWIH_SERVICENAME DS    CL32   Service name
MQWIH_SERVICESTEP DS    CL8    Service step name
MQWIH_MSGTOKEN   DS    XL16   Message token
MQWIH_RESERVED  DS    CL32   Reserved
*
MQWIH_LENGTH     EQU    *-MQWIH
ORG    MQWIH
MQWIH_AREA       DS    CL(MQWIH_LENGTH)

```

MQWIH の Visual Basic 宣言

```

Type MQWIH
    StrucId      As String*4 'Structure identifier'
    Version      As Long     'Structure version number'
    StrucLength  As Long     'Length of MQWIH structure'
    Encoding     As Long     'Numeric encoding of data that follows'
    CodedCharSetId As Long   'MQWIH'
    Format       As String*8 'Character-set identifier of data that'
    Flags        As Long     'follows MQWIH'
    ServiceName  As String*32 'Format name of data that follows MQWIH'
    ServiceStep  As Long     'Flags'
    MsgToken     As String*8  'Service name'
    Reserved     As String*32 'Service step name'
End Type

```

MQWIH のフィールド詳細

メッセージがワークロード z/OS・マネージャ (WLM) によって処理される場合、そのメッセージは MQWIH 作業情報ヘッダ構造で開始されなければならない。この構造は、WLM によって処理されるメッセージの先頭に存在しなければならない情報を記述する。

フィールド

注: 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 570. MQWIH のフィールド		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
<u>StrucId</u> (構造識別子)	MQWIH_STRUC_ID	'WIH~'
<u>バージョン</u> (構造バージョン番号)	MQWIH_VERSION_1	1
<u>StrucLength</u> (MQWIH 構造の長さ)	MQWIH_LENGTH_1	120

表 570. MQWIH のフィールド (続き)		
フィールド名と説明	定数の名前	定数の初期値 (存在する場合)
<u>エンコーディング</u> (MQWIH に続くデータの数字エンコーディング)	なし	0
<u>CodedCharSetId</u> (MQWIH に続くデータの文字セット識別子)	MQCCSI_UNDEFINED	0
<u>フォーマット</u> (MQWIH に続くデータのフォーマット名)	MQFMT_NONE	ブランク
<u>フラグ</u> (フラグ)	MQWIH_NONE	0
<u>ServiceName</u> (サービス名)	なし	ブランク
<u>ServiceStep</u> (サービスステップ名)	なし	ブランク
<u>MsgToken</u> (メッセージトークン)	MQMTOK_NONE	Null
<u>予約済み</u> (予約済み)	なし	ブランク
注: - 記号-は、単一のブランク文字を表します。 - C プログラミング言語では、マクロ変数 MQWIH_DEFAULT には、表にリストされている値が含まれています。このマクロ変数を以下の方法で使用して、構造体のフィールドに初期値を設定します。 <pre>MQWIH MyWIH = {MQWIH_DEFAULT};</pre>		

StrucId (MQCHAR4)

これは作業情報ヘッダー構造の構造識別子です。常に入力フィールドです。その値は MQWIH_STRUC_ID です。

値は次のものでなければなりません。

MQWIH_STRUC_ID

作業情報ヘッダー構造体の識別子。

C プログラミング言語では、定数 MQWIH_STRUC_ID_ARRAY も定義されている。これは MQWIH_STRUC_ID と同じ値を持ちますが、文字列ではなく文字の配列です。

Version (MQLONG)

これは構造体のバージョン番号です。値は次のものでなければなりません。

MQWIH_VERSION_1

バージョン 1 の作業情報ヘッダー構造体。

以下の定数は、現行バージョンのバージョン番号を指定しています。

MQWIH_CURRENT_VERSION

現行バージョンの作業情報ヘッダー構造体。

このフィールドの初期値は、MQWIH_VERSION_1 です。

StrucLength (MQLONG)

これは MQWIH 構造体の長さです。値は次のものでなければなりません。

MQWIH_LENGTH_1

バージョン 1 の作業情報ヘッダー構造体の長さ。

以下の定数は、現行バージョンの長さを指定しています。

MQWIH_CURRENT_LENGTH

現行バージョンの作業情報ヘッダー構造体の長さ。

このフィールドの初期値は、MQWIH_LENGTH_1 です。

Encoding (MQLONG)

これは、MQWIH の後に続くデータの数値エンコードを指定します。MQWIH 構造体自体の数値データには適用されません。

MQPUT または MQPUT1 呼び出しでは、アプリケーションは、このフィールドをデータに適切な値に設定する必要があります。

このフィールドの初期値は 0 です。

CodedCharSetId (MQLONG)

これは、MQWIH の後に続くデータの文字セット ID を指定します。MQWIH 構造体自体の文字データには適用されません。

MQPUT または MQPUT1 呼び出しでは、アプリケーションは、このフィールドをデータに適切な値に設定する必要があります。次の特殊値を使用することができます。

MQCCSI_INHERIT

この構造体の後に続くデータの文字データは、この構造体に設定されているのと同じ文字セットになります。

キュー・マネージャーは、メッセージで送信される構造体の中のこの値を、構造体の実際の文字セット ID に変更します。エラーが発生しない限り、値 MQCCSI_INHERIT が MQGET 呼び出しによって返されることはありません。

MQCCSI_INHERIT は、MQMD の *PutApplType* フィールドの値が MQAT_BROKER である場合は使用できません。

このフィールドの初期値は MQCCSI_UNDEFINED です。

Format (MQCHAR8)

これは、MQWIH 構造体の後に続くデータの形式名を指定します。

MQPUT または MQPUT1 呼び出しでは、アプリケーションは、このフィールドをデータに適切な値に設定する必要があります。このフィールドのコーディングの規則は、MQMD の *Format* フィールドの場合と同じです。

このフィールドの長さは MQ_FORMAT_LENGTH によって指定されます。このフィールドの初期値は MQFMT_NONE です。

フラグ (MQLONG)

値は次のものでなければなりません。

MQWIH_NONE

フラグなし。

このフィールドの初期値は、MQWIH_NONE です。

ServiceName (MQCHAR32)

これは、メッセージを処理するサービスの名前です。

このフィールドの長さは MQ_SERVICE_NAME_LENGTH によって指定されます。このフィールドの初期値は 32 個の空白文字です。

ServiceStep (MQCHAR8)

これは、メッセージに関連する *ServiceName* のステップの名前です。

このフィールドの長さは MQ_SERVICE_STEP_LENGTH によって指定されます。このフィールドの初期値は 8 個のブランク文字です。

MsgToken (MQBYTE16)

これは、メッセージを一意に識別するメッセージ・トークンです。

MQPUT および MQPUT1 呼び出しでは、このフィールドは無視されます。このフィールドの長さは MQ_MSG_TOKEN_LENGTH によって指定されます。このフィールドの初期値は、MQMTOK_NONE です。

Reserved (MQCHAR32)

これは予約フィールドです。フィールドはブランクでなければなりません。

MQXP - 出口パラメーター・ブロック

MQXP 構造体は、API 横断出口への入力/出力パラメータとして使用されます。この出口の詳細については、[API 交差出口](#)を参照してください。

文字セットとエンコード

MQXP 内の文字データは、ローカルキューマネージャの文字セットで保持されます。これはキューマネージャ **CodedCharSetId** 属性によって指定されます。MQXP の数値データは、ネイティブ・マシン・エンコードです。これは、MQENC_NATIVE の値で指定します。

フィールド

注：次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 571. MQXP のフィールド	
フィールド名と説明	定数の名前
<u>StrucId</u> (構造識別子)	MQXP_STRUC_ID
<u>バージョン</u> (構造バージョン番号)	MQXP_VERSION_1
<u>ExitId</u> (終了識別子)	MQXT_API_CROSSING_EXIT
<u>ExitReason</u> (終了呼び出しの理由)	なし
<u>ExitResponse</u> (出口からの応答)	なし
<u>ExitCommand</u> (API 呼び出しコード)	なし
<u>ExitParmCount</u> (パラメータ数)	なし
<u>予約済み</u> (予約済み)	なし
<u>ExitUserArea</u> (ユーザーエリア)	なし

言語ごとの宣言

MQXP の C 宣言

```
typedef struct tagMQXP MQXP;
struct tagMQXP {
    MQCHAR4    StrucId;        /* Structure identifier */
    MQLONG     Version;        /* Structure version number */
    MQLONG     ExitId;         /* Exit identifier */
    MQLONG     ExitReason;     /* Reason for invocation of exit */
}
```

```

MQLONG    ExitResponse;    /* Response from exit */
MQLONG    ExitCommand;     /* API call code */
MQLONG    ExitParmCount;   /* Parameter count */
MQLONG    Reserved;        /* Reserved */
MQBYTE16  ExitUserArea;    /* User area */
};

```

MQXP の COBOL 宣言

```

**  MQXP structure
10 MQXP.
**  Structure identifier
15 MQXP-STRUCID      PIC X(4).
**  Structure version number
15 MQXP-VERSION      PIC S9(9) BINARY.
**  Exit identifier
15 MQXP-EXITID       PIC S9(9) BINARY.
**  Reason for invocation of exit
15 MQXP-EXITREASON   PIC S9(9) BINARY.
**  Response from exit
15 MQXP-EXITRESPONSE PIC S9(9) BINARY.
**  API call code
15 MQXP-EXITCOMMAND  PIC S9(9) BINARY.
**  Parameter count
15 MQXP-EXITPARMCOUNT PIC S9(9) BINARY.
**  Reserved
15 MQXP-RESERVED      PIC S9(9) BINARY.
**  User area
15 MQXP-EXITUSERAREA  PIC X(16).

```

MQXP の PL/I 宣言

```

dcl
1 MQXP based,
3 StrucId      char(4),          /* Structure identifier */
3 Version      fixed bin(31),    /* Structure version number */
3 ExitId       fixed bin(31),    /* Exit identifier */
3 ExitReason    fixed bin(31),   /* Reason for invocation of exit */
3 ExitResponse  fixed bin(31),   /* Response from exit */
3 ExitCommand   fixed bin(31),   /* API call code */
3 ExitParmCount fixed bin(31),   /* Parameter count */
3 Reserved      fixed bin(31),   /* Reserved */
3 ExitUserArea  char(16);        /* User area */

```

MQXP の高水準アセンブラ宣言

```

MQXP          DSECT
MQXP_STRUCID  DS    CL4    Structure identifier
MQXP_VERSION  DS    F      Structure version number
MQXP_EXITID   DS    F      Exit identifier
MQXP_EXITREASON DS    F    Reason for invocation of exit
MQXP_EXITRESPONSE DS    F  Response from exit
MQXP_EXITCOMMAND DS    F   API call code
MQXP_EXITPARMCOUNT DS    F  Parameter count
MQXP_RESERVED DS    F      Reserved
MQXP_EXITUSERAREA DS    XL16 User area
*
MQXP_LENGTH  EQU    *-MQXP
ORG    MQXP
MQXP_AREA    DS     CL(MQXP_LENGTH)

```

MQXP のフィールド詳細

MQXP Exit パラメータブロック構造体は、API 横断出口への入力/出力パラメータとして使用されます。

フィールド

注: 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 572. MQXP のフィールド	
フィールド名と説明	定数の名前
StrucId (構造識別子)	MQXP_STRUC_ID

表 572. MQXP のフィールド (続き)	
フィールド名と説明	定数の名前
<u>バージョン</u> (構造バージョン番号)	MQXP_VERSION_1
<u>ExitId</u> (終了識別子)	MQXT_API_CROSSING_EXIT
<u>ExitReason</u> (終了呼び出しの理由)	なし
<u>ExitResponse</u> (出口からの応答)	なし
<u>ExitCommand</u> (API 呼び出しコード)	なし
<u>ExitParmCount</u> (パラメータ数)	なし
<u>予約済み</u> (予約済み)	なし
<u>ExitUserArea</u> (ユーザーエリア)	なし

StrucId (MQCHAR4)

これは、終了パラメータ構造体の構造体識別子です。常に入力フィールドです。その値は MQXP_STRUC_ID です。

値は次のものでなければなりません。

MQXP_STRUC_ID

終了パラメータ構造体の識別子。

C プログラミング言語では、定数 MQXP_STRUC_ID_ARRAY も定義されている。これは MQXP_STRUC_ID と同じ値を持ちますが、文字列ではなく文字の配列です。

Version (MQLONG)

これは構造体のバージョン番号です。値は次のものでなければなりません。

MQXP_VERSION_1

出口パラメーター・ブロック構造体のバージョン番号。

注：この構造体の新しいバージョンが導入されても、既存の部分のレイアウトは変わりません。したがって、バージョン番号が出口で使用しなければならないフィールドが含まれている最小バージョンと等しいかまたはそれより大きいことを、出口で検査する必要があります。

これは、出口に対する入力フィールドです。

ExitId (MQLONG)

このフィールドは、出口ルーチンへの入り口で設定され、出口のタイプを示します。

MQXT_API_CROSSING_EXIT

CICS 用の API 交差出口。

これは、出口に対する入力フィールドです。

ExitReason (MQLONG)

これは、出口ルーチンへの入り口で設定されます。これは、API 交差出口について、ルーチンの呼び出しが API 呼び出しの実行前なのか実行後なのかを示します。

MQXR_BEFORE

API 実行前。

MQXR_AFTER

API 実行後。

これは、出口に対する入力フィールドです。

ExitResponse (MQLONG)

この値は、呼び出し元と通信するために、出口により設定されます。以下の値が定義されます。

MQXCC_OK

出口が正常に終了した。

MQXCC_SUPPRESS_FUNCTION

機能を抑止。

この値が API 呼び出しの前に呼び出された API 相互出口により設定された場合、その API 呼び出しは実行されません。呼び出しの *CompCode* は MQCC_FAILED に設定され、*Reason* は MQRC_SUPPRESSED_BY_EXIT に設定されます。その他のすべてのパラメーターは、出口が終了するときの状態となります。

この値が API 呼び出しの後に呼び出された API 交差出口により設定された場合、キュー・マネージャーはこの値を無視します。

MQXCC_SKIP_FUNCTION

機能のスキップ。

この値が API 呼び出しより前に呼び出された API 交差出口により設定された場合は、その API 呼び出しは実行されません。*CompCode*、*Reason*、および他のすべてのパラメーターは、出口が終了したときの状態となります。

この値が API 呼び出しの後に呼び出された API 交差出口により設定された場合、キュー・マネージャーはこの値を無視します。

これは、出口からの出力フィールドです。

ExitCommand (MQLONG)

このフィールドは、出口ルーチンへの入り口で設定されます。フィールドは、出口を呼び出す API 呼び出しを識別します。

MQXC_CALLBACK

CALLBACK 呼び出し。

MQXC_MQBACK

MQBACK 呼び出し。

MQXC_MQCB

MQCB 呼び出し。

MQXC_MQCLOSE

MQCLOSE 呼び出し。

MQXC_MQCMIT

MQCMIT 呼び出し。

MQXC_MQCTL

MQCTL 呼び出し。

MQXC_MQGET

MQGET 呼び出し。

MQXC_MQINQ

MQINQ 呼び出し。

MQXC_MQOPEN

MQOPEN 呼び出し。

MQXC_MQPUT

MQPUT 呼び出し。

MQXC_MQPUT1

MQPUT1 呼び出し。

MQXC_MQSET

MQSET 呼び出し。

MQXC_MQSTAT

MQSTAT 呼び出し。

MQXC_MQSUB

MQSUB 呼び出し。

MQXC_MQSUBRQ

MQSUBRQ 呼び出し。

これは、出口に対する入力フィールドです。

ExitParmCount (MQLONG)

このフィールドは、出口ルーチンへの入り口で設定されます。フィールドには、MQ 呼び出しが使用するパラメーターの数が含まれています。

表 573. 各 MQ 呼び出しのパラメーターの数

呼び出し名	パラメーターの数
MQBACK	3
MQCLOSE	5
MQCMIT	3
MQGET	9
MQINQ	10
MQOPEN	6
MQPUT	8
MQPUT1	8
MQSET	10

これは、出口に対する入力フィールドです。

Reserved (MQLONG)

これは予約フィールドです。この値は、出口には無効です。

ExitUserArea (MQBYTE16)

これは、出口が使用できるフィールドです。このフィールドのフィールド長は、タスクで最初にこの出口を呼び出す前に、2 進ゼロに初期設定されます。そして、出口がこのフィールドに対して行った変更は、以後出口を呼び出すたびに保存されます。以下の値が定義されます。

MQXUA_NONE

ユーザー情報なし。

値は、フィールドの長さについては 2 進ゼロです。

C プログラミング言語では、定数 MQXUA_NONE_ARRAY も定義されます。これは、MQXUA_NONE と同じ値ですが、ストリングではなく文字の配列です。

このフィールドの長さは MQ_EXIT_USER_AREA_LENGTH によって指定されます。これは、出口に対する入出力フィールドです。

MQXQH - 伝送キュー・ヘッダー

MQXQH 構造体は、伝送キューに入っているメッセージのアプリケーション・メッセージ・データの接頭部に付けられる情報を記述します。伝送キューは、特殊なタイプのローカル・キューで、リモート・キューに宛先指定された (つまり、ローカル・キュー・マネージャーに属さないキューに宛先指定された) メッセ

ージを一時的に保持します。送信キューは、値 MQUS_TRANSMISSION を持つ queue 属性 **Usage** によって示される。

形式名

MQFMT_XMIT_Q_HEADER

文字セットとエンコード

MQXQH 内のデータは、キューマネージャ **CodedCharSetId** 属性によって指定される文字セットおよびローカルキューマネージャのエンコーディング（MQENC_NATIVE で指定）でなければならない。

MQXQH の文字セットおよびエンコードは、以下の構造体の *CodedCharSetId* および *Encoding* フィールドに設定する必要があります。

- 分離 MQMD (MQXQH 構造体がメッセージ・データの開始点にある場合)
- MQXQH 構造体に先行するヘッダー構造体 (その他のすべての場合)

フィールド

注: 次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 574. MQXQH の MQXQH フィールド		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
StrucId (構造識別子)	MQXQH_STRUC_ID	'XQH'
バージョン (構造バージョン番号)	MQXQH_VERSION_1	1
RemoteQName (宛先キュー名)	なし	ヌル・ストリングまたは ブランク
RemoteQMgrName (宛先キューマネージャ名)	なし	ヌル・ストリングまたは ブランク
MsgDesc (元のメッセージ記述子)	MQMD と同じ名前と値； 参照 391 ページの表 518	-
注: 1. 記号-は、単一のブランク文字を表します。 2. ヌル・ストリングまたはブランクの値は、C 言語ではヌル・ストリングを表し、他のプログラミング言語ではブランク文字を表します。 3. C プログラミング言語では、マクロ変数 MQXQH_DEFAULT には、表にリストされている値が含まれています。このマクロ変数を以下の方法で使用して、構造体のフィールドに初期値を設定します。 MQXQH MyXQH = {MQXQH_DEFAULT};		

言語ごとの宣言

MQXQH の C 宣言

```
typedef struct tagMQXQH MQXQH;
struct tagMQXQH {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQCHAR48   RemoteQName;      /* Name of destination queue */
    MQCHAR48   RemoteQMgrName;   /* Name of destination queue manager */
}
```

```

MQMD1      MsgDesc;          /* Original message descriptor */
};

```

MQXQH の COBOL 宣言

```

**  MQXQH structure
10 MQXQH.
**  Structure identifier
15 MQXQH-STRUCID          PIC X(4).
**  Structure version number
15 MQXQH-VERSION          PIC S9(9) BINARY.
**  Name of destination queue
15 MQXQH-REMOTEQNAME      PIC X(48).
**  Name of destination queue manager
15 MQXQH-REMOTEQMGRNAME   PIC X(48).
**  Original message descriptor
15 MQXQH-MSGDESC.
**  Structure identifier
20 MQXQH-MSGDESC-STRUCID  PIC X(4).
**  Structure version number
20 MQXQH-MSGDESC-VERSION  PIC S9(9) BINARY.
**  Report options
20 MQXQH-MSGDESC-REPORT   PIC S9(9) BINARY.
**  Message type
20 MQXQH-MSGDESC-MSGTYPE  PIC S9(9) BINARY.
**  Expiry time
20 MQXQH-MSGDESC-EXPIRY   PIC S9(9) BINARY.
**  Feedback or reason code
20 MQXQH-MSGDESC-FEEDBACK PIC S9(9) BINARY.
**  Numeric encoding of message data
20 MQXQH-MSGDESC-ENCODING PIC S9(9) BINARY.
**  Character set identifier of message data
20 MQXQH-MSGDESC-CODEDCHARSETID PIC S9(9) BINARY.
**  Format name of message data
20 MQXQH-MSGDESC-FORMAT   PIC X(8).
**  Message priority
20 MQXQH-MSGDESC-PRIORITY PIC S9(9) BINARY.
**  Message persistence
20 MQXQH-MSGDESC-PERSISTENCE PIC S9(9) BINARY.
**  Message identifier
20 MQXQH-MSGDESC-MSGID    PIC X(24).
**  Correlation identifier
20 MQXQH-MSGDESC-CORRELID PIC X(24).
**  Backout counter
20 MQXQH-MSGDESC-BACKOUTCOUNT PIC S9(9) BINARY.
**  Name of reply-to queue
20 MQXQH-MSGDESC-REPLYTOQ PIC X(48).
**  Name of reply queue manager
20 MQXQH-MSGDESC-REPLYTOQMGR PIC X(48).
**  User identifier
20 MQXQH-MSGDESC-USERIDENTIFIER PIC X(12).
**  Accounting token
20 MQXQH-MSGDESC-ACCOUNTINGTOKEN PIC X(32).
**  Application data relating to identity
20 MQXQH-MSGDESC-APPLIDENTITYDATA PIC X(32).
**  Type of application that put the message
20 MQXQH-MSGDESC-PUTAPPLTYPE PIC S9(9) BINARY.
**  Name of application that put the message
20 MQXQH-MSGDESC-PUTAPPLNAME PIC X(28).
**  Date when message was put
20 MQXQH-MSGDESC-PUTDATE PIC X(8).
**  Time when message was put
20 MQXQH-MSGDESC-PUTTIME PIC X(8).
**  Application data relating to origin
20 MQXQH-MSGDESC-APPLORIGINDATA PIC X(4).

```

MQXQH の PL/I 宣言

```

dcl
1 MQXQH based,
3 StrucId          char(4),          /* Structure identifier */
3 Version          fixed bin(31),    /* Structure version number */
3 RemoteQName      char(48),         /* Name of destination queue */
3 RemoteQMgrName   char(48),         /* Name of destination queue
                                     manager */
3 MsgDesc,
5 StrucId          char(4),          /* Structure identifier */
5 Version          fixed bin(31),    /* Structure version number */
5 Report           fixed bin(31),    /* Report options */

```

```

5 MsgType          fixed bin(31), /* Message type */
5 Expiry           fixed bin(31), /* Expiry time */
5 Feedback         fixed bin(31), /* Feedback or reason code */
5 Encoding         fixed bin(31), /* Numeric encoding of message
                                data */
5 CodedCharSetId   fixed bin(31), /* Character set identifier of
                                message data */
5 Format           char(8),        /* Format name of message data */
5 Priority         fixed bin(31), /* Message priority */
5 Persistence      fixed bin(31), /* Message persistence */
5 MsgId           char(24),       /* Message identifier */
5 CorrelId        char(24),       /* Correlation identifier */
5 BackoutCount     fixed bin(31), /* Backout counter */
5 ReplyToQ        char(48),       /* Name of reply-to queue */
5 ReplyToMgr       char(48),       /* Name of reply queue manager */
5 UserIdentifier   char(12),      /* User identifier */
5 AccountingToken  char(32),      /* Accounting token */
5 ApplIdentityData char(32),      /* Application data relating to
                                identity */
5 PutApplType      fixed bin(31), /* Type of application that put the
                                message */
5 PutApplName      char(28),      /* Name of application that put the
                                message */
5 PutDate          char(8),       /* Date when message was put */
5 PutTime          char(8),       /* Time when message was put */
5 ApplOriginData   char(4);      /* Application data relating to
                                origin */

```

MQXQH の高水準アセンブラ宣言

```

MQXQH              DSECT
MQXQH_STRUCID      DS    CL4      Structure identifier
MQXQH_VERSION      DS    F        Structure version number
MQXQH_REMOTEQNAME  DS    CL48     Name of destination queue
MQXQH_REMOTEQMGRNAME DS    CL48     Name of destination queue
*                                     manager
MQXQH_MSGDESC      DS    0F       Force fullword alignment
MQXQH_MSGDESC_STRUCID DS    CL4      Structure identifier
MQXQH_MSGDESC_VERSION DS    F        Structure version number
MQXQH_MSGDESC_REPORT DS    F        Report options
MQXQH_MSGDESC_MSGTYPE DS    F        Message type
MQXQH_MSGDESC_EXPIRY DS    F        Expiry time
MQXQH_MSGDESC_FEEDBACK DS    F        Feedback or reason code
MQXQH_MSGDESC_ENCODING DS    F        Numeric encoding of message
*                                     data
MQXQH_MSGDESC_CODEDCHARSETID DS    F        Character set identifier of
*                                     message data
MQXQH_MSGDESC_FORMAT DS    CL8      Format name of message data
MQXQH_MSGDESC_PRIORITY DS    F        Message priority
MQXQH_MSGDESC_PERSISTENCE DS    F        Message persistence
MQXQH_MSGDESC_MSGID DS    XL24     Message identifier
MQXQH_MSGDESC_CORRELID DS    XL24     Correlation identifier
MQXQH_MSGDESC_BACKOUTCOUNT DS    F        Backout counter
MQXQH_MSGDESC_REPLYTOQ DS    CL48     Name of reply-to queue
MQXQH_MSGDESC_REPLYTOQMGR DS    CL48     Name of reply queue manager
MQXQH_MSGDESC_USERIDENTIFIER DS    CL12     User identifier
MQXQH_MSGDESC_ACCOUNTINGTOKEN DS    XL32     Accounting token
MQXQH_MSGDESC_APPLIDENTITYDATA DS    CL32     Application data relating to
*                                     identity
MQXQH_MSGDESC_PUTAPPLTYPE DS    F        Type of application that put
*                                     the message
MQXQH_MSGDESC_PUTAPPLNAME DS    CL28     Name of application that put
*                                     the message
MQXQH_MSGDESC_PUTDATE DS    CL8      Date when message was put
MQXQH_MSGDESC_PUTTIME DS    CL8      Time when message was put
MQXQH_MSGDESC_APPLORIGINDATA DS    CL4      Application data relating to
*                                     origin
MQXQH_MSGDESC_LENGTH EQU    *-MQXQH_MSGDESC
                        ORG    MQXQH_MSGDESC
MQXQH_MSGDESC_AREA   DS    CL(MQXQH_MSGDESC_LENGTH)
*
MQXQH_LENGTH         EQU    *-MQXQH
                        ORG    MQXQH
MQXQH_AREA            DS    CL(MQXQH_LENGTH)

```

MQXQH の Visual Basic 宣言

```

Type MQXQH
StrucId As String*4 'Structure identifier'

```


Version	As Long	'Structure version number'
RemoteQName	As String*48	'Name of destination queue'
RemoteQMgrName	As String*48	'Name of destination queue manager'
MsgDesc	As MQMD1	'Original message descriptor'
End Type		

独立メッセージ記述子内のフィールド

送信キュー上にあるメッセージには、2つのメッセージ記述子がある：

- メッセージ・データから独立して保管されるメッセージ記述子。これは独立メッセージ記述子と呼ばれ、メッセージが伝送キューに配置される場合、キュー・マネージャーにより生成されます。独立メッセージ記述子内のフィールドのいくつかは、MQPUT または MQPUT1 呼び出しでアプリケーションが提供するメッセージ記述子からコピーされます。

独立メッセージ記述子は、メッセージが伝送キューから除去されると、MQGET 呼び出しの **MsgDesc** パラメーターにあるアプリケーションに戻されます。

- 2 番目のメッセージ記述子は、メッセージ・データの一部として MQXQH 構造体内に保存されます。これは組み込みメッセージ記述子と呼ばれ、MQPUT または MQPUT1 呼び出し (少しのバリエーションあり) でアプリケーションが提供したメッセージ記述子のコピーです。

組み込みメッセージ記述子は、常にバージョン 1 の MQMD です。アプリケーションが書き込んだメッセージでは、MQMD 内の 1 つ以上のバージョン 2 フィールドにデフォルト値ではない値があると、MQMDE 構造体が MQXQH 構造体の後に続き、さらにアプリケーション・メッセージ・データがあればこれが続きます。この MQMDE 構造体は、次のいずれかです。

- キュー・マネージャーによって生成された (アプリケーションがメッセージを書き込むのにバージョン 2 の MQMD を使用した場合)。
- アプリケーション・メッセージ・データ開始時点からすでにあった (アプリケーションがメッセージを書き込むのにバージョン 1 の MQMD を使用した場合)。

組み込みメッセージ記述子は、メッセージが最終宛先キューから除去されると、MQGET 呼び出しの **MsgDesc** パラメーターにあるアプリケーションに戻されます。

個別のメッセージ記述子内のフィールドは、示されているようにキューマネージャによって設定される。キュー・マネージャーがバージョン 2 の MQMD をサポートしていない場合、バージョン 1 の MQMD は機能を低下させることなく使用されます。

表 575. 独立 MQMD のフィールドに使用される値

独立 MQMD のフィールド	使用される値
<i>StrucId</i>	MQMD_STRUC_ID
<i>Version</i>	MQMD_VERSION_2
<i>Report</i>	組み込みメッセージ記述子からコピーされますが、ゼロに設定された MQRO_ACCEPT_UNSUP_IF_XMIT_MASK により識別されるビットがあります。(これにより、メッセージが伝送キューに入れられる場合や伝送キューから除去される場合に、COA または COD レポート・メッセージが生成されなくなります。)
<i>MsgType</i>	組み込みメッセージ記述子からコピーされます。
<i>Expiry</i>	組み込みメッセージ記述子からコピーされます。
<i>Feedback</i>	組み込みメッセージ記述子からコピーされます。
<i>Encoding</i>	MQENC_NATIVE (注を参照)
<i>CodedCharSetId</i>	キュー・マネージャーの CodedCharSetId 属性。
<i>Format</i>	MQFMT_XMIT_Q_HEADER
<i>Priority</i>	組み込みメッセージ記述子からコピーされます。
<i>Persistence</i>	組み込みメッセージ記述子からコピーされます。

表 575. 独立 MQMD のフィールドに使用される値 (続き)

独立 MQMD のフィールド	使用される値
<i>MsgId</i>	新しい値が、キュー・マネージャーにより生成されます。このメッセージ ID は、前述のようにキュー・マネージャーが組み込みメッセージ記述子に対して生成している可能性のある <i>MsgId</i> とは異なります。
<i>CorrelId</i>	組み込みメッセージ記述子からの <i>MsgId</i> 。 SYSTEM.CLUSTER.TRANSMIT.QUEUE に書き込まれるメッセージでは、 <i>CorrelId</i> は内部使用のために予約されています。
<i>BackoutCount</i>	0
<i>ReplyToQ</i>	組み込みメッセージ記述子からコピーされます。
<i>ReplyToQMgr</i>	組み込みメッセージ記述子からコピーされます。
<i>UserIdentifier</i>	組み込みメッセージ記述子からコピーされます。
<i>AccountingToken</i>	組み込みメッセージ記述子からコピーされます。 SYSTEM.CLUSTER.TRANSMIT.QUEUE に書き込まれるメッセージでは、 <i>AccountingToken</i> は内部使用のために予約されています。
<i>ApplIdentityData</i>	組み込みメッセージ記述子からコピーされます。
<i>PutApplType</i>	MQAT_QMGR
<i>PutApplName</i>	キュー・マネージャー名の最初の 28 バイト。
<i>PutDate</i>	メッセージが伝送キューに書き込まれた日付。
<i>PutTime</i>	メッセージが伝送キューに書き込まれた時刻。
<i>ApplOriginData</i>	ブランク
<i>GroupId</i>	MQGI_NONE
<i>MsgSeqNumber</i>	1
<i>Offset</i>	0
<i>MsgFlags</i>	MQMF_NONE
<i>OriginalLength</i>	MQOL_UNDEFINED

- Windows では、Micro Focus COBOL の MQENC_NATIVE の値は、C の値とは異なります。別個のメッセージ記述子の *Encoding* フィールドの値は、これらの環境では常に C の値になります。この値は、10 進数の 546 です。また、MQXQH 構造体の整数フィールドは、この値に対応するエンコード方式になります (ネイティブ Intel エンコード)。

組み込みメッセージ記述子中のフィールド

埋め込みメッセージ記述子のフィールドは、以下の例外を除き、MQPUT または MQPUT1 呼び出しの `PARAMETER **MsgDesc**` パラメータのフィールドと同じ値を持ちます：

- *Version* フィールドの値は、常に MQMD_VERSION_1 です。
- *Priority* フィールドの値が、MQPRI_PRIORITY_AS_Q_DEF の場合、値は、キューの **DefPriority** 属性の値に置き換えられます。
- *Persistence* フィールドの値が、MQPER_PERSISTENCE_AS_Q_DEF の場合、値は、キューの **DefPersistence** 属性の値に置き換えられます。
- *MsgId* フィールドの値が MQMI_NONE であるか、MQPMO_NEW_MSG_ID オプションが指定されるか、またはメッセージが配布リストのメッセージの場合、*MsgId* は、キュー・マネージャーによって生成された新しいメッセージ ID に置き換えられます。

配布リストのメッセージが異なる伝送キューに置かれた短い配布リストのメッセージに細分化される場合、新しいどの組み込みメッセージ記述子の *MsgId* フィールドも、元の配布リストのメッセージのフィールドと同じです。

- MQPMO_NEW_CORREL_ID オプションが指定されると、*CorrelId* はキュー・マネージャーによって生成された新しい相関 ID に置き換えられます。
- コンテキストフィールドは、パラメータ **PutMsgOpts** で指定された MQPMO_*_CONTEXT オプションに従って設定されます。コンテキストフィールドは以下の通りです：
 - *AccountingToken*
 - *ApplIdentityData*
 - *ApplOriginData*
 - *PutApplName*
 - *PutApplType*
 - *PutDate*
 - *PutTime*
 - *UserIdentifier*
- バージョン 2 のフィールドがある場合、1 つまたは複数のバージョン 2 フィールドにデフォルトではない値があると、これは MQMD から取り除かれ、さらに MQMDE 構造体に移動されます。

リモート・キューへのメッセージの書き込み

アプリケーションがリモートキューにメッセージを配置する場合（リモートキュー名を直接指定するか、リモートキューのローカル定義を使用するか）、ローカルキューマネージャは次の処理を行います：

- 組み込みメッセージ記述子が入っている MQXQH 構造体の作成
- 必要な MQMDE 構造体がまだない場合、その MQMDE の追加
- アプリケーション・メッセージ・データの追加
- 該当する伝送キューへのメッセージの格納

伝送キューにメッセージを直接書き込む場合

アプリケーションは、メッセージを直接送信キューに配置することもできる。この場合、アプリケーションは、アプリケーション・メッセージ・データの接頭部に MQXQH 構造体を付け、適切な値でフィールドを初期設定する必要があります。さらに、MQPUT または MQPUT1 呼び出しのパラメータ **MsgDesc** 内のフィールド *Format* には、MQFMT_XMIT_Q_HEADER の値を設定する必要があります。

アプリケーションにより作成された MQXQH 構造体の中の文字データは、ローカル・キュー・マネージャの文字セットに含まれているもの (**CodedCharSetId** キュー・マネージャー属性で定義されたもの) でなければならず、整数データは固有のマシン・エンコードに含まれているものでなければなりません。さらに、MQXQH 構造体の中にある文字データは、フィールドの定義長までブランクを埋め込む必要があります。ヌル文字を使用してデータを未完了で終了させてはなりません。キュー・マネージャは、MQXQH 構造体に含まれるヌル文字とその後続の文字をブランクに変換しないためです。

ただし、キュー・マネージャは、MQXQH 構造体が存在していること、およびそのフィールドに対して有効な値が指定されていることは検査しません。

アプリケーションは、SYSTEM.CLUSTER.TRANSMIT.QUEUE に直接メッセージを書き込むことはできません。

伝送キューからメッセージを読み取る場合

伝送キューからメッセージを読み取るアプリケーションでは、MQXQH 構造体に含まれている情報を適切な方法で処理する必要があります。アプリケーションメッセージデータの先頭に MQXQH 構造体が存在することは、MQGET 呼び出しのパラメータ **MsgDesc** の *Format* フィールドで返される値

MQFMT_XMIT_Q_HEADER によって示される。パラメータ **MsgDesc** のおよび *Encoding CodedCharSetId* フィールドで返される値は、MQXQH 構造体内の文字データおよび整数データの文字セットとエンコーディングを示します。アプリケーション・メッセージ・データの文字セットおよびエンコードは、組み込みメッセージ記述子の中の *CodedCharSetId* および *Encoding* フィールドによって定義されます。

MQXQH のフィールド詳細

MQXQH 送信キューヘッダー構造体は、メッセージが送信キュー上にある際に、アプリケーションメッセージデータに接頭辞として付加される情報を記述する。

フィールド

注：次の表では、フィールドはアルファベット順ではなく、用途ごとにグループ分けされています。

表 576. MQXQH の MQXQH フィールド		
フィールド名と説明	定数の名前	定数の初期値（存在する場合）
<u>StrucId</u> (構造識別子)	MQXQH_STRUC_ID	'XQH↵'
<u>バージョン</u> (構造バージョン番号)	MQXQH_VERSION_1	1
<u>RemoteQName</u> (宛先キュー名)	なし	ヌル・ストリングまたはブランク
<u>RemoteQMgrName</u> (宛先キューマネージャ名)	なし	ヌル・ストリングまたはブランク
<u>MsgDesc</u> (元のメッセージ記述子)	MQMD と同じ名前と値； 参照 391 ページの表 518	-
注： - 記号↵は、単一のブランク文字を表します。 - ヌル・ストリングまたはブランクの値は、C 言語ではヌル・ストリングを表し、他のプログラミング言語ではブランク文字を表します。 - C プログラミング言語では、マクロ変数 MQXQH_DEFAULT には、表にリストされている値が含まれています。このマクロ変数を以下の方法で使用して、構造体のフィールドに初期値を設定します。 <pre>MQXQH MyXQH = {MQXQH_DEFAULT};</pre>		

StrucId (MQCHAR4)

これは、伝送キューヘッダー構造体の構造体識別子です。常に入力フィールドです。その値は MQXQH_STRUC_ID です。

値は次のものでなければなりません。

MQXQH_STRUC_ID

伝送キューヘッダー構造体の識別子。

C プログラミング言語では、定数 MQXQH_STRUC_ID_ARRAY も定義されている。これは MQXQH_STRUC_ID と同じ値を持ちますが、文字列ではなく文字の配列です。

Version (MQLONG)

これは構造体のバージョン番号です。値は次のものでなければなりません。

MQXQH_VERSION_1

伝送キュー・ヘッダー構造体のバージョン番号。

以下の定数は、現行バージョンのバージョン番号を指定しています。

MQXQH_CURRENT_VERSION

伝送キューのヘッダー構造体の現行バージョン。

フィールドの初期値は、MQXQH_VERSION_1 です。

RemoteQName (MQCHAR48)

これは、メッセージの最終的な宛先であるメッセージ・キューの名前です (これは、例えば、このキューが *RemoteQMgrName* で別のリモート・キューのローカル定義として定義されている場合、最終的な宛先ではない可能性があります)。

メッセージが配布リスト・メッセージの場合 (つまり、組み込みメッセージ記述子の *Format* フィールドが MQFMT_DIST_HEADER の場合)、*RemoteQName* はブランクです。

このフィールドの長さは MQ_Q_NAME_LENGTH によって指定されます。このフィールドの初期値は、C 言語ではヌル・ストリングであり、他のプログラミング言語では 48 桁のブランク文字です。

RemoteQMgrName (MQCHAR48)

これは、メッセージの最終宛先であるキューを所有するキュー・マネージャー、またはキュー共有グループの名前です。

メッセージが配布リスト・メッセージの場合、*RemoteQMgrName* はブランクです。

このフィールドの長さは MQ_Q_MGR_NAME_LENGTH で指定します。このフィールドの初期値は、C 言語ではヌル・ストリングであり、他のプログラミング言語では 48 桁のブランク文字です。

MsgDesc (MQMD1)

これは組み込みメッセージ記述子であり、メッセージが最初にリモート・キューに書き込まれたときに MQPUT 呼び出しまたは MQPUT1 呼び出しで **MsgDesc** パラメーターとして指定されたメッセージ記述子 MQMD とほぼ同じコピーです。

注: これはバージョン 1 の MQMD です。

フィールドの初期値は、MQMD 構造体の中のものと同じです。

関数呼び出し

利用可能なすべての MQI 呼び出しに関する情報。記述、構文、パラメーター情報、使用上の注意、および可能な言語ごとの言語呼び出しが、各種の呼び出しごとに提供されています。

  これらの呼び出しについては、オンラインヘルプがおよび Linux プラットフォーム AIX 上で ページ man 形式で提供されています。

注: データ変換に関連する呼び出し、MQXCNCV および MQ_DATA_CONV_EXIT は、[903 ページの『データ変換出口』](#) で説明されています。

C 言語での呼び出しの使用

入力専用で型が MQHCONN、MQHOBJ、MQHMSG、または MQLONG のパラメータは値渡しで渡されます。その他のすべてのパラメータについては、パラメータのアドレスが値渡しで渡されます。

アドレスによって渡されるパラメーターは、必ずしも関数を呼び出すたびに指定する必要はありません。特に必要なパラメーターがない場合は、パラメーター・データのアドレスの代わりに、ヌル・ポインターを関数呼び出しのパラメーターとして指定することができます。これが可能なパラメーターは、呼び出し記述子で識別されます。

呼び出しの値として、パラメーターが戻されることはありません。つまり C 用語では、すべての呼び出しが void を戻すことを意味します。

バッファ・パラメーターの宣言

MQGET、**MQPUT**、および **MQPUT1** の各呼び出しには、未定義データ・タイプを持つパラメーター、つまり **Buffer** パラメーターが1つずつあります。このパラメーターは、アプリケーションのメッセージ・データの送受信に使用します。

この種類のパラメーターは、C の例で **MQBYTE** の配列として示されています。この方法でパラメーターを宣言することは可能ですが、通常は、メッセージのデータ・レイアウトを記述する特定の構造体として宣言する方が便利です。呼び出しパラメーターを **void** を示すポインターとして宣言することで、任意の種類のデータのアドレスを呼び出し方式のパラメーターとして指定することができます。 **void** を示すポインターは、不定形式データを指し示すポインターです。以下のように定義されます。

```
typedef void *PMQVOID;
```

関連資料

 [MQI 呼び出しによる CEDF 出力の例](#)

呼び出しの記述

このセクションでは、MQI 呼び出しについて説明します。

- [620 ページの『MQBACK - バックアウトの変更』](#)
- [624 ページの『MQBEGIN - 作業単位の開始』](#)
- [628 ページの『MQBUFMH - バッファからメッセージ・ハンドルへの変換』](#)
- [631 ページの『MQCB - コールバック管理』](#)
- [641 ページの『MQCB FUNCTION - コールバック関数』](#)
- [642 ページの『MQCLOSE - オブジェクトのクローズ』](#)
- [650 ページの『MQCMIT - 変更のコミット』](#)
- [654 ページの『MQCONN - キュー・マネージャーの接続』](#)
- [662 ページの『MQCONNEX - キュー・マネージャーの接続 \(拡張\)』](#)
- [667 ページの『MQCRTMH - メッセージ・ハンドルの作成』](#)
- [671 ページの『MQCTL - コールバック制御』](#)
- [677 ページの『MQDISC - キュー・マネージャーの切断』](#)
- [681 ページの『MQDLTMH - メッセージ・ハンドルの削除』](#)
- [683 ページの『MQDLTMP - メッセージ・プロパティの削除』](#)
- [685 ページの『MQGET - メッセージの読み取り』](#)
- [698 ページの『MQINQ - オブジェクト属性の照会』](#)
- [715 ページの『MQINQMP - メッセージ・プロパティの照会』](#)
- [720 ページの『MQMHBUF - メッセージ・ハンドルのバッファへの変換』](#)
- [724 ページの『MQOPEN - オブジェクトのオープン』](#)
- [743 ページの『MQPUT - メッセージの書き込み』](#)
- [756 ページの『MQPUT1 - 1 つのメッセージの書き込み』](#)
- [767 ページの『MQSET - オブジェクト属性の設定』](#)
- [773 ページの『MQSETMP - メッセージ・プロパティの設定』](#)
- [777 ページの『MQSTAT - 状況情報の取り出し』](#)
- [720 ページの『MQMHBUF - メッセージ・ハンドルのバッファへの変換』](#)
- [781 ページの『MQSUB - サブスクリプションの登録』](#)
- [788 ページの『MQSUBRQ - サブスクリプション要求』](#)

UNIX プラットフォーム上のオンライン・ヘルプは、*man* ページの形式で提供され、これらの呼び出しに利用できます。

注: データ変換に関連する呼び出し (MQXCNCV および MQ_DATA_CONV_EXIT) については、[903 ページの『データ変換出口』](#)に説明があります。

呼び出しの記述で使用される規則

このトピックのコレクションでは、各呼び出しについて、プログラミング言語とは独立した形式で、パラメーターの記述と呼び出しの使用法を説明します。そのあとで、サポートされているプログラミング言語を1つずつとりあげて、一般的な呼び出し方式と呼び出しに指定するパラメーターの一般的な宣言を説明します。

重要: IBM MQ API 呼び出しをコーディングする際は、すべての関連パラメーターを確実に指定する (以下の各セクションで説明されているように) 必要があります。これを行わないと、予測不能の結果になることがあります。

呼び出しの個別説明は、以下の形式で行います。

呼び出し名

呼び出し名。このあとに呼び出しの目的についての簡単な説明が続きます。

パラメーター

各パラメーターについて、名前のあとにそのデータ・タイプが括弧 () 内に示されています。データ・タイプは、次のいずれかです。

入力

呼び出しを行うときに、このパラメーターに情報を指定します。

出力

呼び出しが完了または失敗したときに、キュー・マネージャーがこのパラメーターに情報を戻します。

入出力

呼び出しを行うときには、このパラメーターに情報を指定し、呼び出しが完了または失敗したときにはキュー・マネージャーが情報を変更します。

以下に例を示します。

Compcode (MQLONG) - 出力

データ・タイプが構造体である場合もあります。いずれの場合も、データ・タイプまたは構造体の詳細については、[177 ページの『基本データ・タイプ』](#)に説明があります。

各呼び出しの最後の2つのパラメーターは、完了コードと理由コードです。完了コードは、呼び出しが正しく完了した、一部だけ完了した、または全く完了しなかった、のいずれかを示します。呼び出しが一部成功したまたは失敗した場合、その詳細は理由コードに示されます。各完了コードと理由コードについて詳しくは、[869 ページの『戻りコード』](#)を参照してください。

使用上の注意

呼び出しに関する追加情報。その使用方法や使用上の制約事項について説明します。

アセンブラー言語呼び出し

アセンブラー言語での、一般的な呼び出し方式およびそのパラメーターの宣言。

C 言語での呼び出し

C での、一般的な呼び出し方式およびそのパラメーターの宣言。

COBOL での呼び出し

COBOL での、一般的な呼び出し方式およびそのパラメーターの宣言。

PL/I での呼び出し

PL/I での、一般的な呼び出し方式およびそのパラメーターの宣言。

パラメーターはすべて、参照によって渡されます。

Visual Basic での呼び出し

Visual Basic での、一般的な呼び出し方式およびそのパラメーターの宣言。

その他の表記規則は以下のとおりです。

定数

定数の名前は、大文字で示されています。例えば、MQOO_OUTPUT。同じ接頭部を持つ定数のセットは、次のように表示されます。MQIA_*。定数の値については、[7 ページの『定数』](#)を参照してください。

配列

一部の呼び出しでは、パラメーターが、固定されたサイズのない文字ストリングの配列です。これらのパラメーターの記述の中では、小文字の n が数字の定数を表します。そのパラメーターの宣言をエンコードするときは、必要な数値で n を置き換えます。

C 言語での呼び出しの使用

入力専用 のパラメーターおよび MQHCONN、MQHOBJ、MQHMSG、または MQLONG タイプのパラメーターは、値によって渡されます。それ以外のすべてのパラメーターの場合、パラメーターのアドレスは値によって渡されます。

アドレスによって渡されるパラメーターは、必ずしも関数を呼び出すたびに指定する必要はありません。特に必要なパラメーターがない場合は、パラメーター・データのアドレスの代わりに、ヌル・ポインターを関数呼び出しのパラメーターとして指定することができます。これが可能なパラメーターは、呼び出し記述子で識別されます。

呼び出しの値として、パラメーターが戻されることはありません。つまり C 用語では、すべての呼び出しが void を戻すことを意味します。

バッファー・パラメーターの宣言

MQGET、**MQPUT**、および **MQPUT1** の各呼び出しには、未定義データ・タイプを持つパラメーター、つまり *Buffer* パラメーターが 1 つずつあります。このパラメーターは、アプリケーションのメッセージ・データの送受信に使用します。

この種類のパラメーターは、C の例で **MQBYTE** の配列として示されています。この方法でパラメーターを宣言することは可能ですが、通常は、メッセージのデータ・レイアウトを記述する特定の構造体として宣言の方が便利です。呼び出しパラメーターを void を示すポインターとして宣言することで、任意の種類のデータのアドレスを呼び出し方式のパラメーターとして指定することができます。

void を示すポインターは、不定形式データを指し示すポインターです。以下のように定義されます。

```
typedef void *PMQVOID;
```

MQBACK - バックアウトの変更

MQBACK 呼び出しは、最後の同期点以降に発生したメッセージの読み取りと書き込みをすべてバックアウトすることをキュー・マネージャーに示します。

作業単位の一部として書き込まれたメッセージは削除されます。作業単位の一部として取り出されたメッセージはキューに戻されます。

- z/OS では、この呼び出しはバッチ・プログラム (IMS バッチ DL/I プログラムを含む) でのみ使用されます。

構文

MQBACK (*Hconn*, *Compcode*, *Reason*)

パラメーター

Hconn

タイプ: MQHCONN - 入力

このハンドルは、キュー・マネージャーに対する接続を表します。*Hconn* の値は、先行の MQCONN または MQCONNX 呼び出しによって戻されたものです。

ComPCODE

タイプ: MQLONG - 出力

完了コード。以下のいずれかです。

MQCC_OK

正常終了。

MQCC_WARNING

警告 (部分完了)。

MQCC_FAILED

呼び出し失敗。

理由

タイプ: MQLONG - 出力

CompCode が MQCC_OK の場合:

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_WARNING の場合:

MQRC_OUTCOME_PENDING

(2124, X'84C') バックアウト操作の結果が保留状態である。

CompCode が MQCC_FAILED の場合:

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852') アダプター・サービス・モジュールをロードできません。

MQRC_API_EXIT_ERROR

(2374, X'946') API 出口で障害が発生しました。

MQRC_ASID_MISMATCH

(2157, X'86D') 1 次 ASID とホーム ASID が異なっています。

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') 前の呼び出しが完了する前に MQI 呼び出しが入力されました。

MQRC_CF_STRUC_IN_USE

(2346, X'92A') カップリング・ファシリティ構造体が使用中です。

MQRC_CONNECTION_BROKEN

(2009, X'7D9') キュー・マネージャーとの接続が失われました。

MQRC_ENVIRONMENT_ERROR

(2012, X'7DC') この環境では呼び出しが無効です。

MQRC_HCONN_ERROR

(2018, X'7E2') 接続ハンドルが無効です。

MQRC_OBJECT_DAMAGED

(2101, X'835') オブジェクトが損傷しました。

MQRC_OUTCOME_MIXED

(2123, X'84B') コミットまたはバックアウト操作の結果が混在している。

MQRC_Q_MGR_STOPPING

(2162, X'872') キュー・マネージャーのシャットダウン中です。

MQRC_RESOURCE_PROBLEM

(2102, X'836') 使用できるシステム・リソースが不足しています。

MQRC_STORAGE_MEDIUM_FULL

(2192, X'890') 外部ストレージ・メディアが満杯です。

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') ストレージが不足しています。

MQRC_UNEXPECTED_ERROR

(2195, X'893') 予期しないエラーが発生しました。

これらのコードの詳細については、[メッセージおよび理由コード](#)を参照してください。

使用上の注意

1. この呼び出しは、キュー・マネージャーそのものが作業単位を調整するときのみ使用できます。次のタイプがあります。

- ローカル作業単位 (変更内容は MQ リソースにのみ影響を及ぼす)。
- グローバル作業単位 (変更内容が、MQ リソースだけでなく、他のリソース・マネージャーに属するリソースにも影響を及ぼす場合がある)。

ローカル作業単位およびグローバル作業単位の詳細については、[624 ページの『MQBEGIN - 作業単位の開始』](#)を参照してください。

2. キュー・マネージャーが作業単位を調整しない環境では、MQBACK ではなく適切なバックアウト呼び出しを使用してください。この環境ではまた、アプリケーションの異常終了を原因とする暗黙的バックアウトをサポートすることもできます。

- z/OS では、以下の呼び出しを使用してください。
 - 作業単位が MQ リソースに対してだけ影響を及ぼす場合は、バッチ・プログラム (IMS バッチ DL/I プログラムを含む) で MQBACK 呼び出しを使用できます。ただし、作業単位が MQ リソースだけでなく他のリソース・マネージャー (Db2® など) に属するリソースにも影響を及ぼす場合には、z/OS Recoverable Resource Service (RRS) が提供する SRRBACK 呼び出しを使用してください。SRRBACK 呼び出しを実行すると、RRS 調整対応のリソース・マネージャーに属するリソースに対する変更がバックアウトされます。
 - CICS アプリケーションは、EXEC CICS SYNCPOINT ROLLBACK コマンドを使用して作業単位をバックアウトする必要があります。CICS アプリケーションには MQBACK 呼び出しを使用しないでください。
 - IMS アプリケーション (バッチ DL/I プログラム以外) は、ROLB などの IMS 呼び出しを使用して、作業単位をバックアウトする必要があります。IMS アプリケーション (バッチ DL/I プログラム以外) には、MQBACK 呼び出しを使用しないでください。
- IBM i では、この呼び出しはキュー・マネージャーで調整されるローカル作業単位で使用してください。これは、コミットメント定義がジョブ・レベルで存在してはならないことを意味します。つまり、**CMTSCOPE(*JOB)** パラメーターを指定した STRCMTCTL コマンドがジョブに対して発行されているわけではありません。

3. 作業単位内にあるコミットされていない変更内容でアプリケーションが終了する場合、それらの変更内容の後処理は、そのアプリケーションが正常に終了するか、異常終了するかで異なります。詳細については、[677 ページの『MQDISC - キュー・マネージャーの切断』](#)の使用上の注意を参照してください。

4. アプリケーションでグループ内のメッセージまたは論理メッセージのセグメントの書き込みまたは読み取りを行う場合、キュー・マネージャーは、最後に MQPUT および MQGET 呼び出しが正常に実行されたメッセージ・グループに関する情報を保存します。この情報は、キュー・ハンドルに関する次のような情報です。

- MQMD 中の *GroupId*、*MsgSeqNumber*、*Offset*、および *MsgFlags* フィールドの値。
- そのメッセージが作業単位の一部かどうか。
- MQPUT 呼び出しについて、そのメッセージが持続メッセージか、非持続メッセージか。

キュー・マネージャーは、次のものについて 1 つずつ、3 セットのグループおよびセグメント情報を保持しています。

- 最後に正常に実行された MQPUT 呼び出し (これは作業単位の一部である場合があります)。
- 最後に正常に実行された MQGET 呼び出しのうちキューからメッセージを削除したもの (作業単位の一部である場合があります)。

- 最後に正常に実行された MQGET 呼び出しのうちキュー上のメッセージをブラウズしたもの (これが作業単位の一部であることはありません)。

5. MQGET 呼び出しについての情報は、現行作業単位内のそのキュー・ハンドルについて最初に正常に実行された MQGET 呼び出し以前の値に復元されます。

作業単位の開始後にアプリケーションによって更新されたキューであっても、それが作業単位の有効範囲外である場合は、その作業単位がバックアウトされても、グループおよびセグメント情報は復元されません。

作業単位のバックアウト時にグループおよびセグメント情報を以前の値に復元する機能により、アプリケーションは、数多くのセグメントで構成される大きなメッセージ・グループまたは大きな論理メッセージをいくつかの作業単位にまたがって広げることができます。そして、いずれかの作業単位が失敗しても、そのメッセージ・グループまたは論理メッセージ内の正しい点でアプリケーションを再始動できます。

ローカル・キュー・マネージャーのキュー・ストレージが限られている場合には、いくつかの作業単位を使用する方が有効となる場合があります。ただし、システム障害の発生時に各メッセージの書き込みまたは読み取りを正しい時点で再始動できるようにするには、アプリケーションが十分な情報を維持している必要があります。

システム障害後に正しい時点から再始動する方法の詳細については、472 ページの『MQPMO - メッセージ書き出しオプション』で説明している MQPMO_LOGICAL_ORDER オプションと、326 ページの『MQGMO - 読み取りメッセージ・オプション』で説明している MQGMO_LOGICAL_ORDER オプションを参照してください。

次の使用上の注意は、キュー・マネージャーで作業単位を調整する場合にのみ適用されます。

6. 作業単位の 1 つには、1 つの接続ハンドルと同じ有効範囲があります。特定の作業単位に影響を与えるすべての MQ 呼び出しは、同じ接続ハンドルを使用して実行しなければなりません。別の接続ハンドルを用いて呼び出しを発行すると (例えば、別のアプリケーションで呼び出しを発行する)、別の作業単位に影響が及びます。接続ハンドルの有効範囲については、654 ページの『MQCONN - キュー・マネージャーの接続』で説明されている **Hconn** パラメーターを参照してください。
7. この呼び出しで影響を受けるメッセージは、現行の作業単位の一部として書き込まれたメッセージ、または取り出されたメッセージに限られます。
8. 長時間実行しているアプリケーションが、1 つの作業単位に対して MQGET、MQPUT、または MQPUT1 呼び出しを発行する一方、コミット呼び出しまたはバックアウト呼び出しを一度も発行しない場合には、キューが他のアプリケーションでは使用できないメッセージで満杯になることがあります。この可能性を回避するために、管理者は、**MaxUncommittedMsgs** キュー・マネージャー属性を、ランナウェイ・アプリケーションがキューを満杯にしないように十分に低い値に設定する必要がありますが、予期されるメッセージング・アプリケーションが正しく機能するように十分に高い値に設定する必要があります。

C 言語での呼び出し

```
MQBACK (Hconn, &CompCode, &Reason);
```

パラメーターを次のように宣言します。

```
MQHCONN  Hconn;      /* Connection handle */
MQLONG   CompCode;   /* Completion code */
MQLONG   Reason;     /* Reason code qualifying CompCode */
```

COBOL での呼び出し

```
CALL 'MQBACK' USING HCONN, COMPCODE, REASON.
```

パラメーターを次のように宣言します。

```
**      Connection handle
01  HCONN      PIC S9(9) BINARY.
```

```

** Completion code
01 COMPCODE PIC S9(9) BINARY.
** Reason code qualifying COMPCODE
01 REASON PIC S9(9) BINARY.

```

PL/I での呼び出し

```
call MQBACK (Hconn, CompCode, Reason);
```

パラメーターを次のように宣言します。

```

dcl Hconn      fixed bin(31); /* Connection handle */
dcl CompCode   fixed bin(31); /* Completion code */
dcl Reason     fixed bin(31); /* Reason code qualifying CompCode */

```

高水準アセンブラー呼び出し

```
CALL MQBACK,(HCONN,COMPCODE,REASON)
```

パラメーターを次のように宣言します。

```

HCONN      DS  F  Connection handle
COMPCODE   DS  F  Completion code
REASON     DS  F  Reason code qualifying COMPCODE

```

Visual Basic での呼び出し

```
MQBACK Hconn, CompCode, Reason
```

パラメーターを次のように宣言します。

```

Dim Hconn      As Long 'Connection handle'
Dim CompCode   As Long 'Completion code'
Dim Reason     As Long 'Reason code qualifying CompCode'

```

MQBEGIN - 作業単位の開始

MQBEGIN 呼び出しは、キュー・マネージャーによって調整される作業単位を開始します。また、この作業単位は外部リソース・マネージャーを伴うこともあります。

構文

```
MQBEGIN (Hconn, BeginOptions, Compcode, Reason)
```

パラメーター

Hconn

タイプ: MQHCONN - 入力

このハンドルは、キュー・マネージャーに対する接続を表します。Hconn の値は、先行の MQCONN または MQCONNXX 呼び出しによって戻されたものです。

Hconn は、非共有接続ハンドルでなければなりません。共有接続ハンドルが指定されると、呼び出しが失敗し、理由コード MQRC_HCONN_ERROR が戻されます。共有ハンドルと非共有ハンドルについては、265 ページの『MQCNO - 接続オプション』にある MQCNO_HANDLE_SHARE_* オプションの説明を参照してください。

BeginOptions

タイプ: MQBO - 入出力

これらは、MQBEGIN のアクションを制御するオプションです。222 ページの『MQBO - 開始オプション』で説明されています。

必須オプションがない場合、C アセンブラーまたは S/390 アセンブラーで作成されたプログラムでは、MQBO 構造体のアドレスを指定せずに、ヌル・パラメーター・アドレスを指定することができます。

CompCode

タイプ: MQLONG - 出力

完了コード。以下のいずれかです。

MQCC_OK

正常終了。

MQCC_WARNING

警告 (部分完了)。

MQCC_FAILED

呼び出し失敗。

理由

タイプ: MQLONG - 出力

CompCode が MQCC_OK の場合:

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_WARNING の場合:

MQRC_NO_EXTERNAL_PARTICIPANTS

(2121, X'849') 参加するリソース・マネージャーが登録されていない。

MQRC_PARTICIPANT_NOT_AVAILABLE

(2122, X'84A') 参加するリソース・マネージャーが利用不能である。

CompCode が MQCC_FAILED の場合:

MQRC_API_EXIT_ERROR

(2374, X'946') API 出口で障害が発生しました。

MQRC_BO_ERROR

(2134, X'856') 開始オプション構造体が無効である。

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') 前の呼び出しが完了する前に MQI 呼び出しが入力されました。

MQRC_CONNECTION_BROKEN

(2009, X'7D9') キュー・マネージャーとの接続が失われました。

MQRC_ENVIRONMENT_ERROR

(2012, X'7DC') この環境では呼び出しが無効です。

MQRC_HCONN_ERROR

(2018, X'7E2') 接続ハンドルが無効です。

MQRC_OPTIONS_ERROR

(2046, X'7FE') オプションが無効であるか、矛盾しています。

MQRC_Q_MGR_STOPPING

(2162, X'872') キュー・マネージャーのシャットダウン中です。

MQRC_RESOURCE_PROBLEM

(2102, X'836') 使用できるシステム・リソースが不足しています。

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') ストレージが不足しています。

MQRC_UNEXPECTED_ERROR

(2195, X'893') 予期しないエラーが発生しました。

MQRC_UOW_IN_PROGRESS

(2128, X'850') 作業単位が開始済みである。

これらのコードについて詳しくは、[メッセージおよび理由コード](#)を参照してください。

使用上の注意

- MQBEGIN 呼び出しは、キュー・マネージャーで調整される作業単位の開始に使用します。また、この作業単位で他のリソース・マネージャー所有のリソースへの変更を行うこともあります。キュー・マネージャーは、次の 3 つのタイプの作業単位をサポートします。
 - ・ **キュー・マネージャーで調整されるローカル作業単位:** 参加するリソース・マネージャーがキュー・マネージャーだけである作業単位。したがって、キュー・マネージャーが作業単位コーディネーターとして機能します。
 - このタイプの作業単位を開始するには、作業単位内の最初の MQPUT、MQPUT1、または MQGET 呼び出しに MQPMO_SYNCPOINT または MQGMO_SYNCPOINT オプションを指定してください。
 - このタイプの作業単位をコミットまたはバックアウトするには、MQCMIT または MQBACK 呼び出しを使用します。
 - ・ **キュー・マネージャーによって調整されるグローバル作業単位:** キュー・マネージャーが、MQ リソースおよび他のリソース・マネージャーに属するリソースの両方に対して作業単位コーディネーターとして機能する作業単位。これらのリソース・マネージャーは、キュー・マネージャーと連携して、必ず作業単位内のリソースへのすべての変更内容が一度にコミットまたはバックアウトされるようにします。
 - このタイプの作業単位を開始するには、MQBEGIN 呼び出しを使用します。
 - このタイプの作業単位をコミットまたはバックアウトするには、MQCMIT および MQBACK 呼び出しを使用します。
 - ・ **外部で調整されるグローバル作業単位:** キュー・マネージャーが参加しているが、そのキュー・マネージャーが作業単位コーディネーターとしては機能しない作業単位。その代わりに、キュー・マネージャーと連携する外部作業単位コーディネーターが存在します。
 - このタイプの作業単位を開始するには、外部作業単位コーディネーターが提供する関連呼び出しを使用します。

作業単位を開始するために MQBEGIN 呼び出しを使用すると失敗し、理由コード MQRC_ENVIRONMENT_ERROR が戻ります。
 - このタイプの作業単位をコミットまたはバックアウトするには、外部作業単位コーディネーターが提供するコミット呼び出しおよびバックアウト呼び出しを使用します。

作業単位をコミットまたはバックアウトするために MQCMIT または MQBACK 呼び出しを使用すると失敗し、理由コード MQRC_ENVIRONMENT_ERROR が戻ります。
- 作業単位内にあるコミットされていない変更内容でアプリケーションが終了する場合、それらの変更内容の後処理は、そのアプリケーションが正常に終了するか、異常終了するかで異なります。詳細については、677 ページの『MQDISC - キュー・マネージャーの切断』の使用上の注意を参照してください。
- アプリケーションが一度に参加プログラムとしてかわることができる作業単位は、1 つだけです。アプリケーションで MQBEGIN 呼び出しを発行する場合、そのアプリケーション用の作業単位がすでに存在していると、その呼び出しは、作業単位のタイプに関係なく失敗し、理由コード MQRC_UOW_IN_PROGRESS が戻ります。
- MQBEGIN 呼び出しは、MQ MQI クライアント環境では無効となります。この呼び出しを使用しようとすると失敗し、理由コード MQRC_ENVIRONMENT_ERROR が戻ります。
- キュー・マネージャーが各グローバル作業単位の作業単位コーディネーターとして機能している場合、作業単位に参加プログラムとしてかわることができるリソース・マネージャーは、キュー・マネージャーの構成ファイル内に定義されます。
- IBM i では、次の 3 つのタイプの作業単位がサポートされています。

- ・キュー管理者が調整するローカル作業単位は、ジョブレベルでコミットメント定義が存在しない場合にのみ使用できます。つまり、そのジョブに対してパラメータ **CMTSCOPE(*JOB)** 付きのコマンド STRCMTCTL が発行されていない必要があります。
- ・キュー・マネージャーで調整されるグローバル作業単位は、サポートされていません。
- ・外部調整型グローバル作業単位は、ジョブレベルでコミットメント定義が存在する場合にのみ使用可能である。すなわち、そのジョブに対してパラメータ **CMTSCOPE(*JOB)** 付き STRCMTCTL コマンドが発行済みである必要がある。それが実行されている場合、IBM i の COMMIT および ROLLBACK 操作が、MQ リソースと他の参加しているリソース・マネージャーのリソースに対して適用されます。

C 言語での呼び出し

```
MQBEGIN (Hconn, &BeginOptions, &CompCode, &Reason);
```

パラメーターを次のように宣言します。

```
MQHCONN  Hconn;           /* Connection handle */
MQBO     BeginOptions;    /* Options that control the action of MQBEGIN */
MQLONG   CompCode;        /* Completion code */
MQLONG   Reason;          /* Reason code qualifying CompCode */
```

COBOL での呼び出し

```
CALL 'MQBEGIN' USING HCONN, BEGINOPTIONS, COMPCODE, REASON.
```

パラメーターを次のように宣言します。

```
**      Connection handle
01  HCONN          PIC S9(9) BINARY.
**      Options that control the action of MQBEGIN
01  BEGINOPTIONS.
    COPY CMQBOV.
**      Completion code
01  COMPCODE       PIC S9(9) BINARY.
**      Reason code qualifying COMPCODE
01  REASON         PIC S9(9) BINARY.
```

PL/I での呼び出し

```
call MQBEGIN (Hconn, BeginOptions, CompCode, Reason);
```

パラメーターを次のように宣言します。

```
dcl Hconn          fixed bin(31); /* Connection handle */
dcl BeginOptions   like MQBO;     /* Options that control the action of
                                   MQBEGIN */
dcl CompCode       fixed bin(31); /* Completion code */
dcl Reason         fixed bin(31); /* Reason code qualifying CompCode */
```

Visual Basic での呼び出し

```
MQBEGIN Hconn, BeginOptions, CompCode, Reason
```

パラメーターを次のように宣言します。

```
Dim Hconn          As Long 'Connection handle'
Dim BeginOptions   As MQBO 'Options that control the action of MQBEGIN'
Dim CompCode       As Long 'Completion code'
Dim Reason         As Long 'Reason code qualifying CompCode'
```


MQBUFMH - バッファからメッセージ・ハンドルへの変換

MQBUFMH 関数呼び出しは、バッファをメッセージ・ハンドルに変換するので、MQMHBUF 呼び出しの逆です。

この呼び出しは、メッセージ記述子と、バッファ内の MQRFH2 プロパティを取り、メッセージ・ハンドルを使用してそれらを使用可能にします。メッセージ・データの MQRFH2 プロパティは、オプションで除去されます。メッセージ記述子の *Encoding*、*CodedCharSetId*、および *Format* フィールドは、必要であればプロパティが除去された後に更新され、バッファの内容が正しく記述されます。

構文

MQBUFMH (*Hconn*, *Hmsg*, *BufMsgHOpts*, *MsgDesc*, *BufferLength*, *Buffer*, *DataLength*, *Compcode*, *Reason*)

パラメーター

Hconn

タイプ: MQHCONN - 入力

このハンドルは、キュー・マネージャーに対する接続を表します。**Hconn** の値は、**Hmsg** パラメーターで指定されているメッセージ・ハンドルの作成に使用された接続ハンドルと一致していなければなりません。

MQHC_UNASSOCIATED_HCONN を使用してメッセージ・ハンドルが作成された場合は、バッファをメッセージ・ハンドルに変換するスレッド上で有効な接続を確立しなければなりません。有効な接続を確立しないと、呼び出しは MQRC_CONNECTION_BROKEN で失敗します。

Hmsg

タイプ: MQHMSG - 入力

これは、バッファが必要なメッセージ・ハンドルです。値は、前の MQCRTMH 呼び出しで戻されたものです。

BufMsgHOpts

タイプ: MQBMHO - 入力

アプリケーションでは、MQBMHO 構造体を使用することによって、バッファからメッセージ・ハンドルを生成する方法を制御するためのオプションを指定することができます。

詳細は [217 ページの『MQBMHO - バッファからメッセージ・ハンドルへの変換オプション』](#)を参照してください。

MsgDesc

タイプ: MQMD - 入出力

MsgDesc 構造体には、メッセージ記述子プロパティが含まれ、バッファ域の内容を記述します。

呼び出しからの出力上で、オプションでプロパティがバッファ域から除去されます。この場合、メッセージ記述子が更新され、バッファ域は正しく記述されます。

この構造体中のデータは、アプリケーションの文字セット内およびエンコード内になければなりません。

BufferLength

タイプ: MQLONG - 入力

BufferLength は、バッファ域の長さです (バイト単位)。

ゼロ・バイトの *BufferLength* は有効であり、バッファ域にデータが入っていないことを示します。

Buffer

タイプ: MQBYTEExBufferLength - 入出力

Buffer は、メッセージ・バッファが入れられる領域を定義します。ほとんどのデータの場合、バッファを 4 バイトの境界に位置合わせする必要があります。

データが文字または数値データを含む **Buffer** 場合、パラメータ **MsgDesc** の `CodedCharSetId` および `Encoding` フィールドをデータに適した値に設定してください。これにより、必要に応じてデータを変換できるようになります。

メッセージ・バッファ中にプロパティがある場合はオプションで除去されます。これらのプロパティは、後で呼び出しから戻す際にメッセージ・ハンドルから使用できるようになります。

C プログラミング言語では、パラメーターは、`void` を示すポインターとして宣言されます。つまり、どのタイプのデータのアドレスもパラメーターとして指定できます。

BufferLength パラメーターがゼロの場合は、**Buffer** は参照されません。この場合、C または System/390 アセンブラーで作成されたプログラムによって渡されるパラメーター・アドレスはヌルのこともあります。

DataLength

タイプ: MQLONG - 出力

プロパティが削除された可能性があるバッファのバイト単位の長さです。

CompCode

タイプ: MQLONG - 出力

完了コード。以下のいずれかです。

MQCC_OK

正常終了。

MQCC_FAILED

呼び出し失敗。

理由

タイプ: MQLONG - 出力

CompCode が MQCC_OK の場合:

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_FAILED の場合:

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'089C') アダプターが利用できません。

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852') アダプター・サービス・モジュールをロードできません。

MQRC_ASID_MISMATCH

(2157, X'86D') 1 次 ASID とホーム ASID が異なります。

MQRC_BMHO_ERROR

(2489, X'09B9') バッファからメッセージ・ハンドルへの変換オプション構造体が無効です。

MQRC_BUFFER_ERROR

(2004, X'07D4') バッファ・パラメーターが無効である。

MQRC_BUFFER_LENGTH_ERROR

(2005, X'07D5') バッファ長パラメーターは無効です。

MQRC_CALL_IN_PROGRESS

(2219, X'08AB') 前の呼び出しが完了する前に MQI 呼び出しが入力された。

MQRC_CONNECTION_BROKEN

(2009, X'07D9') キュー・マネージャーとの接続が失われました。

MQRC_HMSG_ERROR

(2460, X'099C') メッセージ・ハンドルが無効。

MQRC_MD_ERROR

(2026, X'07EA') メッセージ記述子が無効である。

MQRC_MSG_HANDLE_IN_USE

(2499, X'09C3') メッセージ・ハンドルがすでに使用中。

MQRC_OPTIONS_ERROR

(2046, X'07FE') オプションが無効であるか、矛盾しています。

MQRC_RFH_ERROR

(2334, X'091E') MQRFH2 構造体が無効である。

MQRC_RFH_FORMAT_ERROR

(2421, X'0975') プロパティを含む MQRFH2 フォルダーを構文解析できなかった。

MQRC_UNEXPECTED_ERROR

(2195, X'893') 予期しないエラーが発生しました。

これらのコードについて詳しくは、[メッセージおよび理由コード](#)を参照してください。

使用上の注意

MQBUFMH 呼び出しは、API 出口によってインターセプトできません。アプリケーションのスペース内でバッファはメッセージ・ハンドルに変換されます。この呼び出しはキュー・マネージャーに到達しません。

C 言語での呼び出し

```
MQBUFMH (Hconn, Hmsg, &BufMsgHOpts, &MsgDesc, BufferLength, Buffer,
         &DataLength, &CompCode, &Reason);
```

パラメーターを次のように宣言します。

```
MQHCONN  Hconn;           /* Connection handle */
MQHMSG    Hmsg;           /* Message handle */
MQBMHO    BufMsgHOpts;    /* Options that control the action of MQBUFMH */
MQMD      MsgDesc;        /* Message descriptor */
MQLONG    BufferLength;    /* Length in bytes of the Buffer area */
MQBYTE    Buffer[n];       /* Area to contain the message buffer */
MQLONG    DataLength;     /* Length of the output buffer */
MQLONG    CompCode;       /* Completion code */
MQLONG    Reason;         /* Reason code qualifying CompCode */
```

COBOL での呼び出し

```
CALL 'MQBUFMH' USING HCONN, HMSG, BUFMSGHOPTS, MSGDESC, BUFFERLENGTH,
                     BUFFER, DATALENGTH, COMPCODE, REASON.
```

パラメーターを次のように宣言します。

```
** Connection handle
01 HCONN          PIC S9(9) BINARY.
** Message handle
01 HMSG           PIC S9(18) BINARY.
** Options that control the action of MQBUFMH
01 BUFMSGHOPTS.
   COPY CMQBMHOV.
** Message descriptor
01 MSGDESC.
   COPY CMQMD.
** Length in bytes of the Buffer area
01 BUFFERLENGTH  PIC S9(9) BINARY.
** Area to contain the message buffer
01 BUFFER        PIC X(n).
** Length of the output buffer
01 DATALENGTH   PIC S9(9) BINARY.
** Completion code
01 COMPCODE      PIC S9(9) BINARY.
** Reason code qualifying COMPCODE
01 REASON        PIC S9(9) BINARY.
```

PL/I での呼び出し

```
call MQBUFMH (Hconn, Hmsg, BufMsgHOpts, MsgDesc, BufferLength, Buffer,
DataLength, CompCode, Reason);
```

パラメーターを次のように宣言します。

```
dc1 Hconn          fixed bin(31); /* Connection handle */
dc1 Hmsg           fixed bin(63); /* Message handle */
dc1 BufMsgHOpts    like MQBMHO;  /* Options that control the action of
                                MQBUFMH */
dc1 MsgDesc        like MQMD;    /* Message descriptor */
dc1 BufferLength    fixed bin(31); /* Length in bytes of the Buffer area */
dc1 Buffer          char(n);      /* Area to contain the message buffer */
dc1 DataLength     fixed bin(31); /* Length of the output buffer */
dc1 CompCode       fixed bin(31); /* Completion code */
dc1 Reason         fixed bin(31); /* Reason code qualifying CompCode */
```

高水準アセンブラー呼び出し

```
CALL MQBUFMH, (HCONN,HMSG,BUFMSGHOPTS,MSGDESC,BUFFERLENGTH,BUFFER,
DATALENGTH,COMPCODE,REASON)
```

パラメーターを次のように宣言します。

HCONN	DS	F	Connection handle
HMSG	DS	D	Message handle
BUFMSGHOPTS	CMQBMHOA	,	Options that control the action of MQBUFMH
MSGDESC	CMQMDA	,	Message descriptor
BUFFERLENGTH	DS	F	Length in bytes of the BUFFER area
BUFFER	DS	CL(n)	Area to contain the properties
DATALENGTH	DS	F	Length of the output buffer
COMPCODE	DS	F	Completion code
REASON	DS	F	Reason code qualifying COMPCODE

MQCB - コールバック管理

MQCB 呼び出しは、指定されたオブジェクト・ハンドルに関するコールバックを再登録し、コールバックに対するアクティベーションと変更を制御します。

コールバックとは、特定のイベントが発生した時点で IBM MQ によって呼び出されるコードの断片 (動的にリンクできる関数の名前か関数ポインターのいずれかとして指定される) のことです。

クライアントで MQCB および MQCTL を使用するには、ネゴシエーションが行われたチャネルの **SHARECNV** がゼロ以外の値と一致するサーバーに接続する必要があります。

定義できるコールバックのタイプは以下のとおりです。

メッセージ・コンシューマー

メッセージ・コンシューマー・コールバック関数は、指定された選択基準と一致するメッセージがオブジェクト・ハンドル上で使用可能な時点で呼び出されます。

各オブジェクト・ハンドルに対して登録できるコールバック関数は 1 つのみです。複数の選択基準を指定して単一のキューを読み取る場合は、そのキューを複数回オープンしなければならず、各ハンドル上に 1 つのコンシューマー関数が登録されています。

イベント・ハンドラー

コールバック環境全体に影響する条件に関するイベント・ハンドラーが呼び出されます。

キュー・マネージャーや接続の停止や静止などのイベント条件が発生すると、この関数が呼び出されます。

MQRC_GET_INHIBITED などの、単一のメッセージ・コンシューマーに固有の条件に関する機能は呼び出されませんが、コールバック関数が正常に終了しない場合は呼び出されます。

構文

MQCB (*Hconn*, *Operation*, *CallbackDesc*, *Hobj*, *MsgDesc*, *GetMsgOpts*, *CompCode*, *Reason*)

パラメーター

Hconn

タイプ: MQHCONN - 入力

このハンドルは、キュー・マネージャーに対する接続を表します。 *Hconn* の値は、先行の MQCONN または MQCONNX 呼び出しによって戻されたものです。

アプリケーション CICSz/OS では、この実行ユニットに関連付けられた接続ハンドルを使用 *MQHC_DEF_HCONN* するために、次の特別な値を指定できます。

Operation

タイプ: MQLONG - 入力

指定されたオブジェクト・ハンドルに定義されたコールバックで処理されている操作。次のいずれかのオプションを指定する必要があります。複数のオプションを指定するには、値を加算する (同じ定数を複数回加算しない) か、ビット単位 OR 演算を使用して値を結合 (プログラミング言語でビット演算がサポートされる場合) します。

MQOP_REGISTER

指定されたオブジェクト・ハンドルにコールバック関数を定義します。この操作は、呼び出される関数と、使用される選択基準を定義します。

オブジェクト・ハンドルに関するコールバック関数がすでに定義されている場合は、その定義は置き換えられます。コールバックを置き換えている間にエラーが検出されると、関数は登録解除されます。

以前にコールバックが登録解除されたコールバック関数でコールバックが登録される場合は、置き換え操作として扱われます。初期呼び出しまたは最終呼び出しは呼び出されません。

MQOP_REGISTER は、MQOP_SUSPEND または MQOP_RESUME と一緒に使用できます。

MQOP_DEREGISTER

オブジェクト・ハンドルのメッセージのコンシュームを停止し、このハンドルをコールバックに適切なものから除きます。

関連付けられているハンドルがクローズすると、コールバックは自動的に登録解除されます。

MQOP_DEREGISTER がコンシューマー中から呼び出され、コールバックで停止呼び出しが定義されている場合は、コンシューマーから戻る際に呼び出されます。

登録されたコンシューマーのない *Hobj* に対してこの操作が発行されると、呼び出しは MQRC_CALLBACK_NOT_REGISTERED で戻されます。

MQOP_SUSPEND

オブジェクト・ハンドルに関するメッセージのコンシュームを中断します。

この操作がイベント・ハンドラーに適用される場合は、中断している間にイベント・ハンドラーはイベントを読み取らず、中断状態の間に失われたイベントは再開時に操作に提供されません。

中断状態の間、コンシューマー関数は制御タイプのコールバックの取得を続行します。

MQOP_RESUME

オブジェクト・ハンドルに関するメッセージのコンシュームを再開します。

この操作がイベント・ハンドラーに適用される場合は、中断している間にイベント・ハンドラーはイベントを読み取らず、中断状態の間に失われたイベントは再開時に操作に提供されません。

CallbackDesc

タイプ: MQCBD - 入力

これは、アプリケーションによって登録されているコールバック関数と、登録時に使用されるオプションを識別する構造です。

この構造の詳細については、[MQCBD](#) を参照してください。

コールバック記述子が必須なのは `MQOP_REGISTER` オプションのみです。記述子が必須でない場合は、渡されるパラメーター・アドレスをヌルにすることができます。

Hobj

タイプ: `MQHOBJ` - 入力

このハンドルは、メッセージのコンシューム元のオブジェクトに対し設定されたアクセスを表します。これは、以前の [MQOPEN](#) または [MQSUB](#) 呼び出しから返されたハンドルです (パラメータ **Hobj** 内で)。

イベント・ハンドラー・ルーチン (`MQCBT_EVENT_HANDLER`) の定義時には、*Hobj* は必須ではなく、`MQHO_NONE` として指定する必要があります。

Hobj が `MQOPEN` 呼び出しから戻された場合には、キューは次の 1 つまたは複数のオプションでオープンしておく必要があります。

- `MQOO_INPUT_SHARED`
- `MQOO_INPUT_EXCLUSIVE`
- `MQOO_INPUT_AS_Q_DEF`
- `MQOO_BROWSE`

MsgDesc

タイプ: `MQMD` - 入力

この構造体は、必要なメッセージの属性と、取り出されるメッセージの属性を記述します。

MsgDesc パラメーターは、コンシューマーが必要とするメッセージの属性と、メッセージ・コンシューマーに渡される `MQMD` のバージョンを定義します。

`MQMD` 中で、*MsgId*、*CorrelId*、*GroupId*、*MsgSeqNumber*、および *Offset* が、**GetMsgOpts** パラメーターで指定されたオプションに応じて、メッセージの選択に使用されます。

`MQGMO_CONVERT` オプションを指定すると、*Encoding* と *CodedCharSetId* がメッセージの変換に使用されます。

詳細については、[MQMD](#) を参照してください。

MsgDesc は、`MQOP_REGISTER` で、いずれかのフィールドにデフォルト以外の値が必要な場合に使用されます。*MsgDesc* はイベント・ハンドラーには使用されません。

記述子が必須でない場合は、渡されるパラメーター・アドレスをヌルにすることができます。

複数のコンシューマーが、セクターがオーバーラップしている同一のキューに対して登録されている場合は、メッセージごとに選択されるコンシューマーが未定義になることに注意してください。

GetMsgOpts

タイプ: `MQGMO` - 入力

GetMsgOpts パラメーターは、メッセージ・コンシューマーがメッセージを取得する方法を制御します。`MQGET` 呼び出しで使用されている場合、このパラメーターのすべてのオプションは [326 ページの『MQGMO - 読み取りメッセージ・オプション』](#) で述べた意味を持ちますが、以下の例外があります。

MQGMO_SET_SIGNAL

このオプションは許可されていません。

MQGMO_BROWSE_FIRST、MQGMO_BROWSE_NEXT、MQGMO_MARK_*

ブラウズしているコンシューマーに配布されるメッセージの順序は、これらのオプションの組み合わせで指示されます。以下の組み合わせが有効です。

MQGMO_BROWSE_FIRST

キュー上の最初のメッセージが繰り返しコンシューマーに配布されます。このオプションは、コンシューマーがコールバック中のメッセージを破壊的にコンシュームする場合に便利です。このオプションは注意して使用してください。

MQGMO_BROWSE_NEXT

コンシューマーは、現行カーソル位置からキューの終わりに達するまで、キュー上の各メッセージを与えられます。

MQGMO_BROWSE_FIRST + MQGMO_BROWSE_NEXT

カーソルはキューの先頭にリセットされます。リセット後、カーソルがキューの終わりに達するまで、コンシューマーは各メッセージを与えられます。

MQGMO_BROWSE_FIRST + MQGMO_MARK_*

キューの先頭から始まって、コンシューマーはキュー上のマークが付いていない最初のメッセージを与えられ、その後このコンシューマー用にマークが付けられます。この組み合わせにより、コンシューマーは現行のカーソル・ポイントの後に追加された新しいメッセージを確実に受け取ることができます。

MQGMO_BROWSE_NEXT + MQGMO_MARK_*

カーソル位置から始まって、コンシューマーはキュー上のマークが付いていない次のメッセージを与えられ、その後このコンシューマー用にマークが付けられます。メッセージを現行カーソル位置の後のキューに追加できるので、この組み合わせは注意して使用してください。

MQGMO_BROWSE_FIRST + MQGMO_BROWSE_NEXT + MQGMO_MARK_*

この組み合わせは許可されていません。使用すると、呼び出しはMQRC_OPTIONS_ERRORを戻します。

MQGMO_NO_WAIT、MQGMO_WAIT、および WaitInterval

これらのオプションは、コンシューマーを呼び出す方法を制御します。

MQGMO_NO_WAIT

MQRC_NO_MSG_AVAILABLE でコンシューマーが呼び出されることはありません。コンシューマーは、メッセージおよびイベントのみに関して呼び出されます。

MQGMO_WAIT とゼロの WaitInterval

使用可能なメッセージがなく、かつコンシューマーが既に開始しているか、前回の「メッセージなし」理由コードの後で少なくとも1つのメッセージがコンシューマーに送信されている場合は、MQRC_NO_MSG_AVAILABLE コードがコンシューマーに渡されます。

この場合、ゼロの待機間隔が指定されていると、コンシューマーはビジー・ループ中でポーリングできません。

MQGMO_WAIT および正の WaitInterval

コンシューマーは、理由コード MQRC_NO_MSG_AVAILABLE で、指定された待機間隔の後で呼び出されます。この呼び出しは、どのメッセージがコンシューマーに送信されたかに関係なく行われます。これにより、ユーザーは、ハートビートまたはバッチ・タイプの処理を実行できます。

MQGMO_WAIT および MQWI_UNLIMITED の WaitInterval

これは、MQRC_NO_MSG_AVAILABLE を戻す前の無限待機を指定します。
MQRC_NO_MSG_AVAILABLE でコンシューマーが呼び出されることはありません。

GetMsgOpts MQOP_REGISTER でのみ使用され、いずれかのフィールドでデフォルト以外の値が必要な場合にのみ使用されます。GetMsgOpts はイベント・ハンドラーには使用されません。

GetMsgOpts が必須でない場合は、渡されるパラメーター・アドレスをヌルにすることができます。このパラメーターを使用することは、MQGMO_DEFAULT と MQGMO_FAIL_IF_QUIESCING を一緒に指定する場合と同様です。

MQGMO 構造中でメッセージ・プロパティ・ハンドルが提供されている場合は、コンシューマー・コールバック中に渡されるコピーが MQGMO 構造中に提供されます。MQCB 呼び出しから戻る際に、アプリケーションはメッセージ・プロパティ・ハンドルを削除できます。

CompCode

タイプ: MQLONG - 出力

完了コード。以下のいずれかです。

MQCC_OK

正常終了。

MQCC_WARNING
警告 (部分完了)。

MQCC_FAILED
呼び出し失敗。

理由

タイプ: MQLONG - 出力

次のリストに示す理由コードは、キュー・マネージャーが **Reason** パラメーターに対して戻すことのある理由コードです。

CompCode が MQCC_OK の場合:

MQRC_NONE
(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_FAILED の場合:

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'89C') アダプターが利用できません。

MQRC_ADAPTER_CONV_LOAD_ERROR
(2133, X'855') データ変換サービス・モジュールをロードできない。

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') アダプター・サービス・モジュールをロードできません。

MQRC_API_EXIT_ERROR
(2374, X'946') API 出口で障害が発生しました。

MQRC_API_EXIT_LOAD_ERROR
(2183, X'887') API 出口をロードできません。

MQRC_ASID_MISMATCH
(2157, X'86D') 1 次 ASID とホーム ASID が異なっています。

MQRC_BUFFER_LENGTH_ERROR
(2005, X'7D5') バッファ長パラメーターは無効です。

MQRC_CALL_IN_PROGRESS
(2219, X'8AB') 前の呼び出しが完了する前に MQI 呼び出しが入力されました。

MQRC_CALLBACK_LINK_ERROR
(2487, X'9B7') コールバック・タイプ・フィールドが正しくない。

MQRC_CALLBACK_NOT_REGISTERED
(2448, X'990') コールバックが登録されていないので、登録を抹消、中断、または再開できない。

MQRC_CALLBACK_ROUTINE_ERROR
(2486, X'9B6') *CallbackFunction* または *CallbackName* のどちらかを指定しなければならないが、両方指定することはできない。

MQRC_CALLBACK_TYPE_ERROR
(2483, X'9B3') コールバック・タイプ・フィールドが正しくない。

MQRC_CBD_OPTIONS_ERROR
(2484, X'9B4') MQCBD オプション・フィールドが正しくない。

MQRC_CICS_WAIT_FAILED
(2140, X'85C') 待機要求が CICS により拒否された。

MQRC_CONNECTION_BROKEN
(2009, X'7D9') キュー・マネージャーとの接続が失われました。

MQRC_CONNECTION_NOT_AUTHORIZED
(2217, X'8A9') 接続が許可されていません。

MQRC_CONNECTION QUIESCING
(2202, X'89A') 接続が静止しています。

MQRC_CONNECTION_STOPPING

(2203, X'89B') 接続がシャットダウン中です。

MQRC_CORREL_ID_ERROR

(2207, X'89F') 相関 ID のエラー。

MQRC_DATA_LENGTH_ERROR

(2010, X'7DA') データ長パラメーターが無効である。

MQRC_ENVIRONMENT_ERROR

(2012, X'7DC') この環境では呼び出しが無効です。

MQRC_FUNCTION_NOT_SUPPORTED

(2298, X'8FA') 要求された関数は、現在の環境では使用できない。

MQRC_GET_INHIBITED

(2016, X'7E0') キューからの読み取りが禁止されている。

MQRC_GLOBAL_UOW_CONFLICT

(2351, X'92F') グローバル作業単位に矛盾がある。

MQRC_GMO_ERROR

(2186, X'88A') 読み取りメッセージ・オプションの構造体が無効である。

MQRC_HANDLE_IN_USE_FOR_UOW

(2353, X'931') グローバル作業単位のためのハンドルが使用中。

MQRC_HCONN_ERROR

(2018, X'7E2') 接続ハンドルが無効です。

MQRC_HOBJ_ERROR

(2019, X'7E3') オブジェクト・ハンドルが無効です。

MQRC_INCONSISTENT_BROWSE

(2259, X'8D3') ブラウズの指定が不整合である。

MQRC_INCONSISTENT_UOW

(2245, X'8C5') 作業単位の指定が不整合である。

MQRC_INVALID_MSG_UNDER_CURSOR

(2246, X'8C6') カーソル下のメッセージが取り出し対象として無効である。

MQRC_LOCAL_UOW_CONFLICT

(2352, X'930') グローバル作業単位とローカル作業単位に矛盾がある。

MQRC_MATCH_OPTIONS_ERROR

(2247, X'8C7') 突き合わせオプションが無効である。

MQRC_MAX_MSG_LENGTH_ERROR

(2485, X'9B4') *MaxMsgLength* フィールドが正しくない。

MQRC_MD_ERROR

(2026, X'7EA') メッセージ記述子が無効である。

MQRC_MODULE_ENTRY_NOT_FOUND

(2497, X'9C1') 指定された関数入り口点がモジュール中になかった。

MQRC_MODULE_INVALID

(2496, X'9C0') モジュールが見つかったが、タイプが間違っている。32 ビットでも 64 ビットでもない。または有効なダイナミック・リンク・ライブラリーではない。

MQRC_MODULE_NOT_FOUND

(2495, X'9BF') モジュールが検索パス中にないか、またはロードが許可されていない。

MQRC_MSG_SEQ_NUMBER_ERROR

(2250, X'8CA') メッセージ順序番号が無効である。

MQRC_MSG_TOKEN_ERROR

(2331, X'91B') メッセージ・トークンについて無効な使い方をしている。

MQRC_NO_MSG_AVAILABLE

(2033, X'7F1') メッセージが使用できない。

MQRC_NO_MSG_UNDER_CURSOR

(2034, X'7F2') ブラウズ・カーソルがメッセージに位置付けされていない。

MQRC_NOT_OPEN_FOR_BROWSE

(2036, X'7F4') ブラウズのためにキューがオープンされていない。

MQRC_NOT_OPEN_FOR_INPUT

(2037, X'7F5') 入力のためにキューがオープンされていない。

MQRC_OBJECT_CHANGED

(2041, X'7F9') オープンされた後でオブジェクト定義が変更された。

MQRC_OBJECT_DAMAGED

(2101, X'835') オブジェクトが損傷しました。

MQRC_OPERATION_ERROR

(2206, X'89E') API 呼び出し上の命令コードが正しくない。

MQRC_OPTIONS_ERROR

(2046, X'7FE') オプションが無効であるか、矛盾しています。

MQRC_PAGESET_ERROR

(2193, X'891') ページ・セット・データ・セットへのアクセス中にエラーが発生しました。

MQRC_Q_DELETED

(2052, X'804') キューが削除されました。

MQRC_Q_INDEX_TYPE_ERROR

(2394, X'95A') キューの索引タイプが間違っている。

MQRC_Q_MGR_NAME_ERROR

(2058, X'80A') キュー・マネージャー名が無効であるか、認識されていません。

MQRC_Q_MGR_NOT_AVAILABLE

(2059, X'80B') キュー・マネージャーを接続に使用できません。

MQRC_Q_MGR QUIESCING

(2161, X'871') キュー・マネージャーが静止しています。

MQRC_Q_MGR_STOPPING

(2162, X'872') キュー・マネージャーのシャットダウン中です。

MQRC_RESOURCE_PROBLEM

(2102, X'836') 使用できるシステム・リソースが不足しています。

MQRC_SIGNAL_OUTSTANDING

(2069, X'815') このハンドルに未解決のシグナルがある。

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') ストレージが不足しています。

MQRC_SUPPRESSED_BY_EXIT

(2109, X'83D') 出口プログラムにより呼び出しが抑止されました。

MQRC_SYNCPOINT_LIMIT_REACHED

(2024, X'7E8') 現行の作業単位内では、これ以上メッセージを処理できない。

MQRC_SYNCPOINT_NOT_AVAILABLE

(2072, X'818') 同期点サポートが利用できない。

MQRC_UNEXPECTED_ERROR

(2195, X'893') 予期しないエラーが発生しました。

MQRC_UOW_ENLISTMENT_ERROR

(2354, X'932') グローバル作業単位の参加に失敗した。

MQRC_UOW_MIX_NOT_SUPPORTED

(2355, X'933') 作業単位呼び出しの混合はサポートされていない。

MQRC_UOW_NOT_AVAILABLE

(2255, X'8CF') 作業単位がキュー・マネージャーから使用不可。

MQRC_WAIT_INTERVAL_ERROR

(2090, X'82A') MQGMO での待機間隔が無効である。

MQRC_WRONG_GMO_VERSION

(2256, X'8D0') 提供された MQGMO のバージョンが違っている。

MQRC_WRONG_MD_VERSION

(2257, X'8D1') 提供された MQMD のバージョンが違っている。

これらのコードについて詳しくは、[メッセージおよび理由コード](#)を参照してください。

使用上の注意

1. MQCB を使用して、キュー上で使用可能で、指定された基準と一致する、メッセージごとに呼び出されるアクションを定義します。アクションが処理される際には、メッセージがキューから除去されて定義済みのメッセージ・コンシューマーに渡されるか、メッセージ・トークンが提供されてメッセージの取り出しに使用されます。
2. MQCB は、MQCTL を使用したコンシュームを始める前にコールバック・ルーチンを定義するために使用するか、またはコールバック・ルーチン内から使用することができます。
3. コールバック・ルーチン外から MQCB を使用するには、最初に MQCTL を使用したメッセージ・コンシュームを中断し、その後でコンシュームを再開しなければなりません。
4. MQCB は IMS アダプター内ではサポートされません。

メッセージ・コンシューマーのコールバック・シーケンス

コンシューマーを構成して、そのコンシューマーがライフ・サイクル内におけるキーポイントで、コールバックを呼び出すようにすることができます。以下に例を示します。

- コンシューマーが最初に登録される時
- 接続が開始するとき
- 接続が停止するとき
- MQCLOSE により明示的もしくは暗示的に、コンシューマーの登録が解除される時

表 577. MQCTL verb 定義	
動詞	意味
MQCTL(START)	MQOP_START Operation を使用する MQCTL 呼び出し
MQCTL(STOP)	MQOP_STOP Operation を使用する MQCTL 呼び出し
MQCTL(WAIT)	MQOP_START_WAIT Operation を使用する MQCTL 呼び出し

これにより、コンシューマーと関連している状態を維持することができます。コールバックがアプリケーションから要求される場合、コンシューマー呼び出しの規則は以下のようになります。

REGISTER

常にコールバックの最初のタイプの呼び出しになります。

常に MQCB(REGISTER) 呼び出しと同じスレッドで呼び出されます。

START

常に MQCTL(START) verb と同期して呼び出されます。

- MQCTL(START) verb が戻る前にすべての START コールバックは完了します。

THREAD_AFFINITY が要求された場合は、メッセージ配信と同じスレッド上に置かれます。

前のコールバックが MQCTL(START) 中に MQCTL(STOP) を発行した場合などは、START コールは保障されません。

STOP

接続が再開するまで、この呼び出し以降はメッセージやイベントは配信されません。

アプリケーションが START、メッセージ、またはイベントの呼び出しを前に受けていれば、STOP は保障されます。

DEREGISTER

常にコールバックの最後のタイプの呼び出しになります。

アプリケーションでスレッド・ベースの初期化を実行して、必ず START と STOP のコールバックをクリーンアップするようにしてください。非スレッド・ベースの初期化を行って、REGISTER と DEREGISTER のコールバックのクリーンアップができます。

スレッドのライフと可用性に関しては、説明されていること以外の推測を行わないでください。例えば、最後に DEREGISTER の呼び出しがされた後も生きているスレッドを当てにしないでください。同じように、THREAD_AFFINITY を使用しないことを選択してある場合は、接続開始時に必ずそのスレッドがあるとは限りません。

スレッドの特性に対してアプリケーションに特定の要件がある場合は、必ずそれに合わせてスレッドが作成され、それから MQCTL(WAIT) が使用されます。これにより、非同期メッセージ配信のためのスレッドが IBM MQ に提供されることになります。

メッセージ・コンシューマーの接続使用法

コンシューマーを構成して、そのコンシューマーがライフ・サイクル内におけるキーポイントで、コールバックを呼び出すようにすることができます。以下に例を示します。

- コンシューマーが最初に登録される時
- 接続が開始するとき
- 接続が停止するとき
- MQCLOSE により明示的もしくは暗示的に、コンシューマーの登録が解除される時

表 578. MQCTL verb 定義	
動詞	意味
MQCTL(START)	MQOP_START Operation を使用する MQCTL 呼び出し
MQCTL(STOP)	MQOP_STOP Operation を使用する MQCTL 呼び出し
MQCTL(WAIT)	MQOP_START_WAIT Operation を使用する MQCTL 呼び出し

これにより、コンシューマーと関連している状態を維持することができます。コールバックがアプリケーションから要求される場合、コンシューマー呼び出しの規則は以下のようになります。

REGISTER

常にコールバックの最初のタイプの呼び出しになります。

常に MQCB(REGISTER) 呼び出しと同じスレッドで呼び出されます。

START

常に MQCTL(START) verb と同期して呼び出されます。

- MQCTL(START) verb が戻る前にすべての START コールバックは完了します。

THREAD_AFFINITY が要求された場合は、メッセージ配信と同じスレッド上に置かれます。

前のコールバックが MQCTL(START) 中に MQCTL(STOP) を発行した場合などは、START コールは保障されません。

STOP

接続が再開するまで、この呼び出し以降はメッセージやイベントは配信されません。

アプリケーションが START、メッセージ、またはイベントの呼び出しを前に受けていれば、STOP は保障されます。

DEREGISTER

常にコールバックの最後のタイプの呼び出しになります。

アプリケーションでスレッド・ベースの初期化を実行して、必ず START と STOP のコールバックをクリーンアップするようにしてください。非スレッド・ベースの初期化を行って、REGISTER と DEREGISTER のコールバックのクリーンアップができます。

スレッドのライフと可用性に関しては、説明されていること以外の推測を行わないでください。例えば、最後に DEREGISTER の呼び出しがされた後も生きているスレッドを当てにしないでください。同じように、THREAD_AFFINITY を使用しないことを選択してある場合は、接続開始時に必ずそのスレッドがあるとは限りません。

スレッドの特性に対してアプリケーションに特定の要件がある場合は、必ずそれに合わせてスレッドが作成され、それから MQCTL(WAIT) が使用されます。これにより、非同期メッセージ配信のためのスレッドが IBM MQ に提供されることになります。

C 言語での呼び出し

```
MQCB (Hconn, Operation, CallbackDesc, Hobj, MsgDesc,  
GetMsgOpts, &CompCode, &Reason);
```

パラメーターを次のように宣言します。

```
MQHCONN  Hconn;          /* Connection handle */  
MQLONG   Operation;      /* Operation being processed */  
MQCBD    CallbackDesc;   /* Callback descriptor */  
MQHOBj   Hobj;           /* Object handle */  
MQMD     MsgDesc         /* Message descriptor attributes */  
MQGMO    GetMsgOpts      /* Message options */  
MQLONG   CompCode;       /* Completion code */  
MQLONG   Reason;         /* Reason code qualifying CompCode */
```

COBOL での呼び出し

```
CALL 'MQCB' USING HCONN, OPERATION, CBDESC, HOBJ, MSGDESC,  
GETMSGOPTS, COMPCODE, REASON.
```

パラメーターを次のように宣言します。

```
**      Connection handle  
01 HCONN    PIC S9(9) BINARY.  
**      Operation  
01 OPERATION PIC S9(9) BINARY.  
**      Callback Descriptor  
01 CBDESC.  
   COPY CMQCBDEV.  
01 HOBJ     PIC S9(9) BINARY.  
**      Message Descriptor  
01 MSGDESC.  
   COPY CMQMDV.  
**      Get Message Options  
01 GETMSGOPTS.  
   COPY CMQGMV.  
**      Completion code  
01 COMPCODE PIC S9(9) BINARY.  
**      Reason code qualifying COMPCODE  
01 REASON   PIC S9(9) BINARY.
```

PL/I での呼び出し

```
call MQCB(Hconn, Operation, CallbackDesc, Hobj, MsgDesc, GetMsgOpts,  
CompCode, Reason)
```

パラメーターを次のように宣言します。

```
dcl Hconn      fixed bin(31); /* Connection handle */  
dcl Operation  fixed bin(31); /* Operation */  
dcl CallbackDesc like MQCBD;  /* Callback Descriptor */  
dcl Hobj       fixed bin(31); /* Object Handle */  
dcl MsgDesc    like MQMD;     /* Message Descriptor */  
dcl GetMsgOpts like MQGMO;     /* Get Message Options */
```

```
dcl CompCode      fixed bin(31); /* Completion code */
dcl Reason        fixed bin(31); /* Reason code qualifying CompCode */
```

MQCB_FUNCTION - コールバック関数

MQCB_FUNCTION 関数呼び出しは、イベント処理および非同期メッセージ・コンシューム用のコールバック関数です。

MQCB_FUNCTION 呼び出し定義は、単独で提供され、コールバック関数に渡すパラメーターを記述します。キュー・マネージャーは、MQCB_FUNCTION という名前の入り口点を提供しません。

実際に呼び出される機能の仕様は、[MQCB](#) 呼び出しに対する入力で、[MQCBD](#) 構造により渡されます。

構文

MQCB_FUNCTION (*Hconn*, *MsgDesc*, *GetMsgOpts*, *Buffer*, *Context*)

パラメーター

Hconn

タイプ: MQHCONN - 入力

このハンドルは、キュー・マネージャーに対する接続を表します。Hconn の値は、先行の MQCONN または MQCONNX 呼び出しによって戻されたものです。z/OS 上で CICS アプリケーションを使用する場合は、MQCONN 呼び出しを省略でき、Hconn に次の値を指定できます。

MQHC_DEF_HCONN

デフォルトの接続ハンドル。

MsgDesc

タイプ: MQMD - 入力

この構造は、取り出されるメッセージの属性を記述します。

詳細は [385 ページ](#)の『MQMD - メッセージ記述子』を参照してください。

渡される MQMD のバージョンは、コンシューマー機能を定義した MQCB 呼び出し上で渡されるバージョンと同じです。

バージョン 4 の MQGMO を使用して MQMD ではなくメッセージ・ハンドルを戻すよう要求した場合は、MQMD のアドレスはヌル文字として渡されます。

これはメッセージ・コンシューマー関数に対する入力フィールドで、イベント・ハンドラー関数には関係ありません。

GetMsgOpts

タイプ: MQGMO - 入力

メッセージ・コンシューマーのアクションの制御に使用するオプション。このパラメーターには、戻されるメッセージに関する追加情報も含まれます。

詳細については [MQGMO](#) を参照してください。

サポートされている最新バージョンが、渡される MQGMO のバージョンになります。

これはメッセージ・コンシューマー関数に対する入力フィールドで、イベント・ハンドラー関数には関係ありません。

Buffer

タイプ: MQBYTEExBufferLength - 入力

これは、メッセージ・データを含む領域です。

この呼び出しに関する使用可能なメッセージがない場合か、メッセージにメッセージ・データが含まれていない場合は、Buffer のアドレスはヌルとして渡されます。

これはメッセージ・コンシューマー関数に対する入力フィールドで、イベント・ハンドラー関数には関係ありません。

Context

タイプ: MQCBC - 入出力

この構造は、コールバック関数に対してコンテキスト情報を提供します。詳細は [225 ページの『MQCBC - コールバック・コンテキスト』](#) を参照してください。

使用上の注意

1. コールバック・ルーチンが、スレッドを遅らせるかブロックする可能性があるサービス (例えば、待機を指定した MQGET) を使用する場合は、他のコールバックのディスパッチが遅延する可能性があります。
2. コールバック・ルーチンの呼び出しごとに別の作業単位が自動的に確立されないため、ルーチンはコミット呼び出しを発行するか、作業の論理バッチが処理されるまでコミットを延期できます。作業のバッチをコミットすると、最後の同期点以降に呼び出されたすべてのコールバック関数に関するメッセージがコミットされます。
3. CICS LINK または CICS START によって呼び出されるプログラムは、チャンネル・コンテナという名前付きオブジェクトにより、CICS サービスを使用してパラメーターを取り出します。コンテナ名は、パラメーター名と同じです。詳しくは、CICS の資料を参照してください。
4. コールバック・ルーチンは MQDISC 呼び出しを発行できますが、そのルーチン独自の接続では発行しません。例えば、コールバック・ルーチンは、接続を確立している場合、その接続を切断することもできます。
5. 一般的には、コールバック・ルーチンは、毎回同じスレッドから呼び出されることに依存できません。必要な場合は、接続の開始時に MQCTLO_THREAD_AFFINITY を使用してください。
6. コールバック・ルーチンがゼロ以外の理由コードを受け取る場合は、該当するアクションを取らなければなりません。
7. MQCB 関数は IMS アダプター内ではサポートされません。

MQCLOSE - オブジェクトのクローズ

MQCLOSE 呼び出しは、オブジェクトへのアクセスを解放するもので、MQOPEN および MQSUB 呼び出しの逆です。

構文

MQCLOSE (*Hconn*, *Hobj*, *Options*, *CompCode*, *Reason*)

パラメーター

Hconn

タイプ: MQHCONN - 入力

このハンドルは、キュー・マネージャーに対する接続を表します。Hconn の値は、先行の MQCONN または MQCONNX 呼び出しによって戻されたものです。

z/OS 上で CICS アプリケーションを使用する場合は、MQCONN 呼び出しを省略でき、Hconn に次の値を指定できます。

MQHC_DEF_HCONN

デフォルトの接続ハンドル。

Hobj

タイプ: MQHOBJ - 入出力

このハンドルは、クローズするオブジェクトを表します。オブジェクトは、どのタイプでも構いません。Hobj の値は、前の MQOPEN 呼び出しで戻されたものです。

呼び出しが正常に完了すると、キュー・マネージャーは、環境に対して有効なハンドルでない値にこのパラメーターを設定します。値は、以下のとおりです。

MQHO_UNUSABLE_HOBJ

使用できないオブジェクト・ハンドル。

z/OS では、*Hobj* は未定義の値に設定されます。

オプション

タイプ: MQLONG - 入力

このパラメーターは、オブジェクトをクローズする方法を制御します。

複数の方法でクローズできるのは、永続動的キューと永続サブスクリプションだけです。保存するか削除するか、そのどちらかでなければならないからです。永続動的キューとは、**DefinitionType** 属性の値が MQQDT_PERMANENT_DYNAMIC であるキューを指します (830 ページの『[キューの属性](#)』で説明している **DefinitionType** 属性を参照してください)。クローズ・オプションについては、このトピックで要約されています。

永続サブスクリプションについては、保持される場合と除去される場合があります。永続サブスクリプションは MQSO_DURABLE オプションを指定して MQSUB 呼び出しを使用することによって作成されます。

管理対象宛先に対するハンドル (MQSO_MANAGED オプションを使用した MQSUB 呼び出しで戻された **Hobj** パラメーター) をクローズする場合は、キュー・マネージャーが、関連サブスクリプションが除去されたときに取り出されなかったパブリケーションをすべてクリーンアップします。サブスクリプションは、MQSUB 呼び出しで戻される **Hsub** パラメーターに対して MQCO_REMOVE_SUB オプションを使用して除去されます。MQCO_REMOVE_SUB は、非永続サブスクリプションに対する MQCLOSE のデフォルトの動作であることに注意してください。

非管理対象の宛先へのハンドルを閉じる場合、パブリケーションが送信されるキューのクリーンアップはユーザーの責任で行います。まず MQCO_REMOVE_SUB を使ってサブスクリプションをクローズし、その後、キューにメッセージが何も残らなくなるまで処理してください。

以下からオプションを 1 つだけ指定する必要があります。

動的キュー・オプション: このオプションは、永続動的キューをクローズする方法を制御します。

MQCO_DELETE

以下の条件のいずれかが真の場合、キューは削除されます。

- 前の MQOPEN 呼び出しによって作成された永続動的キューであり、メッセージ、およびキューに対して未解決になっているコミットされていない読み取り要求または書き込み要求がない (現行タスクまたは任意のタスクのための)。
- *Hobj* を戻す MQOPEN 呼び出しにより作成された一時動的キューである。この場合、キューに入っているすべてのメッセージは消去されます。

その他の場合 (MQSUB 呼び出しで *Hobj* が戻された場合を含む) はすべて、呼び出しは失敗し、理由コード MQRC_OPTION_NOT_VALID_FOR_TYPE が戻ります。オブジェクトは削除されません。

z/OS では、キューが論理的に削除された動的キューであって、このハンドルがその動的キューの最後のハンドルである場合、キューは物理的に削除されます。詳細については、[648 ページの『使用上の注意』](#)を参照してください。

MQCO_DELETE_PURGE

以下の条件のいずれかが真の場合、キューは削除され、キュー内のメッセージはページされます。

- 前の MQOPEN 呼び出しで作成された永続動的キューであり、キューに対して未解決になっているコミットされていない読み取り要求または書き込み要求がない (現行タスクまたはそれ以外の任意のタスクのための)。
- *Hobj* を戻す MQOPEN 呼び出しにより作成された一時動的キューである。

その他の場合 (MQSUB 呼び出しで *Hobj* が戻された場合を含む) はすべて、呼び出しは失敗し、理由コード MQRC_OPTION_NOT_VALID_FOR_TYPE が戻ります。オブジェクトは削除されません。

表 579. さまざまなオブジェクト・タイプのクローズ・オプション			
オブジェクトまたはキューのタイプ	MQCO_NONE	MQCO_DELETE	MQCO_DELETE_PURGE
キュー以外のオブジェクト	保存	無効	無効
事前定義されたキュー	保存	無効	無効
永続動的キュー	保存	空で、保留中の更新がない場合は削除	メッセージを削除。保留中の更新がないキューを削除
一時動的キュー (キューの作成者から発行された呼び出し)	削除	削除	削除
一時動的キュー (キューの作成者から発行されていない呼び出し)	保存	無効	無効
配布リスト	保存	無効	無効
管理対象サブスクリプション宛先	保存	無効	無効
配布リスト (サブスクリプションは除去済み)	メッセージを削除。キューを削除。	無効	無効

サブスクリプション閉止オプション: このオプションは、ハンドルをクローズしたときに永続サブスクリプションを除去するかどうか、アプリケーションによる読み取りを待機しているパブリケーションをクリーンアップするかどうかを制御します。このオプションは、MQSUB 呼び出しの **Hsub** パラメータで戻されるオブジェクト・ハンドルで使用する場合のみ有効です。

MQCO_KEEP_SUB

サブスクリプションに対するハンドルはクローズされますが、作成されたサブスクリプションは保持されます。パブリケーションは引き続き、サブスクリプションで指定された宛先に送られます。このオプションは、オプション MQSO_DURABLE を指定してサブスクリプションが作成された場合のみ有効です。

サブスクリプションが永続サブスクリプションの場合、MQCO_KEEP_SUB はデフォルトです。

MQCO_REMOVE_SUB

サブスクリプションは除去され、サブスクリプションに対するハンドルはクローズされます。

MQSUB 呼び出しの **Hobj** パラメータは **Hsub** パラメータの閉止によって無効にされず、残りのパブリケーションを受け取るために引き続き MQGET または MQCB で使用することができます。MQSUB 呼び出しの **Hobj** パラメータもクローズされると、それが管理対象宛先だった場合、取得されていないパブリケーションはすべて除去されます。

サブスクリプションが非永続サブスクリプションの場合、MQCO_REMOVE_SUB はデフォルトです。

MQCO_REMOVE_SUB の正常終了は、アクションが完了したことを意味しません。この呼び出しが完了したことを確認するには、[「分散ネットワーク向け非同期コマンドの完了確認」](#) の DELETE SUB ステップを参照してください。

このサブスクリプション閉止オプションについては次の表で要約されています。

表 580. 永続サブスクリプション・ハンドルをクローズしてもサブスクリプションは保持するためのオプション	
タスク	サブスクリプション閉止オプション
MQOPENed ハンドルのパブリケーションを保持する	MQCO_KEEP_SUB
MQOPENed ハンドルのパブリケーションを除去する	アクションは許可されていない

表 580. 永続サブスクリプション・ハンドルをクローズしてもサブスクリプションは保持するためのオプション (続き)	
タスク	サブスクリプション閉止オプション
MQSO_MANAGED ハンドルのパブリケーションを保持する	MQCO_KEEP_SUB
MQSO_MANAGED ハンドルのパブリケーションを除去する	アクションは許可されていない

永続サブスクリプション・ハンドルを閉じてアンサブスクライブするか、または非永続サブスクリプション・ハンドルを閉じることによってアンサブスクライブを行うには、以下のサブスクリプションのクローズ・オプションを使用します。

表 581. アンサブスクライブのオプション	
タスク	サブスクリプション閉止オプション
MQOPENed ハンドルのパブリケーションを保持する	MQCO_REMOVE_SUB
MQOPENed ハンドルのパブリケーションを除去する	アクションは許可されていない
MQSO_MANAGED ハンドルのパブリケーションを保持する	MQCO_REMOVE_SUB

先読みオプション: 次のオプションは、アプリケーションによって要求される前に非永続メッセージがクライアントに送られ、アプリケーションによってコンシュームされなかった場合の非永続メッセージの処理を制御します。これらのメッセージは、クライアント先読みバッファに格納されてアプリケーションによる要求を待機し、MQCLOSE が完了する前にキューから廃棄またはコンシュームすることができます。

MQCO_IMMEDIATE

オブジェクトは即時にクローズされ、アプリケーションが要求する前にクライアントに送信されたメッセージはすべて廃棄されます。アプリケーションがこのメッセージをコンシュームすることはできません。これがデフォルト値です。

MQCO_QUIESCE

オブジェクトのクローズ要求は行われますが、アプリケーションがそれを要求する前にクライアントに送られたメッセージがクライアントの先読みバッファに存在する場合、MQCLOSE を呼び出す際に MQRC_READ_AHEAD_MSGS という警告が戻され、オブジェクト・ハンドルは有効なままとなります。

アプリケーションは引き続きそのオブジェクト・ハンドルを使用することができ、メッセージがなくなるまで取り出しを続行します。その後、オブジェクトを再びクローズします。アプリケーションが要求する前にメッセージがクライアントに送られることはなくなり、先読みはオフになります。

アプリケーションでは、クライアントの先読みバッファにメッセージがなくなる時点まで試すのではなく、MQCO_QUIESCE を使用することをお勧めします。最後に MQGET を呼び出してから次の MQCLOSE までの間にメッセージが到着する可能性があり、MQCO_IMMEDIATE が使用されている場合にそのメッセージは廃棄されてしまうからです。

MQCO_QUIESCE を指定した MQCLOSE を非同期コールバック関数内から発行すると、先読みメッセージと同じ動作になります。警告 MQRC_READ_AHEAD_MSGS が戻された場合、少なくとももう一度コールバック関数が呼び出されます。最後に残っている先読みメッセージがコールバック関数に渡されると、MQCBC ConsumerFlags フィールドが MQCBCF_READA_BUFFER_EMPTY に設定されます。

デフォルト・オプション: 上記で説明されたオプションのいずれも必要としない場合、以下のオプションを使用できます。

MQCO_NONE

オプションのクローズ処理は不要である。

これは、次のものに対して指定しなければなりません。

- キュー以外のオブジェクト
- 事前定義キュー
- 一時動的キュー (ただし、*Hobj* が、キューを作成した MQOPEN 呼び出しにより戻されるハンドルではない場合のみ)
- 配布リスト

上記の場合はすべて、オブジェクトは残され、削除されません。

このオプションが一時動的キューに対して指定されていると、次のようになります。

- *Hobj* を戻した MQOPEN 呼び出しにより作成されたキューは、削除されます。そして、そのキュー内にあるメッセージはすべて除去されます。
- 上記以外の場合、キュー (およびキュー内のすべてのメッセージ) は保存されます。

永続動的キューに対してこのオプションが指定されていると、キューは残され、削除されません。

z/OS では、キューが論理的に削除された動的キューであって、このハンドルがその動的キューの最後のハンドルである場合、キューは物理的に削除されます。詳細については、[648 ページの『使用上の注意』](#)を参照してください。

CompCode

タイプ: MQLONG - 出力

完了コード。以下のいずれかです。

MQCC_OK

正常終了。

MQCC_WARNING

警告 (部分完了)。

MQCC_FAILED

呼び出し失敗。

理由

タイプ: MQLONG - 出力

次に示す理由コードは、キュー・マネージャーが **Reason** パラメーターに対して返すことのある理由コードです。

CompCode が MQCC_OK の場合:

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_WARNING の場合:

MQRC_INCOMPLETE_GROUP

(2241, X'8C1') メッセージ・グループが不完全である。

MQRC_INCOMPLETE_MSG

(2242, X'8C2') 論理メッセージが不完全である。

MQRC_READ_AHEAD_MSGS

(nnnn, X'xxx') クライアントに、まだアプリケーションによって消費されていない先読みメッセージがある。

CompCode が MQCC_FAILED の場合:

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'89C') アダプターが利用できません。

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852') アダプター・サービス・モジュールをロードできません。

MQRC_API_EXIT_ERROR

(2374, X'946') API 出口で障害が発生しました。

MQRC_API_EXIT_LOAD_ERROR

(2183, X'887') API 出口をロードできません。

MQRC_ASID_MISMATCH

(2157, X'86D') 1 次 ASID とホーム ASID が異なっています。

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') 前の呼び出しが完了する前に MQI 呼び出しが入力されました。

MQRC_CF_NOT_AVAILABLE

(2345, X'929') カップリング・ファシリティが使用できません。

MQRC_CF_STRUC_FAILED

(2373, X'945') カップリング・ファシリティ構造体で障害が発生しました。

MQRC_CF_STRUC_IN_USE

(2346, X'92A') カップリング・ファシリティ構造体が使用中です。

MQRC_CICS_WAIT_FAILED

(2140, X'85C') 待機要求が CICS により拒否された。

MQRC_CONNECTION_BROKEN

(2009, X'7D9') キュー・マネージャーとの接続が失われました。

MQRC_CONNECTION_NOT_AUTHORIZED

(2217, X'8A9') 接続が許可されていません。

MQRC_CONNECTION_STOPPING

(2203, X'89B') 接続がシャットダウン中です。

MQRC_DB2_NOT_AVAILABLE

(2342, X'926') Db2 サブシステムが利用できません。

MQRC_HCONN_ERROR

(2018, X'7E2') 接続ハンドルが無効です。

MQRC_HOBJ_ERROR

(2019, X'7E3') オブジェクト・ハンドルが無効です。

MQRC_NOT_AUTHORIZED

(2035, X'7F3') アクセスは許可されません。

MQRC_OBJECT_DAMAGED

(2101, X'835') オブジェクトが損傷しました。

MQRC_OPTION_NOT_VALID_FOR_TYPE

(2045, X'7FD') MQOPEN または MQCLOSE 呼び出しで、オプションが、オブジェクト・タイプとして無効である。

MQRC_OPTIONS_ERROR

(2046, X'7FE') オプションが無効であるか、矛盾しています。

MQRC_PAGESET_ERROR

(2193, X'891') ページ・セット・データ・セットへのアクセス中にエラーが発生しました。

MQRC_Q_MGR_NAME_ERROR

(2058, X'80A') キュー・マネージャー名が無効であるか、認識されていません。

MQRC_Q_MGR_NOT_AVAILABLE

(2059, X'80B') キュー・マネージャーを接続に使用できません。

MQRC_Q_MGR_STOPPING

(2162, X'872') キュー・マネージャーのシャットダウン中です。

MQRC_Q_NOT_EMPTY

(2055, X'807') メッセージ、またはコミットされていない書き込み要求か取得要求が、1 つ以上キューに入っています。

MQRC_RESOURCE_PROBLEM

(2102, X'836') 使用できるシステム・リソースが不足しています。

MQRC_SECURITY_ERROR

(2063, X'80F') セキュリティー・エラーが発生しました。

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') ストレージが不足しています。

MQRC_SUPPRESSED_BY_EXIT

(2109, X'83D') 出口プログラムにより呼び出しが抑止されました。

MQRC_UNEXPECTED_ERROR

(2195, X'893') 予期しないエラーが発生しました。

これらのコードについて詳しくは、[メッセージおよび理由コード](#)を参照してください。

使用上の注意

1. アプリケーションが MQDISC 呼び出しを発行するか、正常終了または異常終了すると、このアプリケーションによってオープンされたままになっているすべてのオブジェクトは、MQCO_NONE オプションで自動的にクローズされます。
2. クローズされるオブジェクトがキューであるときに、以下の点が適用されます。
 - キューに対する操作が作業単位の一部として実行された場合は、そのキューは、同期点の前後のいずれでも、同期点の結果に影響を与えることなくクローズすることができます。キューがトリガーされる場合に、キューをクローズする前にロールバックを実行すると、トリガー・メッセージが発行されます。トリガー・メッセージの詳細については、[トリガー・メッセージの特性](#)を参照してください。
 - キューが MQOO_BROWSE オプションでオープンされた場合、ブラウズ・カーソルは破棄されます。その後で MQOO_BROWSE オプションによってキューを再オープンした場合は、新しいブラウズ・カーソルが作成されます ([MQOO_BROWSE](#) を参照してください)。
 - MQCLOSE 呼び出しを発行した時点で、該当するハンドルに対してメッセージがロックされている場合は、ロックが解除されます ([MQGMO_LOCK](#) を参照してください)。
 - z/OS で、MQGMO_SET_SIGNAL オプションを指定した MQGET 要求がクローズ中キュー・ハンドルに対して処理を終了していない場合、その要求は取り消されます ([MQGMO_SET_SIGNAL](#) オプションを参照してください)。同じキューのシグナル要求でも、異なるハンドル (*Hobj*) に対して設定されている要求は、影響を受けません (ただし、動的キューが削除対象でない場合は、このような要求も取り消されます)。
3. クローズされるオブジェクトが動的キュー (永続または一時) であるとき、以下の点が適用されます。
 - 動的キューの場合、MQCO_DELETE および MQCO_DELETE_PURGE オプションは、対応する MQOPEN 呼び出し上に指定されたオプションに関係なく指定することができます。
 - 動的キューが削除されると、キューに対して未解決になっている、MQGMO_WAIT オプションをもつすべての MQGET 呼び出しは取り消され、理由コード MQRC_Q_DELETED が戻ります。[MQGMO_WAIT](#) を参照してください。

削除されたキューはアプリケーションからアクセスできません。しかし、キューを参照するすべてのハンドルがクローズして、キューに影響を与えるすべての作業単位がコミットされるかまたはバックアウトされるまでは、キューはシステムから除去されず、また関連リソースも解放されません。

z/OS において、論理的には削除されているがまだシステムから除去されていないキューがあると、削除したキューと同じ名前を指定してキューを新規作成することができません。このような場合、MQOPEN 呼び出しは失敗し、理由コード MQRC_NAME_IN_USE が戻されます。また、そのようなキューは MQSC コマンドを使用して表示されたままになる場合があります。ただし、アプリケーションからアクセスすることはできません。

- 永続動的キューが削除されると、MQCLOSE 呼び出しで指定した *Hobj* ハンドルが、キューを作成した MQOPEN 呼び出しによって戻されたハンドルではない場合は、MQOPEN 呼び出しを妥当性検査するために使用されたユーザー ID が、キューを削除する許可を持っているかどうかを検査されます。MQOPEN 呼び出し時に MQOO_ALTERNATE_USER_AUTHORITY オプションが指定された場合、検査されたユーザー ID は *AlternateUserId* になります。

この検査は以下のような場合は実行されません。

- 指定されたハンドルが、キューを作成した MQOPEN 呼び出しによって戻されたハンドルである場合。
- 削除されるキューが一時動的キューの場合。
- ・一時動的キューがクローズされると、MQCLOSE 呼び出しで指定した *Hobj* ハンドルが、キューを作成した MQOPEN 呼び出しによって戻されたハンドルである場合は、キューは削除されます。これは MQCLOSE 呼び出し時にクローズ・オプションが指定されているか否かにかかわらず起こります。キューにメッセージがある場合、それらは廃棄されます。その際、レポート・メッセージは生成されません。

キューに影響を与える、コミットされていない作業単位がある場合でも、キューとそのメッセージは削除されます。しかし、作業単位は失敗しません。ただし、前述のとおり、各作業単位がコミットされるか、またはバックアウトされるまでは、作業単位に関連するリソースは解放されません。

4. クローズされるオブジェクトが配布リストであるとき、以下の点が適用されます。

- ・配布リストに有効なクローズ・オプションは MQCO_NONE だけです。この呼び出しは、他のオプションが指定されている場合には失敗し、理由コード MQRC_OPTIONS_ERROR または MQRC_OPTION_NOT_VALID_FOR_TYPE が戻ります。
- ・配布リストがクローズされると、リスト内のキューについて個々の完了コードおよび理由コードは戻されません。診断目的に利用できるのは、この呼び出しの **CompCode** および **Reason** パラメーターだけです。

いずれかのキューのクローズ時に障害が起こっても、キュー・マネージャーは処理を継続し、配布リスト内の残りのキューをクローズしようとします。次に、この呼び出しの **CompCode** および **Reason** パラメーターが、その障害を記述する情報を戻すよう設定されます。キューのほとんどが正常にクローズされた場合でも、完了コードが MQCC_FAILED になる場合があります。クローズ中にエラーが発生したキューは、識別されません。

複数のキューで障害が発生した場合、どの障害が および **Reason CompCode** パラメータで報告されるかは定義されていません。

C 言語での呼び出し

```
MQCLOSE (Hconn, &Hobj, Options, &CompCode, &Reason);
```

パラメーターを次のように宣言します。

```
MQHCONN  Hconn;      /* Connection handle */
MQHOBJ    Hobj;       /* Object handle */
MQLONG    Options;    /* Options that control the action of MQCLOSE */
MQLONG    CompCode;   /* Completion code */
MQLONG    Reason;     /* Reason code qualifying CompCode */
```

COBOL での呼び出し

```
CALL 'MQCLOSE' USING HCONN, HOBJ, OPTIONS, COMPCODE, REASON.
```

パラメーターを次のように宣言します。

```
** Connection handle
01 HCONN    PIC S9(9) BINARY.
** Object handle
01 HOBJ     PIC S9(9) BINARY.
** Options that control the action of MQCLOSE
01 OPTIONS  PIC S9(9) BINARY.
** Completion code
01 COMPCODE PIC S9(9) BINARY.
** Reason code qualifying COMPCODE
01 REASON   PIC S9(9) BINARY.
```

PL/I での呼び出し

```
call MQCLOSE (Hconn, Hobj, Options, CompCode, Reason);
```

パラメーターを次のように宣言します。

```
dcl Hconn      fixed bin(31); /* Connection handle */
dcl Hobj       fixed bin(31); /* Object handle */
dcl Options    fixed bin(31); /* Options that control the action of
                               MQCLOSE */
dcl CompCode   fixed bin(31); /* Completion code */
dcl Reason     fixed bin(31); /* Reason code qualifying CompCode */
```

高水準アセンブラー呼び出し

```
CALL MQCLOSE,(HCONN,HOBJ,OPTIONS,COMPCODE,REASON)
```

パラメーターを次のように宣言します。

```
HCONN      DS  F  Connection handle
HOBJ       DS  F  Object handle
OPTIONS    DS  F  Options that control the action of MQCLOSE
COMPCODE   DS  F  Completion code
REASON     DS  F  Reason code qualifying COMPCODE
```

Visual Basic での呼び出し

```
MQCLOSE Hconn, Hobj, Options, CompCode, Reason
```

パラメーターを次のように宣言します。

```
Dim Hconn As Long 'Connection handle'
Dim Hobj As Long 'Object handle'
Dim Options As Long 'Options that control the action of MQCLOSE'
Dim CompCode As Long 'Completion code'
Dim Reason As Long 'Reason code qualifying CompCode'
```

MQCMIT - 変更のコミット

MQCMIT 呼び出しは、アプリケーションが同期点に達したこと、および最後の同期点以降に発生したメッセージの読み取りと書き込みをすべて永続化することをキュー・マネージャーに示します。

作業単位の一部として書き込まれたメッセージは、他のアプリケーションで使用できるようになります。作業単位の一部として取り出されたメッセージは削除されます。

- ▶ **z/OS** z/OS では、この呼び出しはバッチ・プログラム (IMS バッチ DL/I プログラムを含む) でのみ使用されます。

構文

MQCMIT (*Hconn*, *CompCode*, *Reason*)

パラメーター

Hconn

タイプ: MQHCONN - 入力

このハンドルは、キュー・マネージャーに対する接続を表します。 *Hconn* の値は、先行の MQCONN または MQCONNX 呼び出しによって戻されたものです。

CompCode

タイプ: MQLONG - 出力

完了コード。以下のいずれかです。

MQCC_OK

正常終了。

MQCC_WARNING

警告 (部分完了)。

MQCC_FAILED

呼び出し失敗。

理由

タイプ: MQLONG - 出力

次に示す理由コードは、キュー・マネージャーが **Reason** パラメーターに対して返すことのある理由コードです。

CompCode が MQCC_OK の場合:

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_WARNING の場合:

MQRC_BACKED_OUT

(2003, X'7D3') 作業単位がバックアウトされた。

MQRC_OUTCOME_PENDING

(2124, X'84C') コミット操作の結果が保留状態である。

CompCode が MQCC_FAILED の場合:

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852') アダプター・サービス・モジュールをロードできません。

MQRC_API_EXIT_ERROR

(2374, X'946') API 出口で障害が発生しました。

MQRC_ASID_MISMATCH

(2157, X'86D') 1 次 ASID とホーム ASID が異なっています。

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') 前の呼び出しが完了する前に MQI 呼び出しが入力されました。

MQRC_CALL_INTERRUPTED

(2549, X'9F5') MQPUT または MQCMIT が中断されたため、再接続処理で確実な成果を再び得ることができない。

MQRC_CF_STRUC_IN_USE

(2346, X'92A') カップリング・ファシリティ構造体が使用中です。

MQRC_CONNECTION_BROKEN

(2009, X'7D9') キュー・マネージャーとの接続が失われました。

MQRC_ENVIRONMENT_ERROR

(2012, X'7DC') この環境では呼び出しが無効です。

MQRC_HCONN_ERROR

(2018, X'7E2') 接続ハンドルが無効です。

MQRC_OBJECT_DAMAGED

(2101, X'835') オブジェクトが損傷しました。

MQRC_OUTCOME_MIXED

(2123, X'84B') コミットまたはバックアウト操作の結果が混在している。

MQRC_Q_MGR_STOPPING

(2162, X'872') キュー・マネージャーのシャットダウン中です。

MQRC_RECONNECT_FAILED

(2548, X'9F4') 再接続後、再接続可能な接続のハンドルの復元中にエラーが発生した。

MQRC_RESOURCE_PROBLEM

(2102, X'836') 使用できるシステム・リソースが不足しています。

MQRC_STORAGE_MEDIUM_FULL

(2192, X'890') 外部ストレージ・メディアが満杯です。

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') ストレージが不足しています。

MQRC_UNEXPECTED_ERROR

(2195, X'893') 予期しないエラーが発生しました。

これらのコードについて詳しくは、[メッセージおよび理由コード](#)を参照してください。

使用上の注意

1. この呼び出しは、キュー・マネージャーそのものが作業単位を調整するときのみ使用します。次のタイプがあります。

- ローカル作業単位 (変更内容は IBM MQ リソースにのみ影響を及ぼす)。
- グローバル作業単位 (変更内容は、IBM MQ リソースだけでなく、他のリソース・マネージャーに属するリソースにも影響を及ぼす場合がある)。

ローカル作業単位およびグローバル作業単位の詳細については、[624 ページの『MQBEGIN - 作業単位の開始』](#)を参照してください。

2. キュー・マネージャーが作業単位を調整しない環境では、MQCMIT ではなく適切なコミット呼び出しを使用する必要があります。この環境ではまた、アプリケーションの正常終了を原因とする暗黙的コミットをサポートしている場合もあります。

- z/OS では、以下の呼び出しを使用してください。

- バッチプログラム (バッチ DL/I プログラム IMS を含む) は、作業単位がリソース IBM MQ のみに影響する場合、MQCMIT 呼び出しを使用できます。ただし、作業単位が IBM MQ リソースに加えて他のリソース・マネージャー (Db2 など) に属するリソースにも影響を及ぼす場合には、z/OS Recoverable Resource Service (RRS) で提供される SRRCMIT 呼び出しを使用してください。SRRCMIT 呼び出しは、RRS 調整に使用可能になっているリソース・マネージャーに属するリソースの変更をコミットします。
- CICS アプリケーションでは、EXEC CICS SYNCPOINT コマンドを使用して作業単位を明示的にコミットする必要があります。また、トランザクションを終了すると、作業単位が暗黙的にコミットされます。CICS アプリケーションで MQCMIT 呼び出しを使用することはできません。
- IMS バッチ DL/I プログラム以外のアプリケーションは、作業単位をコミットするために、`CHKP` や `GU` などの呼び出し IMS を使用しなければならない。IMS アプリケーション (バッチ DL/I プログラム以外) では、MQCMIT 呼び出しは使用できません。

- IBM i では、この呼び出しはキュー・マネージャーで調整されるローカル作業単位で使用してください。これは、コミットメント定義がジョブ・レベルで存在してはならないことを意味します。つまり、**CMTSCOPE(*JOB)** パラメーターを指定した STRCMTCTL コマンドがジョブに対して発行されているわけではありません。

3. 作業単位内にあるコミットされていない変更内容でアプリケーションが終了する場合、それらの変更内容の後処理は、そのアプリケーションが正常に終了するか、異常終了するかで異なります。詳細については、[MQDISC の使用上の注意](#)を参照してください。

4. アプリケーションでグループ内のメッセージまたは論理メッセージのセグメントの書き込みまたは読み取りを行う場合、キュー・マネージャーは、最後に MQPUT および MQGET 呼び出しが正常に実行されたメッセージ・グループに関する情報を保存します。この情報は、キュー・ハンドルに関する次のような情報です。

- MQMD 中の *GroupId*、*MsgSeqNumber*、*Offset*、および *MsgFlags* フィールドの値。
- そのメッセージが作業単位の一部であるかどうか。
- MQPUT 呼び出しについて、そのメッセージが持続メッセージか、非持続メッセージか。

作業単位がコミットされると、キュー・マネージャーは、グループおよびセグメント情報を保持し、アプリケーションは現行メッセージ・グループまたは論理メッセージの書き込みまたは読み取りを継続することができます。

1つの作業単位のコミット時にグループおよびセグメント情報を保持することにより、アプリケーションは、大きなメッセージ・グループまたは数多くのセグメントで構成される大きな論理メッセージを、いくつかの作業単位にスプレッドできます。ローカル・キュー・マネージャーのキュー・ストレージが限られている場合には、いくつかの作業単位を使用する方が有効となります。ただし、システム障害が発生した場合に各メッセージの書き込みまたは読み取りを正しい時点で再始動するには、アプリケーションで十分な情報を維持している必要があります。システム障害後に正しい時点から再始動する方法の詳細については、[MQPMO_LOGICAL_ORDER](#) と [MQGMO_LOGICAL_ORDER](#) を参照してください。

次の『使用上の注意』は、キュー・マネージャーで作業単位を調整する場合にのみ適用されます。

5. 作業単位の1つには、1つの接続ハンドルと同じ有効範囲があります。特定の作業単位に影響を与えるIBM MQ呼び出しは、同じ接続ハンドルを使用して実行しなければなりません。別の接続ハンドルを用いて呼び出しを発行すると(例えば、別のアプリケーションで呼び出しを発行する)、別の作業単位に影響が及びます。接続ハンドルの有効範囲については、MQCONNの項で説明している **Hconn** パラメーターを参照してください。
6. この呼び出しで影響を受けるメッセージは、現行の作業単位の一部として書き込まれたメッセージ、または取り出されたメッセージに限られます。
7. 長時間実行しているアプリケーションで、1つの作業単位に対してMQGET、MQPUT、またはMQPUT1呼び出しを発行している場合、コミット呼び出しまたはバックアウト呼び出しを一度も発行しないと、キューが他のアプリケーションでは使用できないメッセージで満杯になることがあります。これが発生しないようにするには、アドミニストレーターは、**MaxUncommittedMsgs** キュー・マネージャー属性を、ランナウェイ・アプリケーションがキューを満杯にしない程度の低い値に、かつ予期されるメッセージ交換アプリケーションが正常に作動できる程度に高い値に設定する必要があります。
8.  **ALW** AIX, Linux, and Windows システムでは、**Reason** パラメーターがMQRC_CONNECTION_BROKENの場合(そしてCompCodeがMQCC_FAILEDの場合)、あるいはMQRC_UNEXPECTED_ERRORの場合、作業単位は正常にコミットされた可能性があります。
9. アプリケーションが作業単位内に永続メッセージのプットを含み、MQPUTまたはMQPUT1呼び出しでMQPMO_ASYNC_RESPONSEオプションが使用され、かつプットが失敗した場合、MQCMITはMQCC_FAILED、MQRC_BACKED_OUTを返します。[MQPMOのフィールド詳細](#)にあるMQPMO_ASYNC_RESPONSEの説明を参照してください。

同様のシナリオにおいて、作業単位内の非永続メッセージに対して、MQCMITはMQCC_OK、MQRC_NONEを返します。

複数のプットが存在する場合で、MQPMO_ASYNC_RESPONSEが使用されていた場合、いずれかの永続メッセージのプットに失敗すると、MQCMIT呼び出しはMQCC_FAILED、MQRC_BACKED_OUTを返します。

C 言語での呼び出し

```
MQCMIT (Hconn, &CompCode, &Reason);
```

パラメーターを次のように宣言します。

```
MQHCONN  Hconn;      /* Connection handle */
MQLONG   CompCode;   /* Completion code */
MQLONG   Reason;     /* Reason code qualifying CompCode */
```

COBOLでの呼び出し

```
CALL 'MQCMIT' USING HCONN, COMPCODE, REASON.
```

パラメーターを次のように宣言します。

```

**      Connection handle
01      HCONN      PIC S9(9) BINARY.
**      Completion code
01      COMPCODE   PIC S9(9) BINARY.
**      Reason code qualifying COMPCODE
01      REASON     PIC S9(9) BINARY.

```

PL/I での呼び出し

```
call MQCMIT (Hconn, CompCode, Reason);
```

パラメーターを次のように宣言します。

```

dcl Hconn      fixed bin(31); /* Connection handle */
dcl CompCode   fixed bin(31); /* Completion code */
dcl Reason     fixed bin(31); /* Reason code qualifying CompCode */

```

高水準アセンブラー呼び出し

```
CALL MQCMIT,(HCONN,COMPCODE,REASON)
```

パラメーターを次のように宣言します。

```

HCONN      DS  F  Connection handle
COMPCODE   DS  F  Completion code
REASON     DS  F  Reason code qualifying COMPCODE

```

Visual Basic での呼び出し

```
MQCMIT Hconn, CompCode, Reason
```

パラメーターを次のように宣言します。

```

Dim Hconn      As Long 'Connection handle'
Dim CompCode   As Long 'Completion code'
Dim Reason     As Long 'Reason code qualifying CompCode'

```

MQCONN - キュー・マネージャーの接続

MQCONN 呼び出しは、アプリケーション・プログラムをキュー・マネージャーに接続します。

この呼び出しは、キュー・マネージャー接続ハンドルを提供します。アプリケーションはこの接続ハンドルを、後続のメッセージ・キューイング呼び出しで使用します。

- z/OS では、CICS アプリケーションはこの呼び出しを発行する必要はありません。これらのアプリケーションは、CICS システムが接続されているキュー・マネージャーに自動的に接続されます。ただし、MQCONN および MQDISC 呼び出しは、CICS アプリケーションからでも発行できます。
- IBM i では、アプリケーションは、キュー・マネージャーに接続するには MQCONN 呼び出しまたは MQCONNX 呼び出しを使用し、キュー・マネージャーから切断するには MQDISC 呼び出しを使用する必要があります。

サーバーのみのインストールでは、クライアント接続はできません。また、クライアントのみのインストールでは、ローカル接続は使用できません。

構文

```
MQCONN (QMgrName, Hconn, CompCode, Reason)
```

パラメーター

QMgrName

タイプ: MQCHAR48 - 入力

これは、アプリケーションが接続先にしたいキュー・マネージャーの名前です。この名前には、以下に示す文字を使用できます。

- 英大文字 (A から Z まで)
- 英小文字 (a から z まで)
- 数字 (0 から 9 まで)
- ピリオド (.), スラッシュ (/), 下線 (_), パーセント (%)

名前の先頭を空白にしたり、名前に空白を埋め込んだりすることはできませんが、名前の後に空白を入れることはできます。ヌル文字を使用して、名前の中における有効なデータの末尾を示すことができます。ヌル文字とそれに続く文字はすべて空白として扱われます。以下に示す制約事項は、それぞれ明記している環境に適用されます。

- EBCDIC カタカナを使用するシステムでは、小文字を使用できません。
- z/OS では、先頭または末尾に下線がある名前は、各操作や制御パネルで処理できません。そのため、そのような名前は避けてください。
- IBM i では、英小文字、スラッシュ、パーセントの各文字が含まれている名前をコマンドに指定するときは、それを引用符で囲みます。これらの引用符は、**QMgrName** パラメーターでは指定しないでください。

名前全体が空白で構成されている場合は、デフォルトのキュー・マネージャーの名前が使用されます。ただし、アプリケーション IBM MQ MQI client に関するセクションで説明されている空白のキューマネージャ名の使用に注意してください。

指定された名前は、接続可能なキューマネージャの名前、またはキューマネージャグループ **QMgrName** を使用している場合は、そのキューマネージャグループの名前でなければなりません。

z/OS では、接続が可能なキュー・マネージャーは、環境によって決まります。

- CICS の場合は、CICS システムが接続されたキュー・マネージャーのみが使用できます。**QMgrName** パラメーターも指定する必要がありますが、その値は無視されます。ここでは、空白文字にするのが適しています。
- IMS の場合は、サブシステム定義テーブル (CSQQDEFV) に表示されているキュー・マネージャーおよび IMS の SSM テーブルに表示されているキュー・マネージャーのみが接続可能です (使用上の注意 6 を参照)。
- z/OS バッチおよび TSO の場合は、アプリケーションと同じシステムにあるキュー・マネージャーのみが接続可能です (使用上の注意 6 を参照)。

キュー共有グループ: 複数のキュー・マネージャーが存在し、キュー共有グループを形成するように構成されるシステムでは、キュー共有グループの名前をキュー・マネージャーの代わりに **QMgrName** に指定することができます。これにより、アプリケーションは、キュー共有グループ内で利用可能であり、かつアプリケーションと同じ z/OS イメージ上にある任意のキューマネージャに接続できるようになります。また、**QMgrName** を空白にするとデフォルトのキュー・マネージャーではなくキュー共有グループに接続されるようにシステムを構成することもできます。

QMgrName がキュー共有グループの名前を指定し、その名前を持つキュー・マネージャーもシステムに存在する場合、その名前を持つキュー・マネージャーに優先的に接続されます。その接続が失敗した場合のみ、キュー共有グループ内のキュー・マネージャーの 1 つへの接続が試行されます。

接続が正常に行われた場合、MQCONN または MQCONNX 呼び出しで戻されたハンドルを使用して、接続が確立された先のキュー・マネージャーに所属するリソース (共有の場合も共有でない場合も) のすべてにアクセスすることができます。これらのリソースへのアクセスは、通常の許可制御に従います。

アプリケーションが並行する接続を確立するために MQCONN または MQCONNX 呼び出しを 2 つ発行して、どちらかの、または両方の呼び出しがキュー共有グループの名前を指定する場合、2 番目の呼び

出しが最初の呼び出しと同じキュー・マネージャーに接続すると、完了コード MQCC_WARNING および理由コード MQRC_ALREADY_CONNECTED を戻します。

キュー共有グループは、z/OS でのみサポートされています。キュー共有グループへの接続は、バッチ、RRS バッチ、CICS、および TSO 環境でのみサポートされます。CICS の場合は、CICS システムが接続されたキュー共有グループのみが使用できます。**QMgrName** パラメーターも指定する必要がありますが、その値は無視されます。ここでは、ブランク文字にするのが適しています。



重要: IMS はキュー共有グループに接続できません。

IBM MQ MQI client アプリケーション: IBM MQ MQI client アプリケーションの場合、指定されたキュー・マネージャー名を持つクライアント接続チャンネル定義ごとに 1 つの接続が成功するまで、接続が試行されます。ただし、キュー・マネージャーは、指定された名前と同じでなければなりません。名前全体がブランクで指定されている場合は、すべてブランクのキュー・マネージャー名を持つ各クライアント接続チャンネルが、成功するまで試行されます。この場合、キュー・マネージャーの実際の名前についての検査は行われません。

IBM MQ クライアント・アプリケーションは z/OS ではサポートされていませんが、z/OS は IBM MQ サーバーとして機能することができ、IBM MQ クライアント・アプリケーションが接続できるようになります。

IBM MQ MQI client キュー・マネージャー・グループ: 指定された名前がアスタリスク (*) で始まっている場合、接続されるキュー・マネージャーの名前は、アプリケーションによって指定された名前とは異なる場合があります。指定された名前 (アスタリスクなし) は、接続可能なキュー・マネージャーのグループを定義します。実装環境では、接続できるものが見つかるまで、グループ内で 1 個ずつ順に試行しながら 1 つ選択します。接続が試行される順序は、クライアント・チャンネルの重み付けや対象となるチャンネルの接続アフィニティの値が関係します。グループ内のキュー・マネージャーがどれも接続に使用できない場合、呼び出しは失敗します。各キュー・マネージャーは、1 回のみ試行されます。名前にアスタリスクのみが指定されている場合は、実施によって定義されたデフォルトのキュー・マネージャー・グループが使用されます。

キュー・マネージャー・グループは MQ クライアント環境で実行されるアプリケーションについてのみサポートされます。非クライアント・アプリケーションがアスタリスクで始まるキュー・マネージャー名を指定すると、この呼び出しは失敗します。グループ内の各キュー・マネージャーと通信するために、いくつかのクライアント接続チャンネル定義に同じキュー・マネージャー名 (アスタリスクなしで指定された名前) を指定することによって、グループが定義されます。デフォルト・グループの定義は、1 つ以上のクライアント接続チャンネル定義に、それぞれブランクのキュー・マネージャー名を付けることによって行われます (したがって、ブランクのみの名前を定義することは、クライアント・アプリケーションの名前に単一アスタリスクを指定するのと同様です)。

グループの 1 つのキュー・マネージャーに接続したあと、アプリケーションは、メッセージおよびオブジェクト記述子の中のキュー・マネージャー名フィールドに、通常の方法でブランクを指定することで、アプリケーションが接続されたキュー・マネージャーの名前 (ローカル・キュー・マネージャー) を表すことができます。アプリケーションがこの名前を認識する必要がある場合は、MQINQ 呼び出しを使用して **QMgrName** キュー・マネージャー属性を照会することができます。

接続名の先頭にアスタリスクを付けることは、アプリケーションはグループ内の特定のキュー・マネージャーへの接続に依存していないことを意味します。これが有効なアプリケーションは次のとおりです。

- メッセージは書き込むが、メッセージを読み取らないアプリケーション。
- 要求メッセージを書き込んでから、一時動的 キューから応答メッセージを取得するアプリケーション。

有効ではないアプリケーションは、特定のキュー・マネージャーの特定のキューからメッセージを読み取らなければならないアプリケーションなどです。このようなアプリケーションでは、名前の先頭にアスタリスクを付けてはなりません。

アスタリスクを指定する場合、名前の残りの最大長は 47 文字です。

このパラメーターの長さは MQ_Q_MGR_NAME_LENGTH によって指定されます。

Hconn

タイプ: MQHCONN - 出力

このハンドルは、キュー・マネージャーに対する接続を表します。これは、アプリケーションが発行する、以降のすべてのメッセージ・キューイング呼び出しで指定します。MQDISC 呼び出しが発行されたとき、またはハンドルの有効範囲を定義する処理の単位が終了したときに、有効でなくなります。

IBM MQ は、クライアント・パッケージでもサーバー・パッケージと同様に mqm ライブラリーを提供するようになりました。これはつまり、mqm ライブラリーにある MQI 呼び出しが行われる場合、接続タイプがクライアント接続かサーバー接続かが検査され、正しいことが確認された基礎となる呼び出しが呼び出されるということを意味します。そのため、Hconn に渡される出口が、mqm ライブラリーに対してリンク付けされますが、クライアント・インストールの環境で使用されます。

ハンドルの有効範囲: 戻されるハンドルの有効範囲は、キュー・マネージャーへの接続に使用される呼び出し (MQCONN または MQCONNX) によって決まります。使用される呼び出しが MQCONNX の場合は、ハンドルの有効範囲も、MQCNO 構造の Options フィールドに指定される

MQCNO_HANDLE_SHARE_* オプションによって決まります。

- 呼び出しが MQCONN である場合や、MQCNO_HANDLE_SHARE_NONE オプションが指定されている場合、戻されるハンドルは非共有ハンドルになります。

非共有ハンドルの有効範囲は、アプリケーションが実行されているプラットフォームによってサポートされる並列処理の最小単位です (詳細については 657 ページの表 582 を参照)。このハンドルは、呼び出しが発行された並列処理の単位の外側では無効です。

- 一方、MQCNO_HANDLE_SHARE_BLOCK オプションか MQCNO_HANDLE_SHARE_NO_BLOCK オプションを指定した場合、戻されるハンドルは共有ハンドルになります。

共有ハンドルの有効範囲は、呼び出しの発行元のスレッドを所有するプロセスであり、そのプロセスに属するすべてのスレッドでこのハンドルを使用することができます。ただし、すべてのプラットフォームでスレッドがサポートされているわけではありません。

- MQCONN 呼び出しまたは MQCONNX 呼び出しが MQCC_FAILED に等しい完了コードを出して失敗すると、Hconn 値は未定義となります。

表 582. 各種プラットフォームでの非共有ハンドルの有効範囲	
プラットフォーム	非共有ハンドルの有効範囲
z/OS	• CICS: CICS タスク • IMS: 次の同期点までのタスク (そのタスクのサブタスクは除く) • z/OS バッチおよび TSO: タスク (そのタスクのサブタスクは除く)
IBM i	ジョブ
AIX and Linux	スレッド
32 ビットの Windows アプリケーション	スレッド
64 ビット Windows アプリケーション	スレッド

z/OS 上で CICS アプリケーションを使用する場合、戻される値は以下のとおりです。

MQHC_DEF_HCONN

デフォルトの接続ハンドル。

CompCode

タイプ: MQLONG - 出力

完了コード。以下のいずれかです。

MQCC_OK

正常終了。

MQCC_WARNING
警告 (部分完了)。

MQCC_FAILED
呼び出し失敗。

理由

タイプ: MQLONG - 出力

CompCode が MQCC_OK の場合:

MQRC_NONE
(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_WARNING の場合:

MQRC_ALREADY_CONNECTED
(2002, X'7D2') アプリケーションはすでに接続されています。

MQRC_CLUSTER_EXIT_LOAD_ERROR
(2267, X'8DB') クラスタ・ワークロード出口をロードできません。

MQRC_SSL_ALREADY_INITIALIZED
(2391, X'957') SSL はすでに初期化されています。

CompCode が MQCC_FAILED の場合:

MQRC_ADAPTER_CONN_LOAD_ERROR
(2129, X'851') アダプター接続モジュールをロードできません。

MQRC_ADAPTER_DEFS_ERROR
(2131, X'853') アダプター・サブシステム定義モジュールが無効です。

MQRC_ADAPTER_DEFS_LOAD_ERROR
(2132, X'854') アダプター・サブシステム定義モジュールをロードできません。

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'89C') アダプターが利用できません。

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') アダプター・サービス・モジュールをロードできません。

MQRC_ADAPTER_STORAGE_SHORTAGE
(2127, X'84F') アダプター用のストレージが足りません。

MQRC_ANOTHER_Q_MGR_CONNECTED
(2103, X'837') 別のキュー・マネージャーがすでに接続されています。

MQRC_API_EXIT_ERROR
(2374, X'946') API 出口で障害が発生しました。

MQRC_API_EXIT_INIT_ERROR
(2375, X'947') API 出口の初期化に失敗しました。

MQRC_API_EXIT_TERM_ERROR
(2376, X'948') API 出口の終了に失敗しました。

MQRC_ASID_MISMATCH
(2157, X'86D') 1 次 ASID とホーム ASID が異なります。

MQRC_BUFFER_LENGTH_ERROR
(2005, X'7D5') バッファ長パラメーターは無効です。

MQRC_CALL_IN_PROGRESS
(2219, X'8AB') 前の呼び出しが完了する前に MQI 呼び出しが入力されました。

MQRC_CONN_ID_IN_USE
(2160, X'870') 接続 ID はすでに使用中です。

MQRC_CONNECTION_BROKEN
(2009, X'7D9') キュー・マネージャーとの接続が失われました。

MQRC_CONNECTION_ERROR

(2273, X'8E1') MQCONN 呼び出しの処理でエラーが発生しました。

MQRC_CONNECTION_NOT_AVAILABLE

(2568, X'A08') MQCONN または MQCONNX 呼び出しで、現行のインストール環境では、要求された接続タイプの接続をキュー・マネージャーが提供できない場合に発生します。サーバーのみのインストールでは、クライアント接続はできません。クライアントのみのインストールでは、ローカル接続はできません。

MQRC_CONNECTION QUIESCING

(2202, X'89A') 接続が静止しています。

MQRC_CONNECTION_STOPPING

(2203, X'89B') 接続がシャットダウン中です。

MQRC_CRYPTO_HARDWARE_ERROR

(2382, X'94E') 暗号ハードウェアに構成エラーがあります。

MQRC_DUPLICATE_RECOV_COORD

(2163, X'873') リカバリー・コーディネーターが存在します。

MQRC_ENVIRONMENT_ERROR

(2012, X'7DC') この環境では呼び出しが無効です。

さらに、MQCONNX 呼び出しでは、CICS または IMS アプリケーションから [286 ページの『MQCSP - セキュリティー・パラメーター』](#) 制御ブロックを渡します。

MQRC_HCONN_ERROR

(2018, X'7E2') 接続ハンドルが無効です。

MQRC_HOST_NOT_AVAILABLE

(2538, X'9EA') キュー・マネージャーに接続するためにクライアントから MQCONN 呼び出しが出されたが、リモート・システムへの会話の割り振りの試行に失敗しました。

MQRC_INSTALLATION_MISMATCH

(2583, X'A17') キュー・マネージャーのインストール済み環境と、選択されたライブラリーとが一致しません。

MQRC_KEY_REPOSITORY_ERROR

(2381, X'94D') キー・リポジトリが無効です。

MQRC_MAX_CONNS_LIMIT_REACHED

(2025, X'7E9') 接続が最大数に達しました。

MQRC_NOT_AUTHORIZED

(2035, X'7F3') アクセスは許可されません。

MQRC_OPEN_FAILED

(2137, X'859') オブジェクトが正常にオープンされていません。

MQRC_Q_MGR_NAME_ERROR

(2058, X'80A') キュー・マネージャー名が無効であるか、認識されていません。

MQRC_Q_MGR_NOT_AVAILABLE

(2059, X'80B') キュー・マネージャーを接続に使用できません。

MQRC_Q_MGR QUIESCING

(2161, X'871') キュー・マネージャーが静止しています。

MQRC_Q_MGR_STOPPING

(2162, X'872') キュー・マネージャーのシャットダウン中です。

MQRC_RESOURCE_PROBLEM

(2102, X'836') 使用できるシステム・リソースが不足しています。

MQRC_SECURITY_ERROR

(2063, X'80F') セキュリティー・エラーが発生しました。

MQRC_SSL_INITIALIZATION_ERROR

(2393, X'959') SSL 初期化エラーが発生しました。

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') ストレージが不足しています。

MQRC_UNEXPECTED_ERROR

(2195, X'893') 予期しないエラーが発生しました。

これらのコードについて詳しくは、[メッセージおよび理由コード](#)を参照してください。

使用上の注意

1. MQCONN 呼び出しを使用して接続が行われるキュー・マネージャーを、ローカル・キュー・マネージャーと呼びます。
2. ローカル・キュー・マネージャーが所有するキューは、アプリケーションでは、ローカル・キューとして扱われます。これらのキューに対しては、メッセージの書き込みと読み取りが可能です。

ローカル・キュー・マネージャーが所属するキュー共有グループが所有する共有キューは、アプリケーションでは、ローカル・キューとして扱われます。これらのキューに対しては、メッセージの書き込みと読み取りが可能です。

リモート・キュー・マネージャーが所有するキューは、リモート・キューとして扱われます。これらのキューに対しては、メッセージの書き込みは可能ですが、メッセージの読み取りはできません。

3. アプリケーションの実行中にキュー・マネージャーが失敗する場合、アプリケーションは再び MQCONN 呼び出しを発行して、後続の IBM MQ 呼び出しで使用する新規接続ハンドルを取得する必要があります。アプリケーションは、呼び出しが成功するまで定期的に MQCONN 呼び出しを発行することができます。

アプリケーションがキュー・マネージャーに接続されているかどうか分からない場合でも、安全に MQCONN 呼び出しを発行して接続ハンドルを取得することができます。アプリケーションがすでに接続されている場合は、戻されるハンドルは前に発行した MQCONN 呼び出しによって戻されたハンドルと同じですが、完了コード MQCC_WARNING と理由コード MQRC_ALREADY_CONNECTED も共に戻されます。

4. アプリケーションによる IBM MQ 呼び出しの使用が終了した場合、MQDISC 呼び出しを使用してキュー・マネージャーから切断する必要があります。
5. MQCONN 呼び出しが MQCC_FAILED に等しい完了コードを出して失敗すると、Hconn 値は未定義となります。
6. On z/OS:

- ・ バッチ、TSO、および IMS アプリケーションは、別の IBM MQ 呼び出しが使用するには MQCONN 呼び出しを発行する必要があります。これらのアプリケーションは、複数のキュー・マネージャーに並行して接続できます。

キュー・マネージャーが失敗する場合、アプリケーションは、キュー・マネージャーを再始動させた後に再び呼び出しを発行し、新規接続ハンドルを取得する必要があります。

すでに接続済みであっても、IMS アプリケーションは MQCONN 呼び出しを繰り返し発行することは可能ですが、オンライン・メッセージ処理プログラム (MPP) の場合にはお勧めできません。

- ・ CICS アプリケーションは、別の IBM MQ 呼び出しを使用するために MQCONN 呼び出しを発行する必要はありませんが、発行することは可能です。MQCONN 呼び出しおよび MQDISC 呼び出しの両方とも受け入れられます。しかし、複数のキュー・マネージャーに並行して接続することはできません。

キュー・マネージャーが失敗する場合、これらのアプリケーションはキュー・マネージャーが再始動したとき自動的に再接続するので、MQCONN 呼び出しを発行する必要はありません。

7. z/OS では、使用可能なキュー・マネージャーを定義するには次のようにします。

- ・ バッチ・アプリケーションの場合は、システム・プログラマーは、CSQBDEF マクロを使用して、デフォルトのキュー・マネージャー名またはキュー共有グループ名を定義するモジュール (CSQBDEFV) を作成することができます。
- ・ IMS アプリケーションの場合は、システム・プログラマーは、CSQQDEFX マクロを使用して、使用可能なキュー・マネージャーの名前を定義しデフォルトのキュー・マネージャーを指定するモジュール (CSQQDEFV) を作成することができます。

さらに、各キュー・マネージャーを、IMS 制御領域、およびそのキュー・マネージャーにアクセスする各従属領域に対して定義する必要があります。そのためには、IMS.PROCLIB ライブラリー内にサブシステム・メンバーを作成し、適用可能な IMS 領域に対してそのサブシステム・メンバーを識別する必要があります。IMS 領域用のサブシステム・メンバーに定義されていないキュー・マネージャーにアプリケーションが接続しようとする、このアプリケーションは異常終了します。

 これらのマクロの使用方法の詳細については、[お客様が使用するマクロを参照してください](#)。

8. IBM i では、異常終了するプログラムは、キュー・マネージャーから自動的に切断されません。MQCONN または MQCONNX 呼び出しが完了コード MQCC_WARNING および理由コード MQRC_ALREADY_CONNECTED を戻せるように、アプリケーションを作成する必要があります。この状況で戻される接続ハンドルを、通常どおり使用します。

C 言語での呼び出し

```
MQCONN (QMgrName, &Hconn, &CompCode, &Reason);
```

パラメーターを次のように宣言します。

```
MQCHAR48  QMgrName;  /* Name of queue manager */
MQHCONN    Hconn;     /* Connection handle */
MQLONG     CompCode;  /* Completion code */
MQLONG     Reason;    /* Reason code qualifying CompCode */
```

COBOL での呼び出し

```
CALL 'MQCONN' USING QMGRNAME, HCONN, COMPCODE, REASON.
```

パラメーターを次のように宣言します。

```
**      Name of queue manager
01  QMGRNAME  PIC X(48).
**      Connection handle
01  HCONN     PIC S9(9) BINARY.
**      Completion code
01  COMPCODE  PIC S9(9) BINARY.
**      Reason code qualifying COMPCODE
01  REASON    PIC S9(9) BINARY.
```

PL/I での呼び出し

```
call MQCONN (QMgrName, Hconn, CompCode, Reason);
```

パラメーターを次のように宣言します。

```
dcl QMgrName  char(48);      /* Name of queue manager */
dcl Hconn     fixed bin(31); /* Connection handle */
dcl CompCode  fixed bin(31); /* Completion code */
dcl Reason    fixed bin(31); /* Reason code qualifying CompCode */
```

高水準アセンブラー呼び出し

```
CALL MQCONN,(QMGRNAME,HCONN,COMPCODE,REASON)
```

パラメーターを次のように宣言します。

```
QMGRNAME  DS  CL48  Name of queue manager
HCONN      DS  F      Connection handle
COMPCODE   DS  F      Completion code
REASON     DS  F      Reason code qualifying COMPCODE
```

Visual Basic での呼び出し

```
MQCONN QMgrName, Hconn, CompCode, Reason
```

パラメーターを次のように宣言します。

```
Dim QMgrName As String*48 'Name of queue manager'  
Dim Hconn As Long 'Connection handle'  
Dim CompCode As Long 'Completion code'  
Dim Reason As Long 'Reason code qualifying CompCode'
```

MQCONNX - キュー・マネージャーの接続 (拡張)

MQCONNX 呼び出しは、アプリケーション・プログラムをキュー・マネージャーに接続します。この呼び出しによって取得したキュー・マネージャー接続ハンドルを使用して、アプリケーションは、それ以降の、IBM MQ 呼び出しを実行します。

MQCONNX 呼び出しは、MQCONN 呼び出しに似ていますが、MQCONNX では、呼び出しの動作を制御するオプションを指定できます。

- この呼び出しは、すべての IBM MQ システムおよびそれらのシステムに接続されている IBM MQ クライアント上でサポートされます。

サーバーのみのインストールでは、クライアント接続はできません。また、クライアントのみのインストールでは、ローカル接続は使用できません。

構文

MQCONNX (*QMgrName, ConnectOpts, Hconn, CompCode, Reason*)

パラメーター

QMgrName

タイプ: MQCHAR48 - 入力

詳細については、[654 ページの『MQCONN - キュー・マネージャーの接続』](#)の **QMgrName** パラメーターの説明を参照してください。

ConnectOpts

タイプ: MQCNO - 入出力

詳細は [265 ページの『MQCNO - 接続オプション』](#)を参照してください。

Hconn

タイプ: MQHCONN - 出力

このハンドルは、キュー・マネージャーに対する接続を表します。これは、アプリケーションが発行する、以降のすべてのメッセージ・キューイング呼び出しで指定します。MQDISC 呼び出しが発行されたとき、またはハンドルの有効範囲を定義する処理の単位が終了したときに、有効でなくなります。

IBM MQ は、クライアント・パッケージでもサーバー・パッケージと同様に mqm ライブラリーを提供するようになりました。これはつまり、mqm ライブラリーにある MQI 呼び出しが行われる場合、接続タイプがクライアント接続かサーバー接続かが検査され、正しいことが確認された基礎となる呼び出しが呼び出されるということを意味します。そのため、*Hconn* に渡される出口が、mqm ライブラリーに対してリンク付けされますが、クライアント・インストールの環境で使用されます。

ハンドルの有効範囲: 戻されるハンドルの有効範囲は、キュー・マネージャーへの接続に使用される呼び出し (MQCONN または MQCONNX) によって決まります。使用される呼び出しが MQCONNX の場合は、ハンドルの有効範囲も、MQCNO 構造の *Options* フィールドに指定される MQCNO_HANDLE_SHARE_* オプションによって決まります。

- 呼び出しが MQCONN である場合や、MQCNO_HANDLE_SHARE_NONE オプションが指定されている場合、戻されるハンドルは非共有ハンドルになります。

非共有ハンドルの有効範囲は、アプリケーションが実行されているプラットフォームによってサポートされる並列処理の最小単位です（詳細については [663 ページの表 583](#) を参照）。このハンドルは、呼び出しが発行された並列処理の単位の外側では無効です。

- 一方、MQCNO_HANDLE_SHARE_BLOCK オプションか MQCNO_HANDLE_SHARE_NO_BLOCK オプションを指定した場合、戻されるハンドルは共有ハンドルになります。

共有ハンドルの有効範囲は、呼び出しの発行元のスレッドを所有するプロセスであり、そのプロセスに属するすべてのスレッドでこのハンドルを使用することができます。ただし、すべてのプラットフォームでスレッドがサポートされているわけではありません。

- MQCONN 呼び出しまたは MQCONNX 呼び出しが MQCC_FAILED に等しい完了コードを出して失敗すると、Hconn 値は未定義となります。

表 583. 各種プラットフォームでの非共有ハンドルの有効範囲	
プラットフォーム	非共有ハンドルの有効範囲
z/OS	• CICS: CICS タスク • IMS: 次の同期点までのタスク (そのタスクのサブタスクは除く) • z/OS バッチおよび TSO: タスク (そのタスクのサブタスクは除く)
IBM i	ジョブ
AIX and Linux	スレッド
32 ビットの Windows アプリケーション	スレッド
64 ビット Windows アプリケーション	スレッド

z/OS 上で CICS アプリケーションを使用する場合、戻される値は以下のとおりです。

MQHC_DEF_HCONN

デフォルトの接続ハンドル。

CompCode

タイプ: MQLONG - 出力

詳細については、[654 ページの『MQCONN - キュー・マネージャーの接続』](#)の **CompCode** パラメータの説明を参照してください。

理由

タイプ: MQLONG - 出力

MQCONN および MQCONNX 呼び出しから返されるコードは、以下のとおりです。MQCONNX 呼び出しから返される追加コードのリストとして、以下のコードを参照してください。

CompCode が MQCC_OK の場合:

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_WARNING の場合:

MQRC_ALREADY_CONNECTED

(2002, X'7D2') アプリケーションはすでに接続されています。

MQRC_CLUSTER_EXIT_LOAD_ERROR

(2267, X'8DB') クラスター・ワークロード出口をロードできません。

MQRC_SSL_ALREADY_INITIALIZED

(2391, X'957') SSL はすでに初期化されています。

CompCode が MQCC_FAILED の場合:

MQRC_ADAPTER_CONN_LOAD_ERROR

(2129, X'851') アダプター接続モジュールをロードできません。

MQRC_ADAPTER_DEFS_ERROR

(2131, X'853') アダプター・サブシステム定義モジュールが無効です。

MQRC_ADAPTER_DEFS_LOAD_ERROR

(2132, X'854') アダプター・サブシステム定義モジュールをロードできません。

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'89C') アダプターが利用できません。

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852') アダプター・サービス・モジュールをロードできません。

MQRC_ADAPTER_STORAGE_SHORTAGE

(2127, X'84F') アダプター用のストレージが足りません。

MQRC_ANOTHER_Q_MGR_CONNECTED

(2103, X'837') 別のキュー・マネージャーがすでに接続されています。

MQRC_API_EXIT_ERROR

(2374, X'946') API 出口で障害が発生しました。

MQRC_API_EXIT_INIT_ERROR

(2375, X'947') API 出口の初期化に失敗しました。

MQRC_API_EXIT_TERM_ERROR

(2376, X'948') API 出口の終了に失敗しました。

MQRC_ASID_MISMATCH

(2157, X'86D') 1 次 ASID とホーム ASID が異なっています。

MQRC_BUFFER_LENGTH_ERROR

(2005, X'7D5') バッファー長パラメーターは無効です。

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') 前の呼び出しが完了する前に MQI 呼び出しが入力されました。

MQRC_CONN_ID_IN_USE

(2160, X'870') 接続 ID はすでに使用中です。

MQRC_CONNECTION_BROKEN

(2009, X'7D9') キュー・マネージャーとの接続が失われました。

MQRC_CONNECTION_ERROR

(2273, X'8E1') MQCONN 呼び出しの処理でエラーが発生しました。

MQRC_CONNECTION_NOT_AVAILABLE

(2568, X'A08') MQCONN または MQCONNX 呼び出しで、現行のインストール環境では、要求された接続タイプの接続をキュー・マネージャーが提供できない場合に発生します。サーバーのみのインストールでは、クライアント接続はできません。クライアントのみのインストールでは、ローカル接続はできません。

MQRC_CONNECTION QUIESCING

(2202, X'89A') 接続が静止しています。

MQRC_CONNECTION_STOPPING

(2203, X'89B') 接続がシャットダウン中です。

MQRC_CRYPTO_HARDWARE_ERROR

(2382, X'94E') 暗号ハードウェアに構成エラーがあります。

MQRC_DUPLICATE_RECOV_COORD

(2163, X'873') リカバリー・コーディネーターが存在します。

MQRC_ENVIRONMENT_ERROR

(2012, X'7DC') この環境では呼び出しが無効です。

さらに、MQCONNX 呼び出しでは、CICS または IMS アプリケーションから [286 ページの『MQCSP - セキュリティー・パラメーター』](#) 制御ブロックを渡します。

MQRC_HCONN_ERROR

(2018, X'7E2') 接続ハンドルが無効です。

MQRC_HOST_NOT_AVAILABLE

(2538, X'9EA') キュー・マネージャーに接続するためにクライアントから MQCONN 呼び出しが出されたが、リモート・システムへの会話の割り振りの試行に失敗しました。

MQRC_INSTALLATION_MISMATCH

(2583, X'A17') キュー・マネージャーのインストール済み環境と、選択されたライブラリーとが一致しません。

MQRC_KEY_REPOSITORY_ERROR

(2381, X'94D') キー・リポジトリが無効です。

MQRC_MAX_CONNS_LIMIT_REACHED

(2025, X'7E9') 接続が最大数に達しました。

MQRC_NOT_AUTHORIZED

(2035, X'7F3') アクセスは許可されません。

MQRC_OPEN_FAILED

(2137, X'859') オブジェクトが正常にオープンされていません。

MQRC_Q_MGR_NAME_ERROR

(2058, X'80A') キュー・マネージャー名が無効であるか、認識されていません。

MQRC_Q_MGR_NOT_AVAILABLE

(2059, X'80B') キュー・マネージャーを接続に使用できません。

MQRC_Q_MGR_QUIESCING

(2161, X'871') キュー・マネージャーが静止しています。

MQRC_Q_MGR_STOPPING

(2162, X'872') キュー・マネージャーのシャットダウン中です。

MQRC_RESOURCE_PROBLEM

(2102, X'836') 使用できるシステム・リソースが不足しています。

MQRC_SECURITY_ERROR

(2063, X'80F') セキュリティー・エラーが発生しました。

MQRC_SSL_INITIALIZATION_ERROR

(2393, X'959') SSL 初期化エラーが発生しました。

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') ストレージが不足しています。

MQRC_UNEXPECTED_ERROR

(2195, X'893') 予期しないエラーが発生しました。

MQCONN 呼び出しから返される追加の理由コードは、以下のとおりです。

CompCode が MQCC_FAILED の場合:

MQRC_AIR_ERROR

(2385, X'951') 認証情報レコードが無効です。

MQRC_AUTH_INFO_CONN_NAME_ERROR

(2387, X'953') 認証情報接続名が無効です。

MQRC_AUTH_INFO_REC_COUNT_ERROR

(2383, X'94F') 認証情報レコード・カウントが無効です。

MQRC_AUTH_INFO_REC_ERROR

(2384, X'950') 認証情報レコード・フィールドが無効です。

MQRC_AUTH_INFO_TYPE_ERROR

(2386, X'952') 認証情報タイプが無効です。

MQRC_CD_ERROR

(2277, X'8E5') チャンネル定義が無効です。

MQRC_CLIENT_CONN_ERROR

(2278, X'8E6') クライアント接続フィールドが無効です。

MQRC_CNO_ERROR

(2139, X'85B') 接続オプション構造体が無効です。

MQRC_CONN_TAG_IN_USE

(2271, X'8DF') 接続タグが使用されています。

MQRC_CONN_TAG_NOT_USABLE

(2350, X'92E') 接続タグが使用不可です。

MQRC_CSP_ERROR

(2595, X' A23 ') MQCSP 構造が無効です。

MQRC_FUNCTION_NOT_SUPPORTED

(2298, X'8FA') 要求された関数は、現在の環境では使用できない。

MQRC_LDAP_PASSWORD_ERROR

(2390, X'956') LDAP パスワードが無効です。

MQRC_LDAP_USER_NAME_ERROR

(2388, X'954') LDAP ユーザー名フィールドが無効です。

MQRC_LDAP_USER_NAME_LENGTH_ERR

(2389, X'955') LDAP ユーザー名の長さが無効です。

MQRC_OPTIONS_ERROR

(2046, X'7FE') オプションが無効であるか、矛盾しています。

MQRC_SCO_ERROR

(2380, X'94C') SSL 構成オプション構造体が無効です。

MQRC_SSL_CONFIG_ERROR

(2392, X'958') SSL 構成エラーです。

MQRC_TOKEN_TIMESTAMP_NOT_VALID

(2064, X'810') 認証トークンはまだ有効でないか、期限切れです。

これらのコードについて詳しくは、[メッセージおよび理由コード](#)を参照してください。

使用上の注意

Visual Basic プログラミング言語には、以下の点が適用されます。

- **ConnectOpts** パラメーターはタイプ MQCNO として宣言されます。アプリケーションを IBM MQ MQI client として実行する場合にクライアント接続チャンネルのパラメーターを指定するには、**ConnectOpts** パラメーターをタイプ Any として宣言します。これにより、アプリケーションは、呼び出しの時に MQCNO 構造体の代わりに MQCNOCD 構造体を指定できるようになります。しかし、**ConnectOpts** パラメーターを検査して正しいデータ・タイプかどうかを確認することができなくなります。

C 言語での呼び出し

```
MQCONN (QMgrName, &ConnectOpts, &Hconn, &CompCode, &Reason);
```

パラメーターを次のように宣言します。

```
MQCHAR48  QMgrName;      /* Name of queue manager */
MQCNO     ConnectOpts;   /* Options that control the action of MQCONN */
MQHCONN   Hconn;         /* Connection handle */
MQLONG    CompCode;      /* Completion code */
MQLONG    Reason;        /* Reason code qualifying CompCode */
```

COBOL での呼び出し

```
CALL 'MQCONN' USING QMGRNAME, CONNECTOPTS, HCONN, COMPCODE,
REASON.
```

パラメーターを次のように宣言します。

```
**      Name of queue manager
01  QMGRNAME      PIC X(48).
**      Options that control the action of MQCONN
01  CONNECTOPTS.
    COPY CMQCNNOV.
**      Connection handle
01  HCONN         PIC S9(9) BINARY.
**      Completion code
01  COMPCODE      PIC S9(9) BINARY.
**      Reason code qualifying COMPCODE
01  REASON        PIC S9(9) BINARY.
```

PL/I での呼び出し

```
call MQCONN (QMgrName, ConnectOpts, Hconn, CompCode, Reason);
```

パラメーターを次のように宣言します。

```
dcl QMgrName      char(48);      /* Name of queue manager */
dcl ConnectOpts   like MQCNO;    /* Options that control the action of
                                MQCONN */
dcl Hconn         fixed bin(31); /* Connection handle */
dcl CompCode      fixed bin(31); /* Completion code */
dcl Reason        fixed bin(31); /* Reason code qualifying CompCode */
```

高水準アセンブラー呼び出し

```
CALL MQCONN, (QMGRNAME,CONNECTOPTS,HCONN,COMPCODE,REASON)
```

パラメーターを次のように宣言します。

QMGRNAME	DS	CL48	Name of queue manager
CONNECTOPTS	CMQCNNOA	,	Options that control the action of MQCONN
HCONN	DS	F	Connection handle
COMPCODE	DS	F	Completion code
REASON	DS	F	Reason code qualifying COMPCODE

Visual Basic での呼び出し

```
MQCONN QMgrName, ConnectOpts, Hconn, CompCode, Reason
```

パラメーターを次のように宣言します。

```
Dim QMgrName As String*48 'Name of queue manager'
Dim ConnectOpts As MQCNO 'Options that control the action of'
                          'MQCONN'
Dim Hconn As Long 'Connection handle'
Dim CompCode As Long 'Completion code'
Dim Reason As Long 'Reason code qualifying CompCode'
```

MQCRTMH - メッセージ・ハンドルの作成

MQCRTMH 呼び出しは、メッセージ・ハンドルを戻します。

アプリケーションは後続のメッセージ・キューイング呼び出しで、MQCRTMH 呼び出しを使用できます。

- [MQSETMP](#) 呼び出しを使用して、メッセージ・ハンドルのプロパティーを設定します。
- [MQINQMP](#) 呼び出しを使用して、メッセージ・ハンドルのプロパティーの値を照会します。
- [MQDLTMP](#) 呼び出しを使用して、メッセージ・ハンドルのプロパティーを削除します。

メッセージ・ハンドルを MQPUT および MQPUT1 呼び出し上で使用して、メッセージ・ハンドルのプロパティーを、書き込まれるメッセージのプロパティーと関連付けることができます。同様に、メッセージ・

ハンドルを MQGET 呼び出し上で指定して、MQGET 呼び出しの完了時にメッセージ・ハンドルを使用して、取り出されるメッセージのプロパティーにアクセスできます。

MQDLTMH を使用してメッセージ・ハンドルを削除します。

構文

MQCRTMH (*Hconn*, *CrtMsgHOpts*, *Hmsg*, *CompCode*, *Reason*)

パラメーター

Hconn

タイプ: MQHCONN - 入力

このハンドルは、キュー・マネージャーに対する接続を表します。 *Hconn* の値は、先行の MQCONN または MQCONNX 呼び出しによって戻されたものです。キュー・マネージャーへの接続が無効になり、メッセージ・ハンドル上で IBM MQ 呼び出しが作動していない場合は、MQDLTMH が暗黙的に呼び出されてメッセージを削除します。

あるいは、以下の値を指定することができます。

MQHC_UNASSOCIATED_HCONN

接続ハンドルは特定のキュー・マネージャーに対する接続を表しません。

この値を使用する場合、メッセージ・ハンドルに割り振られたストレージを解放するために、MQDLTMH を明示的に呼び出してメッセージ・ハンドルを削除する必要があります。IBM MQ が暗黙的にメッセージ・ハンドルを削除することはありません。

メッセージ・ハンドルを作成するスレッド上に、キュー・マネージャーへの有効な接続が少なくとも 1 つ確立されていることが必要です。確立されていない場合、呼び出しは MQRC_HCONN_ERROR で失敗します。

単一システム上に複数のインストールがある環境では、プロセスにロードされる最初のインストールでのみ MQHC_UNASSOCIATED_HCONN の値が使用されるという制限があります。それ以外のインストールに対してメッセージ・ハンドルが提供されている場合、理由コード MQRC_HMSG_NOT_AVAILABLE が返されます。

z/OS 上で CICS アプリケーションを使用する場合は、MQCONN 呼び出しを省略でき、*Hconn* に次の値を指定できます。

MQHC_DEF_HCONN

デフォルトの接続ハンドル。

CrtMsgHOpts

タイプ: MQCMHO - 入力

MQCRTMH のアクションを制御するオプション。詳細については、MQCMHO を参照してください。

Hmsg

タイプ: MQHMSG - 出力

出力で、メッセージ・ハンドルのプロパティーの設定、照会、および削除に使用できるメッセージ・ハンドルが戻されます。当初、メッセージ・ハンドルにプロパティーは含まれていません。

メッセージ・ハンドルには、メッセージ記述子も関連付けられます。初期状態では、デフォルト値が入っています。関連付けられたメッセージ記述子フィールドの値は、MQSETMP および MQINQMP 呼び出しを使用して設定および照会できます。MQDLTMP 呼び出しは、メッセージ記述子のフィールドをリセットして、デフォルト値に戻します。

Hconn パラメーターが値 MQHC_UNASSOCIATED_HCONN として指定されている場合は、戻されるメッセージ・ハンドルを MQGET、MQPUT、または MQPUT1 呼び出しで処理単位内の接続と併用できますが、一度に 1 つの IBM MQ 呼び出しのみで使用できます。2 番目の IBM MQ 呼び出しで同じメッセージ・ハンドルを使用しようとした際に、そのハンドルが使用中の場合は、2 番目の IBM MQ 呼び出しは失敗し、理由コード MQRC_MSG_HANDLE_IN_USE が戻ります。

Hconn パラメーターが MQHC_UNASSOCIATED_HCONN でない場合、戻されるメッセージ・ハンドルは、指定された接続のみで使用できます。

このメッセージ・ハンドルが使用される以後の MQI 呼び出しでは、同じ *Hconn* パラメーター値を使用する必要があります。

- MQDLTMH
- MQSETMP
- MQINQMP
- MQDLTMP
- MQMHBUF
- MQBUFMH

戻されるメッセージ・ハンドルは、このメッセージ・ハンドルに MQDLTMH 呼び出しが発行されたとき、またはハンドルの有効範囲を定義する処理の単位が終了したときに無効になります。メッセージ・ハンドルの作成時に特定の接続が提供され、キュー・マネージャーに対するこの接続が無効になった場合 (例えば、MQDBC が呼び出された場合)、MQDLTMH が暗黙的に呼び出されます。

CompCode

タイプ: MQLONG - 出力

完了コード。以下のいずれかです。

MQCC_OK

正常終了。

MQCC_FAILED

呼び出し失敗。

理由

タイプ: MQLONG - 出力

CompCode が MQCC_OK の場合:

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_FAILED の場合:

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'089C') アダプターが利用できません。

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852') アダプター・サービス・モジュールをロードできません。

MQRC_ASID_MISMATCH

(2157, X'86D') 1 次 ASID とホーム ASID が異なっています。

MQRC_CALL_IN_PROGRESS

(2219, X'08AB') 前の呼び出しが完了する前に MQI 呼び出しが入力された。

MQRC_CMHO_ERROR

(2461, X'099D') メッセージ・ハンドル作成オプションの構造が無効です。

MQRC_CONNECTION_BROKEN

(2273, X'7D9') キュー・マネージャーへの接続が失われました。

MQRC_HANDLE_NOT_AVAILABLE

(2017, X'07E1') 使用可能なハンドルがなくなりました。

MQRC_HCONN_ERROR

(2018, X'7E2') 接続ハンドルが無効です。

MQRC_HMSG_ERROR

(2460, X'099C') メッセージ・ハンドル・ポインターが無効。

MQRC_OPTIONS_ERROR

(2046, X'07FE') オプションが無効であるか、矛盾しています。

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') ストレージが不足しています。

MQRC_UNEXPECTED_ERROR

(2195, X'893') 予期しないエラーが発生しました。

これらのコードについて詳しくは、[メッセージおよび理由コード](#)を参照してください。

C

```
MQCRTMH (Hconn, &CrtMsgHOpts, &Hmsg, &CompCode, &Reason);
```

パラメーターを次のように宣言します。

```
MQHCONN  Hconn;          /* Connection handle */
MQCMHO   CrtMsgHOpts;    /* Options that control the action of MQCRTMH */
MQHMSG   Hmsg;           /* Message handle */
MQLONG   CompCode;       /* Completion code */
MQLONG   Reason;         /* Reason code qualifying CompCode */
```

COBOL

```
CALL 'MQCRTMH' USING HCONN, CRTMSGHOPTS, HMSG, COMPCODE, REASON.
```

パラメーターを次のように宣言します。

```
** Connection handle
01 HCONN    PIC S9(9) BINARY.
** Options that control the action of MQCRTMH
01 CRTMSGHOPTS.
   COPY CMQCMHOV.
** Message handle
01 HMSG     PIC S9(18) BINARY.
** Completion code
01 COMPCODE PIC S9(9) BINARY.
** Reason code qualifying COMPCODE
01 REASON   PIC S9(9) BINARY.
```

PL/I

```
call MQCRTMH (Hconn, CrtMsgHOpts, Hmsg, CompCode, Reason);
```

パラメーターを次のように宣言します。

```
dcl Hconn      fixed bin(31); /* Connection handle */
dcl CrtMsgHOpts like MQCMHO;  /* Options that control the action of MQCRTMH */
dcl Hmsg       fixed bin(63); /* Message handle */
dcl CompCode   fixed bin(31); /* Completion code */
dcl Reason     fixed bin(31); /* Reason code qualifying CompCode */
```

High Level Assembler

```
CALL MQCRTMH,(HCONN,CRTMSGHOPTS,HMSG,COMPCODE,REASON)
```

パラメーターを次のように宣言します。

```
HCONN      DS      F  Connection handle
CRTMSGHOPTS CMQCMHOA ,  Options that control the action of MQCRTMH
HMSG       DS      D  Message handle
COMPCODE   DS      F  Completion code
REASON     DS      F  Reason code qualifying COMPCODE
```

MQCTL - コールバック制御

MQCTL 呼び出しは、接続に対してオープンされたコールバックおよびオブジェクト処理に対する制御アクションを実行します。

構文

MQCTL (*Hconn*, *Operation*, *ControlOpts*, *CompCode*, *Reason*)

パラメーター

Hconn

タイプ: MQHCONN - 入力

このハンドルは、キュー・マネージャーに対する接続を表します。 *Hconn* の値は、先行の MQCONN または MQCONNX 呼び出しによって戻されたものです。

z/OS 上で CICS アプリケーションを使用する場合は、MQCONN 呼び出しを省略でき、*Hconn* に次の特殊値を指定できます。

MQHC_DEF_HCONN

デフォルトの接続ハンドル。

Operation

タイプ: MQLONG - 入力

指定されたオブジェクト・ハンドルに定義されたコールバックで処理されている操作。以下のオプションのうち、いずれか 1 つだけを指定する必要があります。

MQOP_START

指定された接続ハンドルについて定義されているすべてのメッセージ・コンシューマー関数のためのメッセージのコンシュームを開始します。

コールバックは、システムによって開始されるスレッド上で実行されます。それはアプリケーション・スレッドのいずれとも異なります。

この操作は、提供された接続ハンドルの制御をシステムに渡します。コンシューマー・スレッド以外のスレッドから発行できる MQI 呼び出しは、以下のものだけです。

- Operation が MQOP_STOP の MQCTL
- Operation が MQOP_SUSPEND の MQCTL
- MQDISC - HConn の切断前に Operation が MQOP_STOP の MQCTL を実行します。

MQRC_HCONN_ASYNC_ACTIVE は、接続ハンドルの開始中に IBM MQ API 呼び出しが発行され、この呼び出しがメッセージ・コンシューマー機能から発信されていない場合に戻されます。

MQCBCT_START_CALL 中にメッセージ・コンシューマーが接続を停止すると、MQCTL 呼び出しは失敗理由コード MQRC_CONNECTION_STOPPED で戻ります。

この呼び出しは、コンシューマー関数で発行できます。コールバック・ルーチンと同じ接続の場合、以前に発行された MQOP_STOP 操作の取り消しのみが目的です。

このオプションは、以下の環境ではサポートされていません： CICS ・ IBM MQ 非スレッドライブラリとバインドされている場合 ・ z/OS または

MQOP_START_WAIT

指定された接続ハンドルについて定義されているすべてのメッセージ・コンシューマー関数のためのメッセージのコンシュームを開始します。

メッセージ・コンシューマーは同じスレッド上で実行されます。以下のことが発生する時点まで、制御は MQCTL の呼び出し側に戻されません。

- MQCTL MQOP_STOP または MQOP_SUSPEND 操作を使用して解放された時点、または
- すべてのコンシューマー・ルーチンが登録解除されたか中断された時点。

すべてのコンシューマー・ルーチンが登録解除されるか中断される場合、暗黙の MQOP_STOP 操作が発行されます。

現行接続ハンドルまたはその他の接続ハンドルのいずれかについて、コールバック・ルーチン内からこのオプションを使用することはできません。呼び出しが試行されると、MQRC_ENVIRONMENT_ERROR で戻ります。

MQOP_START_WAIT 操作中のいずれかの時点で、登録も中断もしていないコンシューマーがある場合は、呼び出しは理由コード MQRC_NO_CALLBACKS_ACTIVE で失敗します。

MQOP_START_WAIT 操作中に接続が中断されると、MQCTL 呼び出しは警告の理由コード MQRC_CONNECTION_SUSPENDED を返します。この時点で接続は「開始済み」のままです。

アプリケーションは MQOP_STOP または MQOP_RESUME の発行を選択できます。この場合、MQOP_RESUME 操作はブロックします。

このオプションは、単一スレッド・クライアントではサポートされていません。

MQOP_STOP

メッセージのコンシュームを停止し、すべてのコンシューマーがそれぞれの操作を完了するのを待機します。その後、このオプションが完了します。この操作は、接続ハンドルを解放します。

コールバック・ルーチン内から発行した場合、そのルーチンが終了する時点までこのオプションは有効になりません。既に読んだメッセージのコンシューマー・ルーチンが完了し、コールバック・ルーチンの停止呼び出しが要求されて実行された後は、それ以上メッセージ・コンシューマー・ルーチンは呼び出されません。

コールバック・ルーチン外から発行された場合、既に読んだメッセージのコンシューマー・ルーチンが完了し、コールバックの停止呼び出しが要求されて実行された後、制御は呼び出し元に戻されません。しかし、コールバック自体は登録済みのままです。

この関数は、先読みメッセージに対しては何の効果もありません。コンシューマーがコールバック関数内から MQCLOSE(MQCO_QUIESCE) を実行して、配布可能なメッセージがさらにあるかどうかを判別することを確認しなければなりません。

MQOP_SUSPEND

メッセージのコンシュームを休止します。この操作は、接続ハンドルを解放します。

この機能は、アプリケーションに関するメッセージの先読みに対しては影響しません。長期間メッセージのコンシュームを停止する場合は、キューをクローズし、コンシュームを続行する時に再オープンすることを確認してください。

コールバック・ルーチン内から発行した場合、そのルーチンが終了する時点までは有効になりません。現在のルーチンが終了すると、その後、メッセージ・コンシューマー・ルーチンは呼び出されなくなります。

コールバック外から発行された場合、現在のコンシューマー・ルーチンが完了して、それ以降にルーチンが呼び出されなくなるまで、制御は呼び出し元に戻されません。

MQOP_RESUME

メッセージのコンシュームを再開します。

通常このオプションはメイン・アプリケーション・スレッドから発行されますが、コールバック・ルーチン内から、同じルーチン内でそれより前に発行された中断要求を取り消す目的で使用することも可能です。

MQOP_RESUME を使用して MQOP_START_WAIT を再開すると、操作はブロックします。

ControlOpts

タイプ: MQCTLO - 入力

MQCTL のアクションを制御するオプション

この構造の詳細については、[MQCTLO](#) を参照してください。

CompCode

タイプ: MQLONG - 出力

完了コード。以下のいずれかです。

MQCC_OK

正常終了。

MQCC_WARNING

警告 (部分完了)。

MQCC_FAILED

呼び出し失敗。

理由

タイプ: MQLONG - 出力

CompCode が MQCC_OK の場合:

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_FAILED の場合:

MQRC_ADAPTER_CONV_LOAD_ERROR

(2133, X'855') データ変換サービス・モジュールをロードできない。

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'89C') アダプターが利用できません。

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852') アダプター・サービス・モジュールをロードできません。

MQRC_API_EXIT_ERROR

(2374, X'946') API 出口で障害が発生しました。

MQRC_API_EXIT_LOAD_ERROR

(2183, X'887') API 出口をロードできません。

MQRC_ASID_MISMATCH

(2157, X'86D') 1 次 ASID とホーム ASID が異なっています。

MQRC_BUFFER_LENGTH_ERROR

(2005, X'7D5') バッファー長パラメーターは無効です。

MQRC_CALLBACK_LINK_ERROR

(2487, X'9B7') コールバック・ルーチンを呼び出せない

MQRC_CALLBACK_NOT_REGISTERED

(2448, X'990') コールバックが登録されていないので、登録解除、中断、または再開できない

MQRC_CALLBACK_ROUTINE_ERROR

(2486, X'9B6') MQOP_REGISTER 呼び出し上で CallbackFunction と CallbackName の両方が指定されている。

または、CallbackFunction か CallbackName のどちらかが指定されているが、現在登録されているコールバック関数と一致しない。

MQRC_CALLBACK_TYPE_ERROR

(2483, X'9B3') CallBackType フィールドが正しくない。

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') 前の呼び出しが完了する前に MQI 呼び出しが入力されました。

MQRC_CBD_ERROR

(2444, X'98C') オプション・ブロックが正しくない。

MQRC_CBD_OPTIONS_ERROR

(2484, X'9B4') MQCBD オプション・フィールドが正しくない。

MQRC_CICS_WAIT_FAILED

(2140, X'85C') 待機要求が CICS により拒否された。

MQRC_CONNECTION_BROKEN

(2009, X'7D9') キュー・マネージャーとの接続が失われました。

MQRC_CONNECTION_NOT_AUTHORIZED

(2217, X'8A9') 接続が許可されていません。

MQRC_CONNECTION QUIESCING

(2202, X'89A') 接続が静止しています。

MQRC_CONNECTION_STOPPING

(2203, X'89B') 接続がシャットダウン中です。

MQRC_CORREL_ID_ERROR

(2207, X'89F') 相関 ID のエラー。

MQRC_ENVIRONMENT_ERROR

(2012, X'7DC') この環境では呼び出しが無効です。

MQRC_FUNCTION_NOT_SUPPORTED

(2298, X'8FA') 要求された関数は、現在の環境では使用できない。

MQRC_GET_INHIBITED

(2016, X'7E0') キューからの読み取りが禁止されている。

MQRC_GLOBAL_UOW_CONFLICT

(2351, X'92F') グローバル作業単位に矛盾がある。

MQRC_GMO_ERROR

(2186, X'88A') 読み取りメッセージ・オプションの構造体が無効である。

MQRC_HANDLE_IN_USE_FOR_UOW

(2353, X'931') グローバル作業単位のためのハンドルが使用中。

MQRC_HCONN_ERROR

(2018, X'7E2') 接続ハンドルが無効です。

MQRC_HOBJ_ERROR

(2019, X'7E3') オブジェクト・ハンドルが無効です。

MQRC_INCONSISTENT_BROWSE

(2259, X'8D3') ブラウズの指定が不整合である。

MQRC_INCONSISTENT_UOW

(2245, X'8C5') 作業単位の指定が不整合である。

MQRC_INVALID_MSG_UNDER_CURSOR

(2246, X'8C6') カーソル下のメッセージが取り出し対象として無効である。

MQRC_LOCAL_UOW_CONFLICT

(2352, X'930') グローバル作業単位とローカル作業単位に矛盾がある。

MQRC_MATCH_OPTIONS_ERROR

(2247, X'8C7') 突き合わせオプションが無効である。

MQRC_MAX_MSG_LENGTH_ERROR

(2485, X'9B5') MaxMsgLength フィールドが正しくない

MQRC_MD_ERROR

(2026, X'7EA') メッセージ記述子が無効である。

MQRC_MODULE_ENTRY_NOT_FOUND

(2497, X'9C1') 指定された機能入り口点がモジュール中になかった。

MQRC_MODULE_INVALID

(2496, X'9C0') モジュールが見つかったが、タイプが間違っている (32 ビット/64 ビット) か、有効な DLL ではない。

MQRC_MODULE_NOT_FOUND

(2495, X'9BF') モジュールが検索パス中にないか、またはロードが許可されていない。

MQRC_MSG_ID_ERROR

(2206, X'89E') メッセージ ID のエラー。

MQRC_MSG_SEQ_NUMBER_ERROR

(2250, X'8CA') メッセージ順序番号が無効である。

MQRC_MSG_TOKEN_ERROR

(2331, X'91B') メッセージ・トークンについて無効な使い方をしている。

MQRC_NOT_OPEN_FOR_BROWSE

(2036, X'7F4') ブラウズのためにキューがオープンされていない。

MQRC_NOT_OPEN_FOR_INPUT

(2037, X'7F5') 入力のためにキューがオープンされていない。

MQRC_OBJECT_CHANGED

(2041, X'7F9') オープンされた後でオブジェクト定義が変更された。

MQRC_OBJECT_DAMAGED

(2101, X'835') オブジェクトが損傷しました。

MQRC_OPERATION_ERROR

(2488, X'9B8') API 呼び出し上の命令コードが正しくない。

MQRC_OPTIONS_ERROR

(2046, X'7FE') オプションが無効であるか、矛盾しています。

MQRC_PAGESET_ERROR

(2193, X'891') ページ・セット・データ・セットへのアクセス中にエラーが発生しました。

MQRC_Q_DELETED

(2052, X'804') キューが削除されました。

MQRC_Q_INDEX_TYPE_ERROR

(2394, X'95A') キューの索引タイプが間違っている。

MQRC_Q_MGR_NAME_ERROR

(2058, X'80A') キュー・マネージャー名が無効であるか、認識されていません。

MQRC_Q_MGR_NOT_AVAILABLE

(2059, X'80B') キュー・マネージャーを接続に使用できません。

MQRC_Q_MGR QUIESCING

(2161, X'871') キュー・マネージャーが静止しています。

MQRC_Q_MGR_STOPPING

(2162, X'872') キュー・マネージャーのシャットダウン中です。

MQRC_RESOURCE_PROBLEM

(2102, X'836') 使用できるシステム・リソースが不足しています。

MQRC_SIGNAL_OUTSTANDING

(2069, X'815') このハンドルに未解決のシグナルがある。

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') ストレージが不足しています。

MQRC_SUPPRESSED_BY_EXIT

(2109, X'83D') 出口プログラムにより呼び出しが抑止されました。

MQRC_SYNCPOINT_NOT_AVAILABLE

(2072, X'818') 同期点サポートが利用できない。

MQRC_UNEXPECTED_ERROR

(2195, X'893') 予想しないエラーが発生しました。

MQRC_UOW_ENLISTMENT_ERROR

(2354, X'932') グローバル作業単位の参加に失敗した。

MQRC_UOW_MIX_NOT_SUPPORTED

(2355, X'933') 作業単位呼び出しの混合はサポートされていない。

MQRC_UOW_NOT_AVAILABLE

(2255, X'8CF') 作業単位がキュー・マネージャーから使用不可。

MQRC_WAIT_INTERVAL_ERROR

(2090, X'82A') MQGMO での待機間隔が無効である。

MQRC_WRONG_GMO_VERSION

(2256, X'8D0') 提供された MQGMO のバージョンが違っている。

MQRC_WRONG_MD_VERSION

(2257, X'8D1') 提供された MQMD のバージョンが違っている。

これらのコードについて詳しくは、[メッセージおよび理由コード](#)を参照してください。

使用上の注意

1. コールバック・ルーチンは、呼び出すすべてのサービスからの応答を検査しなければなりません。また解決できない条件をルーチンが検出した場合は、MQCB MQOP_DEREGISTER コマンドを発行してコールバック・ルーチンに対する呼び出しが繰り返されないようにしなければなりません。
2. IBM MQ の更新を含め、XA Transaction Manager がグローバル・トランザクションを管理しているアプリケーションで非同期コンシュームを使用する場合は、さらに以下の点を考慮に入れる必要があります。

- a. **xa_open** を呼び出して **HConn** が作成された後、それを得るために MQCTL(MQOP_START) を呼び出すことは無効です。

理由は、**HConn** は XA コンテキストにアタッチされた状態になっているため、非同期コンシューム・メカニズムによって使用されている別個のスレッドではアクセスできないからです。

- b. そのシナリオで MQCTL(MQOP_START) を呼び出すと、呼び出しは理由コード MQRC_ASYNC_XA_CONFLICT (2350) で失敗します。

- c. **HConn** の MQCTL(MQOP_START_WAIT) の呼び出しは、**HConn** が作成され、**xa_open** を呼び出した後で有効です。

理由は、非同期コンシューム・メカニズムを開始するこの方式では、**HConn** のその後のコールバックはすべて、MQCTL 呼び出しが行われたスレッドで実行されるようになるからです。したがって、**HConn** とスレッドの間のリンクは失われません。

3.  z/OS では、Operation が MQOP_START の場合、以下の注意事項があります。

- 非同期コールバック・ルーチンを使用するプログラムは、z/OS UNIX System Services システム・サービス (z/OS UNIX) の使用を許可されていなければなりません。
- 非同期コールバック・ルーチンを使用する Language Environment (LE) プログラムは、LE 実行時オプション POSIX(ON) を使用しなければなりません。
- 非同期コールバック・ルーチンを使用する LE 以外のプログラムは、z/OS UNIX pthread_create インターフェース (呼び出し可能サービス BPX1PTC) を使用できません。

4.  MQCTL は IMS アダプター内ではサポートされません。

注: CICS の場合、MQOP_START はサポートされていません。代わりに、MQOP_START_WAIT 関数呼び出しを使用してください。

C 言語での呼び出し

```
MQCTL (Hconn, Operation, &ControlOpts, &CompCode, &Reason)
```

パラメーターを次のように宣言します。

```
MQHCONN  Hconn;          /* Connection handle */
MQLONG   Operation;      /* Operation being processed */
MQCTL0   ControlOpts     /* Options that control the action of MQCTL */
MQLONG   CompCode;       /* Completion code */
MQLONG   Reason;         /* Reason code qualifying CompCode */
```

COBOL での呼び出し

```
CALL 'MQCTL' USING HCONN, OPERATION, CTLOPTS, COMPCODE, REASON.
```

パラメーターを次のように宣言します。


```

** Connection handle
01 HCONN PIC S9(9) BINARY.
** Operation
01 OPERATION PIC S9(9) BINARY.
** Control Options
01 CTLOPTS.
   COPY CMQCTLOV.
** Completion code
01 COMPCODE PIC S9(9) BINARY.
** Reason code qualifying COMPCODE
01 REASON PIC S9(9) BINARY.

```

PL/I での呼び出し

```
call MQCTL(Hconn, Operation, CtlOpts, CompCode, Reason)
```

パラメーターを次のように宣言します。

```

dcl Hconn      fixed bin(31); /* Connection handle */
dcl Operation  fixed bin(31); /* Operation */
dcl CtlOpts like MQCTLO;      /* Options that control the action of MQCTL */
dcl CompCode   fixed bin(31); /* Completion code */
dcl Reason     fixed bin(31); /* Reason code qualifying CompCode */

```

MQDISC - キュー・マネージャーの切断

MQDISC 呼び出しはキュー・マネージャーとアプリケーション・プログラムとの接続を切断します。MQCONN および MQCONNX 呼び出しの逆の操作にあたります。

- z/OS では、非同期メッセージ・コンシューム、イベント処理、またはコールバックを使用するすべてのアプリケーションで、メイン制御スレッドが終了前に MQDISC 呼び出しを発行する必要があります。詳細については、[IBM MQ メッセージの非同期コンシューム](#)を参照してください。
- z/OS では、CICS アプリケーションはキュー・マネージャーからの切断のためにこの呼び出しを発行する必要はありません。

CICS アプリケーションはこの呼び出しを行いますが、以下のいずれかを指定して先に MQCONNX 呼び出しを行わなければ、何の効果もありません。

```

MQCNO_SERIALIZE_CONN_TAG_Q_MGR
MQCNO_SERIALIZE_CONN_TAG_QSG
MQCNO_RESTRICT_CONN_TAG_Q_MGR または
MQCNO_RESTRICT_CONN_TAG_QSG

```

オプション。この場合、現在開いているすべてのオブジェクト・ハンドルは閉じられます。

構文

MQDISC (*Hconn*, *CompCode*, *Reason*)

パラメーター

Hconn

タイプ: MQHCONN - 入出力

このハンドルは、キュー・マネージャーに対する接続を表します。Hconn の値は、先行の MQCONN または MQCONNX 呼び出しによって戻されたものです。

z/OS 上で CICS アプリケーションを使用する場合は、MQCONN 呼び出しを省略でき、Hconn に次の値を指定できます。

MQHC_DEF_HCONN

デフォルトの接続ハンドル。

コールが正常に完了すると、キューマネージャは環境に対して有効なハンドルではない値に *Hconn* 設定する。値は、以下のとおりです。

MQHC_UNUSABLE_HCONN

使用できない接続ハンドル。

では z/OS、*Hconn* は未定義の値に設定される。

CompCode

タイプ: MQLONG - 出力

完了コード。以下のコードのいずれかです。

MQCC_OK

正常終了。

MQCC_WARNING

警告 (部分完了)。

MQCC_FAILED

呼び出し失敗。

理由

タイプ: MQLONG - 出力

CompCode が MQCC_OK の場合:

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_WARNING の場合:

MQRC_BACKED_OUT

(2003, X'7D3') 作業単位がバックアウトされた。

MQRC_CONN_TAG_NOT_RELEASED

(2344, X'928') 接続タグが解放されていない。

MQRC_OUTCOME_PENDING

(2124, X'84C') コミット操作の結果が保留状態である。

CompCode が MQCC_FAILED の場合:

MQRC_ADAPTER_DISC_LOAD_ERROR

(2138, X'85A') アダプター切断モジュールをロードできない。

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'89C') アダプターが利用できません。

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852') アダプター・サービス・モジュールをロードできません。

MQRC_API_EXIT_ERROR

(2374, X'946') API 出口で障害が発生しました。

MQRC_API_EXIT_INIT_ERROR

(2375, X'947') API 出口の初期化に失敗しました。

MQRC_API_EXIT_TERM_ERROR

(2376, X'948') API 出口の終了に失敗しました。

MQRC_ASID_MISMATCH

(2157, X'86D') 1 次 ASID とホーム ASID が異なっています。

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') 前の呼び出しが完了する前に MQI 呼び出しが入力されました。

MQRC_CONNECTION_BROKEN

(2009, X'7D9') キュー・マネージャとの接続が失われました。

MQRC_CONNECTION_STOPPING

(2203, X'89B') 接続がシャットダウン中です。

MQRC_HCONN_ERROR

(2018, X'7E2') 接続ハンドルが無効です。

MQRC_OUTCOME_MIXED

(2123, X'84B') コミットまたはバックアウト操作の結果が混在している。

MQRC_PAGESET_ERROR

(2193, X'891') ページ・セット・データ・セットへのアクセス中にエラーが発生しました。

MQRC_Q_MGR_NAME_ERROR

(2058, X'80A') キュー・マネージャー名が無効であるか、認識されていません。

MQRC_Q_MGR_NOT_AVAILABLE

(2059, X'80B') キュー・マネージャーを接続に使用できません。

MQRC_Q_MGR_STOPPING

(2162, X'872') キュー・マネージャーのシャットダウン中です。

MQRC_RESOURCE_PROBLEM

(2102, X'836') 使用できるシステム・リソースが不足しています。

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') ストレージが不足しています。

MQRC_UNEXPECTED_ERROR

(2195, X'893') 予期しないエラーが発生しました。

これらのコードについて詳しくは、[メッセージおよび理由コード](#)を参照してください。

使用上の注意

1. 接続の下でまだオブジェクトがオープンされているときに MQDISC 呼び出しが発行されると、キュー・マネージャーはこれらのオブジェクトをクローズします。このときのクローズ・オプションは MQCO_NONE です。
2. 作業単位内にあるコミットされていない変更内容でアプリケーションが終了する場合、それらの変更内容の後処理は、そのアプリケーションの終了の仕方によって異なります。
 - a. アプリケーションが終了前に MQDISC 呼び出しを発行する場合、
 - ・ キュー・マネージャーが調整する作業単位の場合、キュー・マネージャーがアプリケーションの代わりに MQCMIT 呼び出しを出します。可能であれば作業単位がコミットされ、そうでなければバックアウトされます。
 - ・ 外部的に調整された作業単位の場合、作業単位の状態には変更がありません。しかし、キュー・マネージャーは通常、作業単位コーディネーターに求められると、作業単位がコミットされなければならないことを示します。

z/OS、CICS、IMS（バッチ DL/1 プログラム以外）、および RRS アプリケーションはこのようになります。
 - b. アプリケーションが正常に終了しても、MQDISC 呼び出しを出さない場合、環境によって次のように処置が実行されます。
 - ・ z/OS では、MQ Java および MQ JMS アプリケーションの場合を除き、注 2a に記載している処置が実行されます。
 - ・ その他の場合はすべて、2c で説明された処置が実行されます。環境によって違いが生じるため、移植するアプリケーションでは終了前に作業単位のコミットまたはバックアウトのいずれかを行うようにしてください。
 - c. アプリケーションが MQDISC 呼び出しを出さずに異常終了する場合、作業単位はバックアウトされます。
3. z/OS では、以下の点が適用されます。
 - ・ CICS アプリケーションは、キュー・マネージャーから切断するために MQDISC 呼び出しを発行する必要はありません。これは、CICS システム自体がキュー・マネージャーに接続しており、MQDISC 呼び出しはこの接続では効果がないためです。

- CICS、IMS (バッチ DL/1 プログラム以外)、および RRS アプリケーションは、外部の作業単位コーディネーターによって調整される作業単位を使用します。その結果、MQDISC 呼び出しは、呼び出しが出される時に存在する作業単位があっても、その状況には影響しません。

しかし、MQDISC 呼び出しは、アプリケーションが前に発行した MQCONNX 呼び出しによって、接続と関連付けされた接続タグ *ConnTag* の使用の終了を示します。MQDISC 呼び出しの発行時に接続タグを参照するアクティブな作業単位がある場合、呼び出しは完了し、完了コード MQCC_WARNING と理由コード MQRC_CONN_TAG_NOT_RELEASED が出されます。接続タグは、外部作業単位コーディネーターが作業単位を解決するまで、再利用のために使用可能になりません。

注: CICS の場合、MQOP_START はサポートされていません。代わりに、MQOP_START_WAIT 関数呼び出しを使用してください。

C 言語での呼び出し

```
MQDISC (&Hconn, &CompCode, &Reason);
```

パラメーターを次のように宣言します。

```
MQHCONN  Hconn;      /* Connection handle */
MQLONG   CompCode;   /* Completion code */
MQLONG   Reason;     /* Reason code qualifying CompCode */
```

COBOL での呼び出し

```
CALL 'MQDISC' USING HCONN, COMPCODE, REASON.
```

パラメーターを次のように宣言します。

```
** Connection handle
01 HCONN      PIC S9(9) BINARY.
** Completion code
01 COMPCODE   PIC S9(9) BINARY.
** Reason code qualifying COMPCODE
01 REASON     PIC S9(9) BINARY.
```

PL/I での呼び出し

```
call MQDISC (Hconn, CompCode, Reason);
```

パラメーターを次のように宣言します。

```
dcl Hconn      fixed bin(31); /* Connection handle */
dcl CompCode   fixed bin(31); /* Completion code */
dcl Reason     fixed bin(31); /* Reason code qualifying CompCode */
```

System/390 アセンブラー呼び出し

```
CALL MQDISC,(HCONN,COMPCODE,REASON)
```

パラメーターを次のように宣言します。

```
HCONN      DS  F  Connection handle
COMPCODE   DS  F  Completion code
REASON     DS  F  Reason code qualifying COMPCODE
```

Visual Basic での呼び出し

```
MQDISC Hconn, CompCode, Reason
```

パラメーターを次のように宣言します。

```
Dim Hconn As Long 'Connection handle'
Dim CompCode As Long 'Completion code'
Dim Reason As Long 'Reason code qualifying CompCode'
```

MQDLTMH - メッセージ・ハンドルの削除

MQDLTMH 呼び出しは、メッセージ・ハンドルの削除するので、MQCRTMH 呼び出しの逆です。

構文

MQDLTMH (*Hconn*, *Hmsg*, *DltMsgHOpts*, *CompCode*, *Reason*)

パラメーター

Hconn

タイプ: MQHCONN - 入力

このハンドルは、キュー・マネージャーに対する接続を表します。

値は、**Hmsg** パラメーターで指定されているメッセージ・ハンドルの作成に使用された接続ハンドルと一致していなければなりません。

MQHC_UNASSOCIATED_HCONN を使用してメッセージ・ハンドルが作成された場合は、メッセージ・ハンドルの削除するスレッド上で有効な接続を確立しなければなりません。確立しないと、呼び出しは MQRC_CONNECTION_BROKEN で失敗します。

Hmsg

タイプ: MQHMSG - 入出力

これは削除されるメッセージ・ハンドルです。値は、前の MQCRTMH 呼び出しで戻されたものです。

呼び出しが正常に完了すると、ハンドルは環境に対して無効な値に設定されます。値は、以下のとおりです。

MQHM_UNUSABLE_HMSG

使用できないメッセージ・ハンドル。

同じメッセージ・ハンドルの渡した別の IBM MQ 呼び出しが進行中の場合は、メッセージ・ハンドルの削除できません。

DltMsgHOpts

タイプ: MQDMHO - 入力

詳細については、[MQDMHO](#) を参照してください。

CompCode

タイプ: MQLONG - 出力

完了コード。以下のいずれかです。

MQCC_OK

正常終了。

MQCC_FAILED

呼び出し失敗。

理由

タイプ: MQLONG - 出力

CompCode が MQCC_OK の場合:

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_FAILED の場合、次のようになります。

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'089C') アダプターが利用できません。

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852') アダプター・サービス・モジュールをロードできません。

MQRC_ASID_MISMATCH

(2157, X'86D') 1 次 ASID とホーム ASID が異なっています。

MQRC_CALL_IN_PROGRESS

(2219, X'08AB') 前の呼び出しが完了する前に MQI 呼び出しが入力された。

MQRC_CONNECTION_BROKEN

(2009, X'07D9') キュー・マネージャーとの接続が失われました。

MQRC_DMHO_ERROR

(2462, X'099E') メッセージ・ハンドル削除オプションの構造が無効である。

MQRC_HMSG_ERROR

(2460, X'099C') メッセージ・ハンドル・ポインターが無効。

MQRC_MSG_HANDLE_IN_USE

(2499, X'09C3') メッセージ・ハンドルがすでに使用中。

MQRC_OPTIONS_ERROR

(2046, X'07FE') オプションが無効であるか、矛盾しています。

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') ストレージが不足しています。

MQRC_UNEXPECTED_ERROR

(2195, X'893') 予期しないエラーが発生しました。

これらのコードについて詳しくは、[メッセージおよび理由コード](#)を参照してください。

C 言語での呼び出し

```
MQDLTMH (Hconn, &Hmsg, &DltMsgHOpts, &CompCode, &Reason);
```

パラメーターを次のように宣言します。

```
MQHCONN  Hconn;          /* Connection handle */
MQHMSG    Hmsg;           /* Message handle */
MQDMHO    DltMsgHOpts;    /* Options that control the action of MQDLTMH */
MQLONG    CompCode;       /* Completion code */
MQLONG    Reason;         /* Reason code qualifying CompCode */
```

COBOL での呼び出し

```
CALL 'MQDLTMH' USING HCONN, HMSG, DLTMSGHOPTS, COMPCODE, REASON.
```

パラメーターを次のように宣言します。

```
** Connection handle
01  HCONN    PIC S9(9) BINARY.

** Options that control the action of MQDLTMH
01  DLTMSGHOPTS.
COPY CMQDMHOL.

** Completion code
01  COMPCODE  PIC S9(9) BINARY.

** Reason code qualifying COMPCODE
01  REASON    PIC S9(9) BINARY.
```

PL/I での呼び出し

```
call MQDLTMH (Hconn, Hmsg, DltMsgH0pts, CompCode, Reason);
```

パラメーターを次のように宣言します。

```
dcl Hconn          /* Connection handle */
dcl Hmsg           /* Message handle */
dcl DltMsgH0pts like MQDMH0; /* Options that control the action of MQDLTMH */
dcl CompCode       /* Completion code */
dcl Reason         /* Reason code qualifying CompCode */
```

高水準アセンブラー呼び出し

```
CALL MQDLTMH, (HCONN,HMSG,DLTMSGHOPTS,COMPCODE,REASON)
```

パラメーターを次のように宣言します。

HCONN	DS	F	Connection handle
HMSG	DS	D	Message handle
DLTMSGHOPTS	CMQDMH0A	,	Options that control the action of MQDLTMH
COMPCODE	DS	F	Completion code
REASON	DS	F	Reason code qualifying COMPCODE

MQDLTMP - メッセージ・プロパティの削除

MQDLTMP 呼び出しは、メッセージ・ハンドルからプロパティを削除するので、MQSETTMP 呼び出しの逆です。

構文

MQDLTMP (*Hconn*, *Hmsg*, *DltPropOpts*, *Name*, *CompCode*, *Reason*)

パラメーター

Hconn

タイプ: MQHCONN - 入力

このハンドルは、キュー・マネージャーに対する接続を表します。値は、**Hmsg** パラメーターで指定されているメッセージ・ハンドルの作成に使用された接続ハンドルと一致していなければなりません。

MQHC_UNASSOCIATED_HCONN を使用してメッセージ・ハンドルが作成された場合は、メッセージ・ハンドルを削除するスレッド上で有効な接続を確立しなければなりません。確立しないと、呼び出しは MQRC_CONNECTION_BROKEN で失敗します。

Hmsg

タイプ: MQHMSG - 入力

これは、削除されるプロパティを含むメッセージ・ハンドルです。値は、前の MQCRTMH 呼び出しで戻されたものです。

DltPropOpts

タイプ: MQDMPO - 入力

詳細については、[MQDMPO](#) データ・タイプを参照してください。

名前

タイプ: MQCHARV - 入力

削除するプロパティの名前。プロパティ名の詳細については、[プロパティ名](#)を参照してください。

プロパティ名にワイルドカードを使用することはできません。

CompCode

タイプ: MQLONG - 出力

完了コード。以下のいずれかです。

MQCC_OK

正常終了。

MQCC_WARNING

警告 (部分完了)。

MQCC_FAILED

呼び出し失敗。

理由

タイプ: MQLONG - 出力

CompCode が MQCC_OK の場合:

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_WARNING の場合:

MQRC_PROPERTY_NOT_AVAILABLE

(2471, X'09A7') プロパティが使用できない。

MQRC_RFH_FORMAT_ERROR

(2421, X'0975') プロパティを含む MQRFH2 フォルダーを構文解析できなかった。

CompCode が MQCC_FAILED の場合:

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'089C') アダプターが利用できません。

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'0852') アダプター・サービス・モジュールをロードできない。

MQRC_ASID_MISMATCH

(2157, X'086D') 1 次 ASID とホーム ASID とが異なっている。

MQRC_CALL_IN_PROGRESS

(2219, X'08AB') 前の呼び出しが完了する前に MQI 呼び出しが入力された。

MQRC_CONNECTION_BROKEN

(2009, X'07D9') キュー・マネージャーとの接続が失われました。

MQRC_DMPO_ERROR

(2481, X'09B1') メッセージ・プロパティ削除のオプション構造体が無効です。

MQRC_HMSG_ERROR

(2460, X'099C') メッセージ・ハンドルが無効。

MQRC_MSG_HANDLE_IN_USE

(2499, X'09C3') メッセージ・ハンドルがすでに使用中。

MQRC_OPTIONS_ERROR

(2046, X'07FE') オプションが無効であるか、矛盾しています。

MQRC_PROPERTY_NAME_ERROR

(2442, X'098A') プロパティ名が無効である。

MQRC_SOURCE_CCSID_ERROR

(2111, X'083F') プロパティ名エンコード文字セット ID が無効である。

MQRC_UNEXPECTED_ERROR

(2195, X'0893') 予期しないエラーが発生した。

これらのコードについて詳しくは、[メッセージおよび理由コード](#)を参照してください。

C 言語での呼び出し

```
MQDLTMP (Hconn, Hmsg, &DltPropOpts, &Name, &CompCode, &Reason)
```

パラメーターを次のように宣言します。

```
MQHCONN Hconn;          /* Connection handle */
MQHMSG Hmsg;             /* Message handle */
MQDMPO DltPropOpts;      /* Options that control the action of MQDLTMP */
MQCHARV Name;           /* Property name */
MQLONG CompCode;        /* Completion code */
MQLONG Reason;          /* Reason code qualifying CompCode */
```

COBOL での呼び出し

```
CALL 'MQDLTMP' USING HCONN, HMSG, DLTPROPOPTS, NAME, COMPCODE, REASON.
```

パラメーターを次のように宣言します。

```
** Connection handle
01 HCONN PIC S9(9) BINARY.
** Message handle
01 HMSG PIC S9(18) BINARY.
** Options that control the action of MQDLTMP
01 DLTPROPOPTS.
COPY CMQDMPOV.
** Property name
01 NAME.
COPY CMQCHRVV.
** Completion code
01 COMPCODE PIC S9(9) BINARY.
** Reason code qualifying COMPCODE
01 REASON PIC S9(9) BINARY.
```

PL/I での呼び出し

```
call MQDLTMP (Hconn, Hmsg, DltPropOpts, Name, CompCode, Reason);
```

パラメーターを次のように宣言します。

```
dcl Hconn      fixed bin(31); /* Connection handle */
dcl Hmsg       fixed bin(63); /* Message handle */
dcl DltPropOpts like MQDMPO;  /* Options that control the action of MQDLTMP */
dcl Name       like MQCHARV;  /* Property name */
dcl CompCode   fixed bin(31); /* Completion code */
dcl Reason     fixed bin(31); /* Reason code qualifying CompCode */
```

高水準アセンブラー呼び出し

```
CALL MQDLTMP, (HCONN, HMSG, DLTPROPOPTS, NAME, COMPCODE, REASON)
```

パラメーターを次のように宣言します。

HCONN	DS	F	Connection handle
HMSG	DS	D	Message handle
DLTPROPOPTS	CMQDMPOA	,	Options that control the action of MQDLTMP
NAME	CMQCHRVA	,	Property name
COMPCODE	DS	F	Completion code
REASON	DS	F	Reason code qualifying COMPCODE

MQGET - メッセージの読み取り

MQGET 呼び出しは、MQOPEN 呼び出しを使用してオープンされたローカル・キューからメッセージを取り出します。

構文

MQGET (*Hconn*, *Hobj*, *MsgDesc*, *GetMsgOpts*, *BufferLength*, *Buffer*, *DataLength*, *CompCode*, *Reason*)

パラメーター

Hconn

タイプ: MQHCONN - 入力

このハンドルは、キュー・マネージャーに対する接続を表します。 *Hconn* の値は、先行の MQCONN または MQCONNX 呼び出しによって戻されたものです。

z/OS 上で CICS アプリケーションを使用する場合は、MQCONN 呼び出しを省略でき、*Hconn* に次の値を指定できます。

MQHC_DEF_HCONN

デフォルトの接続ハンドル。

Hobj

タイプ: MQHOBJ - 入力

このハンドルは、メッセージが取り出されるキューを表します。 *Hobj* の値は、前の MQOPEN 呼び出しで戻されたものです。このキューは、次のオプションを 1 つ以上指定してオープンしておく必要があります (詳しくは、[724 ページの『MQOPEN - オブジェクトのオープン』](#)を参照してください)。

- MQOO_INPUT_SHARED
- MQOO_INPUT_EXCLUSIVE
- MQOO_INPUT_AS_Q_DEF
- MQOO_BROWSE

MsgDesc

タイプ: MQMD - 入出力

この構造体は、必要なメッセージの属性と、取り出されるメッセージの属性を記述します。詳細は [385 ページの『MQMD - メッセージ記述子』](#)を参照してください。

メッセージ長が未満 *BufferLength* の場合、*MsgDesc* はキュー・マネージャによって埋められます。これは、パラメータに **GetMsgOpts** MQGMO_ACCEPT_TRUNCATED_MSG が指定されているかどうかにかかわらず発生します ([MQGMO - オプション・フィールド](#)を参照)。

アプリケーションがバージョン 1 の MQMD を提供している場合、戻されるメッセージでは、アプリケーション・メッセージ・データに MQMDE の接頭部が付いていますが、これは MQMDE 内の 1 つ以上のフィールドがデフォルト以外の値を持つ場合のみです。MQMDE 内のすべてのフィールドがデフォルト値を持つ場合、この MQMDE は省略されます。MQMD 内の *Format* フィールドにある MQFMT_MD_EXTENSION という形式名があれば、MQMDE が存在することを意味します。

有効なメッセージ・ハンドルが *MsgHandle* フィールドで提供されている場合、アプリケーションは MQMD 構造を提供する必要はありません。このフィールドに何も提供されていない場合、メッセージの記述子は、メッセージ・ハンドルに関連した記述子から取られます。

アプリケーションが MQMD 構造体ではなくメッセージ・ハンドルを提供し、MQGMO_PROPERTIES_FORCE_MQRFH2 を指定すると、呼び出しは失敗して理由コード MQRC_MD_ERROR が表示されます。アプリケーションが MQMD 構造体を提供しないで MQGMO_PROPERTIES_AS_Q_DEF を指定し、**PropertyControl** キュー属性が MQPROP_FORCE_MQRFH2 である場合も、呼び出しはやはり失敗し、理由コード MQRC_MD_ERROR が表示されます。

一致オプションが指定されており、メッセージ・ハンドルに関連したメッセージ記述子を使用されている場合、一致させるために使用される入力フィールドはメッセージ処理に由来します。

GetMsgOpts

タイプ: MQGMO - 入出力

詳細は [326 ページの『MQGMO - 読み取りメッセージ・オプション』](#)を参照してください。

BufferLength

タイプ: MQLONG - 入力

これは、*Buffer* 域の長さ (バイト数) です。メッセージにデータがない場合、またはメッセージがキューから削除され、データが廃棄される (この場合には MQGMO_ACCEPT_TRUNCATED_MSG の指定が必要) 場合には、ゼロを指定します。

注: キューから読み取り可能な最長メッセージの長さは、キュー属性 **MaxMsgLength** によって示されます (830 ページの『キューの属性』を参照)。

Buffer

タイプ: MQBYTEExBufferLength - 出力

これはメッセージ・データが入れられる領域です。バッファを、メッセージのデータの性質に適した境界に位置合わせします。IBM MQ ヘッダー構造になっているメッセージを含む、ほとんどのメッセージには 4 バイトの位置合わせが適していますが、メッセージによってはより厳しい位置合わせを必要とする場合があります。例えば、64 ビット・バイナリー整数を含むメッセージは 8 バイト境界に合わせる必要がある場合があります。

BufferLength がメッセージの長より短い場合は、可能な限りメッセージが **Buffer** に移動されます。この処理は、**GetMsgOpts** パラメータで MQGMO_ACCEPT_TRUNCATED_MSG が指定されているかどうかにかかわらず行われます (詳細については、[MQGMO - Options フィールド](#)を参照してください)。

データ内の文字セットとエンコーディングは、**MsgDesc** パラメータで返される '**Buffer** charset' および '**CodedCharSetIdEncoding** encoding' フィールドによって指定されます。これらの値が受信側で必要とされている値と異なる場合、受信側はアプリケーション・メッセージ・データを必要な文字セットとエンコードに変換する必要があります。MQGMO_CONVERT オプションを (必要ならユーザー作成の出口で) 使用して、メッセージ・データを変換することができます。このオプションの詳細については、[326 ページの『MQGMO - 読み取りメッセージ・オプション』](#)を参照してください。

注: MQGET 呼び出しのその他のパラメータはすべて、ローカル・キュー・マネージャーの文字セットとエンコードに従っています (**CodedCharSetId** キュー・マネージャー属性と MQENC_NATIVE で指定します)。

呼び出しが失敗した場合は、バッファの内容が変更されてしまっていることもあります。

C プログラミング言語では、パラメータは、void を示すポインターとして宣言されます。つまり、どのタイプのデータのアドレスもパラメータとして指定できます。

BufferLength パラメータがゼロの場合は、*Buffer* は参照されません。この場合、C または System/390 アセンブラで作成されたプログラムによって渡されるパラメータ・アドレスはヌルのこともあります。

DataLength

タイプ: MQLONG - 出力

これは、メッセージ内のアプリケーション・データの長さ (バイト数) です。この値が *BufferLength* より長い場合は、*BufferLength* バイトだけが **Buffer** パラメータに戻されます (つまり、メッセージが切り捨てられます)。この値がゼロの場合は、メッセージにはアプリケーション・データが入っていません。

メッセージ長が未満 *BufferLength* の場合でも、MQGMO_ACCEPT_TRUNCATED_MSG が **GetMsgOpts** パラメータで指定 *DataLength* されているかどうかに関わらず、キュー・マネージャによってメッセージは完了処理されます (詳細は [MQGMO - オプション・フィールド](#)を参照)。これにより、アプリケーションは、メッセージ・データを収容するのに必要なバッファのサイズを判別して、適切なサイズのバッファを用いて呼び出しを再発行することができます。

しかし、MQGMO_CONVERT オプションが指定されている場合に、変換されたメッセージ・データが長すぎて *Buffer* に入りきらないと、*DataLength* について、以下の値が戻されます。

- 未変換 データの長さ (キュー・マネージャー定義の形式の場合)。

この場合、データの性質により変換中に拡張が行われるときは、アプリケーションは *DataLength* としてキュー・マネージャーから戻される値より大きいバッファを割り振る必要があります。

- ・データ変換出口により戻される値 (アプリケーション定義の形式の場合)

CompCode

タイプ: MQLONG - 出力

完了コード。以下のいずれかです。

MQCC_OK

正常終了。

MQCC_WARNING

警告 (部分完了)。

MQCC_FAILED

呼び出し失敗。

理由

タイプ: MQLONG - 出力

次に示す理由コードは、キュー・マネージャーが **Reason** パラメーターに対して返すことのある理由コードです。アプリケーションが MQGMO_CONVERT オプションを指定し、ユーザー作成出口を起動してメッセージ・データの一部またはすべてを変換する場合、**Reason** パラメーターに戻される値はその出口が決定します。このため、次に示す値以外の値が戻ることがあります。

CompCode が MQCC_OK の場合:

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_WARNING の場合:

MQRC_CONVERTED_MSG_TOO_BIG

(2120, X'848') 変換されたデータが、バッファーには大きすぎる。

MQRC_CONVERTED_STRING_TOO_BIG

(2190, X'88E') 変換されたストリングが、フィールドには大きすぎる。

MQRC_DBCS_ERROR

(2150, X'866') DBCS ストリングが無効である。

MQRC_FORMAT_ERROR

(2110, X'83E') メッセージ形式が無効である。

MQRC_INCOMPLETE_GROUP

(2241, X'8C1') メッセージ・グループが不完全である。

MQRC_INCOMPLETE_MSG

(2242, X'8C2') 論理メッセージが不完全である。

MQRC_INCONSISTENT_CCIDS

(2243, X'8C3') 各メッセージ・セグメントが異なる CCSID をもつ。

MQRC_INCONSISTENT_ENCODINGS

(2244, X'8C4') 各メッセージ・セグメントが異なるエンコードをもつ。

MQRC_INCONSISTENT_UOW

(2245, X'8C5') 作業単位の指定が不整合である。

MQRC_MSG_TOKEN_ERROR

(2331, X'91B') メッセージ・トークンの無効な使用。

MQRC_NO_MSG_LOCKED

(2209, X'8A1') ロックされているメッセージがない。

MQRC_NOT_CONVERTED

(2119, X'847') メッセージ・データが変換されなかった。

MQRC_OPTIONS_CHANGED

(nnnn, X'xxx') 整合性を保つために必要なオプションが変更されている。

MQRC_PARTIALLY_CONVERTED

(2272, X'8E0') メッセージ・データが一部変換されなかった。

MQRC_SIGNAL_REQUEST_ACCEPTED

(2070, X'816') メッセージは戻されなかった (ただし、シグナル要求は受け入れられた)。

MQRC_SOURCE_BUFFER_ERROR

(2145, X'861') ソース・バッファー・パラメーターが無効。

MQRC_SOURCE_CCSDID_ERROR

(2111, X'83F') ソース・エンコード文字セット ID が無効である。

MQRC_SOURCE_DECIMAL_ENC_ERROR

(2113, X'841') メッセージ内のパック 10 進数のエンコードが認識できない。

MQRC_SOURCE_FLOAT_ENC_ERROR

(2114, X'842') メッセージ内の浮動小数点のエンコードが認識できない。

MQRC_SOURCE_INTEGER_ENC_ERROR

(2112, X'840') ソース整数エンコードが認識できない。

MQRC_SOURCE_LENGTH_ERROR

(2143, X'85F') ソース長パラメーターが無効である。

MQRC_TARGET_BUFFER_ERROR

(2146, X'862') ターゲット・バッファー・パラメーターが無効である。

MQRC_TARGET_CCSDID_ERROR

(2115, X'843') ターゲット・エンコード文字セット ID が無効である。

MQRC_TARGET_DECIMAL_ENC_ERROR

(2117, X'845') 受信側で指定されたパック 10 進数のエンコードが認識できない。

MQRC_TARGET_FLOAT_ENC_ERROR

(2118, X'846') 受信側で指定された浮動小数点のエンコードが認識できない。

MQRC_TARGET_INTEGER_ENC_ERROR

(2116, X'844') ターゲット整数エンコードが認識できない。

MQRC_TRUNCATED_MSG_ACCEPTED

(2079, X'81F') 切り捨てられたメッセージが戻された (処理は完了している)。

MQRC_TRUNCATED_MSG_FAILED

(2080, X'820') 切り捨てられたメッセージが戻された (処理は完了していない)。

CompCode が MQCC_FAILED の場合:

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'89C') アダプターが利用できません。

MQRC_ADAPTER_CONV_LOAD_ERROR

(2133, X'855') データ変換サービス・モジュールをロードできない。

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852') アダプター・サービス・モジュールをロードできません。

MQRC_API_EXIT_ERROR

(2374, X'946') API 出口で障害が発生しました。

MQRC_API_EXIT_LOAD_ERROR

(2183, X'887') API 出口をロードできません。

MQRC_ASID_MISMATCH

(2157, X'86D') 1 次 ASID とホーム ASID が異なっています。

MQRC_BACKED_OUT

(2003, X'7D3') 作業単位がバックアウトされた。

MQRC_BUFFER_ERROR

(2004, X'7D4') バッファー・パラメーターが無効である。

MQRC_BUFFER_LENGTH_ERROR

(2005, X'7D5') バッファー長パラメーターは無効です。

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') 前の呼び出しが完了する前に MQI 呼び出しが入力されました。

MQRC_CF_NOT_AVAILABLE

(2345, X'929') カップリング・ファシリティーが使用できません。

MQRC_CF_STRUC_FAILED

(2373, X'945') カップリング・ファシリティー構造体で障害が発生しました。

MQRC_CF_STRUC_IN_USE

(2346, X'92A') カップリング・ファシリティー構造体が使用中です。

MQRC_CF_STRUC_LIST_HDR_IN_USE

(2347, X'92B') カップリング・ファシリティー構造体のリスト・ヘッダーが使用中です。

MQRC_CICS_WAIT_FAILED

(2140, X'85C') 待機要求が CICS により拒否された。

MQRC_CONNECTION_BROKEN

(2009, X'7D9') キュー・マネージャーとの接続が失われました。

MQRC_CONNECTION_NOT_AUTHORIZED

(2217, X'8A9') 接続が許可されていません。

MQRC_CONNECTION_QUIESCING

(2202, X'89A') 接続が静止しています。

MQRC_CONNECTION_STOPPING

(2203, X'89B') 接続がシャットダウン中です。

MQRC_CORREL_ID_ERROR

(2207, X'89F') 相関 ID のエラー。

MQRC_DATA_LENGTH_ERROR

(2010, X'7DA') データ長パラメーターが無効である。

MQRC_DB2_NOT_AVAILABLE

(2342, X'926') Db2 サブシステムが利用できません。

MQRC_GET_INHIBITED

(2016, X'7E0') キューからの読み取りが禁止されている。

MQRC_GLOBAL_UOW_CONFLICT

(2351, X'92F') グローバル作業単位に矛盾がある。

MQRC_GMO_ERROR

(2186, X'88A') 読み取りメッセージ・オプションの構造体が無効である。

MQRC_HANDLE_IN_USE_FOR_UOW

(2353, X'931') グローバル作業単位のためのハンドルが使用中。

MQRC_HCONN_ERROR

(2018, X'7E2') 接続ハンドルが無効です。

MQRC_HOBJ_ERROR

(2019, X'7E3') オブジェクト・ハンドルが無効です。

MQRC_INCONSISTENT_BROWSE

(2259, X'8D3') ブラウズの指定が不整合である。

MQRC_INCONSISTENT_UOW

(2245, X'8C5') 作業単位の指定が不整合である。

MQRC_INVALID_MSG_UNDER_CURSOR

(2246, X'8C6') カーソル下のメッセージが取り出し対象として無効である。

MQRC_LOCAL_UOW_CONFLICT

(2352, X'930') グローバル作業単位とローカル作業単位に矛盾がある。

MQRC_MATCH_OPTIONS_ERROR

(2247, X'8C7') 突き合わせオプションが無効である。

MQRC_MD_ERROR

(2026, X'7EA') メッセージ記述子が無効である。

MQRC_MSG_ID_ERROR

(2206, X'89E') メッセージ ID のエラー。

MQRC_MSG_SEQ_NUMBER_ERROR

(2250, X'8CA') メッセージ順序番号が無効である。

MQRC_MSG_TOKEN_ERROR

(2331, X'91B') メッセージ・トークンについて無効な使い方をしている。

MQRC_NO_MSG_AVAILABLE

(2033, X'7F1') メッセージが使用できない。

MQRC_NO_MSG_UNDER_CURSOR

(2034, X'7F2') ブラウズ・カーソルがメッセージに位置付けされていない。

MQRC_NOT_OPEN_FOR_BROWSE

(2036, X'7F4') ブラウズのためにキューがオープンされていない。

MQRC_NOT_OPEN_FOR_INPUT

(2037, X'7F5') 入力のためにキューがオープンされていない。

MQRC_OBJECT_CHANGED

(2041, X'7F9') オープンされた後でオブジェクト定義が変更された。

MQRC_OBJECT_DAMAGED

(2101, X'835') オブジェクトが損傷しました。

MQRC_OPTIONS_ERROR

(2046, X'7FE') オプションが無効であるか、矛盾しています。

MQRC_PAGESET_ERROR

(2193, X'891') ページ・セット・データ・セットへのアクセス中にエラーが発生しました。

MQRC_Q_DELETED

(2052, X'804') キューが削除されました。

MQRC_Q_INDEX_TYPE_ERROR

(2394, X'95A') キューの索引タイプが間違っている。

MQRC_Q_MGR_NAME_ERROR

(2058, X'80A') キュー・マネージャー名が無効であるか、認識されていません。

MQRC_Q_MGR_NOT_AVAILABLE

(2059, X'80B') キュー・マネージャーを接続に使用できません。

MQRC_Q_MGR QUIESCING

(2161, X'871') キュー・マネージャーが静止しています。

MQRC_Q_MGR_STOPPING

(2162, X'872') キュー・マネージャーのシャットダウン中です。

MQRC_RESOURCE_PROBLEM

(2102, X'836') 使用できるシステム・リソースが不足しています。

MQRC_SECOND_MARK_NOT_ALLOWED

(2062, X'80E') メッセージはすでにマークされている。

MQRC_SIGNAL_OUTSTANDING

(2069, X'815') このハンドルに未解決のシグナルがある。

MQRC_SIGNAL1_ERROR

(2099, X'833') シグナル・フィールドが無効である。

MQRC_STORAGE_MEDIUM_FULL

(2192, X'890') 外部ストレージ・メディアが満杯です。

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') ストレージが不足しています。

MQRC_SUPPRESSED_BY_EXIT

(2109, X'83D') 出口プログラムにより呼び出しが抑止されました。

MQRC_SYNCPOINT_LIMIT_REACHED

(2024, X'7E8') 現行の作業単位内では、これ以上メッセージを処理できない。

MQRC_SYNCPOINT_NOT_AVAILABLE

(2072, X'818') 同期点サポートが利用できない。

MQRC_UNEXPECTED_ERROR

(2195, X'893') 予期しないエラーが発生しました。

MQRC_UOW_ENLISTMENT_ERROR

(2354, X'932') グローバル作業単位の参加に失敗した。

MQRC_UOW_MIX_NOT_SUPPORTED

(2355, X'933') 作業単位呼び出しの混合はサポートされていない。

MQRC_UOW_NOT_AVAILABLE

(2255, X'8CF') 作業単位がキュー・マネージャーから使用不可。

MQRC_WAIT_INTERVAL_ERROR

(2090, X'82A') MQGMO での待機間隔が無効である。

MQRC_WRONG_GMO_VERSION

(2256, X'8D0') 提供された MQGMO のバージョンが違っている。

MQRC_WRONG_MD_VERSION

(2257, X'8D1') 提供された MQMD のバージョンが違っている。

これらのコードについて詳しくは、[メッセージおよび理由コード](#)を参照してください。

使用上の注意

1. 取り出されたメッセージは、通常、キューから削除されます。削除は、MQGET 呼び出し自体の一部、または同期点の一部として行われる可能性があります。

ブラウズ・オプションとは、MQGMO_BROWSE_FIRST、MQGMO_BROWSE_NEXT、および MQGMO_BROWSE_MSG_UNDER_CURSOR のことです。

2. いずれかのブラウズ・オプションで MQGMO_LOCK オプションが指定されている場合は、このハンドルにだけ扱われるよう、ブラウズされたメッセージがロックされます。

MQGMO_UNLOCK オプションが指定されていると、以前にロックされたメッセージがロック解除されます。この場合、メッセージは取り出されず、**MsgDesc**、**BufferLength**、**Buffer**、および **DataLength** パラメーターは、検査も変更も行われません。

3. MQGET 呼び出しを発行したアプリケーションについては、呼び出しの処理中にアプリケーションが異常終了するか接続が切断された場合に、取り出されたメッセージが失われることがあります。この問題が発生するのは、キュー・マネージャーと同じプラットフォームで実行されていて、アプリケーションに代わって MQGET 呼び出しを発行する代理がアプリケーションの切断を検出できる時期が、代理からアプリケーションにメッセージを返そうとする直前、つまりメッセージがキューから除去された後になるからです。この問題は、永続メッセージの場合でも、非永続メッセージの場合でも起こります。

このようにしてメッセージを失う危険を回避するには、メッセージを常に作業単位内で取り出すようにしてください。つまり、MQGET 呼び出しで MQGMO_SYNCPOINT オプションを指定し、メッセージ処理が完了した時点で MQCMIT または MQBACK 呼び出しを使用して作業単位をコミットするかバックアウトします。MQGMO_SYNCPOINT が指定されている場合、クライアントが異常終了したり接続が切断されると、代理はキュー・マネージャー上の作業単位をバックアウトして、メッセージはキューに再び入れられます。同期点の詳細については、[IBM MQ アプリケーションでの同期点に関する考慮事項](#)を参照してください。

この状態は、IBM MQ クライアントだけでなく、キュー・マネージャーと同じプラットフォーム上で実行されているアプリケーションで発生することがあります。

4. もしもアプリケーションがメッセージのシーケンスを特定の 1 つの作業単位内のキューを処理し、その作業単位を正常にコミットすると、メッセージは次のように検索できるようになります。
 - キューが非共有キュー（つまりローカル・キュー）である場合は、作業単位のメッセージはすべて同時に利用できます。
 - キューが共有キューである場合、作業単位のメッセージは、書き込まれた順番で利用できます。すべてを同時に利用することはできません。システムが重い負荷を負っている場合、最初に検索する作業単位のメッセージは成功しても、2 番目またはそれ以降の作業単位のメッセージの MQGET 呼び

出しは失敗し、MQRC_NO_MSG_AVAILABLE が戻されることがあります。この場合、アプリケーションでは、少し待機してから操作を再試行してください。

- アプリケーションがメッセージ・グループを使用せずにメッセージ・シーケンスを同じキューに書き込んだ場合、特定の条件が満たされていれば、それらのメッセージの順序は保持されます。詳細については、『MQPUT の使用上の注意』を参照してください。条件が満たされていて、さらに以下の条件も満たされていれば、メッセージは送信された順序で受信側のアプリケーションに提示されます。

- キューからメッセージを読み取る受信側は 1 つだけである。

キューからメッセージを読み取るアプリケーションが 2 つ以上ある場合は、シーケンスに属するメッセージを識別するために使用するメカニズムについて、アプリケーションが送信側と合意している必要があります。例えば、送信側は、シーケンス内メッセージの CorrelId フィールドのすべてを、そのメッセージ・シーケンスに固有な値に設定することができます。

- 受信者は、特定の MsgId や を指定するなど CorrelId して、意図的に取得順序を変更することはありません。

送信側のアプリケーションがメッセージ・グループとしてメッセージを書き込む場合、受信側のアプリケーションが MQGET 呼び出しに MQGMO_LOGICAL_ORDER オプションを指定していれば、メッセージは受信側のアプリケーションに正しい順序で提示されます。メッセージ・グループの詳細については、以下を参照してください。

- [MQMD - MsgFlags フィールド](#)
- [MQPMO_LOGICAL_ORDER](#)
- [MQGMO_LOGICAL_ORDER](#)

ユーザーが同期点下でグループ中のメッセージを取得している場合、トランザクションを終了しようとする前に、グループ全体を処理したことを確認する必要があります。

- アプリケーションは、パラメータ **MsgDesc** のフィールド Feedback においてフィードバックコード MQFB_QUIT を検査し、この値が見つかった場合は終了しなければならない。詳しくは、『MQMD - Feedback フィールド』を参照してください。
- Hobj によって識別されたキューが MQOO_SAVE_ALL_CONTEXT オプション付きでオープンされ、MQGET 呼び出しからの完了コードが MQCC_OK または MQCC_WARNING である場合は、キュー・ハンドル Hobj と関連付けられたコンテキストが、取り出されたメッセージのコンテキストに設定されます (ただし、MQGMO_BROWSE_FIRST、MQGMO_BROWSE_NEXT、または MQGMO_BROWSE_MSG_UNDER_CURSOR オプションが設定されていて、コンテキストに使用不可のマークが付いている場合を除く)。

保管されたコンテキストは、MQPMO_PASS_IDENTITY_CONTEXT または MQPMO_PASS_ALL_CONTEXT オプションを指定することにより、後続の MQPUT または MQPUT1 呼び出しで 사용할 ことができます。これにより、受信したメッセージのコンテキストの全体または一部を別のメッセージに転送することが可能になります (例えば、メッセージを別のキューに転送する場合など)。メッセージのコンテキストの詳細については、[メッセージのコンテキスト](#)を参照してください。

- MQGMO_CONVERT オプションが **GetMsgOpts** パラメーターに指定されている場合は、アプリケーション・メッセージ・データは、受信側アプリケーションで要求される表現に変換されてから **Buffer** パラメーターに入ります。

- メッセージの制御情報内のフィールド Format はアプリケーションデータの構造を識別し、メッセージの制御情報内のフィールド EncodingCodedCharSetId とフィールドはその文字セット識別子とエンコーディングを指定する。
- MQGET 呼び出しを発行するアプリケーションは、パラメータ **MsgDesc** の `charset` および `encoding` Encoding `CodedCharSetId` フィールドにおいて、アプリケーションメッセージデータを変換する対象の文字セット識別子とエンコーディングを指定する。

メッセージ・データの変換が必要な場合は、メッセージの中の制御情報にある Format フィールドの値に応じて、キュー・マネージャー自体またはユーザー作成の出口のどちらかで、変換が実行されます。

- 以下に示す形式名は、キュー・マネージャーによって変換される形式です。これらは、「組み込み」形式と呼ばれています。

- MQFMT_ADMIN
- MQFMT_CICS (z/OS のみ)
- MQFMT_COMMAND_1
- MQFMT_COMMAND_2
- MQFMT_DEAD_LETTER_HEADER
- MQFMT_DIST_HEADER
- MQFMT_EVENT バージョン 1
- MQFMT_EVENT バージョン 2 (z/OS のみ)
- MQFMT_IMS
- MQFMT_IMS_VAR_STRING
- MQFMT_MD_EXTENSION
- MQFMT_PCF
- MQFMT_REF_MSG_HEADER
- MQFMT_RF_HEADER
- MQFMT_RF_HEADER_2
- MQFMT_STRING
- MQFMT_TRIGGER
- MQFMT_WORK_INFO_HEADER (z/OS のみ)
- MQFMT_XMIT_Q_HEADER

- 形式名 MQFMT_NONE は、メッセージ内のデータの性質が未定義であることを示す特殊な値です。その結果、キュー・マネージャーは、メッセージがキューから取り出されるときには、変換を行いません。

注: MQFMT_NONE のフォーマット名を持つメッセージに対して MQGET 呼び出しで MQGMO_CONVERT が指定され、かつメッセージの文字セットまたはエンコーディングがパラメータ **MsgDesc** で指定されたものと異なる場合、メッセージはパラメータ **Buffer** で返されます（他のエラーがない場合）。ただし、呼び出しは完了コード MQCC_WARNING および理由コード MQRC_FORMAT_ERROR で終了します。

そのメッセージ・データの性質から変換が必要とされない場合、または送信側および受信側アプリケーションがそのメッセージ・データの送信形式について合意した場合には、MQFMT_NONE が使用できます。

- それ以外の形式名を使用すると、メッセージは、ユーザー作成出口に渡され、変換が行われます。その出口は、環境固有の追加とは別に、形式と同一の名前を持ちます。ユーザー指定の形式名に、「IBM MQ」という文字で始まる名前は使用しないでください。

データ変換出口の詳細については、[903 ページの『データ変換出口』](#)を参照してください。

メッセージ内のユーザー・データは、サポートされているすべての文字セットおよびエンコードとの間で変換することができます。ただし、メッセージに 1 つ以上の IBM MQ ヘッダー構造体が含まれている場合、キュー名内の有効な文字について 2 バイト文字またはマルチバイト文字を備えている文字セットとの間で、そのメッセージを変換することはできません。そのような変換を試みた場合、MQRC_SOURCE_CCSD_ERROR と MQRC_TARGET_CCSD_ERROR のどちらかの理由コードが発生し、メッセージは変換されずに戻されます。UNICODE 文字セット UTF-16 は、そのような文字セットの一例です。

MQGET から戻る際の、以下の理由コードは、メッセージが正常に変換されたことを示しています。

- MQRC_NONE

次の理由コードは、メッセージが正常に変換された可能性があることを示しています。この場合、アプリケーションで、**MsgDesc** パラメーターの **CodedCharSetId** および **Encoding** フィールドを検査して、以下のコードが戻されているか検査してください。

- **MQRC_TRUNCATED_MSG_ACCEPTED**

その他の理由コードはすべて、メッセージが変換されなかったことを表します。

注: この理由コードの解釈は、ユーザー作成の出口で実行される変換にも適用されます。ただし、その出口が [903 ページの『データ変換出口』](#)に記載されている処理ガイドラインに従っている場合だけです。

9. メッセージを取得するためにオブジェクト指向インターフェースを使用する場合には、**MQGET** 呼び出しのメッセージ・データを保持するためのバッファを指定しないことも選択できます。オブジェクト指向アプリケーションで受信メッセージバッファのサイズを制限せずにメッセージを受信した場合、アプリケーションは **MQRC_CONVERTED_MSG_TOO_BIG** エラーで失敗せず、変換されたメッセージを受信します。これは、以下の環境において当てはまります。

- .NET (完全に管理されるアプリケーションを含む)
- C++
- Java (IBM MQ classes for Java)

注: すべてのクライアントにおいて、の値がゼロ **sharingConversations** の場合、かつバッファが変換済みメッセージを受信するには小さすぎる場合、変換前のメッセージが理由コード **MQRC_CONVERTED_MSG_TOO_BIG** とともに返されます。 **sharingConversations** の詳細については、[クライアント・アプリケーションでの共用会話の使用](#)を参照してください。

10. 組み込み形式では、**MQGMO_CONVERT** オプションが指定されていると、キュー・マネージャーが、メッセージの文字ストリングのデフォルト変換を実行する場合があります。デフォルト変換の場合、キュー・マネージャーによるストリング・データの変換時に、実際の文字セットに近似するインストール指定デフォルト文字セットが使用されます。その結果、**MQGET** 呼び出しは終了し、完了コード **MQCC_OK** が戻ります。**MQCC_WARNING** および理由コード **MQRC_SOURCE_CCSID_ERROR** または **MQRC_TARGET_CCSID_ERROR** が戻ることはありません。

注: 近似する文字セットを使用してストリング・データを変換すると、文字が不正確に変換される場合があります。これを回避するには、実際の文字セットとデフォルト文字セットの両方に共通する文字をストリング内に使用します。

デフォルト変換は、アプリケーション・メッセージ・データにも、また **MQMD** および **MQMDE** 構造体内の各文字フィールドにも適用されます。

- アプリケーション・メッセージ・データのデフォルト変換は、次のすべての記述が該当する場合にだけ生じます。
 - アプリケーションが **MQGMO_CONVERT** を指定している。
 - メッセージに、サポートされていない文字セットからの変換、またはその文字セットへの変換が必要なデータが含まれている。
 - キュー・マネージャーがインストールまたは再始動され、デフォルト変換が使用可能になりました。
- **MQMD** および **MQMDE** 構造体の文字フィールドのデフォルト変換は、キュー・マネージャーでデフォルト変換が使用可能になっている場合に、必要に応じて行われます。この変換は、アプリケーションで **MQGMO_CONVERT** オプションが **MQGET** 呼び出しに指定されていない場合でも実行されます。

11. Visual Basic プログラミング言語には、以下の点が適用されます。

- パラメータ **Buffer** のサイズが **BufferLength** パラメータで指定された長さを下回る場合、呼び出しは理由コード **MQRC_STORAGE_NOT_AVAILABLE** で失敗します。
- **Buffer** パラメータはタイプ **String** として宣言されます。キューから取得するデータが型でない場合は **String**、**MQGET** の代わりに **MQGETAny** 呼び出します。

MQGETAny 呼び出しは MQGET 呼び出しと同じパラメーターを使用しますが、**Buffer** パラメーターはタイプ Any として宣言されるので、どのタイプのデータでも取り出すことができます。ただし、**Buffer** を検査して、サイズが **BufferLength** バイト以上であることを確認することはできません。

12. 先読みが有効になっている場合、すべての MQGET オプションがサポートされるわけではありません。以下の表は、使用可能なオプションと MQGET 呼び出し間でそれらを変更できるかどうかを示しています。

表 584. 先読みが有効になっている場合に使用できる MQGET オプション			
	先読みが有効になっている場合に使用でき、MQGET 呼び出し間で変更できる	先読みが有効になっている場合に使用でき、MQGET 呼び出し間で変更できない ^a	先読みが有効になっている場合に使用できない MQGET オプション ^b
MQGET MD 値	MsgId ^c CorrelId ^c	Encoding CodedCharSetId	
MQGET MQGMO オプション	MQGMO_WAIT MQGMO_NO_WAIT MQGMO_FAIL_IF_QUIESCING MQGMO_BROWSE_FIRST ^d MQGMO_BROWSE_NEXT ^d MQGMO_BROWSE_MESSAGE_UNDER_CURSOR ^d	MQGMO_SYNCPOINT_IF_PERSISTENT MQGMO_NO_SYNCPOINT MQGMO_ACCEPT_TRUNCATED_MSG MQGMO_CONVERT MQGMO_LOGICAL_ORDER MQGMO_COMPLETE_MSG MQGMO_ALL_MSGS_AVAILABLE MQGMO_ALL_SEGMENTS_AVAILABLE MQGMO_MARK_BROWSE_HANDLE MQGMO_MARK_BROWSE_CO_OP MQGMO_UNMARK_BROWSE_CO_OP MQGMO_UNMARK_BROWSE_HANDLE MQGMO_UNMARKED_BROWSE_MSG MQGMO_PROPERTIES_FORCE_MQRFH2 MQGMO_NO_PROPERTIES MQGMO_PROPERTIES_IN_HANDLE MQGMO_PROPERTIES_COMPATIBILITY	MQGMO_SET_SIGNAL MQGMO_SYNCPOINT MQGMO_MARK_SKIP _BACKOUT MQGMO_MSG_UNDER_CURSOR ^d MQGMO_LOCK MQGMO_UNLOCK
MQGMO 値		MsgHandle	

- これらのオプションが MQGET 呼び出し間で変更された場合、MQRC_OPTIONS_CHANGED 理由コードが戻されます。
 - これらのオプションが最初の MQGET 呼び出しで指定されると、先読みは使用不可になります。これらのオプションを後続の MQGET 呼び出しで指定すると、理由コード MQRC_OPTIONS_ERROR が戻されます。
 - クライアント・アプリケーション側で以下の点に留意する必要があります。すなわち、MsgId および CorrelId の値が MQGET 呼び出し間で変更された場合、変更前の値によるメッセージがクライアントに送信済みの可能性があり、コンシューム (または自動的にページ) されるまでクライアントの先読みバッファ内に残るといことです。
 - 最初の MQGET 呼び出しは、先読みが有効である場合にメッセージをキューからブラウズするか取得するかを決定します。アプリケーションがブラウズと取得の組み合わせを使用しようとすると、MQRC_OPTIONS_CHANGED 理由コードが戻されます。
 - MQGMO_MSG_UNDER_CURSOR は先読みでは使用できません。先読みが有効な場合、メッセージのブラウズまたは取得が可能です。ブラウズと取得の組み合わせは指定できません。
13. コミットされていないメッセージをアプリケーションが破壊的に取得できるのは、その取得を行うのと同じローカル作業単位内でそれらのメッセージが書き込まれた場合のみです。アプリケーションは、コミットされていないメッセージを非破壊的に取得できません。
14. ブラウズ・カーソルの下のメッセージは、作業単位で取り出すことができます。コミットされていないメッセージをこの方法で取り出すことはできません。

c 言語での呼び出し

```
MQGET (Hconn, Hobj, &MsgDesc, &GetMsgOpts, BufferLength, Buffer,
      &DataLength, &CompCode, &Reason);
```

パラメーターを次のように宣言します。

```
MQHCONN  Hconn;          /* Connection handle */
MQHOBJ    Hobj;          /* Object handle */
MQMD      MsgDesc;       /* Message descriptor */
MQGMO     GetMsgOpts;    /* Options that control the action of MQGET */
MQLONG    BufferLength;   /* Length in bytes of the Buffer area */
MQBYTE    Buffer[n];      /* Area to contain the message data */
MQLONG    DataLength;    /* Length of the message */
```

```

MQLONG  CompCode;      /* Completion code */
MQLONG  Reason;        /* Reason code qualifying CompCode */

```

COBOL での呼び出し

```

CALL 'MQGET' USING HCONN, HOBJ, MSGDESC, GETMSGOPTS, BUFFERLENGTH,
BUFFER, DATALENGTH, COMPCODE, REASON.

```

パラメーターを次のように宣言します。

```

** Connection handle
01 HCONN          PIC S9(9) BINARY.
** Object handle
01 HOBJ           PIC S9(9) BINARY.
** Message descriptor
01 MSGDESC.
   COPY CMQMDV.
** Options that control the action of MQGET
01 GETMSGOPTS.
   COPY CMQGMV.
** Length in bytes of the BUFFER area
01 BUFFERLENGTH  PIC S9(9) BINARY.
** Area to contain the message data
01 BUFFER        PIC X(n).
** Length of the message
01 DATALENGTH   PIC S9(9) BINARY.
** Completion code
01 COMPCODE      PIC S9(9) BINARY.
** Reason code qualifying COMPCODE
01 REASON        PIC S9(9) BINARY.

```

PL/I での呼び出し

```

call MQGET (Hconn, Hobj, MsgDesc, GetMsgOpts, BufferLength, Buffer,
DataLength, CompCode, Reason);

```

パラメーターを次のように宣言します。

```

dcl Hconn          fixed bin(31); /* Connection handle */
dcl Hobj           fixed bin(31); /* Object handle */
dcl MsgDesc        like MQMD;    /* Message descriptor */
dcl GetMsgOpts     like MQGMO;   /* Options that control the action of
MQGET */
dcl BufferLength    fixed bin(31); /* Length in bytes of the Buffer
area */
dcl Buffer          char(n);      /* Area to contain the message data */
dcl DataLength     fixed bin(31); /* Length of the message */
dcl CompCode       fixed bin(31); /* Completion code */
dcl Reason         fixed bin(31); /* Reason code qualifying CompCode */

```

高水準アセンブラー呼び出し

```

CALL MQGET, (HCONN,HOBJ,MSGDESC,GETMSGOPTS,BUFFERLENGTH,
BUFFER,DATALENGTH,COMPCODE,REASON)

```

パラメーターを次のように宣言します。

HCONN	DS	F	Connection handle
HOBJ	DS	F	Object handle
MSGDESC	CMQMDA	,	Message descriptor
GETMSGOPTS	CMQGMOA	,	Options that control the action of MQGET
BUFFERLENGTH	DS	F	Length in bytes of the BUFFER area
BUFFER	DS	CL(n)	Area to contain the message data
DATALENGTH	DS	F	Length of the message
COMPCODE	DS	F	Completion code
REASON	DS	F	Reason code qualifying COMPCODE

Visual Basic での呼び出し

```
MQGET Hconn, Hobj, MsgDesc, GetMsgOpts, BufferLength, Buffer,  
DataLength, CompCode, Reason
```

パラメーターを次のように宣言します。

```
Dim Hconn      As Long   'Connection handle'  
Dim Hobj       As Long   'Object handle'  
Dim MsgDesc    As MQMD   'Message descriptor'  
Dim GetMsgOpts As MQGMO   'Options that control the action of MQGET'  
Dim BufferLength As Long   'Length in bytes of the Buffer area'  
Dim Buffer      As String  'Area to contain the message data'  
Dim DataLength  As Long   'Length of the message'  
Dim CompCode    As Long   'Completion code'  
Dim Reason      As Long   'Reason code qualifying CompCode'
```

MQINQ - オブジェクト属性の照会

MQINQ 呼び出しは、オブジェクトの属性が入っている整数の配列と一連の文字ストリングを戻します。

次のタイプのオブジェクトが有効です。

- キュー・マネージャー
- キュー
- 名前リスト
- プロセス定義

構文

MQINQ (*Hconn*, *Hobj*, セレクタ数, セレクタ, 整数属性数, 整数属性, 文字属性長, 文字属性, コンパイルコード, 理由)

パラメーター

Hconn

タイプ: MQHCONN - 入力

このハンドルは、キュー・マネージャーに対する接続を表します。 *Hconn* の値は、前の MQCONN 呼び出しまたは MQCONNX 呼び出しによって返されたものです。

z/OS 上で CICS アプリケーションを使用する場合は、MQCONN 呼び出しを省略でき、*Hconn* に次の値を指定できます。

MQHC_DEF_HCONN

デフォルトの接続ハンドル。

Hobj

タイプ: MQHOBJ - 入力

このハンドルが、必要な属性を備えているオブジェクト (任意のタイプ) を表します。 このハンドルは、MQOO_INQUIRE オプションを指定した、前の MQOPEN 呼び出しから戻されたものでなければなりません。

SelectorCount

タイプ: MQLONG - 入力

これは、*Selectors* 配列で提供されるセレクターのカウントです。 これは戻される属性の数です。 ゼロは有効な値です。 許可される最大数は 256 です。

Selectors

タイプ: MQLONG x *SelectorCount* - 入力

これは、**SelectorCount** 個の属性セレクターの配列です。各セレクターは、必要な値を持つ属性 (整数または文字) を識別します。

各セレクターは、*Hobj* が表すオブジェクトのタイプに対して有効でなければなりません。そうでない場合、呼び出しは完了コード MQCC_FAILED および理由コード MQRC_SELECTOR_ERROR で失敗します。

特殊なケースのキューの場合は、以下のようになります。

- セレクターがどのタイプのキューに対しても有効でない場合、呼び出しは完了コード MQCC_FAILED および理由コード MQRC_SELECTOR_ERROR で失敗します。
- セレクターがオブジェクトのタイプ以外のタイプのキューにのみ適用される場合、呼び出しは完了コード MQCC_WARNING および理由コード MQRC_SELECTOR_NOT_FOR_TYPE で成功します。
- 照会するキューがクラスター・キューの場合、どのセレクターが有効かは、キューがどのように解決されたかに応じて決まります。詳細については、[712 ページの『使用上の注意』](#)を参照してください。

セレクターは、任意の順番で指定することができます。整数属性セレクター (MQIA_*セレクター) に対応する属性値は、*Selectors* でこれらのセレクターが出現するのと同じ順序で、*IntAttrs* で返されます。文字属性セレクター (MQCA_*セレクター) に対応する属性値は、セレクターが出現する順序と同じ順序で *CharAttrs* に返されます。MQIA_*セレクターは、MQCA_*セレクターとインターリーブすることができます。各タイプ内の相対順序のみが重要です。

注:

1. 整数および文字属性セレクターは、2 つの異なる範囲内で割り振られます。MQIA_*セレクターは、MQIA_FIRST から MQIA_LAST までの範囲内にあり、MQCA_*セレクターは、MQCA_FIRST から MQCA_LAST までの範囲内にあります。

各範囲について、定数 MQIA_LAST_USED および MQCA_LAST_USED は、キュー・マネージャーが受け入れる最大値を定義します。
2. すべての MQIA_*セレクターが最初に出現する場合は、同じエレメント番号を使用して、*Selectors* 配列および *IntAttrs* 配列内の対応するエレメントをアドレス指定できます。
3. **SelectorCount** パラメーターがゼロの場合、*Selectors* は参照されません。この場合、C または S/390 アセンブラーで作成されたプログラムによって渡されるパラメーター・アドレスはヌルのこともあります。

照会できる属性が、以下の表にリストされています。MQCA_*セレクターの場合、*CharAttrs* の結果のストリングの長さ (バイト単位) を定義する定数は、括弧内に示されます。

以下の表では、セレクターをオブジェクト別、アルファベット順にリストします。

- [699 ページの表 585](#) キューに関する MQINQ 属性セレクター
- [702 ページの表 586](#) 名前リストに関する MQINQ 属性セレクター
- [702 ページの表 587](#) プロセス定義に関する MQINQ 属性セレクター
- [703 ページの表 588](#) キュー・マネージャーに関する MQINQ 属性セレクター

表 585. キューに関する MQINQ 属性セレクター		
セレクター	フィールドの長さ	説明
MQCA_ALTERATION_DATE	MQ_DATE_LENGTH	最後の変更日付
MQCA_ALTERATION_TIME	MQ_TIME_LENGTH	最後の変更時刻
MQCA_BACKOUT_REQ_Q_NAME	MQ_Q_NAME_LENGTH	超過バックアウト再キューイング用のキュー名
MQCA_BASE_Q_NAME	MQ_Q_NAME_LENGTH	別名解決後のキューの名前

表 585. キューに関する MQINQ 属性セレクター (続き)		
セレクター	フィールドの長さ	説明
z/OS MQCA_CF_STRUC_NAME	MQ_CF_STRUC_NAME_LENGTH	カップリング・ファシリティー構造名
MQCA_CLUS_CHL_NAME	MQ_CHANNEL_NAME_LENGTH	このキューを伝送キューとして使用するクラスター送信側チャンネルの名前。
MQCA_CLUSTER_NAME	MQ_CLUSTER_NAME_LENGTH	クラスター名
MQCA_CLUSTER_NAMELIST	MQ_NAMELIST_NAME_LENGTH	クラスター名前リスト
MQCA_CREATION_DATE	MQ_CREATION_DATE_LENGTH	キュー作成日
MQCA_CREATION_TIME	MQ_CREATION_TIME_LENGTH	キュー作成時刻
MQCA_CUSTOM	MQ_CUSTOM_LENGTH	新機能用カスタム属性
MQCA_INITIATION_Q_NAME	MQ_Q_NAME_LENGTH	開始キュー名
MQCA_PROCESS_NAME	MQ_PROCESS_NAME_LENGTH	プロセス定義の名前
MQCA_Q_DESC	MQ_Q_DESC_LENGTH	キューの記述
MQCA_Q_NAME	MQ_Q_NAME_LENGTH	キュー名
MQCA_REMOTE_Q_MGR_NAME	MQ_Q_MGR_NAME_LENGTH	リモート・キュー・マネージャーの名前
MQCA_REMOTE_Q_NAME	MQ_Q_NAME_LENGTH	リモート・キュー・マネージャー上で認識されているリモート・キューの名前
z/OS MQCA_STORAGE_CLASS	MQ_STORAGE_CLASS_LENGTH	ストレージ・クラスの名前
MQCA_TRIGGER_DATA	MQ_TRIGGER_DATA_LENGTH	トリガー・データ
MQCA_XMIT_Q_NAME	MQ_Q_NAME_LENGTH	伝送キュー名
Multi MQIA_ACCOUNTING_Q	MQLONG	キューのアカウントिंग・データのコレクションを制御する
MQIA_BACKOUT_THRESHOLD	MQLONG	バックアウトしきい値
MQIA_CLWL_Q_PRIORITY	MQLONG	キューの優先順位
MQIA_CLWL_Q_RANK	MQLONG	キューのランク
MQIA_CLWL_USEQ	MQLONG	リモート・キュー使用
MQIA_CURRENT_Q_DEPTH	MQLONG	キュー上のメッセージの数
MQIA_DEF_BIND	MQLONG	デフォルトのバインド
MQIA_DEF_INPUT_OPEN_OPTION	MQLONG	デフォルトの入力用オープンのオプション
MQIA_DEF_PERSISTENCE	MQLONG	デフォルトのメッセージ持続性
MQIA_DEF_PRIORITY	MQLONG	デフォルトのメッセージ優先順位

表 585. キューに関する MQINQ 属性セレクトター (続き)		
セレクトター	フィールドの長さ	説明
MQIA_DEFINITION_TYPE	MQLONG	キュー定義タイプ
 MQIA_DIST_LISTS	MQLONG	配布リスト・サポート
MQIA_HARDEN_GET_BACKOUT	MQLONG	バックアウト・カウントをハード化するかどうか
 MQIA_INDEX_TYPE	MQLONG	キュー用に保守される索引のタイプ
MQIA_INHIBIT_GET	MQLONG	取得操作が許可されるかどうか
MQIA_INHIBIT_PUT	MQLONG	PUT 操作が許可されるかどうか
MQIA_MAX_MSG_LENGTH	MQLONG	最大メッセージ長
MQIA_MAX_Q_DEPTH	MQLONG	キューに許可されるメッセージの最大数
MQIA_MSG_DELIVERY_SEQUENCE	MQLONG	メッセージ優先順位が考慮されるかどうか
MQIA_NPM_CLASS	MQLONG	非持続メッセージの信頼性レベル
MQIA_OPEN_INPUT_COUNT	MQLONG	キューを入力用にオープンする MQOPEN 呼び出しの数
MQIA_OPEN_OUTPUT_COUNT	MQLONG	キューを出力用にオープンする MQOPEN 呼び出しの数
MQIA_PROPERTY_CONTROL	MQLONG	プロパティ制御属性
 MQIA_Q_DEPTH_HIGH_EVENT	MQLONG	キュー・サイズ上限イベントの制御属性
 MQIA_Q_DEPTH_HIGH_LIMIT	MQLONG	キュー・サイズの上限
 MQIA_Q_DEPTH_LOW_EVENT	MQLONG	キュー・サイズ下限イベントの制御属性
 MQIA_Q_DEPTH_LOW_LIMIT	MQLONG	キュー・サイズの下限
 MQIA_Q_DEPTH_MAX_EVENT	MQLONG	キュー・サイズ最大イベントの制御属性
 MQIA_Q_SERVICE_INTERVAL	MQLONG	キュー・サービス間隔の限度
 MQIA_Q_SERVICE_INTERVAL_EVENT	MQLONG	キュー・サービス間隔イベントの制御属性
MQIA_Q_TYPE	MQLONG	キュー・タイプ
 MQIA_QSG_DISP	MQLONG	キュー共有グループ後処理
MQIA_RETENTION_INTERVAL	MQLONG	キュー保持期間間隔
 MQIA_SCOPE	MQLONG	キュー定義の有効範囲

表 585. キューに関する MQINQ 属性セクター (続き)		
セクター	フィールドの長さ	説明
MQIA_SHAREABILITY	MQLONG	入力のためのキューを共有できるかどうか
 MQIA_STATISTICS_Q	MQLONG	キューの統計データのコレクションを制御する
MQIA_TRIGGER_CONTROL	MQLONG	トリガー制御
MQIA_TRIGGER_DEPTH	MQLONG	トリガー項目数
MQIA_TRIGGER_MSG_PRIORITY	MQLONG	トリガーのしきい値メッセージ優先順位
MQIA_TRIGGER_TYPE	MQLONG	トリガー・タイプ
MQIA_USAGE	MQLONG	使用法

表 586. 名前リストに関する MQINQ 属性セクター		
セクター	フィールドの長さ	説明
MQCA_ALTERATION_DATE	MQ_DATE_LENGTH	最後の変更の日付
MQCA_ALTERATION_TIME	MQ_TIME_LENGTH	最後の変更の時刻
MQCA_NAMELIST_DESC	MQ_NAMELIST_DESC_LENGTH	名前リストの記述
MQCA_NAMELIST_NAME	MQ_NAMELIST_NAME_LENGTH	名前リスト・オブジェクトの名前。
 MQIA_NAMELIST_TYPE	MQLONG	名前リスト・タイプ
MQCA_NAMES	MQ_Q_NAME_LENGTH x Number of names in the list	名前リストにある名前。
MQIA_NAME_COUNT	MQLONG	名前リスト内の名前数
 MQIA_QSG_DISP	MQLONG	キュー共有グループ後処理

表 587. プロセス定義に関する MQINQ 属性セクター		
セクター	フィールドの長さ	説明
MQCA_ALTERATION_DATE	MQ_DATE_LENGTH	最後の変更の日付
MQCA_ALTERATION_TIME	MQ_TIME_LENGTH	最後の変更の時刻
MQCA_APPL_ID	MQ_PROCESS_APPL_ID_LENGTH	アプリケーション ID
MQCA_ENV_DATA	MQ_PROCESS_ENV_DATA_LENGTH	環境データ
MQCA_PROCESS_DESC	MQ_PROCESS_DESC_LENGTH	プロセス定義の記述。
MQCA_PROCESS_NAME	MQ_PROCESS_NAME_LENGTH	プロセス定義の名前
MQCA_USER_DATA	MQ_PROCESS_USER_DATA_LENGTH	ユーザー・データ

表 587. プロセス定義に関する MQINQ 属性セクター (続き)		
セクター	フィールドの長さ	説明
MQIA_APPL_TYPE	MQLONG	アプリケーション・タイプ
 MQIA_QSG_DISP	MQLONG	キュー共有グループ後処理

表 588. キュー・マネージャーに関する MQINQ 属性セクター		
セクター	フィールドの長さ	説明
MQCA_ALTERATION_DATE	MQ_DATE_LENGTH	最後の変更の日付
MQCA_ALTERATION_TIME	MQ_TIME_LENGTH	最後の変更の時刻
MQCA_CHANNEL_AUTO_DEF_EXIT	MQ_EXIT_NAME_LENGTH	自動チャンネル定義出口名。
MQCA_CHINIT_SERVICE_PARM		IBM の使用のため予約済み
MQCA_CLUSTER_WORKLOAD_DATA	MQ_EXIT_DATA_LENGTH	クラスター・ワークロード出口に渡されるデータ。
MQCA_CLUSTER_WORKLOAD_EXIT	MQ_EXIT_NAME_LENGTH	クラスター・ワークロード出口名
MQCA_COMMAND_INPUT_Q_NAME	MQ_Q_NAME_LENGTH	システム・コマンド入力キュー名。
MQCA_CUSTOM	MQ_CUSTOM_LENGTH	新機能用カスタム属性
MQCA_DEAD_LETTER_Q_NAME	MQ_Q_NAME_LENGTH	送達不能キューの名前。
MQCA_DEF_XMIT_Q_NAME	MQ_Q_NAME_LENGTH	デフォルトの伝送キューの名前。
 MQCA_DNS_GROUP	MQ_DNS_GROUP_NAME_LENGTH	キュー共有グループのインバウンド伝送を処理する TCP リスナーが参加するグループの名前。この名前は、Workload Manager Dynamic Domain Name Services を使用するとき適用されます。
 MQCA_IGQ_USER_ID	MQ_USER_ID_LENGTH	グループ内キューイングのユーザー ID。
MQCA_INITIAL_KEY	MQ_INITIAL_KEY_LENGTH	パスワード保護システムの初期キー 空白でない場合を ***** 返す。デフォルトの初期キーが使用されている場合は空白を返す。
 MQCA_INSTALLATION_DESC	MQ_INSTALLATION_DESC_LENGTH	関連するインストール済み環境の記述。
 MQCA_INSTALLATION_NAME	MQ_INSTALLATION_NAME_LENGTH	キュー・マネージャーに関連付けられたインストールの名前。
 MQCA_INSTALLATION_PATH	MQ_INSTALLATION_PATH_LENGTH	関連する IBM MQ がインストールされる場所のパス
 MQCA_LU_GROUP_NAME	MQ_LU_NAME_LENGTH	使用するキュー共有グループのインバウンド送信を処理する LU 6.2 リスナーの総称 LU 名。

表 588. キュー・マネージャーに関する MQINQ 属性セクター (続き)

セクター	フィールドの長さ	説明
▶ z/OS MQCA_LU_NAME	MQ_LU_NAME_LENGTH	アウトバウンド LU 6.2 伝送で使用する LU の名前。この名前をリスナーがインバウンド伝送で使用するのと同じ LU に設定します。
▶ z/OS MQCA_LU62_ARM_SUFFIX	MQ_ARM_SUFFIX_LENGTH	このチャネル・イニシエーター用の LUADD を指定する、SYS1.PARMLIB メンバー APPCPM <i>xx</i> の接尾部
MQCA_PARENT	MQ_Q_MGR_NAME_LENGTH	このキュー・マネージャーの親として候補に挙げられた、階層的に接続されたキュー・マネージャーの名前。
MQCA_Q_MGR_DESC	MQ_Q_MGR_DESC_LENGTH	キュー・マネージャーの記述。
MQCA_Q_MGR_IDENTIFIER	MQ_Q_MGR_IDENTIFIER_LENGTH	キュー・マネージャー ID (H)
MQCA_Q_MGR_NAME	MQ_Q_MGR_NAME_LENGTH	ローカル・キュー・マネージャーの名前
▶ z/OS MQCA_QSG_NAME	MQ_QSG_NAME_LENGTH	キュー共有グループ名
MQCA_REPOSITORY_NAME	MQ_CLUSTER_NAME_LENGTH	キュー・マネージャーがリポジトリ・サービスを提供しているクラスターの名前。
MQCA_REPOSITORY_NAMELIST	MQ_NAMELIST_NAME_LENGTH	キュー・マネージャーがリポジトリ・サービスを提供しているクラスターの名前を含む名前リスト・オブジェクトの名前。
MQCA_SSL_KEY_REPO_PASSWORD	MQ_SSL_ENCRYPT_KEY_REPO_PWD_LEN	キーリポジトリのパスワード 空白でない場合は ***** その値を返し、設定されていない場合は空白を返す。 保存前に設定すると暗号化されます。
▶ z/OS MQCA_TCP_NAME	MQ_TCP_NAME_LENGTH	使用している TCP/IP システムの名前
▶ Multi MQIA_ACCOUNTING_CONN_OVERRIDE	MQLONG	アカウントिंग設定をオーバーライドする
▶ Multi MQIA_ACCOUNTING_INTERVAL	MQLONG	中間アカウント・レコードを書き込む頻度
▶ Multi MQIA_ACCOUNTING_MQI	MQLONG	MQI データに関するアカウントिंग情報の収集を制御します。
▶ Multi MQIA_ACCOUNTING_Q	MQLONG	キューのアカウントिंग情報のコレクションを制御する
▶ z/OS MQIA_ACTIVE_CHANNELS	MQLONG	いつでもアクティブにできるチャネルの最大数

表 588. キュー・マネージャーに関する MQINQ 属性セクター (続き)

セクター	フィールドの長さ	説明
▶ z/OS MQIA_ADOPTNEWMCA_CHECK	MQLONG	MCA を受け入れるかどうか判別するためにチェックするエレメント。 このチェックは、既にアクティブになっている MCA と同じ名前のインバウンド・チャンネルが新たに検出されたときに行われます。
▶ Multi MQIA_ADOPTNEWMCA_INTERVAL	MQLONG	孤立したチャンネルが終了するまで新しいチャンネルが待機する時間 (秒)
▶ z/OS MQIA_ADOPTNEWMCA_TYPE	MQLONG	AdoptNewMCACheck パラメーターと一致する新しいインバウンド・チャンネル要求が検出された場合に、指定されたチャンネル・タイプの MCA の孤立したインスタンスを自動的に再始動するかどうか
▶ Multi MQIA_AUTHORITY_EVENT	MQLONG	権限イベントの制御属性
▶ z/OS MQIA_BRIDGE_EVENT	MQLONG	IMS ブリッジ・イベントの制御属性
▶ Multi MQIA_CHANNEL_AUTO_DEF	MQLONG	自動チャンネル定義の制御属性
▶ Multi MQIA_CHANNEL_AUTO_DEF_EVENT	MQLONG	自動チャンネル定義イベントの制御属性
MQIA_CHANNEL_EVENT	MQLONG	チャンネル・イベントの制御属性
▶ z/OS MQIA_CHINIT_ADAPTERS	MQLONG	IBM MQ 呼び出しの処理に使用するアダプター・サブタスクの数
▶ z/OS MQIA_CHINIT_DISPATCHERS	MQLONG	チャンネル・イニシエーターで使用するディスパッチャーの数
▶ z/OS MQIA_CHINIT_TRACE_AUTO_START	MQLONG	チャンネル・イニシエーター・トレースを自動的に開始するかどうか
▶ z/OS MQIA_CHINIT_TRACE_TABLE_SIZE	MQLONG	チャンネル・イニシエーターのトレース・データ・スペースのサイズ (MB)
MQIA_CLUSTER_WORKLOAD_LENGTH	MQLONG	クラスター・ワークロードの長さ。
MQIA_CLWL_MRU_CHANNELS	MQLONG	クラスターのワークロード・バランシング用に最近使用されたチャンネルの数
MQIA_CLWL_USEQ	MQLONG	リモート・キュー使用
MQIA_CODED_CHAR_SET_ID	MQLONG	コード化文字セット ID
MQIA_COMMAND_EVENT	MQLONG	コマンド・イベントの制御属性

表 588. キュー・マネージャーに関する MQINQ 属性セレクター (続き)

セレクター	フィールドの長さ	説明
MQIA_COMMAND_LEVEL	MQLONG	キュー・マネージャーでサポートされる コマンド・レベル
 MQIA_CONFIGURATION_EVENT	MQLONG	構成イベントの制御属性
MQIA_DEF_CLUSTER_XMIT_Q_TY PE	MQLONG	クラスター送信側チャンネルにデフォルト で使用する伝送キュー・タイプ。
 MQIA_DIST_LISTS	MQLONG	配布リスト・サポート
 MQIA_DNS_WLM	MQLONG	キュー共有グループのインバウンド伝送 を処理する TCP リスナーを Workload Manager for Dynamic Domain Name Services に登録するかどうか
 MQIA_EXPIRY_INTERVAL	MQLONG	有効期限切れメッセージのスキャンを実 行する間隔
 MQIA_GROUP_UR	MQLONG	このキュー・マネージャーで GROUP リカ バリー単位を有効にするかどうかを指定 する制御属性。 GROUP リカバリー単位 処理は、キュー・マネージャーがキュー 共有グループのメンバーである場合にの み使用可能です。
 MQIA_IGQ_PUT_AUTHORITY	MQLONG	グループ内キューイング書き込み権限
 MQIA_INHIBIT_EVENT	MQLONG	禁止イベントの制御属性
 MQIA_INTRA_GROUP_QUEUING	MQLONG	グループ内キューイングのサポート
 MQIA_LISTENER_TIMER	MQLONG	APPC または TCP/IP が失敗した場合に、 IBM MQ がリスナーの再開を試行するま での時間間隔 (秒)
 MQIA_LOCAL_EVENT	MQLONG	ローカル・イベントの制御属性
 MQIA_LOGGER_EVENT	MQLONG	禁止イベントの制御属性
 MQIA_LU62_CHANNELS	MQLONG	LU 6.2 伝送プロトコルを使用して、現行 チャンネルとして可能なチャンネル、または 接続できるクライアントの最大数
MQIA_MSG_MARK_BROWSE_INTER VAL	MQLONG	キュー・マネージャーがブラウズ・メッ セージからマークを自動的に除去する時 間間隔 (ミリ秒)。  重要: 値をデフォルトの 5000 より 小さく設定しないでください。

表 588. キュー・マネージャーに関する MQINQ 属性セクター (続き)

セクター	フィールドの長さ	説明
 MQIA_MAX_CHANNELS	MQLONG	現行チャンネルとして可能な最大チャンネル数 (接続されたクライアントを使用したサーバー接続チャンネルを含む)
MQIA_MAX_HANDLES	MQLONG	ハンドルの最大数
MQIA_MAX_MSG_LENGTH	MQLONG	最大メッセージ長
MQIA_MAX_PRIORITY	MQLONG	最高の優先順位
MQIA_MAX_UNCOMMITTED_MSGS	MQLONG	1 つの作業単位内のコミットされていないメッセージの最大数
 MQIA_OUTBOUND_PORT_MAX	MQLONG	MQIA_OUTBOUND_PORT_MIN を使用して、発信チャンネルをバインドするときに使用するポート番号の範囲を定義します。
 MQIA_OUTBOUND_PORT_MIN	MQLONG	MQIA_OUTBOUND_PORT_MAX を使用して、発信チャンネルをバインドするときに使用するポート番号の範囲を定義します。
 MQIA_PERFORMANCE_EVENT	MQLONG	パフォーマンス・イベントの制御属性
MQIA_PLATFORM	MQLONG	キュー・マネージャーがあるプラットフォーム
MQIA_PROT_POLICY_CAPABILITY	MQLONG	キュー・マネージャーで Advanced Message Security のセキュリティー機能が使用可能かどうかを示します。
MQIA_PUBSUB_MAXMSG_RETRY_COUNT	MQLONG	同期点における、失敗したコマンド・メッセージの再処理の試行回数。
MQIA_PUBSUB_MODE	MQLONG	パブリッシュ/サブスクライブ・エンジンとキュー・パブリッシュ/サブスクライブ・インターフェースが実行されているかどうか アプリケーション・プログラミング・インターフェースを使用してパブリッシュまたはサブスクライブを行うアプリケーションは、パブリッシュ/サブスクライブ・エンジンが必要です。キュー・パブリッシュ/サブスクライブ・インターフェースがモニターするキューの場合、キュー・パブリッシュ/サブスクライブ・インターフェースが実行されている必要があります。
MQIA_PUBSUB_NP_MSG	MQLONG	未配信の入力メッセージを廃棄 (または保持) するかどうか。
MQIA_PUBSUB_NP_RESP	MQLONG	未配信の応答メッセージの動作を制御します。

表 588. キュー・マネージャーに関する MQINQ 属性セレクター (続き)

セレクター	フィールドの長さ	説明
MQIA_PUBSUB_SYNC_PT	MQLONG	持続メッセージのみ (またはすべてのメッセージ) を同期点で処理するかどうかを指定します。
 MQIA_QMGR_CFCONLOS	MQLONG	CFCONLOS が ASQMGR に設定されている管理構造体または CF 構造体への接続をキュー・マネージャーが失ったときに実行されるアクションを指定します。
 MQIA_RECEIVE_TIMEOUT	MQLONG	TCP/IP チャンネルが、ハートビートを含むデータをそのパートナーから受信してから、非アクティブ状態に戻るまで待機するおおよその時間。値は、MQIA_RECEIVE_TIMEOUT_TYPE で修飾された数値です。
 MQIA_RECEIVE_TIMEOUT_MIN	MQLONG	TCP/IP チャンネルが、ハートビートを含むデータをそのパートナーから受信してから、非アクティブ状態に戻るまで待機する最小時間
 MQIA_RECEIVE_TIMEOUT_TYPE	MQLONG	TCP/IP チャンネルが、ハートビートを含むデータをそのパートナーから受信してから、非アクティブ状態に戻るまで待機するおおよその時間。 MQIA_RECEIVE_TIMEOUT_TYPE は、MQIA_RECEIVE_TIMEOUT に適用される修飾子です。
 MQIA_REMOTE_EVENT	MQLONG	リモート・イベントの制御属性
 MQIA_SECURITY_CASE	MQLONG	セキュリティ・プロファイルのケース
MQIA_SSL_EVENT	MQLONG	チャンネル・イベントの制御属性
MQIA_SSL_FIPS_REQUIRED	MQLONG	暗号化には認定アルゴリズム FIPS のみを使用すること
MQIA_SSL_RESET_COUNT	MQLONG	TLS 鍵リセット・カウント
 MQIA_START_STOP_EVENT	MQLONG	開始 / 停止イベントの制御属性
MQIA_STATISTICS_AUTO_CLUSSDR	MQLONG	クラスター送信側チャンネルの統計モニター情報のコレクションを制御する
MQIA_STATISTICS_CHANNEL	MQLONG	チャンネルの統計データのコレクションを制御する
 MQIA_STATISTICS_INTERVAL	MQLONG	統計モニター・データを書き込む頻度
 MQIA_STATISTICS_MQI	MQLONG	キュー・マネージャーに関する統計モニター情報の収集を制御します。

表 588. キュー・マネージャーに関する MQINQ 属性セレクター (続き)		
セレクター	フィールドの長さ	説明
 MQIA_STATISTICS_Q	MQLONG	キューの統計データのコレクションを制御する
MQIA_SYNCPOINT	MQLONG	同期点の使用可能性
 MQIA_TCP_CHANNELS	MQLONG	TCP/IP 伝送プロトコルを使用して、現行チャンネルとして可能なチャンネル、または接続できるクライアントの最大数
 MQIA_TCP_KEEP_ALIVE	MQLONG	接続のもう一方の側が依然として使用可能であるかを検査する TCP KEEPALIVE 機能を使用するかどうか
 MQIA_TCP_STACK_TYPE	MQLONG	チャンネル・イニシエーターが TCPNAME で指定された TCP/IP アドレス・スペースのみを使用できるか、あるいは、任意で選択した TCP/IP アドレスにオプションでバインドできるか
 MQIA_TRACE_ROUTE_RECORDING	MQLONG	トレース経路指定情報の記録を制御する
MQIA_TREE_LIFE_TIME	MQLONG	未使用の非管理トピックの存続時間
MQIA_TRIGGER_INTERVAL	MQLONG	トリガー間隔

IntAttrCount

タイプ: MQLONG - 入力

これは、*IntAttrs* 配列内のエレメントの数です。ゼロは有効な値です。

IntAttrCount が少なくとも **Selectors** パラメーター内の MQIA_*セレクターの数である場合、要求されたすべての整数属性が返されます。

IntAttrs

タイプ: MQLONG x *IntAttrCount* - 出力

これは、*IntAttrCount* 個の整数属性値の配列です。

整数属性値は、**Selectors** パラメーター内の MQIA_*セレクターと同じ順序で返されます。配列に含まれているエレメントの数が MQIA_*セレクターの数より多い場合、超過分のエレメントは変更されません。

Hobj がキューを表しているが、属性セレクターがそのタイプのキューに適用されない場合は、特定の値 MQIAV_NOT_APPLICABLE が返されます。これは、*IntAttrs* 配列内の対応するエレメントに対して返されます。

IntAttrCount または **SelectorCount** パラメーターがゼロの場合、*IntAttrs* は参照されません。この場合、C または S/390 アセンブラーで作成されたプログラムによって渡されるパラメーター・アドレスはヌルのこともあります。

CharAttrLength

タイプ: MQLONG - 入力

これは、**CharAttrs** パラメーターの長さ (バイト単位) です。

CharAttrLength は、要求された文字属性の長さの合計以上でなければなりません ([Selectors](#) を参照)。ゼロは有効な値です。

CharAttrs

タイプ: MQCHAR x CharAttrLength - 出力

これは、各文字属性が連結されて返されるバッファです。バッファの長さは、**CharAttrLength** パラメーターによって指定されます。

文字属性は、**Selectors** パラメーター内の MQCA_*セクターと同じ順序で返されます。各属性ストリングの長さは、各属性で固定であり (**Selectors** を参照)、その中の値には、必要に応じて右側にブラックが埋め込まれます。要求されたすべての文字属性および埋め込みを含めるために必要なサイズよりも大きいバッファを指定できます。返されるバイトのうち、最後の属性値を超過する分は変更されません。

Hobj がキューを表しているが、属性セクターがそのタイプのキューに適用されない場合は、アスタリスク (*) のみで構成される文字ストリングが返されます。アスタリスクは、その属性の値として CharAttrs に返されます。

CharAttrLength または **SelectorCount** パラメーターがゼロの場合、CharAttrs は参照されません。この場合、C または S/390 アセンブラで作成されたプログラムによって渡されるパラメーター・アドレスはヌルのこともあります。

CompCode

タイプ: MQLONG - 出力

完了コード:

MQCC_OK

正常終了。

MQCC_WARNING

警告 (部分完了)。

MQCC_FAILED

呼び出し失敗。

理由

タイプ: MQLONG - 出力

CompCode が MQCC_OK の場合:

MQRC_NONE

(0, X'000') 報告する理由はありません。

CompCode が MQCC_WARNING の場合:

MQRC_CHAR_ATTRS_TOO_SHORT

(2008, X'7D8') 文字属性に十分なスペースがありません。

MQRC_INT_ATTR_COUNT_TOO_SMALL

(2022, X'7E6') 整数属性に十分なスペースがありません。

MQRC_SELECTOR_NOT_FOR_TYPE

(2068, X'814') セクターはキュー・タイプに適用されません。

CompCode が MQCC_FAILED の場合:

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'89C') アダプターが使用できません。

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852') アダプター・サービス・モジュールをロードできません。

MQRC_API_EXIT_ERROR

(2374, X'946') API 出口が失敗しました。

MQRC_API_EXIT_LOAD_ERROR

(2183, X'887') API 出口をロードできません。

MQRC_ASID_MISMATCH

(2157, X'86D') 1 次 ASID とホーム ASID が異なります。

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') 前の呼び出しが完了する前に MQI 呼び出しが入力されました。

MQRC_CF_STRUC_FAILED

(2373, X'945') カップリング・ファシリティ構造に障害が発生しました。

MQRC_CF_STRUC_IN_USE

(2346, X'92A') カップリング・ファシリティ・ストラクチャーが使用中である。

MQRC_CHAR_ATTR_LENGTH_ERROR

(2006, X'7D6') 文字属性の長さが無効です。

MQRC_CHAR_ATTRS_ERROR

(2007, X'7D7') 文字属性ストリングが無効です。

MQRC_CICS_WAIT_FAILED

(2140, X'85C') 待機要求が CICS によって拒否されました。

MQRC_CONNECTION_BROKEN

(2009, X'7D9') キュー・マネージャーへの接続が失われました。

MQRC_CONNECTION_NOT_AUTHORIZED

(2217, X'8A9') 接続が許可されていません。

MQRC_CONNECTION_STOPPING

(2203, X'89B') 接続をシャットダウンしています。

MQRC_HCONN_ERROR

(2018, X'7E2') 接続ハンドルが無効です。

MQRC_HOBJ_ERROR

(2019, X'7E3') オブジェクト・ハンドルが無効です。

MQRC_INT_ATTR_COUNT_ERROR

(2021, X'7E5') 整数属性のカウントが無効です。

MQRC_INT_ATTRS_ARRAY_ERROR

(2023, X'7E7') 整数属性配列が無効です。

MQRC_NOT_OPEN_FOR_INQUIRE

(2038, X'7F6') キューが照会用にオープンされていません。

MQRC_OBJECT_CHANGED

(2041, X'7F9') オブジェクト定義がオープン後に変更されました。

MQRC_OBJECT_DAMAGED

(2101, X'835') オブジェクトに損傷があります。

MQRC_PAGESET_ERROR

(2193, X'891') ページ・セット・データ・セットへのアクセス・エラー。

MQRC_Q_DELETED

(2052, X'804') キューが削除されました。

MQRC_Q_MGR_NAME_ERROR

(2058, X'80A') キュー・マネージャー名が無効であるか、認識されていません。

MQRC_Q_MGR_NOT_AVAILABLE

(2059, X'80B') キュー・マネージャーが接続に使用できません。

MQRC_Q_MGR_STOPPING

(2162, X'872') キュー・マネージャーがシャットダウンしています。

MQRC_RESOURCE_PROBLEM

(2102, X'836') 使用可能なシステム・リソースが不足しています。

MQRC_SELECTOR_COUNT_ERROR

(2065, X'811') セレクターのカウントが無効です。

MQRC_SELECTOR_ERROR

(2067, X'813') 属性セレクターが無効です。

MQRC_SELECTOR_LIMIT_EXCEEDED

(2066, X'812') セレクターの数が多すぎます。

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') 使用可能なストレージが不足しています。

MQRC_SUPPRESSED_BY_EXIT

(2109, X'83D') 出口プログラムによって呼び出しが抑止されました。

MQRC_UNEXPECTED_ERROR

(2195, X'893') 予期しないエラーが発生しました。

これらのコードの詳細については、[メッセージおよび理由コード](#)を参照してください。

使用上の注意

1. 戻される値は、選択された属性のスナップショットです。戻り値に応じたアクションをアプリケーションが実行するまで、属性が同じ状態を維持するという保証はありません。
2. モデル・キューをオープンする場合は、動的ローカル・キューが作成されます。動的ローカル・キューは、属性に関して照会するためにモデル・キューをオープンする場合にも作成されます。

動的キューの属性は、ほとんどの場合、動的キューが作成された時点のモデル・キューの属性と同じです。その後、ユーザーがこのキューの MQINQ 呼び出しを使用すると、キュー・マネージャーは、モデル・キューの属性ではなく動的キューの属性を戻します。モデル・キューのどの属性が動的キューから継承されるかの詳細については、[832 ページの表 597](#) を参照してください。

3. 照会対象のオブジェクトが別名キューの場合、MQINQ 呼び出しから返される属性値は、その別名キューの属性です。別名が解決される先の基本キューまたはトピックの属性ではありません。
4. 照会するオブジェクトがクラスター・キューの場合、以下のように、照会できる属性は、そのキューをどのようにオープンしたかによって決まります。

- 照会に加えて、入力、ブラウズ、または設定のうちの 1 つ以上の操作を目的としてクラスター・キューをオープンできます。正常にオープンするには、クラスター・キューのローカル・インスタンスが必要です。この場合、照会できる属性は、ローカル・キューに有効な属性です。

クラスター・キューが入力、ブラウズ、または設定を指定せずに照会のためにオープンされている場合、ローカル・キューに対してのみ有効な属性を照会しようとする、呼び出しは完了コード MQCC_WARNING および理由コード MQRC_SELECTOR_NOT_FOR_TYPE (2068) を返します。

- 接続されたキュー・マネージャーの基本キュー・マネージャー名を渡す際に、照会のためにクラスター・キューをオープンすることができます。

正常にオープンするには、クラスター・キューのローカル・インスタンスが必要です。基本キュー・マネージャーが渡されない場合、クラスター・キューではなくローカル・キューでのみ有効な属性を照会しようとする、呼び出しは完了コード MQCC_WARNING および理由コード MQRC_SELECTOR_NOT_FOR_TYPE (2068) を返します。

- 照会だけ、または照会と出力を目的としてクラスター・キューをオープンする場合は、照会できるのはリストにある属性だけです。この場合、**QType** 属性の値は MQQT_CLUSTER になります。

- MQCA_Q_DESC
- MQCA_Q_NAME
- MQIA_DEF_BIND
- MQIA_DEF_PERSISTENCE
- MQIA_DEF_PRIORITY
- MQIA_INHIBIT_PUT
- MQIA_Q_TYPE

クラスター・キューは、バインディングを固定しなくてもオープンできます。オープンするには、MQOPEN 呼び出しに MQOO_BIND_NOT_FIXED を指定します。あるいは、MQOO_BIND_AS_Q_DEF を指定し、キューの **DefBind** 属性を MQBND_BIND_NOT_FIXED に設定します。バインディングを固定

せずにクラスター・キューをオープンすると、キューに対する後続の MQINQ 呼び出しが、そのクラスター・キューの別のインスタンスを照会する可能性があります。ただし、すべてのインスタンスは同じ属性を持つのが通常です。

- 別名キュー・オブジェクトはクラスター用に定義することができます。TARGTYPE および TARGET はクラスター属性ではないため、別名キューで MQOPEN プロセスを実行するプロセスは、別名が解決されるオブジェクトを認識しません。

初回 MQOPEN 時に、別名キューはクラスター内のキュー・マネージャーおよびキューに解決されます。リモート・キュー・マネージャーで名前の解決が再度行われ、別名キューの TARGTYPE もここで解決されます。

別名キューがトピックの別名によって解決された場合、別名キューに書き込まれたメッセージの発行はこのリモート・キュー・マネージャーで行われます。

[クラスター・キュー](#)を参照してください。

5. 多数の属性を照会した後、MQSET 呼び出しを使用して、そのうちの一部の属性を設定することもできます。属性の照会と設定操作を効率的にプログラムするには、設定対象の属性をセレクター配列の先頭に配置します。このようにすると、配列のカウントが減った同じ配列を MQSET に使用できます。
6. 複数の警告状態が発生した場合 (**CompCode** パラメーターを参照)、返される理由コードは、以下のリストのうち該当する最初の理由コードです。
 - a. MQRC_SELECTOR_NOT_FOR_TYPE
 - b. MQRC_INT_ATTR_COUNT_TOO_SMALL
 - c. MQRC_CHAR_ATTRS_TOO_SHORT
7. オブジェクト属性については、以下のトピックで説明しています。
 - [830 ページの『キューの属性』](#)
 - [864 ページの『名前リストの属性』](#)
 - [866 ページの『プロセス定義の属性』](#)
 - [791 ページの『キュー・マネージャーの属性』](#)

C 言語での呼び出し

```
MQINQ (Hconn, Hobj, SelectorCount, Selectors, IntAttrCount, IntAttrs,  
        CharAttrLength, CharAttrs, &CompCode, &Reason);
```

パラメーターを次のように宣言します。

```
MQHCONN  Hconn;           /* Connection handle */  
MQHOBJ   Hobj;            /* Object handle */  
MQLONG   SelectorCount;   /* Count of selectors */  
MQLONG   Selectors[n];    /* Array of attribute selectors */  
MQLONG   IntAttrCount;    /* Count of integer attributes */  
MQLONG   IntAttrs[n];     /* Array of integer attributes */  
MQLONG   CharAttrLength;  /* Length of character attributes buffer */  
MQCHAR   CharAttrs[n];    /* Character attributes */  
MQLONG   CompCode;        /* Completion code */  
MQLONG   Reason;         /* Reason code qualifying CompCode */
```

COBOL での呼び出し

```
CALL 'MQINQ' USING HCONN, HOBJ, SELECTORCOUNT, SELECTORS-TABLE,  
                  INTATTRCOUNT, INTATTRS-TABLE, CHARATTRLENGTH,  
                  CHARATTRS, COMPCODE, REASON.
```

パラメーターを次のように宣言します。

```
**   Connection handle  
01  HCONN          PIC S9(9) BINARY.  
**   Object handle
```

```

01 HOBJ          PIC S9(9) BINARY.
** Count of selectors
01 SELECTORCOUNT PIC S9(9) BINARY.
** Array of attribute selectors
01 SELECTORS-TABLE.
02 SELECTORS     PIC S9(9) BINARY OCCURS n TIMES.
** Count of integer attributes
01 INTATTRCOUNT PIC S9(9) BINARY.
** Array of integer attributes
01 INTATTRS-TABLE.
02 INTATTRS      PIC S9(9) BINARY OCCURS n TIMES.
** Length of character attributes buffer
01 CHARATTRLENGTH PIC S9(9) BINARY.
** Character attributes
01 CHARATTRS      PIC X(n).
** Completion code
01 COMPCODE       PIC S9(9) BINARY.
** Reason code qualifying COMPCODE
01 REASON         PIC S9(9) BINARY.

```

PL/I での呼び出し

```
call MQINQ (Hconn, Hobj, SelectorCount, Selectors, IntAttrCount,
           IntAttrs, CharAttrLength, CharAttrs, CompCode, Reason);
```

パラメーターを次のように宣言します。

```

dcl Hconn      fixed bin(31); /* Connection handle */
dcl Hobj       fixed bin(31); /* Object handle */
dcl SelectorCount fixed bin(31); /* Count of selectors */
dcl Selectors(n) fixed bin(31); /* Array of attribute selectors */
dcl IntAttrCount fixed bin(31); /* Count of integer attributes */
dcl IntAttrs(n)  fixed bin(31); /* Array of integer attributes */
dcl CharAttrLength fixed bin(31); /* Length of character attributes
                                buffer */
dcl CharAttrs   char(n); /* Character attributes */
dcl CompCode    fixed bin(31); /* Completion code */
dcl Reason      fixed bin(31); /* Reason code qualifying
                                CompCode */

```

高水準アセンブラー呼び出し

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

パラメーターを次のように宣言します。

HCONN	DS	F	Connection handle
HOBJ	DS	F	Object handle
SELECTORCOUNT	DS	F	Count of selectors
SELECTORS	DS	(n)F	Array of attribute selectors
INTATTRCOUNT	DS	F	Count of integer attributes
INTATTRS	DS	(n)F	Array of integer attributes
CHARATTRLENGTH	DS	F	Length of character attributes buffer
CHARATTRS	DS	CL(n)	Character attributes
COMPCODE	DS	F	Completion code
REASON	DS	F	Reason code qualifying COMPCODE

Visual Basic での呼び出し

```
MQINQ Hconn, Hobj, SelectorCount, Selectors, IntAttrCount, IntAttrs,
CharAttrLength, CharAttrs, CompCode, Reason
```

パラメーターを次のように宣言します。

Dim Hconn	As Long	'Connection handle'
Dim Hobj	As Long	'Object handle'
Dim SelectorCount	As Long	'Count of selectors'
Dim Selectors	As Long	'Array of attribute selectors'
Dim IntAttrCount	As Long	'Count of integer attributes'

Dim IntAttrs	As Long	'Array of integer attributes'
Dim CharAttrLength	As Long	'Length of character attributes buffer'
Dim CharAttrs	As String	'Character attributes'
Dim CompCode	As Long	'Completion code'
Dim Reason	As Long	'Reason code qualifying CompCode'

MQINQMP - メッセージ・プロパティの照会

MQINQMP 呼び出しは、メッセージのプロパティの値を戻します。

構文

MQINQMP (*Hconn*, *Hmsg*, *InqPropOpts*, *Name*, *PropDesc*, *Type*, *ValueLength*, *Value*, *DataLength*, *CompCode*, *Reason*)

パラメーター

Hconn

タイプ: MQHCONN - 入力

このハンドルは、キュー・マネージャーに対する接続を表します。 *Hconn* の値は、**Hmsg** パラメーターで指定されているメッセージ・ハンドルを作成するために使用された接続ハンドルと一致していなければなりません。

MQHC_UNASSOCIATED_HCONN を使用してメッセージ・ハンドルが作成された場合は、メッセージ・ハンドルのプロパティを照会するスレッド上で有効な接続を確立しなければなりません。確立しないと、呼び出しは MQRC_CONNECTION_BROKEN で失敗します。

Hmsg

タイプ: MQHMSG - 入力

これは照会されるメッセージ・ハンドルです。値は、前の **MQCRTMH** 呼び出しで戻されたものです。

InqPropOpts

タイプ: MQIMPO - 入出力

詳細については、[MQIMPO](#) データ・タイプを参照してください。

名前

タイプ: MQCHARV - 入出力

照会するプロパティの名前。

この名前のプロパティを検出できない場合は、呼び出しは理由 MQRC_PROPERTY_NOT_AVAILABLE で失敗します。

プロパティ名の末尾にワイルドカード文字の % 記号を使用できます。このワイルドカードは、ピリオド (.) を含むゼロ個以上の文字と一致します。そのため、アプリケーションが多数のプロパティの値を照会できます。オプション MQIMPO_INQ_FIRST を指定して MQINQMP を呼び出すと、最初の一致したプロパティが読み取られ、オプション MQIMPO_INQ_NEXT を指定して再び呼び出すと、次の一致したプロパティが読み取られます。選択可能な一致プロパティがなくなると、呼び出しは MQRC_PROPERTY_NOT_AVAILABLE で失敗します。InqPropOpts 構造体の *ReturnedName* フィールドが、プロパティに返された名前のアドレスまたはオフセットで初期化されると、MQINQMP から戻った時点で、一致したプロパティの名前がこのフィールドに取り込まれます。InqPropOpts 構造体中の *ReturnedName* の *VSBufSize* フィールドが、戻されるプロパティ名の長さ未満の場合は、理由 MQRC_PROPERTY_NAME_TOO_BIG で完了コードが MQCC_FAILED に設定されます。

既知の同義語のあるプロパティは以下のように戻されます。

1. 接頭辞「mqps」を持つプロパティ プロパティ IBM MQ 名として返されます。例えば、戻される名前は、「mqps.Top」ではなく「MQTopicString」です。
2. 接頭辞「jms。」を持つプロパティ または「mcd。」ヘッダーフィールド JMS 名として返されます。例えば、「jms.Exp」ではなく「JMSExpiration」が返される名前となります。

3. 接頭辞「usr.」を持つプロパティ その接頭辞なしで返されます。例えば、「usr.Color」ではなく「Color」が返されます。

同義語のあるプロパティは 1 回のみ戻されます。

C プログラミング言語では、以下のマクロ変数が、すべてのプロパティを照会するために定義され、次に「usr.」を開始するすべてのプロパティが定義されます。

MQPROP_INQUIRE_ALL

メッセージのすべてのプロパティに対する照会。

MQPROP_INQUIRE_ALL は、以下のようにして使用できます。

```
MQCHARV Name = {MQPROP_INQUIRE_ALL};
```

MQPROP_INQUIRE_ALL_USR

先頭が「usr.」の、メッセージのすべてのプロパティに対する照会。返される名前は「usr.」を除いて返されます 接頭部。

MQIMP_INQ_NEXT が指定されていて、前回の呼び出し以降名前が変更されているか、または初回の呼び出しである場合には、MQIMPO_INQ_FIRST が暗黙指定されます。

プロパティ名の使用については、[プロパティ名およびプロパティ名に関する制約事項](#)を参照してください。

PropDesc

タイプ: MQPD - 出力

この構造体を使用して、プロパティの属性を定義します。その中には、プロパティがサポートされていない場合に起きること、プロパティが属するメッセージ・コンテキスト、およびプロパティのコピー先のメッセージが含まれます。この構造体の詳細については、[MQPD](#) を参照してください。

タイプ

タイプ: MQLONG - 入出力

MQINQMP 呼び出しから戻される際に、このパラメーターは *Value* のデータ・タイプに設定されます。データ・タイプは次のいずれかです。

MQTYPE_BOOLEAN

ブール値。

MQTYPE_BYTE_STRING

バイト・ストリング。

MQTYPE_INT8

8 ビットの符号付き整数。

MQTYPE_INT16

16 ビットの符号付き整数。

MQTYPE_INT32

32 ビットの符号付き整数。

MQTYPE_INT64

64 ビットの符号付き整数。

MQTYPE_FLOAT32

32 ビットの浮動小数点数。

MQTYPE_FLOAT64

64 ビットの浮動小数点数。

MQTYPE_STRING

文字ストリング。

MQTYPE_NULL

プロパティは存在しますがヌル値です。

プロパティ値のデータ・タイプが認識されない場合は、MQTYPE_STRING が戻され、値のストリング表現が *Value* 域に入れられます。データ・タイプのストリング表現は、*InqPropOpts* パラメーターの

TypeString フィールドにあります。理由 MQRC_PROP_TYPE_NOT_SUPPORTED で、警告の完了コードが戻されます。

さらに、オプション MQIMPO_CONVERT_TYPE を指定すると、プロパティ値の変換が要求されます。プロパティを戻す際のデータ・タイプを指定するには、*Type* を入力として使用します。データ型の変換の詳細については、[MQIMPO 構造体の MQIMPO_CONVERT_TYPE オプションの説明](#)を参照してください。

タイプ変換を要求しない場合、入力で以下の値を使用できます。

MQTYPE_AS_SET

プロパティの値は、そのデータ・タイプを変換せずに戻されます。

ValueLength

タイプ: MQLONG - 入力

Value 域のバイト単位の長さ。値を戻す必要のないプロパティの場合は、ゼロを指定します。これらのプロパティには、アプリケーションによってヌル値または空ストリングを持つように設計されているものがあります。[MQIMPO_QUERY_LENGTH](#) オプションが指定されている場合もゼロを指定します。この場合、値は戻されません。

値

タイプ: MQBYTE*xValueLength* - 出力

これは、照会プロパティ値を含む領域です。バッファは、戻される値に適した境界に位置合わせされなければなりません。この処理に失敗すると、後で値にアクセスする際にエラーが発生する可能性があります。

ValueLength がプロパティ値の長さ未満の場合は、可能な限り多くのプロパティ値が *Value* に移動され、呼び出しは完了コード MQCC_FAILED および理由 MQRC_PROPERTY_VALUE_TOO_BIG で失敗します。

Value 中のデータの文字セットは、InqPropOpts パラメーター中の ReturnedCCSID フィールドで指定されます。*Value* 中のデータのエンコード方式は、InqPropOpts パラメーター中の ReturnedEncoding フィールドで指定されます。

C プログラミング言語では、パラメーターは、void を示すポインターとして宣言されます。つまり、どのタイプのデータのアドレスもパラメーターとして指定できます。

ValueLength パラメーターがゼロの場合は、*Value* は参照されず、C または System/390 アセンブラーで作成されたプログラムによって渡されるこの値はヌルのこともあります。

DataLength

タイプ: MQLONG - 出力

これは、*Value* 域に返される実際のプロパティ値の長さ (バイト数) です。

DataLength がプロパティ値の長さより小さい場合も、MQINQMP の呼び出しから戻った時点で *DataLength* にはデータが入れられます。これにより、アプリケーションは、プロパティ値を入れるのに必要なバッファのサイズを判別して、適切なサイズのバッファを用いて呼び出しを再発行することができます。

以下の値も戻されることがあります。

Type パラメーターが MQTYPE_STRING または MQTYPE_BYTE_STRING に設定されている場合:

MQVL_EMPTY_STRING

プロパティは存在しますが、文字やバイトが含まれていません。

CompCode

タイプ: MQLONG - 出力

完了コード。以下のいずれかです。

MQCC_OK

正常終了。

MQCC_WARNING
警告 (部分完了)。

MQCC_FAILED
呼び出し失敗。

理由

タイプ: MQLONG - 出力

CompCode が MQCC_OK の場合:

MQRC_NONE
(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_WARNING の場合:

MQRC_PROP_NAME_NOT_CONVERTED
(2492, X'09BC') 戻されたプロパティ名が変換されなかった。

MQRC_PROP_VALUE_NOT_CONVERTED
(2466, X'09A2') プロパティ値が変換されなかった。

MQRC_PROP_TYPE_NOT_SUPPORTED
(2467, X'09A3') プロパティのデータ・タイプがサポートされていない。

MQRC_RFH_FORMAT_ERROR
(2421, X'0975') プロパティを含む MQRFH2 フォルダーを構文解析できなかった。

CompCode が MQCC_FAILED の場合:

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'089C') アダプターが利用できません。

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'0852') アダプター・サービス・モジュールをロードできない。

MQRC_ASID_MISMATCH
(2157, X'086D') 1 次 ASID とホーム ASID とが異なっている。

MQRC_BUFFER_ERROR
(2004, X'07D4') 値パラメーターが無効である。

MQRC_BUFFER_LENGTH_ERROR
(2005, X'07D5') 値長パラメーターが無効である。

MQRC_CALL_IN_PROGRESS
(2219, X'08AB') 前の呼び出しが完了する前に MQI 呼び出しが入力された。

MQRC_CONNECTION_BROKEN
(2009, X'07D9') キュー・マネージャーとの接続が失われました。

MQRC_DATA_LENGTH_ERROR
(2010, X'07DA') データ長パラメーターが無効である。

MQRC_IMPO_ERROR
(2464, X'09A0') メッセージ・プロパティ照会オプションの構造体が無効である。

MQRC_HMSG_ERROR
(2460, X'099C') メッセージ・ハンドルが無効。

MQRC_MSG_HANDLE_IN_USE
(2499, X'09C3') メッセージ・ハンドルがすでに使用中。

MQRC_OPTIONS_ERROR
(2046, X'07F8') オプションが無効、または整合性がない。

MQRC_PD_ERROR
(2482, X'09B2') プロパティ記述子の構造体が無効である。

MQRC_PROP_CONV_NOT_SUPPORTED
(2470, X'09A6') 実際のデータ・タイプから要求されたデータ・タイプへの変換がサポートされていない。

MQRC_PROPERTY_NAME_ERROR

(2442, X'098A') プロパティ名が無効である。

MQRC_PROPERTY_NAME_TOO_BIG

(2465, X'09A1') 戻される名前バッファーにとってプロパティ名が大きすぎる。

MQRC_PROPERTY_NOT_AVAILABLE

(2471, X'09A7') プロパティが使用できない。

MQRC_PROPERTY_VALUE_TOO_BIG

(2469, X'09A5') Value 域にとってプロパティ値が大きすぎる。

MQRC_PROP_NUMBER_FORMAT_ERROR

(2472, X'09A8') 値データ中に数字フォーマット・エラーが発生した。

MQRC_PROPERTY_TYPE_ERROR

(2473, X'09A9') 要求されたプロパティ・タイプが無効である。

MQRC_SOURCE_CCSID_ERROR

(2111, X'083F') プロパティ名エンコード文字セット ID が無効である。

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'0871') 使用できるストレージが十分でない。

MQRC_UNEXPECTED_ERROR

(2195, X'0893') 予期しないエラーが発生した。

これらのコードについて詳しくは、[メッセージおよび理由コード](#)を参照してください。

C 言語での呼び出し

```
MQINQMP (Hconn, Hmsg, &InqPropOpts, &Name, &PropDesc, &Type,
ValueLength, Value, &DataLength, &CompCode, &Reason);
```

パラメーターを次のように宣言します。

```
MQHCONN Hconn;          /* Connection handle */
MQHMSG Hmsg;             /* Message handle */
MQIMPO InqPropOpts;      /* Options that control the action of MQINQMP */
MQCHARV Name;           /* Property name */
MQPD PropDesc;          /* Property descriptor */
MQLONG Type;            /* Property data type */
MQLONG ValueLength;      /* Length in bytes of the Value area */
MQBYTE Value[n];        /* Area to contain the property value */
MQLONG DataLength;       /* Length of the property value */
MQLONG CompCode;         /* Completion code */
MQLONG Reason;          /* Reason code qualifying CompCode */
```

COBOL での呼び出し

```
CALL 'MQINQMP' USING HCONN, HMSG, INQMSGOPTS, NAME, PROPDSC, TYPE,
VALUELENGTH, VALUE, DATALENGTH, COMPCODE, REASON.
```

パラメーターを次のように宣言します。

```
** Connection handle
01 HCONN          PIC S9(9) BINARY.
** Message handle
01 HMSG           PIC S9(18) BINARY.
** Options that control the action of MQINQMP
01 INQMSGOPTS.
   COPY CMQIMPOV.
** Property name
01 NAME.
   COPY CMQCHRVV.
** Property descriptor
01 PROPDSC.
   COPY CMQPDV.
** Property data type
01 TYPE          PIC S9(9) BINARY.
** Length in bytes of the VALUE area
01 VALUELENGTH PIC S9(9) BINARY.
```

```

** Area to contain the property value
01 VALUE          PIC X(n).
** Length of the property value
01 DATALENGTH    PIC S9(9) BINARY.
** Completion code
01 COMPCODE        PIC S9(9) BINARY.
** Reason code qualifying COMPCODE
01 REASON          PIC S9(9) BINARY.

```

PL/I での呼び出し

```
call MQINQMP (Hconn, Hmsg, InqPropOpts, Name, PropDesc, Type,
ValueLength, Value, DataLength, CompCode, Reason);
```

パラメーターを次のように宣言します。

```

dcl Hconn      fixed bin(31); /* Connection handle */
dcl Hmsg       fixed bin(63); /* Message handle */
dcl InqPropOpts like MQIMPO; /* Options that control the action of MQINQMP */
dcl Name       like MQCHARV; /* Property name */
dcl PropDesc   like MQPD; /* Property descriptor */
dcl Type       fixed bin (31); /* Property data type */
dcl ValueLength fixed bin (31); /* Length in bytes of the Value area */
dcl Value      char (n); /* Area to contain the property value */
dcl DataLength fixed bin (31); /* Length of the property value */
dcl CompCode   fixed bin (31); /* Completion code */
dcl Reason     fixed bin (31); /* Reason code qualifying CompCode */

```

高水準アセンブラー呼び出し

```
CALL MQINQMP, (HCONN,HMSG,INQMSGOPTS,NAME,PROPDSC,TYPE,
VALUELENGTH,VALUE,DATALength,COMPCODE,REASON)
```

パラメーターを次のように宣言します。

HCONN	DS	F	Connection handle
HMSG	DS	D	Message handle
INQMSGOPTS	CMQIMPOA	,	Options that control the action of MQINQMP
NAME	CMQCHRVA	,	Property name
PROPDSC	CMQPDA	,	Property descriptor
TYPE	DS	F	Property data type
VALUELENGTH	DS	F	Length in bytes of the VALUE area
VALUE	DS	CL(n)	Area to contain the property value
DATALength	DS	F	Length of the property value
COMPCODE	DS	F	Completion code
REASON	DS	F	Reason code qualifying COMPCODE

MQMHBUF - メッセージ・ハンドルのバッファーへの変換

MQMHBUF 呼び出しはメッセージ・ハンドルをバッファーに変換するので、MQBUFMH 呼び出しの逆です。

構文

```
MQMHBUF (Hconn, Hmsg, MsgHBufOpts, Name, MsgDesc, BufferLength, Buffer, DataLength, CompCode, Reason)
```

パラメーター

Hconn

タイプ: MQHCONN - 入力

このハンドルは、キュー・マネージャーに対する接続を表します。Hconn の値は、Hmsg パラメーターで指定されているメッセージ・ハンドルを作成するために使用された接続ハンドルと一致していなければなりません。

MQHC_UNASSOCIATED_HCONN を使用してメッセージ・ハンドルが作成された場合は、メッセージ・ハンドルを削除するスレッド上で有効な接続を確立しなければなりません。有効な接続を確立しないと、呼び出しは MQRC_CONNECTION_BROKEN で失敗します。

Hmsg

タイプ: MQHMSG - 入力

これは、バッファが必要なメッセージ・ハンドルです。値は、前の MQCRTMH 呼び出しで戻されたものです。

MsgHBufOpts

タイプ: MQMHBO - 入力

アプリケーションでは、MQMHBO 構造体を使用することによって、メッセージ・ハンドルからバッファを生成する方法を制御するためのオプションを指定することができます。

詳細は 446 ページの『[MQMHBO - メッセージ・ハンドルからバッファへの変換オプション](#)』を参照してください。

名前

タイプ: MQCHARV - 入力

バッファに書き込む 1 つ以上のプロパティの名前。

この名前と一致するプロパティを検出できない場合は、呼び出しは MQRC_PROPERTY_NOT_AVAILABLE で失敗します。

ワイルドカードを使用して、複数のプロパティをバッファに書き込むことができます。そのためには、プロパティ名の末尾にワイルドカード文字「%」を使用します。このワイルドカードは、「!」を含む 0 個以上の文字に一致します 文字を含むことはできません。

C 言語では、すべてのプロパティに対する照会用、および先頭が「usr」のすべてのプロパティに対する照会用に以下のマクロ変数が定義されます。

MQPROP_INQUIRE_ALL

メッセージのすべてのプロパティをバッファに書き込みます

MQPROP_INQUIRE_ALL_USR

メッセージのすべてのプロパティで、文字列 'usr' で始まるものをすべてバッファに。

プロパティ名の使用については、[プロパティ名およびプロパティ名に関する制約事項](#)を参照してください。

MsgDesc

タイプ: MQMD - 入出力

MsgDesc 構造体は、バッファ領域の内容を記述します。

出力上では、バッファ領域のエンコード方式、文字セット ID、およびデータの形式を、呼び出しによって書き込まれるとおりに正しく記述するように、*Encoding*、*CodedCharSetId*、および *Format* フィールドが設定されます。

この構造体中のデータは、アプリケーションの文字セット内およびエンコード内にあります。

BufferLength

タイプ: MQLONG - 入力

BufferLength は、バッファ領域の長さです (バイト単位)。

Buffer

タイプ: MQBYTEExBufferLength - 出力

Buffer は、メッセージ・プロパティが入るように領域を定義します。バッファを 4 バイトの境界に位置合わせする必要があります。

BufferLength が、*Buffer* 中のプロパティを格納するのに必要な長さ未満の場合は、MQMHBUF は MQRC_PROPERTY_VALUE_TOO_BIG で失敗します。

呼び出しに失敗した場合でも、バッファの内容が変わることがあります。

DataLength

タイプ: MQLONG - 出力

DataLength は、バッファーに入れて戻されるプロパティの長さです (バイト単位)。この値がゼロの場合は、*Name* で指定された値と一致したプロパティはなく、呼び出しは理由コード *MQRC_PROPERTY_NOT_AVAILABLE* で失敗します。

BufferLength が、バッファー中のプロパティを格納するのに必要な長さ未満の場合は、*MQMHBUF* は *MQRC_PROPERTY_VALUE_TOO_BIG* で失敗しますが、それでも値は *DataLength* に入力されます。これにより、アプリケーションは、プロパティを収容するのに必要なバッファーのサイズを判別して、必要な *BufferLength* を用いて呼び出しを再発行することができます。

CompCode

タイプ: MQLONG - 出力

完了コード。以下のいずれかです。

MQCC_OK

正常終了。

MQCC_FAILED

呼び出し失敗。

理由

タイプ: MQLONG - 出力

CompCode を限定する理由コード。

CompCode が *MQCC_OK* の場合:

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が *MQCC_FAILED* の場合:

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'089C') アダプターが利用できません。

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852') アダプター・サービス・モジュールをロードできません。

MQRC_ASID_MISMATCH

(2157, X'86D') 1 次 ASID とホーム ASID が異なります。

MQRC_MHBO_ERROR

(2501, X'095C') メッセージ・ハンドルからバッファーへの変換オプション構造体が無効です。

MQRC_BUFFER_ERROR

(2004, X'07D4') バッファー・パラメーターが無効である。

MQRC_BUFFER_LENGTH_ERROR

(2005, X'07D5') バッファー長パラメーターは無効です。

MQRC_CALL_IN_PROGRESS

(2219, X'08AB') 前の呼び出しが完了する前に MQI 呼び出しが入力された。

MQRC_CONNECTION_BROKEN

(2009, X'07D9') キュー・マネージャーとの接続が失われました。

MQRC_DATA_LENGTH_ERROR

(2010, X'07DA') データ長パラメーターが無効である。

MQRC_HMSG_ERROR

(2460, X'099C') メッセージ・ハンドルが無効。

MQRC_MD_ERROR

(2026, X'07EA') メッセージ記述子が無効である。

MQRC_MSG_HANDLE_IN_USE

(2499, X'09C3') メッセージ・ハンドルがすでに使用中。

MQRC_OPTIONS_ERROR

(2046, X'07FE') オプションが無効であるか、矛盾しています。

MQRC_PROPERTY_NAME_ERROR

(2442, X'098A') プロパティ名が無効である。

MQRC_PROPERTY_NOT_AVAILABLE

(2471, X'09A7') プロパティが使用できない。

MQRC_PROPERTY_VALUE_TOO_BIG

(2469, X'09A5') BufferLength の値が小さすぎるため、指定されたプロパティを入れることができません。

MQRC_UNEXPECTED_ERROR

(2195, X'893') 予期しないエラーが発生しました。

これらのコードについて詳しくは、[メッセージおよび理由コード](#)を参照してください。

C 言語での呼び出し

```
MQMHBUF (Hconn, Hmsg, &MsgHBufOpts, &Name, &MsgDesc, BufferLength, Buffer,  
&DataLength, &CompCode, &Reason);
```

パラメーターを次のように宣言します。

```
MQHCONN Hconn;          /* Connection handle */  
MQHMSG Hmsg;            /* Message handle */  
MQMHBO MsgHBufOpts;     /* Options that control the action of MQMHBUF */  
MQCHARV Name;           /* Property name */  
MQMD MsgDesc;           /* Message descriptor */  
MQLONG BufferLength;     /* Length in bytes of the Buffer area */  
MQBYTE Buffer[n];        /* Area to contain the properties */  
MQLONG DataLength;       /* Length of the properties */  
MQLONG CompCode;         /* Completion code */  
MQLONG Reason;          /* Reason code qualifying CompCode */
```

使用上の注意

MQMHBUF はメッセージ・ハンドルをバッファーに変換します。

MQGET API 出口と併用して、メッセージ・プロパティ API を使って特定のプロパティにアクセスしてから、これらのプロパティをバッファー中に渡して、メッセージ・ハンドルではなく MQRFH2 ヘッダーを使用するように設計されているアプリケーションに戻します。

この呼び出しは MQBUFMH 呼び出しの逆です。MQBUFMH 呼び出しを使用すると、バッファーからメッセージ・ハンドルにメッセージ・プロパティを構文解析できます。

COBOL での呼び出し

```
CALL 'MQMHBUF' USING HCONN, HMSG, MSGHBUFFOPTS, NAME, MSGDESC,  
                     BUFFERLENGTH, BUFFER, DATALENGTH, COMPCODE, REASON.
```

パラメーターを次のように宣言します。

```
** Connection handle  
01 HCONN          PIC S9(9) BINARY.  
** Message handle  
01 HMSG           PIC S9(18) BINARY.  
** Options that control the action of MQMHBUF  
01 MSGHBUFFOPTS.  
   COPY CMQMHBV.  
** Property name  
01 NAME           PIC S9(18) BINARY.  
   COPY CMQCHRVV.  
** Message descriptor  
01 MSGDESC        PIC S9(18) BINARY.  
   COPY CMQMDV.  
** Length in bytes of the Buffer area */  
01 BUFFERLENGTH  PIC S9(9) BINARY.
```

```

** Area to contain the properties
01 BUFFER          PIC X(n).
** Length of the properties
01 DATALENGTH     PIC S9(9) BINARY.
** Completion code
01 COMPCODE        PIC S9(9) BINARY.
** Reason code qualifying COMPCODE
01 REASON          PIC S9(9) BINARY.

```

PL/I での呼び出し

```
call MQMHBUFF (Hconn, Hmsg, MsgHBufOpts, Name, MsgDesc, BufferLength, Buffer,
DataLength, CompCode, Reason);
```

パラメーターを次のように宣言します。

```

dcl Hconn          fixed bin(31); /* Connection handle */
dcl Hmsg           fixed bin(63); /* Message handle */
dcl MsgHBufOpts    like MQMHB0; /* Options that control the action of MQMHBUFF */
dcl Name           like MQCHARV; /* Property name */
dcl MsgDesc        like MQMD; /* Message descriptor */
dcl BufferLength    fixed bin(31); /* Length in bytes of the Buffer area */
dcl Buffer          char(n); /* Area to contain the properties */
dcl DataLength      fixed bin(31); /* Length of the properties */
dcl CompCode       fixed bin(31); /* Completion code */
dcl Reason         fixed bin(31); /* Reason code qualifying CompCode */

```

高水準アセンブラー呼び出し

```
CALL MQMHBUFF, (HCONN,HMSG,MSGHBUFOPTS,NAME,MSGDESC,BUFFERLENGTH,
BUFFER,DATALENGTH,COMPCODE,REASON)
```

パラメーターを次のように宣言します。

HCONN	DS	F	Connection handle
HMSG	DS	D	Message handle
MSGHBUFOPTS	CMQMHB0A	,	Options that control the action of MQMHBUFF
NAME	CMQCHRVA	,	Property name
MSGDESC	CMQMDA	,	Message descriptor
BUFFERLENGTH	DS	F	Length in bytes of the BUFFER area
BUFFER	DS	CL(n)	Area to contain the properties
DATALENGTH	DS	F	Length of the properties
COMPCODE	DS	F	Completion code
REASON	DS	F	Reason code qualifying COMPCODE

MQOPEN - オブジェクトのオープン

MQOPEN の呼び出しはオブジェクトへのアクセスを確立します。

次のタイプのオブジェクトが有効です。

- キュー (配布リストを含む)
- 名前リスト
- プロセス定義
- キュー・マネージャー
- トピック

構文

```
MQOPEN (Hconn, ObjDesc, Options, Hobj, CompCode, Reason)
```

パラメーター

Hconn

タイプ: MQHCONN - 入力

このハンドルは、キュー・マネージャーに対する接続を表します。Hconn の値は、先行の MQCONN または MQCONNX 呼び出しによって戻されたものです。

z/OS z/OS 上で CICS アプリケーションを使用する場合は、MQCONN 呼び出しを省略でき、Hconn に次の値を指定できます。

MQHC_DEF_HCONN

デフォルトの接続ハンドル。

ObjDesc

タイプ: MQOD - 入出力

これは、開くオブジェクトを識別する構造です。詳細については、[449 ページの『MQOD - オブジェクト記述子』](#)を参照してください。

パラメータ **ObjDesc** 内のフィールド **ObjectName** がモデルキューの名前である場合、動的ローカルキューは、モデル・キューの属性を使用して作成されます。これは、**Options** パラメーターで指定するオプションに関係なく起こります。MQOPEN 呼び出しによって返される Hobj を使用する後続の操作は、モデル・キューではなく新しい動的キューで行われます。MQINQ 呼び出しおよび MQSET 呼び出しの場合でも同じです。**ObjDesc** パラメーターのモデル・キューの名前は、作成された動的キューの名前で置き換えられます。動的キューのタイプは、モデル・キューの **DefinitionType** 属性の値によって決まります ([830 ページの『キューの属性』](#)を参照してください)。動的キューに適用されるクローズ・オプションの詳細については、MQCLOSE 呼び出しの記述を参照してください。

オプション

タイプ: MQLONG - 入力

以下のオプションのうち少なくとも 1 つを指定する必要があります。

- MQOO_BROWSE
- MQOO_INPUT_* (このうちの 1 つだけ)
- MQOO_INQUIRE
- MQOO_OUTPUT
- MQOO_SET
- MQOO_BIND_* (このうちの 1 つだけ)

これらのオプションの詳細については、以下の表を参照してください。その他のオプションは必要に応じて指定することができます。複数のオプションを指定するには、値を加算する (同じ定数を複数回加算しない) か、ビット単位 OR 演算を使用して値を結合 (プログラミング言語でビット演算がサポートされる場合) します。有効でない組み合わせについては、注記されています。他のすべての組み合わせは有効です。ObjDesc によって指定されたオブジェクトのタイプに適用されるオプションだけが許可されます。

表 589. キューおよびトピックに有効な MQOPEN オプション						
オプション	別名 ¹	ローカルおよびモデル	リモート	非ローカル・クラスター	配布リスト	トピック
MQOO_INPUT_AS_Q_DEF	はい	はい	いいえ	いいえ	いいえ	いいえ
MQOO_INPUT_SHARED	はい	はい	いいえ	いいえ	いいえ	いいえ
MQOO_INPUT_EXCLUSIVE	はい	はい	いいえ	いいえ	いいえ	いいえ
MQOO_OUTPUT	はい	はい	はい	はい	はい	はい
MQOO_BROWSE	はい	はい	いいえ	いいえ	いいえ	いいえ
MQOO_CO_OP	はい	はい	いいえ	いいえ	いいえ	いいえ

表 589. キューおよびトピックに有効な MQOPEN オプション (続き)

オプション	別名 ¹	ローカルおよびモデル	リモート	非ローカル・クラスター	配布リスト	トピック
<u>MQOO_INQUIRE</u>	はい	はい	<u>2</u>	はい	いいえ	いいえ
<u>MQOO_SET</u>	はい	はい	<u>2</u>	いいえ	いいえ	いいえ
<u>MQOO_BIND_ON_OPEN</u> ³	はい	はい	はい	はい	はい	いいえ
<u>MQOO_BIND_NOT_FIXED</u> ³	はい	はい	はい	はい	はい	いいえ
<u>MQOO_BIND_ON_GROUP</u> ³	はい	はい	はい	はい	はい	いいえ
<u>MQOO_BIND_AS_Q_DEF</u> ³	はい	はい	はい	はい	はい	いいえ
<u>MQOO_SAVE_ALL_CONTEXT</u>	はい	はい	いいえ	いいえ	いいえ	いいえ
<u>MQOO_PASS_IDENTITY_CONTEXT</u>	はい	はい	はい	はい	はい	<u>4</u>
<u>MQOO_PASS_ALL_CONTEXT</u>	はい	はい	はい	はい	はい	はい
<u>MQOO_SET_IDENTITY_CONTEXT</u>	はい	はい	はい	はい	はい	<u>4</u>
<u>MQOO_SET_ALL_CONTEXT</u>	はい	はい	はい	はい	はい	はい
<u>MQOO_NO_READ_AHEAD</u>	はい	はい	いいえ	いいえ	いいえ	いいえ
<u>MQOO_READ_AHEAD</u>	はい	はい	いいえ	いいえ	いいえ	いいえ
<u>MQOO_READ_AHEAD_AS_Q_DEF</u>	はい	はい	いいえ	いいえ	いいえ	いいえ
<u>MQOO_ALTERNATE_USER_AUTHORITY</u>	はい	はい	はい	はい	はい	はい
<u>MQOO_FAIL_IF_QUIESCING</u>	はい	はい	はい	はい	はい	はい
<u>MQOO_RESOLVE_LOCAL_Q</u>	はい	はい	はい	はい	いいえ	いいえ
<u>MQOO_RESOLVE_LOCAL_TOPIC</u>	いいえ	いいえ	いいえ	いいえ	いいえ	はい
<u>MQOO_NO_MULTICAST</u>	いいえ	いいえ	いいえ	いいえ	いいえ	はい

注:

1. 別名のオプションの妥当性は、その別名が解決されるキューのオプションの妥当性に応じて決められます。
2. このオプションは、リモート・キューのローカル定義の場合にのみ有効です。
3. このオプションは、どのタイプのキューについても指定できますが、そのキューがクラスター・キューでない場合は無視されます。ただし **DefBind** キュー属性は、別名キューがクラスター内にない場合でも基本キューをオーバーライドします。
4. これらの属性はトピックで使用できますが、サブスクライバーに送信されるコンテキスト・フィールドではなく、保持されるメッセージに設定されたコンテキストにのみ影響します。

アクセス・オプション: 以下のオプションは、オブジェクトに対して実行できる操作のタイプを制御します。

MQOO_INPUT_AS_Q_DEF

キュー定義のデフォルトを使用してメッセージを取得するためにキューを開きます。

後続の MQGET 呼び出しで使用するために、キューが開かれます。アクセスのタイプは、**DefInputOpenOption** キュー属性の値に応じて、共用または排他のいずれかになります。詳細については、830 ページの『キューの属性』を参照してください。

このオプションは、ローカル・キュー、別名キュー、およびモデル・キューに関してのみ有効です。リモート・キューや配布リスト、さらにキューでないオブジェクトに関しては無効です。

MQOO_INPUT_SHARED

共有アクセスによりメッセージを読み取るためにキューをオープンする。

後続の MQGET 呼び出しで使用するために、キューが開かれます。このアプリケーションまたは別のアプリケーションによって、キューが MQOO_INPUT_SHARED で現在オープンされている場合は、この呼び出しは正常に行われますが、キューが MQOO_INPUT_EXCLUSIVE でオープンされている場合は失敗し、理由コード MQRC_OBJECT_IN_USE が戻ります。

このオプションは、ローカル・キュー、別名キュー、およびモデル・キューに関してのみ有効です。リモート・キューや配布リスト、さらにキューでないオブジェクトに関しては無効です。

MQOO_INPUT_EXCLUSIVE

メッセージを読み取るためにキューを排他アクセス・モードでオープンする。

後続の MQGET 呼び出しで使用するために、キューが開かれます。このアプリケーションまたは別のアプリケーションによって、キューがいずれかのタイプ (MQOO_INPUT_SHARED または MQOO_INPUT_EXCLUSIVE) の入力用に現在開かれている場合、この呼び出しは失敗し、理由コード MQRC_OBJECT_IN_USE が戻ります。

このオプションは、ローカル・キュー、別名キュー、およびモデル・キューに関してのみ有効です。リモート・キューや配布リスト、さらにキューでないオブジェクトに関しては無効です。

MQOO_OUTPUT

キューを開いてメッセージを書き込んだり、トピックまたはトピック・ストリングを開いてメッセージを公開したりします。

後続の MQPUT 呼び出しで使用するために、キューまたはトピックが開かれます。

このオプションを使用して MQOPEN 呼び出しを行うと、**InhibitPut** キュー属性が MQQA_PUT_INHIBITED に設定されていても成功します (ただし、属性がこの値に設定されていると、後続の MQPUT 呼び出しは失敗します)。

このオプションは、配布リストをはじめ、すべてのタイプのキューおよびトピックで有効です。

以下の注は、次のオプションに適用されます。

- これらのオプションは、1 つだけ指定できます。
- これらのオプションのいずれかを指定した MQOPEN 呼び出しは、キュー **InhibitGet** 属性が MQQA_GET_INHIBITED に設定されている場合でも成功することがあります (ただし、この値が設定されている間は、その後の MQGET 呼び出しは失敗します)。
- キューが共有可能でないものとして定義されている場合は (つまり、**Shareability** キュー属性の値が MQQA_NOT_SHAREABLE)、共有アクセスでそのキューを開こうとしても、排他アクセスによるものとして扱われます。
- 別名キューがこれらのオプションのいずれかで開かれている場合は、排他的使用をしているかどうか (または別のアプリケーションで排他的使用をしているか) のテストが、別名解決先の基本キューに対して行われます。
- これらのオプションは、**ObjectQMgrName** がキュー・マネージャーの別名であるときには無効になります。キュー・マネージャーの別名として使用されるリモート・キューのローカル定義の中の **RemoteQMgrName** 属性の値が、ローカル・キュー・マネージャーの名前である場合も同様です。

MQOO_BROWSE

メッセージをブラウズするためにキューを開きます。

以下のいずれかのオプションを使用する後続の MQGET 呼び出しで使用するために、キューが開かれます。

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_NEXT
- MQGMO_BROWSE_MSG_UNDER_CURSOR

これは、キューが現在 MQOO_INPUT_EXCLUSIVE で開かれている場合でも使用できます。MQOO_BROWSE オプションを指定して MQOPEN 呼び出しを行うと、ブラウズ・カーソルが設定され、論理的にはキューにある最初のメッセージの前にそのブラウズ・カーソルが置かれます。詳細については、[MQGMO - Options](#) フィールドを参照してください。

このオプションは、ローカル・キュー、別名キュー、およびモデル・キューに関してのみ有効です。リモート・キューや配布リスト、さらにキューでないオブジェクトに関しては無効です。ObjectQMgrName がキュー・マネージャーの別名であるときにもこれは無効になります。キュー・マネージャーの別名として使用されるリモート・キューのローカル定義の中の **RemoteQMgrName** 属性の値が、ローカル・キュー・マネージャーの名前である場合も同様です。

MQOO_CO_OP

連携するハンドル・セットのメンバーとして開きます。

このオプションは、MQOO_BROWSE オプションの場合にのみ有効です。MQOO_BROWSE を使用せずに指定した場合、MQOPEN の実行結果として MQRC_OPTIONS_ERROR が返されます。

返されるハンドルは、後続の MQGET 呼び出しにおける連携するハンドル・セットのメンバーであると見なされ、以下のいずれかのオプションがあります。

- MQGMO_MARK_BROWSE_CO_OP
- MQGMO_UNMARKED_BROWSE_MSG
- MQGMO_UNMARK_BROWSE_CO_OP

このオプションは、ローカル・キュー、別名キュー、およびモデル・キューに関してのみ有効です。リモート・キューや配布リスト、さらにキューでないオブジェクトに関しては無効です。

MQOO_INQUIRE

属性を照会するためにオブジェクトを開きます。

後続の MQINQ 呼び出しで使用するために、キュー、名前リスト、プロセス定義、またはキュー・マネージャーが開かれます。

このオプションは、配布リスト以外のすべてのタイプのオブジェクトで有効です。ObjectQMgrName がキュー・マネージャーの別名であるときにはこれは無効になります。キュー・マネージャーの別名として使用されるリモート・キューのローカル定義の中の **RemoteQMgrName** 属性の値が、ローカル・キュー・マネージャーの名前である場合も同様です。

MQOO_SET

属性を設定するためにキューを開きます。

後続の MQSET 呼び出しで使用するために、キューが開かれます。

このオプションは、配布リスト以外のすべてのタイプのキューで有効です。ObjectQMgrName がリモート・キューのローカル定義名であるときにはこれは無効になります。キュー・マネージャーの別名として使用されるリモート・キューのローカル定義の中の **RemoteQMgrName** 属性の値が、ローカル・キュー・マネージャーの名前である場合も同様です。

バインディング・オプション: 以下のオプションは、開かれるオブジェクトがクラスター・キューである場合に適用されます。これらのオプションは、クラスター・キューのインスタンスへのキュー・ハンドルのバインディングを制御します。

MQOO_BIND_ON_OPEN

キューが開いたときに、ローカル・キュー・マネージャーが、キュー・ハンドルを宛先キューのインスタンスにバインドします。その結果、このハンドルを使って書き込まれるすべてのメッセージが、宛先キューの同じインスタンスに、同じ経路で送信されます。

このオプションは、キューの場合にのみ有効であり、クラスター・キューにのみ影響します。クラスター・キューではないキューに対して指定された場合、このオプションは無視されます。

MQOO_BIND_NOT_FIXED

このオプションを指定すると、ローカル・キュー・マネージャーは、宛先キューのインスタンスへのキュー・ハンドルのバインドを停止します。その結果、このハンドルを使用するその後の MQPUT 呼び出しでは、メッセージは、その宛先キューのさまざまなインスタンスに送信されたり、同じインスタンスに送信されてもさまざまな経路を経由したりします。また、このオプションを使用すると、選択されたインスタンスを、ネットワーク条件に従って、ローカル・キュー・マネージャー、リモート・キュー・マネージャー、またはメッセージ・チャネル・エージェント (MCA) で後で変更することもできます。

注: トランザクションを完了するために一まとまりのメッセージを交換する必要のあるクライアント・アプリケーションおよびサーバー・アプリケーションでは、MQOO_BIND_NOT_FIXED (DefBind) の値が MQBND_BIND_NOT_FIXED である場合は MQOO_BIND_AS_Q_DEF) を使用しないでください。これを使用すると、その後の一まとまりのメッセージがサーバー・アプリケーションのさまざまなインスタンスに送信されるおそれがあります。

クラスター・キューに MQOO_BROWSE オプションまたはいずれかの MQOO_INPUT_* オプションを指定すると、キュー・マネージャーでは必ずそのクラスター・キューのローカル・インスタンスが選択されます。その結果、MQOO_BIND_NOT_FIXED が指定されている場合でも、キュー・ハンドルのバインディングは固定されます。

通常は、すべてのインスタンスが同じ属性値を持っていますが、MQOO_BIND_NOT_FIXED と MQOO_INQUIRE を組み合わせて指定した場合、そのハンドルを使用するその後の MQINQ 呼び出しでは、クラスター・キューのさまざまなインスタンスが照会される可能性があります。

MQOO_BIND_NOT_FIXED はキューの場合にのみ有効であり、クラスター・キューにのみ影響します。クラスター・キューではないキューに対して指定された場合、このオプションは無視されます。

MQOO_BIND_ON_GROUP

グループ内のメッセージすべてを同じ宛先のインスタンスに割り振る要求をアプリケーションが行えるようになります。

このオプションは、キューの場合にのみ有効であり、クラスター・キューにのみ影響します。クラスター・キューではないキューに対して指定された場合、このオプションは無視されます。

MQOO_BIND_AS_Q_DEF

ローカル・キュー・マネージャーは、DefBind キュー属性で定義された方法でキュー・ハンドルのバインドします。この属性の値は、MQBND_BIND_ON_OPEN、MQBND_BIND_NOT_FIXED、または MQBND_BIND_ON_GROUP のいずれかです。

MQOO_BIND_ON_OPEN、MQOO_BIND_NOT_FIXED、MQOO_BIND_ON_GROUP のいずれも指定されていない場合は、MQOO_BIND_AS_Q_DEF がデフォルトです。

MQOO_BIND_AS_Q_DEF は、プログラムの文書化を助けます。このオプションは、他の 2 つのバインド・オプションのいずれかと組み合わせて使用することを意図して用意されたオプションではありません。しかしその値はゼロであるため、そのように組み合わせて使用しても検出できません。

コンテキスト・オプション: 以下のオプションは、メッセージ・コンテキストの処理を制御します。

MQOO_SAVE_ALL_CONTEXT

コンテキスト情報がこのキュー・ハンドルに関連付けられます。この情報は、このハンドルを使用して取り出されたメッセージのコンテキストから設定されます。メッセージ・コンテキストについて詳しくは、[メッセージ・コンテキスト](#) および [コンテキスト情報の制御](#) を参照してください。

このコンテキスト情報は、メッセージに渡してから、MQPUT 呼び出しまたは MQPUT1 呼び出しを使用してキューに書き込むことができます。472 ページの『MQPMO - メッセージ書き出しオプション』で説明されている MQPMO_PASS_IDENTITY_CONTEXT オプションおよび MQPMO_PASS_ALL_CONTEXT オプションを参照してください。

メッセージが正常に取り出されるまでは、キューに書き込まれるメッセージにコンテキストを渡すことはできません。

MQGMO_BROWSE_* ブラウズ・オプションのいずれかを使用して取り出されるメッセージには、そのコンテキスト情報が保存されません (ただし、**MsgDesc** パラメーターのコンテキスト・フィールドは、ブラウズの後で設定されます)。

このオプションは、ローカル・キュー、別名キュー、およびモデル・キューに関してのみ有効です。リモート・キューや配布リスト、さらにキューでないオブジェクトに関しては無効です。

MQOO_INPUT_* オプションのいずれかを指定する必要があります。

MQOO_PASS_IDENTITY_CONTEXT

これを使用すると、メッセージがキューに書き込まれるときに、**PutMsgOpts** パラメーターで **MQPMO_PASS_IDENTITY_CONTEXT** オプションを指定できます。この結果、**MQOO_SAVE_ALL_CONTEXT** オプションを指定して開かれた入力キューからの識別コンテキスト情報がメッセージに渡されます。メッセージ・コンテキストについての詳細は、[メッセージ・コンテキストおよびコンテキスト情報の制御](#)を参照してください。

指定する必要があるオプションは **MQOO_OUTPUT** です。

このオプションは、配布リストをはじめ、すべてのタイプのキューで有効です。

MQOO_PASS_ALL_CONTEXT

これを使用すると、メッセージがキューに書き込まれるときに、**PutMsgOpts** パラメーターで **MQPMO_PASS_ALL_CONTEXT** オプションを指定できます。この結果、**MQOO_SAVE_ALL_CONTEXT** オプションを指定して開かれた入力キューからの識別コンテキスト情報と発信元コンテキスト情報がメッセージに渡されます。メッセージ・コンテキストについての詳細は、[メッセージ・コンテキストおよびコンテキスト情報の制御](#)を参照してください。

このオプションでは、**MQOO_PASS_IDENTITY_CONTEXT** が暗黙指定されるため、これをあらためて指定する必要はありません。指定する必要があるオプションは **MQOO_OUTPUT** です。

このオプションは、配布リストをはじめ、すべてのタイプのキューで有効です。

MQOO_SET_IDENTITY_CONTEXT

これを使用すると、メッセージがキューに書き込まれるときに、**PutMsgOpts** パラメーターで **MQPMO_SET_IDENTITY_CONTEXT** オプションを指定できます。この結果、**MQPUT** 呼び出しまたは **MQPUT1** 呼び出しで指定された **MsgDesc** パラメーターに格納されている識別コンテキスト情報がメッセージに渡されます。メッセージ・コンテキストについての詳細は、[メッセージ・コンテキストおよびコンテキスト情報の制御](#)を参照してください。

このオプションでは、**MQOO_PASS_IDENTITY_CONTEXT** が暗黙指定されるため、これをあらためて指定する必要はありません。指定する必要があるオプションは **MQOO_OUTPUT** です。

このオプションは、配布リストをはじめ、すべてのタイプのキューで有効です。

MQOO_SET_ALL_CONTEXT

これを使用すると、メッセージがキューに書き込まれるときに、**PutMsgOpts** パラメーターで **MQPMO_SET_ALL_CONTEXT** オプションを指定できます。この結果、**MQPUT** 呼び出しまたは **MQPUT1** 呼び出しで指定された **MsgDesc** パラメーターに格納されている識別およびコンテキスト情報がメッセージに渡されます。メッセージ・コンテキストについての詳細は、[メッセージ・コンテキストおよびコンテキスト情報の制御](#)を参照してください。

このオプションでは、以下のオプションが暗黙指定されるため、これらをあらためて指定する必要はありません。

- **MQOO_PASS_IDENTITY_CONTEXT**
- **MQOO_PASS_ALL_CONTEXT**
- **MQOO_SET_IDENTITY_CONTEXT**

指定する必要があるオプションは **MQOO_OUTPUT** です。

このオプションは、配布リストをはじめ、すべてのタイプのキューで有効です。

先読みオプション:

MQOO_READ_AHEAD を使用して MQOPEN を呼び出すときに、特定の条件が満たされている場合にのみ、IBM MQ クライアントは先読みを使用可能にします。それらの条件には、以下のものが含まれます。

- クライアント・アプリケーションは、スレッド化された IBM MQ MQI クライアント・ライブラリーに対してコンパイルおよびリンクされている必要があります。
- クライアント・チャンネルが TCP/IP プロトコルを使用している必要があります。
- チャンネルでは、クライアントとサーバー両方のチャンネル定義で、SharingConversations (SHARECNV) がゼロ以外に設定されていなければなりません。

以下のオプションは、アプリケーションが非持続メッセージを要求する前にそれらをクライアントに送信するかどうかを制御します。以下の注意事項は、先読みオプションに適用されます。

- これらのオプションは、1 つだけ指定できます。
- これらのオプションは、ローカル・キュー、別名キュー、およびモデル・キューでのみ有効です。これらは、リモート・キュー、配布リスト、トピックおよびキュー・マネージャーでは無効です。
- これらのオプションは、MQOO_BROWSE、MQOO_INPUT_SHARED、MQOO_INPUT_EXCLUSIVE のいずれかと一緒に指定されている場合にのみ適用されます。ただし、これらのオプションを MQOO_INQUIRE または MQOO_SET と一緒に指定してもエラーにはなりません。
- アプリケーションが IBM MQ クライアントとして実行されていない場合、これらのオプションは無視されます。

MQOO_NO_READ_AHEAD

非持続メッセージは、それをアプリケーションが要求する前にクライアントに送信されることはありません。

MQOO_READ_AHEAD

非持続メッセージは、アプリケーションからの要求がある前に、クライアントに送信されます。

MQOO_READ_AHEAD_AS_Q_DEF

先読みするときの振る舞いは、開いているキューのデフォルトの先読み属性によって決まります。これがデフォルト値です。

その他のオプション: 以下のオプションは、許可検査を制御するほか、キュー・マネージャーが静止しているときに発生するイベント、ローカル・キュー名を解決するかどうか、およびマルチキャストを制御します。

MQOO_ALTERNATE_USER_AUTHORITY

パラメータ **ObjDesc** 内の *AlternateUserId* フィールドには、この MQOPEN 呼び出しを検証するために使用するユーザー識別子が含まれています。この呼び出しが成功するのは、指定されたアクセス・オプションでオブジェクトを開く許可を、この *AlternateUserId* が持っている場合だけです。アプリケーションの実行に使用されているユーザー ID が、そのような許可を持っているかどうかは関係ありません。ただし、これは、指定されたコンテキスト・オプションには適用されません。コンテキスト・オプションの場合は常に、そのアプリケーションの実行に使用されているユーザー ID に対して検査されます。

このオプションは、すべてのタイプのオブジェクトで有効です。

MQOO_FAIL_IF_QUIESCING

MQOPEN 呼び出しは、キュー・マネージャーが静止状態になっている場合は失敗します。

 **z/OS** z/OS では、CICS または IMS アプリケーションの場合、このオプションは、接続が静止状態にある場合に MQOPEN 呼び出しが失敗するように強制します。

このオプションは、すべてのタイプのオブジェクトで有効です。

クライアントチャンネルに関する情報はを参照してください [IBM MQ MQI clients](#)。

MQOO_RESOLVE_LOCAL_Q

MQOD 構造の ResolvedQName に、開かれたローカル・キューの名前が入ります。同様に、ResolvedQMgrName には、ローカル・キューをホストするローカル・キュー・マネージャーの名前

が入ります。MQOD 構造がバージョン 3 より小さい場合、MQOO_RESOLVE_LOCAL_Q は無視され、エラーも返されません。

ローカル・キュー、別名キュー、またはモデル・キューのいずれかが開かれている場合、ローカル・キューは常に返されますが、例えば、リモート・キューまたは非ローカル・クラスター・キューが MQOO_RESOLVE_LOCAL_Q オプションを指定せずに開かれている場合には返されません。ResolvedQName および ResolvedQMgrName には、リモート・キュー定義にある RemoteQName および RemoteQMgrName が入ります。あるいは選択されたリモート・クラスター・キューから同様に入ります。

リモート・キューなどを開くときに MQOO_RESOLVE_LOCAL_Q を指定した場合、ResolvedQName はメッセージの書き込み先の伝送キューになります。ResolvedQMgrName には、伝送キューをホストするローカル・キュー・マネージャーの名前が入ります。

キューでの参照、入力、または出力を許可されている場合、このフラグを MQOPEN 呼び出しで指定するために必要な権限があるといえます。特殊権限は必要ありません。

このオプションはキューおよびキュー・マネージャーにのみ有効です。

MQOO_RESOLVE_LOCAL_TOPIC

MQOD 構造の ResolvedQName に、開かれた管理トピックの名前が入られます。

MQOO_NO_MULTICAST

パブリケーション・メッセージの送信に、マルチキャストは使用されません。

このオプションは、MQOO_OUTPUT オプションと併用する場合にのみ有効です。MQOO_OUTPUT を使用せずに指定した場合、MQOPEN の実行結果として MQRC_OPTIONS_ERROR が返されます。

このオプションはトピックでのみ有効です。

Hobj

タイプ: MQHOBJ - 出力

このハンドルは、オブジェクトに対し設定されているアクセスを表します。また、オブジェクトに対して操作される後続の IBM MQ 呼び出しで指定する必要があります。MQCLOSE 呼び出しが発行されたとき、またはハンドルの有効範囲を定義する処理の単位が終了したときに、有効でなくなります。

返されるオブジェクト・ハンドルの有効範囲は、呼び出しで指定される接続ハンドルの有効範囲と同じです。ハンドルの有効範囲について詳しくは、[MQCONN - Hconn パラメーター](#)を参照してください。

CompCode

タイプ: MQLONG - 出力

完了コード。以下のいずれかです。

MQCC_OK

正常終了。

MQCC_WARNING

警告 (部分完了)。

MQCC_FAILED

呼び出し失敗。

理由

タイプ: MQLONG - 出力

CompCode を限定する理由コード。

CompCode が MQCC_OK の場合:

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_WARNING の場合:

MQRC_MULTIPLE_REASONS

(2136, X'858') 複数の理由コードが返されました。

CompCode が MQCC_FAILED の場合:

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'89C') アダプターが利用できません。

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852') アダプター・サービス・モジュールをロードできません。

MQRC_ALIAS_BASE_Q_TYPE_ERROR

(2001, X'7D1') 別名基本キューのタイプは無効です。

MQRC_API_EXIT_ERROR

(2374, X'946') API 出口で障害が発生しました。

MQRC_API_EXIT_LOAD_ERROR

(2183, X'887') API 出口をロードできません。

MQRC_ASID_MISMATCH

(2157, X'86D') 1 次 ASID とホーム ASID が異なっています。

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') 前の呼び出しが完了する前に MQI 呼び出しが入力されました。

MQRC_CF_NOT_AVAILABLE

(2345, X'929') カップリング・ファシリティが使用できません。

MQRC_CF_STRUC_AUTH_FAILED

(2348, X'92C') カップリング・ファシリティ構造の許可検査に失敗しました。

MQRC_CF_STRUC_ERROR

(2349, X'92D') カップリング・ファシリティ構造が無効です。

MQRC_CF_STRUC_FAILED

(2373, X'945') カップリング・ファシリティ構造体で障害が発生しました。

MQRC_CF_STRUC_IN_USE

(2346, X'92A') カップリング・ファシリティ構造体が使用中です。

MQRC_CF_STRUC_LIST_HDR_IN_USE

(2347, X'92B') カップリング・ファシリティ構造体のリスト・ヘッダーが使用中です。

MQRC_CICS_WAIT_FAILED

(2140, X'85C') 待機要求が CICS により拒否された。

MQRC_CLUSTER_EXIT_ERROR

(2266, X'8DA') クラスター・ワークロード出口で障害が発生しました。

MQRC_CLUSTER_PUT_INHIBITED

(2268, X'8DC') クラスター内のすべてのキューで書き込み呼び出しは使用禁止になっています。

MQRC_CLUSTER_RESOLUTION_ERROR

(2189, X'88D') クラスター名の解決に失敗しました。

MQRC_CLUSTER_RESOURCE_ERROR

(2269, X'8DD') クラスター・リソース・エラー。

MQRC_CONNECTION_BROKEN

(2009, X'7D9') キュー・マネージャーとの接続が失われました。

MQRC_CONNECTION_NOT_AUTHORIZED

(2217, X'8A9') 接続が許可されていません。

MQRC_CONNECTION QUIESCING

(2202, X'89A') 接続が静止しています。

MQRC_CONNECTION_STOPPING

(2203, X'89B') 接続がシャットダウン中です。

MQRC_DB2_NOT_AVAILABLE

(2342, X'926') Db2 サブシステムが利用できません。

MQRC_DEF_XMIT_Q_TYPE_ERROR

(2198, X'896') デフォルト伝送キューはローカルではありません。

MQRC_DEF_XMIT_Q_USAGE_ERROR

(2199, X'897') デフォルト伝送キューの使用法エラー。

MQRC_DYNAMIC_Q_NAME_ERROR

(2011, X'7DB') 動的キューの名前が無効です。

MQRC_HANDLE_NOT_AVAILABLE

(2017, X'7E1') 使用可能なハンドルがなくなりました。

MQRC_HCONN_ERROR

(2018, X'7E2') 接続ハンドルが無効です。

MQRC_HOBJ_ERROR

(2019, X'7E3') オブジェクト・ハンドルが無効です。

MQRC_MULTIPLE_REASONS

(2136, X'858') 複数の理由コードが返されました。

MQRC_NAME_IN_USE

(2201, X'899') 名前が使用中です。

MQRC_NAME_NOT_VALID_FOR_TYPE

(2194, X'892') オブジェクト名がオブジェクト・タイプとして無効です。

MQRC_NOT_AUTHORIZED

(2035, X'7F3') アクセスは許可されません。

MQRC_OBJECT_ALREADY_EXISTS

(2100, X'834') オブジェクトが存在しています。

MQRC_OBJECT_DAMAGED

(2101, X'835') オブジェクトが損傷しました。

MQRC_OBJECT_IN_USE

(2042, X'7FA') オプションが矛盾するオブジェクトが既に開いています。

MQRC_OBJECT_LEVEL_INCOMPATIBLE

(2360, X'938') オブジェクト・レベルに互換性がありません。

MQRC_OBJECT_NAME_ERROR

(2152, X'868') オブジェクト名が無効です。

MQRC_OBJECT_NOT_UNIQUE

(2343, X'927') オブジェクトが固有ではありません。

MQRC_OBJECT_Q_MGR_NAME_ERROR

(2153, X'869') オブジェクト・キュー・マネージャー名が無効です。

MQRC_OBJECT_RECORDS_ERROR

(2155, X'86B') オブジェクト・レコードが無効です。

MQRC_OBJECT_STRING_ERROR

(2441, X'0989') Objectstring フィールドが無効です。

MQRC_OBJECT_TYPE_ERROR

(2043, X'7FB') オブジェクト・タイプが無効です。

MQRC_OD_ERROR

(2044, X'7FC') オブジェクト記述子の構造が無効です。

MQRC_OPTION_NOT_VALID_FOR_TYPE

(2045, X'7FD') オプションが、オブジェクト・タイプとして無効です。

MQRC_OPTIONS_ERROR

(2046, X'7FE') オプションが無効であるか、矛盾しています。

MQRC_PAGESET_ERROR

(2193, X'891') ページ・セット・データ・セットへのアクセス中にエラーが発生しました。

MQRC_PAGESET_FULL

(2192, X'890') 外部ストレージ・メディアが満杯です。

MQRC_Q_DELETED

(2052, X'804') キューが削除されました。

MQRC_Q_MGR_NAME_ERROR

(2058, X'80A') キュー・マネージャー名が無効であるか、認識されていません。

MQRC_Q_MGR_NOT_AVAILABLE

(2059, X'80B') キュー・マネージャーを接続に使用できません。

MQRC_Q_MGR_QUIESCING

(2161, X'871') キュー・マネージャーが静止しています。

MQRC_Q_MGR_STOPPING

(2162, X'872') キュー・マネージャーのシャットダウン中です。

MQRC_Q_TYPE_ERROR

(2057, X'809') キュー・タイプが無効です。

MQRC_RECS_PRESENT_ERROR

(2154, X'86A') 存在するレコード数が無効です。

MQRC_REMOTE_Q_NAME_ERROR

(2184, X'888') リモート・キュー名が無効です。

MQRC_RESOURCE_PROBLEM

(2102, X'836') 使用できるシステム・リソースが不足しています。

MQRC_RESPONSE_RECORDS_ERROR

(2156, X'86C') 応答レコードが無効です。

MQRC_SECURITY_ERROR

(2063, X'80F') セキュリティー・エラーが発生しました。

MQRC_SELECTOR_SYNTAX_ERROR

2459 (X'099B') MQOPEN、MQPUT1、または MQSUB の呼び出しが発行されましたが、構文エラーが含まれる選択ストリングが指定されました。

MQRC_STOPPED_BY_CLUSTER_EXIT

(2188, X'88C') クラスタ・ワークロード出口によって呼び出しが拒否されました。

MQRC_STORAGE_MEDIUM_FULL

(2192, X'890') 外部ストレージ・メディアが満杯です。

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') ストレージが不足しています。

MQRC_SUPPRESSED_BY_EXIT

(2109, X'83D') 出口プログラムにより呼び出しが抑止されました。

MQRC_UNEXPECTED_ERROR

(2195, X'893') 予期しないエラーが発生しました。

MQRC_UNKNOWN_ALIAS_BASE_Q

(2082, X'822') 別名の基本キューが不明です。

MQRC_UNKNOWN_DEF_XMIT_Q

(2197, X'895') デフォルト伝送キューが不明です。

MQRC_UNKNOWN_OBJECT_NAME

(2085, X'825') オブジェクト名が不明です。

MQRC_UNKNOWN_OBJECT_Q_MGR

(2086, X'826') オブジェクトのキュー・マネージャーが不明です。

MQRC_UNKNOWN_REMOTE_Q_MGR

(2087, X'827') リモート・キュー・マネージャーが不明です。

MQRC_UNKNOWN_XMIT_Q

(2196, X'894') 伝送キューが不明です。

MQRC_WRONG_CF_LEVEL

(2366, X'93E') カップリング・ファシリティ構造のレベルが正しくありません。

MQRC_XMIT_Q_TYPE_ERROR

(2091, X'82B') 伝送キューはローカルではありません。

MQRC_XMIT_Q_USAGE_ERROR

(2092, X'82C') 伝送キューの使用方法が正しくありません。

これらのコードについて詳しくは、[メッセージおよび理由コード](#)を参照してください。

一般的な使用上の注意

1. 開かれるオブジェクトは、以下のいずれかです。

- 以下の目的をもつキュー。
 - メッセージを取得またはブラウズする (MQGET 呼び出しを使用)。
 - メッセージを書き込む (MQPUT 呼び出しを使用)。
 - キューの属性について照会する (MQINQ 呼び出しを使用)。
 - キューの属性を設定する (MQSET 呼び出しを使用)。

指定されたキューがモデル・キューである場合は、動的ローカル・キューが作成されます。724 ページの『MQOPEN - オブジェクトのオープン』で説明されている **ObjDesc** パラメーターを参照してください。

配布リストは、キューのリストを格納する特殊なタイプのキュー・オブジェクトです。これを開いてメッセージを書き込むことはできますが、メッセージの取得やブラウズ、あるいは属性の照会や設定を行うことはできません。詳しくは、使用上の注意 8 を参照してください。

QSGDISP (GROUP) があるキューは特別なタイプのキュー定義であり、MQOPEN または MQPUT1 呼び出しでは使用できません。

- リスト内のキューの名前について照会する名前リスト (MQINQ 呼び出しを使用)。
 - プロセス属性について照会するプロセス定義 (MQINQ 呼び出しを使用)。
 - ローカル・キュー・マネージャーの属性について照会するキュー・マネージャー (MQINQ 呼び出しを使用)。
 - メッセージをパブリッシュするトピック (MQPUT 呼び出しを使用)。
2. 1つのアプリケーションで同じオブジェクトを複数回オープンすることができます。オープンするたびに異なるオブジェクト・ハンドルが返されます。返されるそれぞれのハンドルは、対応するオープンの実行対象となる関数において使用できます。
3. 開かれるオブジェクトが、クラスター・キュー以外のキューである場合、ローカル・キュー・マネージャー内の名前解決はすべて、MQOPEN 呼び出しの時点で行われます。これには、次のことが含まれます。
- リモート・キューのローカル定義の名前を、リモート・キュー・マネージャーの名前と、そのキューがリモート・キュー・マネージャーで認識されている名前に解決する。
 - リモート・キュー・マネージャーの名前をローカル伝送キューの名前に解決する。
 -  z/OS の場合のみ、リモート・キュー・マネージャーの名前を、IGQ エージェントで 사용되는共有伝送キューの名前に解決する (ローカルおよびリモートのキュー・マネージャーが、同じキュー共有グループに属している場合にのみ適用される)。
 - 別名を基本キューまたはトピック・オブジェクトの名前に解決する。

ただし、そのハンドルに対する後続の MQINQ 呼び出しまたは MQSET 呼び出しは、オープンされている名前に関連するものであり、ネーム・レゾリューションが行われた後の結果のオブジェクトとは関連がない点に注意してください。例えば、オープンされたオブジェクトが別名である場合、MQINQ 呼び出しで返される属性は別名の属性であり、別名の解決先の基本キューまたはトピック・オブジェクトの属性ではありません。

オープンするオブジェクトがクラスター・キューの場合、ネーム・レゾリューションは MQOPEN 呼び出しの時点で行うことも、据え置くこともできます。ネーム・レゾリューションをいつ行うかは、MQOPEN 呼び出しで指定された MQOO_BIND_* オプションで制御されます。

- MQOO_BIND_ON_OPEN
- MQOO_BIND_NOT_FIXED

- MQOO_BIND_AS_Q_DEF
- MQOO_BIND_ON_GROUP

クラスター・キューのネーム・レゾリューションの詳細については、[ネーム・レゾリューション](#)を参照してください。

4. MQOO_BROWSE オプションを使用して MQOPEN 呼び出しを行うと、オブジェクト・ハンドルといずれか 1 つのブラウズ・オプションを指定する MQGET 呼び出しで使用するブラウズ・カーソルが設定されます。これにより、内容を変更せずにキューをスキャンすることができます。ブラウズによって見つかったメッセージは、MQGMO_MSG_UNDER_CURSOR オプションを使用してキューから除去することができます。

同一のキューに対していくつかの MQOPEN 要求を出すと、1 つのアプリケーションに対して複数のブラウズ・カーソルをアクティブにすることができます。

5. トリガー・モニターにより開始されるアプリケーションには、そのアプリケーションに関連付けられているキューの名前が、開始時に渡されます。このキュー名を **ObjDesc** パラメーターに指定してキューを開くことができます。詳細については、595 ページの『MQTMC2 - トリガー・メッセージ 2 (文字フォーマット)』を参照してください。

先読みオプション

MQOO_READ_AHEAD を使用して MQOPEN を呼び出すときに、特定の条件が満たされている場合にのみ、IBM MQ クライアントは先読みを使用可能にします。それらの条件には、以下のものが含まれます。

- クライアント・アプリケーションは、スレッド化された IBM MQ MQI クライアント・ライブラリーに対してコンパイルおよびリンクされている必要があります。
- クライアント・チャンネルが TCP/IP プロトコルを使用している必要があります。
- チャンネルでは、クライアントとサーバー両方のチャンネル定義で、SharingConversations (SHARECNV) がゼロ以外に設定されていなければなりません。

先読みオプションを使用する際には、以下の注記の内容が適用されます。

1. 先読みオプションが適用されるのは、MQOO_BROWSE、MQOO_INPUT_SHARED、または MQOO_INPUT_EXCLUSIVE いずれかのオプションが 1 つだけ一緒に指定されている場合のみです。先読みオプションが MQOO_INQUIRE オプションまたは MQOO_SET オプションと一緒に指定されている場合、エラーはスローされません。
2. 最初の MQGET 呼び出しで使用するオプションが先読みでの使用に対応していない場合、先読みを要求しても使用可能になりません。さらに、クライアントが先読みをサポートしないキュー・マネージャーに接続する場合も、先読みは使用不可になります。
3. 実行されているアプリケーションが IBM MQ クライアントではない場合、先読みオプションは無視されます。

クラスター・キュー

以下の注意事項は、クラスター・キューの使用に適用されます。

1. 初めてクラスター・キューが開かれたとき、ローカル・キュー・マネージャーがフル・リポジトリ・キュー・マネージャーでなければ、ローカル・キュー・マネージャーは、フル・リポジトリ・キュー・マネージャーからそのクラスター・キューに関する情報を取得します。ネットワークが使用中である場合は、ローカル・キュー・マネージャーがリポジトリ・キュー・マネージャーから必要な情報を受信するまでに何秒か要する場合があります。その結果、MQOPEN 呼び出しを発行したアプリケーションは最大 10 秒間待機しなければならない場合があります、その後、MQOPEN 呼び出しから制御が戻ります。この時間内にローカル・キュー・マネージャーがクラスター・キューに関する必要情報を受信しなかった場合、呼び出しは失敗し、理由コード MQRC_CLUSTER_RESOLUTION_ERROR が返されます。
2. あるクラスター・キューが開いており、クラスター内にそのキューのインスタンスが複数存在する場合、どのインスタンスが開かれるかは、MQOPEN 呼び出しで指定されたオプションによって決まります。
 - 指定したオプションに、次のいずれかが含まれている場合:

- MQOO_BROWSE
- MQOO_INPUT_AS_Q_DEF
- MQOO_INPUT_EXCLUSIVE
- MQOO_INPUT_SHARED
- MQOO_SET

開かれるクラスター・キューのインスタンスは、ローカル・インスタンスでなければなりません。そのキューにローカル・インスタンスがない場合は、MQOPEN 呼び出しは失敗します。

- 指定したオプションに上記のオプションがいずれも含まれておらず、次のうち 1 つまたは両方が含まれている場合:

- MQOO_INQUIRE
- MQOO_OUTPUT

開かれるインスタンスは、ローカル・インスタンスがあればローカル・インスタンスになり、なければリモート・インスタンスになります (CLWLUSEQ のデフォルトを使用する場合)。ただし、キュー・マネージャーによって選択されたインスタンスを、クラスター・ワークロード出口によって変更することもできます (そのような出口がある場合)。

3. キューのサブスクリプションが存在するが、フル・リポジトリで認知されていない場合、オブジェクトはクラスター内に存在せず、呼び出しは失敗して理由コード MQRC_OBJECT_NAME が返されます。

クラスター・キューの詳細については、[クラスター・キュー](#)を参照してください。

配布リスト

次の注意事項は、配布リストの使用に適用されます。

配布リストは、次の環境でサポートされます。

-  AIX
-  IBM i
-  Linux
-  Windows

および、これらのシステムに接続された IBM MQ MQI clients。

1. MQOD 構造内の各フィールドは、配布リストを開くときに、次のように設定しなければなりません。

- Version は、MQOD_VERSION_2 以上にする必要があります。
- ObjectType は、MQOT_Q にする必要があります。
- ObjectName は、ブランクまたはヌル・ストリングにする必要があります。
- ObjectQMgrName は、ブランクまたはヌル・ストリングにする必要があります。
- RecsPresent は、ゼロより大きな値にする。
- ObjectRecOffset と ObjectRecPtr のうちの片方をゼロ、もう片方をゼロ以外にする必要があります。
- ResponseRecOffset および ResponseRecPtr のうち、ゼロ以外にできるのは片方のみです。
- ObjectRecOffset または ObjectRecPtr のいずれかにより扱われる RecsPresent オブジェクト・レコードが存在する必要があります。これらのオブジェクト・レコードには、開かれる宛先キューの名前を設定しなければなりません。
- ResponseRecOffset および ResponseRecPtr のうちの片方がゼロ以外であるとき、RecsPresent 応答レコードが存在する必要があります。この応答レコードは、呼び出しが終了して理由コード MQRC_MULTIPLE_REASONS が返された場合に、キュー・マネージャーにより設定されます。

RecsPresent をゼロにすることにより、バージョン 2 の MQOD を使用して、配布リストに存在しない 1 つのキューを開くこともできます。

2. **Options** パラメーターでは、次のオープン・オプションだけが有効です。

- MQOO_OUTPUT
- MQOO_PASS_*_CONTEXT
- MQOO_SET_*_CONTEXT
- MQOO_ALTERNATE_USER_AUTHORITY
- MQOO_FAIL_IF_QUIESCING

3. 配布リスト内の宛先キューとして、ローカル・キュー、別名キュー、またはリモート・キューを指定することは可能ですが、モデル・キューを指定することはできません。モデル・キューを指定すると、キューのオープンが失敗し、理由コード MQRC_Q_TYPE_ERROR が返されます。ただし、このようになっても、リスト内の他のキューは正常に開かれます。

4. 完了コード・パラメーターおよび理由コード・パラメーターは、次のように設定されます。

- 配布リスト内のキューに対するオープン操作がすべて同じ結果になった (すべて成功または失敗した場合)、完了コード・パラメーターおよび理由コード・パラメーターは、この共通の結果を示す値に設定されます。MQRR 応答レコード (アプリケーションにより提供されている場合) は、この場合には設定されません。

例えば、すべてのオープンが成功すると、完了コードは MQCC_OK に設定され、理由コードは MQRC_NONE に設定されます。いずれのキューも存在しないためにすべてのオープンが失敗すると、それらのパラメーターは MQCC_FAILED および MQRC_UNKNOWN_OBJECT_NAME に設定されます。

- 配布リスト内のキューに対するオープン操作が同じ結果にならなかった (すべて成功でもすべて失敗でもない) 場合には、次のようになります。
 - 完了コード・パラメーターは、少なくとも 1 つのオープンが成功した場合には MQCC_WARNING に設定され、すべて失敗した場合には MQCC_FAILED に設定されます。
 - 理由コード・パラメーターは、MQRC_MULTIPLE_REASONS に設定されます。
 - 応答レコード (アプリケーションにより提供されている場合) は、配布リスト内のキューごとに、個別の完了コードおよび理由コードに設定されます。

5. 配布リストが正常にオープンされた場合、呼び出しにより返された Hobj ハンドルを後続の MQPUT 呼び出しで使用して、配布リスト内のキューにメッセージを書き込むことができます。さらにこのハンドルを MQCLOSE 呼び出しで使用して、配布リストへのアクセスを解放することもできます。配布リストで有効なクローズ・オプションは、MQCO_NONE のみです。

配布リストにメッセージを書き込むために、MQPUT1 呼び出しを使用することもできます。このリスト内のキューを定義する MQOD 構造は、その呼び出しのパラメーターとして指定されます。

6. アプリケーションが最大許容ハンドル数を超えたかどうかを検査するとき、配布リスト内の正常に開かれた宛先ごとに、別のハンドルとしてカウントされます (**MaxHandles** キュー・マネージャー属性を参照)。配布リスト内の複数の宛先が同一の物理キューに解決されるときにも同じようにカウントされます。1 つの配布リストについて発行された MQOPEN または MQPUT1 の呼び出しの結果として、アプリケーションで使用されるハンドルの数が MaxHandles を超える場合、呼び出しは失敗し、理由コード MQRC_HANDLE_NOT_AVAILABLE が返されます。

7. 宛先が正常に開かれるたびに、**OpenOutputCount** 属性の値が 1 つずつ加算されます。配布リスト内の複数の宛先が同一の物理キューに解決される場合、そのキューの **OpenOutputCount** 属性は、そのキューに解決される配布リスト内の宛先数だけ加算されます。

8. 各キューを個々にオープンするとハンドルが無効になるようなキュー定義の変更 (例えば、解決パスの変更) があっても、配布リスト・ハンドルは無効にはなりません。しかし、後続の MQPUT 呼び出しで配布リスト・ハンドルが使用される際、その特定のキューについては失敗します。

9. 配布リストに含めることができるのは 1 つの宛先のみです。

リモート・キュー

以下の注意事項は、リモート・キューの使用に適用されます。

この呼び出しの **ObjDesc** パラメーターには、リモート・キューを次の 2 つの方法のいずれかで指定できます。

- **ObjectName** として、リモート・キューのローカル定義の名前を指定する。この場合、**ObjectQMgrName** は、ローカル・キュー・マネージャーを指しており、ブランクまたは (C プログラミング言語では) ヌル・ストリングとして指定することができます。

ローカル・キュー・マネージャーで実行されるセキュリティ妥当性検査では、そのユーザーが、リモート・キューのローカル定義をオープンする許可を持っているかどうかを検査されます。

- **ObjectName** として、リモート・キュー・マネージャーに認識されているリモート・キューの名前を指定する。この場合、**ObjectQMgrName** はリモート・キュー・マネージャーの名前です。

ローカル・キュー・マネージャーによって実行されるセキュリティ妥当性検査では、ユーザーが、名前解決プロセスからの結果の伝送キューにメッセージを送る許可を持っているかどうかを検査されます。

いずれの場合も、次のようになります。

- ユーザーがキューにメッセージを書き込む許可を持っているかどうかを検査するために、ローカル・キュー・マネージャーからリモート・キュー・マネージャーへメッセージが送られることはありません。
- メッセージがリモート・キュー・マネージャーに届くとき、リモート・キュー・マネージャーは、メッセージを発信しているユーザーが許可を持っていないため、それを拒否することがあります。

詳細については、[449 ページの『MQOD - オブジェクト記述子』](#)で説明されている **ObjectName** フィールドおよび **ObjectQMgrName** フィールドを参照してください。

オブジェクト

セキュリティ

以下の注記は、MQOPEN を使用する場合のセキュリティに関連した内容です。

キュー・マネージャーは、MQOPEN 呼び出しが発行されるときにセキュリティ検査を行い、アプリケーションの実行に使用されるユーザー ID に適切なレベルの権限があることを確認してからアクセスが許可されます。許可検査は、オープンされるオブジェクトの名前に対して行われ、名前が解決された後の結果の名前 (1 つ以上) に対しては行われません。

開かれるオブジェクトがトピック・オブジェクトを指す別名キューである場合、キュー・マネージャーは、トピック・オブジェクトが直接使用されているかのように、トピックのセキュリティ検査を行う前に、別名キュー名に対してセキュリティ検査を行います。

開かれるオブジェクトがトピック・オブジェクトである場合、**ObjectName** のみを使用するか、あるいは (ベースになる **ObjectName** を共に使用してあるいは使用せずに) **ObjectString** を使用するにかかわらず、キュー・マネージャーはセキュリティ検査を行います。これは、**ObjectName** で指定されたトピック・オブジェクト内から取られる結果トピック・ストリングを使用して、必要な場合にはそのストリングを **ObjectString** で提供されるストリングと連結してから、セキュリティ検査を行う対象となるトピック・ツリーのそのポイントまたはそれより上のポイントにある最も近いトピック・オブジェクトを検出することによって行われます。これは、**ObjectName** で指定されたトピック・オブジェクトと同じではない場合があります。

オープンされるオブジェクトがモデル・キューの場合、キュー・マネージャーは、モデル・キューの名前と作成された動的キューの名前の両方に対して、完全セキュリティ検査を行います。作成される動的キューが後で明示的に開かれると、さらにリソース・セキュリティ検査が動的キューの名前に対して実行されます。

 **z/OS** では、キュー・マネージャーがセキュリティ検査を実行するのは、セキュリティが有効になっている場合のみです。セキュリティ検査の詳細については、[z/OS でのセキュリティのセットアップ](#)を参照してください。

属性

以下の注記は、属性に関連した内容です。

アプリケーションでオブジェクトがオープンされている間に、そのオブジェクトの属性が変わることもあります。多くの場合、アプリケーションでは属性の変化を通知しませんが、特定の属性についてキュー・マネージャーがハンドルに「無効」としてマーク付けます。以下の属性が該当します。

- オブジェクトの名前の解決に影響するすべての属性。これは使用されるオープン・オプションに関係なく当てはまります。以下のものが含まれます。
 - 開いている別名キューの **BaseQName** 属性に対する変更。
 - 開いている別名キューの **TargetType** 属性に対する変更。
 - **RemoteQName** キュー属性または **RemoteQMgrName** キュー属性に対する変更では、このキュー、またはキュー・マネージャーの別名としてこの定義を使って解決されるキューの、開いているハンドル。
 - リモート・キュー用に現在開いているハンドルの解決先が別の伝送キューになるような変更、あるいはまったく解決できなくなるような変更。例えば、次のものが含まれます。
 - リモート・キューのローカル定義の **XmitQName** 属性に対する変更 (その定義がキュー用に使用されるかキュー・マネージャーの別名用に使用されているかには無関係)。
 -  **z/OS** オン z/OS のみ、キューマネージャ **IntraGroupQueuing** 属性の値の変更、または IGQ エージェントが使用する共有伝送キュー (SYSTEM.QSG.TRANSMIT.QUEUE) の定義の変更。

この条件には例外が 1 つあります。その例外とは、新規伝送キューの作成です。ハンドルを開くとき、このキューが存在していればそれが解決先となっていたものの、実際にはそれがなかったためにデフォルトの伝送キューが解決先となった場合、このハンドルは無効にはなりません。

- **DefXmitQName** キュー・マネージャーに対する変更。この場合は、以前に指定されたキューに解決されたすべてのオープン・ハンドル (デフォルトの伝送キューであるというだけの理由でそれに解決されたオープン・ハンドル) に、無効のマークが付けられます。その他の理由でこのキューに解決されたハンドルは影響を受けません。
- **Shareability** キュー属性では、このキュー、またはこのキューに解決されるキューに対して、現在 **MQOO_INPUT_SHARED** アクセスを提供しているハンドルが 2 つ以上ある場合。この場合は、オープン・オプションとは関係なく、このキュー、またはこのキューに解決されるキューの、開かれているすべてのハンドルに無効のマークが付けられます。

 **z/OS** z/OS では、1 つ以上のハンドルが現在 **MQOO_INPUT_SHARED** アクセスまたは **MQOO_INPUT_EXCLUSIVE** アクセスをキューに提供している場合、上記のハンドルに無効のマークが付けられます。

- オープン・オプションとは関係なく、このキュー、またはこのキューに解決されるキューに対して開かれているすべてのハンドルの **Usage** キュー属性。

ハンドルに無効のマークが付けられると、このハンドルを使用する後続の呼び出し (**MQCLOSE** 以外) はすべて失敗し、理由コード **MQRC_OBJECT_CHANGED** が返されます。アプリケーションで、**MQCLOSE** 呼び出し (元のハンドルを使用) を発行してから、キューを再オープンする必要があります。以前に成功した呼び出しで使用した古いハンドルに対するコミットされていない更新は、この時点でもアプリケーション・ロジックの必要に応じてコミットまたはバックアウトが可能です。

属性を変更した結果このようなことが起こる場合は、特別な強制バージョンの呼び出しを使用します。

C 言語での呼び出し

```
MQOPEN (Hconn, &ObjDesc, Options, &Hobj, &CompCode,
        &Reason);
```

パラメーターを次のように宣言します。

```
MQHCONN  Hconn;      /* Connection handle */
MQOD      ObjDesc;    /* Object descriptor */
MQLONG    Options;    /* Options that control the action of MQOPEN */
MQHOBJ     Hobj;      /* Object handle */
```

```

MQLONG  CompCode;  /* Completion code */
MQLONG  Reason;    /* Reason code qualifying CompCode */

```

COBOL での呼び出し

```
CALL 'MQOPEN' USING HCONN, OBJDESC, OPTIONS, HOBJ, COMPCODE, REASON
```

パラメーターを次のように宣言します。

```

** Connection handle
01 HCONN      PIC S9(9) BINARY.
** Object descriptor
01 OBJDESC.
   COPY CMQODV.
** Options that control the action of MQOPEN
01 OPTIONS    PIC S9(9) BINARY.
** Object handle
01 HOBJ       PIC S9(9) BINARY.
** Completion code
01 COMPCODE   PIC S9(9) BINARY.
** Reason code qualifying COMPCODE
01 REASON     PIC S9(9) BINARY.

```

PL/I での呼び出し

```
call MQOPEN (Hconn, ObjDesc, Options, Hobj, CompCode, Reason);
```

パラメーターを次のように宣言します。

```

dcl Hconn      fixed bin(31); /* Connection handle */
dcl ObjDesc    like MQOD;    /* Object descriptor */
dcl Options    fixed bin(31); /* Options that control the action of
                               MQOPEN */
dcl Hobj       fixed bin(31); /* Object handle */
dcl CompCode   fixed bin(31); /* Completion code */
dcl Reason     fixed bin(31); /* Reason code qualifying CompCode */

```

高水準アセンブラー呼び出し

```
CALL MQOPEN,(HCONN,OBJDESC,OPTIONS,HOBJ,COMPCODE,REASON)
```

パラメーターを次のように宣言します。

```

HCONN      DS      F Connection handle
OBJDESC    CMQODA   , Object descriptor
OPTIONS    DS      F Options that control the action of MQOPEN
HOBJ       DS      F Object handle
COMPCODE   DS      F Completion code
REASON     DS      F Reason code qualifying COMPCODE

```

Visual Basic での呼び出し

 Windows

```
MQOPEN Hconn, ObjDesc, Options, Hobj, CompCode, Reason
```

パラメーターを次のように宣言します。

```

Dim Hconn      As Long 'Connection handle'
Dim ObjDesc    As MQOD 'Object descriptor'
Dim Options    As Long 'Options that control the action of MQOPEN'
Dim Hobj       As Long 'Object handle'
Dim CompCode   As Long 'Completion code'
Dim Reason     As Long 'Reason code qualifying CompCode'

```

MQPUT - メッセージの書き込み

MQPUT 呼び出しは、キューまたは配布リスト上に、あるいはトピックにメッセージを書き込みます。キュー、配布リスト、またはトピックは、既にオープンされていなければなりません。

構文

MQPUT (*Hconn*, *Hobj*, *MsgDesc*, *PutMsgOpts*, *BufferLength*, *Buffer*, *CompCode*, *Reason*)

パラメーター

Hconn

タイプ: MQHCONN - 入力

このハンドルは、キュー・マネージャーに対する接続を表します。Hconn の値は、先行の MQCONN または MQCONNX 呼び出しによって戻されたものです。

 z/OS 上で CICS アプリケーションを使用する場合は、MQCONN 呼び出しを省略でき、Hconn に次の値を指定できます。

MQHC_DEF_HCONN

デフォルトの接続ハンドル。

Hobj

タイプ: MQHOBJ - 入力

このハンドルは、メッセージが追加されるキュー、またはメッセージがパブリッシュされるトピックを表します。Hobj の値は、MQOO_OUTPUT オプションを指定した、前の MQOPEN 呼び出しから戻されたものです。

MsgDesc

タイプ: MQMD - 入出力

この構造体は、送られるメッセージの属性を記述するものであり、書き込み要求が完了した後でメッセージに関する情報を受け取ります。詳細は、[385 ページの『MQMD - メッセージ記述子』](#)を参照してください。

アプリケーションがバージョン 1 の MQMD を提供している場合、メッセージ・データの接頭部に MQMDE 構造体を付ければ、バージョン 2 の MQMD に存在し、バージョン 1 には存在しない各フィールドの値を指定できます。MQMD 内の *Format* フィールドは、MQMDE が存在することを示すため、MQFMT_MD_EXTENSION に設定しておく必要があります。詳細については、[439 ページの『MQMDE - 拡張メッセージ記述子』](#)を参照してください。

MQPMO 構造体の `MessageHandle` NewMsgHandleOriginalMsgHandle フィールドまたは `MessageHandle` フィールドに有効なメッセージハンドルが指定されている場合、アプリケーションは MQMD 構造体を指定する必要はありません。これらのフィールドに何も提供されていない場合、メッセージの記述子は、メッセージ・ハンドルに関連した記述子から取られます。

API 出口を使用する場合、または使用する予定がある場合は、MQMD 構造体を明示的に提供して、メッセージ・ハンドルに関連付けられているメッセージ記述子を使用しないことをお勧めします。これは、MQPUT 呼び出しまたは MQPUT1 呼び出しに関連付けられている API 出口では、キュー・マネージャーが MQPUT 要求または MQPUT1 要求を完了するために使用する MQMD 値を確認できないためです。

PutMsgOpts

タイプ: MQPMO - 入出力

詳細は [472 ページの『MQPMO - メッセージ書き出しオプション』](#)を参照してください。

BufferLength

タイプ: MQLONG - 入力

メッセージ Buffer の長さ。ゼロは有効であり、メッセージにアプリケーション・データが含まれていないことを示します。BufferLength の上限は様々な要因によって異なります。

- 宛先がローカル・キューであるか、またはローカル・キューに解決される場合、上限は以下の条件を満たすかどうかによって異なります。
 - ローカル・キュー・マネージャーがセグメント化をサポートしている。
 - 送信側アプリケーションが、キュー・マネージャーでメッセージのセグメント化を可能にするフラグを指定している。このフラグは `MQMF_SEGMENTATION_ALLOWED` であり、バージョン 2 の `MQMD` 内で指定できるほか、バージョン 1 の `MQMD` を使用する場合は `MQMDE` 内で指定できます。

この 2 つの条件が両方とも満たされている場合は、`BufferLength` は 999 999 999 から `MQMD` 内の `Offset` フィールドの値を引いた値を超えることはできません。したがって、書き込むことのできる最長の論理メッセージは、999 999 999 バイト (`Offset` がゼロの場合) になります。ただし、オペレーティング・システムによって、またはアプリケーションが実行されている環境によってリソースが制約される結果、上限がこれよりさらに小さい値になる場合があります。

上記の条件のいずれか一方または両方が満たされない場合、`BufferLength` は、キューの `MaxMsgLength` 属性とキュー・マネージャーの `MaxMsgLength` 属性のうち小さい方の値以下でなければなりません。

- 宛先がリモート・キューの場合や宛先の解決先がリモート・キューである場合も、ローカル・キューの条件が適用されます。ただし、メッセージが宛先キューに到達するまでに通過するすべてのキュー・マネージャーが適用の対象となります。特に、次のキューに注意してください。
 - ローカル・キュー・マネージャーで一時的にメッセージを保管するために使用されるローカル伝送キュー。
 - ローカルのキュー・マネージャーと宛先のキュー・マネージャーとの間の経路にあるキュー・マネージャーで、メッセージを保管するために使用される中間伝送キュー (それがあつた場合)。
 - 宛先キュー・マネージャーでの宛先キュー。

したがって、書き込み可能なメッセージの最大長は、これらのキューやキュー・マネージャーのうち、もっとも制限の厳しいものによって決まります。

メッセージが伝送キューに入れられる場合は、メッセージ・データと共に追加情報があるため、転送できるアプリケーション・データの量は小さくなります。この状況では、の制限を決定 `BufferLength` する際、送信キューの値 `MaxMsgLength` から `MQ_MSG_HEADER_LENGTH` バイトを差し引きます。

注: メッセージが書き込まれると、条件 1 を満たせなかった場合のみ、同期的に診断することができません。また、この場合は、理由コード `MQRC_MSG_TOO_BIG_FOR_Q` または `MQRC_MSG_TOO_BIG_FOR_Q_MGR` が戻ります。条件 2 または 3 が満たされない場合、メッセージは、中間キュー・マネージャーまたは宛先キュー・マネージャーのいずれかの箇所で送達不能 (未配布メッセージ) キューにリダイレクトされます。これが発生した場合、送信側からの要求があれば、レポート・メッセージが生成されます。

Buffer

タイプ: `MQBYTEExBufferLength` - 入力

これは、送信するアプリケーション・データが入っているバッファです。バッファは、メッセージのデータの性質に適した境界に位置合わせされなければなりません。ほとんどのメッセージ (IBM MQ ヘッダー構造の入ったメッセージを含む) には 4 バイト境界の位置合わせが適していますが、メッセージによってはより厳しい位置合わせを必要とする場合があります。例えば、64 ビット・バイナリ整数を含むメッセージは 8 バイト境界に合わせる必要がある場合があります。

文字データまたは数値データ `Buffer` を含む場合、パラメータ `MsgDesc` 内の `'charset'` `EncodingCodedCharSetId` および `'encoding'` フィールドをデータに適した値に設定してください。これにより、メッセージの受信側は (必要に応じて) データを受信側が使用する文字セットおよびエンコーディングに変換できるようになります。

注: `MQPUT` 呼び出しの他のパラメーターはすべて、(`CodedCharSetId` キュー・マネージャー属性および `MQENC_NATIVE` で指定した) ローカル・キュー・マネージャーの文字セットおよびエンコードで指定する必要があります。

C プログラミング言語では、パラメーターは、void を示すポインターとして宣言されます。つまり、どのタイプのデータのアドレスもパラメーターとして指定できます。

BufferLength パラメーターがゼロの場合は、**Buffer** は参照されません。この場合、C または System/390 アセンブラーで作成されたプログラムによって渡されるパラメーター・アドレスはヌルのこともあります。

CompCode

タイプ: MQLONG - 出力

完了コード。以下のいずれかです。

MQCC_OK

正常終了。

MQCC_WARNING

警告 (部分完了)。

MQCC_FAILED

呼び出し失敗。

理由

タイプ: MQLONG - 出力

CompCode を限定する理由コード。

CompCode が MQCC_OK の場合:

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_WARNING の場合:

MQRC_INCOMPLETE_GROUP

(2241, X'8C1') メッセージ・グループが不完全である。

MQRC_INCOMPLETE_MSG

(2242, X'8C2') 論理メッセージが不完全である。

MQRC_INCONSISTENT_PERSISTENCE

(2185, X'889') 持続性の指定が不整合である。

MQRC_INCONSISTENT_UOW

(2245, X'8C5') 作業単位の指定が不整合である。

MQRC_MULTIPLE_REASONS

(2136, X'858') 複数の理由コードが返されました。

MQRC_PRIORITY_EXCEEDS_MAXIMUM

(2049, X'801') メッセージ優先順位が、サポートされる最大値を超えている。

MQRC_UNKNOWN_REPORT_OPTION

(2104, X'838') メッセージ記述子のレポート・オプション (1 つまたは複数) が認識されない。

CompCode が MQCC_FAILED の場合:

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'89C') アダプターが利用できません。

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852') アダプター・サービス・モジュールをロードできません。

MQRC_ALIAS_TARGTYPE_CHANGED

(2480, X'09B0') サブスクリプションのターゲット・タイプがキューからトピック・オブジェクトに変更された。

MQRC_API_EXIT_ERROR

(2374, X'946') API 出口で障害が発生しました。

MQRC_API_EXIT_LOAD_ERROR

(2183, X'887') API 出口をロードできません。

MQRC_ASID_MISMATCH

(2157, X'86D') 1 次 ASID とホーム ASID が異なっています。

MQRC_BACKED_OUT

(2003, X'7D3') 作業単位がバックアウトされた。

MQRC_BUFFER_ERROR

(2004, X'7D4') バッファー・パラメーターが無効である。

MQRC_BUFFER_LENGTH_ERROR

(2005, X'7D5') バッファー長パラメーターは無効です。

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') 前の呼び出しが完了する前に MQI 呼び出しが入力されました。

MQRC_CALL_INTERRUPTED

(2549, X'9F5') MQPUT または MQCMIT が中断されたため、再接続処理で確実な成果を再び得ることができない。

MQRC_CF_NOT_AVAILABLE

(2345, X'929') カップリング・ファシリティが使用できません。

MQRC_CF_STRUC_FAILED

(2373, X'945') カップリング・ファシリティ構造体で障害が発生しました。

MQRC_CF_STRUC_IN_USE

(2346, X'92A') カップリング・ファシリティ構造体が使用中です。

MQRC_CFGR_ERROR

(2416, X'970') メッセージ・データ内の PCF グループ・パラメーター構造 MQCFGR が無効である。

MQRC_CFH_ERROR

(2235, X'8BB') PCF ヘッダー構造体が無効である。

MQRC_CFIF_ERROR

(2414, X'96E') メッセージ・データ内の PCF 整数フィルターのパラメーター構造が無効である。

MQRC_CFIL_ERROR

(2236, X'8BC') PCF 整数リスト・パラメーター構造または PCIF*64 整数リスト・パラメーター構造が無効である。

MQRC_CFIN_ERROR

(2237, X'8BD') PCF 整数のパラメーター構造体または PCIF*64 整数のパラメーター構造が無効である。

MQRC_CFSF_ERROR

(2415, X'96F') メッセージ・データ内の PCF スtring・フィルターのパラメーター構造が無効である。

MQRC_CFSL_ERROR

(2238, X'8BE') PCF スtring・リストのパラメーター構造体が無効である。

MQRC_CFST_ERROR

(2239, X'8BF') PCF スtringのパラメーター構造体が無効である。

MQRC_CICS_WAIT_FAILED

(2140, X'85C') 待機要求が CICS により拒否された。

MQRC_CLUSTER_EXIT_ERROR

(2266, X'8DA') クラスター・ワークロード出口で障害が発生しました。

MQRC_CLUSTER_RESOLUTION_ERROR

(2189, X'88D') クラスター名の解決に失敗しました。

MQRC_CLUSTER_RESOURCE_ERROR

(2269, X'8DD') クラスター・リソース・エラー。

MQRC_COD_NOT_VALID_FOR_XCF_Q

(2106, X'83A') COD レポート・オプションが XCF キューについて無効である。

MQRC_CONNECTION_BROKEN

(2009, X'7D9') キュー・マネージャーとの接続が失われました。

MQRC_CONNECTION_NOT_AUTHORIZED

(2217, X'8A9') 接続が許可されていません。

MQRC_CONNECTION QUIESCING

(2202, X'89A') 接続が静止しています。

MQRC_CONNECTION_STOPPING

(2203, X'89B') 接続がシャットダウン中です。

MQRC_CONTENT_ERROR

2554 (X'09FA') メッセージの内容を解析することで、拡張メッセージ・セレクターを使用してメッセージをサブスクライバーに送信する必要があるかどうかを判別できなかった。

MQRC_CONTEXT_HANDLE_ERROR

(2097, X'831') 参照されたキュー・ハンドルがコンテキストを保存しない。

MQRC_CONTEXT_NOT_AVAILABLE

(2098, X'832') 参照されたキュー・ハンドルでコンテキストが使用できない。

MQRC_DATA_LENGTH_ERROR

(2010, X'7DA') データ長パラメーターが無効である。

MQRC_DH_ERROR

(2135, X'857') 配布ヘッダー構造体が無効である。

MQRC_DLH_ERROR

(2141, X'85D') 送達不能ヘッダー構造体が無効である。

MQRC_EPH_ERROR

(2420, X'974') 組み込み PCF 構造体が無効である。

MQRC_EXPIRY_ERROR

(2013, X'7DD') 満了時刻が無効である。

MQRC_FEEDBACK_ERROR

(2014, X'7DE') フィードバック・コードが無効である。

MQRC_GLOBAL_UOW_CONFLICT

(2351, X'92F') グローバル作業単位に矛盾がある。

MQRC_GROUP_ID_ERROR

(2258, X'8D2') グループ ID が無効である。

MQRC_HANDLE_IN_USE_FOR_UOW

(2353, X'931') グローバル作業単位のためのハンドルが使用中。

MQRC_HCONN_ERROR

(2018, X'7E2') 接続ハンドルが無効です。

MQRC_HEADER_ERROR

(2142, X'85E') MQ ヘッダー構造体が無効である。

MQRC_HOBJ_ERROR

(2019, X'7E3') オブジェクト・ハンドルが無効です。

MQRC_IIH_ERROR

(2148, X'864') IMS 情報ヘッダー構造体が無効である。

MQRC_INCOMPLETE_GROUP

(2241, X'8C1') メッセージ・グループが不完全である。

MQRC_INCOMPLETE_MSG

(2242, X'8C2') 論理メッセージが不完全である。

MQRC_INCONSISTENT_PERSISTENCE

(2185, X'889') 持続性の指定が不整合である。

MQRC_INCONSISTENT_UOW

(2245, X'8C5') 作業単位の指定が不整合である。

MQRC_LOCAL_UOW_CONFLICT

(2352, X'930') グローバル作業単位とローカル作業単位に矛盾がある。

MQRC_MD_ERROR

(2026, X'7EA') メッセージ記述子が無効である。

MQRC_MDE_ERROR

(2248, X'8C8') メッセージ記述子の拡張子が無効である。

MQRC_MISSING_REPLY_TO_Q

(2027, X'7EB') 応答先キューがないか、または MQPMO_SUPPRESS_REPLYTO が使用されている。

MQRC_MISSING_WIH

(2332, X'91C') メッセージ・データが MQWIH で始まっていない。

MQRC_MSG_FLAGS_ERROR

(2249, X'8C9') メッセージ・フラグが無効である。

MQRC_MSG_SEQ_NUMBER_ERROR

(2250, X'8CA') メッセージ順序番号が無効である。

MQRC_MSG_TOO_BIG_FOR_Q

(2030, X'7EE') メッセージの長さが、キューの最大許容数より大きいです。

MQRC_MSG_TOO_BIG_FOR_Q_MGR

(2031, X'7EF') メッセージ長がキュー・マネージャーの最大許容長より大きいです。

MQRC_MSG_TYPE_ERROR

(2029, X'7ED') メッセージ記述子のメッセージ・タイプが無効である。

MQRC_MULTIPLE_REASONS

(2136, X'858') 複数の理由コードが返されました。

MQRC_NO_DESTINATIONS_AVAILABLE

(2270, X'8DE') 使用可能な宛先キューがない。

MQRC_NOT_OPEN_FOR_OUTPUT

(2039, X'7F7') キューが出力用にオープンされていない。

MQRC_NOT_OPEN_FOR_PASS_ALL

(2093, X'82D') キューが全コンテキスト・パスとしてオープンされていない。

MQRC_NOT_OPEN_FOR_PASS_IDENT

(2094, X'82E') キューが識別コンテキスト・パスとしてオープンされていない。

MQRC_NOT_OPEN_FOR_SET_ALL

(2095, X'82F') キューが全コンテキスト設定用にオープンされていない。

MQRC_NOT_OPEN_FOR_SET_IDENT

(2096, X'830') キューが識別コンテキスト設定用にオープンされていない。

MQRC_OBJECT_CHANGED

(2041, X'7F9') オープンされた後でオブジェクト定義が変更された。

MQRC_OBJECT_DAMAGED

(2101, X'835') オブジェクトが損傷しました。

MQRC_OFFSET_ERROR

(2251, X'8CB') メッセージ・セグメント・オフセットが無効である。

MQRC_OPEN_FAILED

(2137, X'859') オブジェクトが正常にオープンされていません。

MQRC_OPTIONS_ERROR

(2046, X'7FE') オプションが無効であるか、矛盾しています。

MQRC_ORIGINAL_LENGTH_ERROR

(2252, X'8CC') 元の長が無効である。

MQRC_PAGESET_ERROR

(2193, X'891') ページ・セット・データ・セットへのアクセス中にエラーが発生しました。

MQRC_PAGESET_FULL

(2192, X'890') 外部ストレージ・メディアが満杯です。

MQRC_PCF_ERROR

(2149, X'865') PCF 構造体が無効である。

MQRC_PERSISTENCE_ERROR

(2047, X'7FF') 持続性が無効である。

MQRC_PERSISTENT_NOT_ALLOWED

(2048, X'800') キューは永続的なメッセージをサポートしていません。

MQRC_PMO_ERROR

(2173, X'87D') 書き込みメッセージ・オプションの構造体が無効である。

MQRC_PMO_RECORD_FLAGS_ERROR

(2158, X'86E') 書き込みメッセージ・レコード・フラグが無効である。

MQRC_PRIORITY_ERROR

(2050, X'802') メッセージ優先順位が無効である。

MQRC_PUBLICATION_FAILURE

(2502, X'9C6') パブリケーションはどのサブスクライバーにも送達されていない。

MQRC_PUT_INHIBITED

(2051, X'803') 書き込み呼び出しがこのキュー、このキューが解決されるキュー、またはトピックについて使用禁止になっている。

MQRC_PUT_MSG_RECORDS_ERROR

(2159, X'86F') 書き込みメッセージ・レコードが無効である。

MQRC_PUT_NOT_RETAINED

(2479, X'09AF') パブリケーションを保存できなかった。

MQRC_Q_DELETED

(2052, X'804') キューが削除されました。

MQRC_Q_FULL

(2053, X'805') キューには既に最大数のメッセージが入っています。

MQRC_Q_MGR_NAME_ERROR

(2058, X'80A') キュー・マネージャー名が無効であるか、認識されていません。

MQRC_Q_MGR_NOT_AVAILABLE

(2059, X'80B') キュー・マネージャーを接続に使用できません。

MQRC_Q_MGR QUIESCING

(2161, X'871') キュー・マネージャーが静止しています。

MQRC_Q_MGR_STOPPING

(2162, X'872') キュー・マネージャーのシャットダウン中です。

MQRC_Q_SPACE_NOT_AVAILABLE

(2056, X'808') ディスク上にキューのためのスペースがありません。

MQRC_RECONNECT_FAILED

(2548, X'9F4') 再接続後、再接続可能な接続のハンドルの復元中にエラーが発生した。

MQRC_RECS_PRESENT_ERROR

(2154, X'86A') 存在するレコード数が無効です。

MQRC_REPORT_OPTIONS_ERROR

(2061, X'80D') メッセージ記述子のレポート・オプションが無効である。

MQRC_RESOURCE_PROBLEM

(2102, X'836') 使用できるシステム・リソースが不足しています。

MQRC_RESPONSE_RECORDS_ERROR

(2156, X'86C') 応答レコードが無効です。

MQRC_RFH_ERROR

(2334, X'91E') MQRFH または MQRFH2 構造体が無効である。

MQRC_RMH_ERROR

(2220, X'8AC') 参照メッセージ・ヘッダー構造体が無効である。

MQRC_SEGMENT_LENGTH_ZERO

(2253, X'8CD') メッセージ・セグメント内のデータの長さがゼロである。

MQRC_SEGMENTS_NOT_SUPPORTED

(2365, X'93D') セグメントがサポートされていない。

MQRC_SELECTION_NOT_AVAILABLE

2551 (X'09F7') パブリケーションに可能なサブスクライバーが存在しますが、キュー・マネージャーはパブリケーションをサブスクライバーに送信するかどうかを確認できません。

MQRC_STOPPED_BY_CLUSTER_EXIT

(2188, X'88C') クラスター・ワークロード出口によって呼び出しが拒否されました。

MQRC_STORAGE_CLASS_ERROR

(2105, X'839') ストレージ・クラス・エラー。

MQRC_STORAGE_MEDIUM_FULL

(2192, X'890') 外部ストレージ・メディアが満杯です。

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') ストレージが不足しています。

MQRC_SUPPRESSED_BY_EXIT

(2109, X'83D') 出口プログラムにより呼び出しが抑止されました。

MQRC_SYNCPOINT_LIMIT_REACHED

(2024, X'7E8') 現行の作業単位内では、これ以上メッセージを処理できない。

MQRC_SYNCPOINT_NOT_AVAILABLE

(2072, X'818') 同期点サポートが利用できない。

MQRC_TM_ERROR

(2265, X'8D9') トリガー・メッセージ構造体が無効である。

MQRC_TMC_ERROR

(2191, X'88F') 文字トリガー・メッセージ構造体が無効である。

MQRC_UNEXPECTED_ERROR

(2195, X'893') 予期しないエラーが発生しました。

MQRC_UOW_ENLISTMENT_ERROR

(2354, X'932') グローバル作業単位の参加に失敗した。

MQRC_UOW_MIX_NOT_SUPPORTED

(2355, X'933') 作業単位呼び出しの混合はサポートされていない。

MQRC_UOW_NOT_AVAILABLE

(2255, X'8CF') 作業単位がキュー・マネージャーから使用不可。

MQRC_WIH_ERROR

(2333, X'91D') MQWIH 構造体が無効である。

MQRC_WRONG_MD_VERSION

(2257, X'8D1') 提供された MQMD のバージョンが違っている。

MQRC_XQH_ERROR

(2260, X'8D4') 伝送キュー・ヘッダー構造体が無効である。

これらのコードについて詳しくは、[メッセージおよび理由コード](#)を参照してください。

トピックの使用上の注意

1. 以下の注意事項は、トピックの使用に適用されます。

- a. MQPUT を使用してトピックでメッセージをパブリッシュする場合、サブスクライバー・キューで問題が発生した (例えば、キューが満杯である) ために、そのトピックの 1 つ以上のサブスクライバーにパブリケーションを提供できない場合、MQPUT 呼び出しに戻される理由コードおよび配信時の振る舞いは、TOPIC での PMSGDLV または NPMSGDLV 属性の設定によって異なります。
MQRO_DEAD_LETTER_Q が指定されているときに送達不能キューにパブリケーションが配信された場合、または MQRO_DISCARD_MSG が指定されているときにメッセージが廃棄された場合、メッセージは正常に送信されたと思なされることに注意してください。どのパブリケーションも送達されなかった場合、MQPUT は MQRC_PUBLICATION_FAILURE で戻ります。これは次の場合に起こります。

- PMSGDLV または NPMSGDLV (メッセージの持続性によって異なる) が ALL に設定されている TOPIC にメッセージがパブリッシュされ、いずれかのサブスクリプション (永続的かどうかにかかわらず) にパブリケーションを受け取ることができないキューがある。
- メッセージが PMSGDLV または NPMSGDLV (メッセージのパーシスタンスによって異なる) が ALLDUR に設定されている TOPIC にパブリッシュされており、永続サブスクリプションにパブリケーションを受信できないキューが含まれている。

以下のケースでは、パブリケーションを一部のサブスクライバーに送信できなかった場合でも、MQPUT は MQRC_NONE で戻される場合があります。

- PMSGDLV または NPMSGDLV (メッセージの持続性によって異なる) が ALLAVAIL に設定されている TOPIC にメッセージがパブリッシュされ、いずれかのサブスクリプション (永続的かどうかにかかわらず) にパブリケーションを受け取ることができないキューがある。
- PMSGDLV または NPMSGDLV (メッセージの持続性によって異なる) が ALLDUR に設定されている TOPIC にメッセージがパブリッシュされ、非永続サブスクリプションにパブリケーションを受け取ることができないキューがある。

USEDLQ トピック属性を使用すると、パブリケーション・メッセージを正しいサブスクライバー・キューに配信できない場合に送達不能キューを使用するかどうかを決定できます。USEDLQ の使用の詳細については、[DEFINE TOPIC](#) を参照してください。

- 使用されているトピックに対するサブスクライバーが存在しない場合、パブリッシュされるメッセージはどのキューにも送信されずに廃棄されます。メッセージが持続または非持続であるか、あるいはその有効期限が無制限かまたは有効期限が設定されているかは関係ありません。サブスクライバーが存在しない場合は、メッセージは常に廃棄されます。このことの例外となるのは、メッセージが保存される場合です。この場合、メッセージはどのサブスクライバーのキューにも送信されませんが、メッセージはトピックに対して保管され、新規サブスクリプションに対して、または MQSUBRQ を使用して保存パブリケーションを要求するサブスクライバーに対して送達されます。

MQPUT および MQPUT1

MQPUT および MQPUT1 呼び出しを使用して、メッセージをキューに書き込むことができます。どの呼び出しが使用されるかは状況に応じて異なります。

- 同じキュー上に複数のメッセージを配置する場合は MQPUT 呼び出しを使用します。

MQOO_OUTPUT オプションを指定する MQOPEN 呼び出しが最初に発行され、その後に 1 つまたは複数の MQPUT 要求が続き、キューにメッセージを追加します。最後に、キューは MQCLOSE 呼び出しでクローズされます。この結果、MQPUT1 呼び出しを繰り返して使用するよりもパフォーマンスが向上します。

- キュー上に 1 つのメッセージのみを書き込むには MQPUT1 呼び出しを使用します。

この呼び出しは、MQOPEN、MQPUT、および MQCLOSE 呼び出しをまとめて単一の呼び出しにカプセル化するので、発行する必要がある呼び出しの数は最小になります。

宛先キュー

以下の注意事項は、宛先キューの使用に適用されます。

- アプリケーションがメッセージ・グループを使用せずにメッセージ・シーケンスを同じキューに書き込んだ場合、ここで説明する条件が満たされていれば、それらのメッセージの順序は保持されます。ローカル宛先キューとリモート宛先キューの両方に適用される条件と、リモート宛先キューだけに適用される条件とがあります。

ローカル宛先キューおよびリモート宛先キューに適用される条件

- すべての MQPUT 呼び出しが同一作業単位内に含まれている、または作業単位内にまったく含まれていない。

メッセージが 1 つの作業単位内の特定のキューに書き込まれると、他のアプリケーションからのメッセージに、そのキューのメッセージ・シーケンスが散在することがあることに注意してください。

- すべての MQPUT 呼び出しは、同じオブジェクト・ハンドル *Hobj* を使用して実行される。

環境によっては、異なるオブジェクト・ハンドルを使用してもメッセージ順序が保持されることがあります。ただし、呼び出しが同じアプリケーションから実行される場合に限りです。同一アプリケーションの意味は環境によって決定される：

-  **z/OS** z/OS では、アプリケーションは次のとおりです。
 - CICS の場合、CICS タスク。
 - IMS の場合、タスク。
 - z/OS バッチの場合、タスク。
 -  **IBM i** IBM i の場合、アプリケーションはジョブ。
 -  **ALW** AIX, Linux, and Windows の場合、アプリケーションはスレッド。
- どのメッセージも同じ優先順位をもっている。
 - メッセージが、MQOO_BIND_NOT_FIXED が指定されたクラスター・キュー (DefBind キュー属性の値が MQBND_BIND_NOT_FIXED のときは MQOO_BIND_AS_Q_DEF が有効なクラスター・キュー) に書き込まれていない。

リモート宛先キューに適用される追加条件

- 送信側のキュー・マネージャーから宛先キュー・マネージャーへのパスが 1 つしかない。

シーケンス内の一部のメッセージが別のパスを使用する可能性がある場合 (例えば、再構成のため、またはトラフィックのバランスのため、あるいはメッセージ・サイズに基づくパス選択のために) は、宛先キュー・マネージャーでのメッセージの順番は保証できません。
- 送信側、中間、または宛先キュー・マネージャーで、メッセージが一時的に送達不能キューに置かれません。

1 つ以上のメッセージが一時的に送達不能キューに置かれる場合 (例えば、伝送キューまたは宛先キューが一時的に満杯であるために)、メッセージが宛先キューに順序どおりに到達しない可能性があります。
- メッセージがすべて持続メッセージか、あるいはすべて非持続メッセージかのいずれかである。

送信キュー管理者と宛先キュー管理者の間の経路上のチャンネルで、その **NonPersistentMsgSpeed** 属性が MQNPMS_FAST に設定されている場合、非持続メッセージが持続メッセージよりも先に送信される可能性があります。その結果、持続メッセージと非持続メッセージの相対的な順序が保持されなくなることがあります。ただし、持続メッセージ同士および非持続メッセージ同士の相対的順序は保持されます。

これらの条件が満たされない場合、メッセージ・グループを使用して、メッセージの順序を保持することができます。ただし、これには、送信側のアプリケーションと受信側のアプリケーションの両方がメッセージ・グループ化サポートを使用している必要があります。メッセージ・グループの詳細については、以下を参照してください。

- [MQMD - MsgFlags フィールド](#)
- [MQPMO_LOGICAL_ORDER](#)
- [MQGMO_LOGICAL_ORDER](#)

配布リスト

次の注意事項は、配布リストの使用に適用されます。

配布リストは、次の環境でサポートされます。

-  **AIX** AIX
-  **IBM i** IBM i
-  **Linux** Linux
-  **Windows** Windows

および、これらのシステムに接続された IBM MQ MQI clients。

1. バージョン 1 の MQPMO またはバージョン 2 の MQPMO を使用して、各メッセージを配布リストに書き込むことができます。バージョン 1 の MQPMO が使用される (または RecsPresent がゼロに等しいバージョン 2 の MQPMO が使用される) 場合には、書き込みメッセージ・レコードも応答レコードもアプリケーションにより提供されません。メッセージが配布リスト内のいくつかのキューに正常に送信され、それ以外のキューには正常に送信されない場合、エラーが発生したキューを識別することはできません。

書き込みメッセージ・レコードまたは応答レコードがアプリケーションにより提供される場合には、Version フィールドを、MQPMO_VERSION_2 に設定します。

バージョン 2 の MQPMO でも、RecsPresent をゼロにすると、配布リストにない単一キューにメッセージを送信することができます。

2. 完了コード・パラメーターおよび理由コード・パラメーターは、次のように設定されます。

- 配布リスト内のキューへの書き込みがすべて同様に成功または失敗すると、完了コードおよび理由コード・パラメーターがその共通の結果を説明するよう設定されます。MQRR 応答レコード (アプリケーションにより提供されている場合) は、この場合には設定されません。

例えば、すべての書き込みが成功すると、完了コードおよび理由コードは、MQCC_OK および MQRC_NONE に設定されます。すべてのキューが書き込み用に使用禁止になっているため失敗したときは、パラメーターはそれぞれ MQCC_FAILED および MQRC_PUT_INHIBITED に設定されます。

- 配布リスト内のキューに対する書き込みが一部成功した場合または失敗したがその理由が異なる場合は、次のように設定されます。

- 少なくとも 1 つの書き込みが成功した場合、完了コード・パラメーターは MQCC_WARNING に、そしてすべてが失敗した場合には、MQCC_FAILED に設定されます。
- 理由コード・パラメーターは、MQRC_MULTIPLE_REASONS に設定されます。
- 応答レコード (アプリケーションにより提供されている場合) は、配布リスト内のキューごとに、個別の完了コードおよび理由コードに設定されます。

宛先への書き込みが、その宛先のオープンが失敗したために、失敗した場合、応答レコード内の各フィールドは、MQCC_FAILED および MQRC_OPEN_FAILED に設定されます。その宛先は、InvalidDestCount に組み込まれます。

3. 配布リストに指定した宛先のいずれかの解決先がローカル・キューである場合、メッセージは通常形式で (つまり、配布リスト・メッセージとしてではなく) そのキューに登録されます。複数の宛先の解決先が同じローカル・キューである場合は、このローカル・キューには同じメッセージが宛先数分登録されます。

配布リストに指定した宛先がリモート・キューに解決された場合、メッセージは、適切な伝送キュー上に登録されます。いくつかの宛先の解決結果が同じ伝送キューである場合、アプリケーションによって提供される宛先リストの中でそれらの宛先が隣接していない場合でも、これらの宛先を含む単一の配布リスト・メッセージが伝送キューに入れられることがあります。ただし、この処理が行われるのは、伝送キューで配布リスト・メッセージがサポートされている場合だけです (『DistLists』を参照してください)。

伝送キューが配布リストをサポートしていない場合、通常形式のメッセージのコピーが、その伝送キューを使用する各宛先の伝送キュー上に配置されます。

アプリケーション・メッセージ・データをもつ配布リストが伝送キューに対して大きすぎる場合、配布リスト・メッセージは、包含する宛先数の少ない小さな配布リスト・メッセージに分割されます。アプリケーション・メッセージ・データのみがキューに保管される場合、配布リスト・メッセージはまったく使用できず、キュー・マネージャーは、その伝送キューを使用する各宛先用にそのメッセージのコピーを通常形式で生成します。

それぞれの宛先が異なるメッセージ優先順位またはメッセージ持続性を持つ場合 (アプリケーションで MQPRI_PRIORITY_AS_Q_DEF または MQPER_PERSISTENCE_AS_Q_DEF を指定すると、このようなことが起こります) は、メッセージが同じ配布リスト・メッセージ内に保持されません。そのため、キュー・マネージャーは、異なる優先順位および持続性値を収容するのに必要な数の配布リスト・メッセージを生成します。

4. 配布リストへメッセージを書き込むと、メッセージは次のいずれかになる場合があります。

- 1つの配布リスト・メッセージ
- いくつかの小さな配布リスト・メッセージ
- 配布リスト・メッセージと通常メッセージが混在するメッセージ
- 通常メッセージだけ

上記のいずれになるかは、次の内容により異なります。

- リスト内の各宛先がローカル、リモート、またはローカルおよびリモートのいずれであるか。
- 各宛先が同じメッセージ優先順位およびメッセージ持続性を有するかどうか。
- 伝送キューが配布リスト・メッセージを保持できるかどうか。
- 伝送キューの最大メッセージ長が、配布リスト形式でそのメッセージを保管できる長さであるかどうか。

ただし、上記のいずれの場合でも、結果として発生するそれぞれの物理メッセージ(つまり、その書き込みから発生する標準メッセージまたは配布リスト・メッセージ)は、次の場合において、1つのメッセージとしてカウントされます。

- アプリケーションが作業単位内の最大許容メッセージ数を超過したかどうかを検査するとき (**MaxUncommittedMsgs** キュー・マネージャー属性を参照)。
- トリガー発行条件が満たされているかどうかを検査するとき。
- キューのサイズを増加させ、各キューの最大サイズが超過するかどうかを検査するとき。

5. 各キューを個々にオープンするとハンドルが無効になるようなキュー定義の変更(例えば、解決パスの変更)があっても、配布リスト・ハンドルは無効にはなりません。しかし、後続の MQPUT 呼び出しで配布リスト・ハンドルが使用される際、その特定のキューについては失敗します。

ヘッダー

アプリケーション・メッセージ・データの先頭にある1つ以上の IBM MQ ヘッダー構造体を使用してメッセージが書き込まれる場合、キュー・マネージャーはそのヘッダー構造体に対して一定の検査を実行し、それらが有効であるか検証します。キュー・マネージャーがエラーを検出すると、呼び出しは失敗し、該当する理由コードが戻ります。実行される検査は、存在する特定の構造体によって異なります。

- バージョン 2 以降の MQMD が MQPUT または MQPUT1 呼び出し内で使用される場合にだけ、検査が実行されます。メッセージ・データの開始時点で MQMDE が存在していても、バージョン 1 の MQMD が使用されている場合には、これらの検査は実行されません。
- ローカル・キュー・マネージャーによってサポートされない構造体、およびメッセージ内の最初の MQDLH の後の構造体には、妥当性検査は行われません。
- MQDH および MQMDE 構造体の妥当性は、キュー・マネージャーによって完全に検証されます。
- 他の構造体の妥当性は、キュー・マネージャーによって部分的に検証されます(すべてのフィールドが検査されるわけではありません)。

キュー・マネージャーによって実行される一般の検査には、以下が含まれます。

- **StrucId** フィールドが有効でなければならない。
- **Version** フィールドが有効でなければならない。
- **StrucLength** フィールドには、構造体と、その構造体の一部を形成するすべての可変長データが入るだけの十分な大きさの値が指定されなければならない。
- **CodedCharSetId** フィールドは、ゼロや、無効な負の値であってはならない (MQCCSI_DEFAULT、MQCCSI_EMBEDDED、MQCCSI_Q_MGR、および MQCCSI_UNDEFINED は、ほとんどの IBM MQ ヘッダー構造体には無効です)。
- 呼び出しの **BufferLength** パラメーターには、構造体(構造体はメッセージの長さよりも長くなってはならない)が入るだけの大きさを持つ値が指定されなければならない。

構造体での一般検査に加えて、次の条件が満たされている必要があります。

- PCF メッセージ内の構造体の長さの合計は、MQPUT または MQPUT1 呼び出しで指定された **BufferLength** パラメータの長さと同しくなければなりません。PCF メッセージとは、MQFMT_ADMIN、MQFMT_EVENT、または MQFMT_PCF の形式名を持つメッセージのことです。
- IBM MQ 構造体は、切り捨て構造体が許可される次の状況を除いて、切り捨ててはなりません。
 - レポート・メッセージであるメッセージ
 - PCF メッセージ。
 - MQDLH 構造体を含む各メッセージ (最初の MQDLH の後ろの構造体は、切り捨て可能です。この MQDLH の前にある構造体は、切り捨てできません。)
- IBM MQ 構造体は、2 つ以上のセグメントに分割してはなりません。この構造体は 1 つのセグメント内にその全体を含める必要があります。

Buffer

Visual Basic プログラミング言語には、以下の点が適用されます。

- パラメータ **Buffer** のサイズが **BufferLength** パラメータで指定された長さを下回る場合、呼び出しは理由コード MQRC_BUFFER_LENGTH_ERROR で失敗します。
- **Buffer** パラメータはタイプ String として宣言されます。キューに入れるデータが String タイプでない場合は、MQPUT の代わりとしての MQPUTAny 呼び出し。

MQPUTAny 呼び出しは MQPUT 呼び出しと同じパラメータを使用しますが、**Buffer** パラメータはタイプ Any として宣言されるので、どのタイプのデータでもキューに書き込むことができます。ただし、**Buffer** を検査して、サイズが **BufferLength** バイト以上であることを確認することはできません。

C 言語での呼び出し

```
MQPUT (Hconn, Hobj, &MsgDesc, &PutMsgOpts, BufferLength, Buffer,
       &CompCode, &Reason);
```

パラメータを次のように宣言します。

```
MQHCONN  Hconn;           /* Connection handle */
MQHOBJ    Hobj;           /* Object handle */
MQMD      MsgDesc;        /* Message descriptor */
MQPMO     PutMsgOpts;     /* Options that control the action of MQPUT */
MQLONG    BufferLength;   /* Length of the message in Buffer */
MQBYTE    Buffer[n];      /* Message data */
MQLONG    CompCode;       /* Completion code */
MQLONG    Reason;        /* Reason code qualifying CompCode */
```

COBOL での呼び出し

```
CALL 'MQPUT' USING HCONN, HOBJ, MSGDESC, PUTMSGOPTS, BUFFERLENGTH,
                  BUFFER, COMPCODE, REASON.
```

パラメータを次のように宣言します。

```
** Connection handle
01 HCONN          PIC S9(9) BINARY.
** Object handle
01 HOBJ           PIC S9(9) BINARY.
** Message descriptor
01 MSGDESC.
   COPY CMQMDV.
** Options that control the action of MQPUT
01 PUTMSGOPTS.
   COPY CMQPMOV.
** Length of the message in BUFFER
01 BUFFERLENGTH  PIC S9(9) BINARY.
** Message data
01 BUFFER        PIC X(n).
** Completion code
01 COMPCODE      PIC S9(9) BINARY.
```

```

** Reason code qualifying COMPCODE
01 REASON          PIC S9(9) BINARY.

```

PL/I での呼び出し

```

call MQPUT (Hconn, Hobj, MsgDesc, PutMsgOpts, BufferLength, Buffer,
            CompCode, Reason);

```

パラメーターを次のように宣言します。

```

dcl Hconn      fixed bin(31); /* Connection handle */
dcl Hobj       fixed bin(31); /* Object handle */
dcl MsgDesc    like MQMD;     /* Message descriptor */
dcl PutMsgOpts like MQPMO;     /* Options that control the action of
                                MQPUT */
dcl BufferLength fixed bin(31); /* Length of the message in Buffer */
dcl Buffer      char(n);       /* Message data */
dcl CompCode   fixed bin(31); /* Completion code */
dcl Reason     fixed bin(31); /* Reason code qualifying CompCode */

```

高水準アセンブラー呼び出し

```

CALL MQPUT, (HCONN, HOBJ, MSGDESC, PUTMSGOPTS, BUFFERLENGTH, X
            BUFFER, COMPCODE, REASON)

```

パラメーターを次のように宣言します。

HCONN	DS	F	Connection handle
HOBJ	DS	F	Object handle
MSGDESC	CMQMDA	,	Message descriptor
PUTMSGOPTS	CMQPMOA	,	Options that control the action of MQPUT
BUFFERLENGTH	DS	F	Length of the message in BUFFER
BUFFER	DS	CL(n)	Message data
COMPCODE	DS	F	Completion code
REASON	DS	F	Reason code qualifying COMPCODE

Visual Basic での呼び出し

Windows

```

MQPUT Hconn, Hobj, MsgDesc, PutMsgOpts, BufferLength, Buffer, CompCode,
Reason

```

パラメーターを次のように宣言します。

```

Dim Hconn      As Long 'Connection handle'
Dim Hobj       As Long 'Object handle'
Dim MsgDesc    As MQMD 'Message descriptor'
Dim PutMsgOpts As MQPMO 'Options that control the action of MQPUT'
Dim BufferLength As Long 'Length of the message in Buffer'
Dim Buffer      As String 'Message data'
Dim CompCode   As Long 'Completion code'
Dim Reason     As Long 'Reason code qualifying CompCode'

```

MQPUT1 - 1つのメッセージの書き込み

MQPUT1 呼び出しは、キューまたは配布リスト上に、あるいはトピックに1つのメッセージを書き込みます。

キュー、配布リストまたはトピックは、オープンされていなくてもかまいません。

構文

MQPUT1 (*Hconn, ObjDesc, MsgDesc, PutMsgOpts, BufferLength, Buffer, CompCode, Reason*)

パラメーター

Hconn

タイプ: MQHCONN - 入力

このハンドルは、キュー・マネージャーに対する接続を表します。Hconn の値は、先行の MQCONN または MQCONNX 呼び出しによって戻されたものです。

 z/OS 上で CICS アプリケーションを使用する場合は、MQCONN 呼び出しを省略でき、Hconn に次の値を指定できます。

MQHC_DEF_HCONN

デフォルトの接続ハンドル。

ObjDesc

タイプ: MQOD - 入出力

これは、メッセージが追加されるキュー、またはメッセージが公開されるトピックを識別する構造です。詳細は、[449 ページの『MQOD - オブジェクト記述子』](#)を参照してください。

構造がキューである場合、ユーザーは、出力のためのキューをオープンする権限を持っていない限りません。このキューは、モデル・キューであってはなりません。

MsgDesc

タイプ: MQMD - 入出力

この構造体は、送信されるメッセージの属性を記述するものであり、書き込み要求が完了した後で、フィールドバック情報を受け取ります。詳細は、[385 ページの『MQMD - メッセージ記述子』](#)を参照してください。

アプリケーションがバージョン 1 の MQMD を提供している場合、メッセージ・データの接頭部に MQMDE 構造体を付ければ、バージョン 2 の MQMD に存在し、バージョン 1 には存在しない各フィールドの値を指定できます。MQMD 内の Format フィールドを MQFMT_MD_EXTENSION に設定し、MQMDE が存在することを示します。詳細については、[439 ページの『MQMDE - 拡張メッセージ記述子』](#)を参照してください。

アプリケーションは、MQGMO 構造体の `MessageHandle MsgHandle` フィールド、または MQPMO 構造体の `MessageHandle` フィールド NewMsgHandleOriginalMsgHandle もしくは `MessageHandle` フィールドに有効なメッセージハンドルが指定されている場合、MQMD 構造体を指定する必要はありません。これらのフィールドに何も提供されていない場合、メッセージの記述子は、メッセージ・ハンドルに関連した記述子から取られます。

PutMsgOpts

タイプ: MQPMO - 入出力

詳細は [472 ページの『MQPMO - メッセージ書き出しオプション』](#)を参照してください。

BufferLength

タイプ: MQLONG - 入力

メッセージ Buffer の長さゼロは有効であり、メッセージにアプリケーション・データが含まれていないことを示します。この上限は、さまざまな要因によって変化します。BufferLength パラメーターの説明については、[743 ページの『MQPUT - メッセージの書き込み』](#)を参照してください。

Buffer

タイプ: MQBYTEExBufferLength - 入力

これは、送信するアプリケーション・メッセージ・データが入っているバッファーです。バッファーを、メッセージのデータの性質に適した境界に位置合わせします。ほとんどのメッセージ (IBM MQ ヘッダー構造の入ったメッセージを含む) には 4 バイト境界の位置合わせが適していますが、メッセージによってはより厳しい位置合わせを必要とする場合があります。例えば、64 ビット・バイナリー整数を含むメッセージは 8 バイト境界に合わせる必要がある場合があります。

文字データまたは数値データ Buffer を含む場合、パラメータ MsgDesc 内の `charset` EncodingCodedCharSetId および `encoding` フィールドをデータに適した値に設定してください。

これにより、メッセージの受信側は（必要に応じて）データを受信側が使用する文字セットおよびエンコーディングに変換できるようになります。

注：MQPUT1 呼び出しの他のパラメーターはすべて、(**CodedCharSetId** キュー・マネージャー属性およびMQENC_NATIVE で指定した) ローカル・キュー・マネージャーの文字セットおよびエンコードで指定する必要があります。

C プログラミング言語では、パラメーターは、void を示すポインターとして宣言されます。つまり、どのタイプのデータのアドレスもパラメーターとして指定できます。

BufferLength パラメーターがゼロの場合は、**Buffer** は参照されません。この場合、C または System/390 アセンブラーで作成されたプログラムによって渡されるパラメーター・アドレスはヌルのこともあります。

CompCode

タイプ: MQLONG - 出力

完了コード。以下のいずれかです。

MQCC_OK

正常終了。

MQCC_WARNING

警告 (部分完了)。

MQCC_FAILED

呼び出し失敗。

理由

タイプ: MQLONG - 出力

CompCode を限定する理由コード。

CompCode が MQCC_OK の場合:

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_WARNING の場合:

MQRC_MULTIPLE_REASONS

(2136, X'858') 複数の理由コードが返されました。

MQRC_INCOMPLETE_GROUP

(2241, X'8C1') メッセージ・グループが不完全である。

MQRC_INCOMPLETE_MSG

(2242, X'8C2') 論理メッセージが不完全である。

MQRC_PRIORITY_EXCEEDS_MAXIMUM

(2049, X'801') メッセージ優先順位が、サポートされる最大値を超えている。

MQRC_UNKNOWN_REPORT_OPTION

(2104, X'838') メッセージ記述子のレポート・オプションを認識できない。

CompCode が MQCC_FAILED の場合:

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'89C') アダプターが利用できません。

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852') アダプター・サービス・モジュールをロードできません。

MQRC_ALIAS_BASE_Q_TYPE_ERROR

(2001, X'7D1') 別名基本キューのタイプは無効です。

MQRC_API_EXIT_ERROR

(2374, X'946') API 出口で障害が発生しました。

MQRC_API_EXIT_LOAD_ERROR

(2183, X'887') API 出口をロードできません。

MQRC_ASID_MISMATCH

(2157, X'86D') 1 次 ASID とホーム ASID が異なっています。

MQRC_BACKED_OUT

(2003, X'7D3') 作業単位がバックアウトされた。

MQRC_BUFFER_ERROR

(2004, X'7D4') バッファー・パラメーターが無効である。

MQRC_BUFFER_LENGTH_ERROR

(2005, X'7D5') バッファー長パラメーターは無効です。

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') 前の呼び出しが完了する前に MQI 呼び出しが入力されました。

MQRC_CF_NOT_AVAILABLE

(2345, X'929') カップリング・ファシリティが使用できません。

MQRC_CF_STRUC_AUTH_FAILED

(2348, X'92C') カップリング・ファシリティ構造の許可検査に失敗しました。

MQRC_CF_STRUC_ERROR

(2349, X'92D') カップリング・ファシリティ構造が無効です。

MQRC_CF_STRUC_FAILED

(2373, X'945') カップリング・ファシリティ構造体で障害が発生しました。

MQRC_CF_STRUC_IN_USE

(2346, X'92A') カップリング・ファシリティ構造体が使用中です。

MQRC_CF_STRUC_LIST_HDR_IN_USE

(2347, X'92B') カップリング・ファシリティ構造体のリスト・ヘッダーが使用中です。

MQRC_CFGR_ERROR

(2416, X'970') メッセージ・データ内の PCF グループ・パラメーター構造 MQCFGR が無効である。

MQRC_CFH_ERROR

(2235, X'8BB') PCF ヘッダー構造体が無効である。

MQRC_CFIF_ERROR

(2414, X'96E') メッセージ・データ内の PCF 整数フィルターのパラメーター構造が無効である。

MQRC_CFIL_ERROR

(2236, X'8BC') PCF 整数リスト・パラメーター構造または PCIF*64 整数リスト・パラメーター構造が無効である。

MQRC_CFIN_ERROR

(2237, X'8BD') PCF 整数のパラメーター構造体または PCIF*64 整数のパラメーター構造が無効である。

MQRC_CFSF_ERROR

(2415, X'96F') メッセージ・データ内の PCF スtring・フィルターのパラメーター構造が無効である。

MQRC_CFSL_ERROR

(2238, X'8BE') PCF スtring・リストのパラメーター構造体が無効である。

MQRC_CFST_ERROR

(2239, X'8BF') PCF スtringのパラメーター構造体が無効である。

MQRC_CICS_WAIT_FAILED

(2140, X'85C') 待機要求が CICS により拒否された。

MQRC_CLUSTER_EXIT_ERROR

(2266, X'8DA') クラスター・ワークロード出口で障害が発生しました。

MQRC_CLUSTER_RESOLUTION_ERROR

(2189, X'88D') クラスター名の解決に失敗しました。

MQRC_CLUSTER_RESOURCE_ERROR

(2269, X'8DD') クラスター・リソース・エラー。

MQRC_COD_NOT_VALID_FOR_XCF_Q

(2106, X'83A') COD レポート・オプションが XCF キューについて無効である。

MQRC_CONNECTION_BROKEN

(2009, X'7D9') キュー・マネージャーとの接続が失われました。

MQRC_CONNECTION_NOT_AUTHORIZED

(2217, X'8A9') 接続が許可されていません。

MQRC_CONNECTION_QUIESCING

(2202, X'89A') 接続が静止しています。

MQRC_CONNECTION_STOPPING

(2203, X'89B') 接続がシャットダウン中です。

MQRC_CONTENT_ERROR

2554 (X'09FA') メッセージの内容を解析しても、拡張メッセージ・セレクターを使用してメッセージをサブスクライバーに送信できるかどうかを判別できなかった。

MQRC_CONTEXT_HANDLE_ERROR

(2097, X'831') 参照されたキュー・ハンドルがコンテキストを保存しない。

MQRC_CONTEXT_NOT_AVAILABLE

(2098, X'832') 参照されたキュー・ハンドルでコンテキストが使用できない。

MQRC_DATA_LENGTH_ERROR

(2010, X'7DA') データ長パラメーターが無効である。

MQRC_DB2_NOT_AVAILABLE

(2342, X'926') Db2 サブシステムが利用できません。

MQRC_DEF_XMIT_Q_TYPE_ERROR

(2198, X'896') デフォルト伝送キューはローカルではありません。

MQRC_DEF_XMIT_Q_USAGE_ERROR

(2199, X'897') デフォルト伝送キューの使用法エラー。

MQRC_DH_ERROR

(2135, X'857') 配布ヘッダー構造体が無効である。

MQRC_DLH_ERROR

(2141, X'85D') 送達不能ヘッダー構造体が無効である。

MQRC_EPH_ERROR

(2420, X'974') 組み込み PCF 構造が無効である。

MQRC_EXPIRY_ERROR

(2013, X'7DD') 満了時刻が無効である。

MQRC_FEEDBACK_ERROR

(2014, X'7DE') フィードバック・コードが無効である。

MQRC_GLOBAL_UOW_CONFLICT

(2351, X'92F') グローバル作業単位に矛盾がある。

MQRC_GROUP_ID_ERROR

(2258, X'8D2') グループ ID が無効である。

MQRC_HANDLE_IN_USE_FOR_UOW

(2353, X'931') グローバル作業単位のためのハンドルが使用中。

MQRC_HANDLE_NOT_AVAILABLE

(2017, X'7E1') 使用可能なハンドルがなくなりました。

MQRC_HCONN_ERROR

(2018, X'7E2') 接続ハンドルが無効です。

MQRC_HEADER_ERROR

(2142, X'85E') IBM MQ ヘッダー構造体が無効である。

MQRC_IIH_ERROR

(2148, X'864') IMS 情報ヘッダー構造体が無効である。

MQRC_LOCAL_UOW_CONFLICT

(2352, X'930') グローバル作業単位とローカル作業単位に矛盾がある。

MQRC_MD_ERROR

(2026, X'7EA') メッセージ記述子が無効である。

MQRC_MDE_ERROR

(2248, X'8C8') メッセージ記述子の拡張子が無効である。

MQRC_MISSING_REPLY_TO_Q

(2027, X'7EB') 応答先キューがない。

MQRC_MISSING_WIH

(2332, X'91C') メッセージ・データが MQWIH で始まっていない。

MQRC_MSG_FLAGS_ERROR

(2249, X'8C9') メッセージ・フラグが無効である。

MQRC_MSG_SEQ_NUMBER_ERROR

(2250, X'8CA') メッセージ順序番号が無効である。

MQRC_MSG_TOO_BIG_FOR_Q

(2030, X'7EE') メッセージの長さが、キューの最大許容数より大きいです。

MQRC_MSG_TOO_BIG_FOR_Q_MGR

(2031, X'7EF') メッセージ長がキュー・マネージャーの最大許容長より大きいです。

MQRC_MSG_TYPE_ERROR

(2029, X'7ED') メッセージ記述子のメッセージ・タイプが無効である。

MQRC_MULTIPLE_REASONS

(2136, X'858') 複数の理由コードが返されました。

MQRC_NO_DESTINATIONS_AVAILABLE

(2270, X'8DE') 使用可能な宛先キューがない。

MQRC_NOT_AUTHORIZED

(2035, X'7F3') アクセスは許可されません。

MQRC_OBJECT_DAMAGED

(2101, X'835') オブジェクトが損傷しました。

MQRC_OBJECT_IN_USE

(2042, X'7FA') オプションが矛盾するオブジェクトが既に開いています。

MQRC_OBJECT_LEVEL_INCOMPATIBLE

(2360, X'938') オブジェクト・レベルに互換性がありません。

MQRC_OBJECT_NAME_ERROR

(2152, X'868') オブジェクト名が無効です。

MQRC_OBJECT_NOT_UNIQUE

(2343, X'927') オブジェクトが固有ではありません。

MQRC_OBJECT_Q_MGR_NAME_ERROR

(2153, X'869') オブジェクト・キュー・マネージャー名が無効です。

MQRC_OBJECT_RECORDS_ERROR

(2155, X'86B') オブジェクト・レコードが無効です。

MQRC_OBJECT_TYPE_ERROR

(2043, X'7FB') オブジェクト・タイプが無効です。

MQRC_OD_ERROR

(2044, X'7FC') オブジェクト記述子の構造が無効です。

MQRC_OFFSET_ERROR

(2251, X'8CB') メッセージ・セグメント・オフセットが無効である。

MQRC_OPTIONS_ERROR

(2046, X'7FE') オプションが無効であるか、矛盾しています。

MQRC_ORIGINAL_LENGTH_ERROR

(2252, X'8CC') 元の長さが無効である。

MQRC_PAGESET_ERROR

(2193, X'891') ページ・セット・データ・セットへのアクセス中にエラーが発生しました。

MQRC_PAGESET_FULL

(2192, X'890') 外部ストレージ・メディアが満杯です。

MQRC_PCF_ERROR

(2149, X'865') PCF 構造体が無効である。

MQRC_PERSISTENCE_ERROR

(2047, X'7FF') 持続性が無効である。

MQRC_PERSISTENT_NOT_ALLOWED

(2048, X'800') キューは永続的なメッセージをサポートしていません。

MQRC_PMO_ERROR

(2173, X'87D') 書き込みメッセージ・オプションの構造体が無効である。

MQRC_PMO_RECORD_FLAGS_ERROR

(2158, X'86E') 書き込みメッセージ・レコード・フラグが無効である。

MQRC_PRIORITY_ERROR

(2050, X'802') メッセージ優先順位が無効である。

MQRC_PUBLICATION_FAILURE

(2502, X'9C6') パブリケーションはどのサブスクライバーにも送達されていない。

MQRC_PUT_INHIBITED

(2051, X'803') このキューでは書き込み呼び出しが使用禁止になっています。

MQRC_PUT_MSG_RECORDS_ERROR

(2159, X'86F') 書き込みメッセージ・レコードが無効である。

MQRC_Q_DELETED

(2052, X'804') キューが削除されました。

MQRC_Q_FULL

(2053, X'805') キューには既に最大数のメッセージが入っています。

MQRC_Q_MGR_NAME_ERROR

(2058, X'80A') キュー・マネージャー名が無効であるか、認識されていません。

MQRC_Q_MGR_NOT_AVAILABLE

(2059, X'80B') キュー・マネージャーを接続に使用できません。

MQRC_Q_MGR QUIESCING

(2161, X'871') キュー・マネージャーが静止しています。

MQRC_Q_MGR_STOPPING

(2162, X'872') キュー・マネージャーのシャットダウン中です。

MQRC_Q_SPACE_NOT_AVAILABLE

(2056, X'808') ディスク上にキューのためのスペースがありません。

MQRC_Q_TYPE_ERROR

(2057, X'809') キュー・タイプが無効です。

MQRC_RECS_PRESENT_ERROR

(2154, X'86A') 存在するレコード数が無効です。

MQRC_REMOTE_Q_NAME_ERROR

(2184, X'888') リモート・キュー名が無効です。

MQRC_REPORT_OPTIONS_ERROR

(2061, X'80D') メッセージ記述子のレポート・オプションが無効である。

MQRC_RESOURCE_PROBLEM

(2102, X'836') 使用できるシステム・リソースが不足しています。

MQRC_RESPONSE_RECORDS_ERROR

(2156, X'86C') 応答レコードが無効です。

MQRC_RFH_ERROR

(2334, X'91E') MQRFH または MQRFH2 構造体が無効である。

MQRC_RMH_ERROR

(2220, X'8AC') 参照メッセージ・ヘッダー構造体が無効である。

MQRC_SECURITY_ERROR

(2063, X'80F') セキュリティー・エラーが発生しました。

MQRC_SEGMENT_LENGTH_ZERO

(2253, X'8CD') メッセージ・セグメント内のデータの長さがゼロである。

MQRC_SELECTION_NOT_AVAILABLE

2551 (X'09F7') パブリケーションに可能なサブスクライバーが存在しますが、キュー・マネージャーはパブリケーションをサブスクライバーに送信するかどうかを確認できません。

MQRC_STOPPED_BY_CLUSTER_EXIT

(2188, X'88C') クラスター・ワークロード出口によって呼び出しが拒否されました。

MQRC_STORAGE_CLASS_ERROR

(2105, X'839') ストレージ・クラス・エラー。

MQRC_STORAGE_MEDIUM_FULL

(2192, X'890') 外部ストレージ・メディアが満杯です。

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') ストレージが不足しています。

MQRC_SUPPRESSED_BY_EXIT

(2109, X'83D') 出口プログラムにより呼び出しが抑止されました。

MQRC_SYNCPOINT_LIMIT_REACHED

(2024, X'7E8') 現行の作業単位内では、これ以上メッセージを処理できない。

MQRC_SYNCPOINT_NOT_AVAILABLE

(2072, X'818') 同期点サポートが利用できない。

MQRC_TM_ERROR

(2265, X'8D9') トリガー・メッセージ構造体が無効である。

MQRC_TMC_ERROR

(2191, X'88F') 文字トリガー・メッセージ構造体が無効である。

MQRC_UNEXPECTED_ERROR

(2195, X'893') 予期しないエラーが発生しました。

MQRC_UNKNOWN_ALIAS_BASE_Q

(2082, X'822') 別名の基本キューが不明です。

MQRC_UNKNOWN_DEF_XMIT_Q

(2197, X'895') デフォルト伝送キューが不明です。

MQRC_UNKNOWN_OBJECT_NAME

(2085, X'825') オブジェクト名が不明です。

MQRC_UNKNOWN_OBJECT_Q_MGR

(2086, X'826') オブジェクトのキュー・マネージャーが不明です。

MQRC_UNKNOWN_REMOTE_Q_MGR

(2087, X'827') リモート・キュー・マネージャーが不明です。

MQRC_UNKNOWN_XMIT_Q

(2196, X'894') 伝送キューが不明です。

MQRC_UOW_ENLISTMENT_ERROR

(2354, X'932') グローバル作業単位の参加に失敗した。

MQRC_UOW_MIX_NOT_SUPPORTED

(2355, X'933') 作業単位呼び出しの混合はサポートされていない。

MQRC_UOW_NOT_AVAILABLE

(2255, X'8CF') 作業単位がキュー・マネージャーから使用不可。

MQRC_WIH_ERROR

(2333, X'91D') MQWIH 構造体が無効である。

MQRC_WRONG_CF_LEVEL

(2366, X'93E') カップリング・ファシリティ構造のレベルが正しくありません。

MQRC_WRONG_MD_VERSION

(2257, X'8D1') 提供された MQMD のバージョンが違っている。

MQRC_XMIT_Q_TYPE_ERROR

(2091, X'82B') 伝送キューはローカルではありません。

MQRC_XMIT_Q_USAGE_ERROR

(2092, X'82C') 伝送キューの使用方法が正しくありません。

MQRC_XQH_ERROR

(2260, X'8D4') 伝送キュー・ヘッダー構造体が無効である。

これらのコードについて詳しくは、[メッセージおよび理由コード](#)を参照してください。

使用上の注意

- MQPUT および MQPUT1 呼び出しを使用して、メッセージをキューに書き込むことができます。どの呼び出しが使用されるかは状況に応じて異なります。
 - 同じキュー上に複数のメッセージを配置する場合は MQPUT 呼び出しを使用します。
MQOO_OUTPUT オプションを指定する MQOPEN 呼び出しが最初に発行され、その後に 1 つまたは複数の MQPUT 要求が続き、キューにメッセージを追加します。最後に、キューは MQCLOSE 呼び出しでクローズされます。この結果、MQPUT1 呼び出しを繰り返して使用するよりもパフォーマンスが向上します。
 - キュー上に 1 つのメッセージのみを書き込むには MQPUT1 呼び出しを使用します。
この呼び出しは、MQOPEN、MQPUT、および MQCLOSE 呼び出しをまとめて単一の呼び出しにカプセル化するので、発行する必要がある呼び出しの数は最小になります。
- アプリケーションがメッセージ・グループを使用せずにメッセージ・シーケンスを同じキューに書き込んだ場合、特定の条件が満たされていれば、それらのメッセージの順序は保持されます。ただし、ほとんどの環境では MQPUT1 呼び出しはこれらの条件を満たしていないため、メッセージの順序は保たれません。これらの環境では、MQPUT 呼び出しを使用する必要があります。詳細については、『[MQPUT の使用上の注意](#)』を参照してください。
- MQPUT1 呼び出しは、配布リストへの各メッセージの書き込みに使用できます。この点についての一般情報は、MQOPEN および MQPUT 呼び出しの『[使用上の注意](#)』を参照してください。

配布リストは、次の環境でサポートされます。

-  AIX
-  IBM i
-  Linux
-  Windows

および、これらのシステムに接続された IBM MQ クライアント。

MQPUT1 呼び出しを使用する場合、次のような相違点があります。

- MQRR 応答レコードがアプリケーションによって提供されている場合、MQOD 構造体を使用して提供される必要があります。MQPMO 構造体を使用しても提供されません。
- 理由コード MQRC_OPEN_FAILED は、MQPUT1 により応答レコード内に戻されることはありません。キューがオープンに失敗すると、そのキューの応答レコードには、オープン操作の結果発生する理由コードが設定されます。

キューのオープン操作が成功し、完了コード MQCC_WARNING が戻る、そのキューの応答レコード内の完了コードおよび理由コードは、PUT 操作の結果発生する完了コードと理由コードにより置き換えられます。

MQOPEN および MQPUT 呼び出しと同様、キュー・マネージャーは、呼び出しの結果が配布リスト内のすべてのキューについて同じでない場合のみ、応答レコード (提供されている場合) を設定します。これは、呼び出しが終了し、理由コード MQRC_MULTIPLE_REASONS が戻ることにより示されます。

4. クラスター・キューにメッセージを書き込むために MQPUT1 呼び出しを使用すると、MQOPEN 呼び出しで MQOO_BIND_NOT_FIXED を指定した場合と同様の結果になります。
5. アプリケーション・メッセージ・データの先頭にある 1 つ以上の IBM MQ ヘッダー構造体を使用してメッセージが書き込まれる場合、キュー・マネージャーはそのヘッダー構造体に対して一定の検査を実行し、それらが有効であるか検証します。これに関する詳細については、MQPUT 呼び出しの『使用上の注意』を参照してください。
6. 複数の警告状態が発生した場合 (**CompCode** パラメーターを参照)、返される理由コードは、以下のリストのうち該当する最初の理由コードです。
 - a. MQRC_MULTIPLE_REASONS
 - b. MQRC_INCOMPLETE_MSG
 - c. MQRC_INCOMPLETE_GROUP
 - d. MQRC_PRIORITY_EXCEEDS_MAXIMUM または MQRC_UNKNOWN_REPORT_OPTION
7. Visual Basic プログラミング言語には、以下の点が適用されます。
 - ・パラメータ **Buffer** のサイズが **BufferLength** パラメータで指定された長さを下回る場合、呼び出しは理由コード MQRC_BUFFER_LENGTH_ERROR で失敗します。
 - ・**Buffer** パラメーターはタイプ String として宣言されます。キューに入れるデータが String タイプでない場合は、MQPUT1 の代わりに MQPUT1Any を呼び出します。MQPUT1Any 呼び出しは MQPUT1 呼び出しと同じパラメーターを使用しますが、**Buffer** パラメーターはタイプ Any として宣言されるので、どのタイプのデータでもキューに書き込むことができます。ただし、Buffer を検査して、サイズが BufferLength バイト以上であることを確認することはできません。
8. MQPUT1 呼び出しが MQPMO_SYNCPOINT を指定して発行されると、書き込み操作を非同期に完了するように、デフォルトの動作が変更されます。これによって、戻される MQOD および MQMD の構造体内の特定のフィールドに未定義の値が含まれるようになるため、それらのフィールドに依存する一部のアプリケーションの動作が変更される場合があります。アプリケーションでは、MQPMO_SYNC_RESPONSE を指定することで、書き込み操作を同期させて実行すること、および該当するすべてのフィールドの値を完成させることができます。

C 言語での呼び出し

```
MQPUT1 (Hconn, &ObjDesc, &MsgDesc, &PutMsgOpts,  
        BufferLength, Buffer, &CompCode, &Reason);
```

パラメーターを次のように宣言します。

```
MQHCONN  Hconn;           /* Connection handle */  
MQOD      ObjDesc;        /* Object descriptor */  
MQMD      MsgDesc;        /* Message descriptor */  
MQPMO     PutMsgOpts;     /* Options that control the action of MQPUT1 */  
MQLONG    BufferLength;    /* Length of the message in Buffer */  
MQBYTE    Buffer[n];       /* Message data */  
MQLONG    CompCode;       /* Completion code */  
MQLONG    Reason;         /* Reason code qualifying CompCode */
```

COBOL での呼び出し

```
CALL 'MQPUT1' USING HCONN, OBJDESC, MSGDESC, PUTMSGOPTS,  
                   BUFFERLENGTH, BUFFER, COMPCODE, REASON.
```

パラメーターを次のように宣言します。

```

** Connection handle
01 HCONN      PIC S9(9) BINARY.
** Object descriptor
01 OBJDESC.
   COPY CMQODV.
** Message descriptor
01 MSGDESC.
   COPY CMQMDV.
** Options that control the action of MQPUT1
01 PUTMSGOPTS.
   COPY CMQPMOV.
** Length of the message in BUFFER
01 BUFFERLENGTH PIC S9(9) BINARY.
** Message data
01 BUFFER      PIC X(n).
** Completion code
01 COMPCODE     PIC S9(9) BINARY.
** Reason code qualifying COMPCODE
01 REASON       PIC S9(9) BINARY.

```

PL/I での呼び出し

```
call MQPUT1 (Hconn, ObjDesc, MsgDesc, PutMsgOpts, BufferLength, Buffer,
            CompCode, Reason);
```

パラメーターを次のように宣言します。

```

dcl Hconn      fixed bin(31); /* Connection handle */
dcl ObjDesc    like MQOD;    /* Object descriptor */
dcl MsgDesc    like MQMD;    /* Message descriptor */
dcl PutMsgOpts like MQPMO;    /* Options that control the action of
                               MQPUT1 */
dcl BufferLength fixed bin(31); /* Length of the message in Buffer */
dcl Buffer      char(n);       /* Message data */
dcl CompCode   fixed bin(31); /* Completion code */
dcl Reason     fixed bin(31); /* Reason code qualifying CompCode */

```

高水準アセンブラー呼び出し

```
CALL MQPUT1,(HCONN,OBJDESC,MSGDESC,PUTMSGOPTS,BUFFERLENGTH, X
            BUFFER,COMPCODE,REASON)
```

パラメーターを次のように宣言します。

HCONN	DS	F	Connection handle
OBJDESC	CMQODA	,	Object descriptor
MSGDESC	CMQMDA	,	Message descriptor
PUTMSGOPTS	CMQPMOA	,	Options that control the action of MQPUT1
BUFFERLENGTH	DS	F	Length of the message in BUFFER
BUFFER	DS	CL(n)	Message data
COMPCODE	DS	F	Completion code
REASON	DS	F	Reason code qualifying COMPCODE

Visual Basic での呼び出し

Windows

```
MQPUT1 Hconn, ObjDesc, MsgDesc, PutMsgOpts, BufferLength, Buffer,
      CompCode, Reason
```

パラメーターを次のように宣言します。

```

Dim Hconn      As Long   'Connection handle'
Dim ObjDesc    As MQOD   'Object descriptor'
Dim MsgDesc    As MQMD   'Message descriptor'
Dim PutMsgOpts As MQPMO   'Options that control the action of MQPUT1'
Dim BufferLength As Long   'Length of the message in Buffer'
Dim Buffer      As String  'Message data'

```

Dim CompCode	As Long	'Completion code'
Dim Reason	As Long	'Reason code qualifying CompCode'

MQSET - オブジェクト属性の設定

MQSET 呼び出しは、ハンドルで表されるオブジェクトの属性を変更するために使用します。このオブジェクトはキューでなければなりません。

構文

MQSET (*Hconn*, *Hobj*, *SelectorCount*, *Selectors*, *IntAttrCount*, *IntAttrs*, *CharAttrLength*, *CharAttrs*, *Compcode*, *Reason*)

パラメーター

Hconn

タイプ: MQHCONN - 入力

このハンドルは、キュー・マネージャーに対する接続を表します。Hconn の値は、先行の MQCONN または MQCONNX 呼び出しによって戻されたものです。

 z/OS 上で CICS アプリケーションを使用する場合は、MQCONN 呼び出しを省略でき、Hconn に次の値を指定できます。

MQHC_DEF_HCONN

デフォルトの接続ハンドル。

Hobj

タイプ: MQHOBJ - 入力

このハンドルは、属性を設定するキュー・オブジェクトを表します。このハンドルは、MQOO_SET オプションを指定した先行の MQOPEN 呼び出しによって戻されたものです。

SelectorCount

タイプ: MQLONG - 入力

これは、Selectors 配列で提供されるセクターのカウンタです。設定する属性の数です。ゼロは有効な値です。許可される最大数は 256 です。

Selectors

タイプ: MQLONGxSelectorCount - 入力

これは、**SelectorCount** 個の属性セクターで構成される配列です。各セクターは、値を設定する属性 (整数または文字) を示します。

各セクターは、Hobj が表すキューのタイプに対して有効でなければなりません。以降でリストするように、特定の MQIA_* および MQCA_* の値のみが許可されます。

選択子は任意の順序で指定できます。整数属性セクター (MQIA_* セクター) に対応する属性値は、IntAttrs 内に、Selectors 内のセクターの出現順序と同じ順序で指定されている必要があります。文字属性セクター (MQCA_* セクター) に対応する属性値は、CharAttrs 内に、セクターの出現順序と同じ順序で指定されている必要があります。MQIA_* セクターが MQCA_* セクターとインターリーブする可能性があります。このため、各タイプ内での相対順序だけが重要です。

同じセクターを複数回指定することができます。そうした場合は、その特定のセクターに対して最後に指定された値が、実際に反映される値となります。

注:

1. 整数および文字属性セクターは、2 つの異なる範囲に割り振られます。つまり、MQIA_* セクターは、MQIA_FIRST から MQIA_LAST の範囲にあり、MQCA_* セクターは、MQCA_FIRST から MQCA_LAST の範囲にあります。

それぞれの範囲において、定数 MQIA_LAST_USED および MQCA_LAST_USED は、キュー・マネージャーが受け入れる最高値を定義します。

- すべての MQIA_* セレクターが最初に出現する場合は、同じエレメント番号を使用することにより、Selectors および IntAttrs 配列の中の対応するエレメントを指定できます。
- SelectorCount** パラメーターがゼロの場合は、Selectors は参照されません。この場合は、C または System/390 アセンブラー言語で作成されたプログラムによって渡されるパラメーター・アドレスがヌルになる可能性があります。

設定できる属性を、以下の表にリストします。これ以外の属性は、この呼び出しを使用しても設定できません。MQCA_* 属性セレクターの場合、CharAttrs に必要なストリングの長さ (バイト単位) を定義する定数は、括弧内に指定します。

表 590. キューに関する MQSET 属性セレクター		
セレクター	説明	注記
MQCA_TRIGGER_DATA	トリガー・データ (MQ_TRIGGER_DATA_LENGTH)。	
MQIA_DIST_LISTS	配布リスト・サポート。	1
MQIA_INHIBIT_GET	読み取り操作が許されているかどうかを判別します。	
MQIA_INHIBIT_PUT	書き込み操作が許されているかどうかを判別します。	
MQIA_TRIGGER_CONTROL	トリガー制御。	
MQIA_TRIGGER_DEPTH	トリガー項目数。	
MQIA_TRIGGER_MSG_PRIORITY	トリガーのしきい値メッセージ優先順位。	
MQIA_TRIGGER_TYPE	トリガー・タイプ。	

注記:

- 以下のプラットフォームでのみサポートされます。

-  AIX
-  IBM i
-  Linux
-  Windows

および、これらのシステムに接続された IBM MQ MQI clients。

IntAttrCount

タイプ: MQLONG - 入力

これは、IntAttrs 配列のエレメント数です。**Selectors** パラメーター内の MQIA_* セレクターの数以上でなければなりません。何もない場合はゼロが有効な値です。

IntAttrs

タイプ: MQLONGxIntAttrCount - 入力

これは、IntAttrCount 個の整数属性値の配列です。これらの属性値は、Selectors 配列内の MQIA_* セレクターと同じ順序で並んでいなければなりません。

IntAttrCount または **SelectorCount** パラメーターがゼロの場合は、IntAttrs は参照されません。この場合、C または System/390 アセンブラー言語で作成されたプログラムによって渡されるパラメーター・アドレスがヌルのこともあります。

CharAttrLength

タイプ: MQLONG - 入力

これは、**CharAttrs** パラメーターの長さ (バイト単位) であり、少なくとも、Selectors 配列に指定されている各文字属性の長さの合計でなければなりません。Selectors に MQCA_* セレクターが指定されていない場合は、ゼロが有効な値です。

CharAttrs

タイプ: MQCHAR x CharAttrLength - 入力

これは、各文字属性値が連結されて入っている バッファース。 バッファースの長さは、**CharAttrLength** パラメーターによって指定されます。

これらの文字属性は、**Selectors** 配列内の MQCA_* セレクターと同じ順序で 指定されている必要があります。 各文字属性の長さは固定です (**Selectors** を参照)。 属性の設定値に含まれる空白以外の文字数が、属性の定義された長さに満たない場合、その値を **CharAttrs** 右側に空白で埋め、属性の定義された長さに一致させる。

CharAttrLength または **SelectorCount** パラメーターがゼロの場合は、**CharAttrs** は参照されません。 この場合、C または System/390 アセンブラー言語で作成されたプログラムによって渡されるパラメーター・アドレスがヌルのこともあります。

CompCode

タイプ: MQLONG - 出力

完了コード。以下のいずれかです。

MQCC_OK

正常終了。

MQCC_FAILED

呼び出し失敗。

理由

タイプ: MQLONG - 出力

CompCode を限定する理由コード。

CompCode が MQCC_OK の場合:

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_FAILED の場合:

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'89C') アダプターが利用できません。

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852') アダプター・サービス・モジュールをロードできません。

MQRC_API_EXIT_ERROR

(2374, X'946') API 出口で障害が発生しました。

MQRC_API_EXIT_LOAD_ERROR

(2183, X'887') API 出口をロードできません。

MQRC_ASID_MISMATCH

(2157, X'86D') 1 次 ASID とホーム ASID が異なっています。

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') 前の呼び出しが完了する前に MQI 呼び出しが入力されました。

MQRC_CF_NOT_AVAILABLE

(2345, X'929') カップリング・ファシリティが使用できません。

MQRC_CF_STRUC_FAILED

(2373, X'945') カップリング・ファシリティ構造体で障害が発生しました。

MQRC_CF_STRUC_IN_USE

(2346, X'92A') カップリング・ファシリティ構造体が使用中です。

MQRC_CF_STRUC_LIST_HDR_IN_USE

(2347, X'92B') カップリング・ファシリティ構造体のリスト・ヘッダーが使用中です。

MQRC_CHAR_ATTR_LENGTH_ERROR

(2006, X'7D6') 文字属性の長さが無効である。

MQRC_CHAR_ATTRS_ERROR

(2007, X'7D7') 文字属性ストリングが無効である。

MQRC_CICS_WAIT_FAILED

(2140, X'85C') 待機要求が CICS により拒否された。

MQRC_CONNECTION_BROKEN

(2009, X'7D9') キュー・マネージャーとの接続が失われました。

MQRC_CONNECTION_NOT_AUTHORIZED

(2217, X'8A9') 接続が許可されていません。

MQRC_CONNECTION_STOPPING

(2203, X'89B') 接続がシャットダウン中です。

MQRC_DB2_NOT_AVAILABLE

(2342, X'926') Db2 サブシステムが利用できません。

MQRC_HCONN_ERROR

(2018, X'7E2') 接続ハンドルが無効です。

MQRC_HOBJ_ERROR

(2019, X'7E3') オブジェクト・ハンドルが無効です。

MQRC_INHIBIT_VALUE_ERROR

(2020, X'7E4') 取得禁止または書き込み禁止のキュー属性の値が無効です。

MQRC_INT_ATTR_COUNT_ERROR

(2021, X'7E5') 整数属性のカウントが無効です。

MQRC_INT_ATTRS_ARRAY_ERROR

(2023, X'7E7') 整数属性の配列が無効です。

MQRC_NOT_OPEN_FOR_SET

(2040, X'7F8') キューが設定用にオープンされていません。

MQRC_OBJECT_CHANGED

(2041, X'7F9') オープンされた後でオブジェクト定義が変更された。

MQRC_OBJECT_DAMAGED

(2101, X'835') オブジェクトが損傷しました。

MQRC_PAGESET_ERROR

(2193, X'891') ページ・セット・データ・セットへのアクセス中にエラーが発生しました。

MQRC_Q_DELETED

(2052, X'804') キューが削除されました。

MQRC_Q_MGR_NAME_ERROR

(2058, X'80A') キュー・マネージャー名が無効であるか、認識されていません。

MQRC_Q_MGR_NOT_AVAILABLE

(2059, X'80B') キュー・マネージャーを接続に使用できません。

MQRC_Q_MGR_STOPPING

(2162, X'872') キュー・マネージャーのシャットダウン中です。

MQRC_RESOURCE_PROBLEM

(2102, X'836') 使用できるシステム・リソースが不足しています。

MQRC_SELECTOR_COUNT_ERROR

(2065, X'811') セレクターのカウントが無効である。

MQRC_SELECTOR_ERROR

(2067, X'813') 属性選択子が無効です。

MQRC_SELECTOR_LIMIT_EXCEEDED

(2066, X'812') セレクターのカウントが大きすぎる。

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') ストレージが不足しています。

MQRC_SUPPRESSED_BY_EXIT

(2109, X'83D') 出口プログラムにより呼び出しが抑止されました。

MQRC_TRIGGER_CONTROL_ERROR

(2075, X'81B') トリガー制御属性の値が無効です。

MQRC_TRIGGER_DEPTH_ERROR

(2076, X'81C') トリガー項目数属性の値が無効です。

MQRC_TRIGGER_MSG_PRIORITY_ERR

(2077, X'81D') トリガー・メッセージ優先順位属性の値が無効です。

MQRC_TRIGGER_TYPE_ERROR

(2078, X'81E') トリガー・タイプ属性の値が無効です。

MQRC_UNEXPECTED_ERROR

(2195, X'893') 予期しないエラーが発生しました。

これらのコードについて詳しくは、[メッセージおよび理由コード](#)を参照してください。

使用上の注意

1. この呼び出しを使用する場合、アプリケーションは、整数属性の配列、文字属性ストリングの集合、またはその両方を指定できます。エラーが発生しなければ、指定された属性はすべて同時に設定されます。エラーが発生した場合 (セレクターが無効である場合や、属性に無効な値を設定しようとした場合など)、この呼び出しは失敗し、属性は設定されません。

2. 属性の値は、MQINQ 呼び出しを使用して調べることができます。詳細については、[698 ページの『MQINQ - オブジェクト属性の照会』](#)を参照してください。

注：MQINQ 呼び出しを使用して値を照会できる属性のすべてが、MQSET 呼び出しを使用して値を変更できるわけではありません。例えば、プロセス・オブジェクトもキュー・マネージャー属性も、この呼び出しでは設定できません。

3. 属性の変更は、キュー・マネージャーを再始動しても維持されます (一時動的キューへの変更は例外であり、キュー・マネージャーを再始動すると失われます)。
4. モデル・キューの属性は、MQSET 呼び出しを使用して変更できません。ただし、MQOO_SET オプションを指定した MQOPEN 呼び出しを使用してモデル・キューをオープンした場合は、その MQOPEN 呼び出しで作成した動的ローカル・キューの属性を、MQSET 呼び出しを使用して設定することができます。
5. 設定対象のオブジェクトがクラスター・キューの場合、正常にオープンするには、クラスター・キューのローカル・インスタンスが必要です。

オブジェクト属性の詳細については、以下を参照してください。

- [830 ページの『キューの属性』](#)
- [864 ページの『名前リストの属性』](#)
- [866 ページの『プロセス定義の属性』](#)
- [791 ページの『キュー・マネージャーの属性』](#)

C 言語での呼び出し

```
MQSET (Hconn, Hobj, SelectorCount, Selectors, IntAttrCount, IntAttrs,
CharAttrLength, CharAttrs, &CompCode, &Reason);
```

パラメーターを次のように宣言します。

```
MQHCONN  Hconn;           /* Connection handle */
MQHOBJ    Hobj;           /* Object handle */
MQLONG    SelectorCount;  /* Count of selectors */
MQLONG    Selectors[n];   /* Array of attribute selectors */
MQLONG    IntAttrCount;   /* Count of integer attributes */
MQLONG    IntAttrs[n];    /* Array of integer attributes */
MQLONG    CharAttrLength; /* Length of character attributes buffer */
MQCHAR    CharAttrs[n];   /* Character attributes */
MQLONG    CompCode;       /* Completion code */
MQLONG    Reason;         /* Reason code qualifying CompCode */
```

COBOL での呼び出し

```
CALL 'MQSET' USING HCONN, HOBJ, SELECTORCOUNT, SELECTORS-TABLE,  
                  INTATTRCOUNT, INTATTRS-TABLE, CHARATTRLENGTH,  
                  CHARATTRS, COMPCODE, REASON.
```

パラメーターを次のように宣言します。

```
** Connection handle  
01 HCONN          PIC S9(9) BINARY.  
** Object handle  
01 HOBJ          PIC S9(9) BINARY.  
** Count of selectors  
01 SELECTORCOUNT PIC S9(9) BINARY.  
** Array of attribute selectors  
01 SELECTORS-TABLE.  
02 SELECTORS     PIC S9(9) BINARY OCCURS n TIMES.  
** Count of integer attributes  
01 INTATTRCOUNT PIC S9(9) BINARY.  
** Array of integer attributes  
01 INTATTRS-TABLE.  
02 INTATTRS     PIC S9(9) BINARY OCCURS n TIMES.  
** Length of character attributes buffer  
01 CHARATTRLENGTH PIC S9(9) BINARY.  
** Character attributes  
01 CHARATTRS     PIC X(n).  
** Completion code  
01 COMPCODE      PIC S9(9) BINARY.  
** Reason code qualifying COMPCODE  
01 REASON        PIC S9(9) BINARY.
```

PL/I での呼び出し

```
call MQSET (Hconn, Hobj, SelectorCount, Selectors, IntAttrCount,  
            IntAttrs, CharAttrLength, CharAttrs, CompCode, Reason);
```

パラメーターを次のように宣言します。

```
dcl Hconn          fixed bin(31); /* Connection handle */  
dcl Hobj          fixed bin(31); /* Object handle */  
dcl SelectorCount  fixed bin(31); /* Count of selectors */  
dcl Selectors(n)   fixed bin(31); /* Array of attribute selectors */  
dcl IntAttrCount   fixed bin(31); /* Count of integer attributes */  
dcl IntAttrs(n)    fixed bin(31); /* Array of integer attributes */  
dcl CharAttrLength fixed bin(31); /* Length of character attributes  
                                buffer */  
dcl CharAttrs      char(n);       /* Character attributes */  
dcl CompCode       fixed bin(31); /* Completion code */  
dcl Reason         fixed bin(31); /* Reason code qualifying  
                                CompCode */
```

高水準アセンブラー呼び出し

```
CALL MQSET, (HCONN,HOBJ,SELECTORCOUNT,SELECTORS,INTATTRCOUNT, X  
            INTATTRS,CHARATTRLENGTH,CHARATTRS,COMPCODE,REASON)
```

パラメーターを次のように宣言します。

HCONN	DS	F	Connection handle
HOBJ	DS	F	Object handle
SELECTORCOUNT	DS	F	Count of selectors
SELECTORS	DS	(n)F	Array of attribute selectors
INTATTRCOUNT	DS	F	Count of integer attributes
INTATTRS	DS	(n)F	Array of integer attributes
CHARATTRLENGTH	DS	F	Length of character attributes buffer
CHARATTRS	DS	CL(n)	Character attributes
COMPCODE	DS	F	Completion code
REASON	DS	F	Reason code qualifying COMPCODE

Visual Basic での呼び出し

```
MQSET Hconn, Hobj, SelectorCount, Selectors, IntAttrCount, IntAttrs,  
CharAttrLength, CharAttrs, CompCode, Reason
```

パラメーターを次のように宣言します。

```
Dim Hconn As Long 'Connection handle'  
Dim Hobj As Long 'Object handle'  
Dim SelectorCount As Long 'Count of selectors'  
Dim Selectors As Long 'Array of attribute selectors'  
Dim IntAttrCount As Long 'Count of integer attributes'  
Dim IntAttrs As Long 'Array of integer attributes'  
Dim CharAttrLength As Long 'Length of character attributes buffer'  
Dim CharAttrs As String 'Character attributes'  
Dim CompCode As Long 'Completion code'  
Dim Reason As Long 'Reason code qualifying CompCode'
```

MQSETMP - メッセージ・プロパティの設定

MQSETMP 呼び出しを使用して、メッセージ・ハンドルのプロパティを設定したり変更したりします。

構文

MQSETMP (*Hconn, Hmsg, SetPropOpts, Name, PropDesc, Type, ValueLength, Value, Compcode, Reason*)

パラメーター

Hconn

タイプ: MQHCONN - 入力

このハンドルは、キュー・マネージャーに対する接続を表します。

値は、**Hmsg** パラメーターで指定されているメッセージ・ハンドルの作成に使用された接続ハンドルと一致していなければなりません。MQHC_UNASSOCIATED_HCONN を使用してメッセージ・ハンドルが作成された場合は、メッセージ・ハンドルのプロパティを設定するスレッド上で有効な接続を確立しなければなりません。確立しないと、呼び出しは理由コード MQRC_CONNECTION_BROKEN で失敗します。

Hmsg

タイプ: MQHMSG - 入力

これは変更されるメッセージ・ハンドルです。値は、前の MQCRTMH 呼び出しで戻されたものです。

SetPropOpts

タイプ: MQSMPO - 入力

メッセージ・プロパティの設定方法を制御します。

この構造を使用すると、アプリケーションで、メッセージ・プロパティの設定方法を制御するオプションを指定できます。この構造は、MQSETMP 呼び出しの入力パラメーターです。詳細については、[MQSMPO](#) を参照してください。

名前

タイプ: MQCHARV - 入力

これは、設定するプロパティの名前です。

プロパティ名の使用については、[プロパティ名およびプロパティ名に関する制約事項](#)を参照してください。

PropDesc

タイプ: MQPD - 入出力

この構造を使用して、以下を含むプロパティの属性を定義します。

- プロパティがサポートされていない場合に発生すること

- プロパティが属しているメッセージ・コンテキスト
- プロパティがフロー時にコピーされるメッセージ

この構造体の詳細については、[MQPD](#) を参照してください。

タイプ

タイプ: MQLONG - 入力

設定するプロパティのデータ・タイプ。これは以下のいずれかです。

MQTYPE_BOOLEAN

ブール値。 *ValueLength* は 4 でなければなりません。

MQTYPE_BYTE_STRING

バイト・ストリング。 *ValueLength* はゼロ以上でなければなりません。

MQTYPE_INT8

8 ビットの符号付き整数。 *ValueLength* は 1 でなければなりません。

MQTYPE_INT16

16 ビットの符号付き整数。 *ValueLength* は 2 でなければなりません。

MQTYPE_INT32

32 ビットの符号付き整数。 *ValueLength* は 4 でなければなりません。

MQTYPE_INT64

64 ビットの符号付き整数。 *ValueLength* は 8 でなければなりません。

MQTYPE_FLOAT32

32 ビットの浮動小数点数。 *ValueLength* は 4 でなければなりません。

注釈: このタイプは、IBM COBOL for z/OS を使用するアプリケーションではサポートされません。

MQTYPE_FLOAT64

64 ビットの浮動小数点数。 *ValueLength* は 8 でなければなりません。

注釈: このタイプは、IBM COBOL for z/OS を使用するアプリケーションではサポートされません。

MQTYPE_STRING

文字ストリング。 *ValueLength* はゼロ以上または特殊値 MQVL_NULL_TERMINATED でなければなりません。

MQTYPE_NULL

プロパティは存在しますがヌル値です。 *ValueLength* はゼロでなければなりません。

ValueLength

タイプ: MQLONG - 入力

Value パラメーターのプロパティ値の長さ (バイト数)。ヌル値、ストリング、バイト・ストリングの場合のみ、ゼロが有効です。ゼロは、プロパティは存在するものの、値に文字またはバイトが入っていないことを示します。

値は、ゼロ以上であるか、*Type* パラメーターで MQTYPE_STRING が設定されている場合には以下の特殊値でなければなりません。

MQVL_NULL_TERMINATED

値はストリング内で最初に検出されるヌルで区切られます。ヌルはストリングの一部には含められません。この値は、MQTYPE_STRING も設定しないと無効になります。

注: MQVL_NULL_TERMINATED が設定されている場合、ストリングを終了するために使用されるヌル文字は *Value* の文字セットのヌルです。

値

タイプ: MQBYTEValueLength - 入力

設定するプロパティの値。バッファーは、値のデータの性質に適した境界に位置合わせされなければなりません。

C プログラミング言語では、パラメーターは、void を示すポインターとして宣言されます。つまり、どのタイプのデータのアドレスもパラメーターとして指定できます。

ValueLength がゼロの場合は、*Value* は参照されません。この場合、C または System/390 アセンブラーで作成されたプログラムによって渡されるパラメーター・アドレスはヌルのこともあります。

CompCode

タイプ: MQLONG - 出力

完了コード。以下のいずれかです。

MQCC_OK

正常終了。

MQCC_FAILED

呼び出し失敗。

理由

タイプ: MQLONG - 出力

CompCode を限定する理由コード。

CompCode が MQCC_OK の場合、次のようになります。

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_WARNING の場合、次のようになります。

MQRC_RFH_FORMAT_ERROR

(2421, X'0975') プロパティーを含む MQRFH2 フォルダーを構文解析できなかった。

CompCode が MQCC_FAILED の場合、次のようになります。

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'089C') アダプターが利用できません。

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852') アダプター・サービス・モジュールをロードできません。

MQRC_ASID_MISMATCH

(2157, X'86D') 1 次 ASID とホーム ASID が異なっています。

MQRC_BUFFER_ERROR

(2004, X'07D4') 値パラメーターが無効である。

MQRC_BUFFER_LENGTH_ERROR

(2005, X'07D5') 値長パラメーターが無効である。

MQRC_CALL_IN_PROGRESS

(2219, X'08AB') 前の呼び出しが完了する前に MQI 呼び出しが入力された。

MQRC_HMSG_ERROR

(2460, X'099C') メッセージ・ハンドル・ポインターが無効。

MQRC_MSG_HANDLE_IN_USE

(2499, X'09C3') メッセージ・ハンドルがすでに使用中。

MQRC_OPTIONS_ERROR

(2046, X'07FE') オプションが無効であるか、矛盾しています。

MQRC_PD_ERROR

(2482, X'09B2') プロパティー記述子の構造体が無効である。

MQRC_PROPERTY_NAME_ERROR

(2442, X'098A') プロパティー名が無効である。

MQRC_PROPERTY_TYPE_ERROR

(2473, X'09A9') プロパティーのデータ・タイプが無効である。

MQRC_PROP_NUMBER_FORMAT_ERROR

(2472, X'09A8') 値データ中に数字フォーマット・エラーが発生した。

MQRC_SMPO_ERROR

(2463, X'099F') メッセージ・プロパティー設定オプションの構造が無効である。

MQRC_SOURCE_CCSID_ERROR

(2111, X'083F') プロパティ名エンコード文字セット ID が無効である。

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') ストレージが不足しています。

MQRC_UNEXPECTED_ERROR

(2195, X'893') 予期しないエラーが発生しました。

これらのコードについて詳しくは、[メッセージおよび理由コード](#)を参照してください。

C 言語での呼び出し

```
MQSETMP (Hconn, Hmsg, &SetPropOpts, &Name, &PropDesc, Type,  
ValueLength, &Value, &CompCode, &Reason);
```

パラメーターを次のように宣言します。

```
MQHCONN  Hconn;          /* Connection handle */  
MQHMSG   Hmsg;          /* Message handle */  
MQSMPO   SetPropOpts;   /* Options that control the action of MQSETMP */  
MQCHARV  Name;          /* Property name */  
MQPD     PropDesc;      /* Property descriptor */  
MQLONG   Type;          /* Property data type */  
MQLONG   ValueLength;   /* Length of property value in Value */  
MQBYTE   Value[n];      /* Property value */  
MQLONG   CompCode;      /* Completion code */  
MQLONG   Reason;        /* Reason code qualifying CompCode */
```

COBOL での呼び出し

```
CALL 'MQSETMP' USING HCONN, HMSG, SETMSGOPTS, NAME, PROPDSC, TYPE,  
                     VALUELENGTH, VALUE, COMPCODE, REASON.
```

パラメーターを次のように宣言します。

```
** Connection handle  
01 HCONN      PIC S9(9) BINARY.  
** Message handle  
01 HMSG       PIC S9(18) BINARY.  
** Options that control the action of MQSETMP  
01 SETMSGOPTS.  
   COPY CMQSMPOV.  
** Property name  
01 NAME  
   COPY CMQCHRVV.  
** Property descriptor  
01 PROPDSC.  
   COPY CMQPDV.  
** Property data type  
01 TYPE       PIC S9(9) BINARY.  
** Length of property value in VALUE  
01 VALUELENGTH PIC S9(9) BINARY.  
** Property value  
01 VALUE      PIC X(n).  
** Completion code  
01 COMPCODE   PIC S9(9) BINARY.  
** Reason code qualifying COMPCODE  
01 REASON     PIC S9(9) BINARY.
```

PL/I での呼び出し

```
call MQSETMP (Hconn, Hmsg, SetPropOpts, Name, PropDesc, Type, ValueLength,  
              Value, CompCode, Reason);
```

パラメーターを次のように宣言します。

```
dcl Hconn      fixed bin(31); /* Connection handle */  
dcl Hmsg       fixed bin(63); /* Message handle */  
dcl SetPropOpts like MQSMPO; /* Options that control the action of MQSETMP */
```

```

dcl Name      like MQCHARV; /* Property name */
dcl PropDesc  like MQPD; /* Property descriptor */
dcl Type      fixed bin(31); /* Property data type */
dcl ValueLength fixed bin(31); /* Length of property value in Value */
dcl Value     char(n); /* Property value */
dcl CompCode  fixed bin(31); /* Completion code */
dcl Reason    fixed bin(31); /* Reason code qualifying CompCode */

```

高水準アセンブラー呼び出し

```

CALL MQSETMP, (HCONN,HMSG,SETMSGHOPTS,NAME,PROPDESC,TYPE,VALUELENGTH,
              VALUE,COMPCODE,REASON)

```

パラメーターを次のように宣言します。

HCONN	DS	F	Connection handle
HMSG	DS	D	Message handle
SETMSGHOPTS	CMQSMPOA	,	Options that control the action of MQSETMP
NAME	CMQCHRVA	,	Property name
PROPDESC	CMQPDA	,	Property descriptor
TYPE	DS	F	Property data type
VALUELENGTH	DS	F	Length of property value in VALUE
VALUE	DS	CL(n)	Property value
COMPCODE	DS	F	Completion code
REASON	DS	F	Reason code qualifying COMPCODE

MQSTAT - 状況情報の取り出し

MQSTAT 呼び出しを使用して、状況情報を取り出します。戻される状況情報のタイプは、呼び出しで指定されている Type 値によって判別されます。

構文

MQSTAT (*Hconn*, *Type*, *Stat*, *ComPCODE*, *Reason*)

パラメーター

Hconn

タイプ: MQHCONN - 入力

このハンドルは、キュー・マネージャーに対する接続を表します。Hconn の値は、先行の MQCONN または MQCONNX 呼び出しによって戻されたものです。

z/OS 上で CICS アプリケーションを使用する場合は、MQCONN 呼び出しを省略でき、Hconn に次の値を指定できます。

MQHC_DEF_HCONN

デフォルトの接続ハンドル。

タイプ

タイプ: MQLONG - 入力

要求される状況情報のタイプ。有効な値は以下のとおりです。

MQSTAT_TYPE_ASYNC_ERROR

以前の非同期 PUT 操作に関する情報を戻します。

MQSTAT_TYPE_RECONNECTION

再接続に関する情報を戻します。接続が再接続されたり、再接続に失敗したりした場合、この情報は、接続の再接続が開始される原因となった障害を説明しています。

この値は、クライアント接続でのみ有効です。その他のタイプの接続の場合、呼び出しは失敗し、理由コード **MQRC_ENVIRONMENT_ERROR** が返されます

MQSTAT_TYPE_RECONNECTION_ERROR

再接続に関連した直前の障害に関する情報を戻します。接続が再接続に失敗した場合、この情報は、再接続が失敗する原因となった障害を説明しています。

この値は、クライアント接続でのみ有効です。その他のタイプの接続でこれを指定すると、呼び出しは失敗し、理由コード **MQRC_ENVIRONMENT_ERROR** が戻されます。

Stat

タイプ: MQSTS - 入出力

状況情報の構造体。詳細は [578 ページの『MQSTS - 状況報告構造体』](#) を参照してください。

CompCode

タイプ: MQLONG - 出力

完了コード。以下のいずれかです。

MQCC_OK

正常終了。

MQCC_FAILED

呼び出し失敗。

理由

タイプ: MQLONG - 出力

CompCode を限定する理由コード。

CompCode が MQCC_OK の場合:

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_FAILED の場合:

MQRC_API_EXIT_ERROR

(2374, X'946') API 出口で障害が発生しました

MQRC_API_EXIT_LOAD_ERROR

(2183, X'887') API 出口をロードできません。

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') 前の呼び出しが完了する前に MQI 呼び出しが入力されました。

MQRC_CONNECTION_BROKEN

(2009, X'7D9') キュー・マネージャーとの接続が失われました。

MQRC_CONNECTION_STOPPING

(2203, X'89B') 接続がシャットダウン中です。

MQRC_FUNCTION_NOT_SUPPORTED

(2298, X'8FA') 要求された関数は、現在の環境では使用できない。

MQRC_HCONN_ERROR

(2018, X'7E2') 接続ハンドルが無効です。

MQRC_Q_MGR_STOPPING

(2162, X'872') キュー・マネージャーは停止しています。

MQRC_RESOURCE_PROBLEM

(2102, X'836') 使用できるシステム・リソースが不足しています。

MQRC_STAT_TYPE_ERROR

(2430, X'97E') MQSTAT タイプのエラー

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') ストレージが不足しています。

MQRC_STS_ERROR

(2426, X'97A') MQSTS 構造体のエラー

MQRC_UNEXPECTED_ERROR

(2195, X'893') 予期しないエラーが発生しました。

これらのコードについて詳しくは、[メッセージおよび理由コード](#)を参照してください。

使用上の注意

1. MQSTAT_TYPE_ASYNC_ERROR のタイプを指定して MQSTAT を呼び出すと、以前の非同期 MQPUT および MQPUT1 操作に関する情報が戻されます。MQSTAT 呼び出しからの戻りで渡される MQSTS 構造体の内容は、最初に記録された、この接続に関する非同期の警告の情報やエラー情報です。この最初の記録に続いてエラーや警告が発生しても、通常これらの値は変更されません。しかし、MQCC_WARNING の完了コードのエラーが発生した場合は、それ以降は代わりに MQCC_FAILED の完了コードの障害が戻されます。
2. 接続の確立以降または最後の MQSTAT の呼び出し以降にエラーが発生していない場合は、MQSTS 構造体中に MQCC_OK の CompCode と MQRC_NONE の Reason が戻されます。
3. 接続ハンドルの下で処理された非同期呼び出し数のカウントは、PutSuccessCount、PutWarningCount および PutFailureCount の 3 つのカウンター・フィールドによって戻されます。これらのカウンターは、非同期操作が正常に処理されるか、警告を受けるか、または失敗するたびにキュー・マネージャーによって増分されます (会計上の目的で配布リストに挿入する場合、配布リスト当たり 1 回ではなく宛先キュー当たり 1 回ずつカウントされることに注意してください)。カウンターが正の最大値 AMQ_LONG_MAX を超えてインクリメントされることはありません。
4. MQSTAT の呼び出しが正常に実行されると、以前のエラー情報やカウントはリセットされます。
5. MQSTAT の振る舞いは、指定した **MQSTAT Type** パラメーターの値に依存します。
6. **MQSTAT_TYPE_ASYNC_ERROR**
 - a. MQSTAT_TYPE_ASYNC_ERROR のタイプを指定して MQSTAT を呼び出すと、以前の非同期 MQPUT および MQPUT1 操作に関する情報が戻されます。MQSTAT 呼び出しからの戻りで渡される MQSTS 構造体の内容は、最初に記録された、この接続に関する非同期の警告の情報やエラー情報です。この最初の記録に続いてエラーや警告が発生しても、通常これらの値は変更されません。しかし、MQCC_WARNING の完了コードのエラーが発生した場合は、それ以降は代わりに MQCC_FAILED の完了コードの障害が戻されます。
 - b. 接続の確立以降または最後の MQSTAT の呼び出し以降にエラーが発生していない場合は、MQSTS 構造体中に MQCC_OK の CompCode と MQRC_NONE の Reason が戻されます。
 - c. 接続ハンドルの下で処理された非同期呼び出し数のカウントは、PutSuccessCount、PutWarningCount および PutFailureCount の 3 つのカウンター・フィールドによって戻されます。これらのカウンターは、非同期操作が正常に処理されるか、警告を受けるか、または失敗するたびにキュー・マネージャーによって増分されます (会計上の目的で配布リストに挿入する場合、配布リスト当たり 1 回ではなく宛先キュー当たり 1 回ずつカウントされることに注意してください)。カウンターが正の最大値 AMQ_LONG_MAX を超えてインクリメントされることはありません。
 - d. MQSTAT の呼び出しが正常に実行されると、以前のエラー情報やカウントはリセットされます。

MQSTAT_TYPE_RECONNECTION

再接続中にイベント・ハンドラー内で Type を MQSTAT_TYPE_RECONNECTION に設定して MQSTAT を呼び出すとします。以下の例について考えてみます。

クライアントが再接続を試みているか、または再接続に失敗した。

MQSTS 構造体内の CompCode は MQCC_FAILED であり、Reason は MQRC_CONNECTION_BROKEN または MQRC_Q_MGR QUIESCING のいずれかです。ObjectType は MQOT_Q_MGR、ObjectName はキュー・マネージャーの名前、ObjectQMgrName はブランクです。

クライアントの再接続が正常に完了したか、またはクライアントが切断されたことがない。

MQSTS 構造体内の CompCode は MQCC_OK であり、Reason は MQRC_NONE です。

MQSTAT に対する後続の呼び出しでも、同じ結果が戻されます。

MQSTAT_TYPE_RECONNECTION_ERROR

MQI 呼び出しを行って MQRC_RECONNECT_FAILED を受け取ったため、Type を MQSTAT_TYPE_RECONNECTION_ERROR に設定して MQSTAT を呼び出すとします。以下の例について考えてみます。

別のキュー・マネージャーへの再接続中にキューを再オープンしようとしていたときに、許可に失敗した。

MQSTS 構造体内の CompCode は MQCC_FAILED であり、Reason は再接続が失敗した理由 (MQRC_NOT_AUTHORIZED など) です。ObjectType は問題の原因になったオブジェクトのタイプ (MQOT_QUEUE など)、ObjectName はキューの名前、ObjectQMgrName はそのキューを所有しているキュー・マネージャーの名前です。

再接続中にソケット接続エラーが発生した。

MQSTS 構造体内の CompCode は MQCC_FAILED であり、Reason は再接続が失敗した理由 (MQRC_HOST_NOT_AVAILABLE など) です。ObjectType は MQOT_Q_MGR、ObjectName はキュー・マネージャーの名前、ObjectQMgrName はブランクです。

MQSTAT に対する後続の呼び出しでも、同じ結果が戻されます。

C 言語での呼び出し

```
MQSTAT (Hconn, StatType, &Stat, &CompCode, &Reason);
```

パラメーターを次のように宣言します。

```
MQHCONN Hconn;          /* Connection Handle */
MQLONG StatType;         /* Status type */
MQSTS Stat;              /* Status information structure */
MQLONG CompCode;         /* Completion code */
MQLONG Reason;           /* Reason code qualifying CompCode */
```

COBOL での呼び出し

```
CALL 'MQSTAT' USING HCONN, STATTYPE, STAT, COMPCODE, REASON.
```

パラメーターを次のように宣言します。

```
**      Connection handle
01      HCONN      PIC S9(9)          BINARY.
**      Status type
01      STATTYPE   PIC S9(9)          BINARY.
**      Status information
01      STAT.
      COPY CMQSTSV.
**      Completion code
01      COMPCODE   PIC S9(9)          BINARY.
**      Reason code qualifying COMPCODE
01      REASON     PIC S9(9)          BINARY.
```

PL/I での呼び出し

```
call MQSTAT (Hconn, StatType, Stat, Compcode, Reason);
```

パラメーターを次のように宣言します。

```
dc1 Hconn          fixed bin(31); /* Connection handle */
dc1 StatType       fixed bin(31); /* Status type */
dc1 Stat           like MQSTS;     /* Status information structure */
dc1 CompCode       fixed bin(31); /* Completion code */
dc1 Reason         fixed bin(31); /* Reason code qualifying CompCode */
```


System/390 アセンブラー呼び出し

```
CALL MQSTAT, (HCONN, STATTYPE, STAT, COMPCODE, REASON)
```

パラメーターを次のように宣言します。

HCONN	DS	F	Connection handle
STATTYPE	DS	F	Status type
STAT	CMQSTSA,		Status information structure
COMPCODE	DS	F	Completion code
REASON	DS	F	Reason code qualifying COMPCODE

MQSUB - サブスクリプションの登録

MQSUB 呼び出しは、特定のトピックに対するアプリケーションのサブスクリプションを登録するために使用します。

構文

MQSUB (*Hconn*, *SubDesc*, *Hobj*, *Hsub*, *ComPCODE*, *Reason*)

パラメーター

Hconn

タイプ: MQHCONN - 入力

このハンドルは、キュー・マネージャーに対する接続を表します。 *Hconn* の値は、先行の MQCONN または MQCONNX 呼び出しによって戻されたものです。

z/OS 上で CICS アプリケーションを使用する場合は、MQCONN 呼び出しを省略でき、*Hconn* に次の値を指定できます。

MQHC_DEF_HCONN

デフォルトの接続ハンドル。

SubDesc

タイプ: MQSD - 入出力

これは、アプリケーションが登録しようとしている使用中のオブジェクトを識別する構造体です。詳しくは、[551 ページの『MQSD - サブスクリプション記述子』](#)を参照してください。

Hobj

タイプ: MQHOBJ - 入出力

このハンドルは、このサブスクリプションに送信されたメッセージを取得するために設定されたアクセスを表します。これらのメッセージを特定のキューに格納するか、または特定のキューを使用せずにキュー・マネージャーでストレージを管理することができます。

特定のキューを使用するには、サブスクリプションの作成時にサブスクリプションと関連付けなければなりません。これは次の 2 つの方法のうちいずれかで実行できます。

- DEFINE SUB MQSC コマンドを使用し、そのコマンドにキュー・オブジェクトの名前を提供する。
- MQSO_CREATE を指定して MQSUB を呼び出す際に、このハンドルを提供する。

呼び出し上で入力パラメーターとしてこのハンドルを提供する場合は、以下のオプションのうち 1 つ以上を使用してキューの以前の MQOPEN 呼び出しから戻された、有効なオブジェクト・ハンドルでなければなりません。

- MQOO_INPUT_*
- MQOO_BROWSE
- MQOO_OUTPUT (キューがリモートの場合)

上記以外の場合、呼び出しは MQRC_HOBJ_ERROR で失敗します。トピック・オブジェクトに解決する別名キューに対するオブジェクト・ハンドルにすることはできません。その場合、呼び出しは MQRC_HOBJ_ERROR で失敗します。

このサブスクリプションに送信されるメッセージのストレージをキュー・マネージャーに管理させる場合は、MQSO_MANAGED オプションを使用して、サブスクリプションの作成時に設定する必要があります。その後、キュー・マネージャーは呼び出しに対する出力パラメーターとしてこのハンドルを戻します。戻されるハンドルは、管理対象ハンドルと呼ばれます。MQHO_NONE を指定し、MQSO_MANAGED を指定しない場合は、呼び出しは MQRC_HOBJ_ERROR で失敗します。

キュー・マネージャーによって管理対象ハンドルが戻されたら、MQGET または MQCB 呼び出し (ブラウザ・オプションを指定しても指定しなくてもよい)、MQINQ 呼び出し、または MQCLOSE 上で使用できます。MQPUT、MQSUB、MQSET 上で使用することはできません。使用しようとすると、MQRC_NOT_OPEN_FOR_OUTPUT、MQRC_HOBJ_ERROR、または MQRC_NOT_OPEN_FOR_SET で失敗します。

MQSD 構造体で MQSO_RESUME オプションを使用してこのサブスクリプションを再開する場合は、MQSO_MANAGED を MQHO_NONE に設定して、このパラメーター中でハンドルをアプリケーションに戻すことができます。サブスクリプションで管理対象ハンドルを使用しているかどうかにかかわらず戻すことができ、DEFINE SUB を使用し、サブスクリプション・キューに対するハンドルをこのコマンド上で定義して作成したサブスクリプションを提供する際に役立つことがあります。管理用に作成したサブスクリプションを再開する場合、MQOO_INPUT_AS_Q_DEF および MQOO_BROWSE を指定してキューをオープンします。他のオプションを指定する必要がある場合は、アプリケーションで明示的にサブスクリプション・キューをオープンしなければならず、呼び出し上でオブジェクト・ハンドルを提供しなければなりません。キューのオープンに問題がある場合は、呼び出しは MQRC_INVALID_DESTINATION で失敗します。Hobj を提供する場合は、元の MQSUB 呼び出し中の Hobj と等しくなければなりません。つまり、MQOPEN 呼び出しから戻されたオブジェクト・ハンドルを提供する場合、このハンドルは以前に使用されたキューに対するものでなければなりません。同じキューでない場合、呼び出しは MQRC_HOBJ_ERROR で失敗します。

MQSD 構造体で MQSO_ALTER オプションを使用してこのサブスクリプションを変更する場合には、別の Hobj を提供できます。キューに配布されていて、このパラメーターによって以前に識別されたパブリケーションはこのキューに残り、Hobj パラメーターが別のキューを表すようになった場合には、アプリケーションがこれらのメッセージの取り出しを担当します。

表 591. 各種サブスクリプション・オプションでの hobj の使用		
オプション	Hobj	説明
MQSO_CREATE + MQSO_MANAGED	入力では無視される	キュー・マネージャーによって管理されるメッセージのストレージを使用してサブスクリプションを作成します。
MQSO_CREATE	有効なオブジェクト・ハンドル	メッセージの宛先として特定のキューを提供して、サブスクリプションを作成します。
MQSO_RESUME	MQHO_NONE	以前に作成されたサブスクリプションを、管理されていたかどうかにかかわらず再開し、アプリケーションで使用されるオブジェクト・ハンドルをキュー・マネージャーが戻すようにします。
MQSO_RESUME	有効な一致するオブジェクト・ハンドル	メッセージの宛先として特定のキューを使用して以前に作成されたサブスクリプションを再開し、特定のオープン・オプションを指定してオブジェクト・ハンドルを使用します。

表 591. 各種サブスクリプション・オプションでの <i>hobj</i> の使用 (続き)		
オプション	<i>Hobj</i>	説明
MQSO_ALTER + MQSO_MANAGED	MQHO_NONE	以前に特定のキューを使用した既存のサブスクリプションを変更することにより、この時点で管理対象サブスクリプションにします。宛先のクラスは (管理対象かどうかにかかわらず) 変更できません。
MQSO_ALTER	有効なオブジェクト・ハンドル	既存のサブスクリプションを管理対象かどうかにかかわらず変更することにより、この時点で特定のキューを使用するようにします。 MQSO_MANAGED オプションを使用していない場合は、指定されたキューを変更できますが、宛先のクラスは (管理対象であってもなくても) 変更することができません。

提供されたか戻されたかにかかわらず、このサブスクリプションに送信されるパブリケーション・メッセージを受け取る後続の MQGET または MQCB 呼び出しで *Hobj* を指定しなければなりません。

Hobj ハンドルは、それに対して MQCLOSE 呼び出しが発行されたとき、またはハンドルの有効範囲を定義する処理の単位が終了したときに、(アプリケーションが切断されるまで) 無効になります。戻されるオブジェクト・ハンドルの有効範囲は、呼び出しで指定される接続ハンドルの有効範囲と同じです。ハンドルの有効範囲について詳しくは、[Hconn \(MQHCONN\) - 出力](#)を参照してください。 *Hobj* ハンドルの MQCLOSE は、*Hsub* ハンドルには影響を与えません。

Hsub

タイプ: MQHOBJ - 出力

このハンドルは、作成されたサブスクリプションを表します。以下の 2 つの後続操作で使用できます。

- 後続の MQSUBRQ 呼び出しで使用する、サブスクリプションの作成時に MQSO_PUBLICATIONS_ON_REQUEST オプションが使用された場合にパブリケーションが送信されるよう要求できます。
- 後続の MQCLOSE 呼び出しで使用する、作成されているサブスクリプションを除去できます。 *Hsub* ハンドルは、MQCLOSE 呼び出しが発行されたとき、またはハンドルの有効範囲を定義する処理の単位が終了したときに、有効でなくなります。戻されるオブジェクト・ハンドルの有効範囲は、呼び出しで指定される接続ハンドルの有効範囲と同じです。 *Hsub* ハンドルの MQCLOSE は、 *Hobj* ハンドルには影響を与えません。

このハンドルを MQGET または MQCB 呼び出しに渡すことはできません。 **Hobj** パラメーターを使用しなければなりません。MQCLOSE または MQSUBRQ 以外の IBM MQ 呼び出し上でこのハンドルを使用することはできません。このハンドルを他の IBM MQ 呼び出しに渡すと、MQRC_HOBJ_ERROR になります。

CompCode

タイプ: MQLONG - 出力

完了コード。以下のいずれかです。

MQCC_OK

正常終了。

MQCC_WARNING

警告 (部分完了)

MQCC_FAILED

呼び出し失敗

理由

タイプ: MQLONG - 出力

CompCode を限定する理由コード。

CompCode が MQCC_OK の場合、理由コードは以下のとおりです。

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_FAILED の場合、理由コードは以下のいずれかです。

MQRC_CLUSTER_RESOLUTION_ERROR

(2189, X'88D') クラスタ名名の解決に失敗しました。

MQRC_DURABILITY_NOT_ALLOWED

2436 (X'0984') MQSO_DURABLE オプションを使用した MQSUB 呼び出しが失敗した

MQRC_FUNCTION_NOT_SUPPORTED

2298 (X'08FA') 要求された関数は、現在の環境では使用できない。

MQRC_HOBJ_ERROR

2019 (X'07E3') オブジェクト・ハンドル Hobj が無効。

MQRC_IDENTITY_MISMATCH

2434 (X'0982') サブスクリプション名が既存のサブスクリプションと一致する。

MQRC_NOT_AUTHORIZED

2035 (X'07F3') このユーザーはこの操作の実行を許可されていない。

MQRC_NO_SUBSCRIPTION

2428 (X'097C') 指定されたサブスクリプション名が存在しない。

MQRC_OBJECT_STRING_ERROR

2441 (X'0989') Objectstring フィールドが無効。

MQRC_OPTIONS_ERROR

2046 (X'07FE') オプション・パラメーターまたはフィールドに、無効なオプションか、または無効なオプションの組み合わせが含まれている。

MQRC_Q_MGR QUIESCING

2161 (X'0871') キュー・マネージャーが静止中。

MQRC_RECONNECT_Q_MGR REQD

2555 (X'09FB'X) MQCNO_RECONNECT_Q_MGR オプションが必要。

MQRC_RETAINED_MSG_Q_ERROR

2525 (X'09DD') サブスクライブしたトピック・ストリングに対して存在する保存パブリケーションを取得できない。

MQRC_RETAINED_NOT_DELIVERED

2526 (X'09DE') サブスクライブしたトピック・ストリングに対して存在する保存パブリケーションをサブスクリプションの宛先キューに配信できず、送達不能キューにも配信できない。

MQRC_SD_ERROR

2424 (X'0978') サブスクリプション記述子 (MQSD) が無効。

MQRC_SELECTION_NOT_AVAILABLE

2551 (X'09F7') 選択ストリングが IBM MQ セレクター構文規則に従っておらず、使用可能な拡張メッセージ選択プロバイダーがなかった。

MQRC_SELECTION_STRING_ERROR

2519 (X'09D7') MQCHARV 構造体の文書で説明されているとおりに選択ストリングを指定する必要がある。

MQRC_SELECTOR_SYNTAX_ERROR

2459 (X'099B') MQOPEN、MQPUT1、または MQSUB 呼び出しが発行されたが、指定された選択ストリングに構文エラーが含まれる。

MQRC_SUB_USER_DATA_ERROR

2431 (X'097F') SubUserData フィールドが無効。

MQRC_SUB_NAME_ERROR

2440 (X'0988') SubName フィールドが無効。

MQRC_SUB_ALREADY_EXISTS

2432 (X'0980') サブスクリプションが既に存在する。

MQRC_SUB_USER_DATA_ERROR

2431 (X'097F') SubUserData フィールドが無効。

MQRC_TOPIC_STRING_ERROR

2425 (X'0979') トピック・ストリングが無効。

MQRC_UNKNOWN_OBJECT_NAME

2085 (X'0825') MQSD ObjectName フィールド内の指定されたオブジェクトが見つからない。

MQRC_SUB_JOIN_NOT_ALTERABLE

29440 (X'7300') サブスクリプション共有モードは、既存のサブスクリプションと互換性がありません。このエラーは、非 JMS アプリケーションで JMS 2.0 共有サブスクリプションの再開を試行するときに返されることがあります。

これらのコードについて詳しくは、[メッセージおよび理由コード](#)を参照してください。

使用上の注意

- サブスクリプションはトピックに作成され、このトピックは事前定義済みのトピック・オブジェクトの短縮名かトピック・ストリングのフルネームを使用して名前指定されるか、2つの部分を連結して形成されます。551 ページの『MQSD - サブスクリプション記述子』の *ObjectName* と *ObjectString* の説明を参照してください。
- キュー・マネージャーは、MQSUB 呼び出しが発行されるときにセキュリティ検査を行い、アクセス許可が出される前に、アプリケーションの実行に使用されるユーザー ID に適切なレベルの権限が付与されていることを確認します。該当するトピック・オブジェクトがトピック階層内に入れられ、権限検査がこのトピック・オブジェクトに対して行われて、サブスクライブする権限が設定されているか確認します。MQSO_MANAGED オプションを使用しないと、権限検査が宛先キューに対して行われて、出力に関する権限が設定されているか確認します。MQSO_MANAGED オプションを使用すると、出力または照会アクセスに関する権限検査が管理対象キューに対して行われません。
- 入力として Hobj を提供しない場合、MQSUB 呼び出しは、オブジェクト・ハンドル (Hobj) とサブスクリプション・ハンドル (Hsub) の2つのハンドルを割り振ります。
- バックアウトしきい値や余分なバックアウト・リキュー名などの属性を検出するために、MQSO_MANAGED オプションの使用時に MQSUB 呼び出し上で戻される Hobj を照会できます。管理対象キューの名前も照会できますが、このキューを直接オープンしようとししないでください。
- サブスクリプションをグループ化して、そのグループの複数のサブスクリプションが1つのパブリケーションと一致した場合でもこのパブリケーションのみサブスクリプションのグループに配布できます。サブスクリプションをグループ化するには、MQSO_GROUP_SUB オプションを使用します。またサブスクリプションをグループ化するには、以下の条件を満たさなければなりません。
 - 同じキュー・マネージャー上で (MQSO_MANAGED オプションを使用していない) 同じ名前のキューを使用している - MQSUB 呼び出し上の Hobj パラメーターで表される
 - 同じ SubCorrelId を共有している
 - SubLevel が同じこれらの属性は、そのグループに含まれると見なされるサブスクリプションのセットを定義します。さらにサブスクリプションがグループ化される場合、これらは変更できない属性です。SubLevel を変更すると MQRC_SUBLEVEL_NOT_ALTERABLE になり、他の属性 (サブスクリプションがグループ化されていない場合には変更できる) を変更すると MQRC_GROUPING_NOT_ALTERABLE になります。
- MQSUB 呼び出しの正常終了は、アクションが完了したことを意味しません。この呼び出しが完了したことを確認するには、「[分散ネットワーク向け非同期コマンドの終了確認](#)」の **DEFINE SUB** ステップを参照してください。
- MQSO_RESUME オプションを使用する MQSUB 呼び出しから戻る際に、MQSD 中のフィールドが記入されます。戻された MQSD は、MQSD に適用されているサブスクリプションに必要な変更を加えて、

MQSO_ALTER オプションを使用する MQSUB 呼び出しに直接渡すことができます。表に示されているように、一部のフィールドには特別な考慮事項があります。

表 592. MQSD 内のフィールドに関する特別な考慮事項	
MQSD のフィールド名	特別な考慮事項
アクセスまたは作成オプション	MQSUB 呼び出しからの戻り時に、一部のオプションを再設定できます。後で MQSUB 呼び出しの中で MQSD を再利用する場合、必要なオプションは明示的に設定する必要があります。
耐久性オプション、宛先オプション、登録オプション、およびワイルドカード・オプション	これらのオプションは、該当する場合に設定されます。
パブリケーション・オプション	これらのオプションは、該当する場合に設定されます。ただし MQSO_NEW_PUBLICATIONS_ONLY は例外で、MQSO_CREATE のみに適用されます。
その他のオプション	これらのオプションは、MQSUB 呼び出しからの戻り時に変更されません。これらのオプションは、API 呼び出しが発行される方法を制御し、サブスクリプションと共に格納されません。これらは、MQSD を再利用する後続の MQSUB 呼び出しで必要に応じて設定する必要があります。
ObjectName	この入力専用フィールドは、MQSUB 呼び出しからの戻り時に変更されません。
ObjectString	この入力専用フィールドは、MQSUB 呼び出しからの戻り時に変更されません。バッファが提供される場合、使用されるトピック名のフルネームが <i>ResObjectString</i> フィールドに戻されます。
AlternateUserId および AlternateSecurityId	これらの入力専用フィールドは、MQSUB 呼び出しからの戻り時に変更されません。これらのオプションは、API 呼び出しが発行される方法を制御し、サブスクリプションと共に格納されません。これらは、MQSD を再利用する後続の MQSUB 呼び出しで必要に応じて設定する必要があります。
SubExpiry	MQSO_RESUME オプションを使用する MQSUB 呼び出しから戻る際、このフィールドはサブスクリプションの有効期限の残り時間ではなく、元の有効期限に設定されます。後で MQSO_ALTER オプションを使用する MQSUB 呼び出しで MQSD を再利用する場合は、サブスクリプションの期限切れをリセットして、カウントダウンを再開してください。
SubName	このフィールドは MQSUB 呼び出しの入力フィールドで、出力時に変更されません。

表 592. MQSD 内のフィールドに関する特別な考慮事項 (続き)	
MQSD のフィールド名	特別な考慮事項
SubUserData および SelectionString	バッファが提供されている場合は、MQSO_RESUME オプションを使用する MQSUB 呼び出しからの出力上にこれらの可変長フィールドが戻され、VSBufSize 中に正のバッファ長も戻されます。バッファが提供されていない場合は、長さだけが MQCHARV の VSLength フィールドに戻されます。提供されているバッファがフィールドを戻すのに必要なスペースより小さい場合は、提供されているバッファに VSBufSize のバイト数のみ戻されます。 後で MQSO_ALTER オプションを使用する MQSUB 呼び出しの中で MQSD を再利用する場合に、バッファが提供されておらず、VSLength がゼロ以外に設定されていると、その長さがフィールドの既存の長さと一致する場合には、フィールドに変更は加えられません。
SubCorrelId および PubAccountingToken	MQSO_SET_CORREL_ID を使用しない場合は、キュー・マネージャーによって SubCorrelId が生成されます。MQSO_SET_IDENTITY_CONTEXT を使用しない場合は、キュー・マネージャーによって PubAccountingToken が生成されます。 これらのフィールドは、MQSO_RESUME オプションを使用する MQSUB 呼び出しからの MQSD 中に戻されます。キュー・マネージャーによって生成される場合は、生成された値は MQSO_CREATE または MQSO_ALTER オプションを使用する MQSUB 呼び出し上に戻されます。
PubPriority, SubLevel & PubApplIdentityData	これらのフィールドは MQSD 中に戻されます。
ResObjectString	バッファが提供されている場合、この出力専用フィールドは MQSD 中に戻されます。

C 言語での呼び出し

```
MQSUB (Hconn, &SubDesc, &Hobj, &Hsub, &CompCode, &Reason)
```

パラメーターを次のように宣言します。

```
MQHCONN Hconn;    /* Connection handle */
MQSD     SubDesc;  /* Subscription descriptor */
MQHOBJ   Hobj;     /* Object handle */
MQHOBJ   Hsub;     /* Subscription handle */
MQLONG   CompCode; /* Completion code */
MQLONG   Reason;   /* Reason code qualifying CompCode */
```

COBOL での呼び出し

```
CALL 'MQSUB' USING HCONN, SUBDESC, HOBJ, HSUB, COMPCODE, REASON.
```

パラメーターを次のように宣言します。

```
**      Connection handle
01 HCONN    PIC S9(9) BINARY.
**      Subscription descriptor
01 SUBDESC.
```

```

COPY CMQSDV.
** Object handle
01 HOBJ      PIC S9(9) BINARY.
** Subscription handle
01 HSUB      PIC S9(9) BINARY.
** Completion code
01 COMPCODE  PIC S9(9) BINARY.
** Reason code qualifying COMPCODE
01 REASON    PIC S9(9) BINARY.

```

PL/I での呼び出し

```
call MQSUB (Hconn, SubDesc, Hobj, Hsub, CompCode, Reason)
```

パラメーターを次のように宣言します。

```

dcl Hconn      fixed bin(31); /* Connection handle */
dcl SubDesc    like MQSD;    /* Subscription descriptor */
dcl Hobj       fixed bin(31); /* Object handle */
dcl Hsub       fixed bin(31); /* Subscription handle */
dcl CompCode   fixed bin(31); /* Completion code */
dcl Reason     fixed bin(31); /* Reason code qualifying CompCode */

```

高水準アセンブラー呼び出し

```
CALL MQSUB, (HCONN, SUBDESC, HOBJ, HSUB, COMPCODE, REASON)
```

パラメーターを次のように宣言します。

```

HCONN      DS      F  Connection handle
SUBDESC    CMQSDA  ,  Subscription descriptor
HOBJ       DS      F  Object handle
HSUB       DS      F  Subscription handle
COMPCODE   DS      F  Completion code
REASON     DS      F  Reason code qualifying COMPCODE

```

MQSUBRQ - サブスクリプション要求

MQSUBRQ 呼び出しは、サブスクライバーが MQSQ_PUBLICATIONS_ON_REQUEST で登録された場合に、保存パブリケーションを要求するために使用します。

構文

MQSUBRQ (*Hconn*, *Hsub*, *Action*, *SubRqOpts*, *Compcode*, *Reason*)

パラメーター

Hconn

タイプ: MQHCONN - 入力

このハンドルは、キュー・マネージャーに対する接続を表します。 *Hconn* の値は、先行の MQCONN または MQCONNX 呼び出しによって戻されたものです。

z/OS 上で CICS アプリケーションを使用する場合は、MQCONN 呼び出しを省略でき、*Hconn* に次の値を指定できます。

MQHC_DEF_HCONN

デフォルトの接続ハンドル。

Hsub

タイプ: MQHOBJ - 入力

このハンドルは、更新が要求されるサブスクリプションを表します。 *Hsub* の値は、前の MQSUB 呼び出しから戻されたものです。

アクション

タイプ: MQLONG - 入力

このパラメーターは、サブスクリプションで要求される特定のアクションを制御します。以下の値を指定する必要があります。

MQSR_ACTION_PUBLICATION

このアクションは、指定されたトピックに関する更新パブリケーションが送信されることを要求します。これを使用できるのは、サブスクライバーがサブスクリプションを行う際に MQSUB 呼び出しでオプション MQSO_PUBLICATIONS_ON_REQUEST を指定した場合だけです。トピックに関する保存パブリケーションがキュー・マネージャー中にある場合は、サブスクライバーに送信されません。ない場合は、その呼び出しは失敗します。保存されたパブリケーションがアプリケーションに送られると、そのパブリケーションの MQIsRetained メッセージ・プロパティによって示されます。

Hsub パラメーターで表される既存のサブスクリプション中のトピックにワイルドカードを含めることができるので、サブスクライバーが複数の保存パブリケーションを受け取る可能性があります。

SubRqOpts

タイプ: MQSRO - 入出力

これらのオプションは、MQSUBRQ のアクションを制御します。詳しくは、[575 ページの『MQSRO - サブスクリプション要求オプション』](#)を参照してください。

必須オプションがない場合、C アセンブラーまたは S/390 アセンブラーで作成されたプログラムでは、MQSRO 構造のアドレスを指定せずに、ヌル・パラメーター・アドレスを指定することができます。

CompCode

タイプ: MQLONG - 出力

完了コード。以下のいずれかです。

MQCC_OK

正常終了。

MQCC_WARNING

警告 (部分完了)

MQCC_FAILED

呼び出し失敗

理由

タイプ: MQLONG - 出力

CompCode を限定する理由コード。

CompCode が MQCC_OK の場合:

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_FAILED の場合:

MQRC_FUNCTION_NOT_SUPPORTED

2298 (X'08FA') 要求された関数は、現在の環境では使用できない。

MQRC_NO_RETAINED_MSG

2437 (X'0985') このトピックに関する現在格納中の保存パブリケーションがない。

MQRC_OPTIONS_ERROR

2046 (X'07FE') オプション・パラメーターまたはフィールドに、無効なオプションか、または無効なオプションの組み合わせが含まれている。

MQRC_Q_MGR QUIESCING

2161 (X'0871') キュー・マネージャーが静止中。

MQRC_SRO_ERROR

2438 (X'0986') MQSUBRQ 呼び出しで、サブスクリプション要求オプション MQSRO が無効である。

MQRC_RETAINED_MSG_Q_ERROR

2525 (X'09DD') サブスクライブしたトピック・ストリングに対して存在する保存パブリケーションを取得できない。

MQRC_RETAINED_NOT_DELIVERED

2526 (X'09DE') サブスクライブしたトピック・ストリングに対して存在する保存パブリケーションをサブスクリプションの宛先キューに配信できず、送達不能キューにも配信できない。

これらのコードについて詳しくは、[メッセージおよび理由コード](#)を参照してください。

使用上の注意

以下の使用上の注意は、アクション・コード MQSR_ACTION_PUBLICATION の使用に適用されます。

1. この verb が正常に完了した場合、指定されたサブスクリプションと一致する保存パブリケーションは、サブスクリプションに送信されているので、サブスクリプションを作成した元の MQSUB verb 上で戻された Hobj を使用する MQGET または MQCB を使って受け取ることができます。
2. サブスクリプションを作成した元の MQSUB verb によってサブスクライブされたトピックにワイルドカードが含まれている場合には、複数の保存パブリケーションが送信されることがあります。この呼び出しの結果として送信されたパブリケーションの数は、SubRqOpts 構造中の NumPubs フィールド中に記録されます。
3. この verb が理由コード MQRC_NO_RETAINED_MSG で完了する場合は、指定されたトピックに関する保存パブリケーションは現在ありません。
4. この verb が理由コード MQRC_RETAINED_MSG_Q_ERROR または MQRC_RETAINED_NOT_DELIVERED で完了する場合は、指定されたトピックに関する保存パブリケーションは現在ありますが、配布できなかったことを意味するエラーが発生しています。
5. この呼び出しを行う前に、アプリケーションにはトピックへの現行のサブスクリプションがなければなりません。アプリケーションの前のインスタンスでサブスクリプションが行われ、そのサブスクリプションに対する有効なハンドルを使用できない場合は、そのアプリケーションで最初に MQSO_RESUME オプションを指定して MQSUB を呼び出し、サブスクリプションに対するハンドルを入手してこの呼び出しで使用するようにしなければなりません。
6. パブリケーションは、このアプリケーションの現行のサブスクリプションに使用するために登録された宛先に送信されます。パブリケーションを他の場所へ送信する必要がある場合は、最初に MQSO_ALTER オプションを指定して MQSUB 呼び出しを使用し、サブスクリプションに変更を加えなければなりません。

C 言語での呼び出し

```
MQSUB (Hconn, Hsub, Action, &SubRqOpts, &CompCode, &Reason)
```

パラメーターを次のように宣言します。

```
MQHCONN Hconn;      /* Connection handle */
MQHOBJS Hsub;        /* Subscription handle */
MQLONG  Action;      /* Action requested by MQSUBRQ */
MQSRO   SubRqOpts;    /* Options that control the action of MQSUBRQ */
MQLONG  CompCode;     /* Completion code */
MQLONG  Reason;       /* Reason code qualifying CompCode */
```

COBOL での呼び出し

```
CALL 'MQSUBRQ' USING HCONN, HSUB, ACTION, SUBRQOPTS, COMPCODE, REASON.
```

パラメーターを次のように宣言します。

```
** Connection handle
01 HCONN PIC S9(9) BINARY.
** Subscription handle
01 HSUB PIC S9(9) BINARY.
** Action requested by MQSUBRQ
```

```

01 ACTION PIC S9(9) BINARY.
** Options that control the action of MQSUBRQ
01 SUBRQOPTS.
COPY CMQSR0V.
** Completion code
01 COMPCODE PIC S9(9) BINARY.
** Reason code qualifying COMPCODE
01 REASON PIC S9(9) BINARY.

```

PL/I での呼び出し

```
call MQSUBRQ (Hconn, Hsub, Action, SubRqOpts, CompCode, Reason)
```

パラメーターを次のように宣言します。

```

dcl Hconn fixed bin(31); /* Connection handle */
dcl Hsub fixed bin(31); /* Subscription handle */
dcl Action fixed bin(31); /* Action requested by MQSUBRQ */
dcl SubRqOpts like MQSR0; /* Options that control the action of MQSUBRQ */
dcl CompCode fixed bin(31); /* Completion code */
dcl Reason fixed bin(31); /* Reason code qualifying CompCode */

```

高水準アセンブラー呼び出し

```
CALL MQSUBRQ,(HCONN, HSUB, ACTION, SUBRQOPTS,COMPCODE,REASON)
```

パラメーターを次のように宣言します。

```

HCONN DS F Connection handle
HSUB DS F Subscription handle
ACTION DS F Action requested by MQSUBRQ
SUBRQOPTS CMQSR0A , Options that control the action of MQSUBRQ
COMPCODE DS F Completion code
REASON DS F Reason code qualifying COMPCODE

```

オブジェクトの属性

この一連のトピックでは、MQINQ 関数呼び出しの対象となり得る IBM MQ オブジェクトだけをリストし、照会可能な属性や使用されるセレクターの詳細を示します。

キュー・マネージャーの属性

一部のキュー・マネージャー属性は、特定の実装に固定されています。その他の属性は、MQSC コマンド **ALTER QMGR** を使用して変更できます。

属性は、**DISPLAY QMGR** コマンドを使用して表示することもできます。大部分のキュー・マネージャー属性は、特別な MQOT_Q_MGR オブジェクトを開き、返されたハンドルを指定して MQINQ 呼び出しを使用することにより、照会することができます。

次の表では、キュー・マネージャーに固有の属性が要約されています。属性の説明は、アルファベット順に掲載しています。

注：このセクションで示されている属性の名前は、MQINQ 呼び出しで使用する記述名です。この名前は、PCF コマンドの場合も同じです。MQSC コマンドを使用して属性を定義、変更、または表示する場合、代替の短い名前が使用されます。詳細については、[MQSC コマンド](#)を参照してください。

表 593. キュー・マネージャーの属性	
属性	説明
AccountingConnOverride	アカウントリング設定をオーバーライドする。
AccountingInterval	中間アカウント・レコードを書き込む頻度。
ActivityConnOverride	アクティビティ設定をオーバーライドします。
ActivityTrace	IBM MQ MQI アプリケーションのアクティビティ・トレースの収集を制御します。

表 593. キュー・マネージャーの属性 (続き)	
属性	説明
AdoptNewMCACheck	新しい MCA を受け入れるかどうか判別するためにチェックするエレメント。
 AdoptNewMCAType	特定のチャンネル・タイプの MCA の孤立したインスタンスを自動的に再始動するかどうか。
AlterationDate	定義が最後に変更された日付。
AlterationTime	定義が最後に変更された時刻。
AuthorityEvent	 から IBM MQ 10.0.0、認証イベントの生成の有無を制御します。
 AuthorityEventScope	有効化時に認証イベントの範囲を制御します。
 BridgeEvent	ブリッジ・イベントの属性を制御します。
ChannelAutoDef	自動チャンネル定義が許可されているかどうかを制御する。
ChannelAutoDefEvent	チャンネル自動定義のイベントが生成されるかどうかを制御する。
ChannelAutoDefExit	チャンネル自動定義用のユーザー出口の名前。
ChannelEvent	チャンネル・イベントの制御属性。
ChannelInitiatorControl	チャンネル開始者の制御属性。
ChannelMonitoring	チャンネルのオンライン監視データ。
ChannelStatistics	チャンネルの統計データのコレクションを制御する。
 ChinitAdapters	IBM MQ 呼び出しを処理するためのアダプター・サブタスクの数。
 ChinitDispatchers	チャンネル・イニシエーターに使用するディスパッチャーの数。
	IBM で使用するために予約済み。
 ChinitTraceAutoStart	チャンネル開始プログラム・トレースを自動的に開始するかどうか。
 ChinitTraceTableSize	チャンネル開始プログラムのトレース・データ・スペースのサイズ。
ClusterSenderMonitoringDefault	クラスタ送信チャンネルのオンライン監視データデフォルト
ClusterSenderStatistics	クラスター送信側チャンネルの統計モニター情報のコレクションを制御する。
ClusterWorkloadData	クラスター・ワークロード出口についてのユーザー・データ。
ClusterWorkloadExit	クラスター・ワークロード管理のためのユーザー出口の名前。
ClusterWorkloadLength	クラスター・ワークロード出口に引き渡されるメッセージ・データの最大長。
CLWLMRUChannels	クラスターのワークロード分散における直近使用チャンネル数
CLWLUseQ	クラスター・ワークロードでリモート・キューを使用します。
CodedCharSetId	コード化文字セット ID。
CommandEvent	コマンド・イベントの制御属性。
CommandInputQName 属性	コマンド入力キュー名。
CommandLevel	コマンド・レベル。
CommandServerControl 属性	コマンド・サーバーの制御属性。
ConfigurationEvent 属性	構成イベントの制御属性。
DeadLetterQName	送達不能キューの名前。
DefClusterXmitQueueType	デフォルトのクラスタ伝送キュータイプ。
DefXmitQName	デフォルト伝送キュー名。
DistLists	配布リスト・サポート。
 DNS グループ	Workload Manager Dynamic Domain Name Services サポートを使用するときの TCP リスナー・グループの名前。
 DNSWLM	TCP リスナーが Workload Manager for Dynamic Domain Name Services に登録するかどうか。

表 593. キュー・マネージャーの属性 (続き)	
属性	説明
ExpiryInterval	期限切れメッセージのスキャン間隔。
IGQPutAuthority	グループ内キューイング書き込み権限。
IGQUserId	グループ内キューイングのユーザー ID。
InhibitEvent	禁止 (読み取りおよび書き込み禁止) イベントが生成されるかどうかを制御します。
InitialKey 1	パスワード保護システムの初期キー。
IPAddressVersion	Internet Protocol アドレスのバージョン
 IntraGroupQueuing	グループ内キューイングのサポート。
 ListenerTimer	APPC または TCP/IP の障害後にリスナーの再始動を試行する時間間隔。
LocalEvent	ローカル・エラー・イベントが生成されるかどうかを制御する。
LoggerEvent	ロガーイベントが生成されるかどうかを制御します。
 LUGroupName	キュー共有グループのインバウンド送信を処理する LU 6.2 リスナーの総称 LU 名。
 LUName	アウトバウンド LU 6.2 伝送で使用する LU の名前。
LU62ARMSuffix	このチャネル・イニシエーターの LUADD を指定する、SYS1.PARMLIB メンバー APPCPMxx の接尾部。
 LU62Channels	LU 6.2 を使用する現行チャネルまたは接続クライアントの最大数。
MaxActiveChannels	いつでもアクティブにできるチャネルの最大数。
MaxChannels	現行チャネルの最大数。
MaxHandles	ハンドルの最大数です。
MaxMsgLength	メッセージの最大長 (バイト数)。
MaxPriority 属性	最高の優先順位。
MaxPropertiesLength	プロパティデータの最大長 (バイト単位)。
MaxUncommittedMsgs	1 つの作業単位のコミットされていないメッセージの最大数。
MQIAccounting	MQI データに関するアカウンティング情報の収集を制御します。
MQIStatistics	キュー・マネージャーに関する統計モニター情報の収集を制御します。
MsgMarkBrowseInterval	キュー・マネージャーがブラウズされたメッセージからマークを除去できる間隔。
 OutboundPortMin	<i>OutboundPortMin</i> では、発信チャネルをバインドするときに使用するポート番号の範囲を定義します。
 OutboundPortMin	<i>OutboundPortMax</i> では、発信チャネルをバインドするときに使用するポート番号の範囲を定義します。
PerformanceEvent	パフォーマンス関連イベントが生成されるかどうかを制御します。
Platform	キュー・マネージャーが実行されているプラットフォーム。
PubSubNPInputMsg	未配信の入力メッセージを破棄するか (または保持するか)。
PubSubNPResponse	未配信の動作を制御します。
PubSubMaxMsgRetryCount	失敗したコマンド・メッセージを (同期点で) 処理する際の再試行の数
PubSubSyncPoint	同期ポイント下で処理すべきメッセージは永続的なもののみ (または全て) であるべきか。
PubSubMode	キュー化されたバプリッシュ/サブスクライブインターフェースが稼働中かどうか。
QMGrDesc	キュー・マネージャーの記述。
QMGrIdentifier	キューマネージャの固有の内部生成識別子。
QMGrName	キュー・マネージャー名。
QSGName	キュー共有グループ名
QueueAccounting	キューに関するアカウンティング情報の収集を制御します。
QueueMonitoring	キューのオンライン監視データ。
QueueStatistics	キューに関する統計データの収集を制御します。

表 593. キュー・マネージャーの属性 (続き)	
属性	説明
 ReceiveTimeout	非アクティブ状態に戻るまでに TCP/IP チャンネルがデータを待機する長さ。
 ReceiveTimeoutMin	.ReceiveTimeout の予選
 ReceiveTimeoutType	非アクティブ状態に戻るまでに TCP/IP チャンネルがデータを待機する最小時間。
RemoteEvent	リモート・エラー・イベントが生成されるかどうかを制御します。
RepositoryName	このキュー・マネージャーがリポジトリ・サービスを提供するクラスターの名前。
RepositoryNamelist	このキュー・マネージャーがリポジトリ・サービスを提供するクラスターの名前が入った名前リスト・オブジェクトの名前。
ScyCase	セキュリティ・プロファイルのケース
SharedQMgrName	共有キューのキュー・マネージャー名
826 ページの『SPLCAP』	Advanced Message Security キューマネージャの保護機能を有効または無効にする。AMS
SSLCRLNamelist 1	認証情報オブジェクトの名前を含む名前リスト・オブジェクトの名前。
SSLCryptoHardware 1	暗号ハードウェア構成ストリング。
SSLEvent	TLS イベントの属性を制御します。
SSLFIPSRequired	暗号化には認定されたアルゴリズム FIPS のみを使用してください。
SSLKeyRepository 1	TLS 鍵リポジトリの位置。
SSLKeyRepositoryPassword 1	TLS キーリポジトリのパスワード。
SSLKeyResetCount	TLS 鍵リセット・カウント。
SSLTasks 1	TLS 呼び出しを処理するためのサーバー・サブタスクの数。
StatisticsInterval	統計モニター・データを書き込む頻度。
StartStopEvent	開始イベントおよび停止イベントが生成されるかどうかを制御します。
SyncPoint	同期点の可用性。
 TCP チャンネル	TCP/IP を使用する現行チャンネルまたは接続クライアントの最大数。
 TCPKeepAlive	TCP KEEPALIVE を使用して接続の他方の側をチェックするかどうか。
 TCP 名	使用している TCP/IP システムの名前。
 TCPStackType	チャンネル・イニシエーターが TCP/IP アドレスをどのように使用できるか。
TraceRouteRecording 属性	トレース経路指定情報の記録を制御する。
TriggerInterval	トリガー・メッセージの間隔。
バージョン	バージョン。
XrCapability	テレメトリ・コマンドをサポートするかどうかを指定します。
注:	
1. この属性は MQINQ 呼び出しを使用して照会することができず、このセクションでは説明されていません。この属性について詳しくは、 Change Queue Manager を参照してください。	

関連タスク

[MQI クライアントの実行時に、認証済みの CipherSpecs のみ FIPS を使用するよう指定する](#)

関連資料

[Federal Information Processing Standards \(FIPS\) について AIX, Linux, and Windows](#)

AccountingConnOverride (MQLONG)

この属性を使用して、アプリケーションは Qmgr 属性の ACCTMQI および ACCTQDATA 値の設定をオーバーライドすることができます。

値は、次のいずれか 1 つです。

MQMON_DISABLED

アプリケーションは、MQCONN 呼び出しでの MQCNO 構造の Options フィールドを使用して、ACCTMQI および ACCTQ Qmgr 属性の設定をオーバーライドすることはできません。これがデフォルト値です。

MQMON_ENABLED

アプリケーションは、MQCNO 構造の Options フィールドを使用して ACCTQ および ACCTMQI Qmgr 属性をオーバーライドすることができます。

この値の変更は、属性を変更した後に、キュー・マネージャーへの接続でのみ有効です。

この属性は、次のプラットフォームでのみサポートされています。

-  IBM i
-   AIX and Linux
-  Windows

この属性の値を判別するには、MQINQ 呼び出しで MQIA_ACCOUNTING_CONN_OVERRIDE セレクターを使用します。

AccountingInterval (MQLONG)

中間アカウント・レコードが書き込まれるまでの時間を指定します (秒)。

この値は、0 から 604800 までの範囲の整数です。デフォルト値は 1800 (30 分) です。中間レコードをオフにするには 0 を指定します。

この属性は、次のプラットフォームでのみサポートされています。

-  IBM i
-   AIX and Linux
-  Linux
-  Windows

この属性の値を判別するには、MQINQ 呼び出しで MQIA_ACCOUNTING_INTERVAL セレクターを使用します。

ActivityConnOverride (MQLONG)

これは、キュー・マネージャー属性の ACTVTRC 値の設定をアプリケーションがオーバーライドできるようにします。

値は、次のいずれか 1 つです。

MQMON_DISABLED

アプリケーションは、MQCONN 呼び出しで MQCNO 構造の Options フィールドを使用して ACTVTRC キュー・マネージャー属性の設定をオーバーライドすることはできません。これがデフォルト値です。

MQMON_ENABLED

アプリケーションは、MQCNO 構造の Options フィールドを使用して ACTVTRC キュー・マネージャー属性をオーバーライドできます。

この値の変更は、属性を変更した後に、キュー・マネージャーへの接続でのみ有効です。

この属性は、マルチプラットフォームでのみサポートされます。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_ACTIVITY_CONN_OVERRIDE セレクターを使用します。

ActivityTrace (MQLONG)

これは、IBM MQ MQI アプリケーションのアクティビティ・トレースの収集を制御します。

値は、次のいずれか 1 つです。

MQMON_ON

IBM MQ MQI アプリケーションのアクティビティー・トレースを収集します。

MQMON_OFF

IBM MQ MQI アプリケーションのアクティビティー・トレースを収集しません。これがデフォルト値です。

キュー・マネージャー属性 ACTVCON0 を ENABLED に設定した場合、MQCNO 構造の Options フィールドを使用する個別の接続でこの値はオーバーライドされる可能性があります。

この値の変更は、属性を変更した後に、キュー・マネージャーへの接続でのみ有効です。

この属性は、マルチプラットフォームでのみサポートされます。

この属性値を調べるには、MQINQ 呼び出しで MQIA_ACTIVITY_TRACE セレクターを使用します。

AdoptNewMCACheck (MQLONG)

すでにアクティブになっている MCA と同じ名前を持つ新しいインバウンド・チャネルが検出されたときに、MCA を取り入れるかどうかを判別するために検査されるエレメントを定義します。

値は、次のいずれか 1 つです。

MQADOPT_CHECK_Q_MGR_NAME

キュー・マネージャーの名前を検査します。

MQADOPT_CHECK_NET_ADDR

ネットワーク・アドレスを検査します。

MQADOPT_CHECK_ALL

キュー・マネージャー名とネットワーク・アドレスを検査します。可能な場合、この検査を実行してチャネルを不注意または故意にシャットダウンしないようにします。これがデフォルト値です。

MQADOPT_CHECK_NONE

どの要素も検査しません。

この属性の変更は、チャネルが次にチャネルを取り入れようとしたときに有効になります。

 この属性は、z/OS でのみサポートされます。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_ADOPTNEWMCA_CHECK セレクターを使用します。

AdoptNewMCAType (MQLONG)

AdoptNewMCACheck 属性と一致する新しいインバウンド・チャネル要求が検出されたときに、特定のチャネル・タイプの MCA の孤立したインスタンスを自動的に再始動するかどうかを指定します。

これは、次の値のいずれかです。

MQADOPT_TYPE_NO

孤立したチャネル・インスタンスを取り入れる必要はありません。これがデフォルト値です。

MQADOPT_TYPE_ALL

すべてのチャネル・タイプを採用します。

この属性は z/OS でのみサポートされます。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_ADOPTNEWMCA_TYPE セレクターを使用します。

AlterationDate (MQCHAR12)

これは、定義を最後に変更した日付です。日付の形式は YYYY-MM-DD で、その後に 2 つの末尾空白を付けて長さ 12 バイトになります。

この属性の値を判別するには、MQINQ 呼び出しで MQCA_ALTERATION_DATE セレクターを使用します。
この属性の長さは MQ_DATE_LENGTH によって指定されます。

AlterationTime (MQCHAR8)

これは、定義を最後に変更した時刻です。時刻の形式は HH.MM.SS です。

この属性の値を判別するには、MQINQ 呼び出しで MQCA_ALTERATION_TIME セレクターを使用します。
この属性の長さは MQ_TIME_LENGTH によって指定されます。

AuthorityEvent (MQLONG)

 から IBM MQ 10.0.0、認証イベントの生成の有無を制御します。

これは、次の値のいずれかです。

MQEVR_DISABLED

イベント報告は無効です。

MQEVR_ENABLED

イベント報告は有効です。

イベントの詳細については、[イベント・モニター](#)を参照してください。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_AUTHORITY_EVENT セレクターを使用します。

AuthorityEventScope (MQLONG)

有効化すると、認可イベントの範囲を制御します。これは、次の値のいずれかです。

MQAUSC_失敗

エラーイベントの報告は、障害発生時のみ有効（デフォルト）。

MQAUSC_全接続

エラーおよび拡張接続イベントの報告のみ。

MQAUSC_全チェック

エラーおよびすべての拡張イベントの報告。

イベントの詳細については、[イベント・モニター](#)を参照してください。

この属性の値を決定するには、MQINQ 呼び出しで MQIA_AUTHOREV_SCOPE セレクタを使用します。

BridgeEvent (MQLONG)

これは IMS ブリッジ・イベントが生成されるかどうかを特定します。

値は、次のいずれか 1 つです。

MQEVR_ENABLED

IMS ブリッジ・イベントを以下のように生成します。

MQRC_BRIDGE_STARTED

MQRC_BRIDGE_STOPPED

MQEVR_DISABLED

IMS ブリッジ・イベントを生成しません。これはデフォルト値です。

この属性は z/OS でのみサポートされます。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_BRIDGE_EVENT セレクターを使用します。

ChannelAutoDef (MQLONG)

この属性はタイプ MQCHT_RECEIVER および MQCHT_SVRCONN のチャンネルの自動定義を制御します。
MQCHT_CLUSSDR チャンネルの自動定義は常に使用可能です。値は、次のいずれか 1 つです。

MQCHAD_DISABLED

チャンネルの自動定義は無効です。

MQCHAD_ENABLED

チャンネルの自動定義は有効です。

 この属性は、[マルチプラットフォーム](#) でのみサポートされます。

属性の値を判別するには、MQINQ 呼び出しで MQIA_CHANNEL_AUTO_DEF セレクターを使用します。

ChannelAutoDefEvent (MQLONG)

チャンネル自動定義のイベントが生成されるかどうかを制御します。これは、タイプ MQCHT_RECEIVER、MQCHT_SVRCONN および MQCHT_CLUSSDR のチャンネルに適用されます。値は、次のいずれか 1 つです。

MQEVR_DISABLED

イベント報告は無効です。

MQEVR_ENABLED

イベント報告は有効です。

イベントの詳細については、[イベント・モニター](#)を参照してください。

 この属性は、[マルチプラットフォーム](#) でのみサポートされます。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_CHANNEL_AUTO_DEF_EVENT セレクターを使用します。

ChannelAutoDefExit (MQCHARn)

これは、自動チャンネル定義のユーザー出口の名前です。この名前が非空白で、*ChannelAutoDef* に値 MQCHAD_ENABLED がある場合、出口はキュー・マネージャーがチャンネル定義を作成しようとするたびに呼び出されます。これは、タイプ MQCHT_RECEIVER、MQCHT_SVRCONN および MQCHT_CLUSSDR のチャンネルに適用されます。このとき、出口は次のいずれかの処理を行うことができます。

- 変更せずにチャンネル定義を作成する。
- 作成されたチャンネル定義の属性を変更する。
- チャンネルの作成をまったく行わない。

注：この属性の長さや値は、両方とも環境によって異なります。それぞれの環境におけるこの属性の値の詳細については、[1480 ページ](#)の『MQCD - チャンネル定義』の MQCD 構造体の概要を参照してください。

 z/OS では、この属性はクラスター送信側チャンネルおよびクラスター受信側チャンネルにのみ適用されます。

この属性の値を判別するには、MQINQ 呼び出しで MQCA_CHANNEL_AUTO_DEF_EXIT セレクターを使用します。属性の長さは、MQ_EXIT_NAME_LENGTH で指定します。

ChannelEvent (MQLONG)

チャンネル・イベントが生成されるかどうかを指定します。

これは、次の値のいずれかです。

MQEVR_EXCEPTION

以下のチャンネル・イベントのみを生成します。

- MQRC_CHANNEL_ACTIVATED
- MQRC_CHANNEL_CONV_ERROR
- MQRC_CHANNEL_NOT_ACTIVATED
- 以下の ReasonQualifiers の付いた MQRC_CHANNEL_STOPPED。

MQRQ_CHANNEL_STOPPED_ERROR

MQRQ_CHANNEL_STOPPED_RETRY

MQRQ_CHANNEL_STOPPED_DISABLED

MQRC_CHANNEL_STOPPED_BY_USER

MQEVR_ENABLED

すべてのチャンネル・イベントを生成します。つまり、EXCEPTION で生成されたこれらのチャンネル・イベントに加えて、以下のチャンネル・イベントを生成します。

- MQRC_CHANNEL_STARTED
- 以下の ReasonQualifier の付いた MQRC_CHANNEL_STOPPED。

MQRC_CHANNEL_STOPPED_OK

MQEVR_DISABLED

チャンネル・イベントを生成しません。これはデフォルト値です。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_CHANNEL_EVENT セレクターを使用します。

ChannelInitiatorControl (MQLONG)

これは、キュー・マネージャーの開始時にチャンネル・イニシエーターを開始するかどうかを指定します。

これは、次の値のいずれかです。

MQSVC_CONTROL_MANUAL

チャンネル・イニシエーターは自動的に開始されません。

MQSVC_CONTROL_Q_MGR

キュー・マネージャーの始動時にチャンネル・イニシエーターを自動的に開始します。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_CHINIT_CONTROL セレクターを使用します。

ChannelMonitoring (MQLONG)

この属性はチャンネルのオンライン・モニター・データを指定します。

値は、次のいずれか 1 つです。

MQMON_NONE

MONCHL チャンネル属性の設定にかかわらず、すべてのチャンネルでチャンネル・モニターのデータ収集を使用不可にします。これがデフォルト値です。

MQMON_OFF

MONCHL チャンネル属性に QMGR が指定されているチャンネルのモニター・データ収集をオフにします。

MQMON_LOW

MONCHL チャンネル属性に QMGR が指定されているチャンネルに対して、データを低い比率で収集するモニター・データ収集をオンにします。

MQMON_MEDIUM

MONCHL チャンネル属性に QMGR が指定されているチャンネルに対して、データを中程度の比率で収集するモニター・データ収集をオンにします。

MQMON_HIGH

MONCHL チャンネル属性に QMGR が指定されているチャンネルに対して、データを高い比率で収集するモニター・データ収集をオンにします。

 システム上 z/OS では、このパラメータを有効にすると、選択した値にかかわらず、単に統計データの収集が開始されます。LOW、MEDIUM、または HIGH を指定しても、結果に違いはありません。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_MONITORING_CHANNEL セレクターを使用します。

ChannelStatistics (MQLONG)

チャンネルの統計データのコレクションを制御します。

値は、次のいずれか 1 つです。

MQMON_NONE

STATCHL チャンネル属性の設定にかかわらず、すべてのチャンネルでチャンネル統計のデータ収集を使用不可にします。これがデフォルト値です。

MQMON_OFF

STATCHL チャンネル属性で QMGR を指定するチャンネルで、統計データ収集をオフにします。

MQMON_LOW

STATCHL チャンネル属性で QMGR を指定するチャンネルで、低い比率でデータ収集を行う統計データ収集をオンにします。

MQMON_MEDIUM

STATCHL チャンネル属性で QMGR を指定するチャンネルで、中程度の比率でデータ収集を行う統計データ収集をオンにします。

MQMON_HIGH

STATCHL チャンネル属性で QMGR を指定するチャンネルで、高い比率でデータ収集を行う統計データ収集をオンにします。

ほとんどのシステムでは、MEDIUM を使用することをお勧めします。ただし、1 秒ごとに処理するメッセージ・ボリュームが大きいチャンネルの場合、LOW を選択してサンプリング・レベルを減らすこともできます。さらに、いくつかのメッセージのみを処理するチャンネルや、最新の情報が重要なチャンネルの場合は、HIGH を選択することもできます。

z/OS システム上 z/OS では、このパラメータを有効にすると、選択した値にかかわらず、単に統計データの収集が開始されます。LOW、MEDIUM、または HIGH を指定しても、結果に違いはありません。チャンネル・アカウンティング・レコードを収集するには、このパラメーターを有効にしなければなりません。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_STATISTICS_CHANNEL セレクターを使用します。

z/OS ChinitAdapters (MQLONG)

これは、IBM MQ 呼び出しの処理に使用するアダプター・サブタスクの数です。値は 0 から 9999 まででなければなりません。デフォルト値は 8 です。

アダプターのディスパッチャーへの比率 (ChinitDispatchers 属性) は、8 から 5 の値にする必要があります。ただし、チャンネルの数が少ない場合は、このパラメーターの値をデフォルト値から減らす必要はありません。次の値を使用できます。テスト・システムの場合は 8 (デフォルト)、実動システムの場合は 20。20 のアダプターを持つのが理想的です。こうすれば、IBM MQ 呼び出しの並列性を高めることができます。これは、持続メッセージの場合重要です。非持続メッセージの場合には、アダプターの数が多いほうがよいかもしれません。

この属性は z/OS でのみサポートされます。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_CHINIT_ADAPTERS セレクターを使用します。

z/OS ChinitDispatchers (MQLONG)

これは、チャンネル・イニシエーターで使用するディスパッチャーの数です。値は 0 から 9999 まででなければなりません。デフォルト値は 5 です。

ガイドラインとして、50 個の現行チャンネルに対して 1 つのディスパッチャーを使用できます。ただし、チャンネルの数がごくわずかである場合、この属性の値をデフォルト値から減らす必要はありません。TCP/IP を使用する場合、ここで大きな値を指定したとしても、TCP/IP チャンネルに使用されるディスパッチャーの最大数は 100 になります。次の設定を使用できます。テスト・システムの場合は 5 (デフォルト)、実動システムの場合は 20 (最大 1000 個のアクティブ・チャンネルを処理するには 20 個のディスパッチャーが必要です)。

この属性は z/OS でのみサポートされます。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_CHINIT_DISPATCHERS セレクターを使用します。

z/OS ChinitTraceAutoStart (MQLONG)

チャンネル・イニシエーター・トレースを自動的に開始するかどうかを指定します。

値は、次のいずれか 1 つです。

MQTRAXSTR_YES

チャンネル・イニシエーターのトレースを自動的に開始します。これがデフォルト値です。

MQTRAXSTR_NO

チャンネル・イニシエーターのトレースを自動的に開始しません。

この属性は z/OS でのみサポートされます。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_CHINIT_TRACE_AUTO_START セレクターを使用します。

ChinitTraceTableSize (MQLONG)

これはチャンネル・イニシエーターのトレース・データ・スペースのサイズ (MB) です。

値は 0 から 2048 までの範囲内でなければなりません。デフォルト値は 2 です。

注: 大量の z/OS データ・スペースを使用する場合、システム上に関連した z/OS ページング・アクティビティをサポートするための十分な補助ストレージが存在するようにしてください。また、SYS1.DUMP データ・セットのサイズを増やす必要がある可能性があります。

この属性は z/OS でのみサポートされます。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_CHINIT_TRACE_TABLE_SIZE セレクターを使用します。

ClusterSenderMonitoringDefault (MQLONG)

これは、自動的に定義されたクラスター送信側チャンネルの ChannelMonitoring 属性に置換される値を指定します。

値は、次のいずれか 1 つです。

MQMON_Q_MGR

オンライン・モニター・データのコレクションは、キュー・マネージャーの **ChannelMonitoring** 属性の設定から継承されます。これがデフォルト値です。

MQMON_OFF

無効化されているチャンネルのモニター。

MQMON_LOW

ChannelMonitoring が MQMON_NONE である場合を除き、モニターは使用可能になります。データ収集率は低く、システムのパフォーマンスにはほとんど影響しません。収集されるデータは最新のものではない可能性があります。

MQMON_MEDIUM

ChannelMonitoring が MQMON_NONE である場合を除き、モニターは使用可能になります。データ収集率は中程度であり、システムのパフォーマンスへの影響は限られています。

MQMON_HIGH

ChannelMonitoring が MQMON_NONE である場合を除き、モニターは使用可能になります。データ収集率は高く、システムのパフォーマンスに影響を与える可能性があります。収集されるデータは、取得可能なデータの中で最新のものです。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_MONITORING_AUTO_CLUSSDR セレクターを使用します。

ClusterSenderStatistics (MQLONG)

クラスター送信側チャンネルは、リポジトリ内の CLUSRCVR の定義から自動的に定義されるため、ALTER チャンネルを使用してこれらの自動定義されたクラスター送信側チャンネルの STATCHL 属性の設定を変更することはできません。これらのチャンネルでは、オンライン・モニター・データを収集するかどうかは、このキュー・マネージャー属性の設定に基づいて決定されます。

値は、次のいずれか 1 つです。

MQMON_Q_MGR

自動定義されたクラスター送信側チャネルの統計データ収集は、キュー・マネージャー属性 STATCHL の値に基づいています。これがデフォルト値です。

MQMON_OFF

自動定義されたクラスター送信側チャネルの統計データ収集をオフに切り替えます。

MQMON_LOW

低い比率でデータ収集を行う、自動定義されたクラスター送信側チャネルの統計データ収集を使用可能にします。

MQMON_MEDIUM

中程度の比率でデータ収集を行う、自動定義されたクラスター送信側チャネルの統計データ収集を使用可能にします。

MQMON_HIGH

高い比率でデータ収集を行う、自動定義されたクラスター送信側チャネルの統計データ収集を使用可能にします。

ほとんどのシステムでは、MEDIUM をお勧めします。ただし、1 秒ごとに処理するメッセージ・ボリュームが大きい、自動定義されたクラスター送信側チャネルの場合、LOW を選択してサンプリング・レベルを減らすこともできます。さらに、いくつかのメッセージのみを処理するチャネルや、最新の情報が重要なチャネルの場合は、HIGH を選択することもできます。

 システム上 z/OS では、このパラメータを有効にすると、選択した値にかかわらず、単に統計データの収集が開始されます。LOW、MEDIUM、または HIGH を指定しても、結果に違いはありません。チャネル・アカウンティング・レコードを収集するには、このパラメータを有効にしなければなりません。

属性の値を判別するには、MQINQ 呼び出しで MQIA_STATISTICS_AUTO_CLUSSDR セレクターを使用します。

ClusterWorkloadData (MQCHAR32)

これは、クラスター・ワークロード出口が呼び出されたとき、この出口に引き渡される 32 バイトのユーザー定義文字ストリングです。出口に引き渡すデータがない場合、そのストリングはブランクになります。

この属性の値を判別するには、MQINQ 呼び出しで MQCA_CLUSTER_WORKLOAD_DATA セレクターを使用します。

ClusterWorkloadExit (MQCHARn)

これは、クラスター・ワークロード管理のユーザー出口の名前です。この名前がブランクでない場合、メッセージがクラスター・キューに書き込まれるか、あるクラスター送信側キューから別のクラスター送信側キューに移動されるたびに、この出口が呼び出されます。その後、この出口は、メッセージ宛先としてキュー・マネージャーによって選択されたキュー・インスタンスを受け入れるか、または別のキュー・インスタンスを選択することができます。

注：この属性の長さ値は、両方とも環境によって異なります。

この属性の値を判別するには、MQINQ 呼び出しで MQCA_CLUSTER_WORKLOAD_EXIT セレクターを使用します。属性の長さは、MQ_EXIT_NAME_LENGTH で指定します。

ClusterWorkloadLength (MQLONG)

これは、クラスター・ワークロード出口に引き渡されるメッセージ・データの最大長です。出口に引き渡されるデータの実際の長さは、以下の値のうちで最小の値です。

- メッセージの長さ。
- キュー・マネージャーの **MaxMsgLength** 属性。
- **ClusterWorkloadLength** 属性。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_CLUSTER_WORKLOAD_LENGTH セレクターを使用します。

CLWLMRUChannels (MQLONG)

これは、クラスター・ワークロード選択アルゴリズムでの使用が考えられる、最近使用されたクラスター・チャンネルの最大数を指定します。

これは 1 から 9999999999 までの範囲の値です。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_CLWL_MRU_CHANNELS セレクターを使用します。

CLWLUseQ (MQLONG)

クラスター・ワークロードでリモート・キューを使用するかどうかを指定します。

値は、次のいずれか 1 つです。

MQCLWL_USEQ_ANY

ローカル・キューとリモート・キューの両方を使用します。

MQCLWL_USEQ_LOCAL

リモート・キューを使用しません。これがデフォルト値です。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_CLWL_USEQ セレクターを使用します。

CodedCharSetId (MQLONG)

これは、オブジェクトの名前やキュー作成の日付および時刻など、MQI 内に定義されているすべての文字ストリング・フィールドで、キュー・マネージャーが使用する文字セットを定義します。文字セットは、オブジェクト名で有効な文字を指定するために 1 バイト文字を備えている必要があります。メッセージ内に取り込まれるアプリケーション・データには適用されません。値は環境によって異なります。

- z/OS では、この値は、キュー・マネージャーの始動時にシステム・パラメーターから設定されます。デフォルト値は 500 です。
- Windows では、この値は、キュー・マネージャーを作成したユーザーの 1 次 CODEPAGE です。
- IBM i では、この値は、キュー・マネージャーの最初の作成時に環境に設定される値です。
- AIX and Linux では、この値はキュー・マネージャーを作成したユーザーのロケールのデフォルトである CODESET です。

属性の値を判別するには、MQINQ 呼び出しで MQIA_CODED_CHAR_SET_ID セレクターを使用します。

CommandEvent (MQLONG)

以下のように、コマンド・イベントが生成されるかどうかを指定します。

MQEVR_DISABLED

コマンド・イベントを生成しません。これはデフォルトです。

MQEVR_ENABLED

コマンド・イベントを生成します。

MQEVR_NO_DISPLAY

MQINQ 以外の正常に実行されたコマンドでコマンド・イベントが生成されます。

属性の値を判別するには、MQINQ 呼び出しで MQIA_COMMAND_EVENT セレクターを使用します。

CommandInputQName (MQCHAR48)

これは、ローカル・キュー・マネージャーに定義されたコマンド入力キューの名前です。ユーザーがコマンドを送ることができるキューです (ただし、アプリケーションがその許可を持っている場合)。キューの名前は以下のように環境によって変わります。

- z/OS では、キューの名前は SYSTEM.COMMAND.INPUT で、キューに送信できるのは MQSC および PCF コマンドです。MQSC コマンドの詳細については、[MQSC コマンドを参照してください](#)。PCF コマンドの詳細については、[プログラマブル・コマンド・フォーマットの定義を参照してください](#)。

- その他のすべての環境では、キューの名前は `SYSTEM.ADMIN.COMMAND.QUEUE` で、キューに送信できるのは PCF コマンドのみです。ただし、MQSC コマンドがタイプ `MQCMD_ESCAPE` の PCF コマンド内に格納されている場合は、このキューに MQSC コマンドを送信できます。Escape コマンドについて詳しくは、[Escape](#) を参照してください。

この属性の値を判別するには、MQINQ 呼び出しで `MQCA_COMMAND_INPUT_Q_NAME` セレクターを使用します。この属性の長さは `MQ_Q_NAME_LENGTH` によって指定されます。

CommandLevel (MQLONG)

これは、キュー・マネージャーによってサポートされるシステム制御コマンドのレベルを示します。以下のいずれかの値を選択できます。

MQCMDL_LEVEL_800

レベル 800 のシステム制御コマンド。

この値は、以下のバージョンから戻されます。

- IBM MQ for AIX 8.0
- IBM MQ for HP-UX 8.0
- IBM MQ for IBM i 8.0
- IBM MQ for Linux 8.0
- IBM MQ for Solaris 8.0
- IBM MQ for Windows 8.0
- IBM MQ for z/OS 8.0

MQCMDL_LEVEL_801

レベル 801 のシステム制御コマンド。

この値は、以下のバージョンから戻されます。

- IBM MQ for AIX 8.0.0 Fix Pack 2
- IBM MQ for HP-UX 8.0.0 Fix Pack 2
- IBM MQ for IBM i 8.0.0 Fix Pack 2
- IBM MQ for Linux 8.0.0 Fix Pack 2
- IBM MQ for Solaris 8.0.0 Fix Pack 2

MQCMDL_LEVEL_802

レベル 802 のシステム制御コマンド。

この値は、以下のバージョンから戻されます。

- IBM MQ for AIX 8.0.0 Fix Pack 3
- IBM MQ for HP-UX 8.0.0 Fix Pack 3
- IBM MQ for IBM i 8.0.0 Fix Pack 3
- IBM MQ for Linux 8.0.0 Fix Pack 3
- IBM MQ for Solaris 8.0.0 Fix Pack 3
- IBM MQ for Windows 8.0.0 Fix Pack 3
- IBM MQ for HPE NonStop 8.1.0

MQCMDL_LEVEL_900

レベル 900 のシステム制御コマンド。

この値は、以下のバージョンから戻されます。

- IBM MQ for AIX 9.0
- IBM MQ for HP-UX 9.0
- IBM MQ for IBM i 9.0

- IBM MQ for Linux 9.0
- IBM MQ for Solaris 9.0
- IBM MQ for Windows 9.0
- IBM MQ for z/OS 9.0

MQCMDL_LEVEL_901

レベル 901 のシステム制御コマンド。

この値は、以下のバージョンから戻されます。

- IBM MQ for Linux 9.0.1
- IBM MQ for Windows 9.0.1
- IBM MQ for z/OS 9.0.1

MQCMDL_LEVEL_902

レベル 902 のシステム制御コマンド。

この値は、以下のバージョンから戻されます。

- IBM MQ for Linux 9.0.2
- IBM MQ for Windows 9.0.2
- IBM MQ for z/OS 9.0.2

MQCMDL_LEVEL_903

レベル 903 のシステム制御コマンド。

この値は、以下のバージョンから戻されます。

- IBM MQ for Linux 9.0.3
- IBM MQ for Windows 9.0.3
- IBM MQ for z/OS 9.0.3

MQCMDL_LEVEL_904

レベル 904 のシステム制御コマンド。

この値は、以下のバージョンから戻されます。

- IBM MQ for AIX 9.0.4
- IBM MQ for Linux 9.0.4
- IBM MQ for Windows 9.0.4
- IBM MQ for z/OS 9.0.4

MQCMDL_LEVEL_905

レベル 905 のシステム制御コマンド。

この値は、以下のバージョンから戻されます。

- IBM MQ for AIX 9.0.5
- IBM MQ for Linux 9.0.5
- IBM MQ for Windows 9.0.5
- IBM MQ for z/OS 9.0.5

MQCMDL_LEVEL_910

レベル 910 のシステム制御コマンド。

この値は、以下のバージョンから戻されます。

- IBM MQ for AIX 9.1
- IBM MQ for IBM i 9.1
- IBM MQ for Linux 9.1
- IBM MQ for Solaris 9.1

- IBM MQ for Windows 9.1
- IBM MQ for z/OS 9.1

MQCMDL_LEVEL_911

レベル 911 のシステム制御コマンド。

この値は、以下のバージョンから戻されます。

- IBM MQ for AIX 9.1.1
- IBM MQ for Linux 9.1.1
- IBM MQ for Windows 9.1.1
- IBM MQ for z/OS 9.1.1

MQCMDL_LEVEL_912

レベル 912 のシステム制御コマンド。

この値は、以下のバージョンから戻されます。

- IBM MQ for AIX 9.1.2
- IBM MQ for Linux 9.1.2
- IBM MQ for Windows 9.1.2
- IBM MQ for z/OS 9.1.2

MQCMDL_LEVEL_913

レベル 913 のシステム制御コマンド。

この値は、以下のバージョンから戻されます。

- IBM MQ for AIX 9.1.3
- IBM MQ for Linux 9.1.3
- IBM MQ for Windows 9.1.3
- IBM MQ for z/OS 9.1.3

MQCMDL_LEVEL_914

システム制御コマンドのレベル 914。

この値は、以下のバージョンから戻されます。

- IBM MQ for AIX 9.1.4
- IBM MQ for Linux 9.1.4
- IBM MQ for Windows 9.1.4
- IBM MQ for z/OS 9.1.4

MQCMDL_LEVEL_915

システム制御コマンドのレベル 915。

この値は、以下のバージョンから戻されます。

- IBM MQ for AIX 9.1.5
- IBM MQ for Linux 9.1.5
- IBM MQ for Windows 9.1.5
- IBM MQ for z/OS 9.1.5

MQCMDL_LEVEL_910

レベル 910 のシステム制御コマンド。

この値は、以下のバージョンから戻されます。

- IBM MQ for AIX 9.1
- IBM MQ for IBM i 9.1
- IBM MQ for Linux 9.1

- IBM MQ for Windows 9.1
- IBM MQ for z/OS 9.1

MQCMDL_LEVEL_920

システム制御コマンドのレベル 920。

この値は、以下のバージョンから戻されます。

- IBM MQ for AIX 9.2
- IBM MQ for IBM i 9.2
- IBM MQ for Linux 9.2
- IBM MQ for Windows 9.2
- IBM MQ for z/OS 9.2

MQCMDL_LEVEL_921

システム制御コマンドのレベル 921。

この値は、以下のバージョンから戻されます。

- IBM MQ for AIX 9.2.1
- IBM MQ for Linux 9.2.1
- IBM MQ for Windows 9.2.1
- IBM MQ for z/OS 9.2.1

MQCMDL_LEVEL_922

システム制御コマンドのレベル 922。

この値は、以下のバージョンから戻されます。

- IBM MQ for AIX 9.2.2
- IBM MQ for Linux 9.2.2
- IBM MQ for Windows 9.2.2
- IBM MQ for z/OS 9.2.2

MQCMDL_LEVEL_923

システム制御コマンドのレベル 923。

この値は、以下のバージョンから戻されます。

- IBM MQ for AIX 9.2.3
- IBM MQ for Linux 9.2.3
- IBM MQ for Windows 9.2.3
- IBM MQ for z/OS 9.2.3

MQCMDL_LEVEL_924

システム制御コマンドのレベル 924。

この値は、以下のバージョンから戻されます。

- IBM MQ for AIX 9.2.4
- IBM MQ for Linux 9.2.4
- IBM MQ for Windows 9.2.4
- IBM MQ for z/OS 9.2.4

MQCMDL_LEVEL_925

システム制御コマンドのレベル 925。

この値は、以下のバージョンから戻されます。

- IBM MQ for AIX 9.2.5
- IBM MQ for Linux 9.2.5

- IBM MQ for Windows 9.2.5
- IBM MQ for z/OS 9.2.5

MQCMDL_LEVEL_930

システム制御コマンドのレベル 930。

この値は、以下のバージョンから戻されます。

- IBM MQ for AIX 9.3
- IBM MQ for IBM i 9.3
- IBM MQ for Linux 9.3
- IBM MQ for Windows 9.3
- IBM MQ for z/OS 9.3

MQCMDL_LEVEL_931

システム制御コマンドのレベル 931。

この値は、以下のバージョンから戻されます。

- IBM MQ for AIX 9.3.1
- IBM MQ for Linux 9.3.1
- IBM MQ for Windows 9.3.1
- IBM MQ for z/OS 9.3.1

MQCMDL_LEVEL_932

システム制御コマンドのレベル 932。

この値は、以下のバージョンから戻されます。

- IBM MQ for AIX 9.3.2
- IBM MQ for Linux 9.3.2
- IBM MQ for Windows 9.3.2
- IBM MQ for z/OS 9.3.2

MQCMDL_LEVEL_933

システム制御コマンドのレベル 933。

この値は、以下のバージョンから戻されます。

- IBM MQ for AIX 9.3.3
- IBM MQ for Linux 9.3.3
- IBM MQ for Windows 9.3.3
- IBM MQ for z/OS 9.3.3

MQCMDL_LEVEL_934

システム制御コマンドのレベル 934。

この値は、以下のバージョンから戻されます。

- IBM MQ for AIX 9.3.4
- IBM MQ for Linux 9.3.4
- IBM MQ for Windows 9.3.4
- IBM MQ for z/OS 9.3.4

MQCMDL_LEVEL_935

システム制御コマンドのレベル 935。

この値は、以下のバージョンから戻されます。

- IBM MQ for AIX 9.3.5
- IBM MQ for Linux 9.3.5

- IBM MQ for Windows 9.3.5
- IBM MQ for z/OS 9.3.5

MQCMDL_LEVEL_940

システム制御コマンドのレベル 940。

この値は、以下のバージョンから戻されます。

- IBM MQ for AIX 9.4.0
- IBM MQ for Linux 9.4.0
- IBM MQ for Windows 9.4.0
- IBM MQ for z/OS 9.4.0

特定の **CommandLevel** 属性の値に対応するシステム制御コマンドのセットは、**Platform** 属性の値によって異なります。このため、サポートされるシステム制御コマンドを調べるには、両方の属性を使用する必要があります。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_COMMAND_LEVEL セレクターを使用します。

CommandServerControl (MQLONG)

キュー・マネージャーの始動時にコマンド・サーバーを開始するかどうかを指定します。

値には以下のいずれかの値を指定できます。

MQSVC_CONTROL_MANUAL

コマンド・サーバーは自動的に開始されません。

MQSVC_CONTROL_Q_MGR

キュー・マネージャーの始動時にコマンド・サーバーを自動的に開始します。

z/OS では、この属性はサポートされていません。

属性の値を判別するには、MQINQ 呼び出しで MQIA_CMD_SERVER_CONTROL セレクターを使用します。

ConfigurationEvent (MQLONG)

構成イベントの生成を制御します。

属性の値を判別するには、MQINQ 呼び出しで MQIA_CONFIGURATION_EVENT セレクターを使用します。

値には以下のいずれかの値を指定できます。

MQEVR_DISABLED

イベント報告は無効です。

MQEVR_ENABLED

イベント報告は有効です。

Multi CurrentQFileSize (MQLONG)

最も近いメガバイトに丸めてメガバイト単位で示した、キュー・ファイルの現在のサイズ。

表 594. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

このキュー状況属性の値は、キューの現在のサイズに最も近いメガバイト数に切り上げた値です。デフォルト属性を持つ新規キューの場合、**CurrentQFileSize** の値は 1 です。

この属性の最大値は 99,999,9999 MB で、この属性にデフォルト値はありません。

Multi CurrentMaxQFileSize (MQLONG)

キューで現在使用中のブロック・サイズに基づいて、キュー・ファイルが拡張できる現在の最大サイズを最も近いメガバイトに丸めます。

表 595. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

このフィールドは、次の 2 つの用途で使用されます。

- **MaxQFileSize** を現行ブロック・サイズのデフォルト値に設定した場合、**CurrentMaxQFileSize** は、デフォルト値が等しい実際の値を示します。
- **CurrentMaxQFileSize** が **MaxQFileSize** と一致しない場合は、より大きな細分度を採用するためにキューをドレーンする必要があることが分かります。

注: キュー・ファイルのサイズ、ブロック・サイズ、および細分度の変更について詳しくは、[IBM MQ キュー・ファイルの変更](#)を参照してください。

この属性の最大値は 99,999,9999 MB で、デフォルト値はありません。この値は、現在設定されている最大値です。デフォルトの属性を持つ新しいキューの場合、**CurrentMaxQFileSize** の値は 2,088,960 MB です。

DeadLetterQName (MQCHAR48)

これは、送達不能 (未配布メッセージ) キューとしてローカル・キュー・マネージャー上に定義されたキューの名前です。メッセージは、正しい宛先に経路指定されない場合に、このキューに送られます。

例えば、次の場合に、このキューにメッセージが書き込まれます。

- メッセージがキュー・マネージャーに着信したが、宛先のキューが、そのキュー・マネージャーではまだ定義されていない。
- メッセージがキュー・マネージャーに着信したが、宛先のキューがそのメッセージを受信できない。次のような理由が考えられます。
 - キューが満杯である。
 - 書き込み要求が使用禁止になっている。
 - 送信側のノードが、キューにメッセージを書き込む許可を、持っていない。

アプリケーションは、送達不能キューにもメッセージを書き込むことができます。

レポートメッセージは通常のメッセージと同様に扱われます。レポートメッセージが宛先キュー（通常は元のメッセージのメッセージ *ReplyToQ* 記述子で指定されたキュー）に配信できない場合、そのレポートメッセージはデッドレター（未配信メッセージ）キューに配置されます。

注: 有効期限 (『MQMD - Expiry フィールド』を参照) を過ぎたメッセージは、廃棄されても、このキューには**転送されません**。ただし、送信側アプリケーションから要求があった場合は、満了レポート・メッセージ (MQRO_EXPIRATION) が生成され、*ReplyToQ* キューに送られます。

書き込み要求を発行したアプリケーションが、MQPUT または MQPUT1 呼び出しの結果、戻された理由コード (例えば、書き込み要求が使用禁止になっているローカル・キューにメッセージが入れられた) によって同期的に問題を通知された場合、メッセージは送達不能 (未配布メッセージ) キューに書き込まれません。

送達不能 (未配布メッセージ) キューのメッセージには、そのメッセージのアプリケーションのメッセージ・データに MQDLH 構造体の接頭部が付けられた情報があるものもあります。この構造体には、そのメッセージが送達不能 (未配布メッセージ) キューに書き込まれた理由を示す追加情報が入っています。この構造体の詳細については、[306 ページの『MQDLH - 送達不能ヘッダー』](#)を参照してください。

このキューは、**Usage** 属性が MQUS_NORMAL であるローカル・キューでなければなりません。

キュー・マネージャーが送達不能 (未配布メッセージ) キューをサポートしていない、あるいは送達不能キューが定義されていない場合は、名前はすべてブランクです。すべての IBM MQ キュー・マネージャーは送達不能 (未配布メッセージ) キューをサポートしますが、デフォルトでは、送達不能キューは定義されません。

送達不能 (未配布メッセージ) キューが定義されていないか、満杯になっているか、あるいは他のなんらかの理由で使用不可になっている場合には、メッセージ・チャンネル・エージェントによって送達不能キューに転送されるはずのメッセージが代わりに伝送キューに保持されます。

この属性の値を判別するには、MQINQ 呼び出しで MQCA_DEAD_LETTER_Q_NAME セレクターを使用します。この属性の長さは MQ_Q_NAME_LENGTH によって指定されます。

DefClusterXmitQueueType (MQLONG)

DefClusterXmitQueueType 属性は、cluster-sender チャンネルがメッセージを受信し、cluster-receiver チャンネルに送信する際に、デフォルトでどの送信キューが選択されるかを制御します。

DefClusterXmitQueueType の値は MQCLXQ_SCTQ または MQCLXQ_CHANNEL です。

MQCLXQ_SCTQ

すべてのクラスター送信側チャンネルは、メッセージを SYSTEM.CLUSTER.TRANSMIT.QUEUE から送信します。伝送キューに入れられたメッセージの correlID は、メッセージの宛先のクラスター送信側チャンネルを示します。

SCTQ は、キューマネージャが定義されたときに設定されます。

MQCLXQ_CHANNEL

各クラスター送信側チャンネルは、別の伝送キューからメッセージを送信します。各伝送キューは、永続的な動的キューとしてモデル・キュー SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE から作成されます。

キュー・マネージャ属性 DefClusterXmitQueueType を CHANNEL に設定すると、デフォルト構成は変更され、クラスター送信側チャンネルが個々のクラスター伝送キューと関連付けられるようになります。伝送キューは、モデル・キュー SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE から作成される永続的に動的なキューです。各伝送キューは 1 つのクラスター送信側チャンネルに関連付けられます。1 つのクラスター送信側チャンネルが 1 つのクラスター伝送キューにサービスを提供するため、伝送キューにも 1 つのクラスター内の 1 つのキュー・マネージャへのメッセージだけが入ります。クラスター内の各キュー・マネージャが使用するクラスター・キューが 1 つだけになるように構成することもできます。この場合、キュー・マネージャから各クラスター・キューへのメッセージ・トラフィックは、それぞれ他のキューへのメッセージとは別に転送されます。

値を照会するには、MQINQ を呼び出すか、または MQIA_DEF_CLUSTER_XMIT_Q_TYPE セレクターを設定して Inquire Queue Manager (MQCMD_INQUIRE_Q_MGR) の PCF コマンドを送信します。値を変更するには、MQIA_DEF_CLUSTER_XMIT_Q_TYPE セレクターを設定して Change Queue Manager (MQCMD_CHANGE_Q_MGR) の PCF コマンドを送信します。

関連資料

[Change Queue Manager](#)

[Inquire Queue Manager](#)

698 ページの『MQINQ - オブジェクト属性の照会』

MQINQ 呼び出しは、オブジェクトの属性が入っている整数の配列と一連の文字ストリングを戻します。

DefXmitQName (MQCHAR48)

使用する伝送キューが特に示されていない場合、リモート・キュー・マネージャに対するメッセージの伝送に使用される伝送キューの名前です。

デフォルトの伝送キューがない場合には、名前はすべてブランクです。この属性の初期値はブランクです。

この属性の値を判別するには、MQINQ 呼び出しで MQCA_DEF_XMIT_Q_NAME セレクターを使用します。この属性の長さは MQ_Q_NAME_LENGTH によって指定されます。

DistLists (MQLONG)

これは、ローカル・キュー・マネージャが MQPUT および MQPUT1 呼び出しにある配布リストをサポートしているかどうかを示します。これは、次の値のいずれかです。

MQDL_SUPPORTED

配布リストがサポートされています。

MQDL_NOT_SUPPORTED

配布リストはサポートされていません。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_DIST_LISTS セレクターを使用します。

DNSGroup (MQCHAR18)

このパラメーターは、今後使用されません。 [IBM MQ 8.0 の変更点](#)を参照してください。

この属性は z/OS でのみサポートされます。

この属性の値を判別するには、MQINQ 呼び出しで MQCA_DNS_GROUP セレクターを使用します。この属性の長さは MQ_DNS_GROUP_NAME_LENGTH によって指定されます。

DNSWLM (MQLONG)

このパラメーターは、今後使用されません。 [IBM MQ 8.0 の変更点](#)を参照してください。

値は、次のいずれか 1 つです。

MQDNSWLM_YES

この値は、以前のリリースからマイグレーションされたキュー・マネージャーで使用される場合があります。値は無視されます。

MQDNSWLM_NO

この値だけが、キュー・マネージャーによってサポートされます。

この属性は z/OS でのみサポートされます。

属性の値を判別するには、MQINQ 呼び出しで MQIA_DNS_WLM セレクターを使用します。

ExpiryInterval (MQLONG)

この属性は、キュー・マネージャーがキューをスキャンして有効期限切れのメッセージを探す頻度を示します。この属性には、1 から 99 999 999 の範囲の秒数で時間間隔を指定するか、以下の特殊値を指定します。

MQEXPI_OFF

キュー・マネージャーは、キューをスキャンして有効期限切れのメッセージを探しません。

この属性値を判別するには、MQINQ 呼び出しで MQIA_EXPIRY_INTERVAL セレクターを使用します。

 この属性は、z/OS でのみサポートされます。

IGQPutAuthority (MQLONG)

この属性は、ローカル・キュー・マネージャーがキュー共有グループのメンバーである場合にのみ適用されます。これは、ローカルのグループ内キューイング・エージェント (IGQ エージェント) が、共有伝送からメッセージを除去し、そのメッセージをローカル・キューに入れる際に実行される権限検査のタイプを示します。値は、次のいずれか 1 つです。

MQIGQPA_DEFAULT

許可の検査が行われたユーザー ID は、共有伝送キューにメッセージがあるときに、そのメッセージに関連している分離 MQMD の *UserIdentifier* フィールドの値です。これは、共有伝送キューにメッセージを書き込んだプログラムのユーザー ID であり、通常これは実行されているリモート・キュー・マネージャーのユーザー ID として実行されているものと同じです。

RESLEVEL プロファイルが、複数のユーザー ID が検査されることを示す場合、ローカル IGQ エージェントのユーザー ID (*IGQUserId*) も検査されます。

MQIGQPA_CONTEXT

許可の検査が行われたユーザー ID は、共有伝送キューにメッセージがあるときに、そのメッセージに関連している分離 MQMD の *UserIdentifier* フィールドの値です。これは、共有伝送キューにメッ

セージを書き込んだプログラムのユーザー ID であり、通常これは実行されているリモート・キュー・マネージャーのユーザー ID として実行されているものと同じです。

RESLEVEL プロファイルが、複数のユーザー ID が検査されることを示す場合、ローカル IGQ エージェントのユーザー ID (*IGQUserId*) と、組み込み MQMD の *UserIdentifier* フィールドの値も検査されます。後者のユーザー ID は通常は、メッセージを最初に出したアプリケーションのユーザー ID です。

MQIGQPA_ONLY_IGQ

許可の検査が行われるユーザー ID は、ローカル IGQ エージェントのユーザー ID です (*IGQUserId*)。

RESLEVEL プロファイルが複数のユーザー ID を検査することを示す場合、このユーザー ID を使用してすべての検査が行われます。

MQIGQPA_ALTERNATE_OR_IGQ

許可の検査が行われるユーザー ID は、ローカル IGQ エージェントのユーザー ID です (*IGQUserId*)。

RESLEVEL プロファイルが、複数のユーザー ID が検査されることを示す場合、組み込み MQMD の *UserIdentifier* フィールドの値も検査されます。このユーザー ID は通常は、メッセージを最初に出したアプリケーションのユーザー ID です。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_IGQ_PUT_AUTHORITY セレクターを使用します。

 この属性は、z/OS でのみサポートされます。

IGQUserId (MQLONG)

この属性は、ローカル・キュー・マネージャーがキュー共有グループのメンバーである場合にのみ適用されます。これは、ローカル・グループ内キューイング・エージェント (IGQ エージェント) に関連しているユーザー ID を指定します。この ID は、IGQ エージェントがローカル・キューにメッセージを書き込むときに、許可の検査が行われる可能性のあるユーザー ID の 1 つです。検査される実際のユーザー ID は、**IGQPutAuthority** 属性の設定および外部セキュリティ・オプションによって異なります。

IGQUserId がブランクである場合は、IGQ エージェントと関連付けられているユーザー ID はないため、対応する許可検査は実行されません (他のユーザー ID は許可の検査が行われることがあります)。

この属性の値を判別するには、MQINQ 呼び出しで MQCA_IGQ_USER_ID セレクターを使用します。属性の長さは、MQ_USER_ID_LENGTH で指定します。

 この属性は、z/OS でのみサポートされます。

InhibitEvent (MQLONG)

禁止 (読み取りおよび書き込み禁止) イベントが生成されるかどうかを制御します。値は、次のいずれか 1 つです。

MQEVR_DISABLED

イベント報告は無効です。

MQEVR_ENABLED

イベント報告は有効です。

イベントの詳細については、[イベント・モニター](#)を参照してください。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_INHIBIT_EVENT セレクターを使用します。

z/OS では、MQINQ 呼び出しを使用して、この属性の値を判別することはできません。

IntraGroupQueuing (MQLONG)

この属性は、ローカル・キュー・マネージャーがキュー共有グループのメンバーである場合にのみ適用されます。これは、キュー共有グループでグループ内キューイングが使用可能かどうかを示します。

値は、次のいずれか 1 つです。

MQIGQ_DISABLED

キュー共有グループ内の他のキュー・マネージャー宛のすべてのメッセージは、標準的なチャンネルを使用して伝送されます。

MQIGQ_ENABLED

次の条件が満たされている場合は、キュー共有グループにある他のキュー・マネージャーに書き込まれるメッセージは、共有伝送キューを使用して伝送されます。

- メッセージ・データに伝送ヘッダーを加えたものの長さが、63 KB (64 512 バイト) を超えない。

伝送ヘッダーには、MQXQH のサイズより大きいスペースを割り当てることをお勧めします。このため、定数 MQ_MSG_HEADER_LENGTH が提供されています。

この条件が満たされない場合は、従来型チャンネルを使用してメッセージが伝送されます。

注：グループ内キューイングが使用可能であるときは、従来型チャンネルを使用して伝送されたメッセージの順序と関連して、共有伝送キューを使用して伝送されたメッセージの順序が保存されることはありません。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_INTRA_GROUP_QUEUEING セレクターを使用します。

IPAddressVersion (MQLONG)

IPv4 または IPv6 のどちらの IP アドレス・バージョンを使用するかを指定します。

この属性は、IPv4 と IPv6 の両方を実行するシステムにのみ適用され、以下のいずれかの条件が当てはまる場合に、*TransportType* が MQXPY_TCP として定義されているチャンネルにのみ影響します。

- チャンネルの *ConnectionName* が IPv4 および IPv6 アドレスの両方に解決されるホスト名であり、その **LocalAddress** パラメーターが指定されていない。
- チャンネルの *ConnectionName* および *LocalAddress* がどちらも、IPv4 および IPv6 アドレスの両方に変換されるホスト名である。

値には以下のいずれかの値を指定できます。

MQIPADDR_IPV4

IPv4 が使用されます。

MQIPADDR_IPV6

IPv6 が使用されます。

属性の値を判別するには、MQINQ 呼び出しで MQIA_IP_ADDRESS_VERSION セレクターを使用します。

ListenerTimer (MQLONG)

APPC または TCP/IP が失敗した場合に、IBM MQ がリスナーの再開を試行するまでの時間間隔 (秒)。値は 5 から 9999 まででなければなりません。デフォルト値は 60 です。

この属性は z/OS でのみサポートされます。

属性の値を判別するには、MQINQ 呼び出しで MQIA_LISTENER_TIMER セレクターを使用します。

LocalEvent (MQLONG)

ローカル・エラー・イベントが生成されるかどうかを制御します。値は、次のいずれか 1 つです。

MQEVR_DISABLED

イベント報告は無効です。

MQEVR_ENABLED

イベント報告は有効です。

イベントの詳細については、[イベント・モニター](#)を参照してください。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_LOCAL_EVENT セレクターを使用します。

z/OS では、MQINQ 呼び出しを使用して、この属性の値を判別することはできません。

LoggerEvent (MQLONG)

リカバリー・ログ・イベントが生成されるかどうかを制御します。値は、次のいずれか 1 つです。

MQEVR_DISABLED

イベント報告は無効です。

MQEVR_ENABLED

イベント報告は有効です。

イベントの詳細については、[イベント・モニター](#)を参照してください。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_LOGGER_EVENT セレクターを使用します。

Multi この属性は、[マルチプラットフォーム](#)でのみサポートされます。

z/OS LUGroupName (MQCHAR8)

キュー共有グループのインバウンド伝送を処理する LU 6.2 リスナーの総称 LU 名。この名前をブランクにしておくと、このリスナーを使用することはできません。

この属性は z/OS でのみサポートされます。

この属性の値を判別するには、MQINQ 呼び出しで MQCA_LU_GROUP_NAME セレクターを使用します。属性の長さは、MQ_LU_NAME_LENGTH で指定します。

z/OS LUName (MQCHAR8)

アウトバウンド LU 6.2 伝送で使用する LU の名前。これをリスナーがインバウンド伝送で使用するのと同じ LU に設定します。この名前をブランクにしておくと、APPC/MVS デフォルト LU が使用されます。これは変数であるため、LU6.2 を使用する場合は LUName を常に設定してください。

この属性は z/OS でのみサポートされます。

属性の値を判別するには、MQINQ 呼び出しで MQCA_LU_NAME セレクターを使用します。属性の長さは、MQ_LU_NAME_LENGTH で指定します。

LU62ARMSuffix (MQCHAR2)

このチャネル・イニシエーターの LUADD を指名する、SYS1.PARMLIB メンバー APPCPMxx の接尾部。ARM がチャネル・イニシエーターを再始動すると、z/OS コマンド SET APPC=xx が発行されます。この名前をブランクにしておくと、SET APPC=xx は発行されません。

この属性は z/OS でのみサポートされます。

この属性の値を判別するには、MQINQ 呼び出しで MQCA_LU62_ARM_SUFFIX セレクターを使用します。属性の長さは、MQ_ARM_SUFFIX_LENGTH で指定します。

z/OS LU62Channels (MQLONG)

LU 6.2 伝送プロトコルを使用して、現行にできるチャネル、または接続できるクライアントの最大数。

値は 0 から 9999 までの範囲内でなければなりません。デフォルト値は 200 です。これをゼロに設定した場合、LU 6.2 伝送プロトコルは使用されません。

この属性は z/OS でのみサポートされます。

属性の値を判別するには、MQINQ 呼び出しで MQIA_LU62_CHANNELS セレクターを使用します。

MaxActiveChannels (MQLONG)

この属性は、いつでもアクティブにすることができるチャネルの最大数です。

デフォルトは、MaxChannels 属性に指定されている値です。

z/OS の場合、この値は 1 から 9 999 の範囲内でなければなりません。

他のすべてのプラットフォームでは、デフォルト値は 999 999 999 です。これは、アクティブ・チャネルが制限なしであるという意味です。これを実際の数に設定して制限を指定することもできます。

MQ Appliance の **MaxActiveChannels** 値を変更しないで IBM MQ Appliance ください。クライアントチャネルの最大数を制限したい場合は、SVRCONN チャネル定義でチャネルごとの MAXINST および

MAXINSTC 属性を使用し、各 SVRCONN チャンネルの制限を定義します。詳細は、ドキュメント IBM MQ Appliance の「IBM」MQ Appliance の「キューマネージャ構成」を参照してください。

MaxActiveChannels パラメーターは、z/OS でのみ、キュー・マネージャの属性です。その他のプラットフォームでは、**MaxActiveChannels** は `qm.ini` ファイル内の属性となります。分散キューイングの設定ファイルのスタンザを参照し、他のプラットフォームで属 **MaxActiveChannels** 性を設定する方法について確認してください。

属性の値を判別するには、**MQINQ** 呼び出しで `MQIA_ACTIVE_CHANNELS` セレクターを使用します。

関連概念

[チャンネルの状態](#)

MaxChannels (MQLONG)

この属性は、現行チャンネルにすることが可能なチャンネルの最大数です (クライアントが接続されているサーバー接続チャンネルを含みます)。

z/OS の場合、この値は 1 から 9 999 の範囲内でなければなりません。デフォルト値は 200 です。

MQ Appliance IBM MQ Appliance では、デフォルト値は 999 999 999 です。この値は変更しないでください。クライアントチャンネルの最大数を制限したい場合は、SVRCONN チャンネル定義でチャンネルごとの MAXINST および MAXINSTC 属性を使用し、各 SVRCONN チャンネルの制限を定義します。詳細は、ドキュメント IBM MQ Appliance の「IBM」MQ Appliance の「キューマネージャ構成」を参照してください。

ネットワークからの接続を処理するためにシステム・ビジーが発生している場合、デフォルトの設定よりも高い値が必要になる可能性があります。テスト時にシステムの動作を観察して、ご使用の環境に適切な値を決定するのが理想的です。

その他のすべてのプラットフォームのデフォルト値は 100 です。必要に応じて、**MaxChannels** を別の値に設定し、現行チャンネルの最大数を制限することができます。

MaxChannels パラメーターは、z/OS でのみ、キュー・マネージャの属性です。その他のプラットフォームでは、**MaxChannels** は `qm.ini` ファイル内の属性となります。分散キューイングの設定ファイルのスタンザを参照し、他のプラットフォームで属 **MaxChannels** 性を設定する方法について確認してください。

属性の値を判別するには、**MQINQ** 呼び出しで `MQIA_MAX_CHANNELS` セレクターを使用します。

関連概念

[チャンネルの状態](#)

MaxHandles (MQLONG)

任意の 1 つのタスクが並行して使用できるオープン・ハンドルの最大数です。ある 1 つのキュー (またはキュー以外のオブジェクト) に対する **MQOPEN** 呼び出しが正常に実行されるたびに、ハンドルが 1 つ使用されます。オープンしたオブジェクトをクローズすると、そのハンドルを再使用できるようになります。ただし、配布リストをオープンしたときには、配布リスト内のそれぞれのキューに個別のハンドルが割り振られるため、**MQOPEN** 呼び出しによって、配布リスト内にあるキューと同じ数のハンドルが使用されます。**MaxHandles** の適切な値を判断するときには、このことを考慮する必要があります。

MQPUT1 呼び出しでは、処理の一部として **MQOPEN** 呼び出しが実行されます。このため、**MQOPEN** と同じ数のハンドルが使用されますが、**MQPUT1** でハンドルが使用されるのは、その **MQPUT1** 呼び出し自身が実行されている間のみです。

z/OS では、タスクは、CICS タスク、MVS タスク、または IMS 従属領域を意味します。

値の範囲は、1 から 999 999 999 です。デフォルト値は、環境により決まります。

- z/OS の場合、デフォルト値は 100 です。
- それ以外のすべての環境では、デフォルト値は 256 です。

この属性の値を判別するには、**MQINQ** 呼び出しで `MQIA_MAX_HANDLES` セレクターを使用します。

MaxMsgLength (MQLONG)

キュー・マネージャーが処理できる物理メッセージの最大長です。ただし、**MaxMsgLength** キュー・マネージャー属性は、**MaxMsgLength** キュー属性とは別に設定できるため、キューに入れることのできる物理メッセージの最大長は、これら 2 つの値の小さい方の値になります。

キュー・マネージャーがセグメント化をサポートしている場合は、アプリケーションで MQMF_SEGMENTATION_ALLOWED フラグを MQMD に指定しているときに限って、これら 2 つの **MaxMsgLength** 属性の小さい方の値より長い論理メッセージをアプリケーションから書き込むことができます。そのフラグを指定する場合、論理メッセージの長さの上限は 999 999 999 バイトですが、多くの場合、オペレーティング・システムによって、またはアプリケーションが実行されている環境によってリソースが制約される結果、上限はこれよりさらに小さい値になります。

MaxMsgLength 属性の下限は 32 KB (32 768 バイト) です。上限は 100 MB (104 857 600 バイト) です。この属性の値を判別するには、MQINQ 呼び出しで MQIA_MAX_MSG_LENGTH セレクターを使用します。

MaxPriority (MQLONG)

これは、キュー・マネージャーによってサポートされる最高のメッセージ優先順位です。優先順位の範囲は、ゼロ (最低) から *MaxPriority* (最高) までです。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_MAX_PRIORITY セレクターを使用します。

MaxPropertiesLength (MQLONG)

これは、メッセージと一緒に流すことができるプロパティのサイズを制御するために使用します。このサイズには、プロパティ名 (バイト単位) とプロパティ値のサイズ (バイト単位) の両方が含まれます。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_MAX_PROPERTIES_LENGTH セレクターを使用します。

MaxQFileSize (MQLONG)

キュー・ファイルを拡張できる最大サイズ (メガバイト単位)。

表 596. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

この値を現在のキュー・ファイル・サイズより小さい値に構成している場合、キュー・ファイルが最大サイズを超える可能性があります。その場合、キュー・ファイルは新しいメッセージを受け入れなくなりますが、既存のメッセージはコンSUMEできます。キュー・ファイル・サイズを構成済みの値より小さくすると、新しいメッセージをキューに書き出すことができるようになります。

注: この数値は、キューで構成されている属性の値とは異なる場合があります。これは、キュー・マネージャーが、選択されたサイズに到達するために、より大きなブロック・サイズを内部で使用する必要がある可能性があるためです。キュー・ファイルのサイズ、ブロック・サイズ、および細分度の変更について詳しくは、[IBM MQ キュー・ファイルの変更](#)を参照してください。

この属性を増やしたために細分度の変更が必要になる場合は、警告メッセージ AMQ7493W「細分度が変更されました (Granularity changed)」が AMQERR ログに書き込まれます。これにより IBM MQ が新しい細分度を採用するために、キューを空にするための計画が必要となることが示されます。

この属性の最大値は 267,386,880 MB であり、デフォルト値およびマイグレーションされた値は 2,088,960 MB です。これは、細分度が 512 であるキューの現在の最大値です。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_MAX_Q_FILE_SIZE セレクターを使用します。

MaxUncommittedMsgs (MQLONG)

これは、1 つの作業単位に存在できるコミットされていないメッセージの最大数です。コミットされていないメッセージの数は、現行の作業単位が開始されてからの以下の合計です。

- アプリケーションが MQPMO_SYNCPOINT オプションを指定して書き込んだメッセージの数

- アプリケーションが MQGMO_SYNCPOINT オプションを指定して取り出したメッセージの数
- MQPMO_SYNCPOINT オプションを指定して書き込まれたメッセージについてキュー・マネージャーが生成したトリガー・メッセージと COA レポート・メッセージの数
- MQGMO_SYNCPOINT オプションを指定して取り出されたメッセージについてキュー・マネージャーが生成した COD レポート・メッセージの数

以下のメッセージはコミットされていないメッセージとしては数えません。

- 作業単位外でアプリケーションが書き込みまたは検索したメッセージ
- トリガー・メッセージまたは COA/COD レポート・メッセージのうち、作業単位外で書き込んだ、または検索したメッセージの結果としてキュー・マネージャーが生成したメッセージ
- キュー・マネージャーが生成した満了レポート・メッセージ (満了レポート・メッセージを発生させた呼び出しに MQGMO_SYNCPOINT が指定された場合も含まれます。)
- キュー・マネージャーが生成したイベント・メッセージ (イベント・メッセージを発生させた呼び出しに MQGMO_SYNCPOINT または MQGMO_SYNCPOINT が指定された場合も含まれます。)

注:

1. 例外レポート・メッセージは、メッセージ・チャネル・エージェント (MCA) またはアプリケーションによって生成され、アプリケーションが書き込んだ、または検索した通常のメッセージと同様に扱われます。
2. メッセージまたはセグメントが MQPMO_SYNCPOINT オプションを指定して書き込まれると、その書き込みの結果として実際に発行される物理メッセージの数にかかわらず、コミットされていないメッセージの数が 1 つずつ増分されます。(キュー・マネージャーがメッセージまたはセグメントを分割する必要がある場合、複数の物理メッセージが発行されることがあります。)
3. 配布リストが MQPMO_SYNCPOINT オプションを指定して書き込まれると、コミットされていないメッセージの数は、生成される物理メッセージごとに 1 つずつ増分されます。これは、配布リスト中の宛先の数と同じである必要はありません。

この属性の下限は 1 で、上限は 999 999 999 です。デフォルト値は 10000 です。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_MAX_UNCOMMITTED_MSGS セレクターを使用します。

MQIAccounting (MQLONG)

MQI データのアカウンティング情報のコレクションを制御します。

値は、次のいずれか 1 つです。

MQMON_ON

API アカウンティング・データを収集します。

MQMON_OFF

API アカウンティング・データを収集しません。これがデフォルト値です。

キュー・マネージャー属性 ACCTCONO を ENABLED に設定した場合、MQCNO 構造の Options フィールドを使用する個別の接続でこの値はオーバーライドされる可能性があります。この値の変更は、属性を変更した後に行われるキュー・マネージャーへの接続でのみ有効です。

この属性は、[マルチプラットフォーム](#)でのみサポートされます。

属性の値を判別するには、MQINQ 呼び出しで MQIA_ACCOUNTING_MQI セレクターを使用します。

MQIStatistics (MQLONG)

これは、キュー・マネージャーの統計モニター情報のコレクションを制御します。

値は、次のいずれか 1 つです。

MQMON_ON

MQI 統計を収集します。

MQMON_OFF

MQI 統計を収集しません。これがデフォルト値です。

この属性は、[マルチプラットフォーム](#)でのみサポートされます。

属性の値を判別するには、MQINQ 呼び出しで MQIA_STATISTICS_MQI セレクターを使用します。

MsgMarkBrowseInterval (MQLONG)

キュー・マネージャーが自動的にマークをブラウズ・メッセージから削除できるようになった後のミリ秒単位の時間間隔

これは、キュー・マネージャーが自動的にマークをブラウズ・メッセージから削除できるようになった後の時間間隔 (ミリ秒単位) です。

この属性では、読み取りメッセージ・オプション MQGMO_MARK_BROWSE_CO_OP を使用して、MQGET 呼び出しにより参照済みとしてマークが付けられたメッセージに、参照済みのマークを付けたままにしておく予定の時間間隔を示します。

このおおよその時間間隔より長くメッセージに参照済みマークが付けられていると、連動するハンドルのセットで参照済みマークが付けられていた参照済みメッセージから、キュー・マネージャーが自動的にマーク解除する場合があります。

読み取りメッセージ・オプション MQGMO_MARK_BROWSE_HANDLE を使用した、MQGET に対する呼び出しで参照済みとマークされたメッセージの状態に影響を与えるものではありません。

最大値は 999 999 999、デフォルト値は 5000 です。MsgMarkBrowseInterval の特殊値 -1 は、無制限の時間間隔を表します。



重要: この値をデフォルトの 5000 より小さくしないでください。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_MSG_MARK_BROWSE_INTERVAL セレクターを使用します。

z/OS OutboundPortMax (MQLONG)

これは、出力チャネルをバインドするために使用するポート番号の (OutboundPortMin および OutboundPortMax によって定義される) 範囲の中で最も高いポート番号です。

この値は、0 から 65535 までの範囲の整数で、OutboundPortMin 値以上でなければなりません。デフォルト値は 0 です。

この属性は z/OS でのみサポートされます。

属性の値を判別するには、MQINQ 呼び出しで MQIA_OUTBOUND_PORT_MAX セレクターを使用します。

z/OS OutboundPortMin (MQLONG)

これは、出力チャネルをバインドするために使用するポート番号の (OutboundPortMin および OutboundPortMax によって定義される) 範囲の中で最も低いポート番号です。

この値は、0 から 65535 までの範囲の整数で、OutboundPortMax 値以下でなければなりません。デフォルト値は 0 です。

この属性は z/OS でのみサポートされます。

属性の値を判別するには、MQINQ 呼び出しで MQIA_OUTBOUND_PORT_MIN セレクターを使用します。

PerformanceEvent (MQLONG)

パフォーマンス関連のイベントが生成されるかどうかを制御します。これは、次の値のいずれかです。

MQEVR_DISABLED

イベント報告は無効です。

MQEVR_ENABLED

イベント報告は有効です。

イベントの詳細については、[イベント・モニター](#)を参照してください。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_PERFORMANCE_EVENT セレクターを使用します。

Platform (MQLONG)

これは、キュー・マネージャーが実行されているオペレーティング・システムを示します。

MQPL_AIX

AIX (MQPL_UNIX と同じ値)

MQPL_APPLIANCE

IBM MQ Appliance

MQPL_MVS

z/OS (MQPL_ZOS と同じ値)

MQPL_OS390

z/OS (MQPL_ZOS と同じ値)

MQPL_OS400

IBM i.

MQPL_UNIX

UNIX.

MQPL_WINDOWS_NT

Windows システム

MQPL_ZOS

z/OS.

この属性の値を判別するには、MQINQ 呼び出しで MQIA_PLATFORM セレクターを使用します。

PubSubNPInputMsg (MQLONG)

未送達の入力メッセージを廃棄または保持するかどうか

値は、次のいずれか 1 つです。

MQUNDELIVERED_DISCARD

非持続入力メッセージは、処理できない場合は廃棄されることがあります。

これがデフォルト値です。

MQUNDELIVERED_KEEP

非持続入力メッセージは、処理できない場合でも廃棄されません。この状態では、キューに入れられたパブリッシュ/サブスクライブ・インターフェースが適切な間隔で処理の再試行を続行し、後続のメッセージ処理を続けることはありません。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_PUBSUB_NP_MSG セレクターを使用します。

PubSubNPResponse (MQLONG)

未送達の応答メッセージの動作を制御する。

値は、次のいずれか 1 つです。

MQUNDELIVERED_NORMAL

応答キューに入れることができない非持続応答は、送達不能キューに入れられます。送達不能キューにも入れることができない場合は、破棄されます。

MQUNDELIVERED_SAFE

応答キューに入れることができない非持続応答は送達不能キューに入れられます。応答が設定できず、送達不能キューに入れることができない場合は、キューに入れられたパブリッシュ/サブスクライブ・インターフェースが現行のオペレーションをロールバックし、適切な間隔で再試行します。後続のメッセージを続けて処理することはありません。

MQUNDELIVERED_DISCARD

応答キューに入れられない非持続応答は破棄されます。

これは、新規キュー・マネージャーの場合のデフォルト値です。

MQUNDELIVERED_KEEP

非持続応答は送達不能キューに入れられず、廃棄はされない。代わりに、キューに入れられたパブリッシュ/サブスクライブ・インターフェースが現行オペレーションをバックアウトし、適切な間隔で再試行します。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_PUBSUB_NP_RESP セレクターを使用します。

移行されたキュー・マネージャーの場合のデフォルト値。

キュー・マネージャーが IBM MQ V6.0 から移行された場合、この属性の初期値は、次の表に示すように、移行前の *DiscardNonPersistentResponse* および *DLQNonPersistentResponse* の値に依存します。

		DLQNonPersistentResponse		
		はい	いいえ	設定なし
DiscardNonPersistentResponse	はい	MQUNDELIVERED_NORMAL	MQUNDELIVERED_DISCARD	MQUNDELIVERED_NORMAL
	いいえ	MQUNDELIVERED_SAFE	MQUNDELIVERED_KEEP	MQUNDELIVERED_SAFE
	設定されない	SyncPointPersistent = No の場合は MQUNDELIVERED_SAFE、それ以外の場合は MQUNDELIVERED_NORMAL	SyncPointPersistent = No の場合は MQUNDELIVERED_KEEP、それ以外の場合は MQUNDELIVERED_DISCARD	SyncPointPersistent = No の場合は MQUNDELIVERED_SAFE、それ以外の場合は MQUNDELIVERED_NORMAL

PubSubMaxMsgRetryCount (MQLONG)

失敗したコマンド・メッセージを同期点で処理する際の再試行の数

値は、次のいずれか 1 つです。

0 - 999 999 999

デフォルト値は 5 です。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_PUBSUB_MAXMSG_RETRY_COUNT セレクターを使用します。

PubSubSyncPoint (MQLONG)

同期点で永続メッセージのみが処理されるのか、すべてのメッセージが処理されるのか。

値は、次のいずれか 1 つです。

MQSYNCPPOINT_IFPER

待機中のパブリッシュ/サブスクライブ・インターフェースが非持続メッセージを同期点外で受け取るようにします。デーモンが同期点外のパブリケーションを受け取る場合、デーモンはパブリケーションを、同期点外の認識されたサブスクライバーに転送します。

これがデフォルト値です。

MQSYNCPPOINT_YES

待機中のパブリッシュ/サブスクライブ・インターフェースがすべてのメッセージを同期点下で受け取るようにします。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_PUBSUB_SYNC_PT セレクターを使用します。

PubSubMode (MQLONG)

パブリッシュ/サブスクライブ・エンジンおよびキューに入れられたパブリッシュ/サブスクライブ・インターフェースが実行中なので、アプリケーション・プログラミング・インターフェースと、キューに入れられたパブリッシュ/サブスクライブ・インターフェースによってモニターされているキューを使用して、アプリケーションがパブリッシュ/サブスクライブできるかどうか。

値は、次のいずれか 1 つです。

MQPSM_COMPAT

パブリッシュ/サブスクライブ・エンジンが実行中。したがって、アプリケーション・プログラミング・インターフェースを使用してパブリッシュ/サブスクライブできます。キュー・パブリッシュ/サブスクライブ・インターフェースは実行されていないため、キュー・パブリッシュ/サブスクライブ・インターフェースがモニターするキューに書き込まれるメッセージは処理されません。この設定は、キューに入れられたパブリッシュ/サブスクライブ・インターフェースの通常の読み取り元と同じキューを

読み取る必要があるので、このキュー・マネージャーを使用する WebSphere Message Broker V6 以前のバージョンとの互換用に使用されます。

MQPSM_DISABLED

パブリッシュ/サブスクライブ・エンジンとキュー・パブリッシュ/サブスクライブ・インターフェースはどちらも実行されていません。したがって、アプリケーション・プログラミング・インターフェースを使用してパブリッシュ/サブスクライブできません。キュー・パブリッシュ/サブスクライブ・インターフェースがモニターするキューに書き込まれるパブリッシュ/サブスクライブ・メッセージは処理されません。

MQPSM_ENABLED

パブリッシュ/サブスクライブ・エンジンとキュー・パブリッシュ/サブスクライブ・インターフェースはどちらも実行されています。したがって、アプリケーション・プログラミング・インターフェースと、キューに入れられたパブリッシュ/サブスクライブ・インターフェースによってモニターされているキューを使用してパブリッシュ/サブスクライブできます。これがキュー・マネージャーの初期デフォルト値です。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_PUBSUB_MODE セレクターを使用します。

QMGrDesc (MQCHAR64)

このフィールドはキュー・マネージャーについて説明するコメント用に使用します。フィールドの内容はキュー・マネージャーにとって重要なものではありませんが、表示できる文字以外は使用しないでください。フィールドにヌル文字を入れることはできません。また、必要に応じて、右側がブランクで埋められます。DBCS をインストール済みの環境では、このフィールドに DBCS 文字を入れることができます (最大フィールド長として 64 バイトが適用されます)。

注: このフィールドに、(CodedCharSetId キュー・マネージャー属性で定義されている) キュー・マネージャーの文字セットに含まれていない文字が含まれている場合、このフィールドが別のキュー・マネージャーに送信されると、それらの文字が正しく変換されない可能性があります。

- z/OS では、製品名とバージョン番号がデフォルト値になります。
- それ以外のすべての環境では、デフォルト値はブランクです。

この属性の値を判別するには、MQINQ 呼び出しで MQCA_Q_MGR_DESC セレクターを使用します。属性の長さは、MQ_Q_MGR_DESC_LENGTH で指定します。

QMGrIdentifier (MQCHAR48)

これは、キュー・マネージャーの内部的に生成される固有の名前です。

この属性の値を判別するには、MQINQ 呼び出しで MQCA_Q_MGR_IDENTIFIER セレクターを使用します。属性の長さは、MQ_Q_MGR_IDENTIFIER_LENGTH で指定します。

この属性は、次の環境でサポートされます。

-  AIX
-  IBM i
-  Linux
-  Windows
-  z/OS

および、これらのシステムに接続された IBM MQ クライアント。

QMGrName (MQCHAR48)

これは、ローカル・キュー・マネージャーの名前、つまりアプリケーションが接続されているキュー・マネージャーの名前です。

この名前の先頭の 12 文字は、固有のメッセージ ID を生成するために使用されます (『MQMD - MsgId フィールド』を参照してください)。したがって、相互通信するキュー・マネージャーには、キュー・マネー

ャー・ネットワーク内でのメッセージ ID がそれぞれ固有なものになるように、先頭の 12 文字をそれぞれ区別できるような名前を付ける必要があります。

サブシステム名と同じ名前 z/OS であり、4 文字以内の空白を含まない文字に制限されます。

この属性の値を判別するには、MQINQ 呼び出しで MQCA_Q_MGR_NAME セレクターを使用します。属性の長さは、MQ_Q_MGR_NAME_LENGTH で指定します。

QSGName (MQCHAR4)

ローカル・キュー・マネージャーが属するキュー共有グループの名前です。ローカル・キュー・マネージャーがキュー共有グループに属していない場合、名前はブランクになります。

この属性の値を判別するには、MQINQ 呼び出しで MQCA_QSG_NAME セレクターを使用します。属性の長さは、MQ_QSG_NAME_LENGTH で指定します。

 この属性は、z/OS でのみサポートされます。

QueueAccounting (MQLONG)

キューのアカウントリング情報のコレクションを制御します。

値は、次のいずれか 1 つです。

MQMON_NONE

キューのアカウントリング属性 ACCTQ の設定にかかわらず、キューのアカウントリング・データを収集しません。これがデフォルト値です。

MQMON_OFF

ACCTQ キュー属性で QMGR を指定するキューではアカウントリング・データを収集しません。

MQMON_ON

ACCTQ キュー属性で QMGR を指定するキューではアカウントリング・データを収集します。

この値の変更は、属性を変更した後に行われるキュー・マネージャーへの接続でのみ有効です。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_ACCOUNTING_Q セレクターを使用します。

QueueMonitoring (MQLONG)

これは、キューのオンライン・モニタリングのデフォルト設定を指定します。

QueueMonitoring キュー属性が MQMON_Q_MGR に設定されている場合、この属性はチャンネルによって仮定される値を指定します。値は次のいずれかです。

MQMON_OFF

オンライン・モニター・データ収集をオフにします。これがキュー・マネージャーの初期デフォルト値です。

MQMON_NONE

キューの **QueueMonitoring** 属性の設定に関係なく、キューのオンライン・モニター・データ収集はオフになります。

MQMON_LOW

オンライン・モニター・データ収集を、低いデータ収集率でオンにします。

MQMON_MEDIUM

オンライン・モニター・データ収集を、中程度のデータ収集率でオンにします。

MQMON_HIGH

オンライン・モニター・データ収集を、高いデータ収集率でオンにします。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_MONITORING_Q セレクターを使用します。

QueueStatistics (MQLONG)

キューの統計データのコレクションを制御します。

これは、次の値のいずれかです。

MQMON_NONE

QueueStatistics キュー属性の設定にかかわらず、キューの統計データを収集しません。これがデフォルト値です。

MQMON_OFF

QueueStatistics キュー属性で Queue Manager を指定するキューでは統計データを収集しません。

MQMON_ON

QueueStatistics キュー属性で Queue Manager を指定するキューでは統計データを収集します。

属性の値を判別するには、MQINQ 呼び出しで MQIA_STATISTICS_Q セレクターを使用します。

ReceiveTimeout (MQLONG)

TCP/IP チャンネルが、ハートビートを含むデータをそのパートナーから受信してから、非アクティブ状態に戻るまで待機する時間を指定します。これは、メッセージ・チャンネルにのみ適用され、MQI チャンネルには適用されません。

ReceiveTimeout の正確な意味は、ReceiveTimeoutType に指定された値によって変わります。

ReceiveTimeoutType は、次のうちいずれか 1 つに設定できます。

- MQRCVTIME_EQUAL - この値は、チャンネルが待機する秒数です。0 から 999999 の範囲の値を指定します。
- MQRCVTIME_ADD - この値は、折衝された HBINT に加算する秒数で、チャンネルが待機する長さを決定します。1 から 999999 の範囲の値を指定します。
- MQRCVTIME_MULTIPLY - この値は、折衝された HBINT に適用される乗数です。値を 0 または 2 から 99 までの範囲で指定します。

デフォルト値は 0 です。

チャンネルがパートナーからのデータの受信を待機中にタイムアウトになることを止めるには、ReceiveTimeoutType を MQRCVTIME_MULTIPLY または MQRCVTIME_EQUAL に設定して、ReceiveTimeout を 0 に設定してください。

この属性は z/OS でのみサポートされます。

属性の値を判別するには、MQINQ 呼び出しで MQIA_RECEIVE_TIMEOUT セレクターを使用します。

ReceiveTimeoutMin (MQLONG)

TCP/IP チャンネルが、ハートビートを含むデータをそのパートナーから受信してから、非アクティブ状態に戻るまで待機する最小時間 (秒)。

これは、メッセージ・チャンネルにのみ適用されます。MQI チャンネルには適用されません。この値は 0 から 999999 までの範囲内でなければなりません。デフォルトは 0 です。

ReceiveTimeoutType を使用して TCP/IP チャンネルの待ち時間を折衝値 HBINT と比較して計算することを指定し、その結果の値がこのパラメーターの値より小さい場合、この値が代わりに使用されます。

この属性は z/OS でのみサポートされます。

属性の値を判別するには、MQINQ 呼び出しで MQIA_RECEIVE_TIMEOUT_MIN セレクターを使用します。

ReceiveTimeoutType (MQLONG)

TCP/IP チャンネルが、ハートビートを含むデータをそのパートナーから受信してから、非アクティブ状態に戻るまで待機する時間を定義する ReceiveTimeout に適用される修飾子。これは、メッセージ・チャンネルにのみ適用されます。MQI チャンネルには適用されません。

値は、次のいずれか 1 つです。

MQRCVTIME_MULTIPLY

ReceiveTimeout は、折衝 HBINT 値に適用される乗数であり、チャンネルが待機する時間を決定します。これがデフォルト値です。

MQRCVTIME_ADD

ReceiveTimeout は、折衝 HBINT 値に追加される値 (秒数) であり、チャンネルが待機する時間を決定します。

MQRCVTIME_EQUAL

ReceiveTimeout は、チャンネルが待機する値 (秒) です。

チャンネルがそのパートナーからデータを受け取るまでにタイムアウトになるのを停止するには、ReceiveTimeoutType を MQRCVTIME_MULTIPLY または MQRCVTIME_EQUAL に設定し、ReceiveTimeout を 0 に設定します。

この属性は z/OS でのみサポートされます。

属性の値を判別するには、MQINQ 呼び出しで MQIA_RECEIVE_TIMEOUT_TYPE セレクターを使用します。

RemoteEvent (MQLONG)

リモート・エラー・イベントが生成されるかどうかを制御します。これは、次の値のいずれかです。

MQEVR_DISABLED

イベント報告は無効です。

MQEVR_ENABLED

イベント報告は有効です。

イベントの詳細については、[イベント・モニター](#)を参照してください。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_REMOTE_EVENT セレクターを使用します。

RepositoryName (MQCHAR48)

これは、このキュー・マネージャーがリポジトリ・マネージャーのサービスを提供するクラスターの名前です。キュー・マネージャーが複数のクラスターにこのサービスを提供する場合は、それらのクラスターを識別する名前リスト・オブジェクトの名前を *RepositoryNameList* において指定し、*RepositoryName* はブランクにします。*RepositoryName* および *RepositoryNameList* の少なくとも 1 つがブランクでなければなりません。

この属性の値を判別するには、MQINQ 呼び出しで MQCA_REPOSITORY_NAME セレクターを使用します。属性の長さは、MQ_Q_MGR_NAME_LENGTH で指定します。

RepositoryNameList (MQCHAR48)

これは、このキュー・マネージャーがリポジトリ・マネージャーのサービスを提供するクラスターの名前を含む名前リスト・オブジェクトの名前です。キュー・マネージャーが 1 つのクラスターだけにこのサービスを提供する場合、名前リスト・オブジェクトには 1 つの名前だけが含まれます。代わりに *RepositoryName* を使用して、そのクラスターの名前を指定できます。この場合、*RepositoryNameList* はブランクにします。*RepositoryName* および *RepositoryNameList* の少なくとも 1 つがブランクでなければなりません。

この属性の値を判別するには、MQINQ 呼び出しで MQCA_REPOSITORY_NAMELIST セレクターを使用します。属性の長さは、MQ_NAMELIST_NAME_LENGTH で指定します。

ScyCase(MQCHAR8)

キュー・マネージャーが大/小文字混合のセキュリティー・プロファイル名をサポートするか、または大文字のみのセキュリティー・プロファイル名をサポートするかを指定します。

値は、次のいずれか 1 つです。

MQSCYC_UPPER

セキュリティー・プロファイル名は大文字でなければなりません。

MQSCYC_MIXED

セキュリティー・プロファイル名は大文字または大/小文字混合にすることができます。

この属性の変更は、*SecurityType (MQSECTYPE_CLASSES)* を指定して Refresh Security コマンドを実行したときに有効になります。

 この属性は、z/OS でのみサポートされます。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_SECURITY_CASE セレクターを使用します。

SharedQMgrName (MQLONG)

これは、*ObjectQmgrName* が同じキュー共有グループに属する別のキュー・マネージャーの名前である場合、共有キューの MQOPEN 呼び出しで、*ObjectQmgrName* をローカル・キュー・マネージャーとして使用または処理するかどうかを指定します。

値には以下のいずれかの値を指定できます。

MQSQM_USE

ObjectQmgrName が使用され、適切な伝送キューがオープンされます。

MQSQM_IGNORE

ターゲット・キューが共有されており、*ObjectQmgrName* が同じキュー共有グループのキュー・マネージャーの名前である場合、オープンはローカルに実行されます。

この属性は z/OS でのみ有効です。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_SHARED_Q_Q_MGR_NAME セレクターを使用します。

SPLCAP

キュー・マネージャーで Advanced Message Security のセキュリティ機能が使用可能かどうかを示します。

MQCAP_SUPPORTED

キュー・マネージャーが実行されているインストール済み環境に AMS コンポーネントがインストールされている場合、これがデフォルト値になります。

MQCAP_NOT_SUPPORTED

SSLEvent (MQLONG)

TLS イベントを生成するかどうかを指定します。

これは、次の値のいずれかです。

MQEVR_ENABLED

TLS イベントを以下のように生成します。

MQRC_CHANNEL_SSL_ERROR

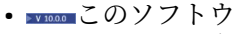
MQEVR_DISABLED

TLS イベントを生成しません。これはデフォルト値です。

属性の値を判別するには、MQINQ 呼び出しで MQIA_SSL_EVENT セレクターを使用します。

SSLFIPSRequired (MQLONG)

注：

-  WindowsAIX、Linux、およびにおいて、IBM MQ は GSKit 9 IBM Crypto for C (ICC) 暗号モジュールを通じてへの準拠を実現 FIPS 140-3 しています。この FIPS 140-3 モジュールに関連する NIST 認証は、<https://csrc.nist.gov/projects/cryptographic-module-validation-program/certificate/4755> で確認できます。
-  Linux s390xFIPS 現在、への対応は行っていません。IBM MQ このプラットフォームをご利用のお客様でサポートが必要な FIPS 場合は、以前のバージョンのままご利用ください。このプラットフォーム上で暗号モジュールが IBM Crypto for C (ICC) 認証を取得次第、今後の FixPack でサポートが FIPS 140-3 有効化されます。
-  このソフトウェアに含まれる暗号 FIPS 140-3 モジュールは、2024 IBM Semeru Runtime 年 8 月に NIST の承認を受けた。IBM Semeru Runtime 11+IBM MQ 10.0.0 TLS を使用した および IBM MQ classes for Java のクライアント接続の IBM MQ classes for JMS 処理機能を、FIPS 140-3 およびに Java

8 追加しました。 <https://csrc.nist.gov/projects/cryptographic-module-validation-program/certificate/4755> この FIPS 140-3 モジュールに関連する NIST 認証は、以下で確認できます。

- Containers では IBM MQ、3.2.0 およびキューマネージャ IBM MQ Operator ーのコンテナイメージ 9.4.0.0 以降は、UBI 9 をベースとしています。FIPS 140-3 Containers における IBM MQ コンプライアンスは、現在審査中です。

 が有効になっている FIPS 場合、コンテナ制御 IBM MQ プロセスでは、認定 FIPS 140-3 OpenSSL モジュールが使用されます。NIST 認証の詳細については、以下の URL をご覧ください： <https://access.redhat.com/compliance/fips>。IBM MQ コンテナイメージ内で実行されるキューマネージャは、ベースイメージのプラットフォームバージョンと同じ FIPS IBM MQ 認証レベルを持っています。

これにより、暗号処理が暗号ハードウェアではなくソフトウェアで実行される場合、認証済みアルゴリズム FIPS IBM MQ のみを使用するよう指定できます。暗号ハードウェアが設定されている場合、使用される暗号モジュールは当該ハードウェア製品が提供するモジュールである。これらのモジュールは、使用中のハードウェア製品に応じて特定のレベルで認証 FIPS されている場合もあれば、そうでない場合もある。

値は、以下のいずれかの値です。

MQSSL_FIPS_NO

使用中のプラットフォームでサポートされている CipherSpec を使用します。この値がデフォルト値です。

MQSSL_FIPS_YES

 TLS、AMS、パスワード保護、JWT 検証を含むすべての暗号化機能に対して、FIPS 認定の暗号ライブラリおよびアルゴリズムを使用すること。

 このキューマネージャとのすべての TLS 接続において、CipherSpecs で許可されている認定暗号アルゴリズムのみを使用してください FIPS。

このパラメータは、z/OS、AIX、Linux、and Windows の各プラットフォームでのみ有効です。

属性の値を判別するには、MQINQ 呼び出しで MQIA_SSL_FIPS_REQUIRED セレクターを使用します。

関連タスク

[MQI クライアントの実行時に、認証済みの CipherSpecs のみ FIPS を使用するように指定する](#)

関連資料

[Federal Information Processing Standards \(FIPS\) について AIX、Linux、and Windows](#)

SSLKeyResetCount (MQLONG)

通信を開始する TLS チャンネルのメッセージ・チャンネル・エージェント (MCA) が、チャンネルの暗号化に使用される共通鍵をリセットするタイミングを指定します。

この値は、秘密鍵を再折衝するまでにチャンネルで送受信される暗号化されていない合計バイト数を表します。このバイト数には、MCA によって送信される制御情報が含まれます。

この値は、0 から 999 999 999 までの範囲の数値です。デフォルト値は 0 です。TLS 秘密鍵のリセット・カウントを 1 バイトから 32 KB の範囲で指定すると、TLS チャンネルは 32 KB の秘密鍵リセット・カウントを使用します。これは、TLS 秘密鍵リセット値が小さい場合に生じる、過剰な鍵リセットによる処理コストを避けるためです。

秘密鍵が再折衝されるのは、開始側のチャンネル MCA によって送受信された、暗号化されていないバイトの総数が指定値を超えた場合です。チャンネル・ハートビートが有効な場合は、チャンネル・ハートビート後にデータが送受信される前か、暗号化されていないバイトの総数が指定値を超えた時か、そのいずれか早いほうの時点で、秘密鍵が再折衝されます。

再折衝用に送受信されるバイト・カウントには、チャンネル MCA によって送受信される制御情報が含まれ、再折衝が行われるときにリセットされます。

共通鍵が再折衝されないことを指定する場合には、値 0 を使用してください。

属性の値を判別するには、MQINQ 呼び出しで MQIA_SSL_RESET_COUNT セレクターを使用します。

StartStopEvent (MQLONG)

開始イベントおよび停止イベントが生成されるかどうかを制御します。値は、次のいずれか 1 つです。

MQEVR_DISABLED

イベント報告は無効です。

MQEVR_ENABLED

イベント報告は有効です。

イベントの詳細については、[イベント・モニター](#)を参照してください。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_START_STOP_EVENT セレクターを使用します。

StatisticsInterval (MQLONG)

統計モニター・データをモニター中のキューに書き込む頻度 (秒) を指定します。

この値は、0 から 604800 までの範囲の整数です。デフォルト値は 1800 (30 分) です。

属性の値を判別するには、MQINQ 呼び出しで MQIA_STATISTICS_INTERVAL セレクターを使用します。

SyncPoint (MQLONG)

これは、ローカル・キュー・マネージャーが複数の作業単位をサポートし、MQGET、MQPUT、および MQPUT1 呼び出しによる同期化をサポートするかどうかを示します。

MQSP_AVAILABLE

作業単位および同期化が可能。

MQSP_NOT_AVAILABLE

作業単位および同期化が不可。

- z/OS では、この値が戻されることはありません。

この属性を判断するには、MQINQ 呼び出しで MQIA_SYNCPOINT セレクターを使用します。

▶ z/OS TCPChannels (MQLONG)

TCP/IP 伝送プロトコルを使用して、現行にできるチャンネル、または接続できるクライアントの最大数。

値は 0 から 9999 までの範囲内でなければなりません。デフォルト値は 200 です。0 を指定した場合、TCP/IP は使用されません。

この属性は z/OS でのみサポートされます。

属性の値を判別するには、MQINQ 呼び出しで MQIA_TCP_CHANNELS セレクターを使用します。

▶ z/OS TCPKeepAlive (MQLONG)

接続の他の端が依然として使用可能であることを確認するために TCP KEEPALIVE を使用するかどうかを指定します。使用可能でない場合、チャンネルはクローズされます。

値は、次のいずれか 1 つです。

MQTCPKEEP_YES

TCP プロファイル構成データ・セットで指定されたとおりに TCP KEEPALIVE を使用します。チャンネル属性 KeepAliveInterval (KAINT) を指定する場合、これに設定された値が使用されます。

MQTCPKEEP_NO

TCP KEEPALIVE を使用しません。これがデフォルト値です。

この属性は z/OS でのみサポートされます。

属性の値を判別するには、MQINQ 呼び出しで MQIA_TCP_KEEP_ALIVE セレクターを使用します。

▶ z/OS TCPName (MQCHAR8)

TCPStackType の値に応じて、使用される唯一の、または優先される TCP/IP スタックの名前。このパラメーターは、CINET マルチ・スタック環境でのみ適用されます。デフォルト値は TCPIP です。

この属性は z/OS でのみサポートされます。

属性の値を判別するには、MQINQ 呼び出しで MQCA_TCP_NAME セレクターを使用します。属性の長さは、MQ_TCP_NAME_LENGTH で指定します。

TCPStackType (MQLONG)

チャンネル・イニシエーターが TCPName に指定された TCP/IP スタックのみを使用できるのか、それともオプションで任意に選択された TCP/IP スタックにバインドできるのかを指定します。このパラメーターは、CINET マルチ・スタック環境でのみ適用されます。

値は、次のいずれか 1 つです。

MQTCPSTACK_SINGLE

チャンネル・イニシエーターは、TCPName で指定された TCP/IP アドレス・スペースのみを使用できます。これがデフォルト値です。

MQTCPSTACK_MULTIPLE

チャンネル・イニシエーターは、使用可能な TCP/IP アドレス・スペースをすべて使用できます。チャンネルまたはリスナーで特に指定されていない場合、TCPName で指定されているものがデフォルトで使用されます。

この属性は z/OS でのみサポートされます。

属性の値を判別するには、MQINQ 呼び出しで MQIA_TCP_STACK_TYPE セレクターを使用します。

TraceRouteRecording (MQLONG)

トレース経路情報のレコードを制御します。

値は、次のいずれか 1 つです。

MQRECORDING_DISABLED

トレース経路メッセージへの追加は許可されていません。

MQRECORDING_Q

トレース経路メッセージを指定された固定キューに書き込みます。

MQRECORDING_MSG

トレース経路メッセージを、メッセージ自体を使用して決定されたキューに書き込みます。これはデフォルト値です。

属性の値を判別するには、MQINQ 呼び出しで MQIA_TRACE_ROUTE_RECORDING セレクターを使用します。

TriggerInterval (MQLONG)

これは、トリガー・メッセージの数を制限するために使用される時間間隔 (単位はミリ秒) です。これは、TriggerType が MQTT_FIRST の場合にのみ関連のある属性です。この場合、通常トリガー・メッセージが生成されるのは、適切なメッセージがキューに到着してそのときにキューが空であった場合に限り限られます。ただし、ある種の環境では、そのキューが空でなかった場合でも、MQTT_FIRST によるトリガー発行によって、追加のトリガー・メッセージを生成することができます。これらの追加のトリガー・メッセージは、TriggerInterval ミリ秒ごとよりも頻繁に生成されることはありません。

トリガー操作の詳細については、[チャンネルのトリガー操作](#)を参照してください。

値は、0 以上で、999 999 999 以下です。デフォルト値は 999 999 999 です。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_TRIGGER_INTERVAL セレクターを使用します。

TriggerInterval (MQLONG)

これは、トリガー・メッセージの数を制限するために使用される時間間隔 (単位はミリ秒) です。これは、TriggerType が MQTT_FIRST の場合にのみ関連のある属性です。この場合、通常トリガー・メッセージが生成されるのは、適切なメッセージがキューに到着してそのときにキューが空であった場合に限り限られます。ただし、ある種の環境では、そのキューが空でなかった場合でも、MQTT_FIRST によるトリガー発行によって、追加のトリガー・メッセージを生成することができます。これらの追加のトリガー・メッセージは、TriggerInterval ミリ秒ごとよりも頻繁に生成されることはありません。

トリガー操作の詳細については、[チャンネルのトリガー操作](#)を参照してください。

値は、0 以上で、999 999 999 以下です。デフォルト値は 999 999 999 です。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_TRIGGER_INTERVAL セレクターを使用します。

Version (MQCFST)

これは IBM MQ コードのバージョンで、VRRMMFF という形式で表されます。意味するところは以下のとおりです。

VV - バージョン

RR - リリース

MM - 保守レベル

FF - フィックス・レベル

XrCapability(MQLONG)

IBM MQ Telemetry コマンドがキュー・マネージャーでサポートされるかどうかを制御します。

値は、次のいずれか 1 つです。

MQCAP_SUPPORTED

IBM MQ Telemetry コンポーネントがインストール済みで、テレメトリー・コマンドがサポートされます。

MQCAP_NOT_SUPPORTED

IBM MQ Telemetry コンポーネントがインストールされていません。

この属性は、[マルチプラットフォーム](#)でのみサポートされます。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_XR_CAPABILITY セレクターを使用します。

キューの属性

キュー定義には、5 つのタイプがあります。キュー属性には、すべてのタイプのキューに適用される属性と、特定のタイプのキューにのみ適用される属性があります。

キューのタイプ

キュー・マネージャーは、以下のタイプのキュー定義をサポートします。

ローカル・キュー

ローカル・キューにメッセージを格納できます。

 **z/OS** z/OS では、このキューを共有キューまたは専用キューにできます。

キューは、プログラムが接続されているキュー・マネージャーに所有される場合、プログラムではローカルと認知されます。メッセージは、ローカル・キューから取得し、ローカル・キューに入れることができます。

キュー定義オブジェクトは、キューに入る物理メッセージと同様、そのキューの定義情報を保持します。

ローカル・キュー・マネージャーのキュー

このキューはローカル・キュー・マネージャー上に存在します。

 **z/OS** このキューは、z/OS では専用キューとして知られています。

 **z/OS** **共有キュー (z/OS のみ)**

このキューは、共有リポジトリ内に存在しており、その共有リポジトリを所有するキュー共有グループに属する全キュー・マネージャーからアクセス可能です。

キュー共有グループ内のキュー・マネージャーに接続しているアプリケーションは、このタイプのキューに対してメッセージを書き込んだり、メッセージを削除したりできます。このようなキューは、実質的にはローカル・キューと同じです。 **QType** キュー属性の値は **MQQ_ローカル** です。

ローカル・キュー・マネージャーに接続しているアプリケーションは、このタイプのキューに対してメッセージを書き込んだり、メッセージを削除したりできます。 **QType** キュー属性の値は **MQQ_ローカル** です。

クラスター・キュー

クラスター・キューが定義されているキュー・マネージャーでは、クラスター・キューにメッセージを格納できます。クラスター・キューとは、クラスター・キュー・マネージャーでホストされ、同じクラスター内の別のキュー・マネージャーで利用できるキューです。 **QType** キュー属性の値は **MQQT_CLUSTER** です。

クラスター・キュー定義は、クラスター内の他のキュー・マネージャーに通知されます。クラスター内にあるその他のキュー・マネージャーは、対応するリモート・キュー定義がなくても、クラスター・キューにメッセージを書き込むことができます。クラスター名前リストを使用して、クラスター・キューを複数のクラスターに通知できます。

キューが通知されると、クラスター内のキュー・マネージャーはそのキューにメッセージを書き込めるようになります。メッセージを書き込むときには、キュー・マネージャーが、フルリポジトリで、そのキューがホストされている場所を調べる必要があります。格納場所が分かったら、宛先情報をメッセージに追加して、クラスター伝送キューにメッセージを書き込みます。

キュー・マネージャーは、同じクラスター内の他のキュー・マネージャーのメッセージを複数の伝送キューに格納することができます。複数のクラスター伝送キューにメッセージを格納するようキュー・マネージャーを構成する方法は2つあります。キュー・マネージャー属性 **DEFCLXQ** を **CHANNEL** に設定すると、クラスター送信側チャンネルごとに異なるクラスター伝送キューが、**SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE** から自動的に作成されます。 **CLCHNAME** 伝送キュー・オプションを1つ以上のクラスター送信側チャンネルに一致するように設定すると、キュー・マネージャーは、一致しているチャンネルのメッセージを、そのチャンネルの伝送キューに格納できます。



重要: IBM WebSphere MQ 7.5 より前のバージョンの製品からアップグレードされたキュー・マネージャーで専用の **SYSTEM.CLUSTER.TRANSMIT.QUEUES** を使用する場合は、**SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE** の **SHARE/NOSHARE** オプションが **SHARE** に設定されている必要があります。

クラスター・キューは、IBM MQ for z/OS でのキュー共有グループのメンバーによって共用されるキューにすることができます。

リモート・キュー

リモート・キューは物理的なキューではありません。リモート・キュー・マネージャー上に存在するキューのローカル定義です。リモート・キューのローカル定義には、リモート・キュー・マネージャーにメッセージを経路指定する方法を、ローカル・キュー・マネージャーに示す情報が入っています。

ローカル・キュー・マネージャーに接続しているアプリケーションは、このタイプのキューにメッセージを書き込むことができます。そのメッセージは、リモート・キュー・マネージャーにメッセージを経路指定するために使用されるローカル伝送キューに入れられます。アプリケーションは、リモート・キューからメッセージを除去することはできません。 **QType** キュー属性の値は **MQQT_REMOTE** です。

リモート・キューの定義は、次の用途にも使用できます。

- 応答キューへの別名割り当て

この場合、定義の名前は、応答先のキューの名前です。詳細については、[応答先キューの別名およびクラスター](#)を参照してください。

- キュー・マネージャーの別名割り当て

この場合、定義の名前は、キューの名前ではなく キュー・マネージャーの別名です。詳細については、[キュー・マネージャーの別名およびクラスター](#)を参照してください。

別名キュー

これは物理的なキューではありません。ローカル・キュー、共有キュー、クラスター・キュー、またはリモート・キューの代替名です。別名から解決されるキューの名前は、別名キューの定義内に含まれています。

ローカル・キュー・マネージャーに接続しているアプリケーションは、このタイプのキューにメッセージを書き込むことができます。そのメッセージは、別名から解決されたキューに入れられます。アプリケーションがこのタイプのキューからメッセージを削除できるのは、別名が、ローカル・キュー、共有キュー、またはローカル・インスタンスを持つクラスター・キューに解決された場合です。QType キュー属性の値はMQQT_ALIASです。

モデル・キュー

これは、物理的なキューではありません。キュー属性の1セットであり、これを基にしてローカル・キューを作成することができます。

このタイプのキューにはメッセージを保管できません。

キューの制限

以前のリリースで使用されていたデフォルトの2テラバイト制限を大幅にIBM MQ上回る容量をサポートするキューの設定と監視が可能です。また、キュー・ファイルを拡張できるサイズの上限を小さくすることもできます。

キューを構成できるようにするには、ローカル・キューおよびモデル・キューでMAXFSIZE属性を使用し、キューをモニターするには、キュー状況属性CURFSIZEおよびCURMAXFSを使用します。

詳しくは、[IBM MQ キュー・ファイルの変更](#)を参照してください。

キューの属性

キュー属性には、すべてのタイプのキューに適用される属性と、特定のタイプのキューにのみ適用される属性があります。属性が適用されるキューのタイプを、[832 ページの表 597](#) およびそれに続く表に示します。

[832 ページの表 597](#) には、キューに固有の属性がまとめられています。属性の説明は、アルファベット順に掲載しています。

注：このセクションに記載している属性の名前は、MQINQ および MQSET 呼び出しで使用する記述名です。これらの名前は、PCF コマンド用のものと同じです。MQSC コマンドを使用して属性を定義、変更、または表示するときには、代替の短縮名が使用されます。詳細については、[MQSC コマンド](#)を参照してください。

以下の表では、列は次のように適用されます。

- ローカル・キューの列は、共有キューにも適用されます。
- モデル・キューの列は、そのモデル・キューを基にして作成されたローカル・キューに継承される属性を示しています。
- クラスター・キューの列は、照会のみ、または照会と出力を目的としてクラスター・キューがオープンされた場合に照会できる属性を示しています。その他の属性を照会すると、呼び出しは完了コードMQCC_WARNING および理由コードMQRC_SELECTOR_NOT_FOR_TYPE (2068) を返します。

照会に加えて、入力、ブラウズ、または設定のうち1つ以上の作業を目的としてクラスター・キューがオープンされた場合は、代わりにローカル・キューの列が適用されます。

クラスター・キューが照会だけでオープンされている場合、または照会と出力のために基本キュー・マネージャー名を指定した場合は、ローカル・キューの列が代わりに適用されます。

表 597. キューの属性						
属性	説明	ローカル	モデル	別名	リモート	クラスター
AlterationDate	定義が最後に変更された日付	X		X	X	
AlterationTime	定義が最後に変更された時刻	X		X	X	

表 597. キューの属性 (続き)						
属性	説明	ローカル	モデル	別名	リモート	クラスター
BackoutRequeueQName	超過バックアウト再キューイング用のキュー名	X	X			
BackoutThreshold	バックアウトしきい値	X	X			
BaseQName	別名が解決されるキュー名			X		
CFStrucName	カップリング・ファシリティ構造体の名前	X	X			
CLCHNAME	クラスター送信側チャネル名	✓	✓			
ClusterName	キューが属するクラスターの名前	X		X	X	X
ClusterNameList	キューが属するクラスターの名前を含む名前リスト・オブジェクトの名前	X		X	X	
CLWLQueuePriority	クラスター・ワークロード・キューの優先順位	X		X	X	X
CLWLQueueRank	クラスター・ワークロード・キューのランク	X		X	X	X
CLWLUseQ	リモート・キューの使用	X				
CreationDate	キューが作成された日付	X				
CreationTime	キューが作成された時刻	X				
CurrentQDepth	現行キュー項目数	X				
DefaultPutResponse	デフォルトの書き込み応答	✓	✓	✓	✓	
DefBind	デフォルトのバインド	X		X	X	X
DefinitionType attribute	キュー定義タイプ	X	X			
DefInputOpenOption	デフォルト入力オープン・オプション	X	X			
DefPersistence	デフォルトのメッセージ持続性	X	X	X	X	X
DefPriority	デフォルトのメッセージ優先順位	✓	✓	✓	✓	✓
DefReadAhead	デフォルト先読み	X	X	X		
DistLists	配布リスト・サポート	X	X			
HardenGetBackout	正確なバックアウト・カウントを維持するかどうか	X	X			
IndexType	索引タイプ	X	X			
InhibitGet	キューに対する取得 (GET) 操作を許可するかどうか	X	X	X		
InhibitPut	キューに対する書き込み (PUT) 操作を許可するかどうか	X	X	X	X	X
InitiationQName	開始キューの名前	X	X			
MaxMsgLength	メッセージの最大長 (バイト数)。	X	X			
MaxQDepth	キューの最大長	X	X			
MsgDeliverySequence attribute	メッセージ・デリバリー・シーケンス	X	X			
NonPersistentMessage Class	非持続メッセージの信頼性に関する目標	X	X			
OpenInputCount	入力のためのオープンの回数	X				
OpenOutputCount	出力のためのオープンの回数	X				
PropertyControl	プロパティ制御	✓	✓	✓		

表 597. キューの属性 (続き)						
属性	説明	ローカル	モデル	別名	リモート	クラスター
ProcessName	プロセス名	X	X			
QDepthHighEvent attribute	キュー項目数高イベントが生成されるかどうかの指定	X	X			
QDepthHighLimit	キュー・サイズの上限	X	X			
QDepthLowEvent attribute	キュー項目数低イベントが生成されるかどうかの指定	X	X			
QDepthLowLimit attribute	キュー・サイズの下限	X	X			
QDepthMaxEvent	キュー満杯イベントが生成されるかどうか	X	X			
QDesc	キューの記述	X	X	X	X	X
QName	キュー名	X		X	X	X
QServiceInterval	キュー・サービス間隔の目標値	X	X			
QServiceIntervalEvent attribute	サービス間隔上限イベントまたはサービス間隔 OK イベントが生成されるかどうか	X	X			
QSGDisp attribute	キュー共有グループ後処理	X		X	X	
QueueAccounting	キュー・アカウンティング・データの収集	X	X	X	X	X
QueueMonitoring	キューのオンライン・モニター・データ。	X	✓			
QueueStatistics	キュー統計データの収集	X	X	X	X	X
QType	キュー・タイプ	X		X	X	X
RemoteQMgrName	リモート・キュー・マネージャーの名前				X	
RemoteQName	リモート・キューの名前				X	
RetentionInterval	保存間隔	X	X			
Scope	キューに関する項目がセル・ディレクトリーにも存在するかどうか	X		X	X	
Shareability	キューの共有可能性	X	X			
StorageClass	キューのストレージ・クラス	X	X			
TriggerControl	トリガー制御	X	X			
TriggerData	トリガー・データ	X	X			
TriggerDepth	トリガー項目数	X	X			
TriggerMsgPriority	トリガーのしきい値メッセージ優先順位	X	X			
TriggerType	トリガー・タイプ	X	X			
Usage attribute	キューの使用法	X	X			
XmitQName	伝送キュー名				X	

関連概念

[クラスター・キュー](#)

[ローカル・キュー](#)

[使用するクラスター伝送キュー・タイプの選択方法](#)

AlterationDate (MQCHAR12)

定義が最後に変更された日付。

表 598. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X		X	X	

これは、定義を最後に変更した日付です。この日付の形式は YYYY-MM-DD であり、長さを 12 バイトにするために、末尾にブランクを 2 つ埋め込みます (例えば 1992-09-23 のようになり、 は 2 つのブランク文字を示しています)。

特定の属性の値 (例えば、*CurrentQDepth*) はキュー・マネージャーが作動すると変更されます。これらの属性を変更しても、*AlterationDate* には影響しません。

この属性の値を判別するには、MQINQ 呼び出しで MQCA_ALTERATION_DATE セレクターを使用します。この属性の長さは MQ_DATE_LENGTH によって指定されます。

AlterationTime (MQCHAR8)

定義が最後に変更された時刻。

表 599. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X		X	X	

これは、定義を最後に変更した時刻です。時刻の形式は HH.MM.SS です。24 時間時計を使用し、10 時未満の場合は先頭にゼロを付けます (例えば、09.10.20)。

- z/OS では、時刻はグリニッジ標準時 (GMT) で、GMT に合わせて正確に設定されたシステム・クロックに従って計時されます。
- その他の環境では、時間はローカル時間です。

特定の属性の値 (例えば、*CurrentQDepth*) はキュー・マネージャーが作動すると変更されます。これらの属性を変更しても、*AlterationTime* には影響しません。

この属性の値を判別するには、MQINQ 呼び出しで MQCA_ALTERATION_TIME セレクターを使用します。この属性の長さは MQ_TIME_LENGTH によって指定されます。

BackoutRequeueQName (MQCHAR48)

これは過剰バックアウト・リキューのキュー名です。キュー・マネージャーは、属性値を照会できますが、この属性の値に基づくアクションをとることはありません。

表 600. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

WebSphere Application Server 内部で実行しているアプリケーション、および IBM MQ Application Server Facilities を使用するアプリケーションは、この属性を使用して、バックアウト済みメッセージの宛先を判別します。その他のすべてのアプリケーションでは、キュー・マネージャーがこの属性の値に基づいて動作を行うことはありません。

IBM MQ classes for JMS は、この属性を使用して、バックアウトされた回数が *BackoutThreshold* 属性に指定された最大回数に達したメッセージの転送先を判別します。

この属性の値を判別するには、MQINQ 呼び出しで MQCA_BACKOUT_REQ_Q_NAME セレクターを使用します。この属性の長さは MQ_Q_NAME_LENGTH によって指定されます。

BackoutThreshold (MQLONG)

これはバックアウトしきい値です。キュー・マネージャーは、属性値を照会できますが、この属性の値に基づくアクションをとることはありません。

表 601. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

WebSphere Application Server 内部で実行しているアプリケーション、および IBM MQ Application Server Facilities を使用するアプリケーションは、この属性を使用して、メッセージをバックアウトする必要があるかどうかを判別します。その他のすべてのアプリケーションでは、キュー・マネージャーがこの属性の値に基づいて動作を行うことはありません。

IBM MQ classes for JMS はこの属性を使用して、*BackoutRequeueQName* 属性で指定されたキューにメッセージを転送する前にメッセージをバックアウトできる回数を決定します。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_BACKOUT_THRESHOLD セレクターを使用します。

BaseQName (MQCHAR48)

これは、ローカル・キュー・マネージャーに対して定義されているキューの名前です。

表 602. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
		X		

(キュー名の詳細については、『MQOD - ObjectName フィールド』を参照してください。) キューは次のいずれかです。

MQQT_LOCAL

ローカル・キュー。

MQQT_REMOTE

リモート・キューのローカル定義。

MQQT_CLUSTER

クラスター・キュー。

この属性の値を判別するには、MQINQ 呼び出しで MQCA_BASE_Q_NAME セレクターを使用します。この属性の長さは MQ_Q_NAME_LENGTH によって指定されます。

BaseType (MQCFIN)

別名の解決先のオブジェクトのタイプ。

表 603. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
		X		

これは、次の値のいずれかです。

MQQT_Q

基本オブジェクト・タイプはキューです。

MQQT_TOPIC

基本オブジェクト・タイプはトピックです。

CFStrucName (MQCHAR12)

これは、キュー上のメッセージが保管されるカップリング・ファシリティ構造体の名前です。名前の最初の文字は A から Z までの範囲にあり、残りの文字は A から Z, 0 から 9, または空白の範囲にあります。

表 604. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

カップリング・ファシリティ構造体のフルネームを取得するには、**QSGName** キュー・マネージャー属性に、**CFStrucName** キュー属性の値を接尾語として付けます。

この属性は共有キューにのみ適用されます。これは、**QSGDisp** に値 **MQQSGD_SHARED** がない場合は無視されます。

この属性の値を判別するには、MQINQ 呼び出しで MQCA_CF_STRUC_NAME セレクターを使用します。属性の長さは、MQ_CF_STRUC_NAME_LENGTH で指定します。

 この属性は、z/OS でのみサポートされます。

ClusterChannelName (MQCHAR20)

ClusterChannelName このキューを送信キューとして使用するクラスター送信チャンネルの総称です。この属性は、このクラスター伝送キューからクラスター受信側チャンネルにメッセージを送信するクラスター送信側チャンネルを指定します。

表 605. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

デフォルトのキュー・マネージャー構成では、すべてのクラスター送信側チャンネルがメッセージを単一の伝送キュー **SYSTEM.CLUSTER.TRANSMIT.QUEUE** から送信します。デフォルト構成は、キュー・マネージャー属性 **DefClusterXmitQueueType** を変更することによって変更できます。属性のデフォルト値は **SCTQ** です。この値は **CHANNEL** に変更できます。**DefClusterXmitQueueType** 属性を **CHANNEL** に設定すると、各クラスター送信側チャンネルは、デフォルトで特定のクラスター伝送キュー **SYSTEM.CLUSTER.TRANSMIT.ChannelName** を使用するようになります。

また、伝送キュー属性である **ClusterChannelName** 属性をクラスター送信側チャンネルに手動で設定することもできます。クラスター送信側チャンネルによって接続されたキュー・マネージャーを宛先とするメッセージは、クラスター送信側チャンネルを識別する伝送キューに保管されます。これらのメッセージがデフォルトのクラスター伝送キューに保管されることはありません。**ClusterChannelName** 属性をブランクに設定すると、チャンネルの再始動時に、チャンネルはデフォルトのクラスター伝送キューに切り替わります。デフォルト・キューは、キュー・マネージャーの **DefClusterXmitQueueType** 属性の値に応じて、**SYSTEM.CLUSTER.TRANSMIT.ChannelName** または **SYSTEM.CLUSTER.TRANSMIT.QUEUE** のどちらかになります。

アスタリスク "*" を **ClusterChannelName** に指定することにより、伝送キューをクラスター送信側チャンネルのセットに関連付けることができます。アスタリスクはチャンネル名ストリングの先頭、末尾、またはそれ以外の場所に任意の数だけ使用できます。**ClusterChannelName** は長さ 20 文字に制限されています: **MQ_CHANNEL_NAME_LENGTH**。

ClusterName (MQCHAR48)

これは、キューが属するクラスターの名前です。

表 606. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X		X	X	X

キューが複数のクラスターに属している場合、*ClusterNameList* はクラスターを識別する名前リスト・オブジェクトの名前を指定し、*ClusterName* はブランクになります。*ClusterName* および *ClusterNameList* の少なくとも 1 つがブランクでなければなりません。

この属性の値を判別するには、MQINQ 呼び出しで MQCA_CLUSTER_NAME セレクターを使用します。属性の長さは、MQ_CLUSTER_NAME_LENGTH で指定します。

ClusterNameList (MQCHAR48)

これは、このキューが属しているクラスターの名前を含む名前リスト・オブジェクトの名前です。

表 607. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X		X	X	

キューが 1 つのクラスターのみに属する場合、名前リスト・オブジェクトには 1 つの名前のみ含まれます。代わりに *ClusterName* を使用して、そのクラスターの名前を指定できます。この場合、*ClusterNameList* はブランクにします。*ClusterName* および *ClusterNameList* の少なくとも 1 つがブランクでなければなりません。

この属性の値を判別するには、MQINQ 呼び出しで MQCA_CLUSTER_NAMELIST セレクターを使用します。属性の長さは、MQ_NAMELIST_NAME_LENGTH で指定します。

CLWLQueuePriority (MQLONG)

これは、クラスター・ワークロード・キュー優先順位です。0 から 9 までの範囲の値によってキューの優先順位を表します。

表 608. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X		X	X	X

詳しくは、[クラスター・キュー](#)を参照してください。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_CLWL_Q_PRIORITY セレクターを使用します。

CLWLQueueRank (MQLONG)

これは、クラスター・ワークロード・キューのランクです。0 から 9 までの範囲の値によってキューのランクを表します。

表 609. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X		X	X	X

詳しくは、[クラスター・キュー](#)を参照してください。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_CLWL_Q_RANK セレクターを使用します。

CLWLUseQ (MQLONG)

これは、ターゲット・キューにローカル・インスタンスと少なくとも 1 つのリモート・クラスター・インスタンスの両方が含まれる場合の MQPUT の振る舞いを定義します。書き込みがクラスター・チャネルから発信されている場合、この属性は適用されません。

表 610. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X				

値は、次のいずれか 1 つです。

MQCLWL_USEQ_ANY

リモート・キューとローカル・キューを使用します。

MQCLWL_USEQ_LOCAL

リモート・キューを使用しません。

MQCLWL_USEQ_AS_Q_MGR

キュー・マネージャーの MQIA_CLWL_USEQ から定義を継承します。

詳しくは、[クラスター・キュー](#)を参照してください。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_CLWL_USEQ セレクターを使用します。属性の長さは、MQ_CLWL_USEQ_LENGTH で指定します。

CreationDate (MQCHAR12)

これは、キューが作成された日付です。

表 611. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X				

この日付の形式は YYYY-MM-DD であり、長さを 12 バイトにするために、末尾に空白を 2 つ埋め込みます (例えば 2013-09-23 のようになり、 は 2 つの空白文字を示しています)。

- IBM i では、キューの作成日付は、オペレーティング・システムの中でキューを表すエンティティ (ファイルまたはユーザー・スペース) の日付と異なる場合があります。

この属性の値を判別するには、MQINQ 呼び出しで MQCA_CREATION_DATE セレクターを使用します。属性の長さは、MQ_CREATION_DATE_LENGTH で指定します。

CreationTime (MQCHAR8)

これは、キューが作成された時刻です。

表 612. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X				

時刻の形式は HH.MM.SS です。24 時間時計を使用し、10 時未満の場合は先頭にゼロを付けます (例えば、09.10.20)。

- z/OS では、時刻はグリニッジ標準時 (GMT) で、GMT に合わせて正確に設定されたシステム・クロックに従って計時されます。
- その他の環境では、時間はローカル時間です。
- IBM i では、キューの作成時刻が、そのキューを表す基礎のオペレーティング・システム・エンティティ (ファイルまたはユーザー・スペース) の作成時刻と異なる場合があります。

この属性の値を判別するには、MQINQ 呼び出しで MQCA_CREATION_TIME セレクターを使用します。属性の長さは、MQ_CREATION_TIME_LENGTH で指定します。

CurrentQDepth (MQLONG)

これは、現在キューにあるメッセージの数です。

表 613. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X				

この値は、MQPUT 呼び出しが行われたとき、および MQGET 呼び出しのバックアウトが行われたときに増加します。また、ブラウズのない MQGET 呼び出しが行われたとき、および MQPUT 呼び出しのバックアウトが行われたときに減少します。このカウントには、1 つの作業単位でそのキューに書き込まれたメッセージのうち、MQGET 呼び出しによる取り出しの対象でないものも含めて、まだコミットされていないメッセージの数が含まれることになります。同様に、1 つの作業単位で MQGET 呼び出しによって取り出されたメッセージは、まだコミットされる必要のあるものも、このカウントから除かれることになります。

また、このカウントには、有効期限を過ぎてもまだ廃棄されていない、取り出し対象以外のメッセージも含まれています。詳細については、『MQMD - Expiry フィールド』を参照してください。

作業単位処理とメッセージの分割は、いずれもがを超える *CurrentQDepth* 原因 *MaxQDepth* となり得る。ただし、このことによってメッセージの検索に影響があるわけではありません。キューにあるすべてのメッセージは、MQGET 呼び出しを使用して通常の方法で検索できます。

属性の値は、キュー・マネージャーの動作に応じて変動します。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_CURRENT_Q_DEPTH セレクターを使用します。

DefaultPutResponse (MQLONG)

アプリケーションで MQPMO_RESPONSE_AS_Q_DEF が指定される際に、キューに対する PUT 操作で使用する応答のタイプを指定します。

表 614. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X	X	X	

これは、次の値のいずれかです。

MQPRT_SYNC_RESPONSE

PUT 操作は同期的に実行され、応答が返されます。

MQPRT_ASYNC_RESPONSE

PUT 操作は非同期的に実行され、MQMD フィールドのサブセットが返されます。

DefBind (MQLONG)

これは、MQOPEN 呼び出しで MQOO_BIND_AS_Q_DEF が指定されていて、キューがクラスター・キューである場合に使用されるデフォルトのバインディングです。

表 615. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X		X	X	X

値は、次のいずれか 1 つです。

MQBND_BIND_ON_OPEN

MQOPEN 呼び出しで固定されたバインディング。

MQBND_BIND_NOT_FIXED

固定されていないバインディング。

MQBND_BIND_ON_GROUP

グループ内のメッセージすべてを同じ宛先のインスタンスに割り振る要求をアプリケーションが行えるようになります。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_DEF_BIND セレクターを使用します。

DefinitionType (MQLONG)

これは、キューの定義方法を示します。

表 616. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

値は、次のいずれか1つです。

MQQDT_PREDEFINED

システム管理者に作成される永続キューであり、システム管理者だけがそのキューを削除できます。

事前定義されたキューが DEFINE MQSC コマンドを使用して作成された場合、このキューを削除するには、DELETE MQSC コマンドしか使用できません。事前定義されたキューは、モデル・キューからは作成できません。

コマンドは、オペレーターが発行する場合と、コマンド入力キューにコマンド・メッセージを送信する許可ユーザーが発行する場合があります (『[CommandInputQName 属性](#)』を参照)。

MQQDT_PERMANENT_DYNAMIC

オブジェクト記述子 MQOD 内に指定されているモデル・キューの名前を指定して MQOPEN 呼び出しを発行しているアプリケーションによって作成された永続キューです。モデル・キュー定義には、**DefinitionType** 属性の値として MQQDT_PERSISTENT_DYNAMIC が指定されていました。

このタイプのキューは、MQCLOSE 呼び出しを使用して削除できます。詳しくは、[642 ページの『MQCLOSE - オブジェクトのクローズ』](#)を参照してください。

永続動的キューの **QSGDsp** 属性の値は、MQQSGD_Q_MGR です。

MQQDT_TEMPORARY_DYNAMIC

オブジェクト記述子 MQOD 内に指定されているモデル・キューの名前を指定して MQOPEN 呼び出しを発行しているアプリケーションによって作成された一時キューです。モデル・キュー定義には、**DefinitionType** 属性の値 MQQDT_TEMPORARY_DYNAMIC が含まれていました。

このタイプのキューは、これを作成したアプリケーションによってクローズされるときに、MQCLOSE 呼び出しで自動的に削除されます。

一時動的キューの **QSGDsp** 属性の値は、MQQSGD_Q_MGR です。

MQQDT_SHARED_DYNAMIC

このキューは、オブジェクト記述子 MQOD で指定されているモデル・キューの名前を指定して MQOPEN 呼び出しを発行したアプリケーションによって作成された共有永続キューです。モデル・キュー定義には、**DefinitionType** 属性の値 MQQDT_SHARED_DYNAMIC があります。

このタイプのキューは、MQCLOSE 呼び出しを使用して削除できます。詳しくは、[642 ページの『MQCLOSE - オブジェクトのクローズ』](#)を参照してください。

共用動的キューの **QSGDsp** 属性の値は、MQQSGD_SHARED です。

モデル・キューは常に事前定義されているため、モデル・キュー定義内のこの属性には、モデル・キューの定義方法は示されていません。代わりに、モデル・キュー内のこの属性の値を使用して、MQOPEN 呼び出しを使用してモデル・キュー定義から作成された各動的キューの *DefinitionType* が決定されます。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_DEFINITION_TYPE セレクターを使用します。

DefInputOpenOption (MQLONG)

これは、入力用にキューをオープンするデフォルトの方法です。

表 617. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

キューがオープンされるときに MQOPEN 呼び出しに MQOO_INPUT_AS_Q_DEF オプションが指定されている場合に適用されます。値は、次のいずれか 1 つです。

MQOO_INPUT_EXCLUSIVE

メッセージを読み取るためにキューを排他アクセス・モードでオープンする。

後続の MQGET 呼び出しで使用するために、キューが開かれます。このアプリケーションまたは別のアプリケーションによって、キューがいずれかのタイプ (MQOO_INPUT_SHARED または MQOO_INPUT_EXCLUSIVE) の入力用に現在開かれている場合、この呼び出しは失敗し、理由コード MQRC_OBJECT_IN_USE が戻ります。

MQOO_INPUT_SHARED

共有アクセスによりメッセージを読み取るためにキューをオープンする。

後続の MQGET 呼び出しで使用するために、キューが開かれます。このアプリケーションまたは別のアプリケーションによって、キューが MQOO_INPUT_SHARED で現在オープンされている場合は、この呼び出しは正常に行われますが、キューが MQOO_INPUT_EXCLUSIVE でオープンされている場合は失敗し、理由コード MQRC_OBJECT_IN_USE が戻ります。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_DEF_INPUT_OPEN_OPTION セレクターを使用します。

DefPersistence (MQLONG)

これは、キューのメッセージのデフォルト持続性です。これは、メッセージの書き込み時にメッセージ記述子に MQPER_PERSISTENCE_AS_Q_DEF が指定されている場合に適用されます。

表 618. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X	X	X	X

キュー名解決パス内に定義が 2 つ以上ある場合、デフォルトの持続性は、MQPUT または MQPUT1 呼び出しが行われる時に、パス内にある複数の定義のうちの最初の定義 (これがキュー・マネージャーの別名であっても) 内の属性値からとられます。これには以下のものが考えられます。

- 別名キュー
- ローカル・キュー
- リモート・キューのローカル定義
- キュー・マネージャーの別名
- 伝送キュー (例えば、DefXmitQName キューなど)

値は、次のいずれか 1 つです。

MQPER_PERSISTENT

メッセージはシステムの障害後、およびキュー・マネージャーの再始動後も存続します。持続メッセージを以下に書き込むことはできません。

- 一時動的キュー
- CFLEVEL(2) 以下で CFSTRUCT オブジェクトにマップする共有キュー、または CFSTRUCT オブジェクトが RECOVER(NO) として定義されている共有キュー。

持続メッセージは、永続動的キュー、および事前定義のキューに書き込むことができます。

MQPER_NOT_PERSISTENT

メッセージは通常は、システムの障害後またはキュー・マネージャーの再始動後は存続しません。これは、キュー・マネージャーの再始動中にメッセージの完全なコピーが補助ストレージで見つかった場合でも適用されます。

共有キューの場合、非持続性メッセージは、キュー共有グループのキュー・マネージャーの再始動後も存続しますが、メッセージを共有キューに保管するために使用するカップリング・ファシリティーの失敗後は存続しません。

持続メッセージと非持続メッセージが同一キューにあっても構いません。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_DEF_PERSISTENCE セレクターを使用します。

DefPriority (MQLONG)

これは、キューのメッセージのデフォルト優先順位です。これは、メッセージがキューに書き込まれたときにメッセージ記述子に MQPRI_PRIORITY_AS_Q_DEF が指定されている場合に適用されます。

表 619. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X	X	X	X

キュー名解決パス内に定義が 2 つ以上ある場合、そのメッセージに関するデフォルト優先順位は、PUT 操作時にパス内にある複数の定義のうちの最初の定義内の属性値からとられます。これには以下のものが考えられます。

- 別名キュー
- ローカル・キュー
- リモート・キューのローカル定義
- キュー・マネージャーの別名
- 伝送キュー (例えば、DefXmitQName キューなど)

メッセージがキューに配置される方法は、キューの **MsgDeliverySequence** 属性の値によって異なります。

- **MsgDeliverySequence** 属性が MQMDS_PRIORITY の場合は、メッセージ記述子内の *Priority* フィールドの値によって、キュー内でのメッセージの論理的な位置が決まります。
- **MsgDeliverySequence** 属性が MQMDS_FIFO の場合は、各メッセージは解決されたキューの *DefPriority* と等しい優先度を前提としてキューに入ります。この場合、メッセージ記述子内の *Priority* フィールドの値は関係しません。ただし、*Priority* フィールドは、メッセージを書き込んだアプリケーションによって指定された値を保持します。詳細については、『[MsgDeliverySequence 属性](#)』を参照してください。

優先順位は、ゼロ (最下位) から *MaxPriority* (最上位) の範囲にあります。『[MaxPriority 属性](#)』を参照してください。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_DEF_PRIORITY セレクターを使用します。

DefReadAhead (MQLONG)

クライアントに配信される非持続メッセージのデフォルトの先読み動作を指定します。

表 620. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X	X		

DefReadAhead は、以下のいずれかの値に設定できます。

MQREADA_NO

アプリケーションが要求する前に、非持続メッセージがクライアントに先送りされません。クライアントが異常終了した場合に失われる非持続メッセージは、最大で 1 つだけです。

MQREADA_YES

非持続メッセージは、アプリケーションで要求される前にクライアントに送信されます。クライアントが異常終了した場合、またはクライアントに送信されたすべてのメッセージをクライアントが消費しない場合に、非持続メッセージは失われることがあります。

MQREADA_DISABLED

このキューに対して、非持続メッセージの先読みは有効になりません。クライアント・アプリケーションによって先読みが要求されているかどうかに関わりなく、メッセージはクライアントに前もって送信されません。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_DEF_READ_AHEAD セレクターを使用します。

DefPResp (MQLONG)

デフォルトの書き込み応答タイプ (DEFPRESP) 属性は、MQPMO 中の PutResponseType が MQPMO_RESPONSE_AS_Q_DEF に設定されている場合にアプリケーションで使用される値を定義します。この属性は、すべてのキュー・タイプに有効です。

表 621. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X	X	X	X

値は、次のいずれか 1 つです。

SYNC

PUT 操作は同期的に実行され、応答が返されます。

ASYN

PUT 操作は非同期的に実行され、MQMD フィールドのサブセットが返されます。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_DEF_PUT_RESPONSE_TYPE セレクターを使用します。

DistLists (MQLONG)

これは、配布リスト・メッセージをキューに入れることができるかどうかを示します。

表 622. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

メッセージ・チャンネル・エージェント (MCA) はこの属性を設定し、ローカル・キュー・マネージャーに、チャンネルのもう一方のキュー・マネージャーが配布リストをサポートしているかどうかを通知します。後者のキュー・マネージャー (パートナー・キュー・マネージャーという) は、送信 MCA によってローカル伝送キューから削除されると、次にメッセージを受信するキュー・マネージャーです。

送信 MCA は、パートナー・キュー・マネージャーで受信 MCA との接続を確立するたびにこの属性を設定します。この方法では、ローカル・キュー・マネージャーは送信 MCA によって、パートナー・キュー・マネージャーが正しく処理できるメッセージのみを伝送キューに置くようになります。

この属性の本来の使用方法では、伝送キューと共に使用しますが、説明にある処理は伝送キューに定義された使用方法とは関係なく行われます (『[Usage 属性](#)』を参照してください)。

値は、次のいずれか 1 つです。

MQDL_SUPPORTED

配布リストのメッセージをキューに保管することができ、その形式でパートナー・キュー・マネージャーに転送できます。これにより、複数の宛先に送信するときの処理量が少なくなります。

MQDL_NOT_SUPPORTED

配布リストのメッセージがキューに保管することができません。パートナー・キュー・マネージャーで配布リストがサポートしていないためです。アプリケーションによって書き込まれた配布リストのメッセージがこのキューに置かれると、キュー・マネージャーは配布リストのメッセージを分割し、個々のメッセージをキューに書き込みます。これにより、複数の宛先にメッセージを送信するときの処理量は増えますが、メッセージはパートナー・キュー・マネージャーによって正しく処理されるようになります。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_DIST_LISTS セレクターを使用します。この属性の値を変更するには、MQSET 呼び出しを使用します。

z/OS では、この属性はサポートされていません。

HardenGetBackout (MQLONG)

メッセージごとに、1つのカウントは、メッセージが1つの作業単位内の MQGET 呼び出しによって取り出される回数、およびそのあとでバックアウトされた作業単位の数によって保持されます。

表 623. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

このカウントは、MQGET 呼び出しの完了後に、メッセージ記述子内の *BackoutCount* フィールドに表示されます。

メッセージのバックアウト・カウントは、キュー・マネージャーが再始動したあとまで残ります。ただし、カウントが正確であるようにするには、MQGET 呼び出しがこのキューの作業単位内でメッセージを取り出すたびに、情報をハード化 (ディスクまたはその他の永続記憶装置に記録) する必要があります。これを行わないと、キュー・マネージャーの障害が発生し、MQGET 呼び出しがバックアウトされ、カウントが増加しないこともあります。

ただし、1つの作業単位の各 MQGET 呼び出しに関する情報のハード化には、追加処理コストが必要になるため、**HardenGetBackout** 属性を MQQA_BACKOUT_HARDENED に設定するのは、正確なカウントが不可欠な場合に限りです。

マルチプラットフォームでは、この属性の設定値とは無関係に、メッセージのバックアウト・カウントが常にハード化されます。

属性の値は以下のとおりです。

MQQA_BACKOUT_HARDENED

このキューのメッセージのバックアウト・カウントを正確にするために、ハード化が行われます。

MQQA_BACKOUT_NOT_HARDENED

このキューのメッセージのバックアウト・カウントを正確にするためのハード化は行われません。したがって、カウントが正しい値よりも小さくなる可能性があります。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_HARDEN_GET_BACKOUT セレクターを使用します。

IndexType (MQLONG)

これは、キュー・マネージャーがキュー上のメッセージに対して保守するインデックスのタイプを指定します。

表 624. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

必要とされる索引のタイプは、アプリケーションがメッセージを検索する方法、そしてキューが共有キューであるか、非共有キューであるか (『QSGDisp 属性』を参照) によって異なります。IndexType の値には次のものがあります。

MQIT_NONE

このキューについては、キュー・マネージャーが保守する索引はありません。この値は、通常は順番に処理されるキュー、つまり MQGET 呼び出しの際に選択基準を何も使用しないキューに使用します。

MQIT_MSG_ID

キュー・マネージャーは索引を保守します。索引には、キューにあるメッセージのメッセージ ID を使用しています。アプリケーションがメッセージを検索する場合に通常、MQGET 呼び出しの際の選択基準としてメッセージ ID を使用するキューでは、この値を使用します。

MQIT_CORREL_ID

キュー・マネージャーは索引を保守します。索引には、キューにあるメッセージの相関 ID を使用しています。アプリケーションがメッセージを検索する場合に通常、MQGET 呼び出しの際の選択基準として相関 ID を使用するキューでは、この値を使用します。

MQIT_MSG_TOKEN

重要：この索引タイプは、IBM MQ Workflow for z/OS 製品で使用されるキューに対してのみ、使用する必要があります。

キュー・マネージャーは索引を保守します。索引には、z/OS のワークロード・マネージャー (WLM) 機能で使用するキューにあるメッセージのメッセージ・トークンを使用しています。

WLM 管理キューでは、このオプションを指定する必要があります。他のタイプのキューにはこのオプションを指定しないでください。さらに、z/OS ワークロード・マネージャー機能を使用しないが、MQGET 呼び出しの際の選択基準としてメッセージ・トークンを使用して、アプリケーションがメッセージを検索するキューについては、この値を使用しないでください。

MQIT_GROUP_ID

キュー・マネージャーは索引を保守します。索引には、キューにあるメッセージのグループ ID が使用されます。アプリケーションが MQGET 呼び出しで MQGMO_LOGICAL_ORDER オプションを使用してメッセージを検索するキューには、必ずこの値を使用します。

この索引タイプのキューは、伝送キューにすることはできません。この索引タイプを使用する共有キューは、CFLEVEL(3) 以上の CFSTRUCT オブジェクトにマップするように定義する必要があります。

注：

1. 索引タイプ MQIT_GROUP_ID のキューでは、メッセージの物理順序は定義されません。キューは、MQGET 呼び出しで MQGMO_LOGICAL_ORDER オプションを使用して、最も効率的にメッセージを検索できるように最適化されるからです。このため、通常、メッセージの物理順序はメッセージがキューに入れられた順序とは異なります。
2. MQIT_GROUP_ID キューに MQMDS_PRIORITY の *MsgDeliverySequence* がある場合、キュー・マネージャーは、メッセージ優先順位 0 および 1 を使用して、メッセージを検索する順序を論理順序に最適化します。結果として、グループ内の最初のメッセージには、0 や 1 の優先順位は付きません。もし優先順位が 0 や 1 であっても、優先順位が 2 であるものとして処理されます。MQMD 構造体の *Priority* フィールドは変更されません。

メッセージ・グループについては、『[MQGMO - Options フィールド](#)』にあるグループおよびセグメント・オプションの説明を参照してください。

846 ページの表 625 および 847 ページの表 626 は、様々な場面に合わせて使用すべき索引タイプを示しています。

表 625. MQGMO_LOGICAL_ORDER が指定されない場合の、キュー索引タイプの推奨値または必須値		
MQGET 呼び出しでの選択基準	非共有キューの索引タイプ	共有キューの索引タイプ
なし	任意	任意
1 つの ID を使用した選択：		
メッセージ ID	MQIT_MSG_ID を推奨	MQIT_NONE または MQIT_MSG_ID が必須。 MQIT_MSG_ID を推奨
相関 ID	MQIT_CORREL_ID を推奨	MQIT_CORREL_ID が必須
グループ ID	MQIT_GROUP_ID を推奨	MQIT_GROUP_ID が必須
2 つの ID を使用した選択：		

表 625. MQGMO_LOGICAL_ORDER が指定されない場合の、キュー索引タイプの推奨値または必須値 (続き)

MQGET 呼び出しでの選択基準	非共有キューの索引タイプ	共有キューの索引タイプ
メッセージ ID および相関 ID	MQIT_MSG_ID または MQIT_CORREL_ID を推奨	MQIT_NONE または MQIT_MSG_ID または MQIT_CORREL_ID が必須 (効率のためには、最も違いのある キーを持つ MQMD フィールドに 合わせて索引タイプを選ぶことを 推奨します)
メッセージ ID およびグループ ID	MQIT_MSG_ID または MQIT_GROUP_ID を推奨	サポート対象外
相関 ID およびグループ ID	MQIT_CORREL_ID または MQIT_GROUP_ID を推奨	サポート対象外
3 つの ID を使用した選択:		
メッセージ ID および相関 ID およ びグループ ID	MQIT_MSG_ID または MQIT_CORREL_ID または MQIT_GROUP_ID を推奨	サポート対象外
グループに関連する基準を用いた選択:		
グループ ID およびメッセージ・シ ーケンス番号	MQIT_GROUP_ID が必須	MQIT_GROUP_ID が必須
メッセージ・シーケンス番号 (必ず 1)	MQIT_GROUP_ID が必須	MQIT_GROUP_ID が必須
メッセージ・トークンを使用した選択:		
アプリケーションが使用するメッ セージ・トークン	MQIT_MSG_TOKEN を使用しないでください	
WLM が使用するメッセージ・トー クン	MQIT_MSG_TOKEN が必須	サポート対象外

表 626. MQGMO_LOGICAL_ORDER が指定される場合の、キュー索引タイプの推奨値または必須値

MQGET 呼び出しでの選択基準	非共有キューの索引タイプ	共有キューの索引タイプ
なし	MQIT_GROUP_ID が必須	MQIT_GROUP_ID が必須
1 つの ID を使用した選択:		
メッセージ ID	MQIT_GROUP_ID が必須	サポート対象外
相関 ID	MQIT_GROUP_ID が必須	サポート対象外
グループ ID	MQIT_GROUP_ID が必須	MQIT_GROUP_ID が必須
2 つの ID を使用した選択:		
メッセージ ID および相関 ID	MQIT_GROUP_ID が必須	サポート対象外
メッセージ ID およびグループ ID	MQIT_GROUP_ID が必須	サポート対象外
相関 ID およびグループ ID	MQIT_GROUP_ID が必須	サポート対象外

表 626. MQGMO_LOGICAL_ORDER が指定される場合の、キュー索引タイプの推奨値または必須値 (続き)		
MQGET 呼び出しでの選択基準	非共有キューの索引タイプ	共有キューの索引タイプ
3 つの ID を使用した選択:		
メッセージ ID および相関 ID およびグループ ID	MQIT_GROUP_ID が必須	サポート対象外

この属性の値を判別するには、MQINQ 呼び出しで MQIA_INDEX_TYPE セレクターを使用します。

 この属性は、z/OS でのみサポートされます。

InhibitGet (MQLONG)

このキューの取得操作を許可するかどうかを制御します。

表 627. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X	X		

別名キューの場合、MQGET 呼び出しを成功させるためには、取得操作時に、別名キューと基本キューの両方に対して取得操作が許可されている必要があります。値は、次のいずれか 1 つです。

MQQA_GET_INHIBITED

取得操作は禁止されています。

MQGET 呼び出しは失敗し、理由コード MQRC_GET_INHIBITED が戻ります。

MQGMO_BROWSE_FIRST または MQGMO_BROWSE_NEXT を指定した MQGET 呼び出しが含まれます。

注: 作業単位内で操作中の MQGET 呼び出しが正常に完了した場合には、**InhibitGet** 属性の値をあとで MQQA_GET_INHIBITED に変更しても、作業単位のコミットが妨げられることはありません。

MQQA_GET_ALLOWED

取得操作は許可されています。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_INHIBIT_GET セレクターを使用します。この属性の値を変更するには、MQSET 呼び出しを使用します。

InhibitPut (MQLONG)

このキューの書き込み操作を許可するかどうかを制御します。

表 628. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X	X	X	X

キュー名解決パス内に定義が 2 つ以上ある場合に、MQPUT または MQPUT1 呼び出しを成功させるためには、PUT 操作時に、そのパス内のすべての定義 (キュー・マネージャーの別名定義を含む) に対して PUT 操作が許可されている必要があります。値は、次のいずれか 1 つです。

MQQA_PUT_INHIBITED

書き込み操作は使用禁止です。

MQPUT および MQPUT1 呼び出しは失敗し、理由コード MQRC_PUT_INHIBITED が戻ります。

注: 作業単位内で操作中の MQPUT 呼び出しが正常に完了した場合には、**InhibitPut** 属性の値をあとで MQQA_PUT_INHIBITED に変更しても、作業単位のコミットが妨げられることはありません。

MQQA_PUT_ALLOWED

書き込み操作が許可されています。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_INHIBIT_PUT セレクターを使用します。この属性の値を変更するには、MQSET 呼び出しを使用します。

InitiationQName (MQCHAR48)

これは、ローカル・キュー・マネージャー上に定義されたキューの名前です。キューのタイプは MQQT_LOCAL でなければなりません。

表 629. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X				

この属性が持っているキューにメッセージが到着した結果、アプリケーションを始動させる必要があったときに、キュー・マネージャーは、トリガー・メッセージを開始キューに送ります。トリガー・メッセージを受け取ったあとで該当するアプリケーションを開始させるトリガー・モニター・アプリケーションによって、開始キューをモニターする必要があります。

この属性の値を判別するには、MQINQ 呼び出しで MQCA_INITIATION_Q_NAME セレクターを使用します。この属性の長さは MQ_Q_NAME_LENGTH によって指定されます。

MaxMsgLength (MQLONG)

これは、キューに入れることができる最長の物理メッセージの長さの上限です。

表 630. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

しかし、**MaxMsgLength** キュー属性は、**MaxMsgLength** キュー・マネージャー属性とは別に設定できるため、キューに置くことのできる物理メッセージの実際の最大長は、これら 2 つの値のうちの小さい方の値になります。

キュー・マネージャーがセグメント化をサポートしている場合は、アプリケーションで MQMF_SEGMENTATION_ALLOWED フラグを MQMD に指定しているときに限って、これら 2 つの **MaxMsgLength** 属性の小さい方の値より長い論理メッセージをアプリケーションから書き込むことができます。そのフラグを指定する場合、論理メッセージの長さの上限は 999 999 999 バイトですが、多くの場合、オペレーティング・システムによって、またはアプリケーションが実行されている環境によってリソースが制約される結果、上限はこれよりさらに小さい値になります。

キューに置こうとするメッセージが長すぎると失敗し、以下のいずれかの理由コードが戻されます。

- MQRC_MSG_TOO_BIG_FOR_Q。これは、キューにとってそのメッセージが長すぎる場合に返ります。
- MQRC_MSG_TOO_BIG_FOR_Q_MGR。これは、キュー・マネージャーにとってそのメッセージが長すぎる場合に返ります。キューにとって長すぎるわけではありません。

MaxMsgLength 属性の下限はゼロで、上限は 100 MB (104 857 600 バイト) です。

詳しくは、『MQPUT - BufferLength パラメーター』を参照してください。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_MAX_MSG_LENGTH セレクターを使用します。

MaxQDepth (MQLONG)

これは、任意の一時点でキューに存在できる物理メッセージ数の定義上の上限です。

表 631. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

すでに **MaxQDepth** 個のメッセージが入っているキューにさらにメッセージを入れようとするとう失敗し、理由コード MQRC_Q_FULL が戻ります。

作業単位ごとの処理やメッセージのセグメント化を行うと、いずれの場合もキューにある物理メッセージの実際の数が **MaxQDepth** を超えてしまいます。ただし、キューにあるすべてのメッセージは、MQGET 呼び出しを使用して検索できるため、このことがメッセージの検索に影響するわけではありません。

値はゼロ以上です。上限は、環境により決定されます。

- 以下のプラットフォームの場合、この値は 999 999 999 を超えてはなりません。

-  AIX
-  Linux
-  Windows
-  z/OS

-  IBM i では、値は 640 000 を超えることはできません。

注：キューに入っているメッセージの数が **MaxQDepth** に満たない場合でも、キューのために使用可能な保管スペースを使い切ってしまう可能性があります。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_MAX_Q_DEPTH セレクターを使用します。

MsgDeliverySequence (MQLONG)

表 632. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

これは、MQGET 呼び出しがメッセージをアプリケーションに戻す順番を決定します。

MQMDS_FIFO

メッセージは FIFO (先入れ先出し法) の順に返されます。

MQGET 呼び出しによって戻されるメッセージは、優先順位に関係なく、その呼び出しで指定されている選択基準を満たすメッセージのうちの最初のメッセージになります。

MQMDS_PRIORITY

メッセージが優先順位順に戻されます。

MQGET 呼び出しによって戻されるメッセージが、その呼び出しで指定されている選択基準を満たすメッセージのうちで、最も優先順位の高いメッセージになります。各優先順位内では、メッセージは FIFO (先入れ先出し) の順番で返されます。

- z/OS では、キューの *IndexType* が MQIT_GROUP_ID である場合、メッセージ・グループがアプリケーションに戻される順序は **MsgDeliverySequence** 属性で指定されます。グループが戻される特定のシーケンスは、各グループの最初のメッセージの位置と優先順位によって判別されます。キューは、MQGET 呼び出しで MQGMO_LOGICAL_ORDER オプションを使用することにより、最も効率的にメッセージを検索できるように最適化されているため、キュー上のメッセージの物理順序は定義されません。
- z/OS では、*IndexType* が MQIT_GROUP_ID で、MQMDS_PRIORITY が **MsgDeliverySequence** になっていると、キュー・マネージャーは、メッセージ優先順位 0 および 1 を使用して、メッセージを検索する順序を論理順序に最適化します。結果として、グループ内の最初のメッセージには、0 や 1 の優先順位は付きません。もし優先順位が 0 や 1 であっても、優先順位が 2 であるものとして処理されます。MQMD 構造体の *Priority* フィールドは変更されません。

キューにメッセージが入っているときに関連する属性が変更されると、配布順序は次のようになります。

- MQGET 呼び出しによってメッセージが戻される順序は、メッセージがキューに到着した時点でキューに対して有効な **MsgDeliverySequence** 属性と **DefPriority** 属性の値によって決定されます。

- メッセージが到着したときに *MsgDeliverySequence* が MQMDS_FIFO であれば、そのメッセージは優先順位が *DefPriority* である場合と同様の位置でキューに入れます。これは、メッセージのメッセージ記述子の *Priority* フィールドの値には影響しません。そのフィールドは、メッセージが最初に書き込まれたときの値を保持します。
- メッセージが到着したときに *MsgDeliverySequence* が MQMDS_PRIORITY であれば、メッセージ記述子内の *Priority* フィールドで指定されている優先順位に該当する位置でキューに入れます。

キュー上にメッセージがあるときに **MsgDeliverySequence** 属性の値が変更された場合、キュー上のメッセージの順序は変更されません。

メッセージがキューにある間に **DefPriority** 属性の値が変更された場合は、**MsgDeliverySequence** 属性が MQMDS_FIFO に設定されていても、メッセージは FIFO の順番に配布されるとは限りません。つまり、より高い優先順位でキューに入れられたメッセージが最初に送られます。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_MSG_DELIVERY_SEQUENCE セレクターを使用します。

NonPersistentMessageClass (MQLONG)

非持続メッセージの信頼性目標。

表 633. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

これは以下のように、このキューに書き込まれた非持続メッセージが廃棄される環境を指定します。

MQNPM_CLASS_NORMAL

非持続メッセージは、キュー・マネージャー・セッションの存続時間に制限されます。メッセージはキュー・マネージャーの再始動時に廃棄されます。これは、非共有キューの場合にのみ有効であり、デフォルト値になります。

MQNPM_CLASS_HIGH

キュー・マネージャーは、キューの存続時間に非持続メッセージを保存しようとしします。非持続メッセージは、障害が発生した場合に失われることがあります。共有キューの場合、この値が適用されます。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_NPM_CLASS セレクターを使用します。

OpenInputCount (MQLONG)

これは、MQGET 呼び出しを使用してキューからメッセージを除去するために現在有効なハンドルの数です。

表 634. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X				

ローカル・キュー・マネージャーに認識されているハンドルの総数です。キューが共有キューである場合、この数には、ローカル・キュー・マネージャーが属しているキュー共有グループにある、他のキュー・マネージャーのキューのために実行された入力オープンは含まれません。

このカウントには、このキューに解決される別名キューが入力用にオープンされたときのハンドルが含まれます。また、カウントには、入力が行われなかったアクション (例えば、ブラウズのためのみにキューがオープンされた場合) のためにキューがオープンされたときのハンドルは含まれません。

属性の値は、キュー・マネージャーの動作に応じて変動します。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_OPEN_INPUT_COUNT セレクターを使用します。

OpenOutputCount (MQLONG)

これは、MQPUT 呼び出しを使用してキューにメッセージを追加するために現在有効なハンドルの数です。

表 635. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X				

ローカル・キュー・マネージャーに認識されているハンドルの総数です。ローカル・キューに対してリモート・キュー・マネージャーで実行された出力のためのオープンは含まれません。キューが共有キューである場合、この数には、ローカル・キュー・マネージャーが属しているキュー共有グループにある、他のキュー・マネージャーのキューのために実行された出力のオープンは含まれません。

このカウントには、このキューに解決される別名キューが出力用にオープンされたときのハンドルが含まれます。また、カウントには、出力が行われなかったアクション (例えば、照会のためのみにキューがオープンされた場合) のためにキューがオープンされたときのハンドルは含まれません。

属性の値は、キュー・マネージャーの動作に応じて変動します。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_OPEN_OUTPUT_COUNT セレクターを使用します。

ProcessName (MQCHAR48)

これは、ローカル・キュー・マネージャーについて定義されたプロセス・オブジェクトの名前です。プロセス・オブジェクトは、キューにサービスを提供できるプログラムを識別します。

表 636. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

この属性の値を判別するには、MQINQ 呼び出しで MQCA_PROCESS_NAME セレクターを使用します。属性の長さは、MQ_PROCESS_NAME_LENGTH で指定します。

PropertyControl (MQLONG)

MQGMO_PROPERTIES_AS_Q_DEF オプションを指定した MQGET 呼び出しを使用して、キューから受け取るメッセージのメッセージ・プロパティの処理方法を指定します。

表 637. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X	X		

値は、次のいずれか 1 つです。

MQPROP_ALL

メッセージのすべてのプロパティは、そのメッセージがアプリケーションに送達される際に組み込まれます。メッセージ記述子 (または拡張) に含まれるものを除くプロパティは、メッセージ・データ中の 1 つ以上の MQRFH2 ヘッダー中に入れられます。メッセージ・ハンドルが供給されている場合は、メッセージ・ハンドル内のプロパティを戻すという振る舞いをします。

MQPROP_COMPATIBILITY

メッセージに含まれるプロパティの接頭部が mcd.、JMSS usr. または mqext.、すべてのメッセージ・プロパティは、MQRFH2 ヘッダーでアプリケーションに配信されます。それ以外の場合、メッセージ記述子 (または拡張) に含まれるものを除くメッセージのプロパティはすべて廃棄され、アプリケーションにアクセスできなくなります。これがデフォルト値です。これにより、JMS 関連プロパティがメッセージ・データ内の MQRFH2 ヘッダーにあると想定するアプリケーションを、変更せずに

そのまま使用することができます。メッセージ・ハンドルが提供される場合は、そのメッセージ・ハンドルにプロパティが戻されるという動作になります。

MQPROP_FORCE_MQRFH2

プロパティは、アプリケーションがメッセージ・ハンドルを指定しているかどうかには関係なく、MQRFH2 ヘッダー内のメッセージ・データによって常に戻されます。MQGET 呼び出し上の MQGMO 構造体の MsgHandle フィールド中で指定された有効なメッセージ・ハンドルは無視されます。メッセージのプロパティは、メッセージ・ハンドル経由ではアクセスできません。

MQPROP_NONE

メッセージ記述子 (または拡張) に含まれるものを除くメッセージのプロパティはすべて、メッセージがアプリケーションに送達される前にメッセージから除去されます。メッセージ・ハンドルが供給されている場合は、メッセージ・ハンドル内のプロパティを戻すという振る舞いをします。

このパラメーターは、ローカル・キュー、別名キュー、およびモデル・キューに適用可能です。この値を判別するには、MQINQ 呼び出しで MQIA_PROPERTY_CONTROL セレクターを使用します。

QDepthHighEvent (MQLONG)

これは、Queue Depth High イベントが生成されるかどうかを制御します。

表 638. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

キュー項目数高イベントは、アプリケーションがキューにメッセージを書き込んだために、キューにあるメッセージの数がキュー項目数高しきい値以上になったことを示します (**QDepthHighLimit** 属性を参照してください)。

注: この属性の値は動的に変化します。

値は、次のいずれか 1 つです。

MQEVR_DISABLED

イベント報告は無効です。

MQEVR_ENABLED

イベント報告は有効です。

イベントの詳細については、[イベント・モニター](#)を参照してください。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_Q_DEPTH_HIGH_EVENT セレクターを使用します。

この属性は z/OS でサポートされていますが、MQINQ 呼び出しを使用して値を判別することはできません。

QDepthHighLimit (MQLONG)

これは、Queue Depth High イベントを生成するためにキューの深さが比較されるしきい値です。

表 639. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

このイベントは、アプリケーションがキューにメッセージを書き込んだため、キューに入っているメッセージの数がキューのサイズ上限のしきい値以上になったことを示します。『**QDepthHighEvent** 属性』を参照してください。

値は、キューの最大サイズ (**MaxQDepth** 属性) に対するパーセントで表され、0 以上、100 以下の値です。デフォルト値は 80 です。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_Q_DEPTH_HIGH_LIMIT セレクターを使用します。

この属性は z/OS でサポートされていますが、MQINQ 呼び出しを使用して値を判別することはできません。

QDepthLowEvent (MQLONG)

これは、Queue Depth Low イベントを生成するかどうかを制御します。

表 640. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

キュー・サイズ下限イベントは、アプリケーションがキューからメッセージを取り出したためキューに入っているメッセージの数がキューのサイズ下限しきい値以下になったことを示します (『QDepthLowLimit 属性』を参照してください)。

注：この属性の値は動的に変化します。

値は、次のいずれか 1 つです。

MQEVR_DISABLED

イベント報告は無効です。

MQEVR_ENABLED

イベント報告は有効です。

イベントの詳細については、[イベント・モニター](#)を参照してください。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_Q_DEPTH_LOW_EVENT セレクターを使用します。

この属性は z/OS でサポートされていますが、MQINQ 呼び出しを使用して値を判別することはできません。

QDepthLowLimit (MQLONG)

これは、Queue Depth Low イベントを生成するためにキューの深さが比較されるしきい値です。

表 641. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

このイベントは、アプリケーションがキューからメッセージを取り出したために、キューに入っているメッセージの数がキューのサイズ下限のしきい値以下になったことを示します。『QDepthLowEvent 属性』を参照してください。

値は、キューの最大サイズ (**MaxQDepth** 属性) に対するパーセントで表され、0 以上、100 以下の値です。デフォルト値は 20 です。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_Q_DEPTH_LOW_LIMIT セレクターを使用します。

この属性は z/OS でサポートされていますが、MQINQ 呼び出しを使用して値を判別することはできません。

QDepthMaxEvent (MQLONG)

これはキュー満杯イベントが生成されるかどうかを制御します。Queue Full イベントは、キューが満杯であるためにキューへの書き込みが拒否されたことを示します。つまり、キュー項目数が既に最大値に達しています。

表 642. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

注：この属性の値は動的に変化します。

値は、次のいずれか 1 つです。

MQEVR_DISABLED

イベント報告は無効です。

MQEVR_ENABLED

イベント報告は有効です。

イベントの詳細については、[イベント・モニター](#)を参照してください。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_Q_DEPTH_MAX_EVENT セレクターを使用します。

この属性は z/OS でサポートされていますが、MQINQ 呼び出しを使用して値を判別することはできません。

QDesc (MQCHAR64)

このフィールドは、記述の注釈として使用します。

表 643. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X	X	X	X

フィールドの内容はキュー・マネージャーにとって重要なものではありませんが、表示できる文字以外は使用しないでください。フィールドにヌル文字を入れることはできません。また、必要に応じて、右側がブランクで埋められます。DBCS をインストール済みの環境では、このフィールドに DBCS 文字を入れることができます (最大フィールド長として 64 バイトが適用されます)。

注: このフィールドに、(**CodedCharSetId** キュー・マネージャー属性で定義されている) キュー・マネージャーの文字セットに含まれていない文字が含まれている場合、このフィールドが別のキュー・マネージャーに送信されると、それらの文字が正しく変換されない可能性があります。

この属性の値を判別するには、MQINQ 呼び出しで MQCA_Q_DESC セレクターを使用します。属性の長さは、MQ_Q_DESC_LENGTH で指定します。

QName (MQCHAR48)

これは、ローカル・キュー・マネージャーで定義されたキューの名前です。

表 644. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X		X	X	X

キュー・マネージャー上で定義されたキューはすべて、同一のキュー名前空間を共有します。したがって、MQQT_LOCAL キューと MQQT_ALIAS キューが同じ名前を持つことはできません。

この属性の値を判別するには、MQINQ 呼び出しで MQCA_Q_NAME セレクターを使用します。この属性の長さは MQ_Q_NAME_LENGTH によって指定されます。

QServiceInterval (MQLONG)

これは、サービス間隔上限イベントおよびサービス間隔 OK イベントを生成するための比較に使用されるサービス間隔です。

表 645. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

『QServiceIntervalEvent 属性』を参照してください。

値は、ミリ秒単位で表された 0 以上、999 999 999 以下の値です。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_Q_SERVICE_INTERVAL セレクターを使用します。

この属性は z/OS でサポートされていますが、MQINQ 呼び出しを使用して値を判別することはできません。

QServiceIntervalEvent (MQLONG)

これは、Service Interval High イベントまたは Service Interval OK イベントの生成の有無を制御します。

表 646. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

- サービス間隔上位イベントが生成されるのは、検査の結果、少なくとも **QServiceInterval** 属性によって示されている期間、このキューから取り出されたメッセージがなかったことが分かった場合です。
- サービス間隔 OK イベントが生成されるのは、検査の結果、**QServiceInterval** 属性によって示されている期間内に、このキューからメッセージが取り出されていることが分かった場合です。

注：この属性の値は動的に変化します。

値は、次のいずれか 1 つです。

MQQSIE_HIGH

キュー・サービス間隔上限イベントは有効です。

- キュー・サービス間隔上位イベントが**使用可能**であり、
- キュー・サービス間隔 OK イベントは**使用不可**である。

MQQSIE_OK

キュー・サービス間隔 OK イベントは有効です。

- キュー・サービス間隔上位イベントが**使用不可**であり、
- キュー・サービス間隔 OK イベントは**使用可能**である。

MQQSIE_NONE

どのキュー・サービス間隔イベントも無効です。

- キュー・サービス間隔上位イベントが**使用不可**であり、
- キュー・サービス間隔 OK イベントも**使用不可**である。

共有キューでは、この属性の値は無視されます。値 MQQSIE_NONE が想定されます。

イベントの詳細については、[イベント・モニター](#)を参照してください。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_Q_SERVICE_INTERVAL_EVENT セレクターを使用します。

z/OS では、MQINQ 呼び出しを使用して、この属性の値を判別することはできません。

QSGDisp (MQLONG)

これは、キューの後処理を指定します。

表 647. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X		X	X	

値は、次のいずれか 1 つです。

MQQSGD_Q_MGR

このオブジェクトには、キュー・マネージャーの後処理が含まれます。これは、このオブジェクトの定義を、ローカル・キュー・マネージャーだけが認識し、キュー共有グループ内の他のキュー・マネージャーは認識しないことを意味します。

キュー共有グループ内の各キュー・マネージャーは、現行オブジェクトと名前およびタイプが同じオブジェクトを持つことができます。ただし、それらは別のオブジェクトであり、相互に何の相関もありません。それらの属性が互いに同じになるように制約されることはありません。

MQQSGD_COPY

このオブジェクトは、共用リポジトリ内に存在するマスター・オブジェクト定義のローカル・コピーです。キュー共有グループ内の各キュー・マネージャーが、このオブジェクトの独自のコピーを持つことができます。最初は、すべてのコピーが同じ属性を持っていますが、MQSC コマンドを使用して、各コピーの属性を他のコピーとは異なるものに変更することができます。共有リポジトリのマスター定義が更新されると、コピーの属性は再同期化されます。

MQQSGD_SHARED

このオブジェクトは共有の属性指定を持ちます。これは、共有リポジトリ内にこのオブジェクトの単一インスタンスが存在していて、それがキュー共有グループ内の全キュー・マネージャーから認識されることを意味します。グループ内のキュー・マネージャーは、このオブジェクトにアクセスするとき、このオブジェクトの単一の共有インスタンスにアクセスしています。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_QSG_DISP セレクターを使用します。

 この属性は、z/OS でのみサポートされます。

QueueAccounting (MQLONG)

表 648. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X	X	X	

これは、キューのアカウンティング・データの収集を制御します。アカウンティング・データをこのキューで収集するには、QMGR 属性 ACCTQ または MQCONNX 呼び出しでの MQCNO 構造の Options フィールドを使用して、この接続のアカウンティング・データも有効にする必要があります。

この属性の値は、次のいずれかです。

MQMON_Q_MGR

このキューのアカウンティング・データは、QMGR 属性 ACCTQ の設定に基づいて収集されます。これはデフォルト設定です。

MQMON_OFF

このキューのアカウンティング・データを収集しません。

MQMON_ON

このキューのアカウンティング・データを収集します。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_ACCOUNTING_Q セレクターを使用します。

QueueMonitoring (MQLONG)

キューのオンライン・モニタリング・データの収集を制御します。

表 649. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

値は、次のいずれか 1 つです。

MQMON_Q_MGR

QueueMonitoring キュー・マネージャー属性の設定に従ってモニター・データを収集します。これがデフォルト値です。

MQMON_OFF

このキューのオンライン・モニター・データ収集はオフになります。

MQMON_LOW

QueueMonitoring キュー・マネージャー属性の値が MQMON_NONE でない場合、オンライン・モニター・データ収集はオンになりますが、このキューのデータ収集率は低くなります。

MQMON_MEDIUM

QueueMonitoring キュー・マネージャー属性の値が MQMON_NONE でない場合、オンライン・モニター・データ収集はオンになりますが、このキューのデータ収集率は中程度です。

MQMON_HIGH

QueueMonitoring キュー・マネージャー属性の値が MQMON_NONE でない場合、オンライン・モニター・データ収集はオンになりますが、このキューのデータ収集率は高くなります。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_MONITORING_Q セレクターを使用します。

QueueStatistics (MQCHAR12)

表 650. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X	X	X	

これは、キューの統計データのコレクションを制御します。

この属性の値は、次のいずれかです。

MQMON_Q_MGR

このキューのアカウンティング・データは、QMGR 属性 STATQ の設定に基づいて収集されます。これはデフォルト設定です。

MQMON_OFF

このキューの統計データ収集をオフに切り替えます。

MQMON_ON

このキューの統計データ収集を使用可能にします。

QType (MQLONG)

表 651. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X		X	X	X

これはキューのタイプです。この値には、次のいずれかが含まれます。

MQQT_ALIAS

別名キュー定義。

MQQT_CLUSTER

クラスター・キュー。

MQQT_LOCAL

ローカル・キュー。

MQQT_REMOTE

リモート・キューのローカル定義。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_Q_TYPE セレクターを使用します。

RemoteQMgrName (MQCHAR48)

表 652. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
			X	

RemoteQName が定義されているリモート・キュー・マネージャーの名前です。 **RemoteQName** キューの **QSGDisp** 値が MQQSGD_COPY または MQQSGD_SHARED である場合は、 **RemoteQMgrName** は **RemoteQName** を所有するキュー共有グループの名前にすることができます。

アプリケーションがリモート・キューのローカル定義をオープンする場合、 **RemoteQMgrName** をブランクにしたりローカル・キュー・マネージャーの名前を指定することはできません。 **XmitQName** がブランクの場合は、 **RemoteQMgrName** と同じ名前のローカル・キューが伝送キューとして使用されます。

RemoteQMgrName という名前のキューが存在しない場合は、 **DefXmitQName** キュー・マネージャー属性に指定されているキューが使用されます。

この定義がキュー・マネージャー別名に使用される場合、 **RemoteQMgrName** は別名が割り当てられるキュー・マネージャーの名前です。これは、ローカル・キュー・マネージャーの名前であっても構いません。そうでない場合に、オープンが行われるときに **XmitQName** がブランクである場合は、 **RemoteQMgrName** と同じ名前のローカル・キューが存在しなければなりません。そのキューが、伝送キューとして使用されます。

定義が応答先別名に使用される場合、この名前は、 **ReplyToQMgr** であるキュー・マネージャーの名前です。

注: キュー定義の作成時または変更時には、この属性に関して指定されている値の妥当性検査は行われません。

この属性の値を判別するには、MQINQ 呼び出しで MQCA_REMOTE_Q_MGR_NAME セレクターを使用します。属性の長さは、MQ_Q_MGR_NAME_LENGTH で指定します。

RemoteQName (MQCHAR48)

表 653. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
			X	

これは、リモート・キュー・マネージャー **RemoteQMgrName** で認識されているキューの名前です。

アプリケーションがリモート・キューのローカル定義をオープンする場合、オープン時に **RemoteQName** をブランクにしてはなりません。

この定義がキュー・マネージャー別名定義に使用される場合、オープン時に **RemoteQName** はブランクでなければなりません。

定義が応答先 (reply-to) 別名に使用されている場合、この名前は、 **ReplyToQ** になるキューの名前です。

注: キュー定義の作成時または変更時には、この属性に関して指定されている値の妥当性検査は行われません。

この属性の値を判別するには、MQINQ 呼び出しで MQCA_REMOTE_Q_NAME セレクターを使用します。この属性の長さは MQ_Q_NAME_LENGTH によって指定されます。

RetentionInterval (MQLONG)

キューを保存する期間です。この時間が経過すると、キューは削除対象となります。

表 654. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

時間は、そのキューが作成された日付と時刻を基点として時間数で測定されます。キューの作成日時は、**CreationDate** 属性および **CreationTime** 属性に記録されます。

この情報を用いて、ハウスキーピング・アプリケーションやオペレーターは、不要になったキューを識別して削除することができます。

注: キュー・マネージャーが、この属性に基づいてキューを削除したり、あるいは保存期間が満了していないキューが削除されないようにしたりするための操作を行うことはありません。何らかの操作が必要な場合は、ユーザーが行わなければなりません。

現実的な保存期間は、永続動的キュー (『**DefinitionType** 属性』を参照) が累積されるのを防ぐ目的で使用します。ただし、この属性は、事前定義されたキューでも使用することができます。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_RETENTION_INTERVAL セレクターを使用します。

Scope (MQLONG)

これは、このキューの項目もセル・ディレクトリーに存在するかどうかを制御します。

表 655. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X		X	X	

セル・ディレクトリーは、インストール可能な名前サービスにより提供されます。値は、次のいずれか 1 つです。

MQSCO_Q_MGR

キューの定義は、そのキューを所有しているキュー・マネージャーの有効範囲を超えません。他のキュー・マネージャーからそのキューを出力用にオープンするときには、キューを所有しているキュー・マネージャーの名前を指定する必要があります。あるいは、呼び出す側のキュー・マネージャーがそのキューのローカル定義を持っていなければなりません。

MQSCO_CELL

キュー定義は、セル内のすべてのキュー・マネージャーが使用できるセル・ディレクトリー内に入っています。キューの名前を指定することによって、セル内のどのキュー・マネージャーからも出力できるように、キューをオープンすることができます。キューを所有するキュー・マネージャーの名前を指定する必要はありません。しかし、その名前を持つキューのローカル定義も持っているセル内のキュー・マネージャーは、そのキュー定義を使用できません。ローカル定義の方が優先されるためです。

セル・ディレクトリーは、インストール可能な名前サービスにより提供されます。

モデル・キューと動的キューは、セル有効範囲を持つことはできません。

値は、セル・ディレクトリーをサポートする名前サービスが構成されている場合にのみ有効です。

この属性を判別するには、MQINQ 呼び出しで MQIA_SCOPE セレクターを使用します。

この属性のサポートには、次のような制約事項があります。

- IBM i ではこの属性がサポートされますが、有効なのは MQSCO_Q_MGR のみです。
- z/OS では、この属性はサポートされていません。

Shareability (MQLONG)

これは、キューを同時に複数回入力できるようにオープンできるかどうかを示します。

表 656. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

値は、次のいずれか 1 つです。

MQQA_SHAREABLE

キューは共有可能。

MQOO_INPUT_SHARED オプションを使用して、複数のオープンを行うことができます。

MQQA_NOT_SHAREABLE

キューは共有不可。

MQOO_INPUT_SHARED オプションを指定した MQOPEN 呼び出しは、MQOO_INPUT_EXCLUSIVE として処理されます。

この属性値を判別するには、MQINQ 呼び出しで MQIA_SHAREABILITY セレクターを使用します。

StorageClass (MQCHAR8)

これは、キューを保持するために使用される物理ストレージを定義するユーザー定義の名前です。実際には、メッセージは、メモリー・バッファーからページアウトされる必要がある場合にのみ、ディスクに書き込まれます。

表 657. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

この属性値を判別するには、MQINQ 呼び出しで MQCA_STORAGE_CLASS セレクターを使用します。属性の長さは、MQ_STORAGE_CLASS_LENGTH で指定します。

 この属性は、z/OS でのみサポートされます。

TriggerControl (MQLONG)

これは、キューをサービスするアプリケーションを開始するためにトリガー・メッセージを開始キューに書き込むかどうかを制御します。

表 658. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

これは、以下のいずれかになります。

MQTC_OFF

トリガー・メッセージはこのキューに書き込まれません。TriggerType の値は、この場合には無効です。

MQTC_ON

該当するトリガー・イベントが起こったときに、トリガー・メッセージがこのキューに書き込まれます。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_TRIGGER_CONTROL セレクターを使用します。この属性の値を変更するには、MQSET 呼び出しを使用します。

TriggerData (MQCHAR64)

これは、キュー・マネージャーがトリガー・メッセージに挿入するフリー・フォーマット・データであり、このキューにメッセージが到着すると、トリガー・メッセージが開始キューに書き込まれます。

表 659. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

このデータの内容は、キュー・マネージャーにとっては意味のないものです。データは、開始キューを処理するトリガー・モニター・アプリケーション、あるいはトリガー・モニターによって開始されるアプリケーションにとって意味があります。

文字ストリング内にヌルを入れることはできません。このストリングは、必要に応じて、右側にブランクが埋め込まれます。

この属性の値を判別するには、MQINQ 呼び出しで MQCA_TRIGGER_DATA セレクターを使用します。この属性の値を変更するには、MQSET 呼び出しを使用します。属性の長さは、MQ_TRIGGER_DATA_LENGTH で指定します。

TriggerDepth (MQLONG)

表 660. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

トリガー・メッセージを書き込む前にキューに入れなければならない、優先順位が *TriggerMsgPriority* 以上のメッセージの数です。この属性は、*TriggerType* が MQTT_DEPTH に設定されている場合に適用されます。*TriggerDepth* の値は、1 以上です。この属性は他の場合には使われません。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_TRIGGER_DEPTH セレクターを使用します。この属性の値を変更するには、MQSET 呼び出しを使用します。

TriggerMsgPriority (MQLONG)

これ以下のメッセージはトリガー・メッセージの生成に寄与しないという優先順位（つまり、トリガー・メッセージを生成するかどうかを決定するときに、キュー・マネージャーがこれらのメッセージを無視する）です。

表 661. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

TriggerMsgPriority は、ゼロ (最下位) から *MaxPriority* (最上位。『[MaxPriority 属性](#)』を参照) の範囲にあります。値がゼロの場合は、すべてのメッセージがトリガー・メッセージ生成の原因となります。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_TRIGGER_MSG_PRIORITY セレクターを使用します。この属性の値を変更するには、MQSET 呼び出しを使用します。

TriggerType (MQLONG)

これは、このキューに到着したメッセージの結果としてトリガー・メッセージが書き込まれる条件を制御します。

表 662. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

以下の値がどれか 1 つ含まれています。

MQTT_NONE

メッセージがキューに到着した結果、トリガー・メッセージは書き込まれません。*TriggerControl* を MQTC_OFF に設定するのと同じ効果があります。

MQTT_FIRST

トリガー・メッセージは、キューの中で優先順位が *TriggerMsgPriority* 以上であるメッセージの数が 0 から 1 に変化すると、必ず書き込まれます。

MQTT_EVERY

トリガー・メッセージは、優先順位が *TriggerMsgPriority* 以上であるメッセージがキューに到着するたびに書き込まれます。

MQTT_DEPTH

キュー上の優先順位 *TriggerMsgPriority* 以上のメッセージの数が *TriggerDepth* 以上になると、トリガー・メッセージが書き込まれます。トリガー・メッセージが書き込まれた後、*TriggerControl* は *MQTC_OFF* に設定され、トリガーが明示的にオンになるまでそれ以上トリガーされないようにします。

この属性の値を判別するには、*MQINQ* 呼び出しで *MQIA_TRIGGER_TYPE* セレクターを使用します。この属性の値を変更するには、*MQSET* 呼び出しを使用します。

Usage (MQLONG)

これは、キューが使用される対象を示します。

表 663. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

値は、次のいずれか 1 つです。

MQUS_NORMAL

アプリケーションがメッセージを書き込んだり読み取ったりする際に使用するキューです。伝送キューではありません。

MQUS_TRANSMISSION

リモート・キュー・マネージャー宛てのメッセージを保存するために使用されるキューです。アプリケーションがリモート・キューにメッセージを送信すると、ローカル・キュー・マネージャーは、そのメッセージを特別な形式で該当する伝送キューに一時保管します。次に、メッセージ・チャンネル・エージェントが、伝送キューからメッセージを読み取り、リモート・キュー・マネージャーに伝送します。リモート管理の構成方法について詳しくは、[キュー・マネージャーのリモート管理の構成](#)を参照してください。

伝送キューに直接メッセージを書き込むために *MQOO_OUTPUT* に備えて伝送キューをオープンできるのは、特権アプリケーションに限られています。通常、これを行えるのはユーティリティ・アプリケーションのみです。メッセージ・データの形式を間違えないようにしてください ([609 ページの『MQXQH - 伝送キュー・ヘッダー』](#)を参照)。形式が正しくないと、伝送処理時にエラーが起こる可能性があります。 *MQPMO_*_CONTEXT* コンテキスト・オプションが指定されていない場合は、コンテキストの引き渡しも設定も行われません。

この属性の値を判別するには、*MQINQ* 呼び出しで *MQIA_USAGE* セレクターを指定します。

XmitQName (MQCHAR48)

これは伝送キュー名です。この属性が非ブランクの場合 (リモート・キューの場合、またはキュー・マネージャー別名定義の場合)、メッセージの転送に使用されるローカル伝送キューの名前を指定します。

表 664. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
			X	

XmitQName がブランクの場合、**RemoteQMgrName** と同じ名前のローカル・キューが伝送キューとして使用されます。**RemoteQMgrName** という名前のキューが存在しない場合は、**DefXmitQName** キュー・マネージャー属性に指定されているキューが使用されます。

定義がキュー・マネージャーの別名として使用され、**RemoteQMgrName** がローカル・キュー・マネージャーの名前である場合には、この属性は無視されます。また、この定義が応答先キュー別名定義として使用されている場合にも、これは無視されます。

この属性の値を判別するには、MQINQ 呼び出しで MQCA_XMIT_Q_NAME セレクターを使用します。この属性の長さは MQ_Q_NAME_LENGTH によって指定されます。

名前リストの属性

次の表は、名前リストに特有の属性を要約したものです。属性の説明は、アルファベット順に掲載しています。

名前リストは、すべての IBM MQ システムおよびそれらのシステムに接続された IBM MQ MQI clients 上でサポートされます。

注：このセクションで示されている属性の名前は、MQINQ および MQSET 呼び出しで使用される記述名です。この名前は PCF コマンドの場合と同じです。MQSC コマンドを使用して属性を定義、変更、または表示するときには、代替の短縮名が使用されます。詳細については、[MQSC コマンド](#)を参照してください。

表 665. 名前リストの属性	
属性	説明
AlterationDate	定義が最後に変更された日付
AlterationTime	定義が最後に変更された時刻
NameCount	名前リスト内の名前の数
NamelistDesc	名前リストの記述
NamelistName	名前リスト名
NAMES	<i>NameCount</i> 個の名前のリスト。
NamelistType	名前リスト・タイプ
QSGDisp	キュー共有グループ後処理

AlterationDate (MQCHAR12)

これは、定義を最後に変更した日付です。日付の形式は YYYY-MM-DD で、その後に 2 つの末尾ブランクを付けて長さ 12 バイトになります。

この属性の値を判別するには、MQINQ 呼び出しで MQCA_ALTERATION_DATE セレクターを使用します。この属性の長さは MQ_DATE_LENGTH によって指定されます。

AlterationTime (MQCHAR8)

これは、定義を最後に変更した時刻です。時刻の形式は HH.MM.SS です。

この属性の値を判別するには、MQINQ 呼び出しで MQCA_ALTERATION_TIME セレクターを使用します。この属性の長さは MQ_TIME_LENGTH によって指定されます。

NameCount (MQLONG)

これは、名前リストの名前の数です。これは、ゼロ以上の値です。以下の値が定義されます。

MQNC_MAX_NAMELIST_NAME_COUNT

名前リスト内の名前の最大数。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_NAME_COUNT セレクターを使用します。

NamelistDesc (MQCHAR64)

このフィールドはコメントの記述に使用します。フィールドの値は、定義プロセスで設定されます。フィールドの内容はキュー・マネージャーにとって重要なものではありませんが、表示できる文字以外は使用しないでください。フィールドにヌル文字を入れることはできません。また、必要に応じて、右側がブランクで埋められます。DBCS をインストール済みの環境では、このフィールドに DBCS 文字を入れることができます (最大フィールド長として 64 バイトが適用されます)。

注：このフィールドに、(**CodedCharSetId** キュー・マネージャー属性で定義されている) キュー・マネージャーの文字セットに含まれていない文字が含まれている場合、このフィールドが別のキュー・マネージャーに送信されると、それらの文字が正しく変換されない可能性があります。

この属性の値を判別するには、MQINQ 呼び出しで MQCA_NAMELIST_DESC セレクターを使用します。

属性の長さは、MQ_NAMELIST_DESC_LENGTH で指定します。

NamelistName (MQCHAR48)

ローカル・キュー・マネージャーに定義されている名前リストの名前です。名前リストの名前の詳細については、[その他のオブジェクト名のセクション](#)を参照してください。

名前リストは、それぞれ、同じキュー・マネージャーに属する他の名前リストとは異なる名前を持ちますが、別のタイプのキュー・マネージャー・オブジェクト (キューなど) の名前とは重複していてもかまいません。

この属性の値を判別するには、MQINQ 呼び出しで MQCA_NAMELIST_NAME セレクターを使用します。

属性の長さは、MQ_NAMELIST_NAME_LENGTH で指定します。

NamelistType (MQLONG)

この属性は名前リスト内の名前の種類を指定し、名前リストの使用法を指示します。これは、次の値のいずれかです。

MQNT_NONE

タイプが割り当てられていない名前リスト。

MQNT_Q

キューの名前を含む名前リスト。

MQNT_CLUSTER

クラスターの名前を含む名前リスト。

MQNT_AUTH_INFO

認証情報オブジェクトの名前を含む名前リスト。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_NAMELIST_TYPE セレクターを使用します。

 この属性は、z/OS でのみサポートされます。

Names (MQCHAR48xNameCount)

これは、NameCount の名前のリストです。各名前は、ローカル・キュー・マネージャーに対して定義されるオブジェクトの名前です。オブジェクト名について詳しくは、[IBM MQ オブジェクトの命名規則](#)を参照してください。

この属性の値を判別するには、MQINQ 呼び出しで MQCA_NAMES セレクターを使用します。

リスト内の各名前の長さは、MQ_OBJECT_NAME_LENGTH で指定します。

QSGDisp (MQLONG)

名前リストの後処理を指定します。値は、次のいずれか 1 つです。

MQQSGD_Q_MGR

オブジェクトにはキュー・マネージャーの後処理が含まれます。オブジェクト定義がローカル・キューにのみ認識されます。この定義は、キュー共有グループの他のキュー・マネージャーには認識されません。

キュー共有グループ内の各キュー・マネージャーは、現行オブジェクトと名前およびタイプが同じオブジェクトを持つことができます。ただし、それらは別のオブジェクトであり、相互に何の相関もありません。それらの属性が互いに同じになるように制約されることはありません。

MQQSGD_COPY

このオブジェクトは、共用リポジトリ内に存在するマスター・オブジェクト定義のローカル・コピーです。キュー共有グループ内の各キュー・マネージャーが、このオブジェクトの独自のコピーを持つことができます。最初はすべてのコピーに同じ属性がありますが、MQSC コマンドを使用することによってそれぞれのコピーを変更でき、その属性を他のコピーのものと異なるようにすることができます。共有リポジトリのマスター定義が更新されると、コピーの属性は再同期化されます。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_QSG_DISP セレクターを使用します。

 この属性は、z/OS でのみサポートされます。

プロセス定義の属性

次の表は、プロセス定義に特有の属性を要約したものです。属性は、アルファベット順に説明されています。

注：このセクションの属性の名前は、MQINQ および MQSET 呼び出しで使用される記述名です。この名前は PCF コマンドの場合と同じです。MQSC コマンドを使用して属性を定義、変更、または表示するときには、代替の短縮名が使用されます。詳細については、[MQSC コマンド](#)を参照してください。

表 666. プロセス定義の属性	
属性	説明
AlterationDate	定義が最後に変更された日付
AlterationTime	定義が最後に変更された時刻
ApplId	アプリケーション ID
ApplType	アプリケーション・タイプ
EnvData	環境データ
ProcessDesc	プロセスの記述
ProcessName	プロセス名
QSGDisp	キュー共有グループ後処理
UserData	ユーザー・データ

AlterationDate (MQCHAR12)

これは、定義を最後に変更した日付です。日付の形式は YYYY-MM-DD で、その後に 2 つの末尾ブランクを付けて長さ 12 バイトになります。

この属性の値を判別するには、MQINQ 呼び出しで MQCA_ALTERATION_DATE セレクターを使用します。この属性の長さは MQ_DATE_LENGTH によって指定されます。

AlterationTime (MQCHAR8)

これは、定義を最後に変更した時刻です。時刻の形式は HH.MM.SS です。

この属性の値を判別するには、MQINQ 呼び出しで MQCA_ALTERATION_TIME セレクターを使用します。この属性の長さは MQ_TIME_LENGTH によって指定されます。

ApplId (MQCHAR256)

これは、開始されるアプリケーションを識別する文字ストリングです。この情報は、開始キュー上のメッセージを処理するトリガー・モニター・アプリケーションが使用するものです。この情報は、トリガー・メッセージの一部として開始キューに送信されます。

ApplId の意味は、トリガー・モニター・アプリケーションによって決まります。IBM MQ によって提供されるトリガー・モニターでは、*ApplId* を実行可能プログラムの名前にする必要があります。以下の注意事項は、指定している特定の環境に適用されます。

- z/OS では、*ApplId* は次のものでなければなりません。
 - CICS トリガー・モニター・トランザクション CKTI を使用して始動されるアプリケーションの場合、CICS トランザクション ID です。
 - IMS トリガー・モニター CSQQTRMN を使用して始動されるアプリケーションの場合、IMS トランザクション ID です。
- Windows の場合、プログラム名の前にディレクトリー・パスを付けることができます。
- AIX and Linux の場合、プログラム名の前にディレクトリー・パスを付けることができます。

文字ストリング内にヌルを入れることはできません。このストリングは、必要に応じて、右側にブランクが埋め込まれます。

この属性の値を判別するには、MQINQ 呼び出しで MQCA_APPL_ID セレクターを使用します。属性の長さは、MQ_PROCESS_APPL_ID_LENGTH で指定します。

ApplType (MQLONG)

アプリケーションタイプは、トリガーメッセージの受信に応じて起動されるプログラムの性質を識別する。この情報は、開始キュー上のメッセージを処理するトリガー監視アプリケーションによって使用されます。この情報はトリガーメッセージの一部として開始キューに送信されます。

ApplType 任意の値を取ることができますが、標準型には以下の値が推奨されます。ユーザー定義アプリケーションタイプを、MQAT_USER_FIRST から MQAT_USER_LAST までの範囲の値に制限する。

MQAT_CICS

CICS トランザクション。

MQAT_DEF

デフォルトのアプリケーション・タイプ。 DEFINE PROCESS (新しいプロセス定義を作成する) を参照してください。

MQAT_OS400

IBM i アプリケーション。

MQAT_UNIX

UNIX アプリケーション。

MQAT_UNKNOWN

不明なアプリケーション・タイプ。

MQAT_USER

ユーザー・アプリケーション。

MQAT_WINDOWS

64 ビット Windows アプリケーション。

MQAT_WINDOWS_NT

32 ビットの Windows アプリケーション。

MQAT_ZOS

z/OS アプリケーション。

MQAT_USER_FIRST

ユーザー定義のアプリケーション・タイプの最低値。

MQAT_USER_LAST

ユーザー定義のアプリケーション・タイプの最高値。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_APPL_TYPE セレクターを使用します。

EnvData (MQCHAR128)

これは、始動するアプリケーションに関する環境関連の情報を含む文字ストリングです。この情報は、開始キュー上のメッセージを処理するトリガー・モニター・アプリケーションが使用するものです。この情報は、トリガー・メッセージの一部として開始キューに送信されます。

EnvData の意味は、トリガー・モニター・アプリケーションによって決まります。IBM MQ によって提供されるトリガー・モニターは、開始されたアプリケーションに渡されるパラメーター・リストに *EnvData* を追加します。パラメーター・リストは、MQTMC2 構造体、1 つのブランク、および末尾ブランクを除去した *EnvData* で構成されます。以下の注意事項は、指定している特定の環境に適用されます。

• On z/OS:

- *EnvData* は、IBM MQ が提供するトリガー・モニター・アプリケーションによって使用されません。
- *ApplType* が MQAT_WLM である場合、作業情報ヘッダー (MQWIH) 内の *ServiceName* フィールドと *ServiceStep* フィールドの *EnvData* にデフォルト値を指定することができます。

- AIX and Linux では、*EnvData* を & 文字に設定すると、起動したアプリケーションをバックグラウンドで実行することができます。

文字ストリング内にヌルを入れることはできません。このストリングは、必要に応じて、右側にブランクが埋め込まれます。

この属性の値を判別するには、MQINQ 呼び出しで MQCA_ENV_DATA セレクターを使用します。属性の長さは、MQ_PROCESS_ENV_DATA_LENGTH で指定します。

ProcessDesc (MQCHAR64)

このフィールドは説明コメントのために使用します。フィールドの内容はキュー・マネージャーにとって重要なものではありませんが、表示できる文字以外は使用しないでください。フィールドにヌル文字を入れることはできません。また、必要に応じて、右側がブランクで埋められます。DBCS をインストール済みの環境では、このフィールドに DBCS 文字を入れることができます (最大フィールド長として 64 バイトが適用されます)。

注: このフィールドに、(CodedCharSetId キュー・マネージャー属性で定義されている) キュー・マネージャーの文字セットに含まれていない文字が含まれている場合、このフィールドが別のキュー・マネージャーに送信されると、それらの文字が正しく変換されない可能性があります。

この属性の値を判別するには、MQINQ 呼び出しで MQCA_PROCESS_DESC セレクターを使用します。

属性の長さは、MQ_PROCESS_DESC_LENGTH で指定します。

ProcessName (MQCHAR48)

これは、ローカル・キュー・マネージャーで定義されるプロセス定義の名前です。

それぞれのプロセス定義には、キュー・マネージャーに所属する他のプロセス定義の名前とは異なる名前があります。しかし、そのプロセス定義の名前は、別のタイプの他のキュー・マネージャー・オブジェクト (キューなど) の名前と同じでも構いません。

この属性の値を判別するには、MQINQ 呼び出しで MQCA_PROCESS_NAME セレクターを使用します。

属性の長さは、MQ_PROCESS_NAME_LENGTH で指定します。

QSGDisp (MQLONG)

プロセス定義の後処理を指定します。値は、次のいずれか 1 つです。

MQQSGD_Q_MGR

オブジェクトにはキュー・マネージャーの後処理が含まれます。オブジェクト定義がローカル・キューにのみ認識されます。この定義は、キュー共有グループの他のキュー・マネージャーには認識されません。

キュー共有グループ内の各キュー・マネージャーは、現行オブジェクトと名前およびタイプが同じオブジェクトを持つことができます。ただし、それらは別のオブジェクトであり、相互に何の関連もありません。それらの属性が互いに同じになるように制約されることはありません。

MQQSGD_COPY

このオブジェクトは、共用リポジトリ内に存在するマスター・オブジェクト定義のローカル・コピーです。キュー共有グループ内の各キュー・マネージャーが、このオブジェクトの独自のコピーを持つことができます。最初はすべてのコピーに同じ属性がありますが、MQSC コマンドを使用することによってそれぞれのコピーを変更でき、その属性を他のコピーのものと異なるようにすることができます。共有リポジトリのマスター定義が更新されると、コピーの属性は再同期化されます。

この属性の値を判別するには、MQINQ 呼び出しで MQIA_QSG_DISP セレクターを使用します。

 この属性は、z/OS でのみサポートされます。

UserData (MQCHAR128)

UserData は、開始されるアプリケーションに関するユーザー情報が入っている文字ストリングです。この情報は、開始キュー上のメッセージを処理するトリガー・モニター・アプリケーション、またはトリガ

ー・モニターによって開始されるアプリケーションが使用するためのものです。この情報は、トリガー・メッセージの一部として開始キューに送信されます。

UserData の意味は、トリガー・モニター・アプリケーションによって決まります。IBM MQ によって提供されるトリガー・モニターは、パラメーター・リストの一部として、始動するアプリケーションに *UserData* を渡します。パラメーター・リストは、MQTMC2 構造体 (*UserData* を含む) と、それに続く 1 つのブランク、および末尾ブランクを除去した *EnvData* で構成されます。

文字ストリング内にヌルを入れることはできません。このストリングは、必要に応じて、右側にブランクが埋め込まれます。Microsoft Windows では、プロセス定義が **runmqtrm** に渡される場合、文字ストリングに二重引用符を含めてはなりません。

この属性の値を判別するには、MQINQ 呼び出しで MQCA_USER_DATA セレクターを使用します。属性の長さは、MQ_PROCESS_USER_DATA_LENGTH で指定します。

戻りコード

IBM MQ Message Queue Interface (MQI) および IBM MQ Administration Interface (MQAI) 呼び出しが行われるたびに、その呼び出しが成功したか失敗したかを示すための完了コードと理由コードが、キュー・マネージャーまたは出口ルーチンによって戻されます。

特に指定がある場合を除いて、エラーが特定の順序で検査されると想定してアプリケーションを作成しないでください。呼び出しによって複数の完了コードまたは理由コードが生じた場合、どのエラーが報告されるかは、実現方法 (キュー・マネージャーか、あるいは出口ルーチンか) によって異なります。

IBM MQ API 呼び出しの後に正常に完了したかどうかを検査するアプリケーションは、完了コードを常に検査する必要があります。完了コードの値は、理由コードの値に基づいて想定しないでください。

完了コード

完了コード・パラメーター (*CompCode*) は、その呼び出しが成功したか、部分的に完了したか、または失敗したかを、呼び出し元で素早く知ることができるようにするためのものです。以下の完了コードのリストには、呼び出しの記述の中で述べた内容よりさらに詳しい記述が示されています。

MQCC_OK

呼び出しはすべて完了しました。すべての出力パラメーターが設定されました。この場合、**Reason** パラメーターの値は常に MQRC_NONE です。

MQCC_WARNING

呼び出しは部分的に完了しました。*CompCode* および *Reason* 出力パラメーターの他にも、いくつかの出力パラメーターが設定されている場合があります。**Reason** パラメーターで部分的な完了についての情報がさらに分かります。

MQCC_FAILED

呼び出しの処理は完了しませんでした。特に示されていない限り、キュー・マネージャーの状態は変わりません。*CompCode* および *Reason* 出力パラメーターが設定されました。その他のパラメーターは (特に断りがない限り) 変更されませんでした。

理由は、アプリケーション・プログラム内の障害である場合や、そのプログラムの外部の状態の結果である場合があります (ユーザーの権限が取り消された場合など)。**Reason** パラメーターでエラーについての情報がさらに分かります。

理由コード

理由コード・パラメーター (*Reason*) は、完了コード・パラメーター (*CompCode*) を修飾します。

特に報告する理由がない場合には、MQRC_NONE が戻ります。呼び出しが成功した場合は、MQCC_OK および MQRC_NONE が返されます。

完了コードが MQCC_WARNING または MQCC_FAILED のいずれかである場合、キュー・マネージャーは常に、それを修飾する理由を報告します。詳細は、各呼び出しの説明で示されています。

ユーザー出口ルーチンが完了コードおよび理由を設定した場合、これらの規則に従う必要があります。また、ユーザー出口により定義される特殊な理由値はゼロ未満にし、キュー・マネージャーにより定義される値と競合しないようにする必要があります。出口ルーチンでは、該当する場合、キュー・マネージャーによって既に定義されている理由を設定することができます。

理由コードは、以下のフィールドにも設定されます。

- MQDLH 構造体の *Reason* フィールド
- MQMD 構造体の *Feedback* フィールド

理由コードの詳細な説明については、[メッセージおよび理由コード](#)を参照してください。

MQI オプションの妥当性検査に関する規則

このセクションでは、MQOPEN、MQPUT、MQPUT1、MQGET、MQCLOSE、またはMQSUB 呼び出しから MQRC_OPTIONS_ERROR 理由コードを生成するシチュエーションをリストします。

MQOPEN 呼び出し

MQOPEN 呼び出しのオプションについては、次の規則が適用されます。

- 以下のいずれかを 1 つ以上指定する必要があります。
 - MQOO_BROWSE
 - MQOO_INPUT_EXCLUSIVE¹
 - MQOO_INPUT_SHARED¹
 - MQOO_INPUT_AS_Q_DEF¹
 - MQOO_INQUIRE
 - MQOO_OUTPUT
 - MQOO_SET
 - MQOO_BIND_ON_OPEN²
 - MQOO_BIND_NOT_FIXED²
 - MQOO_BIND_ON_GROUP²
 - MQOO_BIND_AS_Q_DEF²
- 次のオプションのうち指定できるのは、1 つのみです。
 - MQOO_READ_AHEAD
 - MQOO_NO_READ_AHEAD
 - MQOO_READ_AHEAD_AS_Q_DEF
- 1. 次のオプションのうち指定できるのは、1 つのみです。
 - MQOO_INPUT_EXCLUSIVE
 - MQOO_INPUT_SHARED
 - MQOO_INPUT_AS_Q_DEF
- 2. 次のオプションのうち指定できるのは、1 つのみです。
 - MQOO_BIND_ON_OPEN
 - MQOO_BIND_NOT_FIXED
 - MQOO_BIND_ON_GROUP
 - MQOO_BIND_AS_Q_DEF

注：上にリストされているオプションは相互に排他的です。ただし、MQOO_BIND_AS_Q_DEF の値がゼロのときは、その他の 2 つのバインド・オプションのいずれかと一緒にこのオプションを指定しても、理由コード MQRC_OPTIONS_ERROR は戻されません。MQOO_BIND_AS_Q_DEF は、プログラムの文書化を助けるために指定します。

- MQOO_SAVE_ALL_CONTEXT を指定する場合には、MQOO_INPUT_* オプションのうち、1 つを同時に指定する必要があります。
- MQOO_SET_*_CONTEXT または MQOO_PASS_*_CONTEXT オプションのうちの 1 つを指定する場合には、MQOO_OUTPUT も指定する必要があります。
- MQOO_CO_OP を指定する場合は、MQOO_BROWSE も指定する必要があります。
- MQOO_NO_MULTICAST を指定する場合は、MQOO_OUTPUT も指定する必要があります。

MQPUT 呼び出し

書き込みメッセージ・オプションについては、次の規則が適用されます。

- MQPMO_SYNCPOINT と MQPMO_NO_SYNCPOINT を組み合わせて使用することはできません。
- 次のオプションのうち指定できるのは、1 つのみです。
 - MQPMO_DEFAULT_CONTEXT
 - MQPMO_NO_CONTEXT
 - MQPMO_PASS_ALL_CONTEXT
 - MQPMO_PASS_IDENTITY_CONTEXT
 - MQPMO_SET_ALL_CONTEXT
 - MQPMO_SET_IDENTITY_CONTEXT
- 次のオプションのうち指定できるのは、1 つのみです。
 - MQPMO_ASYNC_RESPONSE
 - MQPMO_SYNC_RESPONSE
 - MQPMO_RESPONSE_AS_TOPIC_DEF
 - MQPMO_RESPONSE_AS_Q_DEF
- MQPMO_ALTERNATE_USER_AUTHORITY は指定できません。これは MQPUT1 呼び出しでのみ有効です。

MQPUT1 呼び出し

書き込みメッセージ・オプションについては、次の点を除いて、MQPUT 呼び出しの場合と同じ規則が適用されます。

- MQPMO_ALTERNATE_USER_AUTHORITY を許可する。
- MQPMO_LOGICAL_ORDER を許可しない。

MQGET 呼び出し

読み取りメッセージ・オプションについては、次の規則が適用されます。

- 次のオプションのうち指定できるのは、1 つのみです。
 - MQGMO_NO_SYNCPOINT
 - MQGMO_SYNCPOINT
 - MQGMO_SYNCPOINT_IF_PERSISTENT
- 次のオプションのうち指定できるのは、1 つのみです。
 - MQGMO_BROWSE_FIRST
 - MQGMO_BROWSE_MSG_UNDER_CURSOR
 - MQGMO_BROWSE_NEXT
 - MQGMO_MSG_UNDER_CURSOR
- MQGMO_SYNCPOINT は、次のオプションと組み合わせて使用することはできません。

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_BROWSE_NEXT
- MQGMO_LOCK
- MQGMO_UNLOCK
- MQGMO_SYNCPOINT_IF_PERSISTENT は次のいずれのオプションとも組み合わせて使用することはできません。
 - MQGMO_BROWSE_FIRST
 - MQGMO_BROWSE_MSG_UNDER_CURSOR
 - MQGMO_BROWSE_NEXT
 - MQGMO_COMPLETE_MSG
 - MQGMO_UNLOCK
- MQGMO_MARK_SKIP_BACKOUT は、MQGMO_SYNCPOINT と組み合わせて使用する必要があります。
- MQGMO_WAIT と MQGMO_SET_SIGNAL を組み合わせて使用することはできません。
- MQGMO_LOCK を指定する場合は、次のいずれか 1 つのオプションと組み合わせて使用する必要があります。
 - MQGMO_BROWSE_FIRST
 - MQGMO_BROWSE_MSG_UNDER_CURSOR
 - MQGMO_BROWSE_NEXT
- MQGMO_UNLOCK を指定する場合には、以下の値のみを使用できます。
 - MQGMO_NO_SYNCPOINT
 - MQGMO_NO_WAIT

MQCLOSE 呼び出し

MQCLOSE 呼び出しのオプションについては、次の規則が適用されます。

- MQCO_DELETE と MQCO_DELETE_PURGE は組み合わせて使用することはできません。
- 次のオプションのうち指定できるのは、1 つのみです。
 - MQCO_KEEP_SUB
 - MQCO_REMOVE_SUB

MQSUB 呼び出し

MQSUB 呼び出しのオプションについては、次の規則が適用されます。

- 以下のいずれかを 1 つ以上指定する必要があります。
 - MQSO_ALTER
 - MQSO_RESUME
 - MQSO_CREATE
- 次のオプションのうち指定できるのは、1 つのみです。
 - MQSO_DURABLE
 - MQSO_NON_DURABLE

注：上にリストされているオプションは相互に排他的です。ただし、MQSO_NON_DURABLE の値がゼロのときは、MQSO_DURABLE と一緒にこのオプションを指定しても、理由コード MQRC_OPTIONS_ERROR は戻されません。MQSO_NON_DURABLE は、プログラムの文書化を支援するために提供されています。

- MQSO_GROUP_SUB と MQSO_MANAGED を組み合わせて使用することはできません。

- MQSO_GROUP_SUB は、MQSO_SET_CORREL_ID と組み合わせて使用する必要があります。
- 次のオプションのうち指定できるのは、1 つのみです。
 - MQSO_ANY_USERID
 - MQSO_FIXED_USERID
- 以下と組み合わせて MQSO_NEW_PUBLICATIONS_ONLY を使用できます:
 - MQSO_CREATE
 - MQSO_ALTER (元のサブスクリプションで MQSO_NEW_PUBLICATIONS_ONLY が設定された場合)
- MQSO_PUBLICATIONS_ON_REQUEST と 1 より大きな値の SubLevel を組み合わせて使用することはできません。
- 次のオプションのうち指定できるのは、1 つのみです。
 - MQSO_WILDCARD_CHAR
 - MQSO_WILDCARD_TOPIC
- MQSO_NO_MULTICAST は、MQSO_MANAGED と組み合わせて使用する必要があります。

キュー型パブリッシュ/サブスクライブ・コマンド・メッセージ

アプリケーションは、MQRFH2 コマンド・メッセージを使用してキュー型パブリッシュ/サブスクライブ・アプリケーションを制御できます。

パブリッシュ/サブスクライブに MQRFH2 を使用しているアプリケーションは、SYSTEM.BROKER.CONTROL.QUEUE に次のコマンド・メッセージを送信できます。

- [874 ページの『Delete Publication メッセージ』](#)
- [875 ページの『Deregister Subscriber メッセージ』](#)
- [879 ページの『Publish メッセージ』](#)
- [881 ページの『Register Subscriber メッセージ』](#)
- [886 ページの『Request Update メッセージ』](#)

キュー型パブリッシュ/サブスクライブ・アプリケーションを作成する場合は、これらのメッセージ、キュー・マネージャーの応答メッセージ、およびメッセージ記述子 (MQMD) について理解する必要があります。次の情報を参照してください。

- [888 ページの『Queue Manager Response メッセージ』](#)
- [894 ページの『パブリケーションをキュー・マネージャーから転送する場合の MQMD の設定』](#)
- [895 ページの『キュー・マネージャーの応答メッセージでの MQMD の設定』](#)
- [890 ページの『パブリッシュ/サブスクライブの理由コード』](#)

コマンドは、MQRFH2 ヘッダーの **NameValueData** フィールド内の psc フォルダーに含まれます。コマンド・メッセージに対する応答としてブローカーによって送信されるメッセージは、pscr フォルダーに含まれています。

各コマンドの説明には、フォルダーに含めることのできるプロパティが示されています。特に明記されていない限り、このプロパティはオプションであり、1 回のみ指定できます。

プロパティの名前は <Command> として表示されます。

値は、ストリング形式でなければなりません (例えば、Publish)。

プロパティの値を表すストリング定数は、小括弧で囲んで示されます (例えば、(MQPSC_PUBLISH))。ストリング定数は、キュー・マネージャーとともに提供されるヘッダー・ファイル cmqpsc.h に定義されます。

Delete Publication メッセージ

Delete Publication コマンド・メッセージは、パブリッシャーからキュー・マネージャーに、またはキュー・マネージャーから別のキュー・マネージャーに送信され、指定されたトピックの保存パブリケーションをすべて削除することをキュー・マネージャーに伝えます。

このメッセージは、キュー・マネージャーのキュー型パブリッシュ/サブスクライブ・インターフェースによってモニターされるキューに送信されます。

入力キューは、元のパブリケーションが送信されたキューである必要があります。

Delete Publication コマンド・メッセージで指定されたすべてのトピックではなく、その一部に対する権限がある場合は、その一部のトピックのみが削除されます。**Broker Response** メッセージで、どのトピックが削除されないかが示されます。

同様に、**Publish** コマンドで複数のトピックが指定されている場合、**Delete Publication** コマンドによって削除されるのは、それらのすべてのトピックではなく、一部と一致する、**Delete Publication** コマンドで指定されたトピックのパブリケーションのみです。

キュー・マネージャーにコマンド・メッセージを送信する場合に必要なメッセージ記述子 (MQMD) パラメーターの詳細については、894 ページの『パブリケーションをキュー・マネージャーから転送する場合の MQMD の設定』を参照してください。

プロパティー

Command (MQPSC_COMMAND)

値は DeletePub(MQPSC_DELETE_PUBLICATION) です。

このプロパティーは必須です。

Topic> (MQPSC_TOPIC)

値は、保存パブリケーションを削除するトピックを含むストリングです。ストリングにワイルドカード文字を使用することで、複数のトピックのパブリケーションを削除できます。

このプロパティーは必須です。必要なトピックの数に合わせて反復できます。

DelOpt (MQPSC_DELETE_OPTION)

削除オプション・プロパティーは、次のいずれかの値を取ることができます。

Local (MQPSC_LOCAL)

パブリッシュの際に Local オプションが指定されたかどうかに関係なく、ローカル・キュー・マネージャー (つまり、このメッセージの送信先のキュー・マネージャー) で、指定トピックのすべての保存パブリケーションが削除されます。

他のキュー・マネージャーのパブリケーションは影響を受けません。

None (MQPSC_NONE)

すべてのオプションは、デフォルト値を取ります。これは、DelOpt プロパティーを省略した場合と同じ結果になります。他のオプションを同時に指定した場合、None は無視されます。

このプロパティーを省略した場合は、デフォルトにより、パブリッシュの際に Local オプションが指定されたかどうかに関係なく、ネットワーク内のすべてのキュー・マネージャーで、指定トピックのすべての保存パブリケーションが削除されます。

例

Delete Publication コマンド・メッセージの NameValueData の例を以下に示します。これは、サンプル・アプリケーションが、ローカル・キュー・マネージャーで、Team1 と Team2 の試合の最新のスコアを含む保存パブリケーションを削除する場合に使用されます。

```
<psc>
  <Command>DeletePub</Command>
  <Topic>Sport/Soccer/State/LatestScore/Team1 Team2</Topic>
```

```
<DelOpt>Local</DelOpt>
</psc>
```

Deregister Subscriber メッセージ

Deregister Subscriber コマンド・メッセージは、サブスクライバーによって、またはサブスクライバーの代理の別のアプリケーションによって、キュー・マネージャーに送信され、所定のパラメーターと一致するメッセージを受信する必要がなくなったことを伝えます。

このメッセージは、SYSTEM.BROKER.CONTROL.QUEUE (キュー・マネージャーの制御キュー) に送信されます。ユーザーは、このキューにメッセージを書き込むために必要な権限を持っている必要があります。

コマンド・メッセージをキュー・マネージャーに送信する場合に必要なメッセージ記述子 (MQMD) パラメーターの詳細については、[パブリケーションをキュー・マネージャーから転送する場合の MQMD の設定](#) を参照してください。

個々のサブスクリプションを登録解除するには、元のサブスクリプションの対応するトピック、サブスクリプション・ポイント、およびフィルター値を指定します。元のサブスクリプションで指定されなかった値がある場合 (つまり、デフォルト値が取られた場合)、サブスクリプションを登録解除するときにもその値を省略する必要があります。

サブスクライバー、またはサブスクライバーのグループのすべてのサブスクリプションを登録解除するには、**DeregAll** オプションを使用します。例えば、サブスクリプション・ポイントと共に (トピックとフィルターは指定せずに) **DeregAll** を指定した場合は、トピックとフィルターに関係なく、指定したサブスクリプション・ポイントにあるサブスクライバーに対するすべてのサブスクリプションが登録解除されます。トピック、フィルター、サブスクリプション・ポイントを組み合わせて指定することもできます。3 つすべてを指定した場合は、1 つのサブスクリプションのみと一致することができ、**DeregAll** オプションは無視されます。

このメッセージは、サブスクリプションを登録したサブスクライバーから送信する必要があります。これは、サブスクライバーのユーザー ID を検査することで確認されます。

サブスクリプションは、システム管理者が MQSC コマンドまたは PCF コマンドを使用して登録解除することもできます。ただし、一時動的キューに登録されたサブスクリプションは、キュー名ではなく、キューに関連付けられます。キューが、明示的に、またはアプリケーションをキュー・マネージャーから切断することによって、削除されると、**Deregister Subscriber** コマンドを使用してそのキューのサブスクリプションを登録解除できなくなります。この場合、サブスクリプションを登録解除するには、デベロッパー・ワークベンチを使用します。サブスクリプションは、次にそのサブスクリプションに一致するパブリケーションがあったとき、または次にキュー・マネージャーが再起動したときに、キュー・マネージャーによって自動的に除去されます。通常の場合は、アプリケーションは、キューを削除する前、またはキュー・マネージャーから切断する前に、サブスクリプションを登録解除します。

サブスクライバーが、サブスクリプションを登録解除することを伝えるメッセージを送信し、その登録解除が正常に処理されたことを伝える応答メッセージを受信しても、サブスクリプションの登録解除と並行してパブリケーションがキュー・マネージャーで処理中だった場合は、まだサブスクライバー・キューにパブリケーションが残っている可能性があります。メッセージをキューから除去しないと、サブスクライバー・キューに未処理のメッセージが蓄積される可能性があります。しばらくスリープした後でアプリケーションが適切な **CorrelId** が指定された **MQGET** 呼び出しを含むループを実行すると、これらのメッセージはキューから消去されます。

同様に、サブスクライバーが永続動的キューを使用している場合は、登録解除し、**MQCLOSE** 呼び出しで **MQCO_DELETE_PURGE** オプションを指定してキューを閉じて、キューが空にならないことがあります。キューを削除するときに、まだコミットされていないパブリケーションがキュー・マネージャーにあると、**MQCLOSE** 呼び出しにより **MQRC_Q_NOT_EMPTY** 戻りコードが出されます。アプリケーションは、スリープした後、**MQCLOSE** 呼び出しを繰り返し実行することで、この問題を回避することができます。

プロパティー

Command (**MQPSC_COMMAND**)

値は **DeregSub (MQPSC_DEREGISTER_SUBSCRIBER)** です。

このプロパティーは必須です。

Topic (MQPSC_TOPIC)

値は、登録解除するトピックを含むストリングです。

複数のトピックを登録解除する場合は、必要に応じてこのプロパティを反復することができます。DeregAll が <RegOpt> に指定されている場合は省略できます。

サブスクライバーが他のトピックのサブスクリプションを保存する必要がある場合、指定するトピックは、登録されているトピックのサブセットでも構いません。ワイルドカード文字を使用することもできますが、ワイルドカード文字を含むトピック・ストリングは、**Deregister Subscriber** コマンド・メッセージで指定された対応するストリングと完全に一致する必要があります。

SubPoint (MQPSC_SUBSCRIPTION_POINT)

値は、サブスクリプションを切り離すサブスクリプション・ポイントを指定するストリングです。

このプロパティは、反復できません。これは、<Topic> が指定されている場合、または <RegOpt> に DeregAll が指定されている場合は省略できます。このプロパティを省略した場合は、次のことが生じます。

- DeregAll を指定しないと、<Topic> プロパティ（さらに、もしあれば <Filter> プロパティ）に一致しているサブスクリプションが、デフォルトのサブスクリプション・ポイントから登録解除されます。
- DeregAll を指定すると、すべてのサブスクリプション・ポイントからすべてのサブスクリプション（<Topic> プロパティと、もしあれば <Filter> プロパティに一致しているもの）が登録解除されます。

デフォルトのサブスクリプション・ポイントを明示的に指定することはできません。したがって、このサブスクリプション・ポイントのみからすべてのサブスクリプションを登録解除することはできません。トピックを指定する必要があります。

SubIdentity (MQPSC_SUBSCRIPTION_IDENTITY)

これは、最大 64 文字の長さの可変長のストリングです。これを使用して、サブスクリプションに関するアプリケーションを指定します。キュー・マネージャーは、各サブスクリプションのサブスクライバー ID セットを保持します。各サブスクリプションの ID セットには、1 つの ID のみを組み込むことも、無限の数の ID を組み込むこともできます。

SubIdentity がサブスクリプションの ID セットに含まれていると、セットから除去されます。この結果、ID セットが空になると、RegOpt プロパティで値として LeaveOnly が指定されている場合を除き、そのサブスクリプションはキュー・マネージャーから除去されます。ID セットにまだ他の ID が含まれている場合は、サブスクリプションはキュー・マネージャーから除去されず、パブリケーション・フローは中断されません。

SubIdentity が指定されているのに、SubIdentity がサブスクリプションの ID セットに含まれていない場合、**Deregister Subscriber** コマンドは失敗し、戻りコード `MQRCCF_SUB_IDENTITY_ERROR` が返されます。

Filter (MQPSC_FILTER)

値は、登録解除するフィルターを指定するストリングです。大/小文字とスペースを含み、既に登録されているサブスクリプション・フィルターと完全に一致する必要があります。

複数のフィルターを登録解除する場合は、必要に応じてこのプロパティを反復できます。これは、<Topic> が指定されているか、<RegOpt> に DeregAll が指定されている場合は省略できます。

サブスクライバーが他のフィルターのサブスクリプションを保存する必要がある場合、指定するフィルターは、登録されているフィルターのサブセットでも構いません。

RegOpt (MQPSC_REGISTRATION_OPTION)

登録オプション・プロパティは、次の値を取ることができます。

DeregAll

(MQPSC_DEREGISTER_ALL)

このサブスクライバーに対して登録されている、一致するすべてのサブスクリプションが登録解除されます。

DeregAll を指定した場合:

- <Topic>、<SubPoint>、さらに <Filter> は省略できます。
- 必要な場合は、<Topic> および <Filter> を繰り返すことができます。
- <SubPoint> を繰り返さないでください。

DeregAll を指定しなかった場合:

- <Topic> を指定する必要がある、必要に応じて反復することができます。
- <SubPoint> および <Filter> は省略できます。
- <SubPoint> を繰り返さないでください。
- 必要な場合は、<Filter> を繰り返すことができます。

トピックとフィルターの両方が反復されると、この 2 つのいずれかの組み合わせに一致するすべてのサブスクリプションが削除されます。例えば、3 つのトピックと 3 つのフィルターを指定する **Deregister Subscriber** コマンドは、9 つのサブスクリプションを削除しようとしています。

CorrelAsId

(MQPSC_CORREL_ID_AS_IDENTITY)

メッセージ記述子 (MQMD) 内の CorrelId には、ゼロ以外の値を指定する必要があります。これを使用して、サブスクライバーを識別します。これは、元のサブスクリプションで使用された CorrelId と一致する必要があります。

FullResp

(MQPSC_FULL_RESPONSE)

FullResp を指定した場合、コマンドが失敗しない限り、応答メッセージでサブスクリプションのすべての属性が返されます。

FullResp を指定した場合、**Deregister Subscriber** コマンドで DeregAll を指定することはできません。また、複数のトピックを指定することもできません。どちらの場合も、コマンドは失敗し、戻りコード **MQRCCF_REG_OPTIONS_ERROR** が返されます。

LeaveOnly

(MQPSC_LEAVE_ONLY)

このプロパティを指定したときに、サブスクリプションの ID セット内にある SubIdentity も指定されていると、SubIdentity は、サブスクリプションの ID セットから除去されます。結果的に ID セットが空になっても、サブスクリプションはキュー・マネージャーから除去されません。SubIdentity 値が ID セットに含まれていない場合、コマンドは失敗し、戻りコード **MQRCCF_SUB_IDENTITY_ERROR** が返されます。

LeaveOnly を指定したときに、SubIdentity が指定されていないと、コマンドは失敗し、戻りコード **MQRCCF_REG_OPTIONS_ERROR** が返されます。

LeaveOnly も SubIdentity も指定されていない場合は、サブスクリプションの ID セットの内容に関係なく、サブスクリプションは除去されます。

なし

(MQPSC_NONE)

すべてのオプションは、デフォルト値を取ります。これは、登録オプション・プロパティを省略した場合と同じ結果になります。他のオプションを同時に指定した場合、None は無視されます。

VariableUserId

(MQPSC_VARIABLE_USER_ID)

このプロパティを指定した場合、サブスクライバーの ID (キュー、キュー・マネージャー、および関連 ID) は 1 つのユーザー ID に限定されなくなります。これは、元の登録メッセージのユーザー ID をサブスクライバーの ID に関連付けて、それ以降、他のユーザーがその ID を使用できない

ようにする、キュー・マネージャーの既存の動作とは異なります。新しいサブスクライバーが同じ ID を使用しようとする、戻りコード `MQRCCF_DUPLICATE_SUBSCRIPTION` が返されます。

どのユーザーでも、適切な権限があれば、サブスクリプションを変更または登録解除することができます。ユーザー ID が元のサブスクライバーの ID と一致していることを確認する既存のチェックを行う必要がありません。

このオプションを既存のサブスクリプションに追加する場合は、元のサブスクリプションと同じユーザー ID でコマンドを実行する必要があります。

登録解除対象のサブスクリプションに `VariableUserId` が設定されている場合は、登録解除時にこのオプションを設定して、どのサブスクリプションを登録解除するかを指定する必要があります。このオプションが設定されていない場合は、**Deregister Subscriber** コマンドのユーザー ID を使用してサブスクリプションを識別することになります。サブスクリプション名が指定されている場合、このオプションは、他のサブスクライバー ID と共にオーバーライドされます。

このプロパティを省略した場合は、デフォルトで、登録オプションは何も設定されません。

QMgrName (MQPSC_Q_MGR_NAME)

値は、サブスクライバー・キューのキュー・マネージャー名です。これは、元のサブスクリプションで使用された `QMgrName` と一致する必要があります。

このプロパティを省略した場合は、デフォルトで、メッセージ記述子 (MQMD) 内の `ReplyToQMgr` 名になります。その名前がブランクである場合は、デフォルトで、キュー・マネージャーの名前になります。

QName (MQPSC_Q_NAME)

値は、サブスクライバー・キューの名前です。これは、元のサブスクリプションで使用された `QName` と一致する必要があります。

このプロパティを省略した場合は、デフォルトで、メッセージ記述子 (MQMD) 内の `ReplyToQ` 名になります (これをブランクにすることはできません)。

SubName (MQPSC_SUBSCRIPTION_NAME)

Deregister Subscriber コマンドで `SubName` を指定した場合、`SubName` 値は、`VariableUserId` がサブスクリプションそのもので設定されている場合を除き、ユーザー ID 以外の他のすべての ID フィールドに優先します。`VariableUserId` が設定されていない場合、**Deregister Subscriber** コマンドは、コマンド・メッセージのユーザー ID がサブスクリプションのユーザー ID と一致する場合のみ、正常に実行されます。設定されている場合、コマンドは失敗し、戻りコード `MQRCCF_DUPLICATE_IDENTITY` が返されます。

このコマンドの従来の ID と一致するサブスクリプションがあるが、そのサブスクリプションが `SubName` を持っていない場合、**Deregister Subscriber** コマンドは失敗し、戻りコード `MQRCCF_SUB_NAME_ERROR` が返されます。`SubName` を持つサブスクリプションを、従来の ID と一致するけれども `SubName` が指定されていないコマンド・メッセージを使用して登録解除しようとした場合、コマンドは正常に実行されます。

SubUserData (MQPSC_SUBSCRIPTION_USER_DATA)

これは、可変長のテキスト・ストリングです。キュー・マネージャーは、この値をサブスクリプションと一緒に保管しますが、この値は、サブスクライバーへのパブリケーションの送信には影響しません。この値を変更する場合は、新しい値で同じサブスクリプションを再登録します。この属性は、アプリケーションが使用するためのものです。

`SubUserData` が存在する場合、`SubUserData` はサブスクリプションのメタトピック情報 (`MQCACF_REG_SUB_USER_DATA`) に返されます。

例

Deregister Subscriber コマンド・メッセージの `NameValueData` の例を以下に示します。この例では、サンプル・アプリケーションが、すべての試合の最新スコアを含むトピックに対するサブスクリプシ

ョンを登録解除します。サブスクライバーの ID (CorrelId など) は、MQMD のデフォルトから取得します。

```
<psc>
  <Command>DeregSub</Command>
  <RegOpt>CorrelAsId</RegOpt>
  <Topic>Sport/Soccer/State/LatestScore/#</Topic>
</psc>
```

Publish メッセージ

Publish コマンド・メッセージは、指定された 1 つ以上のトピックの情報をパブリッシュするために、キューに書き込まれるか、キュー・マネージャーからサブスクライバーに送信されます。

メッセージをキューに書き込む権限、および指定された 1 つ以上のトピックの情報をパブリッシュする権限が必要です。

ユーザーが、すべてではなく、一部のトピックのみの情報をパブリッシュする権限を持っている場合、それらのトピックだけがパブリッシュで使用されます。どのトピックがパブリッシュで使用されないかを示す警告応答が出ます。

一致するサブスクリプションを持つサブスクライバーがある場合、キュー・マネージャーは、対応する **Register Subscriber** コマンド・メッセージで定義されたサブスクライバー・キューに **Publish** メッセージを転送します。

コマンド・メッセージをキュー・マネージャーに送信する場合に必要なメッセージ記述子 (MQMD) パラメーター、およびキュー・マネージャーがパブリケーションをサブスクライバーに転送する場合に使用されるメッセージ記述子 (MQMD) パラメーターの詳細については、[Queue Manager Response メッセージ](#)を参照してください。

キュー・マネージャーは、ローカル・パブリケーションである場合を除き、ネットワーク内の、一致するサブスクリプションを持つその他のキュー・マネージャーに **Publish** メッセージを転送します。

パブリケーション・データがある場合は、メッセージの本体に組み込まれます。このデータは、MQRFH2 ヘッダーの NameValueData フィールド内の <mcd> フォルダーに指定することができます。

プロパティ

Command (MQPSC_COMMAND)

値は Publish(MQPSC_PUBLISH) です。

このプロパティは必須です。

Topic (MQPSC_TOPIC)

値は、このパブリケーションが分類されているトピックを含むストリングです。ワイルドカード文字は使用できません。

トピックを名前リスト SYSTEM.QPUBSUB.QUEUE.NAMELIST に追加する必要があります。このタスクの実行方法についての説明は、[ストリームの追加](#)を参照してください。

このプロパティは必須です。必要に応じて、必要なトピックの数に合わせて反復できます。

SubPoint (MQPSC_SUBSCRIPTION_POINT)

パブリケーションがパブリッシュされるサブスクリプション・ポイント。

WebSphere Event Broker 6.0 では、<SubPoint> プロパティの値は、パブリッシュを処理しているパブリケーション・ノードのサブスクリプション・ポイント属性の値です。

IBM WebSphere MQ 7.0.1 では、<SubPoint> プロパティの値は、サブスクリプション・ポイントの名前と一致していなければなりません。[サブスクリプション・ポイントの追加](#)を参照してください。

PubOpt (MQPSC_PUBLICATION_OPTION)

パブリケーション・オプション・プロパティは、次の値を取ることができます。

RetainPub

(MQPSC_RETAIN_PUB)

キュー・マネージャーは、パブリケーションのコピーを保存します。このオプションが設定されていない場合は、キュー・マネージャーが現在のすべてのサブスクライバーにパブリケーションを送信した時点で、そのパブリケーションは削除されます。

IsRetainedPub

(MQPSC_IS_RETAINED_PUB)

(キュー・マネージャーによってのみ設定できます。) このパブリケーションは、キュー・マネージャーによって保存されています。InformIfRetained オプションを指定してサブスクリプションが登録されている場合、キュー・マネージャーは、このオプションを設定することによって、このパブリケーションが既にパブリッシュ済みで、保存されていることをサブスクライバーに通知します。これは、Register Subscriber または Request Update コマンド・メッセージに対する応答でのみ設定されます。サブスクライバーに直接送信される保存パブリケーションでは、このオプションは設定されません。

local

(MQPSC_LOCAL)

このオプションは、このパブリケーションを他のキュー・マネージャーに送信してはならないことをキュー・マネージャーに伝えます。このキュー・マネージャーに登録されているサブスクライバーはすべて、一致するサブスクリプションがあると、このパブリケーションを受信します。

OtherSubsOnly

(MQPSC_OTHER_SUBS_ONLY)

このオプションでは、パブリッシャーが同じトピックのサブスクライバーでもある場合、会議用アプリケーションの処理が簡素化されます。このオプションは、パブリッシャーがサブスクライバーでもある場合は、一致するサブスクリプションがあっても、そのサブスクライバーにパブリケーションを送信しないことをキュー・マネージャーに伝えます。パブリッシャーのサブスクライバー・キューは、以下のリストに示されているように、その QMgrName、QName、およびオプションの CorrelId で構成されます。

CorrelAsId

(MQPSC_CORREL_ID_AS_IDENTITY)

MQMD 内の CorrelId は (ゼロ以外の値を指定する必要があります)、パブリッシャーがサブスクライバーでもあるアプリケーションでは、パブリッシャーのサブスクライバー・キューに含まれます。

なし

(MQPSC_NONE)

すべてのオプションは、デフォルト値を取ります。これは、パブリケーション・オプション・プロパティーを省略した場合と同じ結果になります。他のオプションを同時に指定した場合、None は無視されます。

追加の <PubOpt> エlementを導入することにより、複数のパブリケーション・オプションを使用できます。

このプロパティーを省略した場合は、デフォルトで、パブリケーション・オプションは何も設定されません。

PubTime (MQPSC_PUBLISH_TIMESTAMP)

値は、パブリッシャー側で設定されたオプションのパブリケーション・タイム・スタンプです。長さは 16 文字で、形式は、次のとおりです。

```
YYYYMMDDHHMSSSTH
```

世界時を使用します。この情報は、サブスクライバーへの送信前に、キュー・マネージャーによる検査を受けません。

SeqNum (MQPSC_SEQUENCE_NUMBER)

値は、パブリッシャー側で設定されたオプションのシーケンス番号です。

この値は、パブリケーションごとに必ず 1 増分します。ただし、この値は、キュー・マネージャーによる検査を受けず、サブスクライバーにそのまま送信されます。

同じトピックのパブリケーションを、相互接続されたさまざまなキュー・マネージャーにパブリッシュする場合は、パブリッシャー側で、シーケンス番号が意味のあるものとなるように設定する必要があります (シーケンス番号を使用する場合)。

QMGrName (MQPSC_Q_MGR_NAME)

値は、パブリッシャーがサブスクライバーでもあるアプリケーションにおける、パブリッシャーのサブスクライバー・キューに対するキュー・マネージャーの名前を含むストリングです (『OtherSubsOnly』を参照してください)。

このプロパティを省略した場合、デフォルトで、メッセージ記述子 (MQMD) 内の ReplyToQMGr 名になります。その名前がブランクである場合は、デフォルトで、キュー・マネージャーの名前になります。

QName (MQPSC_Q_NAME)

値は、パブリッシャーがサブスクライバーでもあるアプリケーションにおける、パブリッシャーのサブスクライバー・キューの名前を含むストリングです (『OtherSubsOnly』を参照してください)。

このプロパティを省略した場合、デフォルトで、メッセージ記述子 (MQMD) 内の ReplyToQ 名になります。ReplyToQ は、OtherSubsOnly が設定されている場合はブランクにできません。

例

Publish コマンド・メッセージの *NameValueData* の例を以下に示します。

最初の例は、サンプル・アプリケーションの試合シミュレーターから送信される、試合が開始されたことを示すパブリケーションの例です。

```
<psc>
  <Command>Publish</Command>
  <Topic>Sport/Soccer/Event/MatchStarted</Topic>
</psc>
```

2 番目の例は、保存パブリケーションの例です。Team1 と Team2 の試合の最後のスコアがパブリッシュされます。

```
<psc>
  <Command>Publish</Command>
  <PubOpt>RetainPub</PubOpt>
  <Topic>Sport/Soccer/State/LatestScore/Team1 Team2</Topic>
</psc>
```

Register Subscriber メッセージ

Register Subscriber コマンド・メッセージは、サブスクライバーからキュー・マネージャー、またはサブスクライバーの代理の別のアプリケーションからキュー・マネージャーに送信され、サブスクリプション・ポイントで 1 つ以上のトピックをサブスクライブする必要があることを伝えます。メッセージ・コンテンツ・フィルターを指定することもできます。

パブリッシュ/サブスクライブ・フィルターの式で、小括弧をネストすると、パフォーマンスが急激に低下します。目安として 6 より大きい深さの小括弧のネストは避けてください。

メッセージは SYSTEM.BROKER.CONTROL.QUEUE (キュー・マネージャーの制御キュー) に送信されます。サブスクリプションでの、トピックに対するアクセス権限 (キュー・マネージャーのシステム管理者によって設定) に加えて、このキューにメッセージを書き込む権限が必要です。

ユーザーが、すべてのトピックではなく、一部のトピックについての権限を持っている場合は、それらの一部のトピックのみが登録されます。警告の応答によって、登録されないトピックが示されます。

キュー・マネージャーにコマンド・メッセージを送信する場合に必要なメッセージ記述子 (MQMD) パラメーターの詳細については、[893 ページの『キュー・マネージャーに送信するコマンド・メッセージ内の MQMD の設定』](#)を参照してください。

キューへの応答が一時動的キューである場合は、キューが閉じたときに、キュー・マネージャーによってサブスクリプションが自動的に登録解除されます。

プロパティー

Command (MQPSC_COMMAND)

値は RegSub (MQPSC_REGISTER_SUBSCRIBER) です。このプロパティーは必須です。

Topic (MQPSC_TOPIC)

サブスクライバーがパブリケーションを受け取りたいと思っているトピック。トピックの一部としてワイルドカード文字を指定することもできます。

MQSC コマンド **display sub** を使用して、この方法で作成されたサブスクリプションを調べると、<Topic> タグの値がサブスクリプションの TOPICSTR プロパティーとして表示されます。

このプロパティーは必須です。必要に応じて、必要なトピックの数に合わせて反復できます。

SubPoint (MQPSC_SUBSCRIPTION_POINT)

値は、サブスクリプションの接続先のサブスクリプション・ポイントです。

このプロパティーを省略した場合は、デフォルトのサブスクリプション・ポイントが使用されます。

WebSphere Event Broker 6.0 では、<SubPoint> プロパティーの値は、サブスクライブされている Publication ノードのサブスクリプション・ポイント属性の値と一致する必要があります。

IBM WebSphere MQ 7.0.1 では、<SubPoint> プロパティーの値は、サブスクリプション・ポイントの名前と一致していなければなりません。[サブスクリプション・ポイントの追加](#)を参照してください。

Filter (MQPSC_FILTER)

値は、パブリケーション・メッセージのコンテンツについてのフィルターとして使用される SQL 式です。指定したトピックについてのパブリケーションがフィルターと一致すると、そのパブリケーションはサブスクライバーに送信されます。このプロパティーは、MQSUB 呼び出しおよび MQOPEN 呼び出しで使用される選択ストリングに対応します。詳しくは、[メッセージの内容の選択](#)を参照してください。

このプロパティーを省略した場合、コンテンツ・フィルタリングは行われません。

RegOpt (MQPSC_REGISTRATION_OPTION)

この登録オプション・プロパティーは、次の値を取ることができます。

AddName

(MQPSC_ADD_NAME)

このオプションを指定した場合、既存のサブスクリプションがサブスクリプションの登録コマンドの従来の ID と一致しても、現在の SubName 値を持たない場合は、このコマンドで指定された SubName がサブスクリプションに追加されます。

AddName を指定した場合、SubName フィールドは必須です。このフィールドを指定しないと、MQRCCF_REG_OPTIONS_ERROR が返されます。

CorrelAsId

(MQPSC_CORREL_ID_AS_IDENTITY)

メッセージ記述子 (MQMD) 内の CorrelId は、一致するパブリケーションをサブスクライバー・キューに送信する場合に使用されます。CorrelId にはゼロ以外の値を指定する必要があります。

FullResp

(MQPSC_FULL_RESPONSE)

このオプションを指定した場合は、コマンドが失敗しない限り、サブスクリプションのすべての属性が応答メッセージに返されます。

FullResp が有効なのは、コマンド・メッセージが 1 つのサブスクリプションを参照している場合のみです。したがって、コマンドで指定できるトピックは 1 つのみです。複数のトピックを指定すると、コマンドは失敗し、戻りコード MQRCCF_REG_OPTIONS_ERROR が返されます。

InformIfRet

(MQPSC_INFORM_IF_RETAINED)

キュー・マネージャーは、**Register Subscriber** または **Request Update** コマンド・メッセージに応答してパブリッシュ・メッセージを送信するときに、パブリケーションが保存されているかどうかをサブスクライバーに通知します。キュー・マネージャーは、メッセージに **IsRetainedPub** パブリケーション・オプションを組み込むことでパブリケーションの保存を行います。

JoinExcl

(MQPSC_JOIN_EXCLUSIVE)

このオプションは、指定された **SubIdentity** がサブスクリプションの ID セットの排他的メンバーとして追加され、そのセットに他の ID を追加できなくなることを示します。

ID が既にセット内の唯一の項目として「共有」で追加されている場合は、そのセットがこの ID による排他ロックに変更されます。それ以外の場合、現在、サブスクリプションの ID セットに他の ID が (共有アクセスで) 含まれていると、コマンドは失敗し、戻りコード **MQRCCF_SUBSCRIPTION_IN_USE** が返されます。

JoinShared

(MQPSC_JOIN_SHARED)

このオプションは、指定された **SubIdentity** がサブスクリプションの ID セットに追加されることを示します。

現在、サブスクリプションに (JoinExcl オプションを使用して) 排他的ロックがかけられている場合は、そのサブスクリプションにロックをかけている ID が、このコマンド・メッセージで指定されている ID と同じである場合を除き、コマンドは失敗し、戻りコード **MQRCCF_SUBSCRIPTION_LOCKED** が返されます。この場合、ロックは自動的に共有ロックに変更されます。

local

(MQPSC_LOCAL)

サブスクリプションはローカルなので、ネットワーク内の他のキュー・マネージャーには配布されません。他のキュー・マネージャーで行われるパブリケーションは、このサブスクライバーで対応するグローバル・サブスクリプションも設定されていない限り、このサブスクライバーには送信されません。

NewPubsOnly

(MQPSC_NEW_PUBS_ONLY)

サブスクリプションの登録時に存在していた保存パブリケーションは、サブスクライバーに送信されません。送信されるのは、新規のパブリケーションのみです。

サブスクライバーが再登録して、このオプションを変更 (設定解除) すれば、既に送信済みのパブリケーションを再度送信することが可能になります。

NoAlter

(MQPSC_NO_ALTER)

既存の一致するサブスクリプションの属性は変更されません。

サブスクリプションが作成中の場合、このオプションは無視されます。その他の指定オプションはすべて、新規のサブスクリプションに適用されます。

SubIdentity で、結合オプションのいずれか (JoinExcl または JoinShared) も指定されている場合は、NoAlter の指定に関係なく、ID は ID セットに追加されます。

なし

(MQPSC_NONE)

登録オプションはすべて、デフォルト値を取ります。

サブスクライバーが既に登録されている場合、そのオプションはデフォルト値にリセットされ (登録オプション・プロパティの省略と同じ影響はないことに注意してください)、サブスクリプションの有効期限は **Register Subscriber** メッセージの MQMD から更新されます。

他の登録オプションを同時に指定した場合、None は無視されます。

NonPers

(MQPSC_NON_PERSISTENT)

このサブスクリプションに一致するパブリケーションは、非永続メッセージとして、サブスクライバーに送信されます。

Pers

(MQPSC_PERSISTENT)

このサブスクリプションに一致するパブリケーションは、永続メッセージとして、サブスクライバーに送信されます。

PersAsPub

(MQPSC_PERSISTENT_AS_PUBLISH)

このサブスクリプションに一致するパブリケーションは、パブリッシャーで指定された持続性で、サブスクライバーに送信されます。これはデフォルトの動作です。

PersAsQueue

(MQPSC_PERSISTENT_AS_Q)

このサブスクリプションに一致するパブリケーションは、サブスクライバー・キューで指定された持続性で、サブスクライバーに送信されます。

PubOnReqOnly

(MQPSC_PUB_ON_REQUEST_ONLY)

キュー・マネージャーは、**Request Update** コマンド・メッセージへの応答の場合を除き、サブスクライバーにパブリケーションを送信しません。

VariableUserId

(MQPSC_VARIABLE_USER_ID)

このプロパティを指定した場合、サブスクライバーの ID (キュー、キュー・マネージャー、および相関 ID) は 1 つのユーザー ID に限定されなくなります。これは、元の登録メッセージのユーザー ID をサブスクライバーの ID に関連付けて、それ以降、他のユーザーがその ID を使用できないようにする、キュー・マネージャーの既存の動作とは異なります。新しいサブスクライバーが同じ ID を使用しようとする、MQRCCF_DUPLICATE_SUBSCRIPTION が返されます。

これにより、どのユーザーでも適切な権限があれば、サブスクリプションを変更したり、登録解除したりできるようになります。したがって、ユーザー ID が元のサブスクライバーのユーザー ID と一致しているかどうかの検査は必要ありません。

このオプションを既存のサブスクリプションに追加する場合は、元のサブスクリプションと同じユーザー ID でコマンドを実行する必要があります。

Request Update コマンドのサブスクリプションで、VariableUserId が設定されている場合は、要求更新時にこのオプションを設定することで、どのサブスクリプションが参照しているかを指定する必要があります。そうしない場合は、**Request Update** コマンドのユーザー ID によって、サブスクリプションが識別されることになります。サブスクリプション名が指定されている場合、このオプションは、他のサブスクライバー ID と共にオーバーライドされます。

このオプション・セットを持たない **Register Subscriber** コマンド・メッセージが、このオプション・セットを持つ既存のサブスクリプションを参照する場合、そのオプションはこのサブスクリプションから削除され、サブスクリプションのユーザー ID は修正されます。同じ ID (キュー、キュー・マネージャー、および相関 ID) を持つサブスクライバーが既に存在しているが、その ID に別のユーザー ID が関連付けられている場合は、コマンドは失敗し、戻りコード MQRCCF_DUPLICATE_IDENTITY が返されます。1 つのサブスクライバー ID に関連付けることのできるユーザー ID は 1 つのみです。

登録オプション・プロパティを省略したときに、サブスクライバーが既に登録されていると、その登録オプションは変更されず、サブスクリプションの有効期限は、**Register Subscriber** メッセージの MQMD から更新されます。

サブスクライバーがまだ登録されていない場合は、すべての登録オプションにデフォルト値が設定されて、新規サブスクリプションが作成されます。

デフォルト値は PersAsPub で、それ以外のオプションは設定されません。

QMgrName (MQPSC_Q_MGR_NAME)

値は、サブスクライバー・キューのキュー・マネージャーの名前です。一致するパブリケーションは、このキュー・マネージャーからサブスクライバー・キューに送信されます。

このプロパティを省略した場合、デフォルトで、メッセージ記述子 (MQMD) 内の ReplyToQMGr 名になります。その名前がブランクである場合は、デフォルトで、キュー・マネージャーの QMgrName に設定されます。

QName (MQPSC_Q_NAME)

値は、キュー・マネージャーから一致するパブリケーションを受信する、サブスクライバー・キューの名前です。

このプロパティを省略した場合、デフォルトで、メッセージ記述子 (MQMD) 内の ReplyToQ 名になります (この場合、ブランクを指定することはできません)。

キューが一時動的キューである場合、パブリケーションの非永続配信 (NonPers) は、<RegOpt> プロパティで指定する必要があります。

キューが一時動的キューである場合は、キューが閉じたときに、キュー・マネージャーによってサブスクリプションが自動的に登録解除されます。

SubName (MQPSC_SUBSCRIPTION_NAME)

これは、特定のサブスクリプションに付けられた名前です。キュー・マネージャー、キュー、および任意の関連 ID の代わりにこの名前を使用して、サブスクリプションを参照することができます。

この **SubName** を持つサブスクリプションが既に存在する場合は、そのサブスクリプションの他のすべての属性 (Topic、QMGrName、QName、CorrelId、UserId、RegOpts、UserSubData、および Expiry) は、新しい Register Subscriber コマンド・メッセージで渡された属性 (指定されている場合) でオーバーライドされます。ただし、**SubName** の使用時に QName フィールドが指定されず、MQMD ヘッダーで ReplyToQ が指定されている場合、サブスクライバー・キューは ReplyToQ に変更されます。

このコマンドの従来の ID と一致するサブスクリプションが既に存在しても、そのサブスクリプションが **SubName** を持っていない場合は、**AddName** オプションが指定されていない限り、コマンドは失敗し、戻りコード *MQRCCF_DUPLICATE_SUBSCRIPTION* が返されます。

別の Register Subscriber コマンドで同じ **SubName** を指定して、既存の名前付きサブスクリプションを変更しようとしたときに、その新しいコマンドの Topic、QMGrName、QName、および CorrelId と一致する、別のサブスクリプションが存在すると、SubName が定義されているかどうかに関係なく、コマンドは失敗し、戻りコード *MQRCCF_DUPLICATE_SUBSCRIPTION* が返されます。これによって、2つのサブスクリプション名が同じサブスクリプションを指すことを防ぎます。

SubIdentity (MQPSC_SUBSCRIPTION_IDENTITY)

このストリングを使用して、サブスクリプションに関心があるアプリケーションを指定します。これは、最大長が 64 文字の可変長の文字ストリングで、オプションです。キュー・マネージャーは、各サブスクリプションのサブスクライバー ID セットを保持します。各サブスクリプションの ID セットには、1つの ID のみを組み込むことも、無限の数の ID を組み込むこともできます (『**JoinShared**』オプションおよび『**JoinExcl**』オプションを参照してください)。

サブスクライブ・コマンドで、**JoinShared** オプションまたは **JoinExcl** オプションを指定したときに、サブスクリプションの ID セットにまだ **SubIdentity** が含まれておらず、既存の ID セットに ID を追加することができれば、つまり、排他的に結合されているサブスクライバーが他にないか、ID セットが空であると、その ID セットに SubIdentity が追加されます。

Register Subscriber コマンドで **SubIdentity** を指定した結果として、サブスクリプションの属性が正常に変更されるのは、SubIdentity がこのサブスクリプションの ID セットの唯一のメンバーであ

る場合に限られます。それ以外の場合は、コマンドは失敗し、戻りコード *MQRCCF_SUBSCRIPTION_IN_USE* が返されます。これにより、他の関係するサブスクライバーに気付かれずにサブスクリプションの属性を変更することはできなくなります。

64 文字を超える文字ストリングを指定すると、コマンドは失敗し、戻りコード *MQRCCF_SUB_IDENTITY_ERROR* が返されます。

SubUserData (MQPSC_SUBSCRIPTION_USER_DATA)

これは、可変長のテキスト・ストリングです。キュー・マネージャーは、この値をサブスクリプションと一緒に保管しますが、この値は、サブスクライバーへのパブリケーションの送信には影響しません。この値を変更する場合は、新しい値で同じサブスクリプションを再登録します。この属性は、アプリケーションが使用するためのものです。

SubUserData が存在する場合、SubUserData はサブスクリプションのメタトピック情報 (*MQCACF_REG_SUB_USER_DATA*) に返されます。

複数の登録オプション値 NonPers, PersAsPub, PersAsQueue, and Pers を指定した場合は、最後の値のみが使用されます。個々のサブスクリプションでこれらのオプションを組み合わせることはできません。

例

以下は、'**Register Subscriber** コマンド・メッセージの NameValueData の例である。サンプル・アプリケーションの結果サービスは、このメッセージを使用して、「パブリッシュとして持続」オプションを設定し、すべての試合の最新スコアを含むトピックのサブスクリプションを登録します。サブスクライバーの ID (CorrelId など) は、MQMD のデフォルトから取得します。

```
<psc>
  <Command>RegSub</Command>
  <RegOpt>PersAsPub</RegOpt>
  <RegOpt>CorrelAsId</RegOpt>
  <Topic>Sport/Soccer/State/LatestScore/#</Topic>
</psc>
```

Request Update メッセージ

Request Update コマンド・メッセージは、サブスクライバーからキュー・マネージャーに送信され、所定の (オプションの) フィルターに一致する、指定トピックおよび指定サブスクリプション・ポイントの現在の保存パブリケーションを要求します。

このメッセージは、*SYSTEM.BROKER.CONTROL.QUEUE* (キュー・マネージャーの制御キュー) に送信されます。このキューへのメッセージの書き込み権限、および更新要求にあるトピックのアクセス権限が必要です。これは、キュー・マネージャーのシステム管理者が設定します。

通常、このコマンドは、サブスクライバーが登録時にオプション PubOnReqOnly を指定した場合に使用されます。キュー・マネージャーに、一致する保存パブリケーションがあると、その保存パブリケーションがサブスクライバーに送信されます。キュー・マネージャーに、一致する保存パブリケーションがない場合は、要求は失敗し、戻りコード *MQRCCF_NO_RETAINED_MSG* が返されます。リクエスターも、同じ Topic 値、SubPoint 値、および Filter 値でサブスクリプションを事前に登録しておく必要があります。

プロパティ

Command (MQPSC_COMMAND)

値は ReqUpdate (*MQPSC_REQUEST_UPDATE*) です。このプロパティは必須です。

Topic (MQPSC_TOPIC)

値は、サブスクライバーが要求しているトピックです。ワイルドカード文字を使用することができません。

このプロパティは指定する必要がありますが、このメッセージで指定できる回数は 1 回のみです。

SubPoint (MQPSC_SUBSCRIPTION_POINT)

値は、サブスクリプションの接続先のサブスクリプション・ポイントです。

このプロパティを省略した場合は、デフォルトのサブスクリプション・ポイントが使用されます。

Filter (MQPSC_FILTER)

値は、パブリケーション・メッセージのコンテンツに対するフィルターとして使用される ESQL 式です。指定したトピックについてのパブリケーションがフィルターと一致すると、そのパブリケーションはサブスクライバーに送信されます。

<Filter> プロパティには、更新を要求している元のサブスクリプションに指定されているものと同じ値を指定する必要があります。

このプロパティを省略した場合、コンテンツ・フィルタリングは行われません。

RegOpt (MQPSC_REGISTRATION_OPTION)

登録オプションのプロパティは、次の値を取ることができます。

CorrelAsId

(MQPSC_CORREL_ID_AS_IDENTITY)

メッセージ記述子 (MQMD) 内の **CorrelId** は、一致するパブリケーションをサブスクライバー・キューに送信する場合に使用されます。これは、ゼロ以外の値でなければなりません。

なし

(MQPSC_NONE)

すべてのオプションは、デフォルト値を取ります。これは、<RegOpt> プロパティを省略するのと同じ効果があります。他のオプションを同時に指定した場合、None は無視されます。

VariableUserId

(MQPSC_VARIABLE_USER_ID)

このプロパティを指定した場合、サブスクライバーの ID (キュー、キュー・マネージャー、および相関 ID) は 1 つのユーザー ID に限定されなくなります。これは、元の登録メッセージのユーザー ID をサブスクライバーの ID に関連付けて、それ以降、他のユーザーがその ID を使用できないようにする、キュー・マネージャーの既存の動作とは異なります。新しいサブスクライバーが同じ ID を使用しようすると、コマンドは失敗し、戻りコード **MQRCCF_DUPLICATE_SUBSCRIPTION** が返されます。

したがって、どのユーザーでも適切な権限があれば、このサブスクリプションを変更したり、登録を解除したりすることができます。したがって、ユーザー ID が元のサブスクライバーのユーザー ID と一致しているかどうかの検査は必要ありません。

このオプションを既存のサブスクリプションに追加する場合は、元のサブスクリプションと同じユーザー ID でコマンドを実行する必要があります。

Request Update コマンドのサブスクリプションで、**VariableUserId** が設定されている場合は、要求更新時にこのオプションを設定することで、どのサブスクリプションが参照しているかを指定する必要があります。そうしない場合は、**Request Update** コマンドのユーザー ID によって、サブスクリプションが識別されることになります。サブスクリプション名が指定されている場合、このオプションは、他のサブスクライバー ID と共にオーバーライドされます。

このプロパティを省略した場合は、デフォルトで、登録オプションは何も設定されません。

QMGrName (MQPSC_Q_MGR_NAME)

値は、キュー・マネージャーから一致する保存アプリケーションを受信する、サブスクライバー・キューのキュー・マネージャーの名前です。

このプロパティを省略した場合、デフォルトで、メッセージ記述子 (MQMD) 内の **ReplyToQMGr** 名になります。その名前がブランクである場合は、デフォルトで、キュー・マネージャーの **QMGrName** に設定されます。

QName (MQPSC_Q_NAME)

値は、キュー・マネージャーから一致する保存パブリケーションを受信する、サブスクライバー・キューの名前です。

このプロパティを省略した場合、デフォルトで、メッセージ記述子 (MQMD) 内の **ReplyToQ** 名になります (この場合、ブランクを指定することはできません)。

SubName (MQPSC_SUBSCRIPTION_NAME)

これは、特定のサブスクリプションに付けられた名前です。 **Request Update** コマンドでこのプロパティを指定した場合、SubName 値は、VariableUserId がサブスクリプションそのもので設定されている場合を除き、ユーザー ID 以外の他のすべての ID フィールドに優先します。VariableUserId が設定されていない場合に、*Request Update* コマンドが成功するのは、コマンド・メッセージのユーザー ID がサブスクリプションのユーザー ID と一致する場合のみです。コマンド・メッセージのユーザー ID がサブスクリプションのユーザー ID と一致しないと、コマンドは失敗し、戻りコード *MQRCCF_DUPLICATE_IDENTITY* が返されます。

VariableUserId が設定され、ユーザー ID がサブスクリプションのユーザー ID と異なる場合に、コマンドが成功するのは、新しいコマンド・メッセージのユーザー ID に、ストリーム・キューを参照する権限と、サブスクリプションのサブスクライバー・キューに書き込む権限がある場合です。それ以外の場合、コマンドは失敗し、戻りコード *MQRCCF_NOT_AUTHORIZED* が返されます。

このコマンドの従来の ID と一致するサブスクリプションが存在しても、そのサブスクリプションが SubName を持っていない場合、**Request Update** コマンドは失敗し、戻りコード *MQRCCF_SUB_NAME_ERROR* が返されます。

SubName を持つサブスクリプションの更新要求を、従来の ID とは一致するが SubName が指定されていないコマンド・メッセージを使用して行った場合、コマンドは成功します。

例

Request Update コマンド・メッセージの NameValueData の例を以下に示します。 サンプル・アプリケーションの結果サービスは、このメッセージを使用して、すべての試合の最新スコアを含んだ保存パブリケーションを要求します。 サブスクライバーの ID (CorrelId など) は、MQMD のデフォルトから取得します。

```
<psc>
  <Command>ReqUpdate</Command>
  <RegOpt>CorrelAsId</RegOpt>
  <Topic>Sport/Soccer/State/LatestScore/#</Topic>
</psc>
```

Queue Manager Response メッセージ

Queue Manager Response メッセージは、キュー・マネージャーからパブリッシャーまたはサブスクライバーの ReplyToQ に送信され、コマンド・メッセージ記述子で応答が必須であることが指定されている場合に、キュー・マネージャーが受信したコマンド・メッセージの成功または応答を伝えます。

応答メッセージは、MQRFH2 ヘッダーの NameValueData フィールド内の <pscr> フォルダーに含まれています。

警告やエラーの場合、応答メッセージにはコマンド・メッセージの <psc> フォルダーに加え、<pscr> フォルダーが含まれます。メッセージ・データは、キュー・マネージャーの応答メッセージには含まれません。エラーの場合、エラーが発生したメッセージは何も処理されていません。警告の場合は、メッセージの一部が正常に処理されていることがあります。

応答を送信する際にエラーが発生した場合は、次のようになります。

- パブリケーション・メッセージの場合、MQPUT が失敗すると、キュー・マネージャーは、応答を IBM MQ 送達不能キューに送信することを試みます。これにより、応答をパブリッシャーに送り返すことができない場合でも、パブリケーションをサブスクライバーに送信できるようになります。
- その他のメッセージの場合、またはパブリケーションの応答を送達不能キューに送信できなかった場合は、エラーがログに記録され、通常、コマンド・メッセージはロールバックされます。これが行われるかどうかは、MQInput ノードの構成によって決まります。

プロパティ

Completion (MQPSCR_COMPLETION)

完了コードです。次の 3 つの値のいずれかを取ることができます。

OK

コマンドが正常に完了しました。

警告

コマンドが完了しましたが、警告が生成されました。

error

コマンドが失敗しました。

Response (MQPSCR_RESPONSE)

コマンドによって生成された完了コードが **warning** または **error** だった場合の、そのコマンド・メッセージへの応答。このプロパティには、「<Reason>」プロパティが含まれており、警告またはエラーの原因を示すその他のプロパティが含まれている場合があります。

エラーが 1 つ以上ある場合、応答フォルダーは 1 つのみで、最初のエラーの原因のみが示されます。警告が 1 つ以上ある場合は、警告ごとに応答フォルダーがあります。

Reason (MQPSCR_REASON)

完了コードが **warning** または **error** の場合に、完了コードを限定する理由コード。これは、以下の例に示されているエラー・コードのいずれかに設定されます。「<Reason>」プロパティは、<Response> フォルダー内に含まれています。理由コードの後には、<psc> フォルダーからの任意の有効なプロパティ (例えば、トピック名) を続けることができ、エラーまたは警告の原因を示しています。理由コード???? を受け取った場合は、データが正しいかどうかを確認します。例えば、不等号括弧 (<>) が一致しているかどうかを確認します。

例

Queue Manager Response メッセージの NameValueData の例を以下に示します。以下は、成功応答と見なすことができます。

```
<pscr>
  <Completion>ok</Completion>
</pscr>
```

障害応答の例を以下に示します。この場合の障害はフィルター・エラーです。最初の NameValueData ストリングには応答が入り、2 番目のストリングには元のコマンドが入ります。

```
<pscr>
  <Completion>error</Completion>
  <Response>
    <Reason>3150</Reason>
  </Reponse>
</pscr>

<psc>
  ...
  command message (to which
  the queue manager is responding)
  ...
</psc>
```

次に、警告応答の例を示します (原因はトピックの権限がないことです)。最初の NameValueData ストリングには応答が入ります。2 番目の NameValueData ストリングには元のコマンドが入ります。

```
<pscr>
  <Completion>warning</Completion>
  <Response>
    <Reason>3081</Reason>
    <Topic>topic1</Topic>
  </Reponse>
  <Response>
    <Reason>3081</Reason>
    <Topic>topic2</Topic>
  </Reponse>
</pscr>

<psc>
  ...
  command message (to which
  the queue manager is responding)
```

</psc>

パブリッシュ/サブスクライブの理由コード

次に、パブリッシュ/サブスクライブ応答用の <psc> フォルダの Reason フィールドに返される可能性のある理由コードを示します。また、C または C++ プログラミング言語でこれらのコードを表す場合に使用できる定数も示します。

MQRC_ 定数は、IBM MQ cmqc.h ヘッダー・ファイルを必要とします。MQRCCF_ 定数は IBM MQ cmqcfh.h ヘッダー・ファイルを必要とします。(cmqpsc.h ヘッダー・ファイルを必要とする MQRCCF_FILTER_ERROR と MQRCCF_WRONG_USER 以外)

理由コードとテキスト	説明	発行元
2336 MQRC_RFH_COMMAND_ERROR	<psc> フォルダの <Command> フィールドに有効な値は以下のとおりです：RegSub, DeregSub, Publish, DeletePub, ReqUpdate。その他の値が指定された場合、このエラー・コードが発行されます。	すべてのコマンド
2337 MQRC_RFH_PARM_ERROR	<psc> と <mcd> フォルダの両方に、それらの中で指定できる有効なパラメーター・セットがあります。これらのフォルダの記述を調べて、誤ったパラメーターを指定していないことを確認してください。	すべてのコマンド
2338 MQRC_RFH_DUPLICATE_PARM	<psc> フォルダ内の一部のパラメーター（例えば、Topic）を繰り返すことができますが、他のパラメーター（例えば、Command）を繰り返すことはできません。反復できないパラメーターを繰り返して指定していないことを確認してください。	すべてのコマンド
2339 MQRC_RFH_PARM_MISSING	<psc> または <mcd> フォルダ内の一部のパラメーターはオプションであり、省略することができます。ただし一部は必須であり、省略することはできません。<psc> フォルダおよび <mcd> フォルダ内にすべての必須パラメーターが含まれていることを確認してください。	すべてのコマンド
2551 MQRC_SELECTION_NOT_AVAILABLE	フィルターが指定されたサブスクライバーから、パブリケーションを受信するサブスクライバーを判別するために使用可能な拡張メッセージ選択プロバイダーが存在しませんでした。	Publish、Register Subscriber、および Request Update
	指定されたサブスクライバーのフィルターを処理するために使用可能な拡張メッセージ選択プロバイダーが存在しませんでした。	Register Subscriber および Request Update

理由コードとテキスト	説明	発行元
2554 MQRC_CONTENT_ERROR	拡張メッセージ選択プロバイダーが、現在のパブリケーションまたは保存パブリケーションでエラーを検出しました。	Publish および Request Update
3008 MQRCCF_COMMAND_FAILED	内部エラーが発生し、コマンドを正常に実行できませんでした。このエラーは、コマンドが再発行された場合に発生することがあります。キュー・マネージャーのシステム・イベント・ログに、IBM に問題を報告する際に必要な情報が含まれています。	すべてのコマンド
3072 MQRCCF_TOPIC_ERROR	Topic パラメーターに指定した値の 1 つ以上が誤っています。Topic の値が、指定された制限事項に準拠していることを確認してください。	すべてのコマンド
3073 MQRCCF_NOT_REGISTERED	DeregSub コマンドまたは ReqUpdate コマンドで指定した SubPoint、Topic、Filter の組み合わせが、以前に登録したときの組み合わせではないか、または DeregSub コマンドで DeregAll オプションが指定された場合に、SubPoint、Topic、または Filter プロパティーの 1 つがサブスクリプションの登録解除に使用されませんでした。	Deregister Subscriber コマンドおよび Request Update コマンド
3074 MQRCCF_Q_MGR_NAME_ERROR	指定されたキュー・マネージャーが有効でないか、またはキュー・マネージャーが使用できないか存在しませんでした。	Deregister Subscriber コマンド、Publish コマンド、Register Subscriber コマンド、および Request Update コマンド
3076 MQRCCF_Q_NAME_ERROR	指定されたキュー名が有効でないか、または指定されたキュー・マネージャーにそのキューが存在しませんでした。	Deregister Subscriber コマンド、Publish コマンド、Register Subscriber コマンド、および Request Update コマンド
3077 MQRCCF_NO_RETAINED_MSG	指定したトピックに対する保存メッセージがありませんでした。これは、アプリケーション・プログラムの設計によっては、エラーにならないこともあります。	Request Update コマンド

理由コードとテキスト	説明	発行元
3079 MQRCCF_INCORRECT_Q	RegSub コマンド、DeregSub コマンド、および ReqUpdate コマンドは、対象となるキュー・マネージャーの SYSTEM.BROKER.CONTROL.QUEUE キューに常に送信されます。Publish コマンドおよび Delete Publication コマンドは、対象となる特定のパブリッシュ/サブスクライブ・メッセージ・フローの入力キューに送信されます。これは、メッセージ・フローの設計時に決定されます。このエラー・コードは、コマンドが誤ったキューに送信された場合に返されます。	すべてのコマンド
3080 MQRCCF_CORREL_ID_ERROR	RegOpt パラメーターの 1 つとして CorrelAsId が指定されています。しかし、MQMD の CorrelId フィールドに、有効な相関 ID が含まれていません (つまり、MQCI_NONE に設定されています)。	Deregister Subscriber コマンドおよび Register Subscriber コマンド
3081 MQRCCF_NOT_AUTHORIZED	要求されたアクションを実行する権限がありません。キュー・マネージャーの権限の設定は、システム管理者が「トピック階層」エディターを使用して処理します。	Publish コマンドおよび Register Subscriber コマンド
3083 MQRCCF_REG_OPTIONS_ERROR	RegSub コマンドまたは DeregSub コマンドを含む <psc> フォルダーに、認識されない RegOpt パラメーターが指定されました。	Deregister Subscriber コマンドおよび Register Subscriber コマンド
3084 MQRCCF_PUB_OPTIONS_ERROR	Publish コマンドが含まれている <psc> フォルダーに、認識されない PubOpt パラメーターが指定されました。	Publish コマンド
3087 MQRCCF_DEL_OPTIONS_ERROR	DeletePub コマンドを含む <psc> フォルダーに、認識されない DelOpt パラメーターが指定されました。	Delete Publication コマンド
3150 MQRCCF_FILTER_ERROR	Filter パラメーターに指定した値が無効です。フィルター式の有効な構文が記述されているセクションを調べて、式が準拠していることを確認してください。	Deregister Subscriber コマンド、Register Subscriber コマンド、および Request Update コマンド
3151 MQRCCF_WRONG_USER	指定されたサブスクリプションと一致するサブスクリプションが既に存在しますが、これは別のユーザーによって登録されたものです。サブスクリプションの変更または登録解除を行えるのは、最初のそのサブスクリプションを登録したユーザーのみです。	Deregister Subscriber コマンド、Register Subscriber コマンド、および Request Update コマンド
3152 MQRCCF_DUPLICATE_SUBSCRIPTION	サブスクリプション名が異なる、一致するサブスクリプションが既に存在します。	

理由コードとテキスト	説明	発行元
3153 MQRCCF_SUB_NAME_ERROR	サブスクリプション名の形式が有効ではないか、またはサブスクリプション名を持たない一致するサブスクリプションが既に存在します。	
3154 MQRCCF_SUB_IDENTITY_ERROR	サブスクリプション ID パラメーターでエラーが検出されました。指定された値が許容される最大長を超えているか、またはサブスクリプション ID が現在、サブスクリプション ID セットのメンバーではなく、Join 登録オプションが指定されていません。	
3155 MQRCCF_SUBSCRIPTION_IN_USE	ID セットのメンバーがサブスクリプションを変更または登録解除しようとしたときに、このセットにはそのメンバー以外のメンバーも含まれていました。	
3156 MQRCCF_SUBSCRIPTION_LOCKED	現在、別の ID によってサブスクリプションに排他ロックがかけられています。	
3157 MQRCCF_ALREADY_JOINED	Join 登録オプションが指定されましたが、そのサブスクライバー ID は、既にサブスクリプションの ID セットのメンバーでした。	

キュー・マネージャーに送信するコマンド・メッセージ内の MQMD の設定

キュー・マネージャーにコマンド・メッセージを送信するアプリケーションは、メッセージ記述子 (MQMD) で次のフィールド設定を使用します。デフォルト値のままにするフィールドや、通常は任意の有効な値に設定できるフィールドは、ここでは取り上げません。

レポート

『MsgType』および『CorrelId』を参照してください。

MsgType

MsgType には *MQMT_REQUEST* または *MQMT_DATAGRAM* のいずれかを設定する必要があります。

MsgType にこれらの値のいずれも設定されていない場合、*MQRC_MSG_TYPE_ERROR* が返されます。

常に応答を必要とするコマンド・メッセージの場合、MsgType は、*MQMT_REQUEST* に設定する必要があります。Report フィールドの *MQRO_PAN* フラグおよび *MQRO_NAN* フラグは、ここでは重要ではありません。

MsgType が *MQMT_DATAGRAM* に設定されている場合、応答は、Report フィールドの *MQRO_PAN* フラグおよび *MQRO_NAN* フラグの設定によって異なります。

- *MQRO_PAN* フラグのみの場合、キュー・マネージャーはコマンドが成功した場合のみ応答を送信します。
- *MQRO_NAN* フラグのみの場合、キュー・マネージャーはコマンドが失敗した場合のみ応答を送信します。
- コマンドの完了時に警告が発行された場合、*MQRO_PAN* または *MQRO_NAN* のいずれが設定されていれば、応答は送信されます。
- *MQRO_PAN* フラグと *MQRO_NAN* フラグの両方が設定されている場合、キュー・マネージャーは、コマンドの成功、失敗に関係なく、応答を送信します。この場合、キュー・マネージャーの観点では、MsgType が *MQMT_REQUEST* に設定されている場合と同じ結果になります。
- *MQRO_PAN* も *MQRO_NAN* も設定されていない場合、応答は送信されません。

形式

MQFMT_RF_HEADER_2 に設定します。

MsgId

通常、このフィールドは MQMI_NONE に設定されるので、キュー・マネージャーは固有値を生成します。

CorrelId

このフィールドは、任意の値に設定できます。送信側の ID に CorrelId が含まれている場合は、Report フィールドに MQRO_PASS_CORREL_ID を指定するだけでなく、この値も指定します。これで、キュー・マネージャーから受信側に送信されるすべての応答メッセージで CorrelID が設定されます。

ReplyToQ

このフィールドでは、応答がある場合にその応答の送信先のキューを定義します。これは、送信側のキューでも構いません。この場合は、メッセージから QName パラメーターを省略できるというメリットがあります。ただし、応答が別のキューに送信される場合は、QName パラメーターが必要です。

ReplyToQMGr

このフィールドでは、応答用のキュー・マネージャーを定義します。このフィールドをブランク (デフォルト値) にすると、ローカル・キュー・マネージャーはこのフィールドに自分の名前を挿入します。

パブリケーションをキュー・マネージャーから転送する場合の MQMD の設定

キュー・マネージャーはパブリケーションをサブスクライバーに送信する場合、メッセージ記述子 (MQMD) で次のフィールド設定を使用します。MQMD 内のその他のフィールドはすべて、デフォルト値に設定されます。

レポート

Report は MQRO_NONE に設定されます。

MsgType

MsgType は MQMT_DATAGRAM に設定されます。

Expiry

Expiry は、パブリッシャーから受信した Publish メッセージの値に設定されます。保存メッセージの場合は、メッセージがキュー・マネージャーのところにどまっていた概算時間に応じて、残り時間が少なくなります。

形式

Format は MQFMT_RF_HEADER_2 に設定されます。

MsgId

MsgId は固有値に設定されます。

CorrelId

サブスクライバーの ID に CorrelId が含まれている場合は、これが、登録時にサブスクライバーによって指定される値になります。それ以外の場合は、キュー・マネージャーによって選択されたゼロ以外の値になります。

priority

Priority は、パブリッシャーによって設定された値を取るか、またはパブリッシャーが MQPRI_PRIORITY_AS_Q_DEF を指定した場合は解決される値を取ります。

永続性

Persistence は、このパブリケーションの送信先のサブスクライバーの Register Subscriber メッセージで特に指定されていない限り、パブリッシャーによって設定された値を取るか、パブリッシャーが MQPER_PERSISTENCE_AS_Q_DEF を指定した場合に解決される値を取ります。

ReplyToQ

ReplyToQ はブランクに設定されます。

ReplyToQMGr

ReplyToQMGr は、キュー・マネージャーの名前に設定されます。

UserIdentifier

UserIdentifier は、サブスクライバーの登録時に設定される、サブスクライバーのユーザー ID です。

AccountingToken

AccountingToken は、サブスクライバーの最初の登録時に設定される、サブスクライバーの会計トークンです。

ApplIdentityData

ApplIdentityData は、サブスクライバーの最初の登録時に設定される、サブスクライバーのアプリケーション ID データです。

PutApplType

PutApplType は MQAT_BROKER に設定されます。

PutApplName

PutApplName は、キュー・マネージャーの名前の先頭の 28 文字に設定されます。

PutDate

PutDate は、メッセージがサブスクライバのキューに入れられた日付です。

PutTime

PutTime は、メッセージがサブスクライバのキューに入れられた時間です。

ApplOriginData

ApplOriginData はブランクに設定されます。

キュー・マネージャーの応答メッセージでの MQMD の設定

キュー・マネージャーは、パブリケーション・メッセージへの応答を送信する場合に、メッセージ記述子 (MQMD) で次のフィールド設定を使用します。MQMD 内のその他のフィールドはすべて、デフォルト値に設定されます。

レポート

Report はすべてゼロに設定されます。

MsgType

MsgType は MQMT_REPLY に設定されます。

形式

Format は MQFMT_RF_HEADER_2 に設定されます。

MsgId

MsgId の設定は、元のコマンド・メッセージの Report オプションによって異なります。デフォルトでは、MQMI_NONE に設定されるので、キュー・マネージャーは固有値を生成します。

CorrelId

CorrelId の設定は、元のコマンド・メッセージの Report オプションによって異なります。デフォルトでは、CorrelId がコマンド・メッセージの MsgId と同じ値に設定されます。この値は、コマンドと応答の相関関係を示すために使用できます。

priority

Priority は、元のコマンド・メッセージと同じ値に設定されます。

永続性

Persistence は、元のコマンド・メッセージと同じ値に設定されます。

Expiry

Expiry は、キュー・マネージャーが受信した元のコマンド・メッセージと同じ値に設定されます。

PutApplType

PutApplType は MQAT_BROKER に設定されます。

PutApplName

PutApplName は、キュー・マネージャーの名前の先頭の 28 文字に設定されます。

その他のコンテキスト・フィールドは、MQPMO_PASS_IDENTITY_CONTEXT で生成された場合と同じように設定されます。

マシン・エンコード

このセクションでは、メッセージ記述子の *Encoding* フィールドの構造について説明します。

構造体のフィールドの要約については、[385 ページの『MQMD - メッセージ記述子』](#)を参照してください。

Encoding フィールドは 32 ビットの整数で、4 つの別個のサブフィールドに分割されています。それらのサブフィールドは、それぞれ次に示すものを表しています。

- 2 進整数用のエンコード
- パック 10 進整数用のエンコード
- 浮動小数点用のエンコード
- 予約ビット

各サブフィールドは、サブフィールドに対応する位置に 1 のビット、それ以外の位置に 0 のビットを持つビット・マスクによって識別されます。これらのビットには、ビット 0 が最上位ビットになり、ビット 31 が最下位ビットになるように番号が付けられています。定義されているマスクは次のとおりです。

MQENC_INTEGER_MASK

2 進整数エンコードをマスクします。

このサブフィールドは、*Encoding* フィールド内のビット位置 28 から 31 を占めています。

MQENC_DECIMAL_MASK

パック 10 進整数エンコードをマスクします。

このサブフィールドは、*Encoding* フィールド内のビット位置 24 から 27 を占めています。

MQENC_FLOAT_MASK

浮動小数点エンコードをマスクします。

このサブフィールドは、*Encoding* フィールド内のビット位置 20 から 23 を占めています。

MQENC_RESERVED_MASK

予約済みビットをマスクします。

このサブフィールドは、*Encoding* フィールド内のビット位置 0 から 19 を占めています。

2 進整数のエンコード

2 進整数のエンコードとして有効な値は次のとおりです。

MQENC_INTEGER_UNDEFINED

2 進整数は、定義されていないエンコードを用いて表されます。

MQENC_INTEGER_NORMAL

2 進整数は標準的な方法で表されます。

- 数値内の最下位バイトは、その数値内のバイトの中で最上位のアドレスを持っています。逆に、最上位バイトは最下位のアドレスを持っています。
- 各バイト内の最下位ビットは、その次に上位のアドレスを持つバイトに隣接しています。各バイト内の最上位ビットは、その次に下位のアドレスを持つバイトに隣接しています。

MQENC_INTEGER_REVERSED

2 進整数は、MQENC_INTEGER_NORMAL と同じ方法で表されますが、バイトの配列順序は逆になります。各バイト内のビットは、MQENC_INTEGER_NORMAL と同じ方法で配列されます。

パック 10 進整数のエンコード

パック 10 進整数のエンコードとして有効な値は次のとおりです。

MQENC_DECIMAL_UNDEFINED

パック 10 進数は、定義されていないエンコードを用いて表されます。

MQENC_DECIMAL_NORMAL

パック 10 進整数は標準的な方法で表されます。

- 10 進数の印刷可能形式の各桁は、パック 10 進数では X'0' から X'9' までの 1 個の 16 進数で表現されます。16 進数の各桁はそれぞれ 4 ビットを占有します。したがって、パック 10 進数の各バイトは、印刷可能な数値形式では 2 桁の 10 進数字を表します。
- パック 10 進数内の最下位バイトは、最下位の 10 進数字が入っているバイトです。そのバイトの中で最上位の 4 ビットには最下位の 10 進数字が入っており、最下位の 4 ビットには符号が入っています。符号は、X'C' (正)、X'D' (負)、あるいは X'F' (無符号) です。
- 数値内の最下位バイトは、その数値内のバイトの中で最上位のアドレスを持っています。逆に、最上位バイトは最下位のアドレスを持っています。
- 各バイト内の最下位ビットは、その次に上位のアドレスを持つバイトに隣接しています。各バイト内の最上位ビットは、その次に下位のアドレスを持つバイトに隣接しています。

MQENC_DECIMAL_REVERSED

パック 10 進整数は、MQENC_DECIMAL_NORMAL と同じ方法で表されますが、バイトの配列順序は逆になります。各バイト内のビットは、MQENC_DECIMAL_NORMAL と同じ方法で配列されます。

浮動小数点エンコード

浮動小数点のエンコードとして有効な値は次のとおりです。

MQENC_FLOAT_UNDEFINED

浮動小数点数は、定義されていないエンコードを用いて表されます。

MQENC_FLOAT_IEEE_NORMAL

浮動小数点数は標準 IEEE を使用して表されます。³ 浮動小数点形式で、バイトの配置は以下の通りです。

- 小数部の中の最下位バイトは、その数値内のすべてのバイトのうちで最上位アドレスを持っています。指数が入っているバイトは最下位アドレスを持っています。
- 各バイト内の最下位ビットは、その次に上位のアドレスを持つバイトに隣接しています。各バイト内の最上位ビットは、その次に下位のアドレスを持つバイトに隣接しています。

IEEE 浮動小数点のエンコードについての詳細は、IEEE 標準 754 で参照できます。

MQENC_FLOAT_IEEE_REVERSED

浮動小数点数は、MQENC_FLOAT_IEEE_NORMAL と同じ方法で表されますが、バイトの配列順序は逆になります。各バイト内のビットは、MQENC_FLOAT_IEEE_NORMAL と同じ方法で配列されます。

MQENC_FLOAT_S390

浮動小数点数は、標準の System/390 浮動小数点形式を用いて表されます。これは、System/370 でも使用されます。

エンコードの組み立て

MQMD の *Encoding* フィールドの値を構築するには、必要なエンコードを記述する関連する定数を加算できます (複数回同じ定数を追加しないでください)。または、ビット単位 OR 演算を使って結合することもできます (プログラミング言語でビット演算がサポートされる場合)。

いずれの方法を使用する場合でも、結合するのは MQENC_INTEGER_* エンコードのうち 1 つと MQENC_DECIMAL_* エンコードのうち 1 つ、および MQENC_FLOAT_* エンコードのうち 1 つのみです。

エンコードの分析

Encoding フィールドには、サブフィールドが入っています。このため、整数のエンコード、パック 10 進数のエンコード、または浮動小数点のエンコードを検査する必要のあるアプリケーションでは、説明されている技法の 1 つを用いる必要があります。

³ 米国電気電子学会

ビット演算の使用

プログラミング言語がビット演算をサポートしている場合は、以下のステップを実行してください。

1. 必須のエンコードのタイプに応じて、以下の値のいずれか 1 つを選択します。
 - バイナリー整数エンコード方式では、MQENC_INTEGER_MASK
 - パック 10 進数エンコード方式では、MQENC_DECIMAL_MASK
 - 浮動小数点エンコード方式では、MQENC_FLOAT_MASK値を A とします。
2. ビット単位の AND 演算を使用して、*Encoding* フィールドと A を結合します。その結果を B とします。
3. B は必須のエンコードであり、そのエンコードのタイプに有効な値のそれぞれについて値が等しいかどうかをテストできます。

算術演算の使用

プログラミング言語がビット演算をサポートしていない場合は、整数の算術演算を使用して、以下のステップを実行してください。

1. 必須のエンコードのタイプに応じて、以下の値のいずれか 1 つを選択します。
 - バイナリー整数エンコード方式では、1
 - パック 10 進数エンコード方式では、16
 - 浮動小数点エンコード方式では、256値を A とします。
2. *Encoding* フィールドの値を A で割り、その結果を B と呼ぶことにします。
3. B を 16 で割り、その結果を C とします。
4. C に 16 を掛けて、B から引きます。その結果を D とします。
5. D に A を掛け、その結果を E とします。
6. E は必須のエンコードであり、そのエンコードのタイプに有効な値のそれぞれについて値が等しいかどうかをテストできます。

マシン・アーキテクチャー・エンコードの要約

マシン・アーキテクチャーのエンコードについては、[898 ページの表 667](#) に示されています。

表 667. マシン・アーキテクチャーのエンコードの要約			
マシン・アーキテクチャー	2 進整数のエンコード	パック 10 進整数のエンコード	浮動小数点エンコード
IBM i	normal	normal	通常 IEEE
Intel x86	逆方向	逆方向	逆方向 IEEE
PowerPC®	normal	normal	通常 IEEE
System/390	normal	normal	System/390

レポート・オプションおよびメッセージ・フラグ

このセクションでは、*Report* フィールドおよび *MsgFlags* フィールドについて説明します。これらは、MQGET、MQPUT、および MQPUT1 呼び出しで指定されるメッセージ記述子 MQMD の一部です。

このセクションでは、以下のトピックについて説明します。

- レポート・フィールドの構造と、キュー・マネージャーがレポート・フィールドを処理する方法

- アプリケーションがレポート・フィールドを分析する方法
- メッセージ・フラグ・フィールドの構造

MQMD メッセージ記述子の詳細については、[385 ページの『MQMD - メッセージ記述子』](#)を参照してください。

レポート・フィールドの構造

この資料では、レポート・フィールドの構造について説明します。

Report フィールドは 32 ビット長の整数で、別々の 3 つのサブフィールドに分かれています。これらのサブフィールドは、次のオプションを識別します。

- ローカル・キュー・マネージャーによって認識されない場合に拒否されるレポート・オプション
- ローカル・キュー・マネージャーによって認識されない場合でも、常に受け入れられるレポート・オプション
- 他の特定の条件を満たした場合にのみ受け入れられるレポート・オプション

各サブフィールドは、サブフィールドに対応する位置に 1 のビット、それ以外の位置に 0 のビットを持つビット・マスクによって識別されます。サブフィールド内のビットは、必ずしも隣接している必要はありません。これらのビットには、ビット 0 が最上位ビットになり、ビット 31 が最下位ビットになるように番号が付けられています。サブフィールドを識別するために定義されているマスクは次のとおりです。

MQRO_REJECT_UNSUP_MASK

このマスクは、*Report* フィールド内のビット位置を識別します。このフィールドには、レポート・オプションがローカル・キュー・マネージャーによってサポートされない場合、MQPUT または MQPUT1 呼び出しが失敗し、完了コード MQCC_FAILED、理由コード MQRC_REPORT_OPTIONS_ERROR が戻ることが示されています。

このサブフィールドは、ビット位置 3、およびビット位置 11 から 13 までを占めます。

MQRO_ACCEPT_UNSUP_MASK

このマスクは、*Report* フィールド内のビット位置を識別します。このフィールドには、レポート・オプションがローカル・キュー・マネージャーによってサポートされない場合でも、MQPUT または MQPUT1 呼び出しで受け入れられることが示されています。この場合は、完了コード MQCC_WARNING および理由コード MQRC_UNKNOWN_REPORT_OPTION が戻ります。

このサブフィールドは、ビット位置 0 から 2 まで、4 から 10 まで、および 24 から 31 までを占めます。

このサブフィールドには、以下のレポート・オプションがあります。

- MQRO_ACTIVITY
- MQRO_COPY_MSG_ID_TO_CORREL_ID
- MQRO_DEAD_LETTER_Q
- MQRO_DISCARD_MSG
- MQRO_EXCEPTION
- MQRO_EXCEPTION_WITH_DATA
- MQRO_EXCEPTION_WITH_FULL_DATA
- MQRO_EXPIRATION
- MQRO_EXPIRATION_WITH_DATA
- MQRO_EXPIRATION_WITH_FULL_DATA
- MQRO_NAN
- MQRO_NEW_MSG_ID
- MQRO_NONE
- MQRO_PAN
- MQRO_PASS_CORREL_ID

- MQRO_PASS_MSG_ID

MQRO_ACCEPT_UNSUP_IF_XMIT_MASK

このマスクは、*Report* フィールド内のビット位置を識別します。このフィールドのビット位置には、ローカル・キュー・マネージャーによってサポートされない場合でも、以下の 2 つの条件が満たされる場合に限り、MQPUT または MQPUT1 呼び出しでレポート・オプションが受け入れられることが示されています。

- メッセージの宛先がリモート・キュー・マネージャーである。
- アプリケーションは、メッセージをローカル伝送キューに直接書き込みません (つまり、MQOPEN または MQPUT1 呼び出しで指定されたオブジェクト記述子の *ObjectQMgrName* および *ObjectName* フィールドで識別されるキューは、ローカル伝送キューではありません)。

これらの条件が満たされた場合は、完了コード MQCC_WARNING、および理由コード MQRC_UNKNOWN_REPORT_OPTION が戻ります。満たされない場合は、完了コード MQCC_FAILED、および理由コード MQRC_REPORT_OPTIONS_ERROR が戻ります。

このサブフィールドは、ビット位置 14 から 23 を占めます。

このサブフィールドには、以下のレポート・オプションがあります。

- MQRO_COA
- MQRO_COA_WITH_DATA
- MQRO_COA_WITH_FULL_DATA
- MQRO_COD
- MQRO_COD_WITH_DATA
- MQRO_COD_WITH_FULL_DATA

Report フィールドにキュー・マネージャーが認識していないオプションが指定されると、キュー・マネージャーはビット単位の AND 演算を用いて各サブフィールドを検査し、*Report* フィールドとそのサブフィールド用のマスクを結合します。この演算の結果が 0 でない場合は、上記の完了コードと理由コードが戻ります。

MQCC_WARNING が戻された場合は、ほかの警告条件があると、どのような理由コードが戻されるかは分かりません。

レポート・オプションは、ローカル・キュー・マネージャーで認識されない場合でも指定でき、受け入れることができます。これは、ローカル・キュー・マネージャーでは認識されないが、リモート・キュー・マネージャーでは認識され、処理できるようなレポート・オプションでメッセージを送信する場合に役立ちます。

レポート・フィールドの分析

Report フィールドには、サブフィールドが含まれます。したがって、メッセージの送信側が特定の報告を要求したかどうかのチェックを必要とするアプリケーションは、説明されている技法のいずれかを使用する必要があります。

ビット演算の使用

プログラミング言語がビット演算をサポートしている場合は、以下のステップを実行してください。

1. 検査するレポートのタイプに応じて、以下の値のいずれかを選択します。

- COA レポートでは MQRO_COA_WITH_FULL_DATA
- COD レポートでは MQRO_COD_WITH_FULL_DATA
- 例外レポートでは MQRO_EXCEPTION_WITH_FULL_DATA
- 例外レポートでは MQRO_EXPIRATION_WITH_FULL_DATA

値を A とします。

z/OS では、MQRO*_WITH_FULL_DATA 値の代わりに、MQRO*_WITH_DATA 値を使用します。

2. ビット単位の AND 演算を使用して、*Report* フィールドと A を結合します。その結果を B とします。
3. B が、レポートのタイプとして可能性のあるそれぞれの値と等しいかどうかをテストします。

例えば、A が MQRO_EXCEPTION_WITH_FULL_DATA であれば、B が以下のそれぞれの値と等しいかどうかをテストして、メッセージの送信側の指定内容を判別します。

- MQRO_NONE
- MQRO_EXCEPTION
- MQRO_EXCEPTION_WITH_DATA
- MQRO_EXCEPTION_WITH_FULL_DATA

テストは、アプリケーション論理として最適であれば、どのような順序で行っても構いません。

同様のメソッドを使用して、MQRO_PASS_MSG_ID または MQRO_PASS_CORREL_ID オプションに対してテストを行います。この 2 つの定数から該当する方を値 A として選択してから、上記の処理を進めてください。

算術演算の使用

プログラミング言語がビット演算をサポートしていない場合は、整数の算術演算を使用して、以下のステップを実行してください。

1. 検査するレポートのタイプに応じて、以下の値のいずれかを選択します。

- COA レポートでは MQRO_COA
- COD レポートでは MQRO_COD
- 例外レポートでは MQRO_EXCEPTION
- 例外レポートでは MQRO_EXPIRATION

値を A とします。

2. *Report* フィールドの値を A で割ります。その結果を B と呼ぶことにします。
3. B を 8 で割り、その結果を C とします。
4. C に 8 を掛けて、B から引きます。その結果を D とします。
5. D に A を掛け、その結果を E とします。
6. E が、レポートのタイプとして可能性のあるそれぞれの値と等しいかどうかをテストします。

例えば、A が MQRO_EXCEPTION であれば、E が以下のそれぞれの値と等しいかどうかをテストして、メッセージの送信側の指定内容を判別します。

- MQRO_NONE
- MQRO_EXCEPTION
- MQRO_EXCEPTION_WITH_DATA
- MQRO_EXCEPTION_WITH_FULL_DATA

テストは、アプリケーション論理として最適であれば、どのような順序で行っても構いません。

以下の疑似コードは、この技法を例外レポート・メッセージに対して使用する場合を示しています。

```
A = MQRO_EXCEPTION
B = Report/A
C = B/8
D = B - C*8
E = D*A
```

同様の方法を使用して、MQRO_PASS_MSG_ID オプションまたは MQRO_PASS_CORREL_ID オプションについてテストします。この 2 つの定数のどちらか適切な方を値 A として選択してから、上記の処理を進めてください。ただし、前の手順の値 8 は値 2 に置き換えてください。

メッセージ・フラグ・フィールドの構造

ここでは、メッセージ・フラグ・フィールドの構造について説明します。

MsgFlags フィールドは 32 ビット長の整数で、別々の 3 つのサブフィールドに分かれています。これらのサブフィールドは、次のオプションを識別します。

- ローカル・キュー・マネージャーによって認識されない場合に拒否されるメッセージ・フラグ
- ローカル・キュー・マネージャーによって認識されない場合でも、常に受け入れられるメッセージ・フラグ
- 他の特定の条件が満たされた場合にのみ受け入れられるメッセージ・フラグ

注: *MsgFlags* のサブフィールドはすべてキュー・マネージャーが使用するために予約済みです。

各サブフィールドは、サブフィールドに対応する位置に 1 のビット、それ以外の位置に 0 のビットを持つビット・マスクによって識別されます。これらのビットには、ビット 0 が最上位ビットになり、ビット 31 が最下位ビットになるように番号が付けられています。サブフィールドを識別するために定義されているマスクは次のとおりです。

MQMF_REJECT_UNSUP_MASK

このマスクは、*MsgFlags* フィールド内のビット位置を識別します。このフィールドでは、ローカル・キュー・マネージャーがサポートしないメッセージ・フラグによって、MQPUT 呼び出しまたは MQPUT1 呼び出しが失敗し、完了コード MQCC_FAILED および理由コード MQRC_MSG_FLAGS_ERROR 戻ります。

このサブフィールドは、ビット位置 20 から 31 までを占めます。

このサブフィールドには、以下のメッセージ・フラグがあります。

- MQMF_LAST_MSG_IN_GROUP
- MQMF_LAST_SEGMENT
- MQMF_MSG_IN_GROUP
- MQMF_SEGMENT
- MQMF_SEGMENTATION_ALLOWED
- MQMF_SEGMENTATION_INHIBITED

MQMF_ACCEPT_UNSUP_MASK

このマスクは、*MsgFlags* フィールド内のビット位置を識別します。このフィールドでは、ローカル・キュー・マネージャーがメッセージ・フラグをサポートしないにもかかわらず、メッセージ・フラグは MQPUT 呼び出しまたは MQPUT1 呼び出しで受け入れられることが示されています。この完了コードは MQCC_OK です。

このサブフィールドは、ビット位置 0 から 11 を占めます。

MQMF_ACCEPT_UNSUP_IF_XMIT_MASK

このマスクは、*MsgFlags* フィールド内のビット位置を識別します。このフィールドのビット位置には、ローカル・キュー・マネージャーによってサポートされない場合でも、以下の 2 つの条件が満たされる場合に限り、MQPUT または MQPUT1 呼び出しでメッセージ・フラグが受け入れられることが示されています。

- メッセージの宛先がリモート・キュー・マネージャーである。
- アプリケーションは、メッセージをローカル伝送キューに直接書き込みません (つまり、MQOPEN または MQPUT1 呼び出しで指定されたオブジェクト記述子の *ObjectQMgrName* および *ObjectName* フィールドで識別されるキューは、ローカル伝送キューではありません)。

上記の条件を 2 つとも満たせば完了コード MQCC_OK が戻りますが、満たさなければ完了コード MQCC_FAILED が理由コード MQRC_MSG_FLAGS_ERROR と共に戻ります。

このサブフィールドは、ビット位置 12 から 19 を占めます。

MsgFlags フィールドにキュー・マネージャーが認識しないフラグが指定されている場合、キュー・マネージャーはビット単位の AND 演算を使用して各サブフィールドを順に検査し、*MsgFlags* フィールドとそ

のサブフィールドのマスクを結合します。この演算の結果が 0 でない場合は、上記の完了コードと理由コードが戻ります。

データ変換出口

この一連のトピックでは、データ変換出口へのインターフェース、およびデータ変換が必要な場合にキュー・マネージャーが行う処理について説明します。

データ変換について詳しくは、「*Data Conversion under IBM MQ*」(<https://www.ibm.com/support/pages/node/317869>)を参照してください。

データ変換出口は、アプリケーション・メッセージ・データを受信側アプリケーションで必須の表示に変換するために、MQGET 呼び出し処理の一部として呼び出されます。アプリケーション・メッセージ・データの変換はオプションです。変換を行う場合は、MQGET 呼び出しに MQGMO_CONVERT オプションを指定する必要があります。

以下の事柄について説明されています。

- MQGMO_CONVERT オプションに応答して、キュー・マネージャーが行う処理。[903 ページの『変換処理』](#)を参照してください。
- 組み込み形式を処理する際にキュー・マネージャーが使用する処理規則。これらの規則はユーザー作成出口に対しても推奨されます。[904 ページの『処理規則』](#)を参照。
- レポート・メッセージを変換するための特別な考慮事項。[908 ページの『レポート・メッセージの変換』](#)を参照してください。
- データ変換出口に渡すパラメーター。[921 ページの『MQ_DATA_CONV_EXIT - データ変換出口』](#)を参照してください。
- 異なる表示間で文字データを変換するために、出口から使用できる呼び出し。[915 ページの『MQXCNCV - 文字の変換』](#)を参照してください。
- 出口に特有のデータ構造のパラメーター。[909 ページの『MQDXP - データ変換出口パラメーター』](#)を参照してください。

変換処理

ここでは、MQGMO_CONVERT オプションへの応答として、キュー・マネージャーが実行する処理について説明します。

MQGMO_CONVERT オプションが MQGET 呼び出しで指定されている場合、およびアプリケーションに戻るメッセージがある場合、キュー・マネージャーは以下の処理を行います。

1. 次の条件を 1 つでも満たせば、変換は不要です。

- メッセージ・データはすでに、MQGET 呼び出しを発行したアプリケーションに必要な文字セットとエンコードで記述されています。アプリケーションは、呼び出しを発行する前に、MQGET 呼び出しの **MsgDesc** パラメーターの *CodedCharSetId* および *Encoding* フィールドを、必要な値に設定する必要があります。
- メッセージ・データの長さがゼロ。
- MQGET 呼び出しの **Buffer** パラメーターの値がゼロ。

このような場合、メッセージは MQGET 呼び出しを発行したアプリケーションに変換されずに戻されます。**MsgDesc** パラメーターの *CodedCharSetId* 値および *Encoding* 値は、メッセージ内の制御情報の値に設定され、呼び出しは以下のいずれかの完了コードと理由コードの組み合わせで完了します。

表 668. 完了コードと理由コードの組み合わせ

完了コード	理由コード
MQCC_OK	MQRC_NONE
MQCC_WARNING	MQRC_TRUNCATED_MSG_ACCEPTED
MQCC_WARNING	MQRC_TRUNCATED_MSG_FAILED

以下のステップは、メッセージ・データの文字セットまたはエンコードが、**MsgDesc** パラメーターにある対応する値と異なり、変換対象のデータがある場合にのみ実行されます。

2. メッセージ内の制御情報にある *Format* フィールドに値 MQFMT_NONE がある場合、メッセージは変換されずに戻り、完了コード MQCC_WARNING と理由コード MQRC_FORMAT_ERROR が戻ります。

これ以外の場合には、処理は継続されます。

3. メッセージはキューから削除され、**Buffer** パラメーターと同じサイズの一時バッファに置かれます。ブラウズ操作の場合、メッセージはキューからは削除されずに、一時バッファにコピーされます。
4. メッセージをバッファのサイズに合わせるために切り捨てる場合は、以下の処理が行われます。

- MQGMO_ACCEPT_TRUNCATED_MSG オプションを指定していない場合、メッセージは変換されずに戻り、完了コード MQCC_WARNING と理由コード MQRC_TRUNCATED_MSG_FAILED が戻る。
- MQGMO_ACCEPT_TRUNCATED_MSG オプションを指定した場合は、完了コードは MQCC_WARNING に、理由コードは MQRC_TRUNCATED_MSG_ACCEPTED に設定され、変換処理は続行される。

5. メッセージを切り捨てずにバッファに収容できる場合、または MQGMO_ACCEPT_TRUNCATED_MSG オプションを指定した場合は、以下の処理が行われます。

- 形式が組み込み形式の場合、バッファはキュー・マネージャーのデータ変換サービスに渡される。
- 形式が組み込み形式ではない場合、バッファは形式と同じ名前をもつユーザー作成出口に渡される。出口が検出できない場合、メッセージは変換されずに戻り、完了コード MQCC_WARNING と理由コード MQRC_FORMAT_ERROR が戻る。

エラーが発生しない場合は、データ変換サービスからの出力またはユーザー作成出口からの出力はメッセージに変換され、さらに完了コードおよび理由コードが MQGET 呼び出しを発行するアプリケーションに戻ります。

6. 変換が成功した場合は、キュー・マネージャーは変換したメッセージをアプリケーションに戻します。この場合、MQGET 呼び出しが戻す完了コードおよび理由コードは、以下の組み合わせのいずれかです。

表 669. 完了コードと理由コードの組み合わせ

完了コード	理由コード
MQCC_OK	MQRC_NONE
MQCC_WARNING	MQRC_TRUNCATED_MSG_ACCEPTED

ただし、変換がユーザー作成出口によって行われた場合、変換が正常に終了しても、これ以外の理由コードが戻される場合があります。

変換が失敗した場合、キュー・マネージャーは、変換されていないメッセージをアプリケーションに戻します。**MsgDesc** パラメーターの *CodedCharSetId* フィールドと *Encoding* フィールドは、メッセージ内の制御情報の値に設定され、完了コード MQCC_WARNING が示されます。

処理規則

組み込みフォーマットを変換する場合、キュー・マネージャーは記述された処理規則に従います。

ユーザー作成出口も下記の規則に従う必要がありますが、これはキュー・マネージャーによって強制されるわけではありません。キュー・マネージャーが変換する組み込み形式は、以下のとおりです。

- MQFMT_ADMIN
- MQFMT_CICS (z/OS のみ)
- MQFMT_COMMAND_1
- MQFMT_COMMAND_2
- MQFMT_DEAD_LETTER_HEADER
- MQFMT_DIST_HEADER
- MQFMT_EVENT バージョン 1
- MQFMT_EVENT バージョン 2

- MQFMT_IMS
- MQFMT_IMS_VAR_STRING
- MQFMT_MD_EXTENSION
- MQFMT_PCF
- MQFMT_REF_MSG_HEADER
- MQFMT_RF_HEADER
- MQFMT_RF_HEADER_2
- MQFMT_STRING
- MQFMT_TRIGGER
- MQFMT_WORK_INFO_HEADER (z/OS のみ)
- MQFMT_XMIT_Q_HEADER

1. メッセージが変換中に拡張し、**Buffer** パラメーターのサイズを超える場合は、次の処理が行われます。

- MQGMO_ACCEPT_TRUNCATED_MSG オプションを指定していない場合、メッセージは変換されずに戻り、完了コード MQCC_WARNING と理由コード MQRC_CONVERTED_MSG_TOO_BIG が戻る。
- MQGMO_ACCEPT_TRUNCATED_MSG オプションを指定した場合は、メッセージは切り捨てられ、完了コードは MQCC_WARNING に、理由コードは MQRC_TRUNCATED_MSG_ACCEPTED に設定されて、変換処理は続行される。

2. 切り捨てが発生した場合は、それが変換の前でも変換中でも、**Buffer** パラメーターに戻る有効なバイト数がバッファの長さより短い可能性があります。

例えば、これは 4 バイトの整数または DBCS 文字がバッファの端にまたがってしまう場合などに発生します。情報の要素が不完全な場合は変換されず、戻ったメッセージ中の未変換部分のバイトには有効な情報が入っていません。これは、変換前に切り捨てられたメッセージが変換中に縮小した場合にも発生します。

戻った有効なバイト数がバッファの長さより短い場合、バッファの末尾の未使用バイトはヌルに設定されます。

3. 配列またはストリングがバッファの端にまたがっている場合、データは最大限変換されます。つまり、特殊な配列エレメントまたは不完全な DBCS 文字だけが変換されず、先行する配列エレメントまたは文字は変換されます。

4. 切り捨てが発生した場合は、それが変換の前でも変換中でも、**DataLength** パラメーターに戻る長さは、切り捨て前の未変換のメッセージの長さになります。

5. ストリングが 1 バイト文字セット (SBCS)、2 バイト文字セット (DBCS)、マルチバイト文字セット (MBCS) の間で変換された場合、ストリングは拡張されるか縮小されることがあります。

- MQFMT_ADMIN、MQFMT_EVENT、および MQFMT_PCF の PCF 形式では、MQCFST 構造体と MQCFSL 構造体の中のストリングは、変換後のストリングを格納するため、必要に応じて拡張されるか縮小されます。

ストリング・リスト構造体 MQCFSL の場合、リスト内のストリングは拡張されるか縮小される場合があります、その量はさまざまです。そのような処理が行われた場合、キュー・マネージャーは変換後に、短いストリングにブランクを埋め込み、最も長いストリングと同じ長さにします。

- MQFMT_REF_MSG_HEADER 形式では、SrcEnvOffset、SrcNameOffset、DestEnvOffset、および DestNameOffset フィールドによってアドレッシングされるストリングは、変換後のストリングを格納するため、必要に応じて拡張されるか縮小されます。
- MQFMT_RF_HEADER 形式では、NameValueString フィールドは、変換後の名前と値の組みを格納するため、必要に応じて拡張されるか縮小されます。
- 固定フィールド・サイズの構造体では、キュー・マネージャーは重要な情報が失われなければ、固定フィールド内でストリングの拡張または縮小を許可します。この関係から、フィールド内の末尾ブランクと最初のヌル文字以降の文字は、重要でないものとして扱われます。

- スtringが拡張された場合、ただし、重要でない文字だけを廃棄して変換後のStringをフィールドに格納する必要がある場合には、変換は成功し、MQCC_OK および理由コード MQRC_NONE (他にエラーがないと見なされる) で呼び出しが完了します。
 - Stringが拡張されたが、変換後のStringをフィールドに収納するために有効な文字を廃棄する必要がある場合は、メッセージが戻されて変換は行われず、MQCC_WARNING および理由コード MQRC_CONVERTED_STRING_TOO_BIG で呼び出しが完了します。
- 注: MQGMO_ACCEPT_TRUNCATED_MSG オプションを指定したかどうかにかかわらず、この場合は理由コード MQRC_CONVERTED_STRING_TOO_BIG が結果となります。
- Stringが縮小された場合、キュー・マネージャーはそのフィールドの長さまでStringにブランクを埋め込みます。
6. 1つ以上の MQ ヘッダー構造体とそれにユーザー・データが続くメッセージについては、1つ以上のヘッダー構造体は変換されるが、残りのメッセージは変換されないということがあり得ます。ただし、各ヘッダー構造体の *CodedCharSetId* フィールドと *Encoding* フィールドは、ヘッダー構造体の後に続くデータの文字セットとエンコードを常に正しく示します (ただし、2つの例外があります)。
- 2つの例外は MQCIH および MQIIH 構造体です。これらの構造体の *CodedCharSetId* および *Encoding* フィールドの値は重要ではありません。これらの構造体については、構造体に続くデータは、MQCIH または MQIIH 構造体と、文字セットとエンコードは同じです。
7. 取り出されるメッセージの制御情報の *CodedCharSetId* または *Encoding* フィールド、あるいは **MsgDesc** パラメーターに、未定義またはサポートされていない値が指定されている場合、未定義またはサポートされていない値をメッセージの変換に使用する必要がなければ、キュー・マネージャーはエラーを無視する可能性があります。
- 例えば、メッセージ内の *Encoding* フィールドがサポートされていない浮動小数点エンコードを指定したが、メッセージには整数データしか入っていない場合、または、浮動小数点データは入っているが変換不要の場合 (ソース浮動小数点エンコードとターゲット浮動小数点エンコードが同一の場合)、エラーの診断は行われません。
- エラーが診断されると、メッセージは変換されずに戻され、完了コード MQCC_WARNING と、MQRC_SOURCE_*_ERROR または MQRC_TARGET_*_ERROR 理由コードのいずれか (該当する場合) が戻されます。 **MsgDesc** パラメーターの *CodedCharSetId* および *Encoding* フィールドは、メッセージ内の制御情報の値に設定されます。
- エラーが診断されず、変換が正常に完了した場合、**MsgDesc** パラメーターの *CodedCharSetId* および *Encoding* フィールドに返される値は、MQGET 呼び出しを発行するアプリケーションによって指定された値です。
8. いずれの場合も、未変換のメッセージがアプリケーションに戻されると、完了コードは MQCC_WARNING に設定され、**MsgDesc** パラメーターの *CodedCharSetId* および *Encoding* フィールドは未変換のデータに該当する値に設定されます。これは MQFMT_NONE の場合にも適用されます。
- Reason** パラメーターは、変換が行われなかった理由を表示するコードに設定されます。ただし、メッセージも切り捨てる必要がある場合は除きます。切り捨てに関する理由コードは、変換に関する理由コードより先に表示されます。(切り捨てられたメッセージが変換されたかどうかを判別するには、**MsgDesc** パラメーターの *CodedCharSetId* および *Encoding* フィールドに返された値を確認します。)
- エラーの診断が行われると、特定の理由コードが戻るか、または一般的な理由コード MQRC_NOT_CONVERTED が戻ります。戻る理由コードは、データ変換サービスに基づいた診断機能によって決まります。
9. 完了コード MQCC_WARNING が戻った場合、また関連のある理由コードが複数ある場合は、コードの順序は次のようになります。
- a. 以下の理由コードはすべての理由コードに優先します。ただし、戻るコードは1つのみです。
 - MQRC_SIGNAL_REQUEST_ACCEPTED
 - MQRC_TRUNCATED_MSG_ACCEPTED

- b. 残りの理由コードの優先順位は定義されていません。
10. MQGET 呼び出し完了時の理由コードの説明は、次のとおりです。
- 次の理由コードは、メッセージが正常に変換されたことを示しています。
 - MQRC_NONE
 - 以下の理由コードは、メッセージが正常に変換された可能性があることを示しています (**MsgDesc** パラメーターの *CodedCharSetId* および *Encoding* フィールドを調べて確認してください)。
 - MQRC_MSG_MARKED_BROWSE_CO_OP
 - MQRC_TRUNCATED_MSG_ACCEPTED
 - その他の理由コードはすべて、メッセージが変換されなかったことを表します。
- 以下の処理は組み込み形式に特有の処理で、ユーザー定義の形式には適用されません。

11. 以下の形式は例外です。

- MQFMT_ADMIN
- MQFMT_COMMAND_1
- MQFMT_COMMAND_2
- MQFMT_EVENT
- MQFMT_IMS_VAR_STRING
- MQFMT_PCF
- MQFMT_STRING

いずれの組み込み形式も、キュー名の中で有効な文字について SBCS 文字を備えていない文字セットとの間では、変換はできません。そのような変換を試みた場合、メッセージは変換されずに戻され、完了コード MQCC_WARNING と、MQRC_SOURCE_CCSID_ERROR または MQRC_TARGET_CCSID_ERROR の理由コードが戻されます。

UNICODE 文字セット UTF-16 は、キュー名の中で有効な文字について SBCS 文字を備えていない文字セットの 1 つの例です。

12. 組み込み形式用のメッセージ・データが所定の長さまで切り捨てられた場合、ストリングの長さや、エレメントあるいは構造体の数が入ったメッセージ内のフィールドは、調整されません。したがって、これらフィールドは、実際にアプリケーションに戻るデータの長さを反映しません。メッセージ・データ内のこのようなフィールドに戻る値は、切り捨て前のメッセージでの値です。
- メッセージ (所定の長さまで切り捨てられた MQFMT_ADMIN メッセージなど) の処理中は、戻されたデータの端を超える部分のデータをアプリケーションからアクセスすることがないようにしてください。

13. 形式の名前が MQFMT_DEAD_LETTER_HEADER の場合、メッセージ・データは MQDLH 構造体で始まり、このあとにおそらく 0 バイト以上のアプリケーション・メッセージ・データが続きます。アプリケーション・メッセージ・データの形式、文字セット、およびエンコードは、メッセージの先頭にある MQDLH 構造体内の *Format*、*CodedCharSetId*、および *Encoding* フィールドでそれぞれ定義されます。MQDLH 構造体およびアプリケーション・メッセージ・データは、異なる文字セットおよび異なるエンコードをもつことができるため、MQDLH 構造体とアプリケーション・メッセージ・データのうちの一方あるいは両方に変換を要求できます。

キュー・マネージャーは必要に応じて MQDLH 構造体を最初に変換します。変換が成功した場合、または MQDLH 構造体に変換を必要としない場合、キュー・マネージャーは MQDLH 構造体の *CodedCharSetId* フィールドと *Encoding* フィールドを調べて、アプリケーション・メッセージ・データの変換が必要かどうかを確認します。変換が必要な場合、キュー・マネージャーは MQDLH 構造体の *Format* フィールドで付けた名前ユーザー作成出口を呼び出します。または *Format* が組み込み形式の名前である場合、変換そのものを行います。

MQGET 呼び出しが完了コード MQCC_WARNING を戻し、理由コードが変換の不成功を示すコードのうちの 1 つである場合、次のどちらかが適用されます。

- MQDLH 構造体は変換できなかった。この場合にはアプリケーション・メッセージ・データも変換されません。
- MQDLH 構造体は変換されたが、アプリケーション・メッセージ・データは変換されなかった。

アプリケーションは、**MsgDesc** パラメーターの **CodedCharSetId** フィールドと **Encoding** フィールドに返された値、および MQDLH 構造体の値を調べて、前に適用された値を判別することができます。

14. 形式の名前が MQFMT_XMIT_Q_HEADER の場合、メッセージ・データは MQXQH 構造体で始まり、このあとにおそらく 0 バイト以上の追加のデータが続く場合があります。この追加データは、長さがゼロの場合もありますが、通常はアプリケーション・メッセージ・データです。ただし、追加データの先頭に 1 つまたは複数の MQ ヘッダー構造体がさらに付いている場合もあります。

MQXQH 構造体はキュー・マネージャーの文字セット内およびエンコード内になければなりません。MQMD 構造体のあとのデータの形式、文字セット、およびエンコードは、MQXQH 内にある MQMD 構造体の **Format**、**CodedCharSetId**、および **Encoding** フィールドで与えられます。以降の各 MQ ヘッダー構造体の場合、構造体の **Format**、**CodedCharSetId**、および **Encoding** フィールドは、その構造体のあとに続くデータを説明しています。つまり、このデータは別の MQ ヘッダー構造体、またはアプリケーション・メッセージ・データです。

MQGMO_CONVERT オプションを MQFMT_XMIT_Q_HEADER メッセージに指定した場合、アプリケーション・メッセージ・データおよびある特定の MQ ヘッダー構造体は変換されます。ただし、MQXQH 構造体のデータは変換されません。したがって、MQGET 呼び出しからの戻り値は、次のようになります。

- **MsgDesc** パラメーター内の **Format**、**CodedCharSetId**、および **Encoding** フィールドの値は、MQXQH 構造体のデータを説明していますが、アプリケーション・メッセージ・データについては説明していません。したがって、値は MQGET 呼び出しを発行したアプリケーションが指定した値と同じではありません。

この結果、MQGMO_CONVERT オプションを指定して伝送キューからメッセージを繰り返し取得するアプリケーションは、各 MQGET 呼び出しの前に、**MsgDesc** パラメーターの **CodedCharSetId** フィールドと **Encoding** フィールドを、アプリケーション・メッセージ・データに必要な値にリセットする必要があります。

- 最後にある MQ ヘッダー構造体内の **Format** フィールド、**CodedCharSetId**、および **Encoding** フィールドの値は、アプリケーション・メッセージ・データを説明します。MQ ヘッダー構造体がない場合、アプリケーション・メッセージ・データは MQXQH 構造体内の MQMD 構造体にある上記と同じフィールドで説明されます。変換が正常に行われると、値は MQGET 呼び出しを発行したアプリケーションによって **MsgDesc** パラメーターに指定された値と同じになります。

メッセージが配布リスト・メッセージの場合、MQXQH 構造体のあとには MQDH 構造体 (それに加えてその MQOR レコードと MQPMR レコードの配列) が続きます。このあとに、さらに 1 つ以上の MQ ヘッダー構造体と、1 バイト以上のアプリケーション・メッセージ・データが続くこともあります。MQXQH 構造体と同様、MQDH 構造体はキュー・マネージャーの文字セット内およびエンコード内になければなりません。また、MQGMO_CONVERT オプションが指定されていても、MQDH 構造体は MQGET 呼び出しでは変換されません。

上述の MQXQH 構造体および MQDH 構造体の処理では、メッセージ・チャネル・エージェントが伝送キューからメッセージを読み取る際、メッセージ・チャネル・エージェントがこの 2 つの構造体を使用することを主な目的としています。

レポート・メッセージの変換

通常、レポート・メッセージに入るアプリケーション・メッセージ・データの量は、元のメッセージの送信側が指定したレポート・オプションに従って変えることができます。ただし、アクティビティー・レポートにはデータを含めることができますが、定数内の *_WITH_DATA を記述するレポート・オプションはありません。

特に、レポート・メッセージに入れるデータを以下のいずれかにすることができます。

1. アプリケーション・メッセージ・データなし
2. 元のメッセージからの複数のアプリケーション・メッセージ・データ

これが発生するのは、元のメッセージの送信側が MQRO*_WITH_DATA を指定し、メッセージが 100 バイトよりも長い場合です。

3. 元のメッセージからのすべてのアプリケーション・メッセージ・データ

これが発生するのは、元のメッセージの送信側が MQRO*_WITH_FULL_DATA または MQRO*_WITH_DATA を指定し、メッセージの長さが 100 バイト以下の場合です。

キュー・マネージャーまたはメッセージ・チャネル・エージェントがレポート・メッセージを生成すると、元のメッセージからレポート・メッセージ内の制御情報の *Format* フィールドに形式名をコピーします。したがって、レポート・メッセージ内の形式名からわかるデータ長は、レポート・メッセージの実際の長さとは異なることがあります (前述の 1 および 2 の場合)。

レポート・メッセージを取り出すときに MQGMO_CONVERT オプションが指定された場合、以下のことが発生します。

- 前述の 1 の場合、データ変換出口は呼び出されません (レポート・メッセージにデータが含まれないため)。
- 前述の 3 の場合、形式名はメッセージ・データの長さを正確に暗黙指定します。
- しかし前述の 2 の場合は、データ変換出口が呼び出されて、形式名で暗黙指定された長さより短いメッセージが変換されます。

さらに、出口に渡される理由コードは通常 MQRC_NONE です (理由コードは、メッセージが切り捨てられたことを表示しません)。メッセージ・データがレポート・メッセージの送信側に切り捨てられ、MQGET 呼び出しに応答した受信側のキュー・マネージャーに切り捨てられるわけではないからです。

上記の問題があるので、データ変換出口に渡すデータの長さを形式名を使用して縮めることは避けてください。その代わりに、出口は与えられたデータの長さをチェックし、さらに形式名で暗黙指定された長さより短いデータを変換する準備を整える必要があります。データが正常に変換された場合は、出口が完了コード MQCC_OK および理由コード MQRC_NONE を戻す必要があります。変換されるメッセージ・データの長さは、**InBufferLength** パラメーターとして出口に渡されます。

製品センシティブ・プログラミング・インターフェース

MQDXP - データ変換出口パラメーター

MQDXP 構造体は、MQGET 呼び出し処理の一環として、メッセージ・データを変換するためにデータ変換出口を呼び出す時に、キュー・マネージャーがこの出口に渡すパラメーターです。データ変換出口の詳細については、MQ_DATA_CONV_EXIT 呼び出しに関する説明を参照してください。

MQDXP の文字データは、ローカル・キュー・マネージャーの文字セットのものであり、この文字セットは **CodedCharSetId** キュー・マネージャー属性によって指定されます。MQDXP の数値データは、ネイティブ・マシンのエンコード方式であり、これは MQENC_NATIVE によって指定されます。

出口で変更できる MQDXP のフィールドは、*DataLength*、*CompCode*、*Reason*、および *ExitResponse* のみです。他のフィールドへの変更は無視されます。ただし、変換対象のメッセージが論理メッセージの一部だけを含むセグメントである場合、*DataLength* フィールドは変更できません。

出口からキュー・マネージャーに制御が戻されると、キュー・マネージャーは、MQDXP 内に返されたこれらの値をチェックします。これらの返された値が有効でない場合、キュー・マネージャーは、出口が *ExitResponse* 内に MQXDR_CONVERSION_FAILED を返したものとして、処理を再開します。ただしこの場合、キュー・マネージャーは出口から返された *CompCode* および *Reason* フィールドの値を無視し、代わりに出口への入力時にこれらのフィールドに指定されていた値を使用します。この処理が行われるのは、MQDXP 内の値が次のような場合です。

- *ExitResponse* フィールドが MQXDR_OK でも MQXDR_CONVERSION_FAILED でもない。
- *CompCode* フィールドが MQCC_OK でも MQCC_WARNING でもない。
- *DataLength* フィールドがゼロより小さい、または変換対象のメッセージが論理メッセージの一部だけを含むセグメントである場合に *DataLength* フィールドが変更された。

以下の表に、この構造体の各フィールドを要約します。

表 670. MQDXP のフィールド		
フィールド	説明	トピック
<i>StrucId</i>	構造体 ID	StrucId
<i>Version</i>	構造体のバージョン番号	バージョン
<i>AppOptions</i>	アプリケーション・オプション	AppOptions
<i>Encoding</i>	アプリケーションが必要とする数値エンコード方式	Encoding
<i>CodedCharSetId</i>	アプリケーションに必須の文字セット	CodedCharSetId
<i>DataLength</i>	メッセージ・データ長 (バイト単位)	DataLength
<i>CompCode</i>	完了コード	CompCode
<i>Reason</i>	<i>CompCode</i> を限定する理由コード。	理由 (Reason)
<i>ExitResponse</i>	出口からの応答	ExitResponse
<i>Hconn</i>	接続ハンドル	Hconn
<i>pEntryPoints</i>	MQIEP 構造体のアドレス	pEntryPoints

フィールド

MQDXP 構造体には、以下のフィールドが含まれます。フィールドはアルファベット順に説明されています。

AppOptions

タイプ: MQLONG

これは、MQGET 呼び出しを発行したアプリケーションが指定していた MQGMO 構造体の *Options* フィールドのコピーです。出口でこれを検査して、MQGMO_ACCEPT_TRUNCATED_MSG オプションが指定されていたかどうかを確認しなければならない場合があります。

これは、出口に対する入力フィールドです。

CodedCharSetId

タイプ: MQLONG

これは、MQGET 呼び出しを発行しているアプリケーションに必須の文字セットのコード化文字セット ID です。詳細については、MQMD 構造体の *CodedCharSetId* フィールドを参照してください。アプリケーションが MQGET 呼び出しに特殊値 MQCCSI_Q_MGR を指定する場合、キュー・マネージャーは、この値をキュー・マネージャーが使用する文字セットの実際の文字セット ID に変更してから、出口を呼び出します。

変換が正常に行われた場合、出口は、これをメッセージ記述子内の *CodedCharSetId* フィールドにコピーする必要があります。

これは、出口に対する入力フィールドです。

CompCode

タイプ: MQLONG

出口の呼び出し時、このフィールドには、出口が処理を行わなかった場合に、MQGET 呼び出しを発行したアプリケーションに返される完了コードが入っています。このコードは常に MQCC_WARNING です。これは、メッセージが切り捨てられたか、メッセージが変換を必要としているがまだ変換されていないかのどちらかであるためです。

出口からの出力時、このフィールドには、アプリケーションの MQGET 呼び出しの **CompCode** パラメーターに返される完了コードが入っています。MQCC_OK および MQCC_WARNING のみが有効です。出力時に出口でこのフィールドをどのように設定するかについての指針は、Reason フィールドの説明を参照してください。

これは、出口に対する入出力フィールドです。

DataLength

タイプ: MQLONG

出口の呼び出し時、このフィールドには、アプリケーションのメッセージ・データの元のデータ長が入っています。メッセージが、アプリケーションで指定されたバッファに収まるように既に切り捨てられている場合、出口に渡されるメッセージのサイズは、この *DataLength* の値よりも小さくなります。出口に渡されるメッセージのサイズは、切り捨てが行われたかどうかにかかわらず、常に出口の **InBufferLength** パラメーターに指定されたサイズになります。

出口に対する入力 of Reason フィールドに 値 MQRC_TRUNCATED_MSG_ACCEPTED が入っている場合は、切り捨てがあったことを意味します。

ほとんどの変換では、この長さを変更する必要はありませんが、必要な場合は、出口で長さを変更できます。出口で設定した値は、アプリケーションの MQGET 呼び出しの **DataLength** パラメーターに返されます。ただし、変換対象のメッセージが論理メッセージの一部だけを含むセグメントである場合、長さは変更できません。これは、長さの変更によって、論理メッセージ内の以降のセグメントのオフセットが不正確になるからです。

出口でデータ長を変更する必要がある場合、キュー・マネージャーが、既に未変換データの長さに基づき、メッセージ・データがアプリケーションのバッファに収まるかどうかを判断していることに留意してください。この判断によって、メッセージをキューから削除するかどうか (あるいはブラウズ要求の場合は、ブラウズ・カーソルを移動させるかどうか) が決定されます。この判断は、変換で生じたデータ長の変更による影響を受けません。このため、変換出口ではアプリケーションのメッセージ・データ長を変更しないことをお勧めします。

文字変換で長さの変化が発生する場合は、末尾ブランクを切り捨てたり、あるいは必要に応じてブランクを埋め込んだりして、ストリングを同じ長さ (バイト数) の別のストリングに変換できます。

メッセージにアプリケーションのメッセージ・データが入っていない場合、出口は呼び出されません。このため、*DataLength* は常にゼロより大きくなります。

これは、出口に対する入出力フィールドです。

Encoding

タイプ: MQLONG

アプリケーションが必要とする数値エンコード方式。

これは、MQGET 呼び出しを発行したアプリケーションが必要とする数値エンコード方式です。詳細については、MQMD 構造体の *Encoding* フィールドを参照してください。

変換が正常に行われた場合、出口は、これをメッセージ記述子内の *Encoding* フィールドにコピーします。

これは、出口に対する入力フィールドです。

ExitOptions

タイプ: MQLONG

これは予約フィールドで、値は 0 です。

ExitResponse

タイプ: MQLONG

出口からの応答。このフィールドは出口によって設定され、変換が成功したかどうかを示します。これは以下のいずれかです。

MQXDR_OK

変換は正常に行われました。

出口がこの値を指定した場合、キュー・マネージャーは、MQGET 呼び出しを発行したアプリケーションに、以下を返します。

- 出口からの出力で *CompCode* フィールドの値
- 出口からの出力で *Reason* フィールドの値
- 出口からの出力で *DataLength* フィールドの値
- 出口の出力バッファー *OutBuffer* の内容。返されるバイト数は、出口の **OutBufferLength** パラメーターと、出口からの出力時の *DataLength* フィールドの値のうち、小さい方です。

出口のメッセージ記述子パラメーター内の *Encoding* および *CodedCharSetId* フィールドが、両方とも変更されていない場合は、キュー・マネージャーは以下を返します。

- 出口に対する入力の MQDXP 構造体の *Encoding* および *CodedCharSetId* フィールドの値。

出口のメッセージ記述子パラメーター内の *Encoding* および *CodedCharSetId* フィールドのうち、一方または両方が変更された場合、キュー・マネージャーは以下を返します。

- 出口からの出力時の出口のメッセージ記述子パラメーター内の *Encoding* および *CodedCharSetId* フィールドの値

MQXDR_CONVERSION_FAILED

変換は失敗しました。

出口がこの値を指定した場合、キュー・マネージャーは、MQGET 呼び出しを発行したアプリケーションに、以下を返します。

- 出口からの出力で *CompCode* フィールドの値
- 出口からの出力で *Reason* フィールドの値
- 出口への入力で *DataLength* フィールドの値
- 出口の入力バッファー *InBuffer* の内容。返されるバイト数は、**InBufferLength** パラメーターによって指定されます。

出口で *InBuffer* を変更した場合の結果は定義されていません。

ExitResponse は、出口からの出力フィールドです。

Hconn

タイプ: MQHCONN

これは、MQXCNVN 呼び出しで利用できる接続ハンドルです。このハンドルは、MQGET 呼び出しを発行したアプリケーションによって指定されたハンドルと、必ずしも同じでなくても構いません。

pEntryPoints

タイプ: PMQIEP

MQIEP 構造体のアドレス。これによって、MQI および DCI 呼び出しを実行できます。

Reason

タイプ: MQLONG

CompCode を限定する理由コード。

出口の呼び出し時、このフィールドには、この出口が処理を行わないことを選択した場合に、MQGET 呼び出しを発行したアプリケーションに返される理由コードが入っています。可能性のある値としては、アプリケーションで指定されたバッファーに収まるようにメッセージが切り捨てられたことを示す MQRC_TRUNCATED_MSG_ACCEPTED、およびメッセージが変換を必要としているがまだ変換されていないことを示す MQRC_NOT_CONVERTED などがあります。

出口からの出力時、このフィールドには、アプリケーションの MQGET 呼び出しの **Reason** パラメーターに返される理由が入っています。以下のことをお勧めします。

- 出口への入力時に、*Reason* に値 MQRC_TRUNCATED_MSG_ACCEPTED が入っていた場合は、変換の成功または失敗にかかわらず、*Reason* および *CompCode* フィールドを変更しないでください。

(*CompCode* フィールドが MQCC_OK でない場合、メッセージを取得するアプリケーションは、メッセージ記述子内に返された *Encoding* および *CodedCharSetId* 値を、要求した値と比較することによって、変換が失敗したかどうかを識別することができます。しかし、切り捨てられたメッセージと、バッファに適合するメッセージとを区別することはできません。このため、MQRC_TRUNCATED_MSG_ACCEPTED を、変換の失敗を示す理由よりも優先して返す必要があります)。

- 出口への入力時に *Reason* に他の値が入っていた場合は、以下のようになります。
 - 変換が成功した場合、*CompCode* には MQCC_OK が設定され、*Reason* には MQRC_NONE が設定されなければなりません。
 - 変換が失敗した場合、またはメッセージが拡張されたためバッファに収まるように切り捨てる必要がある場合は、*CompCode* を MQCC_WARNING に設定し (または変更せずそのままにして)、*Reason* をリストされたいずれかの値に設定して、失敗の性質を示す必要があります。

変換後のメッセージがバッファに比べて大きすぎる場合、MQGET 呼び出しを発行したアプリケーションが MQGMO_ACCEPT_TRUNCATED_MSG オプションを指定している場合に限って、切り捨てる必要があることに注意してください。

- このオプションが指定されていた場合、理由 MQRC_TRUNCATED_MSG_ACCEPTED が返されます。
- このオプションが指定されていなかった場合、メッセージは変換されず、理由コード MQRC_CONVERTED_MSG_TOO_BIG と共に返されます。

リストされている理由コードは、出口が、変換に失敗した理由を示すために使用することが推奨されています。ただし、適切と見なされる場合、出口は MQRC_* コード・セットの他の値を返すこともあります。また、出口が MQGET 呼び出しを発行したアプリケーションに通知する状態を示すために、MQRC_APPL_FIRST から MQRC_APPL_LAST までの範囲の値が割り振られています。

注: メッセージを正常に変換できなかった場合、出口は、キュー・マネージャーに未変換のメッセージを返させるために、MQXDR_CONVERSION_FAILED を *ExitResponse* フィールド内に返す必要があります。これは、*Reason* フィールドに返される理由コードに関係なく該当します。

MQRC_APPL_FIRST

(900, X'384') アプリケーション定義の理由コードの最小値。

MQRC_APPL_LAST

(999, X'3E7') アプリケーション定義の理由コードの最大値。

MQRC_CONVERTED_MSG_TOO_BIG

(2120, X'848') 変換されたデータが、バッファには大きすぎる。

MQRC_NOT_CONVERTED

(2119, X'847') メッセージ・データが変換されなかった。

MQRC_SOURCE_CCSD_ERROR

(2111, X'83F') ソース・エンコード文字セット ID が無効である。

MQRC_SOURCE_DECIMAL_ENC_ERROR

(2113, X'841') メッセージ内のパック 10 進数のエンコードが認識できない。

MQRC_SOURCE_FLOAT_ENC_ERROR

(2114, X'842') メッセージ内の浮動小数点のエンコードが認識できない。

MQRC_SOURCE_INTEGER_ENC_ERROR

(2112, X'840') ソース整数エンコードが認識できない。

MQRC_TARGET_CCSD_ERROR

(2115, X'843') ターゲット・エンコード文字セット ID が無効である。

MQRC_TARGET_DECIMAL_ENC_ERROR

(2117, X'845') 受信側で指定されたパック 10 進数のエンコードが認識できない。

MQRC_TARGET_FLOAT_ENC_ERROR

(2118, X'846') 受信側で指定された浮動小数点のエンコードが認識できない。

MQRC_TARGET_INTEGER_ENC_ERROR

(2116, X'844') ターゲット整数エンコードが認識できない。

MQRC_TRUNCATED_MSG_ACCEPTED

(2079, X'81F') 切り捨てられたメッセージが戻された (処理は完了している)。

これは、出口に対する入出力フィールドです。

StrucId

タイプ: MQCHAR4

構造体 ID 値は次のものでなければなりません。

MQDXP_STRUC_ID

データ変換出口パラメーター構造体の ID。

C 言語の場合、定数 MQDXP_STRUC_ID_ARRAY も定義されます。これは MQDXP_STRUC_ID と同じ値ですが、ストリングではなく文字の配列です。

これは、出口に対する入力フィールドです。

Version

タイプ: MQLONG

構造体のバージョン番号。値は次のものでなければなりません。

MQDXP_VERSION_1

データ変換出口パラメーター構造体のバージョン番号。

以下の定数は、現行バージョンのバージョン番号を指定しています。

MQDXP_CURRENT_VERSION

データ変換出口パラメーター構造体の現行バージョンです。

注: この構造体の新しいバージョンが導入されても、既存の部分のレイアウトは変わりません。したがって、出口は、*Version* フィールドの値が、使用する必要があるフィールドを備えた最低限のバージョン以上であることを検査する必要があります。

これは、出口に対する入力フィールドです。

C 宣言

```
typedef struct tagMQDXP MQDXP;
struct tagMQDXP {
    MQCHAR4   StrucId;           /* Structure identifier */
    MQLONG    Version;           /* Structure version number */
    MQLONG    ExitOptions;       /* Reserved */
    MQLONG    AppOptions;        /* Application options */
    MQLONG    Encoding;          /* Numeric encoding required by
                                application */
    MQLONG    CodedCharSetId;     /* Character set required by application */
    MQLONG    DataLength;        /* Length in bytes of message data */
    MQLONG    CompCode;          /* Completion code */
    MQLONG    Reason;            /* Reason code qualifying CompCode */
    MQLONG    ExitResponse;       /* Response from exit */
    MQHCONN   Hconn;             /* Connection handle */
    PMQIEP    pEntryPoints;      /* Address of the MQIEP structure */
};
```

COBOL 宣言 (IBM i のみ)

```
**      MQDXP structure
10 MQDXP.
**      Structure identifier
15 MQDXP-STRUCID      PIC X(4).
**      Structure version number
15 MQDXP-VERSION      PIC S9(9) BINARY.
**      Reserved
15 MQDXP-EXITOPTIONS  PIC S9(9) BINARY.
**      Application options
15 MQDXP-APPOPTIONS   PIC S9(9) BINARY.
```

```

**      Numeric encoding required by application
15 MQDXP-ENCODING      PIC S9(9) BINARY.
**      Character set required by application
15 MQDXP-CODEDCHARSETID PIC S9(9) BINARY.
**      Length in bytes of message data
15 MQDXP-DATALength    PIC S9(9) BINARY.
**      Completion code
15 MQDXP-COMPCODE      PIC S9(9) BINARY.
**      Reason code qualifying COMPCODE
15 MQDXP-REASON        PIC S9(9) BINARY.
**      Response from exit
15 MQDXP-EXITRESPONSE  PIC S9(9) BINARY.
**      Connection handle
15 MQDXP-HCONN         PIC S9(9) BINARY.

```

System/390 アセンブラ宣言

```

MQDXP      DSECT
MQDXP_STRUCID DS CL4 Structure identifier
MQDXP_VERSION DS F   Structure version number
MQDXP_EXITOPTIONS DS F   Reserved
MQDXP_APPOPTIONS DS F   Application options
MQDXP_ENCODING DS F   Numeric encoding required by application
MQDXP_CODEDCHARSETID DS F   Character set required by application
MQDXP_DATALength DS F   Length in bytes of message data
MQDXP_COMPCODE DS F   Completion code
MQDXP_REASON DS F   Reason code qualifying COMPCODE
MQDXP_EXITRESPONSE DS F   Response from exit
MQDXP_HCONN DS F   Connection handle
*
MQDXP_LENGTH EQU *-MQDXP
ORG MQDXP
MQDXP_AREA DS CL(MQDXP_LENGTH)

```

MQXCNCV - 文字の変換

MQXCNCV 呼び出しは、C プログラミング言語を使用して文字セットを別の文字セットに変換します。

この呼び出しは、IBM MQ フレームワーク・インターフェースの 1 つである IBM MQ データ変換インターフェース (DCI) の一部です。

注: この呼び出しは、アプリケーションとデータ変換出口の両方の環境で使用できます。

構文

MQXCNCV (*Hconn*, *Options*, *SourceCCSID*, *SourceLength*, *SourceBuffer*, *TargetCCSID*, *TargetLength*, *TargetBuffer*, *DataLength*, *CompCode*, *Reason*)

パラメーター

Hconn

タイプ: MQHCONN - 入力

このハンドルは、キュー・マネージャーに対する接続を表します。

データ変換出口において、Hconn は通常、MQDXP 構造体の Hconn フィールドに入れてデータ変換出口に渡されるハンドルです。このハンドルは、必ずしも、MQGET 呼び出しを発行したアプリケーションが指定するハンドルと同じものではありません。

では IBM i、以下の特別な値を指定できます: Hconn

MQHC_DEF_HCONN

デフォルトの接続ハンドル。

CICS TS 3.2 以上のアプリケーションを実行する場合は、MQXCNCV 呼び出しを起動する文字変換出口プログラムが OPENAPI として定義されていることを確認してください。この定義により、正しくない接続が原因で発生する 2018 MQRC_HCONN_ERROR が回避され、MQGET を完了できます。

オプション

タイプ: MQLONG - 入力

MQXCNCV のアクションを制御するオプション。

説明されているオプションの 1 つ以上を指定できます。複数のオプションを指定するには、値を加算する (同じ定数を複数回加算しない) か、ビット単位 OR 演算を使用して値を結合 (プログラミング言語でビット演算がサポートされる場合) します。

デフォルト変換オプション: 以下のオプションはデフォルトの文字変換の使用を制御します。

MQDCC_DEFAULT_CONVERSION

デフォルト変換。

このオプションは、呼び出しで指定された文字セットが片方または両方ともサポートされていない場合、デフォルト文字変換が使用できることを指定します。これにより、キュー・マネージャーはインストール時に指定されたデフォルト文字セットを使用できます。このオプションを指定しておく、と、ストリング変換時には、デフォルトの文字セットのうち、呼び出しで指定された文字セットに最も近いものが使用されます。

注: 呼び出し時に指定された文字セット以外を使用してストリングの変換を行うと、一部の文字が正しく変換されない場合があります。指定された文字セットとデフォルト文字セットの両方に共通する文字だけをストリング内に使用するようになれば、これを回避することができます。

デフォルト文字セットは、キュー・マネージャーのインストール時または再始動時に、構成オプションによって定義されます。

MQDCC_DEFAULT_CONVERSION が指定されない場合は、キュー・マネージャーは指定された文字セットのみを使用してストリングを変換します。この場合、呼び出しは文字セットの片方または両方がサポートされていないと失敗します。

このオプションは、次の環境でサポートされます。

-  AIX
-  IBM i
-  Linux
-  Windows

埋め込みオプション: 以下のオプションを指定すると、変換後のストリングがターゲット・バッファーに収まるよう、キュー・マネージャーは、変換後のストリングにブランクを埋め込んだり、有意でない末尾文字を廃棄します。

MQDCC_FILL_TARGET_BUFFER

ターゲット・バッファーを埋める。

このオプションは、ターゲット・バッファーが完全に埋められるように変換が行われることを要求します。

- ストリングを変換すると短縮する場合、ターゲット・バッファーを埋めるために末尾ブランクが追加されます。
- ストリングを変換すると拡張する場合、変換されたストリングがターゲット・バッファーに収まるよう、有意でない末尾文字が廃棄されます。この廃棄が正常に行われると、MQCC_OK と理由コード MQRC_NONE が戻されて呼び出しが完了します。

有意でない末尾文字の数が足りないときは、ストリングのうちのターゲット・バッファーに収まるだけの数の文字がターゲット・バッファーに入れられ、MQCC_WARNING と理由コード MQRC_CONVERTED_MSG_TOO_BIG が戻されて呼び出しが完了します。

有意でない文字とは、次の文字です。

- 末尾ブランク
- ストリング内の最初のヌル文字に続く文字 (ただし、最初のヌル文字自体は除く)

- スtring、TargetCCSID、および TargetLength が、ターゲット・バッファを有効な文字で完全にセットできないようなものである場合、呼び出しは失敗し、MQCC_FAILED および理由コード MQRC_TARGET_LENGTH_ERROR が戻されます。これは、が純粋な DBCS 文字セット (UTF-16 など) である TargetCCSID 場合に発生する可能性があります。ただし、TargetLength が指定する長さが奇数バイトの場合に限ります。
- TargetLength より小さいか、より大きい、あるいはその両方である可能性がある SourceLength。MQXCNCV から戻ったとき、DataLength の値は、TargetLength と同じになっています。

このオプションを指定しない場合は、以下のようになります。

- String は、ターゲット・バッファ内で必要に応じて短縮したり、拡張することができます。有意でない末尾の文字が追加されることも廃棄されることもあります。

変換された String がターゲット・バッファに収まる場合は、MQCC_OK と理由コード MQRC_NONE が戻されて呼び出しが完了します。

変換された String がターゲット・バッファに対して大き過ぎる場合は、String のうちのそのバッファに収まるだけの数の文字がターゲット・バッファに入れられ、MQCC_WARNING と理由コード MQRC_CONVERTED_MSG_TOO_BIG が戻されて呼び出しが完了します。この場合、戻されるバイト数は TargetLength より少ないことがあるので注意してください。

- TargetLength より小さいか、より大きい、あるいはその両方である可能性がある SourceLength。MQXCNCV から戻ったとき、DataLength の値は、TargetLength 以下になっています。

このオプションは、次の環境でサポートされます。

-  AIX
-  IBM i
-  Linux
-  Windows

エンコード・オプション: ここで説明するオプションを使用して、ソース・String およびターゲット・String の整数エンコードを指定できます。該当するエンコードが使用される条件は、対応する文字セットの ID から決まります。すなわち、この文字セットの主記憶内での表現が 2 進整数のエンコードに依存することを ID が意味している場合にのみ、このエンコードが使用されます。これによって影響を受けるのは、特定のマルチバイト文字セット (例えば、UTF-16 文字セット) だけです。

文字セットが 1 バイト文字セット (SBCS)、または主記憶内の表現が整数エンコードによって変わらないマルチバイト文字セットの場合、エンコードは無視されます。

次に示す、MQDCC_SOURCE_* の値の 1 つと MQDCC_TARGET_* の値の 1 つを組み合わせで指定してください。

MQDCC_SOURCE_ENC_NATIVE

ソース・エンコードはこの環境およびプログラミング言語ではデフォルト。

MQDCC_SOURCE_ENC_NORMAL

ソース・エンコードは順方向。

MQDCC_SOURCE_ENC_REVERSED

ソース・エンコードは逆方向。

MQDCC_SOURCE_ENC_UNDEFINED

ソース・エンコードは未定義。

MQDCC_TARGET_ENC_NATIVE

ターゲット・エンコードはこの環境およびプログラミング言語ではデフォルト。

MQDCC_TARGET_ENC_NORMAL

ターゲット・エンコードは順方向。

MQDCC_TARGET_ENC_REVERSED

ターゲット・エンコードは逆方向。

MQDCC_TARGET_ENC_UNDEFINED

ターゲット・エンコードは未定義。

定義済みのエンコードの値は、Options フィールドに直接加算できます。ただし、ソースまたはターゲットの符号化が MQMD またはその他の構造体の Encoding フィールドから取得される場合、以下の処理を実行しなければならない：

1. Encoding フィールドから浮動小数点エンコードおよびパック 10 進エンコードを除去して、整数エンコードを取り出す。この方法の詳細については、[897 ページの『エンコードの分析』](#)を参照してください。
2. 上記 1 項で得られた整数エンコードに該当する係数を掛けてから、Options フィールドに加える。係数には、次の 2 つがあります。

- MQDCC_SOURCE_ENC_FACTOR (ソース・エンコードの場合)
- MQDCC_TARGET_ENC_FACTOR (ターゲット・エンコードの場合)

C プログラミング言語でこれらの係数をコーディングする方法の例を以下に示します。

```
Options = (MsgDesc.Encoding & MQENC_INTEGER_MASK)
          * MQDCC_SOURCE_ENC_FACTOR
          + (DataConvExitParms.Encoding & MQENC_INTEGER_MASK)
          * MQDCC_TARGET_ENC_FACTOR;
```

エンコード・オプションを指定しない場合は、未定義を表す (MQDCC_*_ENC_UNDEFINED) をデフォルトにとります。ほとんどの場合、MQXCNVC 呼び出しの正常な完了に対してこれが影響を与えることはありません。ただし、対応する文字セットが、表現がエンコードによって変わるマルチバイト文字セット (例えば、UTF-16 文字セット) の場合、呼び出しは失敗し、理由コード

MQRC_SOURCE_INTEGER_ENC_ERROR または MQRC_TARGET_INTEGER_ENC_ERROR のうち該当する方が戻ります。

エンコード・オプションは、次の環境でサポートされます。

-  AIX
-  IBM i
-  Linux
-  Windows
-  z/OS

デフォルト・オプション: これまで説明したオプションを何も指定しない場合、以下のオプションを使用できます。

MQDCC_NONE

指定されるオプションはありません。

MQDCC_NONE は、プログラム・ドキュメンテーションの援助機能として定義されています。このオプションを他の目的で使用することは意図されていませんが、値がゼロであるため、そのように使用しても、それを検出することはできません。

SourceCCSID

タイプ: MQLONG - 入力

SourceBuffer 内の入力ストリングのコード化文字セット ID です。

SourceLength

タイプ: MQLONG - 入力

SourceBuffer 内の入力ストリングの長さ (バイト数) です。これはゼロ以上でなければなりません。

SourceBuffer

タイプ: MQCHAR x SourceLength - 入力

文字セットを変換するストリングが入るバッファです。

TargetCCSID

タイプ: MQLONG - 入力

SourceBuffer の変換先の文字セットのコード化文字セット ID。

TargetLength

タイプ: MQLONG - 入力

出力バッファ TargetBuffer の長さ (バイト数) です。これはゼロ以上でなければなりません。また、SourceLength より小さくても大きくても構いません。

TargetBuffer

タイプ: MQCHAR x TargetLength - 出力

TargetCCSID が指定した文字セットに変換されたあとのストリングです。変換後のストリングは、未変換のストリングより短くなることも長くなることもあります。 **DataLength** パラメーターは、戻される有効バイト数を示します。

DataLength

タイプ: MQLONG - 出力

出力バッファ TargetBuffer に戻されるストリングの長さです。変換後のストリングは、未変換のストリングより短くなることも長くなることもあります。

CompCode

タイプ: MQLONG - 出力

これは、以下のいずれかになります。

MQCC_OK

正常終了。

MQCC_WARNING

警告 (部分完了)。

MQCC_FAILED

呼び出し失敗。

理由

タイプ: MQLONG - 出力

CompCode を限定する理由コード。

CompCode が MQCC_OK の場合:

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_WARNING の場合:

MQRC_CONVERTED_MSG_TOO_BIG

(2120, X'848') 変換されたデータが、バッファには大きすぎる。

CompCode が MQCC_FAILED の場合:

MQRC_DATA_LENGTH_ERROR

(2010, X'7DA') データ長パラメーターが無効である。

MQRC_DBCS_ERROR

(2150, X'866') DBCS ストリングが無効である。

MQRC_HCONN_ERROR

(2018, X'7E2') 接続ハンドルが無効です。

MQRC_OPTIONS_ERROR

(2046, X'7FE') オプションが無効であるか、矛盾しています。

MQRC_RESOURCE_PROBLEM

(2102, X'836') 使用できるシステム・リソースが不足しています。

MQRC_SOURCE_BUFFER_ERROR

(2145, X'861') ソース・バッファ・パラメーターが無効。

MQRC_SOURCE_CCSDID_ERROR

(2111, X'83F') ソース・エンコード文字セット ID が無効である。

MQRC_SOURCE_INTEGER_ENC_ERROR

(2112, X'840') ソース整数エンコードが認識できない。

MQRC_SOURCE_LENGTH_ERROR

(2143, X'85F') ソース長パラメーターが無効である。

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') ストレージが不足しています。

MQRC_TARGET_BUFFER_ERROR

(2146, X'862') ターゲット・バッファ・パラメーターが無効である。

MQRC_TARGET_CCSDID_ERROR

(2115, X'843') ターゲット・エンコード文字セット ID が無効である。

MQRC_TARGET_INTEGER_ENC_ERROR

(2116, X'844') ターゲット整数エンコードが認識できない。

MQRC_TARGET_LENGTH_ERROR

(2144, X'860') ターゲット長パラメーターが無効。

MQRC_UNEXPECTED_ERROR

(2195, X'893') 予期しないエラーが発生しました。

これらのコードについて詳しくは、[メッセージおよび理由コード](#)を参照してください。

C 言語での呼び出し

```
MQXCNCV (Hconn, Options, SourceCCSID, SourceLength, SourceBuffer,
         TargetCCSID, TargetLength, TargetBuffer, &DataLength,
         &CompCode, &Reason);
```

パラメーターを次のように宣言します。

```
MQHCONN  Hconn;           /* Connection handle */
MQLONG   Options;         /* Options that control the action of
                           MQXCNCV */
MQLONG   SourceCCSID;     /* Coded character set identifier of string
                           before conversion */
MQLONG   SourceLength;    /* Length of string before conversion */
MQCHAR   SourceBuffer[n]; /* String to be converted */
MQLONG   TargetCCSID;     /* Coded character set identifier of string
                           after conversion */
MQLONG   TargetLength;    /* Length of output buffer */
MQCHAR   TargetBuffer[n]; /* String after conversion */
MQLONG   DataLength;      /* Length of output string */
MQLONG   CompCode;        /* Completion code */
MQLONG   Reason;          /* Reason code qualifying CompCode */
```

COBOL 宣言 (IBM i のみ)

IBM i

```
CALL 'MQXCNCV' USING HCONN, OPTIONS, SOURCECCSID, SOURCELENGTH,
                     SOURCEBUFFER, TARGETCCSID, TARGETLENGTH,
                     TARGETBUFFER, DATALENGTH, COMPCODE, REASON.
```

パラメーターを次のように宣言します。

```

** Connection handle
01 HCONN          PIC S9(9) BINARY.
** Options that control the action of MQXCNCV
01 OPTIONS        PIC S9(9) BINARY.
** Coded character set identifier of string before conversion
01 SOURCECCSID    PIC S9(9) BINARY.
** Length of string before conversion
01 SOURCELENGTH   PIC S9(9) BINARY.
** String to be converted
01 SOURCEBUFFER    PIC X(n).
** Coded character set identifier of string after conversion
01 TARGETCCSID    PIC S9(9) BINARY.
** Length of output buffer
01 TARGETLENGTH   PIC S9(9) BINARY.
** String after conversion
01 TARGETBUFFER    PIC X(n).
** Length of output string
01 DATALENGTH    PIC S9(9) BINARY.
** Completion code
01 COMPCODE       PIC S9(9) BINARY.
** Reason code qualifying COMPCODE
01 REASON         PIC S9(9) BINARY.

```

S/390 アセンブラ宣言

```

CALL MQXCNCV, (HCONN, OPTIONS, SOURCECCSID, SOURCELENGTH,      X
               SOURCEBUFFER, TARGETCCSID, TARGETLENGTH, TARGETBUFFER, X
               DATALENGTH, COMPCODE, REASON)

```

パラメーターを次のように宣言します。

HCONN	DS	F	Connection handle
OPTIONS	DS	F	Options that control the action of MQXCNCV
SOURCECCSID	DS	F	Coded character set identifier of string before conversion
SOURCELENGTH	DS	F	Length of string before conversion
SOURCEBUFFER	DS	CL(n)	String to be converted
TARGETCCSID	DS	F	Coded character set identifier of string after conversion
TARGETLENGTH	DS	F	Length of output buffer
TARGETBUFFER	DS	CL(n)	String after conversion
DATALENGTH	DS	F	Length of output string
COMPCODE	DS	F	Completion code
REASON	DS	F	Reason code qualifying COMPCODE

MQ_DATA_CONV_EXIT - データ変換出口

MQ_DATA_CONV_EXIT 呼び出しは、データ変換出口に渡されるパラメーターを記述します。

MQ_DATA_CONV_EXIT という名前の入り口点がキュー・マネージャーから提供されるわけではありません(使用上の注意 11 を参照)。

この定義は、IBM MQ フレームワーク・インターフェースの 1 つである IBM MQ データ変換インターフェース (DCI) の一部です。

構文

MQ_DATA_CONV_EXIT (*DataConvExitParms*, *MsgDesc*, *InBufferLength*, *InBuffer*, *OutBufferLength*, *OutBuffer*)

パラメーター

DataConvExitParms

タイプ: MQDXP - 入出力

この構造体には出口の呼び出しに関する情報があります。出口はこの構造体に情報を設定して、変換の結果を表示します。この構造体のフィールドの詳細については、[909 ページの『MQDXP - データ変換出口パラメーター』](#)を参照してください。

MsgDesc

タイプ: MQMD - 入出力

出口への入力時は、これは **InBuffer** パラメーターの出口に渡されるメッセージ・データに関連したメッセージ記述子です。

注: 出口に渡された **MsgDesc** パラメーターは、常に MQMD の最新のバージョンです。これは、出口を呼び出すキュー・マネージャーにサポートされています。出口を異なる環境間で移植可能にする予定の場合、MsgDesc 内の Version フィールドをチェックして、出口がアクセスする必要のあるフィールドがその構造体にあることを確認する必要があります。

以下の環境では、出口はバージョン 2 の MQMD に渡されます。

-  AIX
-  IBM i
-  Linux
-  Windows

データ変換出口をサポートするこれ以外のすべての環境では、出口はバージョン 1 の MQMD に渡されます。

変換が成功した場合、出口は、出力の Encoding および CodedCharSetId フィールドを、アプリケーションが要求した値に変更します。これらの変更はアプリケーションに反映されます。出口が構造体に加えたこれ以外の変更は無視され、アプリケーションには反映されません。

出口が、MQDXP 構造の ExitResponse フィールドに MQXDR_OK を戻すが、メッセージ記述子の Encoding または CodedCharSetId フィールドは変更されない場合、キュー・マネージャーはこれらのフィールドに対して、MQDXP 構造の対応するフィールドで出口に入力した値を戻します。

InBufferLength

タイプ: MQLONG - 入力

InBuffer の長さ (バイト数)。

入力バッファー InBuffer の長さで、出口が処理するバイト数を指定します。InBufferLength は、変換前のメッセージ・データの長さ、MQGET 呼び出しでアプリケーションが指定するバッファーの長さの、どちらか小さい方です。

値は常にゼロより大きくなります。

InBuffer

タイプ: MQBYTEExInBufferLength - 入力

未変換メッセージが入るバッファー。

変換前のメッセージ・データが入ります。出口がデータを変換できない場合は、キュー・マネージャーは、出口の完了後に、このバッファーの内容をアプリケーションに戻します。

注: 出口は、InBuffer を変更できません。このパラメーターを変更した場合は、結果は未定義となります。

C プログラミング言語では、このパラメーターは void を示すポインターとして定義されます。

OutBufferLength

タイプ: MQLONG - 入力

OutBuffer の長さ (バイト数)。

出力バッファー OutBuffer の長さで、アプリケーションが MQGET 呼び出しで指定するバッファーの長さと同じです。

値は常にゼロより大きくなります。

OutBuffer

タイプ: MQBYTEExOutBufferLength - 出力

変換後のメッセージが入るバッファ。

出口からの出力において、変換が成功した場合（パラメータ **DataConvExitParms** のフィールド **ExitResponse** に **MQXDR_OK** の値が示される）、は要求された表現形式で **OutBuffer** アプリケーションに配信されるメッセージデータを含む。変換が失敗した場合、出口がこのバッファに加えた変更はすべて無視されます。

C プログラミング言語では、このパラメーターは **void** を示すポインターとして定義されます。

使用上の注意

1. データ変換出口とは、**MQGET** 呼び出しの処理中に制御を受け取るユーザー作成出口です。データ変換出口が実行する関数は、出口の作成者が定義します。ただし、この出口は、ここで述べる規則および関連のパラメーター構造体 **MQDXP** に示されている規則に従っていなければなりません。

データ変換出口用に使用できるプログラミング言語は、環境によって決まります。

2. この出口は、次のすべての記述が該当する場合に限り、呼び出されます。
 - **MQGMO_CONVERT** オプションが、**MQGET** 呼び出しで指定されている。
 - メッセージ記述子の **Format** フィールドが **MQFMT_NONE** ではない。
 - メッセージがまだ必須表現形式になっていない。つまり、メッセージの **CodedCharSetId** および **Encoding** の一方または両方が、アプリケーションが **MQGET** 呼び出しのメッセージ記述子に指定した値と異なっている。
 - キュー・マネージャーが、まだ変換に成功していない。
 - アプリケーションのバッファの長さがゼロより大きい。
 - メッセージ・データの長さがゼロより大きい。
 - **MQGET** 操作中のこれまでの理由コードが、**MQRC_NONE** または **MQRC_TRUNCATED_MSG_ACCEPTED** である。
3. 出口を作成するときには、切り捨てにより短くなったメッセージを変換できる方法で出口をコーディングすることを検討してください。メッセージの切り捨てが起こるのは次のような場合です。
 - 受信側のアプリケーションがメッセージより小さいバッファを提供したにもかかわらず、**MQGET** 呼び出しで **MQGMO_ACCEPT_TRUNCATED_MSG** オプションを指定した。

この場合、出口への入力におけるパラメータ **DataConvExitParms** のフィールド **Reason** は、**MQRC_TRUNCATED_MSG_ACCEPTED** の値を持つ。

- メッセージの送信元が、送信前にメッセージを切り捨てた。これはレポート・メッセージで発生する可能性があります（詳細については、[908 ページの『レポート・メッセージの変換』](#)を参照してください）。

この場合、出口への入力におけるパラメータ **DataConvExitParms** の **Reason** フィールドには、**MQRC_NONE** の値が設定されます（受信アプリケーションがメッセージを収容できる十分な大きさのバッファを提供した場合）。

このように、出口への入力の **Reason** フィールドの値を使用しても、必ずしもメッセージが切り捨てられたかどうかを判別できるとは限りません。

切り捨てられたメッセージの目印となる特徴は、**InBufferLength** パラメーターに入れて出口に渡される長さが、メッセージ記述子の **Format** フィールドに含まれている形式名に暗黙に示されている長さより短くなるという点です。したがって、出口では、データの変換前に、**InBufferLength** の値をチェックするようにしてください。出口では、形式名に暗黙に示されている全データ量が与えられるものと想定しないでください。

切り捨てられたメッセージを変換するように出口を作成していないで、しかも **InBufferLength** が予期した値より小さい場合、出口は、**DataConvExitParms** パラメーターの **ExitResponse** フィールドに **MQXDR_CONVERSION_FAILED** を戻します。このとき、**CompCode** および **Reason** フィールドは、それぞれ **MQCC_WARNING** および **MQRC_FORMAT_ERROR** に設定されます。

切り捨てられたメッセージを変換するように出口を作成している場合、出口は、できるだけ多くのデータを変換し(次の「使用上の注意」を参照)、InBufferの終わりを越えてデータの検査または変換を行わないように配慮します。変換が正常に完了した場合、終了時には **DataConvExitParms** パラメータ内のフィールド Reason は変更されません。この場合、受信側のキュー・マネージャーがメッセージを切り捨てた場合は MQRC_TRUNCATED_MSG_ACCEPTED が戻され、メッセージの送信側がメッセージを切り捨てた場合は MQRC_NONE が戻されます。

変換中にメッセージが拡張されて、OutBuffer より大きくなってしまうこともあります。この場合、出口はメッセージを切り詰めるかどうかを決定する必要があります。パラメータ **DataConvExitParms** 内のフィールド AppOptions は、受信アプリケーションが MQGMO_ACCEPT_TRUNCATED_MSG オプションを指定したかどうかを示します。

4. 一般には、InBuffer に入れて出口に渡されるメッセージ内のデータをすべて変換するか、そのデータを一切変換しないかのどちらかにしてください。ただし、変換前または変換中にメッセージが切り捨てられた場合は例外です。その場合、バッファの末尾に不完全な項目が入っていることがあります(例えば、2 バイト文字の 1 バイト分、または 4 バイト整数の 3 バイト分など)。そのような状態の場合は、不完全な項目を省略して、OutBuffer の未使用バイトはヌルに設定することを検討してください。しかし、配列またはストリング内の完全な要素または文字は、変換するようにしてください。
5. 出口が初めて必要になった時点で、キュー・マネージャーは、形式と同じ名前(拡張子は除く)を持つオブジェクトをロードしようとします。ロードされるオブジェクトには、その形式名をもつメッセージを処理する出口が含まれていなければなりません。出口の名前と、その出口を含むオブジェクトの名前を同じにすることを検討してください。ただし、すべての環境でこのことが必要なわけではありません。
6. 出口の新規コピーがロードされるのは、アプリケーションがキュー・マネージャーに接続された後、Format を使用する最初のメッセージを検索しようとするときです。CICS または IMS アプリケーションの場合、これは、CICS サブシステムまたは IMS サブシステムがキュー・マネージャーに接続されている場合を意味します。キュー・マネージャーが前回ロードしたコピーを破棄した場合は、別の機会に新しいコピーをロードすることもできます。したがって、出口では、静的ストレージを使用して、出口の呼び出しからその次の呼び出しへと情報を伝えようとしないでください。この 2 回の呼び出しの間に出口がアンロードされることがあります。
7. ユーザー作成の出口の名前が、キュー・マネージャーがサポートする組み込み形式の 1 つと同じ名前でも、組み込み変換ルーチンは置換されません。このような出口が呼び出される状況としては、次の場合があります。
 - ・ 組み込み変換ルーチンが CodedCharSetId または Encoding への変換、あるいはこの 2 つのいずれかのフィールドからの変換を処理できない場合
 - ・ 組み込み変換ルーチンがデータの変換に失敗した場合(変換不能のフィールドまたは文字がある場合など)
8. 出口の有効範囲は環境によって異なります。Format には、他の形式と競合する危険の少ない名前を使用してください。その形式名を定義するアプリケーションを識別するいくつかの文字を名前の先頭に付けることを検討してください。
9. データ変換出口は、MQGET 呼び出しを発行したプログラムの環境に似た環境で実行されます。環境には、アドレス・スペースおよびユーザー・プロファイル(適用される場合)が含まれます。このプログラムは、メッセージ変換をサポートしない宛先キュー・マネージャーにメッセージを送るメッセージ・チャネル・エージェントであることもあります。出口は、キュー・マネージャーの環境で実行されるわけではないので、キュー・マネージャーの整合性を損なうことはありません。
10. 出口により使用できる MQI 呼び出しは MQXCNVC のみです。他の MQI 呼び出しを使用しようとすると、その呼び出しは失敗し、理由コード MQRC_CALL_IN_PROGRESS が戻るか、その他の予測不能のエラーが起こります。
11. MQ_DATA_CONV_EXIT という名前の入り口点がキュー・マネージャーから提供されるわけではありません。ただし、C プログラミング言語では、MQ_DATA_CONV_EXIT という名前に対応する typedef が提供されているので、これを使用してユーザー作成の出口を宣言して、パラメーターが正しいことを確認することができます。出口の名前は、形式名(MQMD の Format フィールドに含まれている名前)と同じであることが望ましいのですが、これは必ずしもすべての環境で必要ではありません。

次の例は、形式 MYFORMAT を処理する出口を C プログラミング言語でどのように宣言するかを示しています。

```
#include "cmqc.h"
#include "cmqxc.h"

MQ_DATA_CONV_EXIT MYFORMAT;

void MQENTRY MYFORMAT(
    PMQDXP  pDataConvExitParms, /* Data-conversion exit parameter
                                block */
    PMQMD   pMsgDesc,           /* Message descriptor */
    MQLONG  InBufferLength,     /* Length in bytes of InBuffer */
    PMQVOID pInBuffer,          /* Buffer containing the unconverted
                                message */
    MQLONG  OutBufferLength,    /* Length in bytes of OutBuffer */
    PMQVOID pOutBuffer)         /* Buffer containing the converted
                                message */
{
    /* C language statements to convert message */
}
```

12.  z/OS では、API 交差出口も強制されている場合、データ変換出口のあとに呼び出されます。

C 言語での呼び出し

```
exitname (&DataConvExitParms, &MsgDesc, InBufferLength,
          InBuffer, OutBufferLength, OutBuffer);
```

出口に渡されるパラメーターは、次のように宣言されます。

```
MQDXP  DataConvExitParms; /* Data-conversion exit parameter block */
MQMD    MsgDesc;          /* Message descriptor */
MQLONG  InBufferLength;    /* Length in bytes of InBuffer */
MQBYTE  InBuffer[n];       /* Buffer containing the unconverted
                             message */
MQLONG  OutBufferLength;   /* Length in bytes of OutBuffer */
MQBYTE  OutBuffer[n];      /* Buffer containing the converted
                             message */
```

COBOL 宣言 (IBM i のみ)

 IBM i

```
CALL 'exitname' USING DATACONVEXITPARMS, MSGDESC, INBUFFERLENGTH,
                      INBUFFER, OUTBUFFERLENGTH, OUTBUFFER.
```

出口に渡されるパラメーターは、次のように宣言されます。

```
** Data-conversion exit parameter block
01 DATACONVEXITPARMS.
   COPY CMQDXPV.
** Message descriptor
01 MSGDESC.
   COPY CMQMDV.
** Length in bytes of INBUFFER
01 INBUFFERLENGTH PIC S9(9) BINARY.
** Buffer containing the unconverted message
01 INBUFFER PIC X(n).
** Length in bytes of OUTBUFFER
01 OUTBUFFERLENGTH PIC S9(9) BINARY.
** Buffer containing the converted message
01 OUTBUFFER PIC X(n).
```

System/390 アセンブラー宣言

```
CALL EXITNAME, (DATACONVEXITPARMS, MSGDESC, INBUFFERLENGTH,      X
                INBUFFER, OUTBUFFERLENGTH, OUTBUFFER)
```

出口に渡されるパラメーターは、次のように宣言されます。

DATA CONVEXITPARMS	CMQDXPA	,	Data-conversion exit parameter block
MSGDESC	CMQMDA	,	Message descriptor
INBUFFERLENGTH	DS	F	Length in bytes of INBUFFER
INBUFFER	DS	CL(n)	Buffer containing the unconverted message
* OUTBUFFERLENGTH	DS	F	Length in bytes of OUTBUFFER
OUTBUFFER	DS	CL(n)	Buffer containing the converted message
*			

MQRFH2 エlementとして指定されるプロパティ

非メッセージ記述子プロパティを、MQRFH2 ヘッダー・フォルダー中のElementとして指定できます。プロパティとして指定されている MQRFH2 Elementの概要。

こうすると、以前のバージョンの IBM MQ JMS および XMS クライアントとの互換性が保たれます。このセクションでは、MQRFH2 ヘッダー中でプロパティを指定する方法について説明します。

プロパティとして MQRFH2 Elementを使用するには、[IBM MQ classes for Java の使用](#)で説明されているようにElementを指定してください。この情報は、503 ページの『MQRFH2 - 規則および書式ヘッダー 2』で説明されている情報の補足です。

プロパティ・データ・タイプの MQRFH2 データ・タイプへのマッピング

このトピックでは、対応する MQRFH2 データ・タイプにマップされるメッセージ・プロパティ・タイプについて説明します。

表 671. サポートされている MQRFH2 データ・タイプ	
メッセージ・プロパティ・タイプ	MQRFH2 データ・タイプ
MQBYTE[]	bin.hex
MQBOOL	boolean
MQINT8	i1
MQINT16	i2
MQINT32	i4
MQINT64	i8
MQFLOAT32	r4
MQFLOAT64	r8
MQCHAR[]	ストリング

データ・タイプのないElementのタイプは「string」とすると想定されます。

MQRFH2 のデータ・タイプが int の場合は、サイズが指定されていない整数を意味し、i8 であるかのよう扱われます。

ヌル値は、Element属性 `xsi:nil='true'` によって示されます。NULL 以外の値に属性 `xsi:nil='false'` を使用しないでください。

例えば、次のプロパティはヌル値を持っています。

```
<NullProperty xsi:nil='true'></NullProperty>
```

バイトや文字ストリングのプロパティに空の値があってもかまいません。この値は、Element値の長さがゼロの MQRFH2 Elementによって表されます。

例えば、次のプロパティは空の値を持っています。

サポートされている MQRFH2 フォルダー

プロパティとしてのメッセージ記述子フィールドの使用の概要。

フォルダー <jms>、<mcd>、<mqext>、および <usr> については、[MQRFH2 ヘッダー](#)および [JMS](#) で説明されています。<usr> フォルダーは、メッセージに関連付けられている任意の JMS アプリケーション定義のプロパティをトランスポートするために使用されます。グループは、<usr> フォルダーでは許可されません。

[MQRFH2 ヘッダーと JMS](#) は、さらに以下のフォルダーをサポートします。

- <mq>

このフォルダーは、IBM MQ によって使用される MQ 定義プロパティでの使用のために予約されています。

- <mq_usr>

このフォルダーは、JMS プロパティの要件を満たしていないなどの理由で JMS ユーザー定義プロパティとしては公開されないアプリケーション定義のプロパティを転送するために使用できます。このフォルダーには、<usr> フォルダーではできないグループを含めることができます。

- content='properties' 属性のマークが付けられているすべてのフォルダー

このようなフォルダーは、コンテンツ内の <mq_usr> フォルダーに相当します。

- <mpps>

このフォルダーは、IBM MQ パブリッシュ/サブスクライブのプロパティに使用されます。

IBM MQ では、すでに WAS/SIB で使用されている以下のフォルダーもサポートしています。

- <sib>

このフォルダーは、WAS/SIB システム・メッセージのプロパティのうち、JMS プロパティとして公開されていないか、または JMS_IBM_* プロパティにマップされているが、WAS/SIB アプリケーションには公開されているもののために使用するよう予約されています。例えば、順方向/逆方向のルーティング・パス・プロパティなどがあります。

少なくとも一部は、バイト配列であるため、JMS プロパティとしては公開できません。アプリケーションがこのフォルダーにプロパティを追加した場合、その値は無視されるか除去されます。

- <sib_usr>

このフォルダーは、サポートされているタイプではないため、JMS ユーザー・プロパティとして公開することのできない WAS/SIB ユーザー・メッセージのプロパティで使用するために予約されています。それらは、WAS/SIB アプリケーションに対しては公開されています。

これらは、SIMessage インターフェースから取得や設定が可能なユーザー・プロパティですが、バイト配列の内容は必須プロパティ値にマップされます。

IBM MQ アプリケーションで任意の bin.hex エlementをこのフォルダーに書き込むと、復元すると期待していたフォーマットではないために、アプリケーション側でたいい IOException を受け取ります。bin.hex 以外のElementを追加した場合は、ClassCastException を受け取ります。

このフォルダーを使用してプロパティを WAS/SIB で使用できるようにしないでください。代わりに、その目的の <usr> フォルダーを使用してください。

- <sib_context>

このフォルダーは、WAS/SIB ユーザー・アプリケーションにも公開されず、JMS プロパティとしても公開されていない、WAS/SIB システム・メッセージのプロパティのために使用されます。これらには、Web サービスや同様のサービスで使用するセキュリティやトランザクションのプロパティが含まれます。

ご使用のアプリケーションでは、このフォルダーにプロパティを追加しないでください。

- <mqema>

このフォルダーは、<mqext> フォルダーの代わりに WAS/SIB によって使用されました。

MQRFH2 のフォルダー名では、大文字小文字を区別します。

大文字か小文字を組み合わせて以下のようなフォルダーは予約済みです。

- 接頭部が mq か wmq のフォルダーは、IBM MQ での使用のために予約済みです。
- 接頭部が sib のフォルダーは、WAS/SIB での使用のために予約済みです。
- <Root> および <Body> フォルダー。予約済みですが、使用されていません。

次のフォルダーは、メッセージ・プロパティーを含むものとしては認識されません。

- <psc>

IBM Integration Bus が、パブリッシュ/サブスクライブ・コマンド・メッセージをブローカーに送るために使用します。

- <pscr>

IBM Integration Bus が、パブリッシュ/サブスクライブ・コマンド・メッセージに応じて、ブローカーからの情報を収納するために使用します。

- IBM によって定義されておらず、content='properties' 属性でマークが付けられていないすべてのフォルダー。

<psc> フォルダーおよび <pscr> フォルダーでは、content='properties' を指定しないでください。これを指定すると、これらのフォルダーはプロパティーとして処理され、IBM Integration Bus が予期されるように機能しなくなる可能性があります。

アプリケーションでプロパティーを持つメッセージを構築する場合、つまり MQRFH2 ヘッダーにおいてプロパティーを持つ MQRFH2 ヘッダーの一つとして認識されるようにするには、そのヘッダーがメッセージのヘッドでチェーニングできるヘッダーの一覧にある必要があります。

MQRFH2 の前には、不定の数の MQH 標準ヘッダーか、あるいは 1 つの MQCIH、MQDLH、MQIIH、MQTM、MQTMC2、または MQXQH が先行します。ストリングまたは MQCFH は、チェーニングできないために構文解析を終了させます。

一つのメッセージに、すべてメッセージ・プロパティーを処理する複数の MQRFH2 ヘッダーを持たせることが可能です。同名のフォルダーも異なるヘッダー内にあれば共存できます。ただし、WAS/SIB などにより制限されている場合は別です。これらのフォルダーがすべて有効なヘッダー内にある場合、それらは 1 つの論理フォルダーとして処理されます。

有効なヘッダー内のフォルダーは、有効でないヘッダー内のフォルダーとマージできませんが、有効なヘッダー内にある同じ名前のフォルダー同士であれば、競合するプロパティーがすべて削除された上で、マージできます。アプリケーションは、メッセージ内のプロパティーのレイアウトに依存できません。

MQRFH2 グループは、ユーザー定義のフォルダー、つまり <wmq>、<jms>、<mcd>、<usr>、<mqext>、<sib>、<sib_usr>、<sib_context> または <mqema> フォルダーではなく、フォルダー内のプロパティーについて解析されます。

IBM 定義のプロパティー・フォルダー内のグループについては、<wmq> および <mq> フォルダーを除いて、プロパティーが構文解析されます。

MQRFH2 フォルダーには、混合した内容を含めることができません。フォルダーまたはグループには、グループかプロパティー、または値のどちらか一方を含めることができますが、両方を含めることはできません。

メッセージのセグメントには、最初のセグメントと後続のセグメントのどちらについても、IBM MQ 定義のプロパティー（メッセージ記述子にあるプロパティーを除く）を含めることができません。そのため、そのようなプロパティーを含むメッセージを MQMF_SEGMENT または MQMF_SEGMENTATION_ALLOWED のいずれかを指定して書き込むと、その書き込みは MQRC_SEGMENTATION_NOT_ALLOWED を出して失敗します。

ただし、メッセージ・グループに IBM MQ 定義のプロパティーを含めることはできます。

MQRFH2 ヘッダーの生成

IBM MQ がメッセージ・プロパティをその MQRFH2 表記に変換する場合、MQRFH2 をそのメッセージに追加する必要があります。これは MQRFH2 を個別のヘッダーとして追加するか、既存のヘッダーとマージします。

新しい MQRFH2 ヘッダーを IBM MQ により生成すると、メッセージ内の既存のヘッダーが破損する場合があります。ヘッダーのメッセージ・バッファを構文解析するアプリケーションでは、バッファ内のヘッダーの数と位置が状況によっては変更される可能性があることを考慮する必要があります。IBM MQ は、可能な場合はメッセージ・プロパティを既存の MQRFH2 ヘッダーにマージすることにより、メッセージにプロパティを追加することによる影響をなるべく少なくしようとします。また、生成された MQRFH2 をメッセージ・バッファ内の他のヘッダーに対して決まった位置に挿入することにより、影響をなるべく少なくなるようにします。

生成された MQRFH2 ヘッダーは、MQMD の後の、任意の数の MQXQH、MQRFH、および MQDLH ヘッダーの後に（これらのヘッダーの順序には関係なく）置かれます。生成された MQRFH2 ヘッダーは、MQMD、MQXQH、MQDLH、および MQRFH 以外のヘッダーの、最初のヘッダーの直前に置かれます。

 z/OS システムでは、生成された MQRFH2 ヘッダーはアプリケーションの CCSID 内に作成されます。これは以下のように定義されます。

- DLL インターフェースを使用するバッチ LE アプリケーションの場合、CCSID は、**MQCONN** の発行時の現行ロケールに関連付けられている CODESET です（デフォルト値は 1047）。
- バッチ MQ スタブの 1 つとバインドされているバッチ LE アプリケーションの場合、CCSID は、**MQCONN** の後の最初の MQI 呼び出しの発行時の現行ロケールに関連付けられている CODESET です（デフォルト値は 1047）。
- (z/OS UNIX)スレッド z/OS UNIX System Services 上で実行されるバッチ非 LE アプリケーションの場合、は CCSID、が実行された後に **MQCONN** 最初に発行された MQI 呼び出し時のの THLICCSID 値となります（デフォルト値は 1047 です）。
- その他のバッチ・アプリケーションの場合、CCSID は、キュー・マネージャーの CCSID です。

LE アプリケーションの場合、ロケールは `setlocale()` / `CEESETL` LE 呼び出し可能サービスを使用して変更できます。z/OS UNIX スレッドで実行されている非 LE アプリケーションの場合、THLICCSID の値は z/OS UNIX マッピング・マクロ **BPXYTHLI** を使用して変更できます。

生成された MQRFH2 のマージ規則

生成された MQRFH2 を既存の MQRFH2 にマージする場合は、以下の規則が適用されます。生成された MQRFH2 ヘッダーは、次の場合に既存の MQRFH2 ヘッダーとマージされます。

1. 既存の MQRFH2 が、ヘッダー・チェーン内で IBM MQ が生成された MQRFH2 を置く位置、またはそれより前にある場合。
2. 生成されたプロパティの CCSID が、既存の MQRFH2 の NameValueCCSID と同じである場合。

それ以外の場合は、生成されたヘッダーは、バッファ内の前述の位置に個別に置かれます。

既存の MQRFH2 内でのフォルダーのマージ規則

メッセージ・プロパティが既存の MQRFH2 にマージされると、そのメッセージ・プロパティに一致するフォルダーが既存の MQRFH2 から検索され、それらがマージされます。一致するフォルダーが存在しない場合は、既存のフォルダーの最後に新規フォルダーが追加されます。一致するフォルダーが存在する場合は、そのフォルダー内が検索されます。一致するプロパティはすべて上書きされます。新しいプロパティはすべてフォルダーの最後に追加されます。

MQRFH2 フォルダーの制約事項

MQRFH2 ヘッダーのフォルダー制約事項について

MQRFH2 制約事項が以下のフォルダーに適用されます。

- **<usr>** フォルダー内のエレメント名は、接頭部 JMS で始まってはなりません。そのようなプロパティ名は、JMS で使用するために予約されており、ユーザー定義プロパティに対しては無効です。

このようなエレメント名でも、MQRFH2 の構文解析で障害が起こることはありませんが、IBM MQ メッセージ・プロパティ API はこれにアクセスできません。

- **<usr>** フォルダー内のエレメント名は、小文字、大文字、NULL、TRUE、FALSE、NOT、AND、OR、BETWEEN、LIKE、IN、IS、および ESCAPE のいずれかの混合で使用することはできません。これらの名前は SQL キーワードと一致し、構文解析セクターをさらに難しくしています。**<usr>** は、セクター内の特定のプロパティに対してフォルダーが指定されていない場合に使用されるデフォルト・フォルダーです。

このようなエレメント名でも、MQRFH2 の構文解析で障害が起こることはありませんが、IBM MQ メッセージ・プロパティ API はこれにアクセスできません。

- **<usr>** フォルダーのコンテンツ・モデルは以下のとおりです。
 - コロンを含んでいなければ、任意の有効な XML 名をエレメント名として使用できる。
 - ネストされたフォルダーではない単純なエレメントのみを指定できる。
 - **dt="xxx"** 属性によって変更されていない限り、すべてのエレメントはデフォルト・タイプのストリングをとる。
 - すべてのエレメントはオプションであるが、フォルダー内に 2 回以上出現してはならない。
- メッセージ・プロパティを含むと見なされるフォルダー内のエレメント名には、ピリオド (.) を含めることはできません。(Unicode 文字 U+002E)、これは階層を示すためにプロパティ名で使用されるためです。

このようなエレメント名でも、MQRFH2 の構文解析で障害が起こることはありませんが、IBM MQ メッセージ・プロパティ API はこれにアクセスできません。

一般的に、有効な XML スタイル・データを持つ MQRFH2 ヘッダーであれば、IBM MQ で問題なく構文解析することができますが、MQRFH2 の一部のエレメントに IBM MQ メッセージ・プロパティ API からアクセスすることはできません。

MQRFH2 エレメント名における競合

MQRFH2 エレメント名内における競合について

メッセージ・プロパティに付加できる値は 1 つだけです。プロパティにアクセスしようとして値の競合が発生すると、1 つの値が別の値に優先して選択されます。

MQRFH2 エレメントにアクセスするための IBM MQ の構文では、フォルダー内に同じ名前のエレメントが複数存在しない限り、エレメントを一意に識別できます。フォルダーに同じ名前を持つ複数のエレメントがある場合、メッセージ・ヘッドに最も近いプロパティの値が使用されます。

これは、同じメッセージのうち異なる有効な MQRFH2 ヘッダーに、同じ名前のフォルダーが 2 つ以上存在する場合に当てはまります。

非メッセージ記述子プロパティが 2 回 (MQSETMP 呼び出しで 1 回と、未加工の MQRFH2 ヘッダー内で直接に 1 回) 設定された後、MQGET 呼び出しが処理されると、競合が起こることがあります。

これが発生した場合、API 呼び出しによるメッセージと関連付けられているプロパティが、メッセージ・データにあるもの (つまり、未加工の MQRFH2 ヘッダーにあるもの) よりも優先されます。競合が生じた場合、それは論理的にメッセージ・データより前に来たものと見なされます。

プロパティ名を MQRFH2 フォルダー名とエレメント名にマップする

MQRFH2 ヘッダーにおけるプロパティ名とエレメント名の違いについて

メッセージ・プロパティ (例えば MQ JMS) を指定するために、最終的に MQRFH2 ヘッダーを生成する定義済み API のいずれかを使用する場合、プロパティ名は必ずしも MQRFH2 フォルダー内のエレメント名であるとは限りません。

そのため、プロパティー名から MQRFH2 エlement へのマッピングと、その逆方向のマッピングが発生します。その際、Element を含むフォルダー名と、Element 名の両方を考慮に入れることになります。IBM MQ classes for JMS の例のいくつかについては、[使用 IBM MQ classes for Java](#) で既に説明しました。

表 672. プロパティー名から MQRFH2 フォルダー名および Element 名へのマッピング		
プロパティー名	MQRFH2 フォルダー名	MQRFH2 Element 名
JMSDestination	jms	Dst
JMSType	mcd	Type, Set, Fmt
xxx (ユーザー定義、この xxx は JMS で始まりません。)	usr	xxx

したがって、JMS アプリケーションが JMSDestination プロパティーにアクセスするときには、<jms> フォルダー内の Dst Element にマップされます。

プロパティーを MQRFH2 Element として指定するとき、IBM MQ はその Element を次のように定義します。

表 673. プロパティー名から MQRFH2 フォルダー名、グループ名、および Element 名へのマッピング			
プロパティー名	MQRFH2 フォルダー名	MQRFH2 グループ名	MQRFH2 Element 名
<Property>	<usr>	n/a	<Property>
<folder>.<Property>	<folder>	n/a	<Property>
<folder>.<group>.<Property>	<folder>	<group>	<Property>

例えば、IBM MQ アプリケーションが Property1 プロパティーにアクセスしようとする、<usr> フォルダー内の Property1 Element にマップされます。wmq.Property2 プロパティーは、<wmq> フォルダー内の Property2 プロパティーにマップされます。

プロパティー名に複数の名前が含まれている場合、使用される MQRFH2 Element 名は、最後の文字の後の名前です。文字、および MQRFH2 グループを使用して階層を形成します。ネストされた MQRFH2 グループは許可されます。

<mcd>、<jms>、および <mqext> フォルダー内の MQRFH2 に含まれている JMS ヘッダーおよびプロバイダー固有のプロパティーは、[IBM MQ classes for Java の使用](#) で定義されている短縮名を使用して、IBM MQ アプリケーションによってアクセスされます。

JMS ユーザー定義プロパティーには、<usr> フォルダーからアクセスします。IBM MQ アプリケーションは、そのアプリケーション・プロパティーに <usr> フォルダーを使用できます。これは、そのプロパティーがユーザー定義プロパティーの 1 つとして JMS アプリケーションに表示されることが許可されている場合に使用できます。

それが受け入れられない場合は、別のフォルダーを選択します。<wmq_usr> フォルダーは、そのような非 JMS プロパティーの標準ロケーションとして提供されます。

926 ページの『MQRFH2 Element として指定されるプロパティー』で説明されていない方法であっても、以下の点に留意すれば、適切に定義された使用方法でアプリケーションから MQRFH2 フォルダーを指定し、使用することができます。

1. フォルダーは、その内部に含まれるプロパティーに対して別のアプリケーションが未定義アクセスを提供することにより、既に使用中になっているか、将来使用される可能性があります。[プロパティー名](#)を参照して、提案されているプロパティー名の命名規則について調べてください。
2. このプロパティーは、以前のバージョンの IBM MQ classes for JMS や XMS クライアントからはアクセスできません。<usr> フォルダーには、ユーザー定義のプロパティーのためにのみアクセスできます。
3. このフォルダーは、属性 content に値 properties を設定してマークを付ける必要があります (例: content='properties')。

773 ページの『MQSETMP - メッセージ・プロパティーの設定』では、必要に応じて自動的にこの属性が追加されます。この属性は、IBM 定義のフォルダー (<jms> や <usr> など) に追加してはなりません。

ん。これを行うと、IBM WebSphere MQ 7.0 の前に IBM MQ classes for JMS クライアントによってメッセージが拒否されます。MessageFormatException を指定します。

<usr> フォルダーは、<Property> 構文のプロパティのデフォルト・ロケーションであるため、IBM MQ アプリケーションと JMS アプリケーションは、同じユーザー定義プロパティ値に同じ名前を使用してアクセスすることができます。

予約済みのフォルダー名

予約済みのフォルダー名がいくつかあります。そのような名前をフォルダーの接頭部として使用することはできません。例えば、Root.Property1 という名前は、Root が予約済みであるため、有効なプロパティにアクセスしません。以下は、予約済みのフォルダー名のリストです。

- Root
- Body
- プロパティ
- Environment
- LocalEnvironment
- DestinationList
- ExceptionList
- InputBody
- InputRoot
- InputProperties
- InputLocalEnvironment
- InputDestinationList
- InputExceptionList
- OutputRoot
- OutputLocalEnvironment
- OutputDestinationList
- OutputExceptionList

プロパティ記述子フィールドを MQRFH2 ヘッダーにマップする

プロパティが MQRFH2 エlementに変換される場合、プロパティ記述子の重要なフィールドを指定するために、以下のElement属性が使用されます。これは、MQPD フィールドが MQRFH2 Element属性に変換される方法を記述します。

サポート

Support プロパティ記述子のフィールドは、3つのElement属性に分割されます。

- **sr** Element属性は、MQPD_REJECT_UNSUP_MASK ビット・マスクの値を指定します。
- **sa** Element属性は、MQPD_ACCEPT_UNSUP_MASK ビット・マスクの値を指定します。
- **sx** Element属性は、MQPD_ACCEPT_UNSUP_IF_XMIT_MASK ビット・マスクの値を指定します。

これらのElement属性は、<mq> フォルダーにおいてのみ有効で、プロパティを持つ他のフォルダーのElementに設定しても無視されます。

表 674. MQPD フィールドから MQRFH2 エlement属性へのマッピング		
Support の値	MQRFH2 エlement属性	MQRFH2 属性値
MQPD_SUPPORT_OPTIONAL	sa	オプション これがデフォルト値です。
MQPD_SUPPORT_REQUIRED	sr	required
MQPD_SUPPORT_REQUIRED_IF_LOCAL	sx	ローカル

Context

この **context** Element属性を使用して、プロパティが所属するメッセージ・コンテキストを示します。一つの値のみ使用します。このElement属性は、プロパティを持つどのフォルダーのプロパティにおいても有効です。

表 675. context 値から MQRFH2 属性値へのマッピング	
Context 値	MQRFH2 属性値
MQPD_NO_CONTEXT	なし これがデフォルト値です。
MQPD_USER_CONTEXT	ユーザー

CopyOptions

この **copy** Element属性を使用して、プロパティのコピー先となるメッセージを示します。複数の値が許可されています。複数の値の場合はコンマで区切ってください。例えば、**copy='reply'** および **copy='publish,report'** はどちらも有効です。このElement属性は、プロパティを持つどのフォルダーのプロパティにおいても有効です。

注: 属性定義では、単一引用符または二重引用符を使用できます。例えば、**copy='reply'** や **copy="report"** のようにします。

表 676. CopyOption 値から MQRFH2 属性値へのマッピング	
CopyOption 値	MQRFH2 属性値
MQPD_COPY_FORWARD	forward
MQPD_COPY_REPLY	reply
MQPD_COPY_REPORT	レポート
MQPD_COPY_PUBLISH	publish
MQPD_COPY_ALL	すべて 他の値でこれを指定しないでください。別の値で使用する と、これが none を除く他のどの値よりも優先します。
MQPD_COPY_DEFAULT	default これがデフォルト値です。これは MQCOPY_FORWARD、 MQCOPY_REPORT、MQCOPY_PUBLISH の 3 つの値を指定する のと同じことになります。 他の値でこれを指定しないでください。

表 676. CopyOption 値から MQRFH2 属性値へのマッピング (続き)	
CopyOption 値	MQRFH2 属性値
MQPD_COPY_NONE	なし 他の値でこれを指定しないでください。他の値で使用する と、これが優先します。

<mq> MQRFH2 フォルダーに対する制約事項

メッセージがキューに置かれると、MQ 定義プロパティに沿ってメッセージが処理できるように、<mq> フォルダーが検索されます。MQ 定義プロパティの効率の良い構文解析を実現するために、以下の制約事項がこのフォルダーに適用されます。

- メッセージ内で有効な最初の <mq> フォルダーにあるプロパティのみが MQ の処理の対象となり、メッセージにある他の <mq> フォルダーのプロパティは無視されます。
- フォルダーが UTF-8 の場合、1 バイトの UTF-8 文字だけがこのフォルダー内で許可されています。このフォルダーにマルチバイトの文字が一つでもあると、構文解析は失敗し、メッセージは拒否されます。
- MQRFH2 グループをこの <mq> フォルダーの中に入れないでください。プロパティ値に Unicode 文字 U+003C が存在すると、メッセージが拒否されます。
- このフォルダーでエスケープ・ストリングを使用しないでください。エスケープ・ストリングは、エレメントの実際の値として扱われてしまいます。
- Unicode 文字 U+0020 だけが、フォルダー内で空白文字として扱われます。その他の文字はすべて有効として扱われ、フォルダー解析の失敗やメッセージの拒否の原因となる場合があります。

<mq> フォルダーの構文解析が失敗した場合、またはフォルダーがこれらの制限を遵守していない場合、メッセージは CompCode **MQCC_FAILED** および Reason **MQRC_RFH_RESTRICTED_FORMAT_ERR** で拒否されます。

無効な MQRFH2 ヘッダー

MQPUT、MQPUT1、または MQGET 呼び出しの処理の時点で、メッセージにある MQRFH2 ヘッダーの部分構文解析が実行され、中にあるフォルダーがチェックされ、フォルダーの中にプロパティがあるかが判別されます。無効な MQRFH2 ヘッダーの概要があります。

StrucLength フィールドが小さ過ぎるなどの理由で、構造が無効であるためにメッセージの部分構文解析が正常に完了できない場合は、以下のようになります。

- アプリケーションに IBM WebSphere MQ 7 オプションが含まれていると判別できた場合は、既存のアプリケーションが失敗しないように、MQPUT または MQPUT1 呼び出しが、理由コード **MQRC_RFH_ERROR** で失敗します。
- MQGET 呼び出しが正常に戻った場合は、エラーがある MQRFH2 は指定されたバッファーに戻されます。

特定のフォルダーにプロパティが含まれているかどうかを検出できないために部分的な構文解析が失敗した場合 (例えば、フォルダーが <<jms で始まる場合)、フォルダー名が判別される前に構文解析が失敗します。以下のようになります。

- アプリケーションに IBM WebSphere MQ 7 オプションが含まれていると判別できた場合は、既存のアプリケーションが失敗しないように、MQPUT または MQPUT1 呼び出しが、理由コード **MQRC_RFH_FORMAT_ERROR** で失敗します。
- MQGET 呼び出しが正常に戻った場合は、エラーがある MQRFH2 は指定されたバッファーに戻されます。
- キュー・マネージャー内部において、正しく形式設定されていないフォルダーが原因でメッセージが拒否されることはありませんが、そのようなフォルダーは中にプロパティが含まれていないかのように常に扱われます。

メッセージ中に以下に示す条件に当てはまるフォルダーが 1 つ以上含まれている場合、そのメッセージの 1 つのフォルダーに前述のような構文エラーが含まれていても、それは構文解析されず、検出もされないため、キュー・マネージャー・ネットワークを通過することがあります。

- 有効
- 構文解析成功
- メッセージの処理で使用済み

このため、検出は保証されていません。

アプリケーションの1つが773ページの『MQSETMP - メッセージ・プロパティの設定』またはMQINQMPを使用することによりプロパティにアクセスし、それによりMQRFH2フォルダーが完全に構文解析されることになり、その結果としてエラーが検出されて構文解析を完了できない場合、そのことは、API呼び出しに対する、該当する戻りコードによって示されます。そのフォルダーに、アプリケーションで使えるようにされたプロパティはありません。

MQRFH2フォルダーの構文解析が試行されて完全に行われ、認識されていないエレメント属性または認識されていないデータ・タイプがパーサーによって検出された場合でも、解析は続行され、警告が出されることなく正常に完了します。これは、解析エラーにはなりません。

コード・ページ変換

このセクションでは、コード・セットの名前とCCSID、各国語、z/OS変換、IBM i変換、およびUnicode変換に関するサポートについて説明します。

各国語のセクションごとに、以下の情報をリストしています。

- サポートされるネイティブCCSID
- サポートされないコード・ページ変換

説明中では、以下の用語を使用しています。

AIX
IBM MQ for AIXを示します。

Linux
zSeriesのLinuxIBM MQのためにIntel示しますIBM MQLinux。

IBM i OS/400
IBM MQ for IBM iを示します。

Windows
IBM MQ for Windowsを示します。

z/OS
IBM MQ for z/OSを示します。

データ変換のデフォルトでは、変換は、宛先(受信側)システムで実行されます。

ソース製品が変換をサポートしている場合は、チャンネル属性CONVERTをソースでYESに設定することによって、チャンネルをセットアップしてデータを交換することができます。

注:

1. IBM MQ MQI client 情報の変換はサーバー内で行われるため、サーバーがクライアントCCSIDからサーバーCCSIDへの変換をサポートしている必要があります。
2. 変換には、CSD/PTFによって最新バージョンのIBM MQに追加されたサポートが含まれることがあります。最新のサービス・レベルの内容を確認し、CSD/PTFをインストールしてその変換を有効にする必要があるかどうかを判断してください。
3. IBM MQ キュー・マネージャーのCCSIDは、混合またはSBCSでなければなりません。
4. 一部のCCSID(例えば、AIXでの850など)は、オペレーティング・システムではサポートされていませんが、アプリケーションでは使用でき、IBM MQ キュー・マネージャーのCCSIDとしても設定できます。これは、後方互換性のためにのみ許可されており、関連する変換テーブルがインストールされていなければ、変換は失敗します。

CCSID番号と業界コード・セット名との相互参照を、いくつか936ページの表677に示しています。

関連資料

936 ページの『各国語の相違』

この資料には、IBM MQ でサポートされる言語が記載されています。

コード・セット名および CCSID

コード・セット名および各コード・セット名に対応する CCSID。

 IBM MQ for z/OS が提供する変換の中には、言語ごとの表にリストされていないものもあります。変換の一覧については、[] を [962 ページの表 710](#) 参照してください [962 ページの『z/OS 変換サポート』](#)。

表 677. コード・セット名および CCSID	
コード・セット名	CCSID
ISO 8859-1	819
ISO 8859-2	912
ISO 8859-3	913
ISO 8859-5	915
ISO 8859-6	1089
ISO 8859-7	813
ISO 8859-8	916
ISO 8859-9	920
ISO 8859-13	921
ISO 8859-15 (ユーロ)	923
big5	950
eucJP	954 5050 33722
eucKR	970
eucTW	964
eucCN	1383
PCK	943
GBK	1386
koi8-r	878
UTF8	1208

各国語の相違

この資料には、IBM MQ でサポートされる言語が記載されています。

IBM MQ でサポートされている言語は以下のとおりです。

- 米国英語 - トピック [937 ページの『米国英語』](#) を参照してください
- ドイツ語 - トピック [938 ページの『ドイツ語』](#) を参照してください
- デンマーク語およびノルウェー語 - トピック [938 ページの『デンマーク語およびノルウェー語』](#) を参照してください
- フィンランド語およびスウェーデン語 - トピック [939 ページの『フィンランド語およびスウェーデン語』](#) を参照してください

- ・ イタリア語 - トピック 940 ページの『[イタリア語](#)』を参照してください
- ・ スペイン語 - トピック 941 ページの『[スペイン語](#)』を参照してください
- ・ 英国英語 / ゲール語 - トピック 942 ページの『[英国英語 / ゲール語](#)』を参照してください
- ・ フランス語 - トピック 942 ページの『[フランス語](#)』を参照してください
- ・ マルチリンガル - トピック 943 ページの『[マルチリンガル](#)』を参照してください
- ・ ポルトガル語 - トピック 943 ページの『[ポルトガル語](#)』を参照してください
- ・ アイスランド語 - トピック 944 ページの『[アイスランド語](#)』を参照してください
- ・ 東ヨーロッパの言語 - トピック 945 ページの『[東ヨーロッパの言語](#)』を参照してください
- ・ キリル文字 - トピック 946 ページの『[キリル文字](#)』を参照してください
- ・ エストニア語 - トピック 947 ページの『[エストニア語](#)』を参照してください
- ・ ラトビア語およびリトアニア語 - トピック 948 ページの『[ラトビア語およびリトアニア語](#)』を参照してください
- ・ ウクライナ語 - トピック 949 ページの『[ウクライナ語](#)』を参照してください
- ・ ギリシャ語 - トピック 950 ページの『[ギリシャ語](#)』を参照してください
- ・ トルコ語 - トピック 950 ページの『[トルコ語](#)』を参照してください
- ・ ヘブライ語 - トピック 951 ページの『[ヘブライ語](#)』を参照してください
- ・ ペルシア語 - トピック 953 ページの『[ペルシア語](#)』を参照してください
- ・ ウルドゥー語 - トピック 953 ページの『[ウルドゥー語](#)』を参照してください
- ・ タイ語 - トピック 954 ページの『[タイ語](#)』を参照してください
- ・ ラオ語 - トピック 954 ページの『[ラオ語](#)』を参照してください
- ・ ベトナム語 - トピック 954 ページの『[ベトナム語](#)』を参照してください
- ・ 日本語英数小文字 SBCS - トピック 955 ページの『[日本語ローマ字 SBCS](#)』を参照してください
- ・ 日本語カタカナ SBCS - トピック 956 ページの『[日本語カタカナ SBCS](#)』を参照してください
- ・ 日本語漢字 / ローマ字の混合 - トピック 957 ページの『[日本語漢字 / ローマ字の混合](#)』を参照してください
- ・ 日本語漢字 / カタカナの混合 - トピック 958 ページの『[日本語漢字 / カタカナの混合](#)』を参照してください
- ・ 韓国語 - トピック 960 ページの『[韓国語](#)』を参照してください
- ・ 中国語 (簡体字) - トピック 960 ページの『[中国語 \(簡体字\)](#)』を参照してください
- ・ 中国語 (繁体字) - トピック 961 ページの『[中国語 \(繁体字\)](#)』を参照してください

米国英語

米国英語用の CCSID 変換の詳細。

表 678. サポートされるプラットフォームにおける米国英語のネイティブ CCSID	
プラットフォーム	ネイティブ CCSID
 IBM i  z/OS	37, 924, 1140
 AIX	819, 923, 5348
 Windows	437, 850, 1252, 5348, 858
 Linux	819, 923
Apple クライアント	1275

すべての非クライアント・プラットフォームで、ネイティブ CCSID と別のプラットフォームのネイティブ CCSID の間の変換がサポートされています。ただし、以下の例外があります。

IBM i



コード・ページ:

37

コード・ページ 923、858 に変換しません。

924

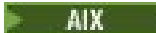
コード・ページ 437、858、1051、1140、1252、1275、5348 に変換しません。

1140

コード・ページ 924、1051、1275 に変換しません。

ドイツ語

ドイツ語用の CCSID 変換の詳細。

表 679. サポートされるプラットフォームにおけるドイツ語のネイティブ CCSID	
プラットフォーム	ネイティブ CCSID
 IBM i	273, 924, 1141
 z/OS	
 AIX	819, 923, 5348
 Windows	437, 850, 858, 1252, 5348
 Linux	819, 923
Apple クライアント	1275

すべての非クライアント・プラットフォームで、ネイティブ CCSID と別のプラットフォームのネイティブ CCSID の間の変換がサポートされています。ただし、以下の例外があります。

IBM i



コード・ページ:

273

コード・ページ 858、923、924、1275 に変換しません。

924

コード・ページ 273、437、858、1051、1141、1252、1275、5348 に変換しません。

1141

コード・ページ 924、1051、1275 に変換しません。

デンマーク語およびノルウェー語

デンマーク語およびノルウェー語用の CCSID 変換の詳細。

表 680. サポートされるプラットフォームにおけるデンマーク語およびノルウェー語のネイティブ CCSID	
プラットフォーム	ネイティブ CCSID
 IBM i  z/OS	277, 924, 1142
 AIX	819, 923, 5348
 Windows	850, 858, 865, 1252, 5348
 Linux	819, 923
Apple クライアント	1275

すべての非クライアント・プラットフォームで、ネイティブ CCSID と別のプラットフォームのネイティブ CCSID の間の変換がサポートされています。ただし、以下の例外があります。

IBM i



コード・ページ:

277

コード・ページ 858、923、924、1275 に変換しません。

924

コード・ページ 277、858、865、1051、1142、1252、1275、5348 に変換しません。

1142

コード・ページ 924、865、1051、1275 に変換しません。

AIX



コード・ページ:

819

コード・ページ 865 に変換しません。

Windows



コード・ページ:

865

コード・ページ 1051、1275 に変換しません。

フィンランド語およびスウェーデン語

フィンランド語およびスウェーデン語用の CCSID 変換の詳細。

表 681. サポートされるプラットフォームにおけるフィンランド語およびスウェーデン語のネイティブ CCSID	
プラットフォーム	ネイティブ CCSID
 IBM i  z/OS	278, 924, 1143

表 681. サポートされるプラットフォームにおけるフィンランド語およびスウェーデン語のネイティブ CCSID (続き)

プラットフォーム	ネイティブ CCSID
 AIX	819, 923, 5348
 Windows	437, 850, 858, 865, 1252, 5348
 Linux	819, 923
Apple クライアント	1275

すべての非クライアント・プラットフォームで、ネイティブ CCSID と別のプラットフォームのネイティブ CCSID の間の変換がサポートされています。ただし、以下の例外があります。

IBM i



コード・ページ:

278

コード・ページ 858、923、924、1275 に変換しません。

924

コード・ページ 278、437、858、865、1051、1143、1252、1275、5348 に変換しません。

1143

コード・ページ 865、924、1051、1275 に変換しません。

AIX



コード・ページ:

819

コード・ページ 865 に変換しません。

850

コード・ページ 865 に変換しません。

Windows



コード・ページ:

865

コード・ページ 1051、1275 に変換しません。

イタリア語

イタリア語用の CCSID 変換の詳細。

表 682. サポートされるプラットフォームにおけるイタリア語のネイティブ CCSID

プラットフォーム	ネイティブ CCSID
 IBM i	280, 924, 1144
 z/OS	
 AIX	819, 923, 5348

表 682. サポートされるプラットフォームにおけるイタリア語のネイティブ CCSID (続き)

プラットフォーム	ネイティブ CCSID
 Windows	437, 850, 858, 1252, 5348
 Linux	819, 923
Apple クライアント	1275

すべての非クライアント・プラットフォームで、ネイティブ CCSID と別のプラットフォームのネイティブ CCSID の間の変換がサポートされています。ただし、以下の例外があります。

IBM i



コード・ページ:

280

コード・ページ 858、923、924、1275 に変換しません。

924

コード・ページ 280、437、858、1051、1144、1252、1275、5348 に変換しません。

1144

コード・ページ 924、1051、1275 に変換しません。

スペイン語

スペイン語用の CCSID 変換の詳細。

表 683. サポートされるプラットフォームにおけるスペイン語のネイティブ CCSID

プラットフォーム	ネイティブ CCSID
 IBM i	284, 924, 1145
 z/OS	
 AIX	819, 923, 5348
 Windows	437, 850, 858, 1252, 5348
 Linux	819, 923
Apple クライアント	1275

すべての非クライアント・プラットフォームで、ネイティブ CCSID と別のプラットフォームのネイティブ CCSID の間の変換がサポートされています。ただし、以下の例外があります。

IBM i



コード・ページ:

284

コード・ページ 858、923、924、1275 に変換しません。

924

コード・ページ 284、437、858、1051、1145、1252、1275、5348 に変換しません。

1145

コード・ページ 924、1051、1275 に変換しません。

英国英語 / ゲール語

英国英語 / ゲール語用の CCSID 変換の詳細。

表 684. サポートされるプラットフォームにおける英国英語 / ゲール語のネイティブ CCSID	
プラットフォーム	ネイティブ CCSID
 IBM i  z/OS	285, 924, 1146
 AIX	819, 923, 5348
 Windows	437, 850, 858, 1252, 5348
 Linux	819, 923
Apple クライアント	1275

すべての非クライアント・プラットフォームで、ネイティブ CCSID と別のプラットフォームのネイティブ CCSID の間の変換がサポートされています。ただし、以下の例外があります。

IBM i



コード・ページ:

285

コード・ページ 858、923、924、1275 に変換しません。

924

コード・ページ 285、437、858、1051、1146、1252、1275、5348 に変換しません。

1146

コード・ページ 924、1051、1275 に変換しません。

フランス語

フランス語用の CCSID 変換の詳細。

表 685. サポートされるプラットフォームにおけるフランス語のネイティブ CCSID	
プラットフォーム	ネイティブ CCSID
 IBM i  z/OS	297, 924, 1147
 AIX	819, 923, 5348
 Windows	437, 850, 858, 1252, 5348
 Linux	819, 923
Apple クライアント	1275

すべての非クライアント・プラットフォームで、ネイティブ CCSID と別のプラットフォームのネイティブ CCSID の間の変換がサポートされています。ただし、以下の例外があります。

IBM i



コード・ページ:

297

コード・ページ 858、923、924、1275、5348 に変換しません。

924

コード・ページ 297、437、858、1051、1147、1252、1275、5348 に変換しません。

1147

コード・ページ 924、1051、1275 に変換しません。

マルチリンガル

マルチリンガル用の CCSID 変換の詳細。

表 686. サポートされるプラットフォームにおけるマルチリンガルのネイティブ CCSID	
プラットフォーム	ネイティブ CCSID
 IBM i	500, 924, 1148
 z/OS	
 AIX	819, 923, 5348
 Windows	437, 850, 858, 1252, 5348
 Linux	819, 923
Apple クライアント	1275

すべての非クライアント・プラットフォームで、ネイティブ CCSID と別のプラットフォームのネイティブ CCSID の間の変換がサポートされています。ただし、以下の例外があります。

IBM i



コード・ページ:

500

コード・ページ 858、923 に変換しません。

924

コード・ページ 437、858、1051、1148、1252、1275、5348 に変換しません。

1148

コード・ページ 924、1051、1275 に変換しません。

ポルトガル語

ポルトガル語用の CCSID 変換の詳細。

表 687. サポートされるプラットフォームにおけるポルトガル語のネイティブ CCSID	
プラットフォーム	ネイティブ CCSID
 IBM i	37, 500, 924, 1140
 z/OS	500, 924, 1140
 AIX	819, 923, 5348

表 687. サポートされるプラットフォームにおけるポルトガル語のネイティブ CCSID (続き)

プラットフォーム	ネイティブ CCSID
 Windows	850, 858, 860, 1252, 5348
 Linux	819, 923
Apple クライアント	1275

すべての非クライアント・プラットフォームで、ネイティブ CCSID と別のプラットフォームのネイティブ CCSID の間の変換がサポートされています。ただし、以下の例外があります。

IBM i



コード・ページ:

37

コード・ページ 858、923、1275 に変換しません。

500

コード・ページ 858、923、1275 に変換しません。

924

コード・ページ 858、860、1051、1140、1252、1275、5348 に変換しません。

1140

コード・ページ 860、924、1051、1275 に変換しません。

Windows



コード・ページ:

860

コード・ページ 1051、1275 に変換しません。

アイスランド語

アイスランド語用の CCSID 変換の詳細。

表 688. サポートされるプラットフォームにおけるアイスランド語のネイティブ CCSID

プラットフォーム	ネイティブ CCSID
 IBM i	871, 924, 1149
 z/OS	
 AIX	819, 923, 5348
 Windows	850, 858, 861, 1252, 5348
 Linux	819, 923
Apple クライアント	1275

すべての非クライアント・プラットフォームで、ネイティブ CCSID と別のプラットフォームのネイティブ CCSID の間の変換がサポートされています。ただし、以下の例外があります。

IBM i



コード・ページ:

871

コード・ページ 858、923、924、1275、5348 に変換しません。

924

コード・ページ 858、861、871、1051、1149、1252、1275、5348 に変換しません。

1149

コード・ページ 924、1051、1275 に変換しません。

Windows



コード・ページ:

861

コード・ページ 1051、1275 に変換しません。

東ヨーロッパの言語

東ヨーロッパの言語用の CCSID 変換の詳細。上記の CCSID を使用する代表的な言語には、アルバニア語、クロアチア語、チェコ語、ハンガリー語、ポーランド語、ルーマニア語、セルビア語、スロバキア語、およびスロベニア語があります。

表 689. サポートされるプラットフォームにおける東ヨーロッパの言語のネイティブ CCSID	
プラットフォーム	ネイティブ CCSID
IBM i z/OS	870, 1153
Windows	852, 1250, 5346, 9044
AIX Linux	912
東ヨーロッパ Apple クライアント	1282
ルーマニア語 Apple クライアント	1285
クロアチア語 Apple クライアント	1284

すべての非クライアント・プラットフォームで、ネイティブ CCSID と別のプラットフォームのネイティブ CCSID の間の変換がサポートされています。ただし、以下の例外があります。

z/OS



コード・ページ:

870

コード・ページ 1284、1285 に変換しません。

1153

コード・ページ 1250、1284、1285 に変換しません。

IBM i



コード・ページ:

870

コード・ページ 1284、1285、5346、9044 に変換しません。

1153

コード・ページ 1282、1284、1285、5346、9044 に変換しません。

Linux



コード・ページ:

912

コード・ページ 1284、1285 に変換しません。

Windows



コード・ページ:

852

コード・ページ 1284、1285 に変換しません。

1250

コード・ページ 1284、1285 に変換しません。

9044

コード・ページ 912、1282、1284、1285 に変換しません。

キリル文字

キリル文字用の CCSID 変換の詳細。これらの CCSID を使用する代表的な言語には、ベラルーシ語、ブルガリア語、マケドニア語、およびセルビア語が含まれます。

表 690. サポートされるプラットフォームにおけるキリル文字のネイティブ CCSID	
プラットフォーム	ネイティブ CCSID
z/OS	1025
IBM i	880, 1025
Windows	855, 866, 1131, 1251, 5347
AIX	915
Linux	
Apple クライアント	1283

すべての非クライアント・プラットフォームで、ネイティブ CCSID と別のプラットフォームのネイティブ CCSID の間の変換がサポートされています。ただし、以下の例外があります。

IBM i



コード・ページ:

880

コード・ページ 855、866、878、1131、5347 に変換しません。

1025

コード・ページ 878、5347 に変換しません。

Windows



コード・ページ:

855

コード・ページ 1131 に変換しません。

866

コード・ページ 1131 に変換しません。

1131

コード・ページ 855、866、880、1283 に変換しません。

エストニア語

エストニア語用の CCSID 変換の詳細。

表 691. サポートされるプラットフォームにおけるエストニア語のネイティブ CCSID	
プラットフォーム	ネイティブ CCSID
 IBM i	1122, 1157
 z/OS	
 Windows	902, 922, 1257, 5353, 9449
 AIX	902, 922
 Linux	

すべてのプラットフォームで、ネイティブ CCSID と別のプラットフォームのネイティブ CCSID の間の変換がサポートされています。ただし、以下の例外があります。

z/OS



コード・ページ:

1122

コード・ページ 902、1157、9449 に変換しません。

1157

コード・ページ 922、1122、1257、9449 に変換しません。

IBM i



コード・ページ:

1122

コード・ページ 902、5353、9449 に変換しません。

1157

コード・ページ 922、5353、9449 に変換しません。

Linux



コード・ページ:

902

コード・ページ 922、1122、9449 に変換しません。

922

コード・ページ 902、1157、9449 に変換しません。

Windows



コード・ページ:

5353

コード・ページ 9449 に変換しません。

9449

コード・ページ 902、922、1122、1157、1257、5353 に変換しません。

902

コード・ページ 922、1122、9449 に変換しません。

ラトビア語およびリトアニア語

ラトビア語およびリトアニア語用の CCSID 変換の詳細。

表 692. サポートされるプラットフォームにおけるラトビア語およびリトアニア語のネイティブ CCSID	
プラットフォーム	ネイティブ CCSID
 IBM i	1112, 1156
 z/OS	
 Windows	901, 921, 1257, 5353, 9449
 AIX	901, 921
 Linux	

すべてのプラットフォームで、ネイティブ CCSID と別のプラットフォームのネイティブ CCSID の間の変換がサポートされています。ただし、以下の例外があります。

z/OS



コード・ページ:

1112

コード・ページ 901、1156、9449 に変換しません。

1156

コード・ページ 901、1156、9449 に変換しません。

IBM i



コード・ページ:

1112

コード・ページ 5353 に変換しません。

1153

コード・ページ 921、5353、9449 に変換しません。

Linux



コード・ページ:

902

コード・ページ 921、1112、1257、9449 に変換しません。

921

コード・ページ 901、1156、9449 に変換しません。

Windows



コード・ページ:

901

コード・ページ 921、1112、1257、9449 に変換しません。

5355

コード・ページ 9449 に変換しません。

9449

コード・ページ 901、921、1112、1156、1257 に変換しません。

ウクライナ語

ウクライナ語用の CCSID 変換の詳細。

表 693. サポートされるプラットフォームにおけるウクライナ語のネイティブ CCSID	
プラットフォーム	ネイティブ CCSID
 IBM i	1123
 z/OS	
 Windows	1124, 1125, 1251, 5347
 AIX	1124
 Linux	

すべてのプラットフォームで、ネイティブ CCSID と別のプラットフォームのネイティブ CCSID の間の変換がサポートされています。ただし、以下の例外があります。

IBM i



コード・ページ:

1123

コード・ページ 5347 に変換しません。

Windows



コード・ページ:

1125

コード・ページ 1123 に変換しません。

ギリシャ語

ギリシャ語用の CCSID 変換の詳細。

表 694. サポートされるプラットフォームにおけるギリシャ語のネイティブ CCSID	
プラットフォーム	ネイティブ CCSID
 IBM i  z/OS	875
 Windows	869, 1253, 5349
 AIX  Linux NCR	813
Apple クライアント	1280
DOS クライアント	737

すべての非クライアント・プラットフォームで、ネイティブ CCSID と別のプラットフォームのネイティブ CCSID の間の変換がサポートされています。ただし、以下の例外があります。

IBM i



コード・ページ:

875

コード・ページ 5349 に変換しません。

Windows



コード・ページ:

1253

コード・ページ 737 に変換しません。

5349

コード・ページ 737 に変換しません。

トルコ語

トルコ語用の CCSID 変換の詳細。

表 695. サポートされるプラットフォームにおけるトルコ語のネイティブ CCSID	
プラットフォーム	ネイティブ CCSID
 IBM i  z/OS	1026

表 695. サポートされるプラットフォームにおけるトルコ語のネイティブ CCSID (続き)	
プラットフォーム	ネイティブ CCSID
 Windows	857, 1254, 5350
 AIX	920
 Linux	
Apple クライアント	1281

すべての非クライアント・プラットフォームで、ネイティブ CCSID と別のプラットフォームのネイティブ CCSID の間の変換がサポートされています。ただし、以下の例外があります。

IBM i



コード・ページ:

1026

コード・ページ 5350 に変換しません。

ヘブライ語

ヘブライ語用の CCSID 変換の詳細。

表 696. サポートされるプラットフォームにおけるヘブライ語のネイティブ CCSID	
プラットフォーム	ネイティブ CCSID
 z/OS	424, 803, 4899, 12712
 IBM i	424
 AIX	916, 9048
 Windows	1255, 5351
 Linux	916

すべてのプラットフォームで、ネイティブ CCSID と別のプラットフォームのネイティブ CCSID の間の変換がサポートされています。ただし、以下の例外があります。

z/OS



コード・ページ:

424

コード・ページ 867、4899、9048、12712 に変換しません。

803

コード・ページ 867、4899、5351、9048、12712 に変換しません。

4899

コード・ページ 424、803、856、862、916、1255 に変換しません。

12712

コード・ページ 424、803、856、916、1255 に変換しません。

IBM i



コード・ページ:

424

コード・ページ 803、867、4899、5351、9048、12712 に変換しません。

コード・ページ 424 と CCSID 4952 の間の変換も行われます。CCSID 4952 は、856 のバリエーションです。

AIX



コード・ページ:

916

コード・ページ 867、4899、9048、12712 に変換しません。

9048

コード・ページ 424、803、856、862、916、1255 に変換しません。

Windows



コード・ページ:

1255

コード・ページ 867、4899、9048、12712 に変換しません。

5351

コード・ページ 803 に変換しません。

アラビア語

アラビア語用の CCSID 変換の詳細。

表 697. サポートされるプラットフォームにおけるアラビア語のネイティブ CCSID		
プラットフォーム		ネイティブ CCSID
IBM i		420
z/OS		
AIX		1046, 1089
Windows		720, 864, 1256, 5352
Linux		1089

すべてのプラットフォームで、ネイティブ CCSID と別のプラットフォームのネイティブ CCSID の間の変換がサポートされています。ただし、以下の例外があります。

IBM i



コード・ページ:

420

コード・ページ 5352 に変換しません。

Linux, Tru64



コード・ページ:

1089

コード・ページ 720 に変換しません。

Windows



コード・ページ:

720

コード・ページ 1089、5352 に変換しません。

5352

コード・ページ 720 に変換しません。

ペルシア語

ペルシア語用の CCSID 変換の詳細。

表 698. サポートされるプラットフォームにおけるペルシア語のネイティブ CCSID	
プラットフォーム	ネイティブ CCSID
 IBM i  z/OS	1097
 AIX  Linux  Windows	1098 (注を参照)

注: これらのプラットフォームのネイティブ CCSID は標準化されておらず、変更される場合があります。

すべてのプラットフォームで、ネイティブ CCSID と別のプラットフォームのネイティブ CCSID の間の変換がサポートされています。

ウルドゥー語

ウルドゥー語用の CCSID 変換の詳細。

表 699. サポートされるプラットフォームにおけるウルドゥー語のネイティブ CCSID	
プラットフォーム	ネイティブ CCSID
 IBM i  z/OS	918
 Windows	868
 AIX  Linux	1006

すべてのプラットフォームで、ネイティブ CCSID と別のプラットフォームのネイティブ CCSID の間の変換がサポートされています。ただし、以下の例外があります。

IBM i



コード・ページ:

918

コード・ページ 1006 に変換しません。

タイ語

タイ語用の CCSID 変換の詳細。

表 700. サポートされるプラットフォームにおけるタイ語のネイティブ CCSID	
プラットフォーム	ネイティブ CCSID
IBM i z/OS	838
AIX Linux Windows	874 (注を参照)

注: これらのプラットフォームのネイティブ CCSID は標準化されておらず、変更される場合があります。

すべてのプラットフォームで、ネイティブ CCSID と別のプラットフォームのネイティブ CCSID の間の変換がサポートされています。

ラオ語

ラオ語用の CCSID 変換の詳細。

表 701. サポートされるプラットフォームにおけるラオ語のネイティブ CCSID	
プラットフォーム	ネイティブ CCSID
IBM i z/OS	1132
AIX Linux Windows	1133

すべてのプラットフォームで、ネイティブ CCSID と別のプラットフォームのネイティブ CCSID の間の変換がサポートされています。

ベトナム語

ベトナム語用の CCSID 変換の詳細。

表 702. サポートされるプラットフォームにおけるベトナム語のネイティブ CCSID	
プラットフォーム	ネイティブ CCSID
IBM i z/OS	1130

表 702. サポートされるプラットフォームにおけるベトナム語のネイティブ CCSID (続き)

プラットフォーム	ネイティブ CCSID
 Windows	1258, 5354
 AIX	1129
 Linux	

すべてのプラットフォームで、ネイティブ CCSID と別のプラットフォームのネイティブ CCSID の間の変換がサポートされています。ただし、以下の例外があります。

IBM i



コード・ページ:

1130

コード・ページ 1129、5354 に変換しません。

日本語ローマ字 SBCS

日本語ローマ字 SBCS の場合の CCSID 間変換の詳細。

表 703. サポートされるプラットフォームにおける日本語ローマ字のネイティブ CCSID

プラットフォーム	ネイティブ CCSID
 IBM i	1027
 z/OS	
 AIX	932, 5050, 33722 (注 1 を参照)
 Windows	932, 943 (注 2 を参照)
 Linux	943, 5050

注:

-  5050 および 33722 は、AIX の基本コード・ページ 954 に関連する CCSID です。オペレーティング・システムにより報告される CCSID は 33722 です。
-  Windows NT はコード・ページ 932 を使用していますが、最も近い表現を備えた CCSID は 943 です。ただし、IBM MQ の一部のプラットフォームでは、この CCSID がサポートされない場合があります。

コード・ページ 932 を表すために IBM MQ for Windows CCSID 932 が使用されますが、ファイル ../conv/table/ccsid.tbl への変更を行うことができます。これにより、使用される CCSID が 943 に変更されます。

すべてのプラットフォームで、ネイティブ CCSID と別のプラットフォームのネイティブ CCSID の間の変換がサポートされています。ただし、以下の例外があります。

z/OS



コード・ページ:

1027

コード・ページ 932、942、943、954、5050、33722 に変換しません。

IBM i



コード・ページ:

1027

コード・ページ 932 に変換しません。

AIX



コード・ページ:

932

コード・ページ 1027 に変換しません。

5050

コード・ページ 1027 に変換しません。

33722

コード・ページ 1027 に変換しません。

Linux



コード・ページ:

943

コード・ページ 1027 に変換しません。

5050

コード・ページ 1027 に変換しません。

日本語カタカナ SBCS

日本語カタカナ SBCS の場合の CCSID 間変換の詳細。

表 704. サポートされるプラットフォームにおける日本語カタカナのネイティブ CCSID	
プラットフォーム	ネイティブ CCSID
IBM i	290
z/OS	
AIX	932, 5050, 33722 (注 1 を参照)
Windows	932, 943 (注 2 を参照)
Linux	943, 5050

注:

- 5050 および 33722 は、AIX の基本コード・ページ 954 に関連する CCSID です。オペレーティング・システムにより報告される CCSID は 33722 です。
- Windows NT はコード・ページ 932 を使用していますが、最も近い表現を備えた CCSID は 943 です。ただし、IBM MQ の一部のプラットフォームでは、この CCSID がサポートされない場合があります。

コード・ページ 932 を表すために IBM MQ for Windows CCSID 932 が使用されますが、ファイル ../conv/table/ccsid.tbl への変更を行うことができます。これにより、使用される CCSID が 943 に変更されます。

3. 上記の変換以外にも、IBM MQ は以下のプラットフォームで、CCSID 897 から CCSID 37、273、277、278、280、284、285、290、297、437、500、819、850、1027、および 1252 への変換をサポートしています。

-  AIX
-  Linux

すべてのプラットフォームで、ネイティブ CCSID と別のプラットフォームのネイティブ CCSID の間の変換がサポートされています。ただし、以下の例外があります。

z/OS



コード・ページ:

290

コード・ページ 932、943、954、5050、33722 に変換しません。

IBM i



コード・ページ:

290

コード・ページ 932 に変換しません。

AIX



コード・ページ:

932

コード・ページ 290、897 に変換しません。

5050

コード・ページ 290、897 に変換しません。

33722

コード・ページ 290、897 に変換しません。

Linux



コード・ページ:

943

コード・ページ 290、897 に変換しません。

5050

コード・ページ 290、897 に変換しません。

日本語漢字 / ローマ字の混合

日本語漢字 / ローマ字の混合の場合の CCSID 間変換の詳細

表 705. サポートされるプラットフォームにおける日本語漢字 / ローマ字の混合のネイティブ CCSID	
プラットフォーム	ネイティブ CCSID
 IBM i  z/OS	1399、5035 (958 ページの『1』)
 AIX	932、5050、33722 (958 ページの『2』)
 Windows	932、943 (958 ページの『3』)
 Linux	943、5050

注：

1.   5035 は、コード・ページ 939 に対応する CCSID です
2.  5050 および 33722 は、AIX の基本コード・ページ 954 に関連する CCSID です。オペレーティング・システムにより報告される CCSID は 33722 です。
3.  Windows NT はコード・ページ 932 を使用していますが、最も近い表現を備えた CCSID は 943 です。ただし、IBM MQ の一部のプラットフォームでは、この CCSID がサポートされない場合があります。

コード・ページ 932 を表すために IBM MQ for Windows CCSID 932 が使用されますが、ファイル ../conv/table/ccsid.tbl への変更を行うことができます。これにより、使用される CCSID が 943 に変更されます。

すべてのプラットフォームで、ネイティブ CCSID と別のプラットフォームのネイティブ CCSID の間の変換がサポートされています。ただし、以下の例外があります。

z/OS



コード・ページ:

1399

コード・ページ 954、5035、5050、33722 に変換しません。

5035

コード・ページ 954、1399、5050、33722 に変換しません。

IBM i



コード・ページ:

1399

コード・ページ 5039 に変換しません。

5035

コード・ページ 5039 に変換しません。

日本語漢字 / カタカナの混合

日本語漢字 / カタカナの混合の場合の CCSID 間変換の詳細

表 706. サポートされるプラットフォームにおける日本語漢字 / カタカナの混合のネイティブ CCSID	
プラットフォーム	ネイティブ CCSID
 z/OS	1390, 5026 (注参照 959 ページの『1』)
 IBM i	5026 (注参照 959 ページの『1』)
 AIX	932, 5050, 33722 (注参照 959 ページの『2』)
 Windows	932, 943 (注参照 959 ページの『3』)
 Linux	943, 5050

注:

1.   EBCDIC における CCSID 1390 および 5026 のシングルバイトモードでは、基本ラテン文字の標準レイアウトまたは不変レイアウトとは異なる位置に小文字が含まれる。したがって、メッセージデータを他の CCSID に変換する際にデータが失われないよう注意を払わなければなりません。また、これらの CCSID をキューマネージャのデフォルト CCSID として使用すると、他のキューマネージャとの通信時に問題が発生する可能性があります。たとえば、小文字を使用したチャンネル名は、リモートシステムで正しく解釈されない可能性があります。5026 は、コード・ページ 930 に対応する CCSID です。CCSID 5026 は、IBM i で日本語カタカナ (DBCS) 機能を選択したときに報告される CCSID です。
2.  5050 および 33722 は、AIX の基本コード・ページ 954 に関連する CCSID です。オペレーティング・システムにより報告される CCSID は 33722 です。
3.  Windows NT はコード・ページ 932 を使用していますが、最も近い表現を備えた CCSID は 943 です。ただし、IBM MQ の一部のプラットフォームでは、この CCSID がサポートされない場合があります。

Windows では、コード・ページ 932 を表すために CCSID 932 が使用されていますが、ファイル ../conv/table/ccsid.tbl を変更することで、使用されている CCSID を 943 に変更することが可能です。

すべてのプラットフォームで、ネイティブ CCSID と別のプラットフォームのネイティブ CCSID の間の変換がサポートされています。ただし、以下の例外があります。

z/OS



コード・ページ:

1390

コード・ページ 954、5026、5050、33722 に変換しません。

小文字を受け入れません。

5026

コード・ページ 954、1390、5050、33722 に変換しません。

IBM i



コード・ページ:

5026

コード・ページ 1390、5039 に変換しません。

韓国語

韓国語の場合の CCSID 間変換の詳細

表 707. サポートされるプラットフォームにおける韓国語のネイティブ CCSID	
プラットフォーム	ネイティブ CCSID
 IBM i  z/OS	933, 1364
 AIX  Linux	970
 Windows	949, 1363

すべてのプラットフォームで、ネイティブ CCSID と別のプラットフォームのネイティブ CCSID の間の変換がサポートされています。ただし、以下の例外があります。

z/OS



コード・ページ:

933

コード・ページ 970 に変換しません。

1364

コード・ページ 970 に変換しません。

中国語 (簡体字)

中国語 (簡体字) の場合の CCSID 間変換の詳細

表 708. サポートされるプラットフォームにおける中国語 (簡体字) のネイティブ CCSID	
プラットフォーム	ネイティブ CCSID
 z/OS	935, 1388
 IBM i	935, 1388
 AIX	1383, 1386
 Windows	1381、1386(注 2 を参照)
 Linux	1383

注:

-  Windows はコード・ページ 936 を使用していますが、最も近い表現を備えた CCSID は 1386 です。ただし、IBM MQ の一部のプラットフォームでは、この CCSID がサポートされない場合があります。

IBM MQ for Windows では、CCSID 1381 を使用してコード・ページ 936 を表現していますが、ファイル ../conv/table/ccsid.tbl を変更して、CCSID 1386 を使用することもできます。

- IBM MQ は、中国語 GB18030 規格をサポートします。

   以下の日付において LinuxWindowsz/OS、Unicode (UTF-8 および UTF-16) と CCSID 1388 (GB18030 拡張機能付き EBCDIC)、Unicode (UTF-8 および UTF-16)

と CCSID 5488 (GB18030)、ならびに CCSID 1388 と CCSID 5488 間の変換サポートが提供されます。

注:  マルチプラットフォームでは、キューやトピックなどのオブジェクトの説明フィールドで GB18030 文字を使用するには、CCSID を 5488 に設定する必要があります。ただし、または IBM MQ Explorer で作成された IBM MQ Console キュー・マネージャでは、CCSID を設定することはできません。代わりに、CCSID 5488 で CLI を使用してキューマネージャを作成するか、キューマネージャ作成後に CLI コマンドラインで CCSID を変更する必要があります。

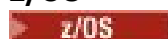
 IBM i では、Unicode (UTF-8 および UTF-16) と CCSID 1388 (GB18030 拡張機能を持つ EBCDIC) の間の変換サポートがオペレーティング・システムによって提供されています。

 GB18030 の文字をキューやトピックなどのオブジェクトの説明フィールドで使用するには、キューマネージャを [QMCCSID 値 1388](#) で実行するように設定する必要があります z/OS。

 から IBM MQ 10.0.0z/OS、GB18030-2022 レベル 1 およびレベル 2 文字 (CCSID 1388) と Unicode (UTF-8、CCSID 1208、および UTF-16、CCSID 1200) 間の変換サポートが提供されます。

すべてのプラットフォームで、ネイティブ CCSID と別のプラットフォームのネイティブ CCSID の間の変換がサポートされています。ただし、以下の例外があります。

z/OS



コード・ページ:

935

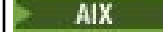
コード・ページ 1383 に変換しません。

1388

コード・ページ 1383 に変換しません。

中国語 (繁体字)

中国語 (繁体字) の場合の CCSID 間変換の詳細

表 709. サポートされるプラットフォームにおける中国語 (繁体字) のネイティブ CCSID	
プラットフォーム	ネイティブ CCSID
 IBM i  z/OS	937
 Windows	950
 AIX  Linux	950, 964

すべてのプラットフォームで、ネイティブ CCSID と別のプラットフォームのネイティブ CCSID の間の変換がサポートされています。ただし、以下の例外があります。

z/OS



コード・ページ:

937

コード・ページ 964 に変換しません。

Linux



コード・ページ:

964

コード・ページ 938 に変換しません。

z/OS z/OS 変換サポート

サポートされている CCSID 変換に関する情報。これには、データを変換可能な CCSID 間の変換対象と、発生する可能性のある文字の違いの一部が含まれます

サポートされている変換

以下の表は、各 CCSID が変換可能な他の CCSID への対応関係を示しています。

注: [V 5R0.0](#) 一部の Java バージョンでは、いくつかの CCSID が廃止されています。詳細については、「[サポート終了した CSSID](#)」を参照してください。

表 710. IBM MQ for z/OS CCSID 変換サポート	
CCSID	変換元および変換先の CCSID
37	256, 273, 275, 277-278, 280, 284-285, 290, 297, 367, 420, 423-424, 437, 500, 720, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-866, 869-871, 874-875, 880, 897, 903-905, 912, 914-916, 920-924, 1009, 1025-1027, 1040-1043, 1047, 1051, 1088, 1097, 1100, 1112, 1114-1115, 1122, 1124, 1126, 1130-1132, 1137, 1140-1149, 1200, 1208, 1250-1255, 1257-1258, 1275, 1280-1281, 1283, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5210-5211, 5346, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 16804, 17248, 17584, 25473, 25479, 25480, 25617, 25619, 25664, 28709
256	37, 273, 277-278, 280, 284-285, 290, 297, 367, 420, 423-424, 437, 500, 737, 775, 819, 833, 836, 838, 850, 852, 857, 860-866, 869-871, 875, 880, 905, 1025-1027, 1112, 1122, 1200, 1208, 1251-1252, 1275, 4386, 4929, 4932, 4934, 4946, 4948, 4953, 4960, 4971, 5123, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9056, 9061, 13121, 13488, 16804, 17248, 17584, 28709
259	437, 808, 850-852, 855-858, 860-865, 867, 869, 872, 874, 899, 901-902, 915, 1098, 1161-1162, 1200, 1208, 1250-1258, 4946, 4948, 4951-4953, 4960, 4970, 5346, 5348, 9044, 9049, 9056, 9061, 9066, 13488, 17248, 17584
273	37, 256, 277-278, 280, 284-285, 290, 297, 367, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855-858, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 923-924, 1009, 1025-1027, 1040-1043, 1047, 1051, 1088, 1100, 1112, 1122, 1140-1149, 1200, 1208, 1250, 1252, 1275, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951-4953, 4960, 4970-4971, 5012, 5123, 5346, 5348, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
274	500, 1047
275	37, 437, 500, 819, 850, 1047, 1200, 1208, 1252, 4946, 5348, 8229, 13488, 17584, 28709

表 710. IBM MQ for z/OS CCSID 変換サポート (続き)	
CCSID	変換元および変換先の CCSID
277	37, 256, 273, 278, 280, 284-285, 290, 297, 367, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 923-924, 1009, 1025-1027, 1040-1043, 1047, 1051, 1088, 1100, 1112, 1122, 1140-1149, 1200, 1208, 1252, 1275, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5348, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
278	37, 256, 273, 277, 280, 284-285, 290, 297, 367, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 923-924, 1009, 1025-1027, 1040-1043, 1047, 1051, 1088, 1100, 1112, 1122, 1140-1149, 1200, 1208, 1252, 1275, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5348, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
280	37, 256, 273, 277-278, 284-285, 290, 297, 367, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 923-924, 1009, 1025-1027, 1040-1043, 1047, 1051, 1088, 1100, 1112, 1122, 1140-1149, 1200, 1208, 1252, 1275, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5348, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
281	1047
282	500, 1047, 1200, 1208, 13488, 17584
284	37, 256, 273, 277-278, 280, 285, 290, 297, 367, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 923-924, 1009, 1025-1027, 1040-1043, 1047, 1051, 1088, 1100, 1112, 1122, 1140-1149, 1200, 1208, 1252, 1275, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5348, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
285	37, 256, 273, 277-278, 280, 284, 290, 297, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 923-924, 1025-1027, 1040-1043, 1047, 1051, 1088, 1100, 1112, 1122, 1140-1149, 1200, 1208, 1252, 1275, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5348, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
290	37, 256, 273, 277-278, 280, 284-285, 297, 367, 437, 500, 737, 775, 819, 833, 836, 850, 852, 855, 857, 860-865, 870-871, 895-897, 1009, 1025-1027, 1040-1043, 1047, 1088, 1112, 1122, 1139, 1200, 1208, 1252, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 4960, 4992, 5123, 8229, 8482, 9025, 9044, 9049, 9056, 13121, 13488, 17248, 17584, 25473, 25617, 25619, 25664, 28709
293	1200, 1208, 13488, 17584

表 710. IBM MQ for z/OS CCSID 変換サポート (続き)	
CCSID	変換元および変換先の CCSID
297	37, 256, 273, 277-278, 280, 284-285, 290, 367, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 923-924, 1009, 1025-1027, 1040-1043, 1047, 1051, 1088, 1100, 1112, 1122, 1140-1149, 1200, 1208, 1252, 1275, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5348, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
300	301, 941, 1200, 1208, 1351, 4396, 8492, 13488, 16684, 17584
301	300, 941, 1200, 1208, 1351, 4396, 8492, 13488, 16684, 17584
367	37, 256, 273, 277-278, 280, 284, 290, 297, 500, 819, 833, 836, 850, 871, 875, 1009, 1026-1027, 1041, 1088, 1115, 1126, 1200, 1208, 4386, 4929, 4932, 4946, 4971, 5123, 5211, 8229, 8482, 9025, 13121, 13488, 17584, 25617, 25664, 28709
420	37, 256, 424, 437, 500, 720, 737, 775, 819, 850, 852, 857, 860-865, 1008, 1046, 1089, 1098, 1112, 1122, 1127, 1200, 1208, 1252, 1256, 4946, 4948, 4953, 4960, 5104, 5142, 5352, 8229, 8612, 9044, 9049, 9056, 9238, 13488, 16804, 17248, 17584, 28709
423	37, 256, 273, 277-278, 280, 284-285, 297, 437, 500, 737, 775, 813, 819, 838, 850-852, 857, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1009, 1025-1027, 1041-1043, 1112, 1122, 1200, 1208, 1252-1253, 1280, 4909, 4934, 4946, 4948, 4953, 4960, 4970-4971, 5012, 5123, 8229, 9030, 9044, 9049, 9056, 9061, 9066, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 28709
424	37, 256, 420, 437, 500, 737, 775, 803, 819, 836, 850, 852, 856-857, 860-865, 916, 1112, 1122, 1200, 1208, 1252, 1255, 4932, 4946, 4948, 4952-4953, 4960, 5012, 5351, 8229, 8612, 9044, 9049, 9056, 13488, 16804, 17248, 17584, 28709
437	37, 256, 259, 273, 275, 277-278, 280, 284-285, 290, 297, 420, 423-424, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-863, 865-866, 869-871, 874-875, 880, 897, 903, 905, 912, 914-916, 920-924, 1025-1027, 1040-1043, 1047, 1051, 1097, 1098, 1114-1115, 1126, 1140-1149, 1200, 1208, 1252, 1257, 1275, 1280-1281, 1283, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4970-4971, 5012, 5123, 5210-5211, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 13488, 16804, 17584, 25473, 25479, 25617, 25619, 28709
500	37, 256, 273-275, 277-278, 280, 282, 284-285, 290, 297, 367, 420, 423-424, 437, 737, 775, 813, 819, 833, 836, 838, 850-852, 855-858, 860-866, 869-871, 874-875, 880, 891, 895, 897, 903-905, 912, 914-916, 920-924, 1004, 1009-1021, 1023, 1025-1027, 1040-1043, 1046-1047, 1051, 1088-1089, 1097, 1100-1107, 1112, 1114-1115, 1122, 1124-1126, 1129-1133, 1137, 1140-1149, 1200, 1208, 1250-1258, 1275, 1280-1283, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951-4953, 4960, 4970-4971, 5012, 5123, 5142, 5210-5211, 5346, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 9238, 13121, 13488, 16804, 17248, 17584, 25473, 25479, 25480, 25617, 25619, 25664, 28709
720	37, 420, 864, 1200, 1208, 1256, 4960, 8229, 8612, 9056, 13488, 16804, 17248, 17584, 28709

表 710. IBM MQ for z/OS CCSID 変換サポート (続き)	
CCSID	変換元および変換先の CCSID
737	37, 256, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 833, 836, 838, 850, 869-871, 875, 880, 905, 1025-1027, 1097, 1200, 1208, 1252-1253, 1280, 4386, 4909, 4929, 4932, 4934, 4946, 4971, 5123, 8229, 8482, 8612, 9025, 9030, 9061, 13121, 13488, 16804, 17584, 28709
775	37, 256, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 833, 836, 838, 850, 870-871, 875, 880, 905, 1025-1027, 1097, 1112, 1122, 1200, 1208, 1252, 1257, 4386, 4929, 4932, 4934, 4946, 4971, 5123, 8229, 8482, 8612, 9025, 9030, 13121, 13488, 16804, 17584, 28709
803	424, 819, 850, 856, 862, 916, 1200, 1208, 1252, 1255, 4946, 4952, 5012, 13488, 17584
806	1200, 1208, 13488, 17584
808	259, 858-859, 872, 923-924, 1140, 1148, 1153-1154, 1200, 1208, 5347, 5348, 13488, 17584
813	37, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 737, 819, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1200, 1208, 1252-1253, 1280, 4909, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 5349, 8229, 9030, 9044, 9049, 9061, 9066, 13488, 17584, 25473, 25479, 25617, 25619, 28709
819	37, 256, 273, 275, 277-278, 280, 284-285, 290, 297, 367, 420, 423-424, 437, 500, 803, 813, 833, 836, 838, 850, 852, 855, 857-858, 860-861, 863-866, 869-871, 874-875, 880, 897, 903, 905, 912, 914-916, 920-924, 1004, 1025-1027, 1041-1043, 1047, 1051, 1088-1089, 1097, 1098, 1112, 1114, 1122-1123, 1126, 1130, 1132, 1137, 1140-1149, 1200, 1208, 1250-1255, 1257-1258, 1275, 1280-1281, 1283, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5210, 5346, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 16804, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
833	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 437, 500, 737, 775, 819, 836, 850, 852, 855, 857, 860-865, 870-871, 891, 1009, 1025-1027, 1040-1043, 1088, 1112, 1122, 1126, 1200, 1208, 1252, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 4960, 5123, 8229, 8482, 9025, 9044, 9049, 9056, 13121, 13488, 17248, 17584, 25617, 25619, 25664, 28709
834	926, 951, 1200, 1208, 1362, 4930, 9026, 13488, 17584
835	927, 947, 1200, 1208, 4931, 9027, 13488, 17584, 21427
836	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 424, 437, 500, 737, 775, 819, 833, 850, 852, 855, 857, 870-871, 875, 903, 1009, 1025-1027, 1040-1043, 1088, 1112, 1114-1115, 1122, 1200, 1208, 1252, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 4971, 5123, 5210-5211, 8229, 8482, 9025, 9044, 9049, 13121, 13488, 17584, 25479, 25617, 25619, 25664, 28709
837	928, 1200, 1208, 1380, 1385, 4933, 13488, 17584
838	37, 256, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 737, 775, 813, 819, 850, 852, 857, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1112, 1122, 1200, 1208, 1252, 4909, 4934, 4946, 4948, 4953, 4960, 4970-4971, 5012, 5123, 8229, 9030, 9044, 9049, 9056, 9061, 9066, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 28709

表 710. IBM MQ for z/OS CCSID 変換サポート (続き)	
CCSID	変換元および変換先の CCSID
848	924, 1148, 1158, 1200, 1208, 5347, 13488, 17584
849	924, 1148, 1154, 1200, 1208, 5347, 13488, 17584
850	37, 256, 259, 273, 275, 277-278, 280, 284-285, 290, 297, 367, 420, 423-424, 437, 500, 737, 775, 803, 813, 819, 833, 836, 838, 852, 855-858, 860-866, 869-871, 874-875, 880, 897, 903, 905, 912, 914-916, 920-924, 1004, 1025-1027, 1040-1043, 1047, 1051, 1088-1089, 1097, 1098, 1100, 1112, 1114, 1122, 1126, 1130, 1132, 1140-1149, 1200, 1208, 1250-1257, 1275, 1280-1281, 1283, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951-4953, 4960, 4970-4971, 5012, 5123, 5210, 5346, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 16804, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
851	259, 423, 500, 875, 1200, 1208, 4971, 13488, 17584
852	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 819, 833, 836, 838, 850, 855, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 1025-1027, 1040-1043, 1047, 1088, 1097, 1200, 1208, 1250, 1252, 1282, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4970-4971, 5012, 5123, 5346, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 13488, 16804, 17584, 25473, 25479, 25617, 25619, 25664, 28709
855	37, 259, 273, 277-278, 280, 284-285, 290, 297, 437, 500, 819, 833, 836, 850, 852, 857, 866, 870-871, 878, 880, 912, 915, 1025-1027, 1040-1043, 1088, 1200, 1208, 1250-1252, 1283, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 5123, 5346, 5347, 8229, 8482, 9025, 9044, 9049, 13121, 13488, 17584, 25617, 25619, 25664, 28709
856	259, 273, 424, 500, 803, 850, 862, 916, 1200, 1208, 1255, 4946, 4952, 5012, 5351, 13488, 17584
857	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 860-861, 863, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 1025-1027, 1040-1043, 1088, 1097, 1200, 1208, 1252, 1254, 1281, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4970-4971, 5012, 5123, 5350, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 13488, 16804, 17584, 25473, 25479, 25617, 25619, 25664, 28709
858	37, 259, 273, 277-278, 280, 284-285, 297, 437, 500, 808, 819, 850, 860-861, 865, 871-872, 901-902, 923-924, 1047, 1051, 1140-1149, 1153-1157, 1160-1162, 1164, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
859	808, 872, 901-902, 1153-1157, 1160-1162, 1164, 1200, 1208, 13488, 17584
860	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 819, 833, 838, 850, 852, 857-858, 861, 863, 865, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 923-924, 1025-1027, 1041-1043, 1097, 1140, 1145-1146, 1148, 1200, 1208, 1252, 4386, 4909, 4929, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 13488, 16804, 17584, 25473, 25479, 25617, 25619, 28709

表 710. IBM MQ for z/OS CCSID 変換サポート (続き)

CCSID	変換元および変換先の CCSID
861	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 819, 833, 838, 850, 852, 857-858, 860, 863, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 923-924, 1025-1027, 1041-1043, 1097, 1148, 1149, 1200, 1208, 1252, 4386, 4909, 4929, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 13488, 16804, 17584, 25473, 25479, 25617, 25619, 28709
862	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 803, 833, 838, 850, 856, 870-871, 875, 880, 905, 916, 1025-1027, 1097, 1200, 1208, 1252, 1255, 4386, 4929, 4934, 4946, 4952, 4971, 5012, 5123, 5351, 8229, 8482, 8612, 9025, 9030, 12712, 13121, 13488, 16804, 17584, 28709
863	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 819, 833, 838, 850, 852, 857, 860-861, 865, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 1025-1027, 1041-1043, 1051, 1097, 1140-1149, 1200, 1208, 1252, 1275, 4386, 4909, 4929, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 13488, 16804, 17584, 25473, 25479, 25617, 25619, 28709
864	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 500, 720, 819, 833, 838, 850, 870-871, 875, 880, 905, 918, 1008, 1025-1027, 1046, 1089, 1097, 1127, 1200, 1208, 1252, 1256, 4386, 4929, 4934, 4946, 4960, 4971, 5104, 5123, 5142, 5352, 8229, 8482, 8612, 9025, 9030, 9056, 9238, 13121, 13488, 16804, 17248, 17584, 28709
865	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 819, 833, 838, 850, 858, 860, 863, 870-871, 875, 880, 905, 923-924, 1025-1027, 1097, 1142-1143, 1148, 1200, 1208, 1252, 4386, 4929, 4934, 4946, 4971, 5123, 5348, 8229, 8482, 8612, 9025, 9030, 13121, 13488, 16804, 17584, 28709
866	37, 256, 437, 500, 819, 850, 855, 870, 878, 880, 915, 1025, 1200, 1208, 1251-1252, 1283, 4946, 4951, 5347, 8229, 13488, 17584, 28709
867	259, 1153-1155, 1160, 1200, 1208, 4899, 5351, 9048, 12712, 13488, 17584
868	918, 1006, 1200, 1208, 13488, 17584
869	37, 256, 259, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 737, 813, 819, 838, 850, 852, 857, 860-861, 863, 870-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1200, 1208, 1252-1254, 1280, 4909, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 5349, 8229, 9030, 9044, 9049, 9061, 9066, 13488, 17584, 25473, 25479, 25617, 25619, 28709
870	37, 256, 273, 277-278, 280, 284-285, 290, 297, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-866, 869, 871, 874-875, 880, 897, 903, 912, 915-916, 920, 1009, 1025-1027, 1040-1043, 1047, 1088, 1112, 1122, 1200, 1208, 1250, 1252, 1282, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5346, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709

表 710. IBM MQ for z/OS CCSID 変換サポート (続き)	
CCSID	変換元および変換先の CCSID
871	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-865, 869, 870, 874-875, 880, 897, 903, 912, 916, 920, 923-924, 1009, 1025-1027, 1040-1043, 1047, 1051, 1088, 1112, 1122, 1140-1149, 1200, 1208, 1252, 1275, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5348, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
872	259, 808, 858-859, 923-924, 1140-1149, 1153-1155, 1200, 1208, 5347, 5348, 13488, 17584
874	37, 259, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 813, 819, 838, 850, 852, 857, 860-861, 863, 869-871, 875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1200, 1208, 1252, 4909, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 8229, 9030, 9044, 9049, 9061, 9066, 13488, 17584, 25473, 25479, 25617, 25619, 28709
875	37, 256, 273, 277-278, 280, 284-285, 297, 367, 423, 437, 500, 737, 775, 813, 819, 836, 838, 850-852, 857, 860-865, 869-871, 874, 880, 897, 903, 912, 916, 920, 1009, 1025-1027, 1041-1043, 1047, 1088, 1112, 1122, 1200, 1208, 1252-1253, 1280, 4909, 4932, 4934, 4946, 4948, 4953, 4960, 4970-4971, 5012, 5123, 5349, 8229, 9030, 9044, 9049, 9056, 9061, 9066, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
878	855, 866, 880, 915, 1025, 1131, 1200, 1208, 1251, 1283, 4951, 5347, 13488, 17584
880	37, 256, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 737, 775, 813, 819, 838, 850, 852, 855, 857, 860-866, 869-871, 874-875, 878, 897, 903, 912, 915-916, 920, 1009, 1025-1027, 1041-1043, 1112, 1122, 1200, 1208, 1251-1252, 1283, 4909, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5347, 8229, 9030, 9044, 9049, 9056, 9061, 9066, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 28709
891	500, 833, 1088, 1200, 1208, 4929, 9025, 13121, 13488, 17584, 25664
895	290, 500, 1027, 1041, 1200, 1208, 4386, 5123, 8482, 13488, 17584, 25617
896	290, 1027, 1041, 1200, 1208, 4386, 4992, 5123, 8482, 13488, 17584, 25617
897	37, 273, 277-278, 280, 284-285, 290, 297, 423, 437, 500, 813, 819, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 903, 912, 916, 920, 1025-1027, 1041-1043, 1200, 1208, 1252, 4386, 4909, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 8229, 8482, 9030, 9044, 9049, 9061, 9066, 13488, 17584, 25473, 25479, 25617, 25619, 28709
899	259
901	259, 858-859, 902, 923-924, 1140, 1148, 1156-1157, 1200, 1208, 5348, 5353, 13488, 17584
902	259, 858-859, 901, 923-924, 1140, 1148, 1156-1157, 1200, 1208, 5348, 5353, 13488, 17584

表 710. IBM MQ for z/OS CCSID 変換サポート (続き)	
CCSID	変換元および変換先の CCSID
903	37, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 813, 819, 836, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 897, 912, 916, 920, 1025-1027, 1041-1043, 1115, 1200, 1208, 1252, 4909, 4932, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 5211, 8229, 9030, 9044, 9049, 9061, 9066, 13488, 17584, 25473, 25479, 25617, 25619, 28709
904	37, 500, 1114, 1200, 1208, 5210, 8229, 13488, 17584, 25480, 28709
905	37, 256, 437, 500, 737, 775, 819, 850, 852, 857, 860-865, 920, 1026, 1112, 1122, 1200, 1208, 1252, 1254, 1281, 4946, 4948, 4953, 4960, 8229, 9044, 9049, 9056, 13488, 17248, 17584, 28709
912	37, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 813, 819, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 916, 920, 1025-1027, 1041-1043, 1047, 1200, 1208, 1250, 1252, 1282, 4909, 4934, 4946, 4948, 4951, 4953, 4970-4971, 5012, 5123, 5346, 8229, 9030, 9044, 9049, 9061, 9066, 13488, 17584, 25473, 25479, 25617, 25619, 28709
914	37, 437, 500, 819, 850, 1200, 1208, 1252, 1257, 4946, 8229, 13488, 17584, 28709
915	37, 259, 437, 500, 819, 850, 855, 866, 870, 878, 880, 1025, 1131, 1200, 1208, 1251-1252, 1283, 4946, 4951, 5347, 8229, 13488, 17584, 28709
916	37, 273, 277-278, 280, 284-285, 297, 423-424, 437, 500, 803, 813, 819, 838, 850, 852, 856-857, 860-863, 869-871, 874-875, 880, 897, 903, 912, 920, 1025-1027, 1041-1043, 1200, 1208, 1252, 1255, 4909, 4934, 4946, 4948, 4952-4953, 4970-4971, 5012, 5123, 5351, 8229, 9030, 9044, 9049, 9061, 9066, 13488, 17584, 25473, 25479, 25617, 25619, 28709
918	864, 868, 1006, 1200, 1208, 4960, 9056, 13488, 17248, 17584
920	37, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 813, 819, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 1025-1026, 1200, 1208, 1252, 1254, 1281, 4909, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5350, 8229, 9030, 9044, 9049, 9061, 9066, 13488, 17584, 25473, 25479, 28709
921	37, 437, 500, 819, 850, 922, 1112, 1122, 1200, 1208, 1252, 1257, 4946, 5353, 8229, 13488, 17584, 28709
922	37, 437, 500, 819, 850, 921, 1112, 1122, 1200, 1208, 1252, 1257, 4946, 5353, 8229, 13488, 17584, 28709
923	37, 273, 277-278, 280, 284-285, 297, 437, 500, 808, 819, 850, 858, 860-861, 865, 871-872, 901-902, 924, 1047, 1051, 1140-1149, 1153-1158, 1160-1162, 1164, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
924	37, 273, 277-278, 280, 284-285, 297, 437, 500, 808, 819, 848-850, 858, 860-861, 865, 871-872, 901-902, 923, 1047, 1051, 1140-1149, 1153-1157, 1160-1164, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
926	834, 951, 9026
927	835, 947, 1200, 1208, 4931, 9027, 13488, 17584, 21427
928	837, 1200, 1208, 1380, 13488, 17584

表 710. IBM MQ for z/OS CCSID 変換サポート (続き)	
CCSID	変換元および変換先の CCSID
930	931-932, 939, 942-943, 1200, 1208, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 13488, 17314, 17584, 25508, 25518, 29614, 33698-33700, 37796
931	930, 932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
932	930-931, 939, 942-943, 1200, 1208, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 13488, 17314, 17584, 25508, 25518, 29614, 33698-33700, 37796
933	934, 944, 949, 1200, 1208, 1363-1364, 5029, 5045, 5460, 9125, 9555, 13221, 13488, 13651, 17317, 17584, 25510, 25520, 25525, 29616, 29621, 33717, 37813
934	933, 949, 5029, 5045, 5460, 9125, 13221, 17317, 25510, 25525, 29621, 33717, 37813
935	936, 946, 1200, 1208, 1381, 1386, 1388, 5031, 5477, 5482, 5484, 9127, 13223, 13488, 17584, 25512
936	935, 946, 1381, 5031, 5477, 5484, 9127, 13223, 25512
937	938, 948, 950, 1200, 1208, 1370, 5033, 5046, 9142, 13488, 17584, 25514, 25524, 29620
938	937, 950, 1370, 5033, 5046, 9142, 25514
939	930-932, 942-943, 1200, 1208, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 13488, 17314, 17584, 25508, 25518, 29614, 33698-33700, 37796
941	300-301, 1200, 1208, 1351, 4396, 8492, 13488, 16684, 17584
942	930-932, 939, 943, 1200, 1208, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 13488, 17314, 17584, 25508, 25518, 29614, 33698-33700, 37796
943	930-932, 939, 942, 1200, 1208, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 13488, 17314, 17584, 25508, 25518, 29614, 33698-33700, 37796
944	933, 949, 1200, 1208, 5029, 5045, 5460, 9125, 13221, 13488, 17317, 17584, 25520, 25525, 29616, 29621, 33717, 37813
946	935-936, 1200, 1208, 5031, 5484, 9127, 13223, 13488, 17584, 25512
947	835, 927, 1200, 1208, 4931, 9027, 13488, 17584, 21427
948	937, 950, 1200, 1208, 1370, 5033, 5046, 9142, 13488, 17584, 25524, 29620
949	933-934, 944, 1200, 1208, 1363-1364, 5029, 5045, 5460, 9125, 9555, 13221, 13488, 13651, 17317, 17584, 25510, 25520, 25525, 29616, 29621, 33717, 37813
950	937-938, 948, 1200, 1208, 1370, 5033, 5046, 9142, 13488, 17584, 25514, 25524, 29620
951	834, 926, 1200, 1208, 1362, 4930, 9026, 13488, 17584

表 710. IBM MQ for z/OS CCSID 変換サポート (続き)	
CCSID	変換元および変換先の CCSID
1004	500, 819, 850, 1200, 1208, 4946, 13488, 17584
1006	868, 918, 1200, 1208, 13488, 17584
1008	420, 864, 1200, 1208, 4960, 5104, 8612, 9056, 13488, 16804, 17248, 17584
1009	37, 273, 277-278, 280, 284, 290, 297, 367, 423, 500, 833, 836, 870-871, 875, 880, 1025-1026, 1200, 1208, 4386, 4929, 4932, 4971, 8229, 8482, 9025, 13121, 13488, 17584, 28709
1010	500, 1200, 1208, 13488, 17584
1011	500, 1200, 1208, 13488, 17584
1012	500, 1200, 1208, 13488, 17584
1013	500, 1140, 1200, 1208, 13488, 17584
1014	500, 1200, 1208, 13488, 17584
1015	500, 1200, 1208, 13488, 17584
1016	500, 1200, 1208, 13488, 17584
1017	500, 1200, 1208, 13488, 17584
1018	500, 1200, 1208, 13488, 17584
1019	500, 1200, 1208, 13488, 17584
1020	500
1021	500
1023	500
1025	37, 256, 273, 277-278, 280, 284-285, 290, 297, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-866, 869-871, 874-875, 878, 880, 897, 903, 912, 915-916, 920, 1009, 1026-1027, 1040-1043, 1051, 1088, 1112, 1122, 1131, 1200, 1208, 1251-1252, 1283, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5347, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
1026	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-865, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 1009, 1025, 1027, 1040-1043, 1047, 1088, 1112, 1122, 1200, 1208, 1252, 1254, 1281, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5350, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
1027	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-865, 869-871, 874-875, 880, 895-897, 903, 912, 916, 1025-1026, 1040-1043, 1047, 1088, 1112, 1122, 1139, 1200, 1208, 1252, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 4992, 5012, 5123, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709

表 710. IBM MQ for z/OS CCSID 変換サポート (続き)	
CCSID	変換元および変換先の CCSID
1040	37, 273, 277-278, 280, 284-285, 290, 297, 437, 500, 833, 836, 850, 852, 855, 857, 870-871, 1025-1027, 1041-1043, 1088, 1200, 1208, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 5123, 8229, 8482, 9025, 9044, 9049, 13121, 13488, 17584, 25617, 25619, 25664, 28709
1041	37, 273, 277-278, 280, 284-285, 290, 297, 367, 423, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 895-897, 903, 912, 916, 1025-1027, 1040, 1042-1043, 1088, 1200, 1208, 1252, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4970-4971, 4992, 5012, 5123, 8229, 8482, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 13488, 17584, 25473, 25479, 25617, 25619, 25664, 28709
1042	37, 273, 277-278, 280, 284-285, 290, 297, 423, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 1025-1027, 1040, 1041, 1043, 1088, 1200, 1208, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4970-4971, 5012, 5123, 8229, 8482, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 13488, 17584, 25473, 25479, 25617, 25619, 25664, 28709
1043	37, 273, 277-278, 280, 284-285, 290, 297, 423, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 1025-1027, 1040, 1041, 1042, 1088, 1114, 1200, 1208, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4970-4971, 5012, 5123, 5210, 8229, 8482, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 13488, 17584, 25473, 25479, 25617, 25619, 25664, 28709
1046	420, 500, 864, 1089, 1127, 1200, 1208, 1256, 4960, 5142, 5352, 8612, 9056, 9238, 13488, 16804, 17248, 17584
1047	37, 273-275, 277-278, 280, 281, 282, 284-285, 290, 297, 437, 500, 819, 850, 852, 858, 870-871, 875, 912, 923-924, 1026-1027, 1140-1149, 1200, 1208, 1252, 1254, 4946, 4948, 5123, 8229, 8482, 13488, 17584, 28709
1051	37, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 863, 871, 923-924, 1025, 1097, 1140-1149, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
1088	37, 273, 277-278, 280, 284-285, 290, 297, 367, 500, 819, 833, 836, 850, 852, 855, 857, 870-871, 875, 891, 1025-1027, 1040-1043, 1126, 1200, 1208, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 4971, 5123, 8229, 8482, 9025, 9044, 9049, 13121, 13488, 17584, 25617, 25619, 25664, 28709
1089	420, 500, 819, 850, 864, 1046, 1127, 1200, 1208, 1256, 4946, 4960, 5142, 5352, 8612, 9056, 9238, 13488, 16804, 17248, 17584
1097	37, 437, 500, 737, 775, 819, 850, 852, 857, 860-865, 1051, 1098, 1112, 1122, 1200, 1208, 1252, 4946, 4948, 4953, 4960, 8229, 9044, 9049, 9056, 13488, 17248, 17584, 28709
1098	259, 420, 437, 819, 850, 1097, 1200, 1208, 1252, 4946, 8612, 13488, 16804, 17584
1100	37, 273, 277-278, 280, 284-285, 297, 500, 850, 4946, 8229, 28709
1101	500
1102	500

表 710. IBM MQ for z/OS CCSID 変換サポート (続き)	
CCSID	変換元および変換先の CCSID
1103	500
1104	500
1105	500
1106	500
1107	500
1112	37, 256, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 500, 775, 819, 833, 836, 838, 850, 870-871, 875, 880, 905, 921-922, 1025-1027, 1097, 1122, 1200, 1208, 1252, 1257, 4386, 4929, 4932, 4934, 4946, 4971, 5123, 5353, 8229, 8482, 8612, 9025, 9030, 13121, 13488, 16804, 17584, 28709
1114	37, 437, 500, 819, 836, 850, 904, 1043, 1115, 1200, 1208, 4932, 4946, 5210-5211, 8229, 13488, 17584, 25480, 25619, 28709
1115	37, 367, 437, 500, 836, 903, 1114, 1200, 1208, 4932, 5210-5211, 8229, 13488, 17584, 25479, 28709
1122	37, 256, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 500, 775, 819, 833, 836, 838, 850, 870-871, 875, 880, 905, 921-922, 1025-1027, 1097, 1112, 1200, 1208, 1252, 1257, 4386, 4929, 4932, 4934, 4946, 4971, 5123, 5353, 8229, 8482, 8612, 9025, 9030, 13121, 13488, 16804, 17584, 28709
1123	819, 1124-1125, 1148, 1200, 1208, 1251-1252, 1283, 5347, 13488, 17584
1124	37, 500, 1123, 1125, 1200, 1208, 1251, 1283, 5347, 8229, 13488, 17584, 28709
1125	500, 1123, 1124, 1200, 1208, 1251, 1283, 5347, 13488, 17584
1126	37, 367, 437, 500, 819, 833, 850, 1088, 1200, 1208, 1252, 4929, 4946, 8229, 9025, 13121, 13488, 17584, 25664, 28709
1127	420, 864, 1046, 1089, 1256, 4960, 5142, 8612, 9056, 9238, 16804, 17248
1129	500, 1130, 1200, 1208, 1258, 5354, 13488, 17584
1130	37, 500, 819, 850, 1129, 1200, 1208, 1252, 1258, 4946, 5354, 8229, 13488, 17584, 28709
1131	37, 500, 878, 915, 1025, 1200, 1208, 1251, 1283, 5347, 8229, 13488, 17584, 28709
1132	37, 500, 819, 850, 1133, 1200, 1208, 1252, 4946, 8229, 13488, 17584, 28709
1133	500, 1132, 1200, 1208, 13488, 17584
1137	37, 500, 819, 1200, 1208, 8229, 13488, 17584, 28709
1139	290, 1027, 4386, 5123, 8482
1140	37, 273, 277-278, 280, 284-285, 297, 437, 500, 808, 819, 850, 858, 860, 863, 871-872, 901-902, 923-924, 1013, 1047, 1051, 1141-1149, 1153-1157, 1160-1162, 1164, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709

表 710. IBM MQ for z/OS CCSID 変換サポート (続き)	
CCSID	変換元および変換先の CCSID
1141	37, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 863, 871-872, 923-924, 1047, 1051, 1140, 1142-1149, 1153-1157, 1160-1162, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
1142	37, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 863, 865, 871-872, 923-924, 1047, 1051, 1140-1141, 1143-1149, 1153-1157, 1160-1162, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
1143	37, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 863, 865, 871-872, 923-924, 1047, 1051, 1140-1142, 1144-1149, 1153-1157, 1160-1162, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
1144	37, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 863, 871-872, 923-924, 1047, 1051, 1140-1143, 1145-1149, 1153-1157, 1160-1162, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
1145	37, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 860, 863, 871-872, 923-924, 1047, 1051, 1140-1144, 1146-1149, 1153-1157, 1160-1162, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
1146	37, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 860, 863, 871-872, 923-924, 1047, 1051, 1140-1145, 1147-1149, 1153-1157, 1160-1162, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
1147	37, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 863, 871-872, 923-924, 1047, 1051, 1140-1146, 1148-1149, 1153-1157, 1160-1162, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
1148	37, 273, 277-278, 280, 284-285, 297, 437, 500, 808, 819, 848-850, 858, 860-861, 863, 865, 871-872, 901-902, 923-924, 1047, 1051, 1123, 1140-1147, 1149, 1153-1164, 1200, 1208, 1252, 1275, 4899, 4946, 5348, 5349, 8229, 12712, 13488, 17584, 28709
1149	37, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 861, 863, 871-872, 923-924, 1047, 1051, 1140-1148, 1153-1157, 1160-1162, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
1153	808, 858-859, 867, 872, 923-924, 1140-1149, 1154-1157, 1160-1162, 1200, 1208, 5348, 9044, 13488, 17584
1154	808, 849, 858-859, 867, 872, 923-924, 1140-1149, 1153, 1155-1157, 1160-1162, 1200, 1208, 5347, 5348, 13488, 17584
1155	858-859, 867, 872, 923-924, 1140-1149, 1153-1154, 1156-1157, 1160-1162, 1200, 1208, 5348, 5350, 9049, 13488, 17584
1156	858-859, 901-902, 923-924, 1140-1149, 1153-1155, 1157, 1160, 1200, 1208, 5348, 5353, 12712, 13488, 17584
1157	858-859, 901-902, 923-924, 1140-1149, 1153-1156, 1160, 1200, 1208, 5348, 5353, 12712, 13488, 17584
1158	848, 923, 1148, 1200, 1208, 5347, 5348, 13488, 17584

表 710. IBM MQ for z/OS CCSID 変換サポート (続き)	
CCSID	変換元および変換先の CCSID
1159	1148, 1200, 1208, 13488, 17584
1160	858-859, 867, 923-924, 1140-1149, 1153-1157, 1161-1162, 1200, 1208, 5348, 13488, 17584
1161	259, 858-859, 923-924, 1140-1149, 1153-1155, 1160, 5348, 17584
1162	259, 858-859, 923-924, 1140-1149, 1153-1155, 1160, 5348, 17584
1163	924, 1148, 1164, 5354, 17584
1164	858-859, 923-924, 1140, 1148, 1163, 1200, 1208, 5348, 5354, 13488, 17584
1166	1200, 1208, 13488, 17584
1200	37, 256, 259, 273, 275, 277-278, 280, 282, 284-285, 290, 293, 297, 300-301, 367, 420, 423-424, 437, 500, 720, 737, 775, 803, 806, 808, 813, 819, 833-838, 848-852, 855-872, 874-875, 878, 880, 891, 895-897, 901-905, 912, 914-916, 918, 920-924, 927-928, 930, 932-933, 935, 937, 939, 941-944, 946-951, 1004, 1006, 1008-1019, 1025-1027, 1040-1043, 1046-1047, 1051, 1088-1089, 1097-1098, 1112, 1114-1115, 1122-1126, 1129-1133, 1137, 1140-1149, 1153-1160, 1164, 1166, 1208, 1250-1258, 1275-1277, 1280-1285, 1351, 1362-1364, 1370-1371, 1374-1379, 1380-1381, 1385-1386, 1388, 1390, 1399, 4899, 4909, 4930, 4933, 4948, 4951-4952, 4960, 4971, 5012, 5039, 5104, 5123, 5142, 5210, 5346-5354, 8482, 8612, 9027, 9030, 9044, 9048-9049, 9056, 9061, 9066, 9238, 12712, 13121, 13218, 13488, 16684, 16804, 17248, 17584, 21427, 28709
1208	37, 256, 259, 273, 275, 277-278, 280, 282, 284-285, 290, 293, 297, 300-301, 367, 420, 423-424, 437, 500, 720, 737, 775, 803, 806, 808, 813, 819, 833-838, 848-852, 855-872, 874-875, 878, 880, 891, 895-897, 901-905, 912, 914-916, 918, 920-924, 927-928, 930, 932-933, 935, 937, 939, 941-944, 946-951, 1004, 1006, 1008-1019, 1025-1027, 1040-1043, 1046-1047, 1051, 1088-1089, 1097-1098, 1112, 1114-1115, 1122-1126, 1129-1133, 1137, 1140-1149, 1153-1160, 1164, 1166, 1200, 1250-1258, 1275-1277, 1280-1285, 1351, 1362-1364, 1370-1371, 1374-1379, 1380-1381, 1385-1386, 1388, 1390, 1399, 4899, 4909, 4930, 4933, 4948, 4951-4952, 4960, 4971, 5012, 5026, 5035, 5039, 5104, 5123, 5142, 5210, 5346-5354, 8482, 8612, 9027, 9030, 9044, 9048-9049, 9056, 9061, 9066, 9238, 12712, 13121, 13218, 13488, 16684, 16804, 17248, 17584, 21427, 28709
1250	37, 259, 273, 500, 819, 850, 852, 855, 870, 912, 1200, 1208, 1252, 1282, 4946, 4948, 4951, 5346, 8229, 9044, 13488, 17584, 28709
1251	37, 256, 259, 500, 819, 850, 855, 866, 878, 880, 915, 1025, 1123-1125, 1131, 1200, 1208, 1252, 1283, 4946, 4951, 5347, 8229, 13488, 17584, 28709

表 710. IBM MQ for z/OS CCSID 変換サポート (続き)	
CCSID	変換元および変換先の CCSID
1252	37, 256, 259, 273, 275, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 737, 775, 803, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-866, 869-871, 874-875, 880, 897, 903, 905, 912, 914-916, 920-924, 1025-1027, 1041, 1047, 1051, 1097-1098, 1112, 1122-1123, 1126, 1130, 1132, 1140-1149, 1200, 1208, 1250-1251, 1254-1255, 1257, 1275, 1280-1281, 1283, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5346, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 16804, 17248, 17584, 25473, 25479, 25617, 28709
1253	37, 259, 423, 500, 737, 813, 819, 850, 869, 875, 1200, 1208, 1280, 4909, 4946, 4971, 5349, 8229, 9061, 13488, 17584, 28709
1254	37, 259, 500, 819, 850, 857, 869, 905, 920, 1026, 1047, 1200, 1208, 1252, 1281, 4946, 4953, 5350, 8229, 9049, 9061, 13488, 17584, 28709
1255	37, 259, 424, 500, 803, 819, 850, 856, 862, 916, 1200, 1208, 1252, 1281, 4946, 4952, 5012, 5351, 8229, 13488, 17584, 28709
1256	259, 420, 500, 720, 850, 864, 1046, 1089, 1127, 1200, 1208, 4946, 4960, 5142, 5352, 8612, 9056, 9238, 13488, 16804, 17248, 17584
1257	37, 259, 437, 500, 775, 819, 850, 914, 921-922, 1112, 1122, 1200, 1208, 1252, 4946, 5353, 8229, 13488, 17584, 28709
1258	37, 259, 500, 819, 1129-1130, 1200, 1208, 5354, 8229, 13488, 17584, 28709
1275	37, 256, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 863, 871, 923-924, 1051, 1140-1149, 1200, 1208, 1252, 4946, 5348, 8229, 13488, 17584, 28709
1276	1200, 1208, 13488, 17584
1277	1200, 1208, 13488, 17584
1280	37, 423, 437, 500, 737, 813, 819, 850, 869, 875, 1200, 1208, 1252-1253, 4909, 4946, 4971, 5349, 8229, 9061, 13488, 17584, 28709
1281	37, 437, 500, 819, 850, 857, 905, 920, 1026, 1200, 1208, 1252, 1254-1255, 4946, 4953, 5350, 8229, 9049, 13488, 17584, 28709
1282	500, 852, 870, 912, 1200, 1208, 1250, 4948, 5346, 9044, 13488, 17584
1283	37, 437, 500, 819, 850, 855, 866, 878, 880, 915, 1025, 1123-1125, 1131, 1200, 1208, 1251-1252, 4946, 4951, 5347, 8229, 13488, 17584, 28709
1284	1200, 1208, 13488, 17584
1285	1200, 1208, 13488, 17584
1351	300-301, 941, 1200, 1208, 4396, 8492, 13488, 16684, 17584
1362	834, 951, 1200, 1208, 4930, 9026, 13488, 17584
1363	933, 949, 1200, 1208, 1364, 5029, 5045, 5460, 9125, 9555, 13221, 13488, 13651, 17317, 17584, 25525, 29621, 33717, 37813
1364	933, 949, 1200, 1208, 1363, 5029, 5045, 5460, 9125, 9555, 13221, 13488, 13651, 17317, 17584, 25525, 29621, 33717, 37813

表 710. IBM MQ for z/OS CCSID 変換サポート (続き)	
CCSID	変換元および変換先の CCSID
1370	937-938, 948, 950, 1200, 1208, 1371, 5033, 5046, 9142, 13488, 17584, 25514, 25524, 29620
1371	1200, 1208, 1370, 13488, 17584
1374	1200, 1208
1375	1200, 1208
1376	1200, 1208
1377	1200, 1208
1378	1200, 1208
1379	1200, 1208
1380	837, 928, 1200, 1208, 1385, 4933, 13488, 17584
1381	935-936, 1200, 1208, 1386, 1388, 5031, 5477, 5482, 5484, 9127, 13223, 13488, 17584, 25512
1385	837, 1200, 1208, 1380, 4933, 13488, 17584
1386	935, 1200, 1208, 1381, 1388, 5031, 5477, 5482, 5484, 9127, 13223, 13488, 17584
1388	935, 1200, 1208, 1381, 1386, 5031, 5477, 5482, 5484, 5488, 9127, 13223, 13488, 17584
1390	930-932, 939, 942-943, 1200, 1208, 1399, 5026, 5028, 5035, 5038-5039, 5055, 9122, 9124, 9131, 9135, 13218-13219, 13231, 13488, 17314, 17584, 25508, 25518, 29614, 33698-33700, 37796
1399	930-932, 939, 942-943, 1200, 1208, 1390, 5026, 5028, 5035, 5038-5039, 5050, 9122, 9124, 9131, 9135, 13218-13219, 13231, 13488, 17314, 17584, 25508, 25518, 29614, 33698-33700, 37796
4386	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 437, 500, 737, 775, 819, 833, 836, 850, 852, 855, 857, 860-865, 870-871, 895-897, 1009, 1025-1027, 1040-1043, 1088, 1112, 1122, 1139, 1252, 4929, 4932, 4946, 4948, 4951, 4953, 4960, 4992, 5123, 8229, 8482, 9025, 9044, 9049, 9056, 13121, 17248, 25473, 25617, 25619, 25664, 28709
4396	300-301, 941, 1351, 8492, 16684
4899	867, 1148, 1200, 1208, 5351, 9048, 12712, 13488, 17584
4909	37, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 737, 813, 819, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1200, 1208, 1252-1253, 1280, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 5349, 8229, 9030, 9044, 9049, 9061, 9066, 13488, 17584, 25473, 25479, 25617, 25619, 28709
4929	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 437, 500, 737, 775, 819, 833, 836, 850, 852, 855, 857, 860-865, 870-871, 891, 1009, 1025-1027, 1040-1043, 1088, 1112, 1122, 1126, 1252, 4386, 4932, 4946, 4948, 4951, 4953, 4960, 5123, 8229, 8482, 9025, 9044, 9049, 9056, 13121, 17248, 25617, 25619, 25664, 28709
4930	834, 951, 1200, 1208, 1362, 9026, 13488, 17584
4931	835, 927, 947, 9027, 21427

表 710. IBM MQ for z/OS CCSID 変換サポート (続き)	
CCSID	変換元および変換先の CCSID
4932	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 424, 437, 500, 737, 775, 819, 833, 836, 850, 852, 855, 857, 870-871, 875, 903, 1009, 1025-1027, 1040-1043, 1088, 1112, 1114-1115, 1122, 1252, 4386, 4929, 4946, 4948, 4951, 4953, 4971, 5123, 5210-5211, 8229, 8482, 9025, 9044, 9049, 13121, 25479, 25617, 25619, 25664, 28709
4933	837, 1200, 1208, 1380, 1385, 13488, 17584
4934	37, 256, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 737, 775, 813, 819, 838, 850, 852, 857, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1112, 1122, 1252, 4909, 4946, 4948, 4953, 4960, 4970-4971, 5012, 5123, 8229, 9030, 9044, 9049, 9056, 9061, 9066, 17248, 25473, 25479, 25617, 25619, 28709
4946	37, 256, 259, 273, 275, 277-278, 280, 284-285, 290, 297, 367, 420, 423-424, 437, 500, 737, 775, 803, 813, 819, 833, 836, 838, 850, 852, 855-858, 860-866, 869-871, 874-875, 880, 897, 903, 905, 912, 914-916, 920-924, 1004, 1025-1027, 1040-1043, 1047, 1051, 1088-1089, 1097-1098, 1100, 1112, 1114, 1122, 1126, 1130, 1132, 1140-1149, 1250-1257, 1275, 1280-1281, 1283, 4386, 4909, 4929, 4932, 4934, 4948, 4951-4953, 4960, 4970-4971, 5012, 5123, 5210, 5346, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 16804, 17248, 25473, 25479, 25617, 25619, 25664, 28709
4948	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 1025-1027, 1040-1043, 1047, 1088, 1097, 1200, 1208, 1250, 1252, 1282, 4386, 4909, 4929, 4932, 4934, 4946, 4951, 4953, 4970-4971, 5012, 5123, 5346, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 13488, 16804, 17584, 25473, 25479, 25617, 25619, 25664, 28709
4951	37, 259, 273, 277-278, 280, 284-285, 290, 297, 437, 500, 819, 833, 836, 850, 852, 855, 857, 866, 870-871, 878, 880, 912, 915, 1025-1027, 1040-1043, 1088, 1200, 1208, 1250-1252, 1283, 4386, 4929, 4932, 4946, 4948, 4953, 5123, 5346, 5347, 8229, 8482, 9025, 9044, 9049, 13121, 13488, 17584, 25617, 25619, 25664, 28709
4952	259, 273, 424, 500, 803, 850, 856, 862, 916, 1200, 1208, 1255, 4946, 5012, 5351, 13488, 17584
4953	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 1025-1027, 1040-1043, 1088, 1097, 1252, 1254, 1281, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4970-4971, 5012, 5123, 5350, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 16804, 25473, 25479, 25617, 25619, 25664, 28709
4960	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 500, 720, 819, 833, 838, 850, 864, 870-871, 875, 880, 905, 918, 1008, 1025-1027, 1046, 1089, 1097, 1127, 1200, 1208, 1252, 1256, 4386, 4929, 4934, 4946, 4971, 5104, 5123, 5142, 5352, 8229, 8482, 8612, 9025, 9030, 9056, 9238, 13121, 13488, 16804, 17248, 17584, 28709

表 710. IBM MQ for z/OS CCSID 変換サポート (続き)	
CCSID	変換元および変換先の CCSID
4970	37, 259, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 813, 819, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1252, 4909, 4934, 4946, 4948, 4953, 4971, 5012, 5123, 8229, 9030, 9044, 9049, 9061, 9066, 25473, 25479, 25617, 25619, 28709
4971	37, 256, 273, 277-278, 280, 284-285, 297, 367, 423, 437, 500, 737, 775, 813, 819, 836, 838, 850-852, 857, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1009, 1025-1027, 1041-1043, 1047, 1088, 1112, 1122, 1200, 1208, 1252-1253, 1280, 4909, 4932, 4934, 4946, 4948, 4953, 4960, 4970, 5012, 5123, 5349, 8229, 9030, 9044, 9049, 9056, 9061, 9066, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
4992	290, 896, 1027, 1041, 4386, 5123, 8482, 25617
5012	37, 273, 277-278, 280, 284-285, 297, 423-424, 437, 500, 803, 813, 819, 838, 850, 852, 856-857, 860-863, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1200, 1208, 1252, 1255, 4909, 4934, 4946, 4948, 4952-4953, 4970-4971, 5123, 5351, 8229, 9030, 9044, 9049, 9061, 9066, 13488, 17584, 25473, 25479, 25617, 25619, 28709
5026	930-932, 939, 942-943, 1390, 1399, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 1208, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
5028	930-932, 939, 942-943, 1390, 1399, 5026, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
5029	933-934, 944, 949, 1363-1364, 5045, 5460, 9125, 9555, 13221, 13651, 17317, 25510, 25520, 25525, 29616, 29621, 33717, 37813
5031	935-936, 946, 1381, 1386, 1388, 5477, 5482, 5484, 9127, 13223, 25512
5033	937-938, 948, 950, 1370, 5046, 9142, 25514, 25524, 29620
5035	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5038-5039, 9122, 9124, 9131, 9135, 1208, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
5038	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
5039	930-932, 939, 942-943, 1200, 1208, 1390, 1399, 5026, 5028, 5035, 5038, 9122, 9124, 9131, 9135, 13218-13219, 13231, 13488, 17314, 17584, 25508, 25518, 29614, 33698-33700, 37796
5045	933-934, 944, 949, 1363-1364, 5029, 5460, 9125, 9555, 13221, 13651, 17317, 25510, 25520, 25525, 29616, 29621, 33717, 37813
5046	937-938, 948, 950, 1370, 5033, 9142, 25514, 25524, 29620
5104	420, 864, 1008, 1200, 1208, 4960, 8612, 9056, 13488, 16804, 17248, 17584
5123	290, 367, 423, 437, 819, 1027, 1041, 1047, 1140-1149, 1156, 1157, 1160, 1200, 1208, 1252, 4948, 5348, 8482, 13488

表 710. IBM MQ for z/OS CCSID 変換サポート (続き)	
CCSID	変換元および変換先の CCSID
5142	420, 500, 864, 1046, 1089, 1127, 1200, 1208, 1256, 4960, 5352, 8612, 9056, 9238, 13488, 16804, 17248, 17584
5210	37, 437, 500, 819, 836, 850, 904, 1043, 1114-1115, 1200, 1208, 4932, 4946, 5211, 8229, 13488, 17584, 25480, 25619, 28709
5211	37, 367, 437, 500, 836, 903, 1114-1115, 4932, 5210, 8229, 25479, 28709
5346	37, 259, 273, 500, 819, 850, 852, 855, 870, 912, 1200, 1208, 1250, 1252, 1282, 4946, 4948, 4951, 8229, 9044, 13488, 17584, 28709
5347	808, 848-849, 855, 866, 872, 878, 880, 915, 1025, 1123-1125, 1131, 1154, 1158, 1200, 1208, 1251, 1283, 4951, 13488, 17584
5348	37, 259, 273, 275, 277-278, 280, 284-285, 297, 437, 500, 808, 819, 850, 858, 860-861, 863, 865, 871-872, 901-902, 923-924, 1051, 1140-1149, 1153-1158, 1160-1162, 1164, 1200, 1208, 1252, 1275, 4946, 8229, 13488, 17584, 28709
5349	813, 869, 875, 1148, 1200, 1208, 1253, 1280, 4909, 4971, 9061, 13488, 17584
5350	857, 920, 1026, 1155, 1200, 1208, 1254, 1281, 4953, 9049, 13488, 17584
5351	424, 856, 862, 867, 916, 1200, 1208, 1255, 4899, 4952, 5012, 9048, 12712, 13488, 17584
5352	420, 864, 1046, 1089, 1200, 1208, 1256, 4960, 5142, 8612, 9056, 9238, 13488, 16804, 17248, 17584
5353	901-902, 921-922, 1112, 1122, 1156-1157, 1200, 1208, 1257, 13488, 17584
5354	1129-1130, 1163, 1164, 1200, 1208, 1258, 13488, 17584
5460	933-934, 944, 949, 1363-1364, 5029, 5045, 9125, 9555, 13221, 13651, 17317, 25510, 25520, 25525, 29616, 29621, 33717, 37813
5477	935-936, 1381, 1386, 1388, 5031, 5482, 5484, 9127, 13223, 25512
5482	935, 1381, 1386, 1388, 5031, 5477, 5484, 9127, 13223
5484	935-936, 946, 1381, 1386, 1388, 5031, 5477, 5482, 9127, 13223, 25512
5488	1388
8229	37, 256, 273, 275, 277-278, 280, 284-285, 290, 297, 367, 420, 423-424, 437, 500, 720, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-866, 869-871, 874-875, 880, 897, 903-905, 912, 914-916, 920-924, 1009, 1025-1027, 1040-1043, 1047, 1051, 1088, 1097, 1100, 1112, 1114-1115, 1122, 1124, 1126, 1130-1132, 1137, 1140-1149, 1250-1255, 1257-1258, 1275, 1280-1281, 1283, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5210-5211, 5346, 5348, 8482, 8612, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 16804, 17248, 25473, 25479, 25480, 25617, 25619, 25664, 28709

表 710. IBM MQ for z/OS CCSID 変換サポート (続き)	
CCSID	変換元および変換先の CCSID
8482	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 437, 500, 737, 775, 819, 833, 836, 850, 852, 855, 857, 860-865, 870-871, 895-897, 1009, 1025-1027, 1040-1043, 1047, 1088, 1112, 1122, 1139, 1200, 1208, 1252, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 4960, 4992, 5123, 8229, 9025, 9044, 9049, 9056, 13121, 13488, 17248, 17584, 25473, 25617, 25619, 25664, 28709
8492	300-301, 941, 1351, 4396, 16684
8612	37, 256, 420, 424, 437, 500, 720, 737, 775, 819, 850, 852, 857, 860-865, 1008, 1046, 1089, 1098, 1112, 1122, 1127, 1200, 1208, 1252, 1256, 4946, 4948, 4953, 4960, 5104, 5142, 5352, 8229, 9044, 9049, 9056, 9238, 13488, 16804, 17248, 17584, 28709
9025	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 437, 500, 737, 775, 819, 833, 836, 850, 852, 855, 857, 860-865, 870-871, 891, 1009, 1025-1027, 1040-1043, 1088, 1112, 1122, 1126, 1252, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 4960, 5123, 8229, 8482, 9044, 9049, 9056, 13121, 17248, 25617, 25619, 25664, 28709
9026	834, 926, 951, 1362, 4930
9027	835, 927, 947, 1200, 1208, 4931, 13488, 17584, 21427
9030	37, 256, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 737, 775, 813, 819, 838, 850, 852, 857, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1112, 1122, 1200, 1208, 1252, 4909, 4934, 4946, 4948, 4953, 4960, 4970-4971, 5012, 5123, 8229, 9044, 9049, 9056, 9061, 9066, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 28709
9044	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 1025-1027, 1040-1043, 1047, 1088, 1097, 1153, 1200, 1208, 1250, 1252, 1282, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4970-4971, 5012, 5123, 5346, 8229, 8482, 8612, 9025, 9030, 9049, 9061, 9066, 13121, 13488, 16804, 17584, 25473, 25479, 25617, 25619, 25664, 28709
9048	867, 1200, 1208, 4899, 5351, 12712, 13488, 17584
9049	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 1025-1027, 1040-1043, 1088, 1097, 1155, 1200, 1208, 1252, 1254, 1281, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4970-4971, 5012, 5123, 5350, 8229, 8482, 8612, 9025, 9030, 9044, 9061, 9066, 13121, 13488, 16804, 17584, 25473, 25479, 25617, 25619, 25664, 28709
9056	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 500, 720, 819, 833, 838, 850, 864, 870-871, 875, 880, 905, 918, 1008, 1025-1027, 1046, 1089, 1097, 1127, 1200, 1208, 1252, 1256, 4386, 4929, 4934, 4946, 4960, 4971, 5104, 5123, 5142, 5352, 8229, 8482, 8612, 9025, 9030, 9238, 13121, 13488, 16804, 17248, 17584, 28709

表 710. IBM MQ for z/OS CCSID 変換サポート (続き)	
CCSID	変換元および変換先の CCSID
9061	37, 256, 259, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 737, 813, 819, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1200, 1208, 1252-1254, 1280, 4909, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 5349, 8229, 9030, 9044, 9049, 9066, 13488, 17584, 25473, 25479, 25617, 25619, 28709
9066	37, 259, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 813, 819, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1200, 1208, 1252, 4909, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 8229, 9030, 9044, 9049, 9061, 13488, 17584, 25473, 25479, 25617, 25619, 28709
9122	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
9124	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
9125	933-934, 944, 949, 1363-1364, 5029, 5045, 5460, 9555, 13221, 13651, 17317, 25510, 25520, 25525, 29616, 29621, 33717, 37813
9127	935-936, 946, 1381, 1386, 1388, 5031, 5477, 5482, 5484, 13223, 25512
9131	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
9135	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
9142	937-938, 948, 950, 1370, 5033, 5046, 25514, 25524, 29620
9238	420, 500, 864, 1046, 1089, 1127, 1200, 1208, 1256, 4960, 5142, 5352, 8612, 9056, 13488, 16804, 17248, 17584
9555	933, 949, 1363-1364, 5029, 5045, 5460, 9125, 13221, 13651, 17317, 25525, 29621, 33717, 37813
12712	862, 867, 1148, 1156-1157, 1200, 1208, 4899, 5351, 9048, 13488, 17584
13121	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 437, 500, 737, 775, 819, 833, 836, 850, 852, 855, 857, 860-865, 870-871, 891, 1009, 1025-1027, 1040-1043, 1088, 1112, 1122, 1126, 1200, 1208, 1252, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 4960, 5123, 8229, 8482, 9025, 9044, 9049, 9056, 13488, 17248, 17584, 25617, 25619, 25664, 28709
13218	930-932, 939, 942-943, 1200, 1208, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13219, 13231, 13488, 17314, 17584, 25508, 25518, 29614, 33698-33700, 37796
13219	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
13221	933-934, 944, 949, 1363-1364, 5029, 5045, 5460, 9125, 9555, 13651, 17317, 25510, 25520, 25525, 29616, 29621, 33717, 37813

表 710. IBM MQ for z/OS CCSID 変換サポート (続き)	
CCSID	変換元および変換先の CCSID
13223	935-936, 946, 1381, 1386, 1388, 5031, 5477, 5482, 5484, 9127, 25512
13231	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 17314, 25508, 25518, 29614, 33698-33700, 37796
13488	37, 256, 259, 273, 275, 277-278, 280, 282, 284-285, 290, 293, 297, 300-301, 367, 420, 423-424, 437, 500, 720, 737, 775, 803, 806, 808, 813, 819, 833-838, 848-852, 855-872, 874-875, 878, 880, 891, 895-897, 901-905, 912, 914-916, 918, 920-924, 927-928, 930, 932-933, 935, 937, 939, 941-944, 946-951, 1004, 1006, 1008-1019, 1025-1027, 1040-1043, 1046-1047, 1051, 1088-1089, 1097-1098, 1112, 1114-1115, 1122-1126, 1129-1133, 1137, 1140-1149, 1153-1160, 1164, 1166, 1200, 1208, 1250-1258, 1275-1277, 1280-1285, 1351, 1362-1364, 1370-1371, 1380-1381, 1385-1386, 1388, 1390, 1399, 4899, 4909, 4930, 4933, 4948, 4951-4952, 4960, 4971, 5012, 5039, 5104, 5123, 5142, 5210, 5346-5354, 8482, 8612, 9027, 9030, 9044, 9048-9049, 9056, 9061, 9066, 9238, 12712, 13121, 13218, 16684, 16804, 17248, 17584, 21427, 28709
13651	933, 949, 1363-1364, 5029, 5045, 5460, 9125, 9555, 13221, 17317, 25525, 29621, 33717, 37813
16684	300-301, 941, 1200, 1208, 1351, 4396, 8492, 13488, 17584
16804	37, 256, 420, 424, 437, 500, 720, 737, 775, 819, 850, 852, 857, 860-865, 1008, 1046, 1089, 1098, 1112, 1122, 1127, 1200, 1208, 1252, 1256, 4946, 4948, 4953, 4960, 5104, 5142, 5352, 8229, 8612, 9044, 9049, 9056, 9238, 13488, 17248, 17584, 28709
17248	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 500, 720, 819, 833, 838, 850, 864, 870-871, 875, 880, 905, 918, 1008, 1025-1027, 1046, 1089, 1097, 1127, 1200, 1208, 1252, 1256, 4386, 4929, 4934, 4946, 4960, 4971, 5104, 5123, 5142, 5352, 8229, 8482, 8612, 9025, 9030, 9056, 9238, 13121, 13488, 16804, 17584, 28709
17314	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 25508, 25518, 29614, 33698-33700, 37796
17317	933-934, 944, 949, 1363-1364, 5029, 5045, 5460, 9125, 9555, 13221, 13651, 25510, 25520, 25525, 29616, 29621, 33717, 37813
17584	37, 256, 259, 273, 275, 277-278, 280, 282, 284-285, 290, 293, 297, 300-301, 367, 420, 423-424, 437, 500, 720, 737, 775, 803, 806, 808, 813, 819, 833-838, 848-852, 855-872, 874-875, 878, 880, 891, 895-897, 901-905, 912, 914-916, 918, 920-924, 927-928, 930, 932-933, 935, 937, 939, 941-944, 946-951, 1004, 1006, 1008-1019, 1025-1027, 1040-1043, 1046-1047, 1051, 1088-1089, 1097-1098, 1112, 1114-1115, 1122-1126, 1129-1133, 1137, 1140-1149, 1153-1160, 1164, 1166, 1200, 1208, 1250-1258, 1275-1277, 1280-1285, 1351, 1362-1364, 1370-1371, 1380-1381, 1385-1386, 1388, 1390, 1399, 4899, 4909, 4930, 4933, 4948, 4951-4952, 4960, 4971, 5012, 5039, 5104, 5123, 5142, 5210, 5346-5354, 8482, 8612, 9027, 9030, 9044, 9048-9049, 9056, 9061, 9066, 9238, 12712, 13121, 13218, 13488, 16684, 16804, 17248, 21427, 28709
21427	835, 927, 947, 1200, 1208, 4931, 9027, 13488, 17584

表 710. IBM MQ for z/OS CCSID 変換サポート (続き)	
CCSID	変換元および変換先の CCSID
25473	37, 273, 277-278, 280, 284-285, 290, 297, 423, 437, 500, 813, 819, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1252, 4386, 4909, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 8229, 8482, 9030, 9044, 9049, 9061, 9066, 25479, 25617, 25619, 28709
25479	37, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 813, 819, 836, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1115, 1252, 4909, 4932, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 5211, 8229, 9030, 9044, 9049, 9061, 9066, 25473, 25617, 25619, 28709
25480	37, 500, 904, 1114, 5210, 8229, 28709
25508	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25518, 29614, 33698-33700, 37796
25510	933-934, 949, 5029, 5045, 5460, 9125, 13221, 17317, 25525, 29621, 33717, 37813
25512	935-936, 946, 1381, 5031, 5477, 5484, 9127, 13223
25514	937-938, 950, 1370, 5033, 5046, 9142
25518	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 29614, 33698-33700, 37796
25520	933, 944, 949, 5029, 5045, 5460, 9125, 13221, 17317, 25525, 29616, 29621, 33717, 37813
25524	937, 948, 950, 1370, 5033, 5046, 9142, 29620
25525	933-934, 944, 949, 1363-1364, 5029, 5045, 5460, 9125, 9555, 13221, 13651, 17317, 25510, 25520, 29616, 29621, 33717, 37813
25617	37, 273, 277-278, 280, 284-285, 290, 297, 367, 423, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 895-897, 903, 912, 916, 1025-1027, 1040-1043, 1088, 1252, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4970-4971, 4992, 5012, 5123, 8229, 8482, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 25473, 25479, 25619, 25664, 28709
25619	37, 273, 277-278, 280, 284-285, 290, 297, 423, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 1025-1027, 1040-1043, 1088, 1114, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4970-4971, 5012, 5123, 5210, 8229, 8482, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 25473, 25479, 25617, 25664, 28709
25664	37, 273, 277-278, 280, 284-285, 290, 297, 367, 500, 819, 833, 836, 850, 852, 855, 857, 870-871, 875, 891, 1025-1027, 1040-1043, 1088, 1126, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 4971, 5123, 8229, 8482, 9025, 9044, 9049, 13121, 25617, 25619, 28709

表 710. IBM MQ for z/OS CCSID 変換サポート (続き)	
CCSID	変換元および変換先の CCSID
28709	37, 256, 273, 275, 277-278, 280, 284-285, 290, 297, 367, 420, 423-424, 437, 500, 720, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-866, 869-871, 874-875, 880, 897, 903-905, 912, 914-916, 920-924, 1009, 1025-1027, 1040-1043, 1047, 1051, 1088, 1097, 1100, 1112, 1114-1115, 1122, 1124, 1126, 1130-1132, 1137, 1140-1149, 1200, 1208, 1250-1255, 1257-1258, 1275, 1280-1281, 1283, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5210-5211, 5346, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 16804, 17248, 17584, 25473, 25479, 25480, 25617, 25619, 25664
29614	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 33698-33700, 37796
29616	933, 944, 949, 5029, 5045, 5460, 9125, 13221, 17317, 25520, 25525, 29621, 33717, 37813
29620	937, 948, 950, 1370, 5033, 5046, 9142, 25524
29621	933-934, 944, 949, 1363-1364, 5029, 5045, 5460, 9125, 9555, 13221, 13651, 17317, 25510, 25520, 25525, 29616, 33717, 37813
33698	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33699-33700, 37796
33699	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698, 33700, 37796
33700	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33699, 37796
33717	933-934, 944, 949, 1363-1364, 5029, 5045, 5460, 9125, 9555, 13221, 13651, 17317, 25510, 25520, 25525, 29616, 29621, 37813
37796	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700
37813	933-934, 944, 949, 1363-1364, 5029, 5045, 5460, 9125, 9555, 13221, 13651, 17317, 25510, 25520, 25525, 29616, 29621, 33717

EBCDIC と Latin 1 コードページの相違点

多くの国では、歴史的な理由から独自の EBCDIC コードページを有している。EBCDIC コードページが定義された際、ラテン 1 関連の全コードページに含まれる 192 文字はすべて含まれているが、コードポイントは異なる。したがって、EBCDIC コードページでデータを入力・表示しながら、データ変換に別のコードページ（デフォルト z/OS であるコードページ 500 がよく使われる）を使用すると、一部の文字が正しく表示されない場合があります。

以下の表は、一般的な欧米のコードページにおける 16 進値の違いを示しています。コードページ 1047 はコンパイラのデフォルトのコード z/OS ページです。最も一般的な値と異なる 16 進数値は下線が引かれています。

表 711. 各種 EBCDIC コードページにおける文字とその 16 進値

キャラクタ (名前)	37 (ア メリ カ、カ ナダ)	273 (ドイ ツ、オ ースト リア)	277 (デン マー ク、ノ ルウェ ー)	278 (フィン ランド、ス ウェー デン)	280 (イタ リア)	284 (スペ イン)	285 (英国)	297 (フラン ス)	500 (国際)	871 (アイ スラン ド)	1047 (ラテン 1、オー プンシ ステム ズ)
【(開き 括弧)	BA	63	9E	B5	90	4A	B1	90	4A	AE	西暦
】(閉じ 括弧)	BB	FC (FC)	9F	9F	51	5A	BB	B5	5A	9E	BD
{(開き 括弧)	C0	44	9C	44	44	C0	C0	51	C0	8E	C0
} (閉じ 括弧)	D0	DC	47	47	54	D0	D0	54	D0	9C	D0
~ (波形 記号)	A1	59	DC	DC	58	BD	紀元前	BD	A1	CC	A1
@ (ア ットマ ーク)	7C	B5	80	EC	B5	7C	7C	44	7C	AC	7C
(縦 棒)	6A	CC	70	CC	CD	49	6A	DD	6A	6A	6A
# (番号 記号)	7B	7B	4A	63	B1	69	7B	B1	7B	7B	7B
` (アク サンテ ギュ)	79	79	79	51	DD	79	79	A0	79	8C	79
^ (キャ レット、サ ーカム フレッ クス)	B0	5F	5F	5F	5F	BA	BA	5F	5F	EC	5F
§ (セ クショ ン記 号)	B5	7C	B5	4A	7C	B5	B5	5A	B5	B5	B5
!	5A	4F	4F	4F	4F	BB	5A	4F	4F	4F	5A
\$ (ドル 記号)	5B	5B	67	67	5B	5B	4A	5B	5B	5B	5B
¬ (論 理否 定)	5F	BA	BA	BA	BA	5F	5F	BA	BA	BA	B0
™ (ウム ラウト)	BD	BD	BD	BD	BD	A1	BD	A1	BD	BD	BB

表 711. 各種 EBCDIC コードページにおける文字とその 16 進値 (続き)

キャラクタ (名前)	37 (ア メリ カ、カ ナダ)	273 (ドイ ツ、オ ースト リア)	277 (デン マー ク、ノ ルウェ ー)	278 (フィン ランド、ス ウェー デン)	280 (イタ リア)	284 (スペ イン)	285 (英国)	297 (フラン ス)	500 (国際)	871 (アイ スラン ド)	1047 (ラテン 1、 オープ ンシス テム ズ)
£ (ポ ンド記 号)	B1	B1	B1	B1	<u>7B</u>	B1	<u>5B</u>	<u>7B</u>	B1	B1	B1
¢ (セ ント記 号)	<u>4A</u>	B0	B0	B0	B0	B0	B0	B0	B0	B0	<u>4A</u>
l (縦 棒、論 理和)	<u>4F</u>	BB	BB	BB	BB	<u>4F</u>	<u>4F</u>	BB	BB	BB	<u>4F</u>
¤ (国 際通 貨、ス プート ニク)	9F	9F	<u>5A</u>	<u>5A</u>	9F	9F	9F	9F	9F	9F	9F
μ (ミ ュ ー)	A0	A0	A0	A0	A0	A0	A0	<u>79</u>	A0	A0	A0
ß (ベ ー タ)	59	<u>A1</u>	59	59	59	59	59	59	59	59	59
\ (バ ッ ク ス ラ ッ シ ュ)	E0	<u>EC</u>	E0	<u>71</u>	<u>48</u>	E0	E0	<u>48</u>	E0	<u>BE</u>	E0
° (度記 号)	90	90	90	90	<u>4A</u>	90	90	<u>4A</u>	90	90	90
¯ (オー バー ライ ン)	紀元前	紀元前	紀元前	紀元前	紀元前	紀元前	<u>A1</u>	紀元前	紀元前	紀元前	紀元前
˘ (アク サンテ ギユ)	BE	BE	BE	BE	BE	BE	BE	BE	BE	<u>E0</u>	BE
ä (ウム ラウト 付き の a)	43	<u>C0</u>	43	<u>C0</u>	43	43	43	43	43	43	43
à (アク セント 記号付 き a)	44	44	44	44	<u>C0</u>	44	44	<u>7C</u>	44	44	44

表 711. 各種 EBCDIC コードページにおける文字とその 16 進値 (続き)

キャラクタ (名前)	37 (ア メリ カ、カ ナダ)	273 (ドイ ツ、オ ースト リア)	277 (デン マー ク、ノ ルウェ ー)	278 (フィン ランド、ス ウェー デン)	280 (イタ リア)	284 (スペ イン)	285 (英国)	297 (フラ ンス)	500 (国際)	871 (アイ スラン ド)	1047 (ラテン 1、 オープ ンシス テム ズ)
å (ア の上 点)	47	47	<u>D0</u>	<u>D0</u>	47	47	47	47	47	47	47
Ä (ウ ムラウ ト付き の A)	63	<u>4A</u>	63	<u>7B</u>	63	63	63	63	63	63	63
Å (ア ロベ ル)	67	<u>67</u>	5B	<u>5B</u>	67	67	67	67	67	67	67
æ (ae 合字)	9C	<u>9C</u>	C0	<u>9C</u>	9C	9C	9C	9C	9C	D0	9C
Æ (AE 合字)	9E	<u>9E</u>	7B	<u>9E</u>	9E	9E	9E	9E	9E	5A	9E
ç (セデ ィーユ 付き c)	48	<u>48</u>	48	<u>48</u>	E0	48	48	E0	48	48	48
ð (小文 字のエ ト)	8C	<u>8C</u>	8C	<u>8C</u>	8C	8C	8C	8C	8C	79	8C
Ð (大文 字のエ ト)	AC	<u>AC</u>	AC	<u>AC</u>	AC	AC	AC	AC	AC	7C	AC
é (アク セント 記号付 きの e)	51	<u>51</u>	51	<u>79</u>	5A	51	51	C0	51	51	51
è (アク セント 記号付 きの e)	54	<u>54</u>	54	<u>54</u>	D0	54	54	D0	54	54	54
エ (エ にアク セント 記号)	71	<u>71</u>	71	<u>E0</u>	71	71	71	71	71	71	71

表 711. 各種 EBCDIC コードページにおける文字とその 16 進値 (続き)

キャラクタ (名前)	37 (ア メリ カ、カ ナダ)	273 (ドイ ツ、オ ースト リア)	277 (デン マー ク、ノ ルウェ ー)	278 (フィン ランド、ス ウェー デン)	280 (イタ リア)	284 (スペ イン)	285 (英国)	297 (フラ ンス)	500 (国際)	871 (アイ スラン ド)	1047 (ラテン 1、オープ ンシス テムズ)
i (アクセント 記号付きの i)	58	<u>58</u>	58	<u>58</u>	A1	58	58	58	58	58	58
ñ (ティ ルデ付 き n)	49	<u>49</u>	49	<u>49</u>	49	6A	49	49	49	49	49
Ñ (ティ ルデ付 き N)	69	<u>69</u>	69	<u>69</u>	69	7B	69	69	69	69	69
ø (o に スラッ シュ)	70	<u>70</u>	6A	<u>70</u>	70	70	70	70	70	70	70
Ø (O に 斜線)	80	<u>80</u>	7C	<u>80</u>	80	80	80	80	80	80	80
ö (ウム ラウト 付きの o)	CC	<u>6A</u>	CC	<u>6A</u>	CC	CC	CC	CC	CC	A1	CC
ò (アクセント 記号付きの o)	CD	<u>CD</u>	CD	<u>CD</u>	6A	CD	CD	CD	CD	CD	CD
ü (ウム ラウト 付きの u)	DC	<u>D0</u>	A1	<u>A1</u>	DC	DC	DC	DC	DC	DC	DC
Û (ウに アクセ ント記 号)	DD	<u>DD</u>	DD	<u>DD</u>	79	DD	DD	6A	DD	DD	DD
Ö (ウム ラウト 付きの O)	EC	<u>E0</u>	EC	<u>7C</u>	EC	EC	EC	EC	EC	5F	EC
Ü (ウム ラウト 付きの U)	FC (FC)	<u>5A</u>	FC (FC)	<u>FC</u>	FC (FC)	FC (FC)	FC (FC)	FC (FC)	FC (FC)	FC (FC)	FC (FC)

表 711. 各種 EBCDIC コードページにおける文字とその 16 進値 (続き)

キャラクタ (名前)	37 (ア メリ カ、カ ナダ)	273 (ドイ ツ、オ ースト リア)	277 (デン マー ク、ノ ルウェ ー)	278 (フィン ランド、ス ウェー デン)	280 (イタ リア)	284 (スペ イン)	285 (英国)	297 (フラン ス)	500 (国際)	871 (アイ スラン ド)	1047 (ラテン 1、 オープ ンシス テム ズ)
Ÿ (アク セント 記号付 きの Y)	西暦	西暦	西暦	西暦	西暦	西暦	西暦	西暦	西暦	西暦	BA
␣ (小文 字のソ ー)	8E	8E	8E	8E	8E	8E	8E	8E	8E	C0	8E
␣ (大文 字のソ ー)	AE	AE	AE	AE	AE	AE	AE	AE	AE	4A	AE

EBCDIC と ASCII の制御コード文字の違い

EBCDIC と ASCII 間の制御文字の変換は問題を引き起こす可能性があります。一部のプラットフォームではこれらの文字を変換しようと試みるが、ASCII の制御文字用コードポイントは 32 しかないのに対し、EBCDIC は 64 を割り当てている。そのため、EBCDIC の制御文字のうち 32 個が ASCII の文字値に変換される。以下の表は、データ変換時に一般的に保持される制御文字の一覧です。この表には NL (15 進数 15) 文字は含まれていません。ほとんどの ASCII CCSID には改行文字が含まれていません。この IBM MQ 文字の変換方法を指定するには、[mqsr.ini ファイルの AllQueueManagers セクション](#)で設定するか、[AMQ_CONVEBDICNEWLINE](#) 環境変数を使用します。

名前	EBCDIC 16 進コード	ASCII 16 進コード
NUL (ヌル)	00	00
見出しの開始	01	01
STX (テキストの開始)	02	02
ETX (ヘッダー終了)	03	03
HT (水平タブ)	05	09
VT (垂直タブ)	0B	0B
FF (フォームフィード)	0C	0C
CR (改行)	0D	0D
SO (シフトアウト)	0E	0E
SI (シフトイン)	0F	0F
DLE (データリンクエスケープ)	10	10
DC1 (デバイス制御 1)	11	11
DC2 (デバイス制御 2)	12	12
DC3 (デバイス制御 3)	13	13
BS (バックスペース)	16	08

名前	EBCDIC 16 進コード	ASCII 16 進コード
CAN (キャンセル)	18	18
EM (メッセージ終了)	19	19
FS (ファイル区切り文字)	1C	1C
IGS (インターチェンジグループセパレータ)	1D	1D
IRS (インターチェンジレコードセパレータ)	1E	1E
IUS/ITB (インターチェンジユニットセパレータまたは中間伝送ブロック)	1F	1F
LF (行送り)	25	0A
ETB (伝送ブロック終了)	26	17
ESC (エスケープ)	27	1B
ENQ (問い合わせ)	2D	05
ACK (応答)	2E	06
ベル	2F	07
SYN (同期アイドル)	32	16
EOT (伝送ブロック終了)	37	04
DC4 (デバイス制御 4)	3C	14
NAK (応答なし)	3D	15
SUB (代用)	3F	1A

IBM i 変換サポート

CCSID の全リスト、および IBM i でサポートされる変換については、適切な IBM i 資料に記載されています。

サポート対象のコード・ページは、[サポートされる CCSID マッピング](#)にリストされています。

Unicode 変換サポート

Unicode エンコードとユーザー・データ間の変換がサポートされているプラットフォームもあります。Unicode エンコードは、UTF-16 (CCSID 1200、13488、および 17584) および UTF-8 (CCSID 1208) の 2 つの形式がサポートされています。CCSID 1200 または 1208 を使用してください。これらはサポートされている最新の Unicode バージョンを表すためです。

IBM MQ データ変換において、Unicode 8.0 標準で定義されているすべての Unicode 文字をサポートします。このサポートには、UTF-16 の完全なサポートが含まれます。これには、UTF-16 サロゲートペア (X'D800' から X'DFFF' の範囲にある 2 バイトの UTF-16 文字のペアで、U+FFFF を超える Unicode コードポイントを表現するもの) も含まれます。ターゲット CCSID が、UTF-16 サロゲートペアで表現されるコードポイントのマッピングを含まない場合、その文字ペアは単一の置換文字に変換される。

IBM MQ 結合文字シーケンスをサポートします。このサポートにより、場合によっては、ソース CCSID の事前合成文字がターゲット CCSID の合成文字シーケンスに変換されるか、その逆の変換が行われることがあります。

注: IBM MQ では、UTF-16 キュー・マネージャー CCSID がサポートされていないため、メッセージ・ヘッダー・データは UTF-16 でエンコードできません。

IBM MQ AIX の Unicode サポート



以下のリストに記載されている非 Unicode CCSID については、サポートされている Unicode CCSID（推奨は 1200 または 1208）との IBM MQ for AIX 相互変換がサポートされています：

037
273、278、280、284、285、297
423、437
500
813、819、850、852、856、857、858、860、861、865、867、869、875、878、880
901、902、912、915、916、920、923、924、932、933、935、937、938、939、942、943、
948、949、950、954、964、970
1026、1046、1089
1129、1130、1131、1132、1133、1140、1141、1142、1143、1144、1145、1146、1147、
1148、1149、1153、1156、1157
1200、1208、1250、1251、1253、1254、1258、1280、1281、1282、1283、1284、1285
1363、1364、1381、1383、1386、1388
4899
5026、5035、5050、5346、5347、5348、5349、5350、5351、5352、5353、5354、5488
9044、9048、9449
12712
13488
17584
33722

IBM MQ for Windows および Linux Unicode のサポート



および IBM MQ for Windows において IBM MQ for Linux、以下のリストに記載された非 Unicode CCSID について、サポートされている Unicode CCSID（推奨は 1200 または 1208）との相互変換がサポートされています：

037,
277、278、280、284、285、290、297
300、301
420、424、437
500
813、819、833、835、836、837、838、850、852、855、856、857、858、860、861、862、863、864、
865、866、867、868、869、870、871、874、875、878、880、891、897
901、902、903、904、912、[913^{993 ページの『5』}](#)、915、916、918、920、921、922、923、924、927、928、930、
[931^{993 ページの『1』}](#)、[932^{993 ページの『2』}](#)、933、935、937、[938^{993 ページの『3』}](#)、939、941、942、943、
947、948、949、950、951、[954^{993 ページの『4』}](#)、964、970
1006、1025、1026、1027、1040、1041、1042、1043、1046、1047、1051、1088、1089、
1097、1098
1112、1114、1115、1122、1123、1124、1129、1130、1132、1133、1140、1141、1142、1143、
1144、1145、1146、1147、1148、1149、1153、1156、1157
1200、1208、1250、1251、1252、1253、1254、1255、1256、1257、1258、1275、1280、
1281、1282、1283
1363、1364、1374、1375、1376、1377、1378、1379、1380、1381、1383、1386、1388
4899
5050、5346、5347、5348、5349、5350、5351、5352、5353、5354、5488[^{993 ページの『5』}](#)
9044、9048、9449
12712

13488
17584
33722^{993 ページの『4』}

注:

1. 931 では、変換に 939 を使用します。
2. 932 では、変換に 942 を使用します。
3. 938 では、変換に 948 を使用します。
4. 954 および 33722 では、変換に 5050 を使用します。
5. オン Windows および Linux のみ。

IBM i の Unicode サポート



UNICODE サポートの詳細については、ご使用のオペレーティング・システムに関連した、適切な IBM i 資料を参照してください。

IBM MQ for z/OS の Unicode サポート



以下のリストに記載されている非 Unicode CCSID については、サポートされている Unicode CCSID（推奨は 1200 または 1208）との IBM MQ for z/OS 相互変換がサポートされています：

37
256、259、273、275、277、278、280、282、284、285、290、293、297
300、301、367
420、423、424、437
500
720、737、775
803、806、808、813、819、833、834、835、836、837、838、848、849、850、851、852、
855、856、857、858、859、860、861、862、863、864、865、866、867、868、869、870、
871、872、874、875、878、880、891、895、896、897
901、902、903、904、905、912、914、915、916、918、920、921、922、923、924、927、928、930、932、
933、935、937、939、941、942、943、944、946、947、948、949、950、951
1004、1006、1008、1009、1010、1011、1012、1013、1014、1015、1016、1017、1018、1019、1025、
1026、1027、1040、1041、1042、1043、1046、1047、1051、1088、1089、1097、1098
1112、1114、1115、1122、1123、1124、1125、1126、1129、1130、1131、1132、1133、1137、
1140、1141、1142、1143、1144、1145、1146、1147、1148、1149、1153、1154、1155、
1156、1157、1158、1159、1160、1161、1162、1164
1200、1208、1250、1251、1252、1253、1254、1255、1256、1257、1258、1275、1276、1277、1280、
1281、1282、1283、1284、1285
1351、1362、1363、1364、1370、1371、1380、1381、1385、1386、1388、1390、1399
4899、4909、4930、4933、4948、4951、4952、4960、4971
5012 5039 5104 5123 5142 5210 5346 5347 5348 5349 5350 5351 5352 5353 5354 5488
8482 8612
9027903090449048904990569061906692389449
1166
12712
13121、13218、13488、1374、1375、1376、1377、1378、1379
16684、16804
17248、17584
21427
28709

サポートが終了した CCSID

. IBM Semeru Runtime 21 でサポートされていないコードセットおよび各コードセットに対応する CCSID に関する情報。

以下の機種では CCSID に対応していません AIX, Linux, and Windows



AIX, Linux, and Windows に記載 [994 ページの表 712](#) されている CCSID は、において IBM Semeru Runtime 21 サポートされていません。

表 712. コードセット名と CCSID	
コードセット名	CCSID
x-IBM301	cp301、 ibm301、 301、 ibm-301
x-IBM720	ibm-720, ibm720, 720, cp720
x-IBM835	ibm835, 835, cp835, ibm-835
x-IBM836	cp836, ibm836, 836, ibm-836
x-IBM837	cp837、 ibm837、 837、 ibm-837
x-IBM867	ibm867, 867, cp867, ibm-867
x-IBM897	ibm-897、 ibm897、 897、 cp897
IBM00924	ibm924, ebcdic-latin9--euro, cp924, 924, ibm-924, cp00924, ibm00924, ccsid00924
x-IBM927	ibm-927、 ibm927、 927、 cp927
x-IBM941	ibm-941、 ibm941、 941、 cp941
x-IBM947	cp947、 ibm947、 947、 ibm-947
x-IBM951	ibm-951、 ibm951、 951、 cp951
x-IBM1027	cp1027、 ibm1027、 ibm-1027、 1027
x-IBM1041	cp1041, ibm1041, ibm-1041, 1041
x-IBM1043	ibm-1043, 1043, cp1043, ibm1043
x-IBM1088	1088, ibm1088, cp1088, ibm-1088
x-IBM1114	cp1114, ibm-1114, 1114, ibm1114
x-IBM1115	cp1115、 ibm1115, ibm-1115, 1115
x-IBM1130	ibm1130, ibm-1130, 1130, cp1130
x-IBM1153	cp01153, ibm1153, ccsid01153, ibm01153, ibm-1153, 1153, cp1153
x-MacRoman	macroman, 1275, ibm1275, cp1275, ibm-1275
x-MacGreek	macgreek, ibm1280, ibm-1280, 1280, cp1280
x-MacTurkish	ibm-1281, 1281, cp1281, ibm1281, macturkish
x-MacCentralEurope	cp1282, ibm-1282, 1282, maccentraleurope, ibm1282
x-MacCyrillic	cp1283、 ibm1283, maccyrillic, ibm-1283, 1283
x-MacCroatian	ibm1284, ibm-1284, 1284, cp1284, maccroatian

表 712. コードセット名と CCSID (続き)

コードセット名	CCSID
x-MacRomania	ibm-1285,1285,cp1285, ibm1285,macromania
x-IBM1351	ibm-1351,1351,cp1351, ibm1351
x-IBM1362	ibm-1362,1362,ibm1362, cp1362
x-IBM1363	1363,ibm1363, cp1363,ibm-1363
x-IBM1370	cp1370, ibm-1370,1370,ibm1370
x-IBM1371	ibm1371,ibm-9563,cp9563,9563,x-ibm9563, cp1371,ibm-1371,1371,ibm9563
x-IBM1380	ibm-1380,1380,cp1380, ibm1380
x-IBM1385	1385,ibm1385, cp1385,ibm-1385
x-IBM1386	cp1386, ibm1386, x-ibm1386,ibm-1386,1386
x-IBM1388	ibm-1388,1388,cp1388, ibm1388
x-IBM1390	ibm1390, ibm-1390,1390,cp1390
x-IBM1399	ibm-1399,1399,cp1399, ibm1399
x-IBM4933	cp4933, ibm4933,4933,ibm-4933
x-IBM16684	cp16684、 ibm16684,ibm-16684,16684
UTF16	UTF_16、Unicode、utf16、 UnicodeBig, UCS-2

以下の機種では CCSID に対応していません z/OS



z/OS に記載 995 ページの表 713 されている CCSID は、において IBM Semeru Runtime 21 サポートされていません。

表 713. コードセット名と CSSID

コードセット名	CCSID
x-IBM301	cp301、 ibm301、 301、 ibm-301
x-IBM835	ibm835, 835, cp835, ibm-835
x-IBM836	cp836, ibm836, 836,ibm-836
x-IBM837	cp837、 ibm837、 837、 ibm-837
x-IBM897	ibm-897、 ibm897、 897、 cp897
x-IBM927	ibm-927、 ibm927、 927、 cp927
x-IBM941	ibm-941、 ibm941、 941、 cp941
x-IBM947	cp947、 ibm947、 947、 ibm-947
x-IBM951	ibm-951、 ibm951、 951、 cp951
x-IBM1041	cp1041, ibm1041, ibm-1041,1041
x-IBM1043	ibm-1043,1043,cp1043, ibm1043
x-IBM1088	1088,ibm1088, cp1088,ibm-1088

表 713. コードセット名と CCSID (続き)	
コードセット名	CCSID
x-IBM1114	cp1114, ibm-1114,1114,ibm1114
x-IBM1115	cp1115、 ibm1115,ibm-1115,1115
x-IBM1130	ibm1130, ibm-1130,1130,cp1130
x-MacRoman	macroman,1275,ibm1275, cp1275,ibm-1275
x-MacGreek	macgreek,ibm1280,ibm-1280,1280,cp1280
x-MacTurkish	ibm-1281,1281,cp1281, ibm1281,macturkish
x-MacCentralEurope	cp1282, ibm-1282,1282,maccentraleurope,ibm1282
x-MacCyrillic	cp1283、 ibm1283,maccyrillic,ibm-1283,1283
x-MacCroatian	ibm1284, ibm-1284,1284,cp1284,maccroatian
x-MacRomania	ibm-1285,1285,cp1285, ibm1285,macromania
x-IBM1351	ibm-1351,1351,cp1351, ibm1351
x-IBM1362	ibm-1362,1362,ibm1362, cp1362
x-IBM1370	cp1370, ibm-1370,1370,ibm1370
x-IBM1380	ibm-1380,1380,cp1380, ibm1380
x-IBM1385	1385,ibm1385, cp1385,ibm-1385
x-IBM1386	cp1386, ibm1386, x-ibm1386,ibm-1386,1386
x-IBM4933	cp4933, ibm4933,4933,ibm-4933
UTF16	UTF_16、 Unicode、 utf16、 UnicodeBig, UCS-2

注 : 995 ページの表 713IBM MQ 9.4.5 から、の Java runtime environment (JRE)は z/OS 、内の CCSID を除き、をサポート IBM Semeru Runtime 21 します。

関連資料

[MFT で使用できるコード・ページ](#)

64 ビット・プラットフォームでのコーディング標準

この情報を使用して、64 ビット・プラットフォームでのコーディング標準および優先データ・タイプについて学びます。

優先データ・タイプ

以下に示すデータ・タイプは、サイズが不変で、32 ビットおよび 64 ビットの両方の IBM MQ プラットフォームで使用可能です。

表 714. データ・タイプ名および長さ

名前	長さ
MQLONG	4 バイト
MQULONG	4 バイト
MQINT32	4 バイト
MQUINT32	4 バイト

表 714. データ・タイプ名および長さ (続き)

名前	長さ
MQINT64	8 バイト
MQUINT64	8 バイト

AIX, Linux, and Windows における標準データ・タイプ

32 ビット AIX and Linux アプリケーション、および 64 ビット AIX, Linux, and Windows アプリケーションにおける標準データ・タイプについて学習します。

32 ビットの AIX and Linux アプリケーション



表 715. 32 ビットの AIX and Linux アプリケーションのデータ・タイプの名前および長さ

名前	長さ
char	1 バイト
short	2 バイト
int	4 バイト
long	4 バイト
float	4 バイト
double	8 バイト
long double	8 バイト
pointer	4 バイト
ptrdiff_t	4 バイト
size_t	4 バイト
time_t	4 バイト
clock_t	4 バイト
wchar_t	4 バイト

 注: AIX の場合、wchar_t は 2 バイトです。

64 ビットの AIX and Linux アプリケーション



表 716. 64 ビットの AIX and Linux アプリケーションのデータ・タイプの名前および長さ

名前	長さ
char	1 バイト
short	2 バイト
int	4 バイト
long	8 バイト
float	4 バイト
double	8 バイト
long double	8 バイト

表 716. 64 ビットの AIX and Linux アプリケーションのデータ・タイプの名前および長さ (続き)

名前	長さ
pointer	8 バイト
ptrdiff_t	8 バイト
size_t	8 バイト
time_t	8 バイト
clock_t	4 バイト
wchar_t	4 バイト

 注: AIX の場合、wchar_t は 2 バイトです。

Windows 64 ビット・アプリケーション



表 717. Windows 64 ビット・アプリケーションのデータ・タイプ名および長さ

名前	長さ
char	1 バイト
short	2 バイト
int	4 バイト
long	4 バイト
float	4 バイト
double	8 バイト
long double	8 バイト
pointer	8 バイト
	注: pointer はすべて 8 バイトです。
ptrdiff_t	8 バイト
size_t	8 バイト
time_t	8 バイト
clock_t	4 バイト
wchar_t	2 バイト
WORD	2 バイト
DWORD	4 バイト
HANDLE	8 バイト
HFILE	4 バイト

Windows におけるコーディングの考慮事項



HANDLE hf;

以下を使用してください。

```
hf = CreateFile((LPCTSTR) FileName,
               Access,
```

```
ShareMode,
xihSecAttsNTRestrict,
Create,
AttrAndFlags,
NULL);
```

以下は使用しないでください。

```
HFILE hf;
hf = (HFILE) CreateFile((LPCTSTR) FileName,
                        Access,
                        ShareMode,
                        xihSecAttsNTRestrict,
                        Create,
                        AttrAndFlags,
                        NULL);
```

このコードでは、エラーが発生します。

size_t len fgets

以下を使用してください。

```
size_t len
while (fgets(string1, (int) len, fp) != NULL)
len = strlen(buffer);
```

以下は使用しないでください。

```
int len;

while (fgets(string1, len, fp) != NULL)
len = strlen(buffer);
```

printf

以下を使用してください。

```
printf("My struc pointer: %p", pMyStruc);
```

以下は使用しないでください。

```
printf("My struc pointer: %x", pMyStruc);
```

16 進数出力を必要とする場合、上位および下位の 4 バイトを別個にプリントする必要があります。

char *ptr

以下を使用してください。

```
char * ptr1;
char * ptr2;
size_t bufLen;

bufLen = ptr2 - ptr1;
```

以下は使用しないでください。

```
char *ptr1;
char *ptr2;
UINT32 bufLen;

bufLen = ptr2 - ptr1;
```

alignBytes

以下を使用してください。

```
alignBytes = (unsigned short) ((size_t) address % 16);
```

以下は使用しないでください。

```
void *address;
unsigned short alignBytes;

alignBytes = (unsigned short) ((UINT32) address % 16);
```

len

以下を使用してください。

```
len = (UINT32) ((char *) address2 - (char *) address1);
```

以下は使用しないでください。

```
void *address1;
void *address2;
UINT32 len;

len = (UINT32) ((char *) address2 - (char *) address1);
```

sscanf

以下を使用してください。

```
MQLONG SBCSprt;

sscanf(line, "%d", &SBCSprt);
```

以下は使用しないでください。

```
MQLONG SBCSprt;

sscanf(line, "%1d", &SBCSprt);
```

%1d は、8 バイトの型を 4 バイトの型に書き込もうとします。%1 は、実際に long データ・タイプを取り扱う場合にのみ使用してください。MQLONG、UINT32、および INT32 は、すべての IBM MQ プラットフォームにおいて 4 バイトと定義 int されます。これは、

IBM i IBM i アプリケーション・プログラミングの参照情報 (ILE/RPG)

IBM i のためのアプリケーション・プログラミング。

以下の情報を使用して、IBM i のアプリケーションの開発に役立ててください。

- [1002 ページの『IBM i でのデータ・タイプの説明』](#)
- [1260 ページの『IBM i での関数呼び出し』](#)
- [1380 ページの『IBM i でのオブジェクトの属性』](#)
- [1426 ページの『IBM i の戻りコード \(ILE RPG\)』](#)
- [1428 ページの『IBM i の MQI オプションの妥当性検査に関する規則 \(ILE RPG\)』](#)
- [1430 ページの『IBM i でのマシン・エンコーディング』](#)
- [1433 ページの『IBM i でのレポート・オプションおよびメッセージ・フラグ』](#)

IBM i での RPG および COBOL アプリケーションの互換モードの非推奨

IBM i

IBM MQ for IBM i 9.0 以降では、互換モードと呼ばれる動的リンケージを使用する RPG または COBOL アプリケーションがサポートされなくなりました。この操作モードは、MQSeries 5.1 より前に作成されたアプリケーションが必要でした。また、この製品の後続バージョンではこれらのアプリケーション用に互換性のあるランタイム環境が提供されていました (ただし、それらのアプリケーションをコンパイルするのに必要なコピーブックは IBM WebSphere MQ 6.0 で削除されていました)。動的リンケージ (互換モード) は、

ライブラリー QMQM 内の以下のプログラムで提供されていました。これらのプログラムは IBM MQ for IBM i 9.0 では除去されています。

- AMQVSTUB
- AMQZSTUB
- QMQM
- MQCLOSE
- MQCONN
- MQDISC
- MQGET
- MQINQ
- MQOPEN
- MQPUT
- MQPUT1
- MQSET

IBM MQ for IBM i 9.0 以降、この互換操作モードを使用するアプリケーションは、LIBMQM と LIBMQM_R のサービス・プログラムによって提供される静的にバインドされた MQ 呼び出しを使用するように、再コンパイルする必要があります。サンプル・プログラム (AMQ3PUT4 や AMQ3GET4) によって、このプログラミング・モデルの使用方法を学べます。これらの MQ 呼び出しの使用については、[IBM i アプリケーション・プログラミングの参照情報 \(ILE/RPG\)](#) を参照してください。

注：

- CALL 'QMQM' インターフェースを現在使用しているアプリケーションは、LIBMQM サービス・プログラムを代わりに使用するように再コーディングが必要です。

上のリストに含まれているプログラム・オブジェクトやサービス・プログラム (QMQM、MQCONN、MQPUT、AMQVSTUB、AMQZSTUB など) は、IBM MQ for IBM i 9.0 で削除されており、互換モードを使用するようにコーディングされていたアプリケーションは機能しなくなります。

- IBM MQ for IBM i 8.0 で LIBMQM サービス・プログラムにアプリケーションをバインドしていた場合は、IBM MQ for IBM i 9.0 以降でそのアプリケーションを再コンパイルしたり再リンクしたりする必要はありません。
- IBM MQ for IBM i の複数のバージョンを同じパーティションにインストールすることはできません。

RPG または COBOL プログラムが互換モードを使用しているかどうかを調べるには、**DSPPGMREF** (プログラム参照表示) コマンドを使用して、アプリケーション・プログラムによって呼び出された外部プログラムを表示します。このセクションにリストされているプログラムへの参照がある場合、そのプログラムは IBM MQ for IBM i 9.0 以降では実行されません。次の **DSPPGMREF** の出力例は、非推奨の 3 つのプログラム・オブジェクト MQCONN、MQOPEN、MQCLOSE を示しています。

```
Program . . . . . : MYAPPPGM
Library . . . . . : MYLIB
Text 'description'. . . . . : ILE/COBOL SAMPLE PUT TO QUEUE (MQPUT)
Number of objects referenced . . . . . : 5
Object . . . . . : MQCONN
Library . . . . . : *LIBL
Object type . . . . . : *PGM
Object . . . . . : MQOPEN
Library . . . . . : *LIBL
Object type . . . . . : *PGM
Object . . . . . : MQCLOSE
Library . . . . . : *LIBL
Object type . . . . . : *PGM
```

このようなプログラムは、[IBM i での COBOL プログラムの作成](#)で説明している結合プロシージャ呼び出しメソッドを使用して再コンパイルする必要があります。

互換モードを使用する IBM MQ for IBM i 9.0 以降のアプリケーション・プログラムを実行しようとする場合に、最もよく発生する最初のエラーは、プログラム MQCONN または QMQM を呼び出そうとしている MCH3401 です。

関連タスク

IBM iでの RPG プログラムの作成

RPG サンプルプログラムを使用して IBM i

IBM i IBM iでのデータ・タイプの説明

この一連のトピックでは、IBM iプログラミングで使用されるデータ・タイプを説明します。

データ・タイプの説明で使用する規則

ここでは、各基本データ・タイプについて、プログラミング言語に依存しない形で、その使用法について説明します。各説明のあとに、RPG プログラム言語の ILE バージョンで一般的な宣言を示します。一貫性を持たせるため、基本データ・タイプの定義をここに示します。RPG で使用する「D」仕様では、必要な任意の属性を使用して作業フィールドを宣言することができます。ただし、これはフィールドが使用される演算仕様書で行います。

基本データ・タイプを使用するには、次のいずれかを作成します。

- すべてのデータ・タイプが入った /COPY メンバー、または
- すべてのデータ・タイプが入った外部データ構造体 (PF)。次に、該当するデータ・タイプ・フィールドに「似た」属性を持つ作業フィールドを指定する必要があります。

2 番目のオプションの利点は、他の IBM i オブジェクト用の「フィールド参照ファイル」として定義を使用できることです。IBM MQ データ・タイプ定義が変更された場合、これらのオブジェクトの再作成は比較的簡単です。

基本データ型について IBM i

このセクションで説明する他のデータ・タイプはすべて、これらの基本データ・タイプとまったく等しいか、またはこれらの基本データ・タイプの集合体 (配列または構造体) と等しくなります。

表 718. 基本データ・タイプ名、タイプ、および説明		
基本データ型	データ・タイプ	説明
MQBOOL	10 桁の符号付き整数	MQBOOL データ・タイプはブール値を表します。値 0 は偽を表します。その他の値は真を表します。MQBOOL は MQLONG データ・タイプに関する限り位置合わせしなけりばなりません。
MQBYTE	1 バイト英数字フィールド	MQBYTE データ・タイプは、1 バイトのデータを表します。このバイトには特に解釈は与えられていない。ビット列として扱われ、二進数や文字として扱われることはありません。特殊な位置合わせは必要ありません。 MQBYTE の配列は、キュー・マネージャでは性質が認識されない主記憶域の領域を表す場合に使用されることがあります。例えば、その領域にアプリケーション・メッセージ・データまたは構造体が入っている可能性がある場合です。この領域の境界合わせは、その領域に含まれるデータの性質と合っていなければなりません。

表 718. 基本データ・タイプ名、タイプ、および説明 (続き)		
基本データ型	データ・タイプ	説明
MQBYTE16	16 バイト英数字フィールド	各 MQBYTE n データ型は、 n バイトの文字列を表します。ここで n は次のいずれかの値を取ります：16、24、32、または 64。
MQBYTE24	24 バイト英数字フィールド	
MQBYTE32	32 バイト英数字フィールド	
MQBYTE64	64 バイト英数字フィールド	各バイトは MQBYTE データ・タイプにより記述されます。特殊な位置合わせは必要ありません。 ストリング内のデータがストリングの定義長より短い場合は、ストリングの定義長に達するまで、データをヌルで埋める必要があります。 キュー・マネージャーは、アプリケーションにバイト・ストリングを戻すときは (MQGET 呼び出しなど) 常に、ストリングの定義長に達するまでヌルを埋め込みます。 バイト・ストリング・フィールドの長さを定義する定数を使用することができます。
MQCHAR	1 バイト英数字フィールド	MQCHAR データ・タイプは、単一の文字を表します。 この文字のコード化文字セット識別子は、キューマネージャのものと同じです (トピック内の CodedCharSetId 属性 CodedCharSetId 参照)。特殊な位置合わせは必要ありません。 注：MQGET、MQPUT、および MQPUT1 呼び出しで指定されるアプリケーションメッセージデータは、MQCHAR データ型ではなく MQBYTE データ型で記述されます。

表 718. 基本データ・タイプ名、タイプ、および説明 (続き)

基本データ型	データ・タイプ	説明
MQCHAR4	4 バイト英数字フィールド	各 MQCHARn データ型は、n 文字の文字列を表します。ここで n は次のいずれかの値を取ります：4、8、12、16、20、28、32、48、64、128、または 256。 各文字は MQCHAR データ・タイプにより定義されます。特殊な位置合わせは必要ありません。 ストリング内のデータがストリングの定義長より短い場合は、ストリングの定義長になるまで、データをブランクで埋める必要があります。ブランクを埋め込む代わりにヌル文字を使用して、ストリングを途中で終了させることができる場合があります。この場合、ヌル文字とそれに続く文字は、ストリングの定義長に達するまでブランクとして取り扱われます。ヌル文字を使用できる位置は、呼び出し記述およびデータ・タイプ記述に示されています。 キュー・マネージャーは、(例えば MQGET 呼び出しで) アプリケーションに文字ストリングを戻すときは常に、ストリングの定義長に達するまでブランクを埋め込みます。キュー・マネージャーは、ストリングを区切るためにヌル文字を使用することはありません。 文字ストリング・フィールドの長さを定義する定数を使用することができます。
MQCHAR8	8 バイト英数字フィールド	
MQCHAR12	12 バイト英数字フィールド	
MQCHAR16	16 バイト英数字フィールド	
MQCHAR20	20 バイト英数字フィールド	
MQCHAR28	28 バイト英数字フィールド	
MQCHAR32	32 バイト英数字フィールド	
MQCHAR48	48 バイト英数字フィールド	
MQCHAR64	64 バイト英数字フィールド	
MQCHAR128	128 バイト英数字フィールド	
MQCHAR256	256 バイト英数字フィールド	
MQFLOAT32	4 バイト浮動小数点数	MQFLOAT32 データ・タイプは、標準の IEEE 浮動小数点形式を使用して表される 32 ビット浮動小数点数です。MQFLOAT32 は、4 バイトの境界に位置合わせされていなければなりません。
MQFLOAT64	8 バイト浮動小数点数	MQFLOAT64 データ・タイプは、標準の IEEE 浮動小数点形式を使用して表される 64 ビット浮動小数点数です。MQFLOAT64 は、8 バイトの境界に位置合わせされていなければなりません。

表 718. 基本データ・タイプ名、タイプ、および説明 (続き)

基本データ型	データ・タイプ	説明
MQHCONFIG	構成ハンドル	MQHCONFIG データ・タイプは、構成ハンドル、つまり特定のインストール可能なサービス用に構成されているコンポーネントを表します。構成ハンドルは、その本来の境界に位置合わせされる必要があります。 注: このタイプの変数については、値が等しいかどうかのみをアプリケーションでテストする必要があります。
MQHCONN	10 桁の符号付き整数	MQHCONN データ・タイプは、接続ハンドル、つまり特定のキュー・マネージャーへの接続を表します。接続ハンドルは、その本来の境界に位置合わせされていなければなりません。 注: このタイプの変数については、値が等しいかどうかのみをアプリケーションでテストする必要があります。
MQHMSG	メッセージ・ハンドル	MQHMSG データ・タイプは、メッセージにアクセスできるようにするメッセージ・ハンドルを表します。メッセージ・ハンドルは、8 バイト境界に位置合わせされていなければなりません。 注: このタイプの変数については、値が等しいかどうかのみをアプリケーションでテストする必要があります。
MQHOBJ	10 桁の符号付き整数	MQHOBJ データ・タイプは、オブジェクトへのアクセスを提供するオブジェクト・ハンドルを表します。オブジェクト・ハンドルは、その本来の境界に位置合わせされていなければなりません。 注: このタイプの変数については、値が等しいかどうかのみをアプリケーションでテストする必要があります。
MQINT8	8 ビットの符号付き整数	MQINT8 データ・タイプは 8 ビットの符号付き整数で、コンテキストにより特に制限されていない限り、-128 から +127 までの範囲内の任意の値をとることができます。

表 718. 基本データ・タイプ名、タイプ、および説明 (続き)

基本データ型	データ・タイプ	説明
MQINT16	16 ビットの符号付き整数	MQINT16 データ・タイプは 16 ビットの符号付き整数で、コンテキストにより特に制限されていない限り、-32 768 から +32 767 までの範囲内の任意の値をとることができます。MQINT16 は、2 バイトの境界に位置合わせされていなければなりません。
MQINT32	32 ビットの符号付き整数	MQINT32 データ・タイプは、32 ビットの符号付き整数です。 <u>MQLONG</u> に相当します。
MQINT64	64 ビットの符号付き整数	MQINT64 データ・タイプは 64 ビットの符号付き整数で、コンテキストにより特に制限されていない限り、-9 223 372 036 854 775 808 から +9 223 372 036 854 775 807 の範囲内に任意の値をとることができます。 COBOL の場合、有効範囲は -999 999 999 999 999 から +999 999 999 999 999 に制限されます。MQINT64 は 8 バイト境界に位置合わせする必要があります。
MQLONG	32 ビットの符号付き整数	MQLONG データ・タイプは 32 ビットの符号付き 2 進整数で、境界に位置合わせされるようなコンテキストにで特に制限されていない限り、-2 147 483 648 から +2 147 483 647 の範囲内で任意の値をとることができます。
MQPID	プロセス ID	IBM MQ プロセス ID。これは IBM MQ トレースおよび FFST ダンプで使用されるものと同じ ID ですが、オペレーティング・システムのプロセス ID とは異なる場合があります。
MQPTR	ポインター	MQPTR データ・タイプは、すべてのタイプのデータのアドレスです。ポインターは、その自然な境界上に位置合わせされなければなりません。これは、IBM i 上の 16 バイト境界です。 一部のプログラミング言語では型付きポインターをサポートします。いくつかのケースでは MQI もこれらを使用します。

表 718. 基本データ・タイプ名、タイプ、および説明 (続き)

基本データ型	データ・タイプ	説明
MQTID	スレッド ID	IBM MQ スレッド ID。これはトレース FFSIBM MQ およびダンプで使用されるのと同じ識別子ですが、オペレーティングシステムのスレッド識別子とは異なる場合があります。
MQUINT8	8 ビットの符号なし整数	MQUINT8 データ・タイプは 8 ビットの符号なし整数で、コンテキストにより特に制限されていない限り、0 から +255 までの範囲内の任意の値をとることができます。
MQUINT16	16 ビットの符号なし整数	MQUINT16 データ・タイプは 16 ビットの符号なし整数で、コンテキストにより特に制限されていない限り、0 から +65 535 までの範囲内の任意の値をとることができます。MQUINT16 は、2 バイトの境界に位置合わせされていなければなりません。
MQUINT32	32 ビットの符号なし整数	MQUINT32 データ・タイプは、32 ビットの符号なし整数です。それは MQULONG に相当する。
MQUINT64	64 ビットの符号なし整数	MQUINT64 データ・タイプは 64 ビットの符号なし整数で、コンテキストにより特に制限されていない限り、0 から +18 446 744 073 709 551 615 の範囲内に任意の値をとることができます。 COBOL の場合、有効範囲は 0 から +999 999 999 999 999 999 に制限されます。MQUINT64 は 8 バイト境界に位置合わせする必要があります。
MQULONG	32 ビットの符号なし整数	MQULONG データ・タイプは 32 ビットの符号なし 2 進整数で、コンテキストにより特に制限されていない限り、0 から +4 294 967 294 の範囲内の任意の値をとることができます。 MQULONG は、4 バイトの境界に位置合わせされていなければなりません。
PMQACH	ポインター	タイプ MQACH のデータ構造体へのポインター。
PMQAIR	ポインター	MQAIR 型のデータ構造へのポインター。

表 718. 基本データ・タイプ名、タイプ、および説明 (続き)

基本データ型	データ・タイプ	説明
PMQAXC	ポインター	タイプ MQAXC のデータ構造体へのポインター。
PMAXP	ポインター	タイプ MQAXP のデータ構造体へのポインター。
PMQBMHO	ポインター	<u>M</u> QBMHO 型のデータ構造へのポインタ。
PMQBO	ポインター	<u>MQBO</u> 型のデータ構造へのポインタ。
PMQBOOL	ポインター	タイプ MQBOOL のデータへのポインター。
PMQBYTE	ポインター	データ・タイプ MQBYTE へのポインター。
PMQBYTE _n	ポインター	MQBYTE <i>n</i> 型のデータ構造へのポインタ。ここで <i>n</i> は 8、12、16、24、32、40、48、または 128 のいずれかである。
PMQCBC	ポインター	<u>MQCBC</u> 型のデータ構造へのポインタ。
PMQCBD	ポインター	タイプ MQCBD のデータ構造体へのポインター。
PMQCHAR	ポインター	タイプ MQCHAR のデータへのポインター。
PMQCHARV	タイプ MQCHARV のデータ構造体へのポインター	タイプ MQCHARV のデータ構造体へのポインター。
PMQCHAR _n	タイプ MQCHAR _n のデータへのポインター	<u>MQCHAR_n</u> 型のデータへのポインタ。ここで <i>n</i> は 4、8、12、20、28、32、64、128、256、264 のいずれかである。
PMQCIH	ポインター	タイプ MQCIH のデータ構造体へのポインター。
PMQCMHO	ポインター	タイプ MQCMHO のデータ構造体へのポインター。
PMQCNO	ポインター	タイプ MQCNO のデータ構造体へのポインター。
PMQCSP	ポインター	タイプ MQCSP のデータ構造体へのポインター。
PMQCTLO	ポインター	タイプ MQCTLO のデータ構造体へのポインター。
PMQDH	ポインター	タイプ MQDH のデータ構造体へのポインター。
PMQDHO	ポインター	タイプ MQDHO のデータ構造体へのポインター。

表 718. 基本データ・タイプ名、タイプ、および説明 (続き)

基本データ型	データ・タイプ	説明
PMQDLH	ポインター	タイプ MQDLH のデータ構造体へのポインター。
PMQDMHO	ポインター	タイプ MQDMHO のデータ構造体へのポインター。
PMQDMPO	ポインター	タイプ MQDMPO のデータ構造体へのポインター。
PMQEPH	ポインター	タイプ MQEPH のデータ構造体へのポインター。
PMQFLOAT32	ポインター	タイプ MQFLOAT32 のデータへのポインター。
PMQFLOAT64	ポインター	タイプ MQFLOAT64 のデータへのポインター。
PMQFUNC	ポインター	関数へのポインター。
PMQGMO	ポインター	タイプ MQGMO のデータ構造体へのポインター。
PMQHCONFIG	ポインター	データ・タイプ MQHCONFIG へのポインター。
PMQHCONN	ポインター	データ・タイプ MQHCONN へのポインター。
PMQHMSG	ポインター	データ・タイプ MQHMSG へのポインター。
PMQHOBJ	タイプ MQHOBJ のデータへのポインター	タイプ MQSMPO のデータへのポインター。
PMQIIH	ポインター	タイプ MQIIH のデータ構造体へのポインター。
PMQIMPO	ポインター	タイプ MQIMPO のデータ構造体へのポインター。
PMQINT8	ポインター	タイプ MQINT8 のデータへのポインター。
PMQINT16	ポインター	タイプ MQINT16 のデータへのポインター。
PMQINT32	ポインター	PMQINT32 データ・タイプは MQINT32 タイプのデータへのポインターです。 PMQLONG と同等です。
PMQINT64	ポインター	PMQINT64 データ・タイプは MQINT64 タイプのデータへのポインターです。
PMQLONG	ポインター	タイプ MQLONG のデータへのポインター。
PMQMD	ポインター	タイプ MQMD の構造体へのポインター。

表 718. 基本データ・タイプ名、タイプ、および説明 (続き)

基本データ型	データ・タイプ	説明
PMQMDE	ポインター	タイプ MQMDE のデータ構造体へのポインター。
PMQMD1	ポインター	タイプ MQMDI のデータ構造体へのポインター。
PMQMD2	ポインター	MQMD2 型のデータ構造へのポインター。
PMQMHBO	ポインター	タイプ MQMHBO のデータ構造体へのポインター。
PMQOD	ポインター	タイプ MQOD のデータ構造体へのポインター。
PMQOR	ポインター	タイプ MQOR のデータ構造体へのポインター。
PMQPD	ポインター	タイプ MQPD のデータ構造体へのポインター。
PMQPID	ポインター	プロセス ID へのポインター。
PMQPMO	ポインター	タイプ MQPMO のデータ構造体へのポインター。
PMQPTR	ポインター	タイプ MQPTR のデータへのポインター。
PMQRFH	ポインター	タイプ MQRFH のデータ構造体へのポインター。
PMQRFH2	ポインター	タイプ MQRFH2 のデータ構造体へのポインター。
PMQRMH	ポインター	タイプ MQRMH のデータ構造体へのポインター。
PMQRR	ポインター	タイプ MQRR のデータ構造体へのポインター。
PMQSCO	ポインター	タイプ MQSCO のデータ構造体へのポインター。
PMQSD	ポインター	タイプ MQSD のデータ構造体へのポインター。
PMQSMPO	ポインター	タイプ MQSMPO のデータ構造体へのポインター。
PMQSRO	ポインター	タイプ MQSRO のデータ構造体へのポインター。
PMQSTS	ポインター	タイプ MQSTS のデータ構造体へのポインター。
PMQTID	ポインター	タイプ MQTID のデータ構造体へのポインター。
PMQTM	ポインター	タイプ MQTM のデータ構造体へのポインター。

表 718. 基本データ・タイプ名、タイプ、および説明 (続き)

基本データ型	データ・タイプ	説明
PMQTM2	ポインター	タイプ MQTM2 のデータ構造体へのポインター。
PMQUINT8	ポインター	タイプ MQUINT8 のデータへのポインター。
PMQUINT16	ポインター	タイプ MQUINT16 のデータへのポインター。
PMQUINT32	ポインター	PMQUINT32 データ・タイプは MQUINT32 タイプのデータへのポインターです。 PMQULONG と同等である。
PMQUINT64	ポインター	PMQUINT64 データ・タイプは MQUINT64 タイプのデータへのポインターです。
PMQULONG	ポインター	タイプ MQULONG のデータへのポインター。
PMQVOID	ポインター	ポインター。
PMQWIH	ポインター	タイプ MQWIH のデータ構造体へのポインター。
PMQXQH	ポインター	タイプ MQXQH のデータ構造体へのポインター。

言語に関する考慮事項

このトピックでは、RPG プログラミング言語から MQI を使用する際に役立つ情報を記載しています。

これらの言語に関する考慮事項は、次のとおりです。

- [1011 ページの『COPY ファイル』](#)
- [1013 ページの『呼び出し』](#)
- [1013 ページの『呼び出しパラメーター』](#)
- [1013 ページの『構造体』](#)
- [1014 ページの『名前付き定数』](#)
- [1014 ページの『MQI プロシージャ』](#)
- [1014 ページの『スレッド化に関する考慮事項』](#)
- [1015 ページの『コミットメント制御』](#)
- [1015 ページの『バインド済み呼び出しのコーディング』](#)
- [1016 ページの『表記上の規則』](#)

COPY ファイル

メッセージ・キューイングを使用する RPG アプリケーション・プログラムの作成に役立つように、各種の COPY ファイルが提供されています。次の 3 組の COPY ファイルがあります。

- 名前が文字 G で終る COPY ファイルは、静的リンケージを使用するプログラム用です。これらのファイルは、[1013 ページの『構造体』](#)に示す例外を除き、初期化されます。
- 名前が文字 H で終る COPY ファイルは、静的リンケージを使用するプログラム用ですが、**初期化されません**。

- 名前が文字 *R* で終る COPY ファイルは、動的リンケージを使用するプログラム用です。これらのファイルは、1013 ページの『構造体』に示す例外を除き、初期化されます。

これらの COPY ファイルは QMQM ライブラリーの QRPGLSRC に常駐します。

COPY ファイルの各セットには、名前定数を含むファイルが 2 つあり、各構造体についてファイルが 1 つずつあります。COPY ファイルの要約を 1012 ページの表 719 に示します。

表 719. RPG の COPY ファイル			
ファイル名 (静的リンケージ、初期化あり、CMQ*G)	ファイル名 (静的リンケージ、初期化なし、CMQ*H)	ファイル名 (動的リンケージ、初期化あり、CMQ*R)	内容
CMQBOG	CMQBOH	-	開始オプション構造体
CMQCDG	CMQCDH	CMQCDR	チャネル定義構造体
CMQCFBFG	CMQCFBFH	-	PCF ビット・フィルタ・パラメーター
CMQCFG	-	-	PCF およびイベントの定数
CMQCFBSG	CMQCFBSH	-	PCF バイト・ストリング
CMQCFGRG	CMQCFGRH	-	PCF グループ・パラメーター
CMQCFIFG	CMQCFIFH	-	PCF 整数フィルタ・パラメーター
CMQCFHG	CMQCFHH	-	PCF ヘッダー
CMQCFILG	CMQCFILH	-	PCF 整数リスト・パラメーター構造体
CMQCFING	CMQCFINH	-	PCF 整数パラメーター構造体
CMQCFSG	CMQCFSH	-	PCF ストリング・フィルタ・パラメーター
CMQCFSLG	CMQCFSLH	-	PCF ストリング・リスト・パラメーター構造体
CMQCFSTG	CMQCFSTH	-	PCF ストリング・パラメーター構造体
CMQCFXLG	CMQCFXLH	-	PCF ショート・ネーム (CFIL64 用)
CMQCFXNG	CMQCFXNH	-	PCF ショート・ネーム (CFIN64 用)
CMQCIHG	CMQCIHH	-	CICS 情報ヘッダー構造体
CMQCNQG	CMQCNQH	-	接続オプション構造体
CMQCSPG	CMQCSPH	-	セキュリティ・パラメーター。
CMQCXPG	CMQCXPH	CMQCXPR	チャネル出口パラメーター構造体
CMQDHG	CMQDHH	CMQDHR	配布ヘッダー構造体
CMQDLHG	CMQDLHH	CMQDLHR	送達不能ヘッダー構造体
CMQDXPG	CMQDXPH	CMQDXPR	データ変換出口パラメーター構造体
CMQEPHG	CMQEPHH	-	組み込み PCF ヘッダー構造体
CMQG	-	CMQR	メイン MQI 用の名前付き定数
CMQGMQG	CMQGMQH	CMQGMOR	読み取りメッセージ・オプション構造体
CMQIIHG	CMQIIHH	CMQIIHR	IMS 情報ヘッダー構造体
CMQMDEG	CMQMDEH	CMQMDER	拡張メッセージ記述子構造体
CMQMDG	CMQMDH	CMQMDR	メッセージ記述子構造体

表 719. RPG の COPY ファイル (続き)			
ファイル名 (静的リ ンケージ、初期化あ り、CMQ*G)	ファイル名 (静的リ ンケージ、初期化な し、CMQ*H)	ファイル名 (動的リ ンケージ、初期化あ り、CMQ*R)	内容
CMQMD1G	CMQMD1H	CMQMD1R	メッセージ記述子構造体 - バージョン 1
CMQMD2G	CMQMD2H	-	メッセージ記述子構造体 - バージョン 2
CMQODG	CMQODH	CMQODR	オブジェクト記述子構造体
CMQORG	CMQORH	CMQORR	オブジェクト・レコード構造体
CMQPMOG	CMQPMOH	CMQPMOR	書き込みメッセージ・オプション構造体
CMQPSG	-	-	Publish/Subscribe の定数
CMQRFHG	CMQRFHH	-	規則および書式ヘッダー構造体
CMQRFH2G	CMQRFH2H	-	規則およびフォーマット・ヘッダー 2 構造体
CMQRMHG	CMQRMHH	CMQRMHR	参照メッセージ・ヘッダー構造体
CMQRRG	CMQRRH	CMQRRR	応答レコード構造体
CMQTMCG	CMQTMCH	CMQTMCR	トリガー・メッセージ構造体 (文字形式)
CMQTMCG	CMQTMCH	CMQTMCR	トリガー・メッセージ構造体 (文字形式) - バージョン 2
CMQTMG	CMQTMH	CMQTMR	トリガー・メッセージ構造体
CMQWIHG	CMQWIHH	-	作業情報ヘッダー構造体
CMQXG	-	CMQXR	データ変換出口用の名前付き定数
CMQXQHG	CMQXQHH	CMQXQHR	伝送キュー・ヘッダー構造体

呼び出し

呼び出しは、呼び出しの個別の名前を使用して説明されます。

呼び出しパラメーター

MQI に渡されるいくつかのパラメーターは、複数の並行関数を持つことができます。これは、渡される整数値が、合計値についてではなく フィールド内の個々のビットの設定値について頻繁にテストされるためです。その結果、いくつかの関数を「加算」して、それらを単一のパラメーターとして渡すことができます。

構造体

以下の例外を除いて、すべての IBM MQ 構造体はフィールドの初期値で定義されます。

- 接尾部が H の構造体
- MQTMC
- MQTMC2

これらの初期値は、各構造体の関連テーブルで定義されます。

構造体の宣言に DS ステートメントは含まれていません。したがって、アプリケーションでは、次に示すように、DS ステートメントをコーディングしてから、/COPY ステートメントを使用して宣言の残りの部分をコピーし、1 以上のデータ構造体を宣言することができます。

```
D*.1.....2.....3.....4.....5.....6.....7
D* Declare an MQMD data structure with 5 occurrences
DMYMD          DS          5
D/COPY CMQMDR
```

名前付き定数

アプリケーション・プログラムとキュー・マネージャー間のデータ交換を提供する多くの整数値および文字値があります。読みやすくするため、値の使用に一貫したアプローチを与えるため、名前定数が定義されています。名前定数が表す値ではなく、名前定数自体を使用することができます。そうすれば、プログラム・ソース・コードがさらに読みやすくなります。

COPY ファイル CMQG をプログラムに組み込んで定数を定義する場合は、RPG コンパイラーは、プログラムによって使用されない定数について多くの重大度ゼロのメッセージを発行します。これらのメッセージが出て問題はないため、無視しても安全です。

MQI プロシージャ

ILE バインドの呼び出しを使用する場合は、プログラムの作成時に MQI プロシージャにバインドする必要があります。このプロシージャは、以下のサービス・プログラムから適宜エクスポートされます。

QMQM/LIBMQM

このサービス・プログラムには、バージョン 5.1 以降用の単スレッド化バインディングが含まれます。スレッド・アプリケーションを作成する際には、以下のセクションの特別な考慮事項を参照してください。

QMQM/LIBMQM_R

このサービス・プログラムには、バージョン 5.1 以降用のマルチスレッド化バインディングが含まれます。スレッド・アプリケーションを作成する際には、以下のセクションの特別な考慮事項を参照してください。

QMQM/LIBMQIC

このサービス・プログラムは、非スレッド・クライアント・アプリケーションのバインド用に使用されます。

QMQM/LIBMQIC_R

このサービス・プログラムは、スレッド・クライアント・アプリケーションのバインド用に使用されます。

CRTPGM コマンドを使用して、プログラムを作成します。例えば、次のコマンドを使用すると、ILE バインド済み呼び出しを使用する単スレッド・プログラムが作成されます。

```
CRTPGM PGM(MYPROGRAM) BNDSRVPGM(QMQM/LIBMQM)
```

スレッド化に関する考慮事項

IBM i で使用される RPG コンパイラーは、WebSphere Development Toolset および WebSphere Development Studio for IBM i の一部であり、ILE RPG IV コンパイラーとして知られています。

一般的に、RPG プログラムではマルチスレッド化サービス・プログラムを使用しないでください。例外は、ILE RPG IV コンパイラーを使用して作成され、制御仕様書に THREAD(*SERIALIZE) キーワードを含む RPG プログラムです。ただし、これらのプログラムはスレッド・セーフですが、THREAD(*SERIALIZE) によって RPG プロシージャのモジュール・レベルでのシリアライゼーションが強制されるため、全体的なアプリケーション設計については慎重に考慮する必要があります。これは、全体のパフォーマンスに悪影響を与える可能性があります。

RPG プログラムをデータ変換出口として使用する場合は、スレッド・セーフにする必要があります。また、制御仕様書に THREAD(*SERIALIZE)を指定して、バージョン 4.4 以上の ILE RPG コンパイラーを使用して再コンパイルする必要があります。

スレッド化について詳しくは、「IBM i IBM MQ Development Studio: ILE RPG 解説書」および「IBM i IBM MQ Development Studio: ILE RPG プログラマーの手引き」を参照してください。

コミットメント制御

MQI 同期点関数 MQCMIT および MQBACK は、通常モードで実行する ILE RPG プログラムで使用できます。これらの関数を呼び出すと、プログラムは MQ リソースへの変更をコミットしたり、それらの変更を元に戻したりできます。

バインド済み呼び出しのコーディング

1015 ページの表 720 に、MQI ILE プロシージャをリストします。

表 720. 各サービス・プログラムがサポートする ILE RPG バインド済み呼び出し		
呼び出しの名前	LIBMQM および LIBMQM_R	LIBMQIC および LIBMQIC_R
MQBACK	Y	Y
MQBEGIN	Y	Y
MQCMIT	Y	Y
MQCLOSE	Y	Y
MQCONN	Y	Y
MQCONNX	Y	Y
MQDISC	Y	Y
MQGET	Y	Y
MQINQ	Y	Y
MQOPEN	Y	Y
MQPUT	Y	Y
MQPUT1	Y	Y
MQSET	Y	Y
MQXCNVC	Y	Y

これらのプロシージャを使用するには、以下を実行します。

1. 「D」仕様で外部プロシージャを定義します。これらはすべて名前定数を含む COPY ファイル・メンバー CMQGL 内で使用できます。
2. CALLP 操作コードを使用して、プロシージャをそのパラメーターと共に呼び出します。

例えば、MQOPEN 呼び出しでは次のコードの組み込みが必要です。

```

D*****
D**  MQOPEN Call -- Open Object (From COPY file CMQGL)      **
D*****
D*
D*..1.....2.....3.....4.....5.....6.....7..
DMQOPEN          PR          EXTPROC('MQOPEN')
D* Connection handle
D HCONN          10I 0 VALUE
D* Object descriptor
D OBJDSC          224A
D* Options that control the action of MQOPEN
D OPTS           10I 0 VALUE

```

```

D* Object handle
D HOBJ                                10I 0
D* Completion code
D CMPCOD                             10I 0
D* Reason code qualifying CMPCOD
D REASON                             10I 0
D*

```

このプロシーチャーを呼び出すには、各パラメーターの初期設定のあと、次のコードが必要です。

```

...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8
  C              CALLP      MQOPEN(HCONN : MQOD : OPTS : HOBJ :
  C              CMPCOD : REASON)

```

ここでは、MQOD 構造体は COPY メンバー CMQODG を使用して定義され、この COPY メンバーによりコンポーネントに分割されます。

表記上の規則

このセクションの以降のトピックには、以下の方法が示されています。

- 呼び出しの指定方法
- パラメーターの宣言方法
- 各種データ・タイプの宣言方法

多くの場合、パラメーターは、サイズが固定されていない配列または文字ストリングです。これらの場合、小文字の "n" は数値定数を表すために使用されます。そのパラメーターの宣言がコーディングされている場合は、"n" を必要な数値で置き換える必要があります。

IBM i IBM i での MQAIR (認証情報レコード)

MQAIR 構造体は認証情報レコードを表します。

概要

目的: MQAIR 構造体を使用すると、IBM MQ クライアントとして実行するアプリケーションで、クライアント接続に使用する認証子の情報を指定することができます。この構造体は、MQCONN 呼び出しの入力パラメーターです。

文字セットおよびエンコード: MQAIR のデータは、**CodedCharSetId** キュー・マネージャー属性で指定された文字セットと、ENNAT で指定されたローカル・キュー・マネージャーのエンコードになっていなければなりません。

- [1016 ページの『フィールド』](#)
- [1018 ページの『初期値』](#)
- [1019 ページの『RPG 宣言』](#)

フィールド

MQAIR 構造体には、以下のフィールドが含まれます。フィールドは**アルファベット順**に説明されています。

AICN (10 桁の符号付き整数)

これは、LDAP サーバーが稼働しているホストのホスト名またはネットワーク・アドレスのどちらかです。この後にオプションのポート番号を括弧で囲んで指定できます。

値がフィールドの長さより短い場合、値をヌル文字で終了するか、フィールドの長さまでブランクを埋め込みます。値が無効の場合、呼び出しは失敗し、理由コード RC2387 が戻されます。

デフォルトのポート番号は 389 です。

これは入力フィールドです。このフィールドの長さは、LNAICN により指定されます。このフィールドの初期値はブランク文字です。

AITYP (10 桁の符号付き整数)

これは、レコードに含まれる認証情報のタイプです。

値は次のものでなければなりません。

AITLDP

LDAP サーバーを使用するための証明書の失効

値が無効の場合、呼び出しは失敗し、理由コード RC2386 が戻されます。

これは入力フィールドです。このフィールドの初期値は AITLDP です。

AIPW (10 桁の符号付き整数)

これは、LDAP CRL サーバーへのアクセスに必要なパスワードです。

値がフィールドの長さより短い場合、値をヌル文字で終了するか、フィールドの長さまでブランクを埋め込みます。LDAP サーバーがパスワードを必要としない場合や、LDAP ユーザー名を省略する場合は、AIPW はヌルまたはブランクでなければなりません。LDAP ユーザー名を省略する場合、AIPW が非ヌルまたは非ブランクであれば、呼び出しは失敗して理由コード RC2390 が戻されます。

これは入力フィールドです。このフィールドの長さは LNLDPW によって指定されます。このフィールドの初期値はブランク文字です。

AILUL (10 桁の符号付き整数)

これは、AILUP または AILUO フィールドで指定される LDAP ユーザー名の長さ (バイト数) です。値は、ゼロから LNDISN までの範囲内であればなりません。値が無効の場合、呼び出しは失敗し、理由コード RC2389 が戻されます。

関係する LDAP サーバーがユーザー名を必要としない場合は、このフィールドをゼロに設定します。

これは入力フィールドです。このフィールドの初期値は 0 です。

AILUO (10 桁の符号付き整数)

これは、MQAIR 構造体の先頭から LDAP ユーザー名のオフセットをバイト数で表したものです。

オフセットの値は、正負どちらの値にもなります。LDAPUserNameLength がゼロの場合、このフィールドは無視されます。

LDAP ユーザー名の指定には、LDAPUserNamePtr または LDAPUserNameOffset のどちらか一方を使用します。両方とも使用することはできません。詳細については LDAPUserNamePtr フィールドの説明を参照してください。

これは入力フィールドです。このフィールドの初期値は 0 です。

AILUP (10 桁の符号付き整数)

これは LDAP ユーザー名です。

これは、LDAP CRL サーバーへのアクセスを試行するユーザーの Distinguished Name から成ります。AILUL で指定された長さよりも値が短い場合、ヌル文字で値を終了するか、AILUL の長さまでブランクを埋め込んでください。このフィールドは AILUL がゼロの場合、無視されます。

LDAP ユーザー名は、次の 2 つの方法で提供できます。

• AILUP ポインター・フィールドを使用する

この場合、アプリケーションは MQAIR 構造体とは別個のストリングを宣言して、AILUP をそのストリングのアドレスに設定することができます。

他の環境へ移植できる形式のポインター・データ・タイプをサポートするプログラミング言語 (例えば C プログラミング言語など) には、AILUP の使用を検討してください。

• AILUO オフセット・フィールドを使用する

この場合、アプリケーションは MQSCO 構造体と、その後に続く MQAIR レコードの配列、およびその後に続く LDAP ユーザー名のストリングを含む、複合の構造体を宣言する必要があります。さらに

AILUO を、MQAIR 構造体の先頭からの該当する名前ストリングのオフセットに設定する必要があります。この値が正しいこと、および値が *MQLONG* 内に収まることを確認してください (最も制限の大きいプログラミング言語は COBOL で、有効範囲は -999 999 999 から +999 999 999 です)。

ポインター・データ・タイプをサポートしないプログラミング言語や、さまざまな環境に移植できない形式のポインター・データ・タイプを実装するプログラミング言語 (例えば COBOL プログラミング言語など) には、*AILUO* の使用を検討してください。

どちらの方法を選ぶにしても、*AILUP* または *AILUO* のいずれか一方だけを使用してください。呼び出しが失敗すると、理由コード RC2388 が戻されます。

これは入力フィールドです。このフィールドの初期値は、ポインターをサポートするプログラミング言語のヌル・ポインターです。それ以外の場合は、すべてヌルのバイトのストリングです。

注：プログラミング言語がポインターのデータ・タイプをサポートしていないプラットフォームでは、このフィールドは適切な長さのバイト・ストリングとして宣言されます。

AISID (10 桁の符号付き整数)

値は次のものでなければなりません。

AISIDV

認証情報レコードの ID。

これは常に入力フィールドです。このフィールドの初期値は AISIDV です。

AIVER (10 桁の符号付き整数)

値は次のものでなければなりません。

AIVER1

Version-1 認証情報レコード。

以下の定数は、現行バージョンのバージョン番号を指定しています。

AIVERC

認証情報レコードの現行バージョン。

これは常に入力フィールドです。このフィールドの初期値は AIVER1 です。

初期値

表 721. MQAIR における MQAIR のフィールド		
フィールド名	定数の名前	定数の値
<i>AISID</i>	AISIDV	'AIR-'
<i>AIVER</i>	AIVERC	1
<i>AITYP</i>	AITLDP	1
<i>AICN</i>	なし	ヌル・ストリングまたは ブランク
<i>AILUP</i>	なし	ヌル・ポインターまたは ヌル・バイト
<i>AILUO</i>	なし	0
<i>AILUL</i>	なし	0
<i>AIPW</i>	なし	ヌル・ストリングまたは ブランク

注：

1. 記号-は、単一のブランク文字を表します。

RPG 宣言

```
D*..1.....2.....3.....4.....5.....6.....7..
D* MQAIR Structure
D*
D* Structure identifier
D AISID          1          4      INZ('AIR ')
D* Structure version number
D AIVER          5          8I 0  INZ(1)
D* Type of authentication information
D AITYP          9         12I 0  INZ(1)
D* Connection name of CRL LDAP server
D AICN          13         276    INZ
D* Address of LDAP user name
D AILUP         277        292*    INZ(*NULL)
D* Offset of LDAP user name from start of MQAIR structure
D AILUO         293        296I 0  INZ(0)
D* Length of LDAP user name
D AILUL         297        300I 0  INZ(0)
D* Password to access LDAP server
D AIPW          301        332    INZ
```

IBM i IBM i での MQBMHO (バッファからメッセージ・ハンドルへの変換オプション)

バッファからメッセージ・ハンドルへの変換オプションを定義する構造体。

概要

目的: MQBMHO 構造体を使用すると、アプリケーションで、バッファからメッセージ・ハンドルを作成する方法を制御するオプションを指定できます。この構造体は、MQBUFMH 呼び出しの入力パラメーターです。

文字セットとエンコード: MQBMHO 内のデータは、アプリケーションの文字セットおよびアプリケーションのエンコードでなければなりません (ENNAT)。

- [1019 ページの『フィールド』](#)
- [1020 ページの『初期値』](#)
- [1020 ページの『RPG 宣言』](#)

フィールド

MQBMHO 構造体には、以下のフィールドが含まれます。フィールドは**アルファベット順**に説明されています。

BMSID (10 桁の符号付き整数)

バッファからメッセージ・ハンドルへの変換の構造体 - StrucId フィールド。

これは構造体 ID です。値は次のものでなければなりません。

BMSIDV

バッファからメッセージ・ハンドルへの変換構造の ID。

これは常に入力フィールドです。このフィールドの初期値は BMSIDV です。

BMVER (10 桁の符号付き整数)

バッファからメッセージ・ハンドルへの変換の構造体 - Version フィールド。

これは構造体のバージョン番号です。値は次のものでなければなりません。

BMVER1

バッファからメッセージ・ハンドルへの変換構造のバージョン番号。

以下の定数は、現行バージョンのバージョン番号を指定しています。

BMVERVC

バッファからメッセージ・ハンドルへの変換構造の現行バージョン。

これは常に入力フィールドです。このフィールドの初期値は BMVER1 です。

BMOPT (10 桁の符号付き整数)

バッファからメッセージ・ハンドルへの変換の構造体 - Options フィールド。

値は次のいずれかです。

BMDLPR

メッセージ・ハンドルに追加されるプロパティが、バッファから削除される。呼び出しが失敗すると、プロパティは削除されません。

デフォルト・オプション: 説明されているオプションが必要でない場合は、以下のオプションを使用してください。

BMNONE

指定されるオプションはありません。

これは常に入力フィールドです。このフィールドの初期値は BMDLPR です。

初期値

表 722. MQBMHO のフィールド		
フィールド名	定数の名前	定数の値
BMSID	BMSIDV	'BMHO'
BMVER	BMVER1	1
BMOPT	BMNONE	0

RPG 宣言

```
D* MQBMHO Structure
D*
D*
D* Structure identifier
D  BMSID          1          4      INZ('BMHO')
D*
D* Structure version number
D  BMVER          5          8I 0  INZ(1)
D*
D* Options that control the action of MQBUFMH
D  BMOPT          9         12I 0  INZ(1)
```

IBM i IBM i での MQBO (開始オプション)

MQBO 構造体を使用すると、アプリケーションで、作業単位の作成に関するオプションを指定できます。

概要

目的: この構造体は、MQBEGIN 呼び出しの入出力パラメーターです。

文字セットとエンコード: MQBO 内のデータは、**CodedCharSetId** キュー・マネージャー属性で指定された文字セットと、ENNAT で指定されたローカル・キュー・マネージャーのエンコードで記述されていなければなりません。

- [1020 ページの『フィールド』](#)
- [1021 ページの『初期値』](#)
- [1021 ページの『RPG 宣言』](#)

フィールド

MQBO 構造体には、以下のフィールドが含まれます。フィールドはアルファベット順に説明されています。

BOOPT (10 桁の符号付き整数)

MQBEGIN のアクションを制御するオプション。

値は次のものでなければなりません。

BONONE

指定されるオプションはありません。

これは常に入力フィールドです。このフィールドの初期値は BONONE です。

BOSID (4 バイトの文字ストリング)

構造体 ID

値は次のものでなければなりません。

BOSIDV

開始オプション構造体の ID。

これは常に入力フィールドです。このフィールドの初期値は BOSIDV です。

BOVER (10 桁の符号付き整数)

構造体のバージョン番号。

値は次のものでなければなりません。

BOVER1

開始オプション構造体のバージョン番号。

以下の定数は、現行バージョンのバージョン番号を指定しています。

BOVERC

開始オプション構造体の現行バージョン。

これは常に入力フィールドです。このフィールドの初期値は BOVER1 です。

初期値

表 723. MQBO のフィールド		
フィールド名	定数の名前	定数の値
BOSID	BOSIDV	'B0--'
BOVER	BOVER1	1
BOOPT	BONONE	0

注：

1. 記号-は、単一のブランク文字を表します。

RPG 宣言

```
D*..1.....2.....3.....4.....5.....6.....7..
D* MQBO Structure
D*
D* Structure identifier
D BOSID          1          4    INZ('B0 ')
D* Structure version number
D BOVER          5          8I 0 INZ(1)
D* Options that control the action of MQBEGIN
D BOOPT          9          12I 0 INZ(0)
```

IBM i IBM i での MQCBC (コールバック・コンテキスト)

コールバック・ルーチンを記述する構造。

概要

目的

MQCBC 構造を使用して、コールバック関数に渡されるコンテキスト情報を指定します。

この構造は、メッセージ・コンシューマー・ルーチンに対する呼び出しの入出力パラメーターです。

バージョン

MQCBC の現行バージョンは CBCV2 です。

文字セットとエンコード

MQCBC 内のデータは、**CodedCharSetId** キュー・マネージャー属性で指定した文字セットと ENNAT で指定したローカル・キュー・マネージャーのエンコードで記述されています。しかし、アプリケーションが IBM MQ クライアントとして実行している場合、構造体はクライアントの文字セットおよびエンコードに従います。

- [1022 ページの『フィールド』](#)
- [1027 ページの『初期値』](#)
- [1028 ページの『RPG 宣言』](#)

フィールド

MQCBC 構造には、以下のフィールドが含まれます。フィールドはアルファベット順に説明されています。

CBCBUFFLEN (10 桁の符号付き整数)

このバッファーは、コンシューマーに関して定義された MaxMsgLength 値と、MQGMO 内の ReturnedLength 値の両方よりも大きくなる場合があります。

コールバック・コンテキスト構造 - BufferLength フィールド。

これは、この機能に渡されているメッセージ・バッファーの長さ (バイト数) です。

実際のメッセージ長は、[DataLength](#) フィールドで提供されます。

コールバック関数の所要時間中、アプリケーションは独自の目的でバッファー全体を使用できます。

これはメッセージ・コンシューマー関数の入力フィールドです。例外ハンドラー関数とは関係ありません。

CBCCALLBA (10 桁の符号付き整数)

コールバック・コンテキスト構造 - CallbackArea フィールド。

これは、コールバック関数ができるフィールドです。

キュー・マネージャーは、このフィールドの内容に基づいて決定せず、MQCBD 構造中の [CBDCALLBA](#) フィールドから、コールバック関数の定義に使用する MQCB 呼び出し上のパラメーターが変更せずに渡されます。

CBCCALLBA の変更内容は、CBCHOBJ に関するコールバック関数の複数の呼び出しにわたって保存されます。このフィールドは、他のハンドルに関するコールバック関数と共有されません。

これは、コールバック関数への入出力フィールドです。このフィールドの初期値は、ヌル・ポインターまたはヌル・バイトです。

CBCCALLT (10 桁の符号付き整数)

コールバック・コンテキスト構造 - CallType フィールド。

この機能が呼び出された理由に関する情報を格納するフィールド。以下の呼び出しタイプが定義されています。

メッセージ送達呼び出しタイプ: これらの呼び出しタイプには、メッセージに関する情報が含まれます。

CBCLen および **CBCBUFFLEN** パラメーターは、これらの呼び出しタイプに対して有効です。

CBCTMR

メッセージ・コンシューマー機能が呼び出され、メッセージがオブジェクト・ハンドルから破壊除去されました。

CBCCC の値が CCWARN である場合、*Reason* フィールドの値は RC2079 またはデータ変換問題を示すいずれかのコードになります。

CBCTMN

メッセージ・コンシューマー機能が呼び出され、メッセージはまだオブジェクト・ハンドルから破壊除去されていません。*MsgToken* を使用すると、メッセージをオブジェクト・ハンドルから破壊除去できます。

メッセージが除去されていない理由は以下のとおりです。

- MQGMO オプションがブラウズ操作 GMBR* を要求した。
- メッセージが使用可能なバッファより大きく、MQGMO オプションが *gmatm* を指定していない。

CBCCC の値が CCWARN である場合、*Reason* フィールドの値は RC2080 またはデータ変換問題を示すいずれかのコードになります。

コールバック制御呼び出しタイプ: これらの呼び出しタイプには、コールバックの制御に関する情報が含まれ、メッセージに関する詳細情報は含まれません。これらの呼び出しタイプは、MQCBD 構造体内の *CBDOPT* を使用して要求されます。

CBCCLEN および CBCBUFFLEN パラメーターは、これらの呼び出しタイプに対して無効です。

CBCTRC

この呼び出しタイプの目的は、コールバック関数がいくつかの初期セットアップを実行できるようにすることです。

コールバック関数は、コールバックが登録された直後に呼び出されます (つまり、CBREG の *Operation* フィールドの値を使用した MQCB 呼び出しからの戻り時)。

この呼び出しタイプは、メッセージ・コンシューマーとイベント・ハンドラーの両方に使用されます。

要求された場合、このタイプがコールバック関数の最初の呼び出しになります。

CBCREA フィールドの値は RCNONE です。

CBCTSC

この呼び出しタイプの目的は、コールバック関数が開始時にいくつかのセットアップを実行できるようにすることです。例えば、以前の停止時に終結処理されたリソースの再インストールなどが含まれます。

コールバック関数は、接続が CTLSR または CTLSW のいずれかを使用して開始されたときに呼び出されます。

コールバック関数が別のコールバック関数内に登録されている場合、この呼び出しタイプはコールバックが戻るときに呼び出されます。

この呼び出しタイプは、メッセージ・コンシューマー専用です。

CBCREA フィールドの値は RCNONE です。

CBCTTC

この呼び出しタイプの目的は、コールバック関数がしばらくの間停止している際にいくつかの終結処理を実行できるようにすることです。例えば、メッセージのコンシューム中に獲得した追加リソースの終結処理などが含まれます。

コールバック関数は、CTLSP の *Operation* フィールドの値を使用して MQCTL 呼び出しが発行されるときに呼び出されます。

この呼び出しタイプは、メッセージ・コンシューマー専用です。

CBCREA フィールドの値は、停止の理由を示すように設定されます。

CBCTDC

この呼び出しタイプの目的は、コールバック関数がコンシューム・プロセスの終わりに最終終結処理を実行できるようにすることです。このコールバック関数は、以下の時点で呼び出されます。

- コールバック関数は、BCUNR を指定した MQCB 呼び出しを使用して登録解除されます。
- キューがクローズされるために、暗黙的な登録解除が発生する時点。この場合、コールバック関数は HOUNUH にオブジェクト・ハンドルとして渡されます。
- MQDISC 呼び出しが完了する時点。暗黙的なクローズが発生し、そのために登録解除が発生します。この場合、接続は即時に切断されず、実行中のトランザクションはまだコミットされません。

これらのいずれかのアクションがコールバック関数自体の内部で取られる場合、コールバックが戻るとアクションが呼び出されます。

この呼び出しタイプは、メッセージ・コンシューマーとイベント・ハンドラーの両方に使用されます。

要求された場合、このタイプがコールバック関数の最後の呼び出しになります。

CBCREA フィールドの値は、停止の理由を示すように設定されます。

CBCTEC

イベント・ハンドラー機能

イベント・ハンドラー関数は、以下の場合にはメッセージなしで呼び出されています。

- MQCTL 呼び出しが CTLSP の *Operation* フィールドの値を使用して呼び出されている。または、
- キュー・マネージャーまたは接続が、停止または静止している。

この呼び出しを使用して、すべてのコールバック関数に対し適切な処置を行うことができます。

メッセージ・コンシューマー機能

メッセージ・コンシューマー関数は、オブジェクト・ハンドルに固有のエラー (CBCCC= CCFAIL) が検出されている場合に (例えば CBCREA コード = RC2016)、メッセージなしで呼び出されます。

CBCREA フィールドの値は、呼び出しの理由を示すように設定されます。

これは入力フィールドです。CBCTMR および CMCTMN は、メッセージ・コンシューマー関数だけに適用されます。

CBCCC (10 桁の符号付き整数)

コールバック・コンテキスト構造 - CompCode フィールド。

これは完了コードです。メッセージのコンシュームに関する問題があったかどうかを示します。以下のいずれかです。

CCOK

正常終了。

CCWARN

警告 (部分完了)

CCFAIL

呼び出し失敗

これは入力フィールドです。このフィールドの初期値は CCOK です。

CBCCONNAREA (10 桁の符号付き整数)

コールバック・コンテキスト構造 - ConnectionArea フィールド。

これは、コールバック関数ができるフィールドです。

キュー・マネージャーは、このフィールドの内容に基づいて決定を行いません。これは、コールバック関数の制御に使われる MQCTL 呼び出しのパラメーターである、MQCTLO 構造内の ConnectionArea フィールドから不変のまま渡されます。

コールバック関数がこのフィールドに対して加える変更は、コールバック関数を呼び出すたびに保存されます。この領域を使用して、すべてのコールバック関数で共有される情報を渡すことができます。この領域は、*CallbackArea* とは違って、接続ハンドルに関するすべてのコールバックで共通です。

これは入出力フィールドです。このフィールドの初期値は、ヌル・ポインターまたはヌル・バイトです。

CBCLEN (10 桁の符号付き整数)

これは、メッセージ内のアプリケーション・データの長さ (バイト数) です。この値がゼロの場合は、メッセージにアプリケーション・データがないことを意味します。

CBCLEN フィールドにはメッセージの長さが格納されますが、必ずしもコンシューマーに渡されたメッセージ・データの長さであるとは限りません。メッセージが切り捨てられている可能性もあります。コンシューマーに渡されたデータの量を判別するには、MQGMO で GMRL フィールドを使用します。

メッセージが切り捨てられたことが理由コードに示される場合は、CBCLEN フィールドを使用して、実際のメッセージの大きさを判別できます。これにより、メッセージ・データを収容するのに必要なバッファのサイズを判別してから、MQCB 呼び出しを発行し、適切な値を使って MQCBD の CBDMMML を更新することができます。

GMCONV オプションが指定されている場合、変換されたメッセージは *DataLength* の戻り値より大きくなる可能性があります。この場合、おそらく、アプリケーションで MQCB 呼び出しを発行し、MQCBD の CBDMMML をキュー・マネージャーから戻される *DataLength* の値よりも大きくなるように更新する必要があります。

メッセージ切り捨ての問題を防ぐには、*MaxMsgLength* を CBDFM と指定します。これにより、キュー・マネージャーはデータ変換後のメッセージ全長に対応するバッファを割り振ります。ただし、このオプションが指定された場合でも、要求を正しく処理するのに十分なストレージが使用できない可能性があることに注意してください。アプリケーションが、戻される理由コードを常に検査する必要があります。例えば、メッセージを変換するのに十分なストレージを割り振ることができない場合は、メッセージは変換されずにアプリケーションに戻されます。

これはメッセージ・コンシューマー関数に対する入力フィールドで、イベント・ハンドラー関数には関係ありません。

CBCFLG (10 桁の符号付き整数)

このコンシューマーに関する情報を含むフラグ。

以下のオプションが定義されます。

CBCFBE

このフラグは、COQSC オプションを使用した以前の MQCLOSE 呼び出しが理由コード RC2458 で失敗した場合に戻される可能性があります。

このコードは、最後の先読みメッセージが戻され、バッファが空になったことを示しています。アプリケーションが COQSC オプションを使用して別の MQCLOSE 呼び出しを発行すると、正常に実行されます。

このフラグが設定されたメッセージがアプリケーションに確実に与えられるとは限らないことに注意してください。これは、現在の選択基準と一致しないメッセージが依然として先読みバッファ内に入っている可能性があるためです。そのような場合、理由コード RC2019 でコンシューマー関数が呼び出されます。

先読みバッファが空の場合は、CBCFBE フラグおよび理由コード RC2518 でコンシューマーが呼び出されます。

これはメッセージ・コンシューマー関数に対する入力フィールドで、イベント・ハンドラー関数には関係ありません。

CBCHOBJ (10 桁の符号付き整数)

コールバック・コンテキスト構造 - CBCHOBJ フィールド。

メッセージ・コンシューマーに対する呼び出しの場合、これはメッセージ・コンシューマーに関連したオブジェクトのハンドルです。

イベント・ハンドラーの場合、この値は HONONE です。

メッセージがキューから除去されていない場合、アプリケーションはこのハンドルとメッセージ読み取りオプション・ブロック中のメッセージ・トークンを使用して、メッセージを読み取ることができます。

これは常に入力フィールドです。このフィールドの初期値は HOUNUH です。

CBRCRD (10 桁の符号付き整数)

CBRCRD は、再接続を試みるまでにキュー・マネージャーが待機する時間の長さを示します。このフィールドをイベント・ハンドラーによって変更し、再接続の遅延または停止を完全に変更することができます。

コールバック・コンテキストの **Reason** フィールドの値が RC2545 である場合にのみ、**CBRCRD** フィールドを使用してください。

イベント・ハンドラーに入る時点で、**CBRCRD** の値は、再接続を試行するまでにキュー・マネージャーが待機するミリ秒数です。イベント・ハンドラーからの戻り時にキュー・マネージャーの動作を変更するための設定可能な値が、[1026 ページの表 724](#) にリストされています。

表 724. CBRCRD の値	
値	説明
-1	それ以上再接続を試みません。アプリケーションにはエラーが戻されます。
0	すぐに再接続を試みます。
>0	接続を再試行するまで、このミリ秒数の間待ちます。

CBCREA (10 桁の符号付き整数)

コールバック・コンテキスト構造 - Reason フィールド。

これは、CBCCC を修飾する理由コードです。

これは入力フィールドです。このフィールドの初期値は RCNONE です。

CBCSTATE (10 桁の符号付き整数)

現在のコンシューマーの状態を示す標識。ゼロ以外の理由コードがコンシューマー関数に渡される場合、このフィールドはアプリケーションにとって非常に重要となります。

理由コードごとに動作をコーディングする必要がなくなるので、このフィールドを使用するとアプリケーション・プログラミングを単純化できます。

これは入力フィールドです。このフィールドの初期値は CSNONE です。

表 725. CBCSTATE の値とその結果のアクション		
State	キュー・マネージャーのアクション	定数の値
CSNONE この理由コードは、追加の理由情報のない通常の呼び出しを表します。	なし。これは通常の操作です。	0

表 725. CBCSTATE の値とその結果のアクション (続き)		
State	キュー・マネージャーのアクション	定数の値
CSSUST これらの理由コードは、一時的な条件を表します。	コールバック・ルーチンが呼び出されて条件が報告されてから、中断されます。特定の期間の後で、システムが操作を再試行することもあります。同じ条件が発生することになります。	1
CSSUSU これらの理由コードは、解決するためにコールバックがアクションを取る必要がある条件を表します。	コンシューマーが中断され、コールバック・ルーチンが呼び出されて条件が報告されます。コールバック・ルーチンが、条件を解決し (可能な場合)、接続の RESUME または閉鎖を行う必要があります。	2
CSSUS これらの理由コードは、以後メッセージのコールバックが行えなくなる障害を表します。	キュー・マネージャーはコールバック関数を自動的に中断します。コールバック関数を再開すると、おそらく同じ理由コードを再び受け取ります。	3
CSSTOP これらの理由コードは、メッセージのコンシュームの終わりを表します。	例外ハンドラーおよび CBDC 指定したコールバックに配布されます。これ以上のメッセージはコンシュームできません。	4

CBCSID (10 桁の符号付き整数)

コールバック・コンテキスト構造 - StrucId フィールド。

これは構造体 ID です。値は以下のものでなければなりません。

CBCSI

コールバック・コンテキスト構造の ID。

これは常に入力フィールドです。このフィールドの初期値は CBCSI です。

CBCVER (10 桁の符号付き整数)

コールバック・コンテキスト構造 - Version フィールド。

これは構造体のバージョン番号です。値は以下のものでなければなりません。

CBCV1

バージョン 1 のコールバック・コンテキストの構造。

以下の定数は、現行バージョンのバージョン番号を指定しています。

CBCCV

コールバック・コンテキストの構造の現行バージョン。

これは常に入力フィールドです。このフィールドの初期値は CBCV1 です。

初期値

表 726. MQCBC のフィールド		
フィールド名	定数の名前	定数の値
CBCSID	CBCSI	'CBC'
CBCVER	CBCV1	1
CBCCALLT	なし	0
CBCHOBJ	HOUNUH	-1

表 726. MQCBC のフィールド (続き)		
フィールド名	定数の名前	定数の値
CBCCALLBA	なし	ヌル・ポインターまたはヌル・バイト
CBCCONNAREA	なし	ヌル・ポインターまたはヌル・バイト
CBCCC	CCOK	0
CBCREA	RCNONE	0
CBCSTATE	CSNONE	0
CBCLEN	なし	0
CBCBUFFLEN	なし	0
CBCFLG	なし	0
CBCRCD	なし	0

注:

1. 記号-は、単一のブランク文字を表します。

RPG 宣言

```

D* MQCBC Structure
D*
D*
D* Structure identifier
D  CBCSID              1          4      INZ('CBC ')
D*
D* Structure version number
D  CBCVER              5          8I 0 INZ(1)
D*
D* Why Function was called
D  CBCCALLT           9          12I 0 INZ(0)
D*
D* Object Handle
D  CBCHOBJ            13          16I 0 INZ(-1)
D*
D* Callback data passed to the function
D  CBCCALLBA          17          32*   INZ(*NULL)
D*
D* MQCTL Data area passed to the function
D  CBCCONNAREA        33          48*   INZ(*NULL)
D*
D* Completion Code
D  CBCCC              49          52I 0 INZ(0)
D*
D* Reason Code
D  CBCREA             53          56I 0 INZ(0)
D*
D* Consumer State
D  CBCSTATE           57          60I 0 INZ(0)
D*
D* Message Data Length
D  CBCLEN             61          64I 0 INZ(0)
D*
D* Buffer Length
D  CBCBUFFLEN         65          68I 0 INZ(0)
D*
D* Flags containing information about
D* this consumer
D  CBCFLG             69          72I 0 INZ(0)
D* Ver:1 **
D* Number of milliseconds before reconnect attempt
D  CBCRCD             73          76I 0 INZ(0)
D* Ver:2 **
D*

```


コールバック関数を指定する構造。

概要

目的: MQCBD 構造体を使用して、コールバック関数と、キュー・マネージャーによるこの関数の使用法を制御するオプションを指定します。

この構造体は、MQCB 呼び出しの入力パラメーターです。

バージョン: MQCBD の現行バージョンは CBDV1 です。

文字セットおよびエンコード: MQCBD のデータは、ローカル・キュー・マネージャーの文字セットおよびエンコードになっていなければなりません。これは、**CodedCharSetId** キュー・マネージャー属性および ENNAT で指定されます。ただし、アプリケーションを IBM MQ MQI client として実行する場合は、構造体はクライアントの文字セットとエンコードに従っている必要があります。

- [1029 ページの『フィールド』](#)
- [1033 ページの『初期値』](#)
- [1033 ページの『RPG 宣言』](#)

フィールド

MQCBD 構造体には、以下のフィールドが含まれます。フィールドは**アルファベット順**に説明されています。

CBDCALLBA (10 桁の符号付き整数)

これは、コールバック関数ができるフィールドです。

キュー・マネージャーは、このフィールドの内容に基づいて何かを決定することではなく、この値はコールバック関数宣言のパラメーターとして、MQCBD 構造体の **CBCCALLBA** フィールドから未変更のまま渡されます。

この値は、値 CBREG を持つ、現在定義されたコールバックがない *Operation* だけで使用され、以前の定義を置き換えることはありません。

これは、コールバック関数への入出力フィールドです。このフィールドの初期値は、ヌル・ポインターまたはヌル・バイトです。

CBDCALLBF (10 桁の符号付き整数)

このコールバック関数は、関数呼び出しとして呼び出されます。

このフィールドを使用して、コールバック関数へのポインターを指定します。

CallbackFunction または *CallbackName* のどちらかを指定しなければなりません。両方を指定すると、理由コード RC2486 が返されます。

CallbackName または *CallbackFunction* のどちらも設定されない場合、呼び出しは失敗し、理由コード RC2486 が返されます。

このオプションは、以下の環境ではサポートされません。

- CICS on z/OS
- 関数ポインター参照をサポートしていないプログラミング言語およびコンパイラ

このような場合、呼び出しは失敗し、理由コード RC2486 が返されます。

これは入力フィールドです。このフィールドの初期値は、ヌル・ポインターまたはヌル・バイトです。

CBDCALLBN (10 桁の符号付き整数)

このコールバック関数は、動的リンク・プログラムとして呼び出されます。

CallbackFunction または *CallbackName* のどちらかを指定しなければなりません。両方を指定すると、理由コード RC2486 が返されます。

CallbackName または *CallbackFunction* のいずれかが真でない場合、呼び出しは失敗し、理由コード RC2486 が返されます。

モジュールは、最初に使用するコールバック・ルーチンが登録される際にロードされ、最後に使用するコールバック・ルーチンが登録解除される際にアンロードされます。

続くテキストで注記されている場合を除き、名前はフィールド中で左寄せされ、埋め込みブランクはありません。名前自体はフィールドの長さまでブランクが埋め込まれます。下記の説明で、大括弧 ([]) はオプションの情報を示します。

IBMi

コールバック名は、次のいずれの形式にすることができます。

- Library "/" Program
- Library "/" ServiceProgram "("FunctionName")"

例えば、MyLibrary/MyProgram(MyFunction) などです。

ライブラリー名を *LIBL にすることができます。ライブラリー名とプログラム名は両方とも、最大 10 文字に制限されます。

AIX and Linux

コールバック名は、動的ロード可能なモジュールまたはライブラリーの名前であり、そのライブラリーにある関数の名前が接尾部となります。関数名は括弧で囲む必要があります。ライブラリー名の前にはオプションでディレクトリー・パスを付けることができます。

```
[path]library(function)
```

パスを指定しないと、システム検索パスが使用されます。

この名前は最大 128 文字までに制限されています。

Windows

コールバック名は、ダイナミック・リンク・ライブラリーの名前で、このライブラリー中にある関数の名前が接尾部になります。関数名は括弧で囲む必要があります。ライブラリー名の前にはオプションでディレクトリー・パスとドライブを付けることができます。

```
[d:][path]library(function)
```

ドライブとパスを指定しないと、システム検索パスが使用されます。

この名前は最大 128 文字までに制限されています。

z/OS

コールバック名は、LINK または LOAD マクロの EP パラメーター上の仕様によって有効なロード・モジュールの名前です。

この名前は最大 8 文字までに制限されています。

z/OS CICS

コールバック名は、EXEC CICS LINK コマンド・マクロの PROGRAM パラメーター上の仕様によって有効なロード・モジュールの名前です。

この名前は最大 8 文字までに制限されています。

プログラムは、インストール済みの PROGRAM 定義の REMOTESYTEM オプションを使用してリモートとして定義するか、または動的ルーティング・プログラムによって定義することができます。

プログラムが API 呼び出し IBM MQ を使用するには、遠隔 CICS 地域が IBM MQ 接続されている必要があります。しかし、MQCBC 構造中の CBCHOBJ フィールドはリモート・システム中では無効であることに注意してください。

CallbackName をロードしようとして障害が発生すると、以下のいずれかのエラー・コードがアプリケーションに戻されます。

- RC2495
- RC2496
- RC2497

また、メッセージがエラー・ログに書き込まれ、ロードが試行されたモジュールの名前と、オペレーティング・システムからの失敗理由コードが含まれます。

これは入力フィールドです。このフィールドの初期値は、ヌル・ストリングまたはブランクです。

CBDCALLBT (10 桁の符号付き整数)

これは、コールバック関数のタイプです。値は次のいずれかです。

CBTMC

このコールバックをメッセージ・コンシューマー関数として定義します。

メッセージ・コンシューマー・コールバック関数は、指定された選択基準と一致するメッセージがオブジェクト・ハンドル上で使用可能であり、接続が開始されている場合に呼び出されます。

CBTEH

このコールバックを非同期イベント・ルーチンとして定義します。これはハンドルのメッセージをコンSUMするするためには使用されません。

Hobj は、イベント・ハンドラーを定義する MQCB 呼び出しでは必要ないので、指定すると無視されます。

イベント・ハンドラーは、メッセージ・コンシューマー環境全体に影響が及ぶ場合に呼び出されます。コンシューマー関数は、例えばキュー・マネージャーまたは接続が停止中または静止中であるといったイベントが発生する場合に、メッセージなしで呼び出されます。これは、単一のメッセージ・コンシューマー固有の条件 (例えば RC2016) では呼び出されません。

接続が開始されているか停止しているかにかかわらず、イベントはアプリケーションに配布されますが、以下の環境は例外です。

- CICS 環境 z/OS について
- 非スレッド・アプリケーション

呼び出し元がこれらの値の 1 つを渡さない場合、呼び出しは失敗し、理由コード RC2483 が返されます。

これは常に入力フィールドです。このフィールドの初期値は CBTMC です。

CBDMML (10 桁の符号付き整数)

これは、ハンドルから読み取り、コールバック・ルーチンに渡すことができる、最長メッセージの長さ (バイト数) です。メッセージの長さがこれより長い場合は、コールバック・ルーチンは *MaxMsgLength* バイトのメッセージと以下の理由コードを受け取ります。

- RC2080、または
- RC2079 (GMATM を指定した場合)

実際のメッセージ長は、MQCBC 構造体の [1025 ページの『CBCLEN \(10 桁の符号付き整数\)』](#) フィールドで提供されます。

以下のような特殊値が定義されます。

CBDFM

バッファ長はシステムにより、切り捨てなしでメッセージを返すように調整されます。

メッセージを受け取るバッファを割り振るために使用できる十分なメモリがない場合、システムはコールバック関数を呼び出し、理由コード RC2071 を出します。

例えば、データ変換を要求し、メッセージ・データを変換するために使用できる十分なメモリがない場合、未変換のメッセージはコールバック関数に渡されます。

これは入力フィールドです。 *MaxMsgLength* フィールドの初期値は CBDFM です。

CBDOPT (10 桁の符号付き整数)

コールバック記述子構造 - Options フィールド。

以下のいずれかまたはすべてを指定することができます。複数のオプションを指定するには、値を加算する (同じ定数を複数回加算しない) か、ビット単位 OR 演算を使用して値を結合 (プログラミング言語でビット演算がサポートされる場合) します。有効でない組み合わせについては、注記されています。それ以外の組み合わせは有効です。

CBDFQ

MQCB 呼び出しは、キュー・マネージャーが静止状態にあるときは失敗します。

z/OS では、接続 (CICS または IMS アプリケーション用の) が静止状態になっている場合、このオプションは MQCB 呼び出しを強制的に失敗させる役割もします。

メッセージ・コンシューマーが静止する場合に、メッセージ・コンシューマーに通知されるようにするには、MQCB 呼び出しに渡される MQGMO オプションに GMFIQ を指定します。

制御オプション: 以下のオプションでは、コンシューマーの状態の変更時に、メッセージなしで、コールバック関数が呼び出されるかどうかを制御します。

CBDRRC

コールバック関数は、呼び出しタイプ CBCTRC で呼び出されます。

CBDSRC

コールバック関数は、呼び出しタイプ CBCTSC で呼び出されます。

CBDTTC

コールバック関数は、呼び出しタイプ CBCTTC で呼び出されます。

CBDDC

コールバック関数は、呼び出しタイプ CBCTDC で呼び出されます。

これらの呼び出しタイプに関する詳細については、[1022 ページの『CBCCALLT \(10 桁の符号付き整数\)』](#)を参照してください。

デフォルト・オプション: 上記のいずれのオプションも必要でない場合は、以下のオプションを使用します。

CBDNO

この値は、他のオプションが指定されなかったことを示すために使用します。すべてのオプションはデフォルト値であるとみなされます。

CBDNO は、プログラムの文書化をサポートするために定義します。したがって、このオプションは、他のオプションと同時に使用するものではありません。しかしこのオプションの値はゼロのため、他のオプションと同時に使用してもそれを検出できません。

これは入力フィールドです。Options フィールドの初期値は、CBDNO です。

CBDSID (10 桁の符号付き整数)

コールバック記述子構造 - StrucId フィールド。

これは構造体 ID です。値は以下のものでなければなりません。

CBDSI

コールバック記述子構造の ID。

これは常に入力フィールドです。このフィールドの初期値は CBDSI です。

CBDVER (10 桁の符号付き整数)

コールバック記述子構造 - Version フィールド。

これは構造体のバージョン番号です。値は以下のものでなければなりません。

CBDV1

バージョン 1 のコールバック記述子の構造。

以下の定数は、現行バージョンのバージョン番号を指定しています。

CBDV

コールバック記述子の構造の現行バージョン。

これは常に入力フィールドです。このフィールドの初期値は CBDV1 です。

初期値

表 727. MQCBD のフィールド		
フィールド名	定数の名前	定数の値
<i>StrucId</i>	CBD SI	'CBD-'
<i>Version</i>	CBDV1	1
<i>CallbackType</i>	CBTMC	1
<i>Options</i>	CBDNO	0
<i>CallbackArea</i>	なし	ヌル・バイト
<i>CallbackFunction</i>	なし	ヌル・バイト
<i>CallbackName</i>	なし	ブランク
<i>MaxMsgLength</i>	CBD FM	-1

注:

1. 記号-は、単一のブランク文字を表します。

RPG 宣言

```
D* MQCBD Structure
D*
D*
D* Structure identifier
D  CBDSID             1         4      INZ('CBD ')
D*
D* Structure version number
D  CBDVER             5         8I 0 INZ(1)
D*
D* Callback function type
D  CBDCALLBT         9        12I 0 INZ(1)
D*
** Options controlling message
D* consumption
D  CBDOPT            13        16I 0 INZ(0)
D*
D* User data passed to the function
D  CBDCALLBA        17        32*
D*
D* FP: Callback function pointer
D  CBDCALLBF        33        48*
D*
D* Callback name
D  CBDCALLBN        49        176     INZ('\0')
D*
D* Maximum message length
D  CBDMML           177       180I 0 INZ(-1)
```

IBM i IBM i での MQCHARV (可変長ストリング)

MQCHARV 構造体は、可変長ストリングを記述するのに使用します。

概要

文字セットとエンコード: MQCHARV 内のデータは、ENNAT で指定されるローカル・キュー・マネージャーのエンコードと、その構造体の中の VCHRC フィールドの文字セットで記述されていなければなりません。

アプリケーションが IBM MQ MQI client として実行される場合、構造体はクライアントのエンコードに従っている必要があります。エンコードによって表記が変わる文字セットもあります。VCHRC がそうした文字セットである場合、使用されるエンコードは MQCHARV 内の他のフィールドと同じエンコードです。VSCCSID で識別される文字セットは、2 バイト文字セット (DBCS) も可能です。

使用法: MQCHARV 構造体は、それを含む構造体とは連続していない可能性があるデータをアドレス指定します。このデータをアドレッシングするには、ポインター・データ・タイプで宣言されるフィールドを使用できます。

- [1034 ページの『フィールド』](#)
- [1035 ページの『初期値』](#)
- [1035 ページの『RPG 宣言』](#)
- [1035 ページの『CSAPL の再定義』](#)

フィールド

MQCHARV 構造体には、以下のフィールドが含まれます。フィールドは**アルファベット順**に説明されています。

VCHRC (10 桁の符号付き整数)

これは、VCHRP または VCHRO フィールドによりアドレス指定される可変長ストリングの文字セット ID です。

このフィールドの初期値は CSAPL です。これは IBM MQ により定義され、これをキュー・マネージャー側でキュー・マネージャーの本当の文字セット ID に変更する必要があることを示します。この方法は、CSQM の動作と同じです。したがって、値 CSAPL が可変長ストリングに関連付けられることはありません。このフィールドの初期値は、ご使用のアプリケーションのプログラミング言語に適した方法でコンパイル単位の定数 CSAPL に別の値を定義することにより変更できます。

VCHRL (10 桁の符号付き整数)

VCHRP または VCHRO フィールドによりアドレス指定される可変長ストリングの長さ (バイト単位)。

このフィールドの初期値は 0 です。この値は、ゼロ以上、または認識される以下の特殊値のいずれかでなければなりません。

VSNLT

VSNLT が指定されていない場合、VCHRL のバイトがストリングの一部に含まれます。ヌル文字があってもストリングは区切られません。

VSNLT が指定されている場合、ストリングは、その中にある最初のヌルで区切られます。ヌル自体はそのストリングの一部として組み込まれません。

注: VSNLT が指定されている場合にストリングの終端となるヌル文字は、VCHRC により指定されたコード・セットのヌルです。

例えば、UTF-16 (CCSID 1200、13488、および 17584) では 2 バイトの Unicode エンコードで、ヌルはすべてゼロの 16 ビットの数値で表されます。UTF-16 では、すべてゼロに設定された 1 バイトが文字の一部になっていることは一般的ですが (例えば、7 ビットの ASCII 文字)、偶数バイト境界に 2 つの「ゼロ」バイトがある場合のみストリングはヌル終了になります。奇数境界に 2 つの「ゼロ」バイトがある場合は、それらのバイトが有効な文字の個々の部分である場合に読み取れます。例えば、x'01' x'00 x'00' x'30' は 2 つの有効な Unicode 文字なので、ストリングはヌル終了しません。

VCHRO (10 桁の符号付き整数)

MQCHARV または MQCHARV を含む構造体の先頭からの可変長ストリングのオフセットをバイト数で表したものです。

MQCHARV 構造体が別の構造体に組み込まれている場合、この値は、この MQCHARV 構造体が含まれる構造体の先頭からの可変長ストリングのオフセット (バイト単位) です。MQCHARV 構造体が別の構造体に組み込まれていない場合、例えば、これが関数呼び出しにおけるパラメーターとして指定された場合、オフセットは MQCHARV 構造体の先頭からの相対位置です。

オフセットの値は、正負どちらの値にもなります。VCHRP または VCHRO フィールドのいずれか一方 (両方は不可) を使用して、可変長ストリングを指定できます。

このフィールドの初期値は 0 です。

VCHRP (ポインター)

これは可変長ストリングを指すポインターです。

VCHRP または VCHRO フィールドのいずれか一方 (両方は不可) を使用して、可変長ストリングを指定できます。

このフィールドの初期値は、ヌル・ポインターまたはヌル・バイトです。

VCHRS (10 桁の符号付き整数)

VCHRP または VCHRO フィールドによりアドレス指定されるバッファのサイズ (バイト単位)。

MQCHARV 構造体関数呼び出しの出力フィールドとして使用される場合、このフィールドは、提供されているバッファの長さで初期化される必要があります。VCHRL の値が VCHRS より大きい場合、データの VCHRS バイトのみがバッファ内の呼び出し元に戻されます。

この値は、0 以上であるか、または認識される以下の特殊値でなければなりません。

VSUSL

VSUSL が指定された場合、バッファの長さは MQCHARV 構造体内の VCHRL フィールドから取られます。構造体出力フィールドとして使用されており、バッファが提供されている場合には、この特殊値は適切ではありません。これはこのフィールドの初期値です。

初期値

表 728. MQCHARV の定数の初期値		
フィールド名	定数の名前	定数の値
VCHRP	なし	ヌル・ポインターまたはヌル・バイト。
VCHRO	なし	0
VCHRS	VSUSL	-1
VCHRL	なし	0
VCHRC	CSAPL	-3

RPG 宣言

```
D*..1.....2.....3.....4.....5.....6.....7..
D* MQCHARV Structure
D*
D* Address of variable length string
D VCHRP          1      16*
D* Offset of variable length string
D VCHRO          17      20I 0
D* Size of buffer
D VCHRS          21      24I 0
D* Length of variable length string
D VCHRL          25      28I 0
D* CCSID of variable length string
D VCHRC          29      32I 0
```

CSAPL の再定義

他のプラットフォームでサポートされるプログラミング言語とは異なり、RPG には定義済み定数を再定義する方法がないため、CSAPL 以外の値を使用する場合には、それぞれの VCHRC を明確に設定する必要があります。

IBM i MQCIH (CICS bridge ヘッダー) オン IBM i

MQCIH 構造は、メッセージの開始時に存在し得る情報を記述する。メッセージは、IBM MQ for z/OSMQCIH CICS bridge を介して送信される。

概要

形式名: FMCICS。

バージョン: MQCIH の現行バージョンは CIVER2 です。これより新しいバージョンの構造体にのみ存在するフィールドは、続く説明にその旨記載されています。

提供される COPY ファイルには最新バージョンの MQCIH が含まれており、CIVER フィールドの初期値は CIVER2 に設定されています。

文字セットおよびエンコード方式: MQCIH 構造体およびアプリケーション・メッセージ・データに使用する文字セットとエンコードに関しては、以下の特別条件に従うものとします。

- CICS bridge・キューを所有するキュー・マネージャーに接続するアプリケーションは、そのキュー・マネージャーの文字セットとエンコードで記述した MQCIH 構造体を渡す必要があります。この場合には MQCIH 構造体のデータ変換が実行されないためです。
- 他のキュー・マネージャーに接続するアプリケーションは、サポートされている任意の文字セットとエンコードで記述した MQCIH 構造体を渡すことができます。CICS bridge・キューを所有するキュー・マネージャーに接続された受信側のメッセージ・チャンネル・エージェントが、MQCIH 構造体を変換するからです。

注: この条件には例外が 1 つあります。CICS bridge・キューを所有するキュー・マネージャーが CICS を分散キューイングに使用している場合、MQCIH 構造体は、CICS bridge・キューを所有するキュー・マネージャーの文字セットとエンコードで記述されていなければなりません。

- MQCIH 構造体の後に続くアプリケーション・メッセージ・データは、MQCIH 構造体と同じ文字セットとエンコードで記述されていなければなりません。MQCIH 構造体の CICS フィールドおよび CIENC フィールドを使用して、そのアプリケーション・メッセージ・データの文字セットとエンコードを指定することはできません。

データがキュー・マネージャーのサポートする組み込み形式ではない場合、アプリケーション・メッセージ・データを変換するには、ユーザーがデータ変換出口を提供する必要があります。

使用法: アプリケーションで必要な値が 1046 ページの表 730 に示されている初期値と同じであり、ブリッジが AUTH=LOCAL または AUTH=IDENTIFY の設定で動作している場合は、MQCIH 構造体をメッセージから省略することができます。それ以外の場合、この構造体は必要です。

ブリッジは、バージョン 1 またはバージョン 2 の MQCIH 構造体を受け入れますが、3270 トランザクションではバージョン 2 を使用する必要があります。

アプリケーションは、"request" フィールドとして文書化されたフィールドが、ブリッジに送信されるメッセージに適切な値を持っていることを確認する必要があります。これらのフィールドはブリッジに入力されます。

"レスポンス" フィールドとして文書化されたフィールドは、ブリッジがアプリケーションに送信する応答メッセージ内の CICS bridge によって設定されます。CIRET、CIFNC、CICC、CIREA、および CIAC などのフィールド内には、エラー情報が戻されます。1036 ページの表 729 は、各種の CIRET の値に対して設定されるフィールドを示しています。

表 729. MQCIH 構造体のエラー情報フィールドの内容				
CIRET	CIFNC	CICC	CIREA	CIAC
CRC000	-	-	-	-
CRC003	-	-	FBC*	-
CRC002 CRC008	IBM MQ 呼び出し名	IBM MQ CMPCOD	IBM MQ REASON	-

表 729. MQCIH 構造体のエラー情報フィールドの内容 (続き)

CIRET	CIFNC	CICC	CIREA	CIAC
CRC001 CRC006 CRC007 CRC009	CICS EIBFN	CICS EIBRESP	CICS EIBRESP2	-
CRC004 CRC005	-	-	-	CICS ABCODE

- [1037 ページの『フィールド』](#)
- [1046 ページの『初期値』](#)
- [1047 ページの『RPG 宣言』](#)

フィールド

MQCIH 構造体には、以下のフィールドが含まれます。フィールドはアルファベット順に説明されています。

CIAC (4 バイトの文字ストリング)

異常終了コード。

このフィールドで戻される値が意味を持つのは、**CIRET** フィールドに値 CRC005 または CRC004 がある場合のみです。その場合、**CIAC** には CICS ABCODE 値が含まれます。

これは応答フィールドです。このフィールドの長さは LNABNC によって指定されます。このフィールドの初期値は 4 個のブランク文字です。

これは、SEND および RECEIVE BMS 要求で ADS 記述子を送信するかどうかを指定する標識です。以下の値が定義されます。

ADNONE

ADS 記述子は送受信しないでください。

ADSEND

ADS 記述子の送信。

ADRECV

ADS 記述子の受信。

ADMSGF

ADS 記述子のメッセージ・フォーマットを使用します。

これにより、ADS 記述子は、long 形式の ADS 記述子で送受信されます。長形式では、各フィールドは 4 バイトの境界で位置合わせされます。

CIADS フィールドは以下のように設定します。

- ADS 記述子を使用していない場合、このフィールドは ADNONE に設定します。
- ADS 記述子を使用しており、各環境の CCSID が同じである場合、このフィールドは ADSEND と ADRECV の合計に設定します。
- ADS 記述子を使用しているが、各環境の CCSID が異なる場合、このフィールドは ADSEND、ADRECV、ADMSGF の合計に設定します。

これは、3270 トランザクションにのみ使用される要求フィールドです。このフィールドの初期値は ADNONE です。

CIADS (10 桁の符号付き整数)

ADS 記述子の送受信を行います。

これは、SEND および RECEIVE BMS 要求で ADS 記述子を送信するかどうかを指定する標識です。以下の値が定義されます。

ADNONE

ADS 記述子は送受信しないでください。

ADSEND

ADS 記述子の送信。

ADRECV

ADS 記述子の受信。

ADMSGF

ADS 記述子のメッセージ・フォーマットを使用します。

これにより、ADS 記述子は、long 形式の ADS 記述子で送受信されます。長形式では、各フィールドは 4 バイトの境界で位置合わせされます。

CIADS フィールドは以下のように設定します。

- ADS 記述子を使用していない場合、このフィールドは ADNONE に設定します。
- ADS 記述子を使用しており、各環境の CCSID が同じである場合、このフィールドは ADSEND と ADRECV の合計に設定します。
- ADS 記述子を使用しているが、各環境の CCSID が異なる場合、このフィールドは ADSEND、ADRECV、ADMSGF の合計に設定します。

これは、3270 トランザクションにのみ使用される要求フィールドです。このフィールドの初期値は ADNONE です。

CIAI (4 バイトの文字ストリング)

AID キー。

これは、トランザクション開始時の AID キーの初期値です。値は 1 バイトで、左寄せされます。

これは、3270 トランザクションにのみ使用される要求フィールドです。このフィールドの長さは LNATID によって指定されます。初期値は 4 桁のブランクです。

CIAUT (8 バイトの文字ストリング)

パスワードまたはパスチケット。

パスワードまたはパスチケットです。ユーザー ID の認証が CICS bridge でアクティブな場合は、メッセージの送信側を認証するために、CIAUT を使用して MQMD の ID コンテキスト内のユーザー ID を指定します。

これは要求フィールドです。フィールドの長さは LNAUTH で指定されます。このフィールドの初期値は 8 ブランクです。

CICC (10 桁の符号付き整数)

IBM MQ 完了コードまたは CICS EIBRESP。

このフィールドに戻される値は、CIRET の値によって決まります。[1036 ページの表 729](#) を参照してください。

これは応答フィールドです。このフィールドの初期値は CCOK です。

CICNC (4 バイトの文字ストリング)

トランザクション異常終了コード。

これは、トランザクション (通常は、さらにデータを要求する会話型トランザクション) を終了するために使用される異常終了コードです。それ以外の場合は、このフィールドはブランクに設定されます。

これは、3270 トランザクションにのみ使用される要求フィールドです。このフィールドの長さは LNCNCL によって指定されます。初期値は 4 桁のブランクです。

CICP (10 桁の符号付き整数)

カーソル位置。

これは、トランザクション開始時の初期カーソル位置です。会話型トランザクションの場合、その後のカーソル位置は RECEIVE ベクトルで示されます。

これは、3270 トランザクションにのみ使用される要求フィールドです。このフィールドの初期値は 0 です。CIVER が CIVER2 よりも小さい場合、このフィールドは存在しません。

CICSI (10 桁の符号付き整数)

予約済み。

これは、予約フィールドです。したがって、値に意味はありません。このフィールドの初期値は 0 です。

CICT (10 桁の符号付き整数)

タスクが会話型かどうか。

これは、タスクがさらに情報を要求できるか、異常終了するかを指定する標識です。値は次のいずれかでなければなりません。

CTYES

タスクは会話型。

CTNO

タスクは非会話型。

これは、3270 トランザクションにのみ使用される要求フィールドです。フィールドの初期値は CTNO です。

CIENC (10 桁の符号付き整数)

予約済み。

これは、予約フィールドです。したがって、値に意味はありません。このフィールドの初期値は 0 です。

CIEO (10 桁の符号付き整数)

メッセージ内のエラーのオフセット。

これは、ブリッジ出口で検出された無効なデータの位置です。このフィールドは、メッセージの先頭から無効なデータの位置までのオフセットを示します。

これは、3270 トランザクションにのみ使用される応答フィールドです。このフィールドの初期値は 0 です。CIVER が CIVER2 よりも小さい場合、このフィールドは存在しません。

CIFAC (8 バイトのビット・ストリング)

ブリッジ機能トークン。

これは、8 バイトのブリッジ機能トークンです。ブリッジ機能トークンの目的は、疑似会話内の複数のトランザクションが同じブリッジ機能 (仮想 3270 端末) を使用できるようにすることです。疑似会話の最初のメッセージ (メッセージが 1 つのみの場合はそのメッセージ) では、値 FCNONE に設定します。これにより CICS は、そのメッセージに新しいブリッジ機能を割り振ります。入力メッセージでゼロ以外の CIFKT が指定されていると、応答メッセージでブリッジ機能トークンが戻されます。その後の入力メッセージでは、同じブリッジ機能トークンを使用できます。

以下のような特殊値が定義されます。

FCNONE

BVT トークンは指定されていません。

これは、3270 トランザクションにのみ使用される要求フィールドおよび応答フィールドです。このフィールドの長さは LNFAC によって指定されます。フィールドの初期値は FCNONE です。

CIFKT (10 桁の符号付き整数)

ブリッジ機能の解放時間。

これは、ユーザー・トランザクションが終了した後、ブリッジ機能が保持される長さ (秒数) です。非会話型トランザクションの場合、値はゼロになります。

これは、3270 トランザクションにのみ使用される要求フィールドです。このフィールドの初期値は 0 です。

CIFL (4 バイトの文字ストリング)

端末エミュレート属性。

これは、ブリッジ機能のモデルとして使用する設置済み端末の名前です。値としてブランクを指定すると、*CIFL* はブリッジ・トランザクション・プロファイル定義からとられるか、デフォルトの値が使用されます。

これは、3270 トランザクションにのみ使用される要求フィールドです。このフィールドの長さは LNFACL によって指定されます。初期値は 4 桁のブランクです。

CIFLG (10 桁の符号付き整数)

フラグ。

値は次のものでなければなりません。

CIFNON

フラグなし。

これは要求フィールドです。フィールドの初期値は CIFNON です。

CIFMT (8 バイトの文字ストリング)

MQCIH に続くデータの IBM MQ 形式名。

ここでは、MQCIH 構造体の後に続くデータの IBM MQ 形式名を指定します。

MQPUT または MQPUT1 呼び出しでは、アプリケーションは、このフィールドをデータに適切な値に設定する必要があります。このフィールドのコーディングの規則は、MQMD の *MDFMT* フィールドの場合と同じです。

CIRFM フィールドの値が FMNONE である場合は、この形式名が応答メッセージでも使用されます。

- DPL 要求の場合、*CIFMT* は COMMAREA の形式名です。
- 3270 要求の場合、*CIFMT* は CSQCBDCI、*CIRFM* は CSQCBDCO である必要があります。

これらの形式に対するデータ変換出口は、それを実行するキュー・マネージャーにインストールする必要があります。

要求メッセージによってエラー応答メッセージが生成された場合、エラー応答メッセージの形式名は FMSTR となります。

これは要求フィールドです。このフィールドの長さは LNFMT によって指定されます。このフィールドの初期値は FMNONE です。

CIFNC (4 バイトの文字ストリング)

IBM MQ 呼び出し名または CICS EIBFN 関数。

このフィールドに戻される値は、*CIRET* の値によって決まります。[1036 ページの表 729](#) を参照してください。*CIFNC* に IBM MQ 呼び出し名が格納されている場合は、次のような値が戻されます。

CFCONN

MQCONN 呼び出し。

CFGET

MQGET 呼び出し。

CFINQ

MQINQ 呼び出し。

CFOPEN

MQOPEN 呼び出し。

CFPUT

MQPUT 呼び出し。

CFPUT1

MQPUT1 呼び出し。

CFNONE

呼び出しなし。

これは応答フィールドです。このフィールドの長さはLNFUNCによって指定されます。このフィールドの初期値はCFNONEです。

CIGWI (10 桁の符号付き整数)

ブリッジ・タスクによって発行されたMQGET呼び出しの待機間隔。

このフィールドは、*CIUOW* の値が *CUFRST* である場合にのみ適用されます。送信側アプリケーションでは、このフィールドを使用して、ブリッジで発行されたMQGET呼び出しがこのメッセージによって開始された作業単位に関する2番目およびそれ以降の要求メッセージを待機するおよその時間をミリ秒単位で指定できます。ブリッジで使用されているデフォルトの待機間隔は、このフィールドによって指定変更されます。次の特別な値を使用できます。

WIDFLT

デフォルト待機間隔。

ブリッジの開始時に指定された期間だけCICS bridgeが待機します。

WIULIM

無制限の待機間隔。

これは要求フィールドです。このフィールドの初期値はWIDFLTです。

CIHI (10 桁の符号付き整数)

予約済み。

これは予約フィールドです。値はゼロでなければなりません。*CIVER* が *CIVER2* よりも小さい場合、このフィールドは存在しません。

CILEN (10 桁の符号付き整数)

MQCIH 構造体の長さ。

値は次のいずれかでなければなりません。

CILEN1

バージョン1のCICS情報ヘッダー構造体の長さ。

CILEN2

バージョン2のCICS情報ヘッダー構造体の長さ。

以下の定数は、現行バージョンの長さを指定しています。

CILENC

現行バージョンのCICS情報ヘッダー構造体の長さ。

これは要求フィールドです。フィールドの初期値はCILEN2です。

CILT (10 桁の符号付き整数)

リンクのタイプ。

ブリッジがリンクを試みるオブジェクトのタイプを指定します。値は次のいずれかでなければなりません。

LTPROG

DPL プログラム。

LTTRAN

3270 トランザクション。

これは要求フィールドです。フィールドの初期値はLTPROGです。

CINTI (4 バイトの文字ストリング)

生成する次のトランザクション。

これは、ユーザー・トランザクション (通常は EXEC CICS RETURN TRANSID) が戻す次のトランザクションの名前です。次のトランザクションがない場合は、このフィールドはブランクに設定されます。

これは、3270 トランザクションにのみ使用される応答フィールドです。このフィールドの長さは LNTRID によって指定されます。初期値は 4 桁のブランクです。

CIODL (10 桁の符号付き整数)

出力 COMMAREA データ長。

応答メッセージでクライアントに戻されるユーザー・データの長さです。この長さには、8 バイトのプログラム名も含まれます。リンクされたプログラムに渡される COMMAREA の長さは、このフィールドの長さと、要求メッセージ内のユーザー・データの長さから 8 を引いた長さのどちらか大きい方です。

注: メッセージ内のユーザー・データの長さは、MQCIH 構造体を除いたメッセージの長さです。

リクエストメッセージ内のユーザーデータの長さがより *CIODL* 小さい場合、コマンド LINK のオプション DATALENGTH が使用される。これにより、LINK を別の CICS リージョンへ効率的に機能転送することが可能となる。

以下のような特別な値を使用することができます。

OLINPT

出力長を入力長と同じにする。

リンクされたプログラムに渡す COMMAREA が必ず十分なサイズになるように、応答の要求がなくてもこの値が必要になることがあります。

これは、DPL プログラムにのみ使用される要求フィールドです。このフィールドの初期値は OLINPT です。

CIREA (10 桁の符号付き整数)

IBM MQ 理由コードまたはフィードバック・コード、あるいは CICS EIBRESP2。

このフィールドに戻される値は、*CIRET* の値によって決まります。[1036 ページの表 729](#) を参照してください。

これは応答フィールドです。このフィールドの初期値は RCNONE です。

CIRET (10 桁の符号付き整数)

ブリッジからの戻りコード。

CICS bridge で実行された処理の結果を示すブリッジからの戻りコードです。*CIFNC*、*CICC*、*CIREA*、および *CIAC* の各フィールドに、追加情報が格納されることがあります ([1036 ページの表 729](#) を参照してください)。値は、次のいずれか 1 つです。

CRC000

(0, X'000') エラーなし。

CRC001

(1, X'001') EXEC CICS 文でエラーが検出された。

CRC002

(2, X'002') IBM MQ 呼び出しでエラーが検出された。

CRC003

(3, X'003') CICS bridge でエラーが検出された。

CRC004

(4, X'004') CICS bridge が異常終了した。

CRC005

(5, X'005') アプリケーションが異常終了した。

CRC006

(6, X'006') セキュリティー・エラーが発生した。

CRC007

(7, X'007') プログラムが使用できない。

CRC008

(8, X'008') 指定された時間内に現行作業単位内の 2 番目以降のメッセージを受信しなかった。

CRC009

(9, X'009') トランザクションが使用できない。

これは応答フィールドです。このフィールドの初期値は CRC000 です。

CIRFM (8 バイトの文字ストリング)

応答メッセージの IBM MQ 形式名。

このフィールドは、現行メッセージに応答して送信される応答メッセージの IBM MQ 形式名です。コーディングの規則は、MQMD の *MDFMT* フィールドの場合と同じです。

これは、DPL プログラムにのみ使用される要求フィールドです。このフィールドの長さは *LNFMFT* によって指定されます。このフィールドの初期値は *FMNONE* です。

CIRSI (4 バイトの文字ストリング)

予約済み。

これは予約フィールドです。値は 4 個のブランクでなければなりません。このフィールドの長さは *LNRSID* によって指定されます。

CIRS1 (8 バイトの文字ストリング)

予約済み。

これは予約フィールドです。値は 8 個のブランクでなければなりません。

CIRS2 (8 バイトの文字ストリング)

予約済み。

これは予約フィールドです。値は 8 個のブランクでなければなりません。

CIRS3 (8 バイトの文字ストリング)

予約済み。

これは予約フィールドです。値は 8 個のブランクでなければなりません。

CIRS4 (10 桁の符号付き整数)

予約済み。

これは予約フィールドです。値はゼロでなければなりません。*CIVER* が *CIVER2* よりも小さい場合、このフィールドは存在しません。

CIRTI (4 バイトの文字ストリング)

予約済み。

これは予約フィールドです。値は 4 個のブランクでなければなりません。このフィールドの長さは *LNTRID* によって指定されます。

CISC (4 バイトの文字ストリング)

トランザクション開始コード。

これは、ブリッジが端末トランザクションをエミュレートするか、*START* されたトランザクションをエミュレートするかを指定する標識です。値は次のいずれかでなければなりません。

SCSTRT

開始。

SCDATA

データを開始する。

SCTERM

入力を終了。

SCNONE

なし。

ブリッジからの応答では、このフィールドは *CINTI* フィールド内の次のトランザクション ID に適した開始コードに設定されます。応答では、次の開始コードが使用されます。

- SCSTRT
- SCDATA
- SCTERM

CICS Transaction Server 1.2 の場合、このフィールドは要求フィールドのみです。応答での値は未定義です。

CICS Transaction Server 1.3 以降のリリースでは、これは要求フィールドでもあり、応答フィールドでもあります。

このフィールドは、3270 トランザクションにのみ使用されます。このフィールドの長さは *LNSTCO* によって指定されます。このフィールドの初期値は *SCNONE* です。

CISID (4 バイトの文字ストリング)

構造体 ID

値は次のものでなければなりません。

CISIDV

CICS 情報ヘッダー構造体の ID。

これは要求フィールドです。フィールドの初期値は *CISIDV* です。

CITES (10 桁の符号付き整数)

タスク終了時の状況。

このフィールドは、タスク終了時のユーザー・トランザクションの 状況を示します。次のいずれかの値が戻されます。

TENOSY

同期していない。

ユーザー・トランザクションはまだ完了しておらず、同期点に達していません。この場合、MQMD 内の *MDMT* フィールドは *MTRQST* です。

TECMIT

作業単位をコミットする。

ユーザー・トランザクションはまだ完了していませんが、最初の作業単位の同期点に達しています。この場合、MQMD 内の *MDMT* フィールドは *MTDGRM* です。

TEBACK

作業単位をバックアウトする。

ユーザー・トランザクションはまだ完了していません。現行の作業単位はバックアウトされます。この場合、MQMD 内の *MDMT* フィールドは *MTDGRM* です。

TEENDT

タスクを終了する。

ユーザー・トランザクションは終了 (または異常終了) しました。この場合、MQMD 内の *MDMT* フィールドは *MTRPLY* です。

これは、3270 トランザクションにのみ使用される応答フィールドです。このフィールドの初期値は *TENOSY* です。

CITI (4 バイトの文字ストリング)

生成するトランザクション。

CILT の値が *LTTRAN* の場合、*CITI* は、実行するユーザー・トランザクションのトランザクション ID です。この場合は、非ブランクの値を指定する必要があります。

CILT の値が *LTPROG* の場合、*CITI* は、該当する作業単位内のすべてのプログラムで使用するトランザクション・コードです。値としてブランクを指定した場合は、*CICS DPL* ブリッジのデフォルト・トランザクション・コード (*CKBP*) が使用されます。非ブランクの値を指定する場合は、初期プログラムが *CSQCBP00* のローカル TRANSACTION として *CICS* に定義してある必要があります。このフィールドは、*CIUOW* の値が *CUFRST* または *CUONLY* である場合にのみ適用されます。

これは要求フィールドです。このフィールドの長さは *LNTRID* によって指定されます。初期値は 4 桁のブランクです。

CIUOW (10 桁の符号付き整数)

作業単位制御。

これは、*CICS bridge* によって実行される作業単位の処理を制御します。ブリッジに対して、単一トランザクションの実行を要求すること、1つの作業単位内で1つ以上のプログラムの実行を要求することもできます。このフィールドでは、*CICS bridge* で別の作業単位を開始するか、要求された機能を現行の作業単位の中で実行するか、それとも、作業単位をコミットまたはバックアウトして終了させるかを指定します。データ伝送の流れを最適化するために、様々な組み合わせがサポートされます。

値は次のいずれかでなければなりません。

CUONLY

作業単位を開始し、関数を実行した上で、その作業単位をコミット (*DPL* および 3270)。

CUCONT

現行の作業単位の追加データ (3270 のみ)。

CUFRST

作業単位を開始し、関数を実行 (*DPL* のみ)。

CUMIDL

現行の作業単位の中で関数を実行 (*DPL* のみ)。

CULAST

関数を実行した上で、その作業単位をコミット (*DPL* のみ)。

CUCMIT

作業単位をコミットする (*DPL* のみ)。

CUBACK

作業単位をバックアウトする (*DPL* のみ)。

これは要求フィールドです。このフィールドの初期値は *CUONLY* です。

CIVER (10 桁の符号付き整数)

構造体のバージョン番号。

値は次のいずれかでなければなりません。

CIVER1

バージョン 1 の *CICS* 情報ヘッダー構造体。

CIVER2

バージョン 2 の *CICS* 情報ヘッダー構造体。

これより新しいバージョンの構造体にのみ存在するフィールドは、そのフィールドの説明にその旨記載されています。以下の定数は、現行バージョンのバージョン番号を指定しています。

CIVERC

CICS 情報ヘッダー構造体の現行バージョン。

これは要求フィールドです。このフィールドの初期値は *CIVER2* です。

初期値

表 730. MQCIH のフィールド		
フィールド名	定数の名前	定数の値
CISID	CISIDV	'CIH'
CIVER	CIVER2	2
CILEN	CILEN2	180
CIENC	なし	0
CICSI	なし	0
CIFMT	FMNONE	ブランク
CIFLG	CIFNON	0
CIRET	CRC000	0
CICC	CCOK	0
CIREA	RCNONE	0
CIUOW	CUONLY	273
CIGWI	WIDFLT	-2
CILT	LTPROG	1
CIODL	OLINPT	-1
CIFKT	なし	0
CIADS	ADNONE	0
CICT	CTNO	0
CITES	TENOSY	0
CIFAC	FCNONE	Null
CIFNC	CFNONE	ブランク
CIAC	なし	ブランク
CIAUT	なし	ブランク
CIRS1	なし	ブランク
CIRFM	FMNONE	ブランク
CIRSI	なし	ブランク
CIRTI	なし	ブランク
CITI	なし	ブランク
CIFL	なし	ブランク
CAIAI	なし	ブランク
CISC	SCNONE	ブランク
CICNC	なし	ブランク
CINTI	なし	ブランク
CIRS2	なし	ブランク

表 730. MQCIH のフィールド (続き)		
フィールド名	定数の名前	定数の値
CIRS3	なし	ブランク
CICP	なし	0
CIE0	なし	0
CIII	なし	0
CIRS4	なし	0

注:

1. 記号-は、単一のブランク文字を表します。

RPG 宣言

```

D*.1.....2.....3.....4.....5.....6.....7..
D* MQCIH Structure
D*
D* Structure identifier
D  CISID          1          4      INZ('CIH ')
D* Structure version number
D  CIVER          5          8I 0 INZ(2)
D* Length of MQCIH structure
D  CILEN          9         12I 0 INZ(180)
D* Reserved
D  CIENC         13         16I 0 INZ(0)
D* Reserved
D  CICSID        17         20I 0 INZ(0)
D* MQ format name of data that followsMQCIH
D  CIFMT         21         28      INZ('          ')
D* Flags
D  CIFLG         29         32I 0 INZ(0)
D* Return code from bridge
D  CIRET         33         36I 0 INZ(0)
D* MQ completion code or CICSEIBRESP
D  CICC          37         40I 0 INZ(0)
D* MQ reason or feedback code, or CICSEIBRESP2
D  CIREA         41         44I 0 INZ(0)
D* Unit-of-work control
D  CIUOW         45         48I 0 INZ(273)
D* Wait interval for MQGET call issuedby bridge task
D  CIGWI         49         52I 0 INZ(-2)
D* Link type
D  CILT          53         56I 0 INZ(1)
D* Output COMMAREA data length
D  CIODL         57         60I 0 INZ(-1)
D* Bridge facility release time
D  CIFKT         61         64I 0 INZ(0)
D* Send/receive ADS descriptor
D  CIADS         65         68I 0 INZ(0)
D* Whether task can beconversational
D  CICT          69         72I 0 INZ(0)
D* Status at end of task
D  CITES         73         76I 0 INZ(0)
D* Bridge facility token
D  CIFAC         77         84      INZ(X'00000000000000-
D                                     00')
D* MQ call name or CICS EIBFNfunction
D  CIFNC         85         88      INZ('          ')
D* Abend code
D  CIAC          89         92      INZ
D* Password or passticket
D  CIAUT         93        100      INZ
D* Reserved
D  CIRS1        101        108      INZ
D* MQ format name of reply message
D  CIRFM        109        116      INZ('          ')
D* Remote CICS system ID to use
D  CIRSI        117        120      INZ
D* CICS RTRANSID to use
D  CIRTI        121        124      INZ

```

D* Transaction to attach			
D CITI	125	128	INZ
D* Terminal emulated attributes			
D CIFL	129	132	INZ
D* AID key			
D CIAI	133	136	INZ
D* Transaction start code			
D CISC	137	140	INZ(' ')
D* Abend transaction code			
D CICNC	141	144	INZ
D* Next transaction to attach			
D CINTI	145	148	INZ
D* Reserved			
D CIRS2	149	156	INZ
D* Reserved			
D CIRS3	157	164	INZ
D* Cursor position			
D CICP	165	168I 0	INZ(0)
D* Offset of error in message			
D CIE0	169	172I 0	INZ(0)
D* Reserved			
D CIII	173	176I 0	INZ(0)
D* Reserved			
D CIRS4	177	180I 0	INZ(0)
D*			

IBM i IBM i での MQCMHO (メッセージ・ハンドルの作成オプション)

MQCMHO 構造体を使用すると、アプリケーションで、メッセージ・ハンドルを作成する方法を制御するオプションを指定できます。

概要

目的

この構造は、**MQCRTMH** 呼び出しの入力パラメーターです。

文字セットとエンコード

MQCMHO 内のデータは、アプリケーションの文字セットおよびアプリケーションのエンコードでなければなりません (ENNAT)。

- [1048 ページの『フィールド』](#)
- [1050 ページの『初期値』](#)
- [1050 ページの『RPG 宣言』](#)

フィールド

MQCMHO 構造体には、以下のフィールドが含まれます。フィールドはアルファベット順に説明されています。

CMOPT (10 桁の符号付き整数)

以下のいずれかのオプションを指定できます。

CMVAL

MQSETMP を呼び出してこのメッセージ・ハンドル中のプロパティを設定する際には、プロパティ名が妥当性検査されて、以下のことが確認されます。

- 無効文字が含まれていない。
- 以下を除き、"JMS" または "usr.JMS" で始まらない。
 - JMSCorrelationID
 - JMSReplyTo
 - JMSType
 - JMSXGroupID
 - JMSXGroupSeq

これらの名前は JMS プロパティ用に予約されています。

- 以下のいずれかのキーワードではない (小文字と大文字のすべての組み合わせを含む)。
 - "AND"
 - "BETWEEN"
 - "ESCAPE"
 - "false"
 - "IN"
 - "IS"
 - "LIKE"
 - "NOT"
 - "NULL"
 - 「または」
 - "true"
- 「本文」で始まらないまたは「ルート」 (「Root.MQMD」を除く)。

プロパティが MQ で定義されている場合 (「mq.*」) 名前が認識されると、プロパティ記述子フィールドはプロパティの正しい値に設定される。プロパティが認識されない場合は、プロパティ記述子の *Support* フィールドが **PDSUPO** に設定されます。(詳細は、[PDSUP](#) を参照してください。)

CMDEFV

これは、デフォルト・レベルのプロパティ名の妥当性検査が行われることを指定します。

妥当性検査のデフォルト・レベルは、**CMVAL** で指定されるレベルに相当します。

今後のリリースでは、**CMDEFV** の定義時に行われる妥当性検査のレベルを変更する管理オプションが定義される可能性があります。

これがデフォルト値です。

CMNOVA

プロパティ名に対する妥当性検査は行われません。**CMVAL** についての説明を参照してください。

デフォルト・オプション: このセクションで説明したオプションがいずれも必要ない場合、以下のオプションを使用できます。

CMNONE

すべてのオプションでデフォルト値が想定されます。この値を使用して、他のオプションが指定されていないことを示します。**CMNONE** は、プログラムの文書化を支援します。このオプションを他のオプションと一緒に使用することは意図されていませんが、値がゼロであるため、そのような使用を検出できません。

これは常に入力フィールドです。このフィールドの初期値は **CMDEFV** です。

CMSID (10 桁の符号付き整数)

これは構造体 ID です。値は以下のものでなければなりません。

CMSIDV

メッセージ・ハンドル作成オプション構造の ID。

これは常に入力フィールドです。このフィールドの初期値は **CMSIDV** です。

CMVER (10 桁の符号付き整数)

これは構造体のバージョン番号です。値は以下のものでなければなりません。

CMVER1

バージョン 1 のメッセージ・ハンドルの作成オプション構造。

以下の定数は、現行バージョンのバージョン番号を指定しています。

CMVERC

メッセージ・ハンドルの作成オプション構造の現行バージョン。

これは常に入力フィールドです。このフィールドの初期値は **CMVER1** です。

初期値

表 731. MQCMHO のフィールド		
フィールド名	定数の名前	定数の値
CMSID	CMSIDV	'CMHO'
CMVER	CMVER1	1
CMOPT	CMDEFV	0

RPG 宣言

```
D* MQCMHO Structure
D*
D*
D* Structure identifier
D CMSID 1 4 INZ('CMHO')
D*
D* Structure version number
D CMVER 5 8I 0 INZ(1)
D*
D* Options that control the action of MQCRTMH
D CMOPT 9 12I 0 INZ(0)
```

IBM i IBM i での MQCNO (接続オプション)

MQCNO 構造体を使用すると、アプリケーションで、ローカル・キュー・マネージャーへの接続に関するオプションを指定できます。

概要

目的: この構造体は、MQCONN 呼び出しの入出力パラメーターです。

バージョン: MQCNO の現行バージョンは CNVER6 です。これより新しいバージョンの構造体にのみ存在するフィールドについては、そのフィールドの説明にその旨を記載しています。

提供されている COPY ファイルには、環境でサポートされている最新バージョンの MQCNO が含まれていますが、CNVER フィールドの初期値は CNVER1 に設定されています。version-1 構造体に存在しないフィールドを使用するには、アプリケーションで、CNVER フィールドを必要なバージョンのバージョン番号に設定する必要があります。

文字セットとエンコード: MQCNO 内のデータは、**CodedCharSetId** キュー・マネージャー属性で指定された文字セットと、ENNAT で指定されたローカル・キュー・マネージャーのエンコードで記述されていなければなりません。

- [1051 ページの『フィールド』](#)
- [1056 ページの『初期値』](#)
- [1057 ページの『RPG 宣言』](#)

フィールド

MQCNO 構造体には、以下のフィールドが含まれます。フィールドはアルファベット順に説明されています。

CCDTUL (10 桁の符号付き整数)

CCDTUL は、接続に使用するクライアント接続チャンネル・テーブルの場所を示す URL が含まれる CCDTUP または CCDTUO のどちらかで指定されるストリングの長さです。

CCDTUL は、MQCONN 呼び出しを発行するアプリケーションが IBM MQ MQI client として実行されている場合のみ使用してください。

これは、[MQCHLLIB](#) および [MQCHLTAB](#) の環境変数を設定することに対する、プログラムによる代替手段です。

アプリケーションがクライアントとして実行されていない場合、CCDTUL は無視されます。

CNVER の値が CNVER6 より小さい場合は、このフィールドは無視されます。

CCDTUO (10 桁の符号付き整数)

CCDTUO は、MQCNO 構造体の始まりから、接続に使用するクライアント接続チャンネル・テーブルの場所を示す URL が含まれるストリングまでの、オフセットです (バイト単位)。オフセットの値は、正負どちらの値にもなります。

CCDTUL は、MQCONN 呼び出しを発行するアプリケーションが IBM MQ MQI client として実行されている場合のみ使用してください。

重要: CCDTUP と CCDTUO のいずれか 1 つだけ使用できます。両方のフィールドがゼロ以外の場合、呼び出しは理由コード RC2600 で失敗します。

これは、[MQCHLLIB](#) および [MQCHLTAB](#) の環境変数を設定することに対する、プログラムによる代替手段です。

アプリケーションがクライアントとして実行されていない場合、CCDTUO は無視されます。

CNVER の値が CNVER6 より小さい場合は、このフィールドは無視されます。

CCDTUP (ポインター)

CCDTUP は、接続に使用するクライアント接続チャンネル・テーブルの場所を示す URL が含まれるストリングへの、オプションのポインターです。

CCDTUP は、MQCONN 呼び出しを発行するアプリケーションが IBM MQ MQI client として実行されている場合のみ使用してください。

重要: CCDTUP と CCDTUO のいずれか 1 つだけ使用できます。両方のフィールドがゼロ以外の場合、呼び出しは理由コード RC2600 で失敗します。

これは、[MQCHLLIB](#) および [MQCHLTAB](#) の環境変数を設定することに対する、プログラムによる代替手段です。

アプリケーションがクライアントとして実行されていない場合、CCDTUP は無視されます。

CNVER の値が CNVER6 より小さい場合は、このフィールドは無視されます。

CNAN (28 バイトの文字ストリング)

キュー・マネージャーへの接続を識別するために、アプリケーションによって設定された名前。フィールドの初期値はヌル文字です。

CNVER の値が CNVER7 より小さい場合は、このフィールドは無視されます。

CNCCO (10 桁の符号付き整数)

これは、MQCNO 構造体の先頭からの MQCD チャンネル定義構造体のオフセットをバイト数で表したものです。

CNCCP (ポインター)

これは、MQCD チャネル定義構造体へのポインターです。

CNCONID (24 バイトの文字ストリング)

固有の接続 ID。このフィールドを使用すると、最初のキュー・マネージャーへの接続時に、アプリケーション・プロセスに固有 ID を割り当てることによって、キュー・マネージャーはそれを確実に識別できるようになります。

アプリケーションは、PUT および GET 呼び出しを行うときに相関を目的として接続 ID を使用します。すべての接続には、接続が確立された方法にかかわらず、キュー・マネージャーによって ID が割り当てられます。

長時間実行されている作業単位を強制的に終了するために接続 ID を使用することができます。これを行うには、PCF コマンド「Stop Connection」または MQSC コマンド STOP CONN を使用して接続 ID を指定します。これらのコマンドの使用法について詳しくは、関連リンクを参照してください。

フィールドの初期値は、24 個のヌル・バイトです。

CNCT (128 バイトのビット・ストリング)

これは、キュー・マネージャーが、アプリケーションの接続の際に影響を受けるリソースに関連付けるタグです。

キュー・マネージャー接続タグ。

各アプリケーションやアプリケーション・インスタンスのタグにはそれぞれ異なる値を使用することによって、キュー・マネージャーが、影響を受けるリソースへのアクセスを正しくシリアライズできるようにする必要があります。詳細については、CN*CT* オプションを参照してください。アプリケーションが終了すると、または、MQDISC 呼び出しが行われると、タグは無効になります。

タグが必要ない場合は、次の特殊値を使用します。

CTNONE

接続は指定されません。

値は、フィールドの長さについては 2 進ゼロです。

これは入力フィールドです。このフィールドの長さは LNCTAG によって指定されます。フィールドの初期値は CTNONE です。CNVER が CNVER3 より小さい場合、このフィールドは無視されます。

z/OS キュー・マネージャーに接続する場合はフィールド ConnTag を使用します。

CNNORES2 (4 バイトの文字ストリング)

構造体を 64 ビット境界外に埋め込む予約フィールド。フィールドの初期値は、フィールドの長さを示す 2 進ゼロです。

CNVER の値が CNVER7 より小さい場合は、このフィールドは無視されます。

CNOPT (10 桁の符号付き整数)

MQCONN のアクションを制御するオプション。

バインディング・オプション

バインディング・オプションを指定すると、使用される IBM MQ バインディングのタイプが制御されます。これらのオプションのうち指定できるのは 1 つのみです。

CNSBND

標準バインディング。

標準バインディング・オプションを指定すると、アプリケーションとローカル・キュー・マネージャー・エージェントがそれぞれ別の実行単位 (通常は、別のプロセス) で実行されます。この構成によって、キュー・マネージャーの健全性が維持されます。つまり、エラーが発生したプログラムからキュー・マネージャーが保護されます。

アプリケーションのテストがまだ不十分な場合や、確実性や信頼性に欠ける場合は、CNSBND を使用してください。CNSBND はデフォルトです。

CNSBND は、プログラム文書化を支援するために定義します。このオプションは、使用するバインディングのタイプを制御するその他のオプションと組み合わせて使用しないでください。ただし、このオプションの値はゼロと等価なため、そのような使い方をしても、エラーとして検出されることはありません。

このオプションはすべての環境でサポートされます。

CNFBND

ファースト・パス・バインディング。

ファースト・パス・バインディング・オプションを指定すると、アプリケーションとローカル・キュー・マネージャー・エージェントが同じ実行単位で実行されます。ファースト・パスは、アプリケーションとローカル・キュー・マネージャー・エージェントが別々の実行単位で実行される標準バインディングとは対照的です。

キュー・マネージャーがこのタイプのバインディングをサポートしていない場合、CNFBND は無視されます。この場合の処理は、このオプションが指定されていない場合と同じように実行されます。

複数のプロセスのリソース消費量がアプリケーションで使用される全リソースと比較して多い条件では、CNFBND を指定すると有効な場合があります。ファースト・パス・バインディングを使用するアプリケーションのことを、トラステッド・アプリケーションと呼びます。

ファースト・パス・バインディングを使用するかどうかを決める際には、必ず以下の重要事項を考慮に入れてください。

- **CNFBND オプションを使用すると、アプリケーションにより、キュー・マネージャーに属するメッセージや他のデータ領域が変更または破壊されることがあります。このオプションは、これらの問題を十分に評価した状況においてのみ使用してください。**
- アプリケーションは、CNFBND で非同期シグナルまたはタイマー割り込み (sigkill など) を使用してはなりません。また、共用メモリー・セグメントを使用する場合にもいくつかの制限があります。
- このアプリケーションでは、一度に複数のスレッドをキュー・マネージャーに接続することはできません。
- キュー・マネージャーから切断するには、アプリケーションは MQDISC 呼び出しを使用する必要があります。
- `endmqm` コマンドを使用してキュー・マネージャーを終了する前に、アプリケーションを終了する必要があります。

以下に示す点は、記されている環境における CNFBND の使用に適用されます。

- IBM i では、ジョブは、QMADM グループに属するユーザー・プロファイル QMQM の下で実行する必要があります。さらに、プログラムは決して異常終了させないでください。そうしないと予期しない結果が発生することがあります。

トラステッド・アプリケーションの使用の詳細については、[MQCONN 呼び出しを使用したキュー・マネージャーへの接続](#) および [トラステッド・アプリケーションの制約事項](#)を参照してください。

CNSHBD

共有バインディング。

共有バインディング・オプションを指定すると、アプリケーションとローカル・キュー・マネージャー・エージェントがそれぞれ別の実行単位 (通常は、別のプロセス) で実行されます。この構成によって、キュー・マネージャーの安全性が維持されます。つまり、エラーが発生したプログラムからキュー・マネージャーが保護されます。ただし、一部のリソースはアプリケーションとローカル・キュー・マネージャー・エージェントとの間で共有されます。キュー・マネージャーがこのタイプのバインディングをサポートしていない場合、CNSHBD は無視されます。処理は、オプションが指定されなかったものとして続行します。

CNIBND

分離バインディング。

分離バインディング・オプションを指定すると、アプリケーションとローカル・キュー・マネージャー・エージェントがそれぞれ別の実行単位 (通常は、別のプロセス) で実行されます。この構成によって、キュー・マネージャーの健全性が維持されます。つまり、エラーが発生したプログラムからキュー・マネージャーが保護されます。アプリケーション・プロセスおよびローカル・キュー・マネージャー・エージェントは、リソースを共有しないという点で互いに分離しています。キュー・マネージャーがこのタイプのバインディングをサポートしていない場合、CNIBND は無視されます。処理は、オプションが指定されなかったものとして続行します。

ハンドル共有オプション

以下のオプションは、同じプロセス内の異なるスレッド (並列処理の単位) 間でのハンドルの共有を制御します。これらのオプションは、1 つだけ指定できます。

CNHSN

スレッド間のハンドル共有なし。

スレッド間のハンドル共有なしのオプションは、接続とオブジェクトのハンドルを使用できるのは、そのハンドルを割り振りしたスレッド、つまり、MQCONN、MQCONNX、または MQOPEN 呼び出しを発行したスレッドのみであることを示します。同じプロセスに属する他のスレッドはそのハンドルを使用できません。

CNHSB

呼び出しブロックのある、スレッド間のシリアル・ハンドル共有。

呼び出しブロックのあるスレッド間のシリアル・ハンドル共有オプションは、プロセス内のあるスレッドによって割り振られた 接続とオブジェクトのハンドルを、同じプロセスに属する他のスレッドも使用できることを示します。ただし、特定のハンドルを使用できるのは一度に 1 つのスレッドのみです。つまり、ハンドルのシリアル使用のみが許可されています。すでに別のスレッドによって使用されているハンドルを使用しようとすると、そのハンドルが使用可能になるまで呼び出しはブロック (待機) されます。

CNHSNB

呼び出しブロックのない、スレッド間のシリアル・ハンドル共有。

スレッド間でのシリアル・ハンドル共有 (呼び出しブロックなし) オプションは、「ブロックあり」オプションと同じです。ただし、ハンドルが別のスレッドによって使用されている場合、呼び出しは、ハンドルが使用可能になるまでブロックするのではなく、CCFAIL および RC2219 で即時に完了します。

スレッドはゼロまたは 1 つの非共有ハンドルと、ゼロまたは 1 つ以上の共有ハンドルを持つことができます。

- CNHSN を指定する MQCONN または MQCONNX 呼び出しはそれぞれ、最初の呼び出しで新しい非共有ハンドルを戻し、2 回目以降の呼び出しで同じ非共有ハンドルを戻します (MQDISC 呼び出しが介入しないと想定しています)。2 回目以降の呼び出しの理由コードは RC2002 です。
- CNHSB または CNHSNB を指定する MQCONNX 呼び出しはそれぞれ、各呼び出しで新しい共有ハンドルを戻します。

オブジェクト・ハンドルは、そのオブジェクト・ハンドルを作成した MQOPEN 呼び出しで指定されている接続ハンドルと同じ共有プロパティを継承します。また、作業単位は、作業単位を開始するために使用される接続ハンドルと同じ共有プロパティを継承します。共有ハンドルを使用するスレッドで開始された作業単位は、同じハンドルを使用する別のスレッドで更新できます。

ハンドル共有オプションを指定しない場合、環境別に次のようなデフォルトが指定されます。

- Microsoft Transaction Server (MTS) 環境では、デフォルトは CNHSB と同じです。
- その他の環境では、デフォルトは CNHSN と同じです。

再接続オプション

再接続オプションにより、ある接続が再接続可能かどうかが決まります。再接続可能なのはクライアント接続のみです。

CNRCDF

再接続オプションは、デフォルト値に解決されます。デフォルトが設定されない場合、このオプションの値は **DISABLED** に解決されます。オプションの値はサーバーに渡され、**PCF** および **MQSC** が照会できるようになります。

CNRC

アプリケーションは、MQCONN の **QMNAME** パラメーターの値と整合した任意のキュー・マネージャーと再接続できます。CNRC オプションは、クライアント・アプリケーションとそれが最初に接続を確立したキュー・マネージャーの間にアフィニティーがない場合にのみ使用します。オプションの値はサーバーに渡され、**PCF** および **MQSC** が照会できるようになります。

CNRCDF

アプリケーションを再接続できません。このオプションの値はサーバーに渡されません。

CNRCQM

アプリケーションは、最初に接続したキュー・マネージャーとのみ再接続できます。この値は、クライアントに再接続できるが、クライアント・アプリケーションとそれが最初に接続を確立したキュー・マネージャーとの間にアフィニティーがある場合に使用します。高可用性キュー・マネージャーの待機インスタンスにクライアントを自動再接続する場合にこの値を選択します。オプションの値はサーバーに渡され、**PCF** および **MQSC** が照会できるようになります。

オプション CNRC、CNRCDF、および CNRCQM は、クライアント接続に対してのみ使用します。オプションがバインディング接続に対して使用されると、MQCONN は、完了コード MQCC_FAILED および理由コード MQRC_OPTIONS_ERROR で失敗します。

デフォルト・オプション: 上記で説明されたオプションがいずれも必要でない場合、以下のオプションを使用できます。

CNNONE

オプションが指定されていません。

CNNONE は、プログラム文書化を支援するために定義します。このオプションは、他の CN* オプションと組み合わせて使用するオプションではありません。ただし、このオプションの値はゼロと等価なため、他のオプションと組み合わせて使用しても、エラーとして検出されることはありません。

CNSCO (10 桁の符号付き整数)

これは、MQCNO 構造体の先頭からの MQSCO 構造体のオフセットをバイト数で表したものです。

CNVER が CNVER4 より小さい場合、このフィールドは無視されます。

CNSCP (ポインター)

これは MQSCO 構造体のアドレスです。

CNVER が CNVER4 より小さい場合、このフィールドは無視されます。

CNSECPO (10 桁の符号付き整数)

セキュリティ・パラメーターのオフセット。ユーザー ID およびパスワードを指定するために使用される MQCSP 構造体のオフセット。

この値は、正負いずれの値にもなります。このフィールドの初期値は 0 です。

CNVER が CNVER5 より小さい場合、このフィールドは無視されます。

CNSECPP (ポインター)

セキュリティ・パラメーターのポインター。ユーザー ID およびパスワードを指定するために使用される MQCSP 構造体のアドレス。

このフィールドの初期値は、ヌル・ポインターまたはヌル・バイトです。

CNVER が CNVER5 より小さい場合、このフィールドは無視されます。

CNSID (4 バイトの文字ストリング)

MQCNO 構造体の構造体 ID。

値は次のものでなければなりません。

CNSIDV

接続オプション構造体の ID。

これは常に入力フィールドです。このフィールドの初期値は CNSIDV です。

CNVER (10 桁の符号付き整数)

MQCNO 構造体の構造体バージョン番号。

値は次のものでなければなりません。

CNVER6

バージョン 6 の接続オプション構造体。

このバージョンはすべての環境でサポートされます。

CNVER7

バージョン 7 の接続オプション構造体。

このバージョンはすべての環境でサポートされます。

以下の定数は、現行バージョンのバージョン番号を指定しています。

CNVERC

接続オプション構造体の現行バージョン。

これは常に入力フィールドです。このフィールドの初期値は CNVER7 です。

初期値

表 732. MQCNO のフィールド		
フィールド名	定数の名前	定数の値
CNSID	CNSIDV	'CNO~
CNVER	CNVER5	1
CNOPT	CNNONE	0
CNCCO	なし	0
CNCCP	なし	ヌル・ポインターまたはヌル・バイト
CNCT	CTNONE	Null
CNSCP	なし	ヌル・ポインターまたはヌル・バイト
CNSCO	なし	0
CNCONID	なし	Null
CNSECPO	なし	0
CNSECPP	なし	ヌル・ポインターまたはヌル・バイト
CCDTUL	なし	0
CCDTUO	なし	0
CCDTUP	なし	ヌル・ポインターまたはヌル・バイト

注:

1. 記号-は、単一のブランク文字を表します。

RPG 宣言

```
D*****
D**                                     **
D**          IBM MQ for IBM i          **
D**                                     **
D** FILE NAME:      CMQCN OG          **
D**                                     **
D** DESCRIPTION:    MQCNO Structure -- Connect Options **
D**                                     **
D*****
D** <N_OCO_COPYRIGHT>                  **
D** Licensed Materials - Property of IBM **
D**                                     **
D** 5724-H72                          **
D** (c) Copyright IBM Corp. 1993, 2026. All Rights Reserved. **
D**                                     **
D** US Government Users Restricted Rights - Use, duplication or **
D** disclosure restricted by GSA ADP Schedule Contract with **
D** IBM Corp.                          **
D** <NOC_COPYRIGHT>                    **
D*****
D** FUNCTION:        This file declares the structure MQCNO, **
D**                  which is used by the main MQI.          **
D**                                     **
D** PROCESSOR:       RPG (ILE)          **
D**                                     **
D*****
D*
D*
D*****
D** <BEGIN_BUILDINFO>                  **
D** Generated on:    08/02/16 13:50      **
D** Build Level:     L000000            **
D** Build Type:      Production          **
D** Pointer Size:    128 Bit             **
D** Source File:     **
D** CMQCN OG         **
D** <END_BUILDINFO> **
D*****
D*
D*.1.....2.....3.....4.....5.....6.....7..
D*
D*
D* MQCNO Structure
D*
D* Structure identifier
D  CNSID          1      4      INZ('CNO ')
D* Structure version number
D  CNVER          5      8I 0 INZ(1)
D* Options that control the action of MQCONN
D  CNOPT          9      12I 0 INZ(0)
D* Ver:1 **
D* Offset of MQCD structure for client connection
D  CNCCO          13     16I 0 INZ(0)
D* Address of MQCD structure for client connection
D  CNCCP          17     32*   INZ(*NULL)
D* Ver:2 **
D* Queue managerconnection tag
D  CNCT          33     160    INZ(X'0000000000000000-
D                                000000000000000000000000-
D                                000000000000000000000000-
D                                000000000000000000000000-
D                                000000000000000000000000-
D                                000000000000000000000000-
D                                000000000000000000000000-
D                                000000000000000000000000-
D                                000000000000000000000000-
D                                000000000000000000000000-
D                                000000000000000000000000-
D                                000000000000000000000000-
D                                000000000000000000000000-
D                                000000000000000000000000-
D* Ver:3 **
D* Address of MQSCO structure for client connection
D  CNSCP          161    176*   INZ(*NULL)
D* Offset of MQSCO structure for client connection
```

```

D  CNSCO                177      180I 0 INZ(0)
D* Ver:4 **
D* Unique Connection Identifier
D  CNCONID              181      204      INZ(X'0000000000000000-
D                                0000000000000000000000-
D                                000000')
D* Offset of MQCSP structure
D  CNSECP0              205      208I 0 INZ(0)
D* Address of MQCSP structure
D  CNSECPP              209      224*      INZ(*NULL)
D* Ver:5 **
D* Address of CCDT URL string
D  CNCCDTUP            225      240*      INZ(*NULL)
D* Offset of CCDT URL string
D  CNCCDTU0            241      244I 0 INZ(0)
D* Length of CCDT URL
D  CNCCDTUL            245      248I 0 INZ(0)
D* Ver:6 **
D*
D*****
D** End of CMQCN0G **
D*****

```

IBM i での MQCSP (セキュリティ・パラメーター)

IBM i の MQCSP 構造体の要約。

概要

目的: MQCSP 構造体は、ユーザー ID とパスワードを認証する許可サービスを使用可能にします。MQCONN 呼び出しで、MQCSP 接続セキュリティ・パラメーター構造体を指定します。

文字セットとエンコード: MQCSP 内のデータは、**CodedCharSetId** キュー・マネージャー属性で指定された文字セットと、ENNAT で指定されたローカル・キュー・マネージャーのエンコードで記述されていなければなりません。

- [1058 ページの『フィールド』](#)
- [1060 ページの『初期値』](#)
- [1061 ページの『RPG 宣言』](#)

フィールド

MQCSP 構造体には、以下のフィールドが含まれます。フィールドは**アルファベット順**に説明されています。

CSAUTH (10 桁の符号付き整数)

これは、実行する認証のタイプです。

有効な値は以下のとおりです。

CSAN

ユーザー ID とパスワードのフィールドを使用しません。

CSAUIAP

ユーザー ID とパスワードのフィールドを認証します。

これは入力フィールドです。このフィールドの初期値は CSAN です。

CSCPPL (10 桁の符号付き整数)

これは、認証で使用するパスワードの長さです。

パスワードの最大長は、プラットフォームに依存しません。パスワードの長さが許可されている長さを超える場合は、認証要求は失敗し、RC2035 が戻ります。

これは入力フィールドです。このフィールドの初期値は 0 です。

CSCPPO (10 桁の符号付き整数)

これは、認証で使用するパスワードのオフセットをバイト数で表したものです。

オフセットの値は、正負どちらの値にもなります。

これは入力フィールドです。このフィールドの初期値は 0 です。

CSCPPP (ポインター)

認証に使用されるパスワードのアドレスです。

これは入力フィールドです。このフィールドの初期値は、ヌル・ポインターです。

CSCSPUIL (10 桁の符号付き整数)

これは、認証で使用するユーザー ID の長さです。

ユーザー ID の最大長は、プラットフォームに依存しません。ユーザー ID の長さが許可されている長さを超える場合は、認証要求は失敗し、RC2035 が戻ります。

これは入力フィールドです。このフィールドの初期値は 0 です。

CSCSPUIO (10 桁の符号付き整数)

これは、認証で使用するユーザー ID のオフセットをバイト数で表したものです。

オフセットの値は、正負どちらの値にもなります。

これは入力フィールドです。このフィールドの初期値は 0 です。

CSCSPUIP (ポインター)

これは、認証に使用されるユーザー ID のアドレスです。

これは入力フィールドです。このフィールドの初期値は、ヌル・ポインターです。CSVER の値が CSVER5 より小さい場合は、このフィールドは無視されます。

CSINITKL (10 桁の符号付き整数)

これはパスワード保護システムの初期キーの長さです。

これは入力フィールドです。このフィールドの初期値は 0 です。

CSINITKO (10 桁の符号付き整数)

これは、パスワード保護システムの初期キーのバイト単位のオフセットです。オフセットの値は、正負どちらの値にもなります。

初期キーを指定するには *CSINITKP*、または *CSINITKO* のいずれかを使用できますが、両方を使用することはできません。詳細については、フィールド *CSINITKP* の説明を参照してください。

これは入力フィールドです。このフィールドの初期値は 0 です。

CSINITKP (ポインタ)

これはパスワード保護システムの初期キーのアドレス（バイト単位）です。

これは入力フィールドです。このフィールドの初期値は、ヌル・ポインターです。

IBM MQ MQI clients 一部のフィールドの値を、パスワード IBM MQ 保護システムを使用して暗号化された値として提供できます。以下のフィールドには暗号化された値を含めることができます：

- MQSCO 構造内のキーリポジトリパスワード。

暗号化アルゴリズムは、これらの値を暗号化および復号化するために初期鍵を使用します。これらのフィールドの値をユーティリティ **runmqicred** で暗号化する際に初期キーが指定された場合、クライアントがキューマネージャに接続する際には、同じ初期キーを指定する必要があります。

このフィールドを使用して指定された初期キーは、*MQS MQI KEYFILE* 環境変数またはクライアント設定ファイルの Security スタンス内の *MQIInitialKeyFile* プロパティを使用して指定された初期キーを上書きします。

初期キーを指定するには *CSINITKP*、または *CSINITKO* のいずれかを使用できますが、両方を使用することはできません。

CSRE1 (4 バイトの文字ストリング)

IBM i 上のポインターの位置合わせに必要な予約フィールド。

これは入力フィールドです。このフィールドの初期値は、すべてヌルです。

CSRS2 (8 バイトの文字ストリング)

IBM i 上のポインターの位置合わせに必要な予約フィールド。

これは入力フィールドです。このフィールドの初期値は、すべてヌルです。

CSSID (4 バイトの文字ストリング)

構造体 ID

値は次のものでなければなりません。

CSSIDV

セキュリティ・パラメーター構造体の ID。

CSVER (10 桁の符号付き整数)

構造体のバージョン番号。

値は次のものでなければなりません。

CSVER1

バージョン 1 のセキュリティ・パラメーター構造体。

CSVER2

Version-2 セキュリティパラメータ構造体。

以下の定数は、現行バージョンのバージョン番号を指定しています。

CSVERC

セキュリティ・パラメーター構造体の現行バージョン。

これは常に入力フィールドです。このフィールドの初期値は CSVER1 です。

初期値

表 733. MQCNO のフィールド		
フィールド名	定数の名前	定数の値
CSSID	CSSIDV	'CSP-'
CSVER	CSVER1	1
CSAUTHT	なし	0
CSRE1	なし	Null
CSCSPUIP	なし	NULL ポインター
CSCSPUIO	なし	0
CSCSPUIL	なし	0
CSRS2	なし	Null
CSCPPP	なし	NULL ポインター
CSCPP0	なし	0
CSCPPL	なし	0
CSINITKP	なし	NULL ポインター

表 733. MQCNO のフィールド (続き)		
フィールド名	定数の名前	定数の値
CSINITKO	なし	0
CSINITKL	なし	0

注:

1. 記号~は、単一のブランク文字を表します。

RPG 宣言

```

D*..1.....2.....3.....4.....5.....6.....7..
D*
D* MQCSP Structure
D*
D* Structure identifier
D CSSID          1      4      INZ('CSP ')
D* Structure version number
D CSVER          5      8I 0 INZ(1)
D* Type of authentication
D CSAUTHT        9     12I 0 INZ(0)
D* Reserved
D CSRE1         13     16      INZ(X'00000000')
D* Address of user ID
D CSCSPUIP      17     32*      INZ(*NULL)
D* Offset of user ID
D CSCSPUIO     33     36I 0 INZ(0)
D* Length of user ID
D CSCSPUIL     37     40I 0 INZ(0)
D* Reserved
D CSRS2         41     48      INZ(X'0000000000000000')
D* Address of password
D CSCPPP        49     64*      INZ(*NULL)
D* Offset of password
D CSCPP0       65     68I 0 INZ(0)
D* Length of password
D CSCPPL       69     72I 0 INZ(0)

```

IBM i IBM i での MQCTLO (コールバック制御オプション構造)

コールバック制御関数を指定する構造。

概要

目的

MQCTLO 構造体は、コールバック制御関数に関係したオプションを指定するために使用されます。

この構造は、[MQCTL](#) 呼び出しの入出力パラメーターです。

バージョン

MQCTLO の現行バージョンは CTLV1 です。

文字セットとエンコード

MQCTLO 内のデータは、**CodedCharSetId** キュー・マネージャー属性で指定された文字セットと、ENNAT で指定されたローカル・キュー・マネージャーのエンコードで記述されていなければなりません。ただし、アプリケーションが IBM MQ クライアントとして実行されている場合、構造体はクライアントの文字セットとエンコードに従っている必要があります。

- [1062 ページの『フィールド』](#)
- [1063 ページの『初期値』](#)
- [1063 ページの『RPG 宣言』](#)

フィールド

MQCTLO 構造体には、以下のフィールドが含まれます。フィールドはアルファベット順に説明されています。

COCONNAREA (10 桁の符号付き整数)

制御オプション構造体 - ConnectionArea フィールド。

これは、コールバック関数ができるフィールドです。

キュー・マネージャーは、このフィールドの内容に基づいて何かを決定することではなく、この値は MQCB 呼び出しのパラメーターとして、MQCBC 構造体の CBCCONNAREA フィールドから未変更のまま渡されます。

このフィールドは、CTLSR および CTLSW 以外のすべての操作では無視されます。

これは、コールバック関数への入出力フィールドです。このフィールドの初期値は、ヌル・ポインターまたはヌル・バイトです。

COOPT (10 桁の符号付き整数)

MQCTLO のアクションを制御するオプション。

CTLFQ

キュー・マネージャーまたは接続が静止状態にある場合、MQCTLO 呼び出しを強制的に失敗させます。

メッセージ・コンシューマーが静止する場合に、メッセージ・コンシューマーに通知されるようにするには、MQCB 呼び出しに渡される MQGMO オプションに GMFIQ を指定します。

CTLTHR

このオプションは、アプリケーションで、同じ接続に関するすべてのメッセージ・コンシューマーが同じスレッド上で呼び出される必要があることを、システムに通知します。

デフォルト・オプション: 上記のいずれのオプションも必要でない場合は、以下のオプションを使用します。

CTLNO

この値は、他のオプションが指定されなかったことを示すために使用します。すべてのオプションはデフォルト値であるとみなされます。CTLNO は、プログラムの文書化を支援するために定義されています。このオプションは、他のオプションと組み合わせて使用するオプションではありません。ただし、このオプションの値はゼロであるため、他のオプションを組み合わせて使用しても、それを検出できません。

これは入力フィールドです。COOPT フィールドの初期値は CTLNO です。

CORSV (10 桁の符号付き整数)

これは予約フィールドです。このフィールドの初期値は、ブランク文字です。

COSID (10 桁の符号付き整数)

制御オプション構造体 - StrucId フィールド。

これは構造体 ID です。値は以下のものでなければなりません。

CTLSI

制御オプションの構造の ID。

これは常に入力フィールドです。このフィールドの初期値は CTLSI です。

COVER (10 桁の符号付き整数)

制御オプション構造体 - Version フィールド。

これは構造体のバージョン番号です。値は以下のものでなければなりません。

CTLV1

バージョン 1 の制御オプション構造。

以下の定数は、現行バージョンのバージョン番号を指定しています。

CTLCV

制御オプション構造の現行バージョン。

これは常に入力フィールドです。このフィールドの初期値は CTLV1 です。

初期値

表 734. MQCTLO のフィールド		
フィールド名	定数の名前	定数の値
COSID	CTLSI	'CTLO'
COVER	CTLV1	1
COOPT	CTLNO	Null
CORSV	予約フィールド	
COCONNAREA	なし	ヌル・ポインターまたはヌル・バイト

RPG 宣言

```
D* MQCTLO Structure
D*
D*
D* Structure identifier
D COSID          1      4    INZ('CTLO')
D*
D* Structure version number
D COVER          5      8I 0 INZ(1)
D*
D* Options that control the action of MQCTL
D COOPT          9     12I 0 INZ(0)
D*
D* Reserved
D CORSV         13     16I 0 INZ(-1)
D*
D* MQCTL Data area passed to the function
D COCONNAREA    17     32*  INZ(*NULL)
```

IBM i IBM i での MQDH (配布ヘッダー)

MQDH 構造体は、メッセージが伝送キューに格納されている配布リスト・メッセージである場合に、そのメッセージ内の追加データを記述します。

概要

目的: 配布リスト・メッセージは、複数の宛先キューに送信されるメッセージです。追加データは、MQDH 構造体、MQOR レコードの配列、MQPMR レコードの配列の順番で構成されています。

この構造体は、メッセージを伝送キューに直接書き込んだり、伝送キューからメッセージを除去したりする特殊なアプリケーション (例えば、メッセージ・チャンネル・エージェント) で使用するためのものです。

メッセージを単に配布リストに書き込むだけの通常のアプリケーションでは、この構造体は使用しないでください。このようなアプリケーションの場合、MQOD 構造体を使用して配布リストに宛先を定義したり、MQPMO 構造体を使用してメッセージ・プロパティの指定や個々の宛先に送信されるメッセージについての情報を受信したりします。

文字セットとエンコード: MQDH 内のデータは、C プログラミング言語の場合、**CodedCharSetId** キュー・マネージャー属性で指定した文字セットと ENNAT で指定したローカル・キュー・マネージャーのエンコードで記述されていなければなりません。

MQDH の文字セットおよびエンコードは、以下の **MDCSI** および **MDENC** フィールドで設定されます。

- MQMD (MQDH 構造体がメッセージ・データの開始点にある場合)
- MQDH 構造体に先行するヘッダー構造体 (上記以外の場合)

使用法: アプリケーションによりメッセージが配布リストに書き込まれ、一部またはすべての宛先がリモートである場合には、キュー・マネージャーは、アプリケーション・メッセージ・データに MQXQH および MQDH 構造体を接頭部として付け、そのメッセージを該当する伝送キューに入れます。したがって、伝送キューにメッセージが入っている場合は、データの順序は以下のようになります。

- MQXQH 構造
- MQDH 構造体、および、MQOR レコードと MQPMR レコードの配列
- アプリケーション・メッセージ・データ

宛先によっては、このようなメッセージが 2 つ以上、キュー・マネージャーで生成され、別々の伝送キューに配置される場合があります。この場合には、これらのメッセージ内の MQDH 構造体によって、アプリケーションでオープンされている配布リストに定義された宛先ごとのサブセットをそれぞれ識別します。

配布リスト・メッセージを伝送キューに直接書き込むアプリケーションでは、データの順序は上記と同じであり、また MQDH 構造体が正しく指定されている必要があります。MQDH 構造体が無効の場合、キュー・マネージャーにより、MQPUT または MQPUT1 呼び出しが失敗し、理由コード RC2135 が戻される場合もあります。

メッセージを配布リスト形式でキューに入れられるのは、そのキューが配布リスト・メッセージをサポートできるように定義されている場合のみです (1380 ページの『[キューの属性](#)』で説明している **DistLists** キュー属性を参照)。配布リストをサポートしないキューにアプリケーションから配布リスト・メッセージを直接書き込んだ場合、キュー・マネージャーは配布リストを個別のメッセージに分割し、代わりにそれらのメッセージをキューに入れます。

- [1064 ページの『フィールド』](#)
- [1067 ページの『初期値』](#)
- [1067 ページの『RPG 宣言』](#)

フィールド

MQDH 構造体には、以下のフィールドが含まれます。フィールドは**アルファベット順**に説明されています。

DHCNT (10 桁の符号付き整数)

存在している MQOR レコードの数。

このフィールドには、宛先数を定義します。配布リストには常に少なくとも 1 つの宛先が含まれていなければならないため、DHCNT は常にゼロより大きくなければなりません。

このフィールドの初期値は 0 です。

DHCSI (10 桁の符号付き整数)

MQOR レコードまたは MQPMR レコードに続くデータの文字セット ID。

ここでは、MQOR および MQPMR レコードの配列の後に続くデータの文字セット ID を指定します。これは、MQDH 構造体自体の文字データには適用されません。

MQPUT または MQPUT1 呼び出しでは、アプリケーションは、このフィールドをデータに適切な値に設定する必要があります。以下のような特別な値を使用することができます。

CSINHT

この構造体の文字セット ID を継承する。

この構造体の後に続くデータの文字データは、この構造体に設定されているのと同じ文字セットになります。

キュー・マネージャーは、メッセージで送信される構造体の中のこの値を、構造体の実際の文字セット ID に変更します。エラーが発生しない限り、値 CSINHT が MQGET 呼び出しによって返されることはありません。

MQMD の MDPAT フィールドの値が ATBRKR の場合、CSINHT は使用できません。

このフィールドの初期値は CSUNDF です。

DHENC (10 桁の符号付き整数)

MQOR および MQPMR レコードの後に続くデータの数値エンコード。

ここでは、MQOR および MQPMR レコードの配列の後に続くデータの数値エンコードを指定します。これは、MQDH 構造体自体の文字データには適用されません。

MQPUT または MQPUT1 呼び出しでは、アプリケーションは、このフィールドをデータに適切な値に設定する必要があります。

このフィールドの初期値は 0 です。

DHFLG (10 桁の符号付き整数)

汎用フラグ。

以下のフラグを指定できます。

DHFNEW

新しいメッセージ ID を生成します。

このフラグは、配布リスト内の宛先ごとにそれぞれ新しいメッセージ ID が生成されることを示します。これは、put-message レコードが存在しない場合、またはレコードが存在してもそのレコードにフィールド *PRMID* が含まれていない場合のみ設定できます。

このフラグを使用すると、メッセージ ID の生成が、可能な限り最後まで、つまり配布リスト・メッセージが最終的に個々のメッセージに分割されるときまで延期されます。これにより、配布リスト・メッセージを介してやりとりする必要がある制御情報の量を最小限に抑えられます。

アプリケーションによりメッセージが配布リストに書き込まれると、キュー・マネージャーは、以下の記述がいずれも該当する場合に MQDH を生成し、その中に DHFNEW を設定します。

- アプリケーションによって提供された書き込みメッセージ・レコードがないか、提供されたレコードに *PRMID* フィールドが含まれていません。
- MQMD の *MDMID* フィールドに MINONE が指定されている、または MQPMO 内の *PMOPT* フィールドに *PMNMID* が指定されている。

フラグが必要ない場合は、以下を指定します。

DHFNON

フラグなし。

この定数は、フラグがなにも指定されていないことを示します。DHFNON は、プログラムの文書化を支援するために定義します。この定数は他の定数と組み合わせて使用する実数ではありません。ただし、この定数の値はゼロと等価なので、ほかの実数と組み合わせて使用しても、エラーとして検出されることはありません。

フィールドの初期値は DHFNON です。

DHFMT (8 バイトの文字ストリング)

MQOR および MQPMR レコードの後に続くデータの形式名。

ここでは、MQOD または MQPMR レコード (最後に発生した方) の配列に続くデータの形式名を指定します。

MQPUT または MQPUT1 呼び出しでは、アプリケーションは、このフィールドをデータに適切な値に設定する必要があります。このフィールドのコーディングの規則は、MQMD の *MDFMT* フィールドの場合と同じです。

このフィールドの初期値は FMNONE です。

DHLEN (10 桁の符号付き整数)

MQDH 構造体とそれに続く MQOR および MQPMR レコードを合わせた長さ。

これは、MQDH 構造体の先頭から、MQOR および MQPMR レコードの配列に続くメッセージ・データの先頭までのバイト数です。データの順序は、以下のとおりです。

- MQDH 構造体
- MQOR レコードの配列
- MQPMR レコードの配列
- メッセージ・データ

MQOR および MQPMR レコードの配列は、MQDH 構造体に含まれるオフセットによってアドレッシングされます。これらのオフセットにより、1 つ以上の MQDH 構造体、レコードの配列、およびメッセージ・データの間に未使用のバイトが生じる場合、それらの未使用のバイトを *DHLEN* の値に含める必要がありますが、それらのバイトの内容はキュー・マネージャーによって保持されません。MQPMR レコードの配列は MQOR レコードの配列より先に処理されても構いません。

このフィールドの初期値は 0 です。

DHORO (10 桁の符号付き整数)

最初の MQOR レコードの MQDH の先頭からのオフセット。

このフィールドには、MQOR オブジェクト・レコードの配列内の最初のレコードのオフセットを宛先キューの名前も含めてバイト単位で指定します。この配列には *DHCNT* 個のレコードがあります。これらのレコード (および最初のオブジェクト・レコードと前のフィールドの間でスキップされたバイト数) は、*DHLEN* フィールドで指定された長さに含まれます。

配布リストには常に少なくとも 1 つの宛先が含まれていなければならないため、*DHORO* は常にゼロより大きくなければなりません。

このフィールドの初期値は 0 です。

DHPRF (10 桁の符号付き整数)

どの MQPMR フィールドが存在しているかを示すフラグ。

以下のフラグをいくつか指定できます (または、なにも指定しなくても構いません)。

PFMID

メッセージ ID フィールドがある。

PFCID

相関 ID フィールドがある。

PFGID

グループ ID フィールドがある。

PFFB

フィードバック・フィールドがある。

PFACC

会計トークン・フィールドがある。

MQPMR フィールドがない場合は、以下を指定できます。

PFNONE

書き込みメッセージ・レコードのフィールドがない。

PFNONE は、プログラムの文書化を支援するために定義します。この定数は他の定数と組み合わせて使用するようには意図されていません。ただし、この定数の値はゼロなので、ほかの実数と組み合わせて使用しても、検出されることはありません。

このフィールドの初期値は PFNONE です。

DHPRO (10 桁の符号付き整数)

最初の MQPMR レコードの MQDH の先頭からのオフセット。

このフィールドには、MQPMR 書き込みメッセージ・レコードの配列内の最初のレコードのオフセットをメッセージ・プロパティも含めてバイト単位で指定します。存在する場合、この配列には *DHCNT* 個のレコードがあります。これらのレコード (最初の書き込みメッセージ・レコードと前のフィールドの間でスキップされたバイト数を含む) は、*DHLEN* フィールドに指定された長さで組み込まれます。

書き込みメッセージ・レコードは、オプションです。レコードがなにも提供されていないと、*DHPRO* の値はゼロ、*DHPRF* の値は PFNONE となります。

このフィールドの初期値は 0 です。

DHSID (4 バイトの文字ストリング)

構造体 ID

値は次のものでなければなりません。

DHSIDV

配布ヘッダー構造の ID。

このフィールドの初期値は DHSIDV です。

DHVER (10 桁の符号付き整数)

構造体のバージョン番号。

値は次のものでなければなりません。

DHVER1

配布ヘッダー構造体のバージョン番号。

以下の定数は、現行バージョンのバージョン番号を指定しています。

DHVERC

配布ヘッダー構造体の現行バージョン。

このフィールドの初期値は DHVER1 です。

初期値

表 735. MQDH のフィールド		
フィールド名	定数の名前	定数の値
<i>DHSID</i>	DHSIDV	'DH' followed by two blank characters
<i>DHVER</i>	DHVER1	1
<i>DHLEN</i>	なし	0
<i>DHENC</i>	なし	0
<i>DHCSI</i>	CSUNDF	0
<i>DHFMT</i>	FMNONE	blank
<i>DHFLG</i>	DHFNON	0
<i>DHPRF</i>	PFNONE	0
<i>DHCNT</i>	なし	0
<i>DHORO</i>	なし	0
<i>DHPRO</i>	なし	0

注：

1. 記号 followed by a blank character は、単一の空白文字を表します。

RPG 宣言

```
D*..1.....2.....3.....4.....5.....6.....7..  
D* MQDH Structure  
D*  
D* Structure identifier
```

```

D  DHSID                1      4    INZ('DH ')
D* Structure version number
D  DHVER                 5      8I 0 INZ(1)
D* Length of MQDH structure plusfollowing MQOR and MQPMR records
D  DHLEN                 9     12I 0 INZ(0)
D* Numeric encoding of data that followsthe MQOR and MQPMR records
D  DHENC                13     16I 0 INZ(0)
D* Character set identifier of data thatfollows the MQOR and MQPMR
D* records
D  DHCSI                17     20I 0 INZ(0)
D* Format name of data that follows theMQOR and MQPMR records
D  DHFMT                21     28    INZ(' ')
D* General flags
D  DHFLG                29     32I 0 INZ(0)
D* Flags indicating which MQPMR fieldsare present
D  DHPRF                33     36I 0 INZ(0)
D* Number of MQOR records present
D  DHCNT                37     40I 0 INZ(0)
D* Offset of first MQOR record from startof MQDH
D  DHORO                41     44I 0 INZ(0)
D* Offset of first MQPMR record fromstart of MQDH
D  DHPRO                45     48I 0 INZ(0)

```

IBM i MQDLH (デッドレターヘッダー) on IBM i

概要

目的

MQDLH 構造体は、送達不能 (未配布メッセージ) キューに入っているメッセージのアプリケーション・メッセージ・データの接頭部に付けられた情報を記述します。メッセージは、キュー・マネージャーまたはメッセージ・チャンネル・エージェントにより送達不能キューにリダイレクトされたために、送達不能キューに入れられる場合があります。アプリケーションはメッセージをキューに直接書き込むことができます。

形式名

FMDLH

文字セットとエンコード

MQDLH は、アプリケーション・メッセージ・データの先頭に置かれる場合があります。そうである場合、MQDLH 構造体内のフィールドは、MDCSI フィールドと MDENC フィールドで指定された文字セットとエンコードになります。そうでない場合、文字セットとエンコードは、MQDLH に先行するヘッダー構造体内の MDCSI フィールドと MDENC フィールドによって設定されます。

文字セットは、キュー名に有効な文字用の 1 バイト文字を持つ文字セットでなければなりません。

使用法

送達不能キューに直接メッセージを書き込むアプリケーションは、MQDLH 構造体をメッセージ・データの接頭部に付け、適切な値でフィールドを初期設定する必要があります。ただし、キュー・マネージャーには、MQDLH 構造体が存在する必要や、そのフィールドに対して有効な値が指定されている必要はありません。

メッセージが長すぎて送達不能キューに入りきらない場合、アプリケーションの側で以下の処理のいずれか 1 つを検討する必要があります。

- 送達不能キューに収容できる長さにメッセージ・データを切り捨てる。
- メッセージを補助ストレージに記録して、メッセージが長すぎることを示す例外レポート・メッセージを送達不能キューに入れる。
- メッセージを廃棄して、エラーをその発信元に戻す。メッセージが重大メッセージである場合は、発信元にメッセージのコピーがまだあることが分かっている場合にのみ、メッセージを廃棄します。これは例えば、通信チャンネルからメッセージ・チャンネル・エージェントが受け取ったメッセージなどです。

どちらの選択が適切であるかは、アプリケーションの設計によって異なります。

セグメントになっているメッセージが先頭に MQDLH 構造体を付けて書き込まれている場合、キュー・マネージャーは特殊な処理を実行します。詳細については、MQMDE 構造体の説明を参照してください。

- [1069 ページの『送達不能キューへのメッセージの書き込み』](#)
- [1069 ページの『送達不能キューからのメッセージの読み取り』](#)
- [1069 ページの『フィールド』](#)
- [1073 ページの『初期値』](#)
- [1074 ページの『RPG 宣言』](#)

送達不能キューへのメッセージの書き込み

メッセージを送達不能キューに書き込む場合は、MQPUT または MQPUT1 呼び出しに使用する MQMD 構造体が、メッセージに関連する MQMD と同一である必要があります。MQMD は一般に、MQGET 呼び出しによって返されます。ただし以下の場合を除きます。

- MDCSI および MDENC フィールドは MQDLH 構造体のフィールドで使用される文字セットおよびエンコードに設定する必要があります。
- MDFMT フィールドは FMDLH に設定し、データが MQDLH 構造体で始まることを示す必要があります。
- コンテキスト・フィールド MDACC、MDAID、MDAOD、MDPAN、MDPAT、MDPD、MDPT、および MDUID は、以下のような状況に応じたコンテキスト・オプションを使用して設定する必要があります。
 - 先行するどのメッセージにも関係しないメッセージを送達不能キューに書き込むアプリケーションでは、PMDEFC オプションを使用する必要があります。PMDEFC オプションを使用すると、キュー・マネージャーはメッセージ記述子内のコンテキスト・フィールドをすべてデフォルト値に設定します。
 - 受け取ったメッセージを送達不能キューに書き込むサーバー・アプリケーションでは、元のコンテキスト情報を保存するために、PMPASA オプションを使用する必要があります。
 - 受け取ったメッセージに対する応答を送達不能キューに書き込むサーバー・アプリケーションでは、PMPASI オプションを使用する必要があります。PMPASI を使用すると、識別情報は保存されますが、元の情報はサーバー・アプリケーションの情報に設定されます。
 - 通信チャネルから受け取ったメッセージを送達不能キューに書き込むメッセージ・チャネル・エージェントでは、PMSETA オプションを使用する必要があります。PMSETA オプションは、元のコンテキスト情報を保存します。

MQDLH 構造体自体では、フィールドは以下に示すように設定してください。

- DLCSI、DLENC、および DLFMT フィールドは、MQDLH 構造体が続くデータを説明する値に設定する必要があります。これらの値は一般に、元のメッセージ記述子からの値です。
- DLPAT、DLPAN、DLPD、および DLPT コンテキスト・フィールドには、送達不能キューにメッセージを書き込むアプリケーションに適した値を設定する必要があります。これらの値は元のメッセージには関係しません。
- その他のフィールドは、それぞれ適切な値を設定してください。

アプリケーションは、すべてのフィールドに有効な値が入っており、定義されたフィールド長になるまで文字フィールドにブランクが埋め込まれていることを確認する必要があります。文字データは、定義された長さに満たないままヌル文字を使用して終わらせることはできません。キュー・マネージャーは、MQDLH 構造体内でヌル文字および後続の文字をブランクに変換することはありません。

送達不能キューからのメッセージの読み取り

送達不能キューからメッセージを読み取るアプリケーションは、メッセージが MQDLH 構造体で始まっていることを確認する必要があります。アプリケーションはメッセージ記述子 MQMD 内の MDFMT フィールドを調べることによって、MQDLH 構造体が存在しているかどうかを判別できます。フィールドに値 FMDLH が入る場合、メッセージ・データの先頭は MQDLH 構造体になります。デッド・レター・キューのメッセージは、宛先としていたキューに対して最初から長すぎる場合には切り捨てられる可能性があります。

フィールド

MQDLH 構造には、以下のフィールドが含まれます。フィールドはアルファベット順に説明されています。

DLC SI (10 桁の符号付き整数)

MQDLH に続くデータの文字セット ID。

DLC SI は、MQDLH 構造体に続くデータの文字セット ID を指定します。このデータは一般に元のメッセージからのものです。これは MQDLH 構造体そのものの文字データには適用されません。

MQPUT または MQPUT1 呼び出しでは、アプリケーションは、このフィールドをデータに適切な値に設定する必要があります。以下のような特別な値を使用することができます。

CSINHT

この構造体の文字セット ID を継承する。

この構造体の後に続くデータの文字データは、この構造体に設定されているのと同じ文字セットになります。

キュー・マネージャーは、メッセージで送信される構造体の中のこの値を、構造体の実際の文字セット ID に変更します。エラーが発生しない場合、MQGET 呼び出しから値 CSINHT が戻されることはありません。

MQMD の MDPAT フィールドの値が ATBRKR の場合、CSINHT は使用できません。

このフィールドの初期値は CSUNDF です。

DLD M (48 バイトの文字ストリング)

元の宛先キュー・マネージャーの名前。

これは、メッセージの元の宛先であったキュー・マネージャーの名前です。

このフィールドの長さは LNQM N によって指定されます。このフィールドの初期値は 48 個のブランク文字です。

DLD Q (48 バイトの文字ストリング)

元の宛先キューの名前。

これは、メッセージの元の宛先であったメッセージ・キューの名前です。

このフィールドの長さは LNQN によって指定されます。このフィールドの初期値は 48 個のブランク文字です。

DLE NC (10 桁の符号付き整数)

MQDLH に続くデータの数値エンコード。

DLE NC は、MQDLH 構造体に続くデータの数値エンコードを指定します。このデータは一般に元のメッセージからのものです。これは MQDLH 構造体そのものの数値データには適用されません。

MQPUT または MQPUT1 呼び出しでは、アプリケーションは、このフィールドをデータに適切な値に設定する必要があります。

このフィールドの初期値は 0 です。

DLF MT (8 バイトの文字ストリング)

MQDLH に続くデータの形式名。

ここでは、MQDLH 構造体の後に続くデータ (通常は、元のメッセージのデータ) の形式名を指定します。

MQPUT または MQPUT1 呼び出しでは、アプリケーションは、このフィールドをデータに適切な値に設定する必要があります。このフィールドのコーディング規則は、MQMD 内の MDFMT フィールドの規則と同じです。

このフィールドの長さは LNFMT によって指定されます。このフィールドの初期値は FMNONE です。

DLP AN (28 バイトの文字ストリング)

送達不能 (未配布メッセージ) キューにメッセージを書き込むアプリケーションの名前。

名前の形式は DLPAT フィールドに応じて異なります。1114 ページの『[IBM i での MQMD \(メッセージ記述子\)](#)』にある、MDPAN フィールドに関する説明を参照してください。

キュー・マネージャーがメッセージを送達不能キューにリダイレクトした場合、DLPAN にはキュー・マネージャー名の最初の 28 文字が入ります。名前には必要な場合はブランクが埋め込まれます。

このフィールドの長さは LNPAN によって指定されます。このフィールドの初期値は 28 個のブランク文字です。

DLPAT (10 桁の符号付き整数)

送達不能 (未配布メッセージ) キューにメッセージを書き込むアプリケーションのタイプ。

このフィールドは、メッセージ記述子 MQMD の MDPAT フィールドと同じ意味です (詳細については、1114 ページの『[IBM i での MQMD \(メッセージ記述子\)](#)』を参照してください)。

キュー・マネージャーがメッセージを送達不能キューにリダイレクトした場合、DLPAT の値は ATQM になります。

このフィールドの初期値は 0 です。

DLPD (8 バイトの文字ストリング)

メッセージが送達不能 (未配布メッセージ) キューに書き込まれた日付。

キュー・マネージャーがこのフィールドを生成する際に使用する日付の形式は、以下のとおりです。

• YYYYMMDD

文字は、以下のものを表します。

YYYY

年 (4 桁の数字)

MM

月 (01 から 12 まで)

DD

日 (01 から 31 まで)

DLPD と DLPT フィールドでは、グリニッジ標準時 (GMT) に正確に設定されているシステム・クロックに従って、GMT が使用されます。

このフィールドの長さは LNPDAT によって指定されます。このフィールドの初期値は 8 個のブランク文字です。

DLPT (8 バイトの文字ストリング)

送達不能 (未配布メッセージ) キューにメッセージが書き込まれた時刻。

キュー・マネージャーがこのフィールドを生成する際に使用する時刻の形式は、以下のとおりです。

• HHMMSSSTH

文字は、順に以下のものを表します。

HH

時間 (00 から 23 まで)

MM

分 (00 から 59 まで)

SS

秒 (00 から 59 まで、このトピックの後の注を参照)

T

10 分の 1 秒 (0 から 9 まで)

H

100 分の 1 秒 (0 から 9 まで)

注：システム・クロックが正確な時間標準に同期している場合は、DLPT の秒数として 60 または 61 が戻されることがあります。この余分の秒は、グローバル時間標準にうるう秒が挿入されたときに発生します。

DLPD と DLPT フィールドでは、グリニッジ標準時 (GMT) に正確に設定されているシステム・クロックに従って、GMT が使用されます。

このフィールドの長さは LNPTIM によって指定されます。このフィールドの初期値は 8 個のブランク文字です。

DLREA (10 桁の符号付き整数)

送達不能 (未配布メッセージ) キューに到着した理由メッセージ。

これは、メッセージが、元の宛先キューではなく、送達不能キューに置かれた理由を識別します。これは、FB* または RC* 値 (例えば、RC2053 など) のうちの 1 つでなければなりません。一般に使用可能な FB* 値の詳細については、[1114 ページの『IBM i での MQMD \(メッセージ記述子\)』の MDFB フィールドを参照してください。](#)

値がの範囲 FBIFST 内にある場合、実際 IMS のエラーコードは FBILST フィールド DLREA の値から FBIERR を差し引くことで決定できます。

FB* 値の中には、このフィールドにしか出てこないものもあります。これらの値は、送達不能キューに転送されたリポジトリ・メッセージ、トリガー・メッセージ、または伝送キュー・メッセージに関係しています。これらの値は以下のとおりです。

FBABEG

アプリケーションを開始できません。

トリガー・メッセージを処理するアプリケーションは、トリガー・メッセージの TMAI フィールドに示されているアプリケーションを開始することができませんでした ([1245 ページの『MQTM - トリガー・メッセージ』を参照してください。](#))

FBATYP

アプリケーション・タイプのエラー。

トリガー・メッセージを処理するアプリケーションは、トリガー・メッセージの TMAT フィールドが無効であるためアプリケーションを開始できませんでした ([1245 ページの『MQTM - トリガー・メッセージ』を参照してください。](#))

FBOCD

クラスター受信側チャンネルが削除されています。

メッセージは、FBIERR オプションによって開かれたクラスター・キューを宛先とするクラスター伝送キュー上にありました。宛先キューへメッセージの送信に使用されるリモート・クラスター受信側チャンネルは、メッセージが送信可能になる前に削除されました。FBIERR が指定されているため、メッセージの送信に使用できるのは、キューが開かれたときに選択されたチャンネルのみです。このチャンネルが使用可能でなくなったため、このメッセージは送達不能キューに置かれています。

FBNARM

メッセージはリポジトリ・メッセージではありません。

FBSBCX

メッセージはチャンネル自動定義出口によって停止されました。

FBSBMX

メッセージはチャンネル・メッセージ出口によって停止されました。

FBTM

MQTM 構造体が無効か、または欠落しています。

MQMD の MDFMT フィールドには、FMTM が指定されていますが、メッセージが有効な MQTM 構造体で始まっていません。例えば、TMSID 簡略記号目印は無効である可能性があります。TMVER は認識されない可能性があります。トリガー・メッセージの長さは、MQTM 構造体を収めるために不十分である可能性があります。

FBXQME

伝送キューにあるメッセージが正しい形式ではありません。

メッセージ・チャネル・エージェントは、伝送キューにあるメッセージが正しい形式でないことを検出しました。メッセージ・チャネル・エージェントは、このフィードバック・コードを使用してメッセージを送達不能キューに書き込みます。

このフィールドの初期値は RCNONE です。

DLSID (4 バイトの文字ストリング)

構造体 ID

値は次のものでなければなりません。

DLSIDV

送達不能ヘッダー構造体の ID。

このフィールドの初期値は DLSIDV です。

DLVER (10 桁の符号付き整数)

構造体のバージョン番号。

値は次のものでなければなりません。

DLVER1

送達不能ヘッダー構造体のバージョン番号。

以下の定数は、現行バージョンのバージョン番号を指定しています。

DLVERC

送達不能ヘッダー構造体の現行バージョン。

このフィールドの初期値は DLVER1 です。

初期値

表 736. 次の中のフィールド(F): MQDLH		
フィールド名	定数の名前	定数の値
DLSID	DLSIDV	'DLH-'
DLVER	DLVER1	1
DLREA	RCNONE	0
DLDQ	なし	ブランク
DLDM	なし	ブランク
DLENC	なし	0
DLCSI	CSUNDF	0
DLFMT	FMNONE	ブランク
DLPAT	なし	0
DLPAN	なし	ブランク
DLPD	なし	ブランク
DLPT	なし	ブランク

注:

1. 記号-は、単一のブランク文字を表します。

RPG 宣言

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D* MQDLH Structure
D*
D* Structure identifier
D  DLSID          1          4      INZ('DLH ')
D* Structure version number
D  DLVER          5          8I 0 INZ(1)
D* Reason message arrived on dead-letter(undelivered-message) queue
D  DLREA          9         12I 0 INZ(0)
D* Name of original destination queue
D  DLDQ          13         60      INZ
D* Name of original destination queue manager
D  DLDM          61         108     INZ
D* Numeric encoding of data that followsMQDLH
D  DLENC         109        112I 0 INZ(0)
D* Character set identifier of data thatfollows MQDLH
D  DLCSI         113        116I 0 INZ(0)
D* Format name of data that followsMQDLH
D  DLFMT         117        124     INZ(' ')
D* Type of application that put messageon dead-letter
D* (undelivered-message)queue
D  DLPAT         125        128I 0 INZ(0)
D* Name of application that put messageon dead-letter
D* (undelivered-message)queue
D  DLPAN         129        156     INZ
D* Date when message was put ondead-letter (undelivered-message)queue
D  DLPD          157        164     INZ
D* Time when message was put on thedead-letter (undelivered-message)queue
D  DLPT          165        172     INZ
```

IBM i IBM i での MQDMHO (メッセージ・ハンドル削除オプション)

MQDMHO 構造体を使用すると、アプリケーションで、メッセージ・ハンドルを削除する方法を制御するオプションを指定できます。

概要

目的: この構造体は、**MQDLTMH** 呼び出しの入力パラメーターです。

文字セットとエンコード: **MQDMHO** 内のデータは、アプリケーションの文字セットおよびアプリケーションのエンコードでなければなりません (ENNAT)。

- [1074 ページの『フィールド』](#)
- [1075 ページの『初期値』](#)
- [1075 ページの『RPG 宣言』](#)

フィールド

MQDMHO 構造体には、以下のフィールドが含まれます。フィールドはアルファベット順に説明されています。

DMOPT (10 桁の符号付き整数)

値は次のものでなければなりません。

DMNONE

指定されるオプションはありません。

これは常に入力フィールドです。このフィールドの初期値は **DMNONE** です。

DMSID (10 桁の符号付き整数)

これは構造体 ID です。値は以下のものでなければなりません。

DMSIDV

メッセージ・ハンドル削除オプション構造の ID。

これは常に入力フィールドです。このフィールドの初期値は **DMSIDV** です。

DMVER (10 桁の符号付き整数)

これは構造体のバージョン番号です。値は以下のものでなければなりません。

DMVER1

バージョン 1 のメッセージ・ハンドル削除オプション構造。

以下の定数は、現行バージョンのバージョン番号を指定しています。

DMVERC

メッセージ・ハンドル削除オプション構造の現行バージョン。

これは常に入力フィールドです。このフィールドの初期値は **DMVER1** です。

初期値

表 737. MQDMHO のフィールド		
フィールド名	定数の名前	定数の値
DMSID	DMSIDV	'DMHO'
DMVER	DMVER1	1
DMOPT	DMNONE	0

RPG 宣言

```
D* MQDMHO Structure
D*
D*
D* Structure identifier
D  DMSID          1      4      INZ('DMHO')
D*
D* Structure version number
D  DMVER          5      8I 0 INZ(1)
D*
D* Options that control the action of MQDLTMH
D  DMOPT          9      12I 0 INZ(0)
```

IBM i IBM i での MQDMPO (メッセージ・プロパティ削除オプション)

メッセージ・プロパティ削除オプションを定義する構造。

概要

目的: MQDMPO 構造を使用すると、アプリケーションで、メッセージのプロパティを削除する方法を制御するオプションを指定できます。この構造体は、MQDLTMP 呼び出しの入力パラメーターです。

文字セットとエンコード: MQDMPO 内のデータは、アプリケーションの文字セットおよびアプリケーションのエンコードでなければなりません (ENNAT)。

- [1076 ページの『フィールド』](#)
- [1077 ページの『初期値』](#)
- [1077 ページの『RPG 宣言』](#)

フィールド

MQDMPQ 構造体には、以下のフィールドが含まれます。フィールドはアルファベット順に説明されています。

DPOPT (10 桁の符号付き整数)

メッセージ・プロパティ削除オプション構造体 - DPOPT フィールド。

位置オプション: 以下は、プロパティ・カーソルと比較したプロパティの相対位置に関するオプションです。

DPDELF

指定された名前と一致する最初のプロパティに対して削除します。

DPDELC

プロパティ・カーソルによってポイントされるプロパティ (つまり、IPINQF または IPINQN オプションを使って最後に照会されたプロパティ) を削除します。

プロパティ・カーソルは、メッセージ・ハンドルが再使用されるときにリセットされます。また、MQGET 呼び出しの MQGMO 構造体または MQPUT 呼び出しの MQPMO 構造体の *HMSG* フィールドにメッセージ・ハンドルが指定されている場合にも、リセットされます。

プロパティ・カーソルは、メッセージ・ハンドルが再使用されるときにリセットされます。あるいは MQGET 呼び出しの MQGET 構造体または MQPUT 呼び出しの MQPMO 構造体の *HMSG* フィールドにメッセージ・ハンドルが指定されている場合にも、リセットされます。

プロパティ・ハンドルが未設定の時点で、このオプションを使用すると、呼び出しは失敗して、完了コード CCFAIL および理由コード RC2471 が返されます。また、プロパティ・カーソルによってポイントされるプロパティがすでに削除されている場合にも、これらのコードで失敗します。

これらのオプションがどれも必要でない場合には、以下のオプションを使用できます。

DPNONE

指定されるオプションはありません。

この入力フィールドの初期値は DPDELF です。

DPSID (10 桁の符号付き整数)

メッセージ・プロパティ削除オプション構造体 - DPSID フィールド。

これは構造体 ID です。値は次のものでなければなりません。

DPSIDV

メッセージ・プロパティ削除オプション構造体の ID。

このフィールドは常に入力フィールドです。このフィールドの初期値は DPSIDV です。

DPVER (10 桁の符号付き整数)

メッセージ・プロパティ削除オプション構造体 - DPVER フィールド。

これは構造体のバージョン番号です。値は次のものでなければなりません。

DPVER1

メッセージ・プロパティ削除オプション構造のバージョン番号。

以下の定数は、現行バージョンのバージョン番号を指定しています。

DPVERC

メッセージ・プロパティ削除オプション構造の現行バージョン。

このフィールドは常に入力フィールドです。このフィールドの初期値は DPVER1 です。

初期値

表 738. MQDPMO のフィールド		
フィールド名	定数の名前	定数の値
DPSID	DPSIDV	'DPMO'
DPVER	DPVER1	1
DPOPT	MQDLTMP のアクションを制御するオプション	DPNONE

RPG 宣言

```
D* MQDPMO Structure
D*
D*
D* Structure identifier
D  DPSID              1      4      INZ('DPMO')
D*
D* Structure version number
D  DPVER              5      8I 0 INZ(1)
D*
** Options that control the action of
D* MQDLTMP
D  DPOPT              9      12I 0 INZ(0)
```

IBM i IBM i での MQEPH (組み込み PCF ヘッダー)

概要

目的

MQEPH 構造体は、メッセージがプログラマブル・コマンド・フォーマット (PCF) メッセージである場合に、そのメッセージ内の追加データを記述します。EPPFH フィールドでは、この構造体続く PCF パラメーターを定義します。これにより、PCF メッセージ・データの後に他のヘッダーを続けることができます。

形式名

EPFMT

文字セットとエンコード

MQEPH のデータは、ローカル・キュー・マネージャーの文字セットおよびエンコードになっていなければなりません。これは **CCSID** キュー・マネージャー属性で指定します。

MQEPH の文字セットおよびエンコードは、以下の **MDCSI** および **MDENC** フィールドで設定します。

- MQMD (MQEPH 構造体がメッセージ・データの開始点にある場合)
- MQEPH 構造体に先行するヘッダー構造体 (上記以外の場合)

使用法

コマンドをコマンド・サーバーまたは PCF を受け入れるその他のキュー・マネージャーのサーバーに送信するために MQEPH 構造体を使用することはできません。

同様に、コマンド・サーバーまたは PCF を受け入れるその他のキュー・マネージャーのサーバーでも、MQEPH 構造体を含む応答やイベントは生成されません。

- [1078 ページの『フィールド』](#)
- [1079 ページの『初期値』](#)
- [1080 ページの『RPG 宣言』](#)

フィールド

MQEPH 構造体には、以下のフィールドが含まれます。フィールドはアルファベット順に説明されています。

EPCSI (10 桁の符号付き整数)

これは、MQEPH 構造体とそれに関連する PCF パラメーターに続くデータの文字セット ID です。これは、MQEPH 構造体自体の文字データには適用されません。

このフィールドの初期値は EPCUND です。

EPENC (10 桁の符号付き整数)

これは、MQEPH 構造体とそれに関連する PCF パラメーターに続くデータの数値エンコードです。これは、MQEPH 構造体自体の文字データには適用されません。

このフィールドの初期値は 0 です。

EPFLG (10 桁の符号付き整数)

以下の値を使用できます。

EPNONE

フラグは指定されていません。MDCSIEPNONE は、プログラムの文書化を支援するために定義されています。この定数は他の定数と組み合わせて使用するには意図されていません。ただし、この定数の値はゼロなので、ほかの実数と組み合わせて使用しても、検出されることはありません。

EPCSEM

文字データを含むパラメーターの文字セットは、各構造体の CCSID フィールド内で個別に指定されます。EPSID および EPFMT フィールドの文字セットは、MQEPH 構造体の前に置かれるヘッダー構造体の CCSID で定義されます。MQEPH がメッセージの先頭にある場合は、MQMD の MDCSI フィールドで定義されます。

このフィールドの初期値は EPNONE です。

EPFMT (8 バイトの文字ストリング)

これは、MQEPH 構造体とそれに関連する PCF パラメーターに続くデータの形式名です。

このフィールドの初期値は EPFMNO です。

EPLEN (10 桁の符号付き整数)

これは、次のヘッダー構造体の前に置かれるデータ量です。これには以下のものが含まれます。

- MQEPH ヘッダーの長さ
- ヘッダーのあとに続くすべての PCF パラメーターの長さ
- それらのパラメーターのあとに続くブランクによる埋め込み

EPLEN は、4 の倍数でなければなりません。

構造体の固定長部分は EPSTLF で定義されます。

このフィールドの初期値は 68 です。

EPPCFH (MQCFH)

これはプログラマブル・コマンド・フォーマット (PCF) ヘッダーで、MQEPH 構造体の後に続く PCF パラメーターを定義します。これにより、ヘッダーの異なる PCF メッセージ・データを続けることが可能になります。

PCF ヘッダーは、初期の状態では以下の値が定義されています。

表 739. EPPCFH のフィールド		
フィールド名	定数の名前	定数の値
EP3TYP	CFTNON	0

表 739. EPPCFH のフィールド (続き)		
フィールド名	定数の名前	定数の値
EP3LEN	FHLENV	36
EP3VER	FHVER3	3
EP3CMD	CMNONE	0
EP3SEQ	なし	1
EP3CTL	CFCLST	1
EEP3CC	CCOK	0
EP3REA	RCNONE	0
EP3CNT	なし	0

アプリケーションでは、EP3TYP の CFTNON を、組み込み PCF ヘッダーを使用する方法に合わせて有効な構造体タイプに変更する必要があります。

EPSID (4 バイトの文字ストリング)

値は次のものでなければなりません。

EPSTID

組み込み PCF ヘッダー構造体の ID。

このフィールドの初期値は EPSTID です。

EPVER (10 桁の符号付き整数)

値は次のいずれかです。

EPVER1

組み込み PCF ヘッダー構造のバージョン番号。

以下の定数は、現行バージョンのバージョン番号を指定しています。

EPVER3

組み込み PCF ヘッダー構造体の現行バージョン。

このフィールドの初期値は EPVER3 です。

初期値

表 740. MQEPH のフィールド		
フィールド名	定数の名前	定数の値
EPSID	EPSTID	'EP---'
EPVER	EPVER1	1
EPLEN	EPSTLF	68
EPENC	なし	0
EPCSI	EPCUND	0
EPFMT	EPFMNO	ブランク
EPFLG	EPNONE	0
EPPCFH	1078 ページの表 739 で定義されている名前と値	0

注:

1. 記号~は、単一のブランク文字を表します。

RPG 宣言

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..  
D* MQEPH Structure  
D*  
D* Structure identifier  
D EPSID 1 4  
D* Structure version number  
D EPVER 5 8I 0  
D* Total length of MQEPH including MQCFHand parameter structures  
D* that follow  
D EPLEN 9 12I 0  
D* Numeric encoding of data that follows last PCF parameter structure  
D EPENC 13 16I 0  
D* Character set identifier of data that follows last PCF parameter  
D* structure  
D EPCSI 17 20I 0  
D* Format name of data that follows last PCF parameter structure  
D EPFMT 21 28  
D* Flags  
D EPFLG 29 32I 0  
D* Programmable Command Format Header  
D EP3TYP 33 36I 0  
D EP3LEN 37 40I 0  
D EP3VER 41 44I 0  
D EP3CMD 45 48I 0  
D EP3SEQ 49 52I 0  
D EP3CTL 53 56I 0  
D EP3CC 57 60I 0  
D EP3REA 61 64I 0  
D EP3CNT 65 68I 0
```

IBM i IBM i での MQGMO (読み取りメッセージ・オプション)

MQGMO 構造体を使用すると、アプリケーションで、メッセージをキューから除去する方法を制御するオプションを指定できます。

概要

目的

この構造体は、MQGET 呼び出しの入出力パラメーターです。

バージョン

MQGMO の現行バージョンは GMVER4 です。これより新しいバージョンの構造体にもみ存在するフィールドについては、そのフィールドの説明にその旨を記載しています。

提供されている COPY ファイルには、環境でサポートされている最新バージョンの MQGMO が含まれていますが、GMVER フィールドの初期値は GMVER1 に設定されています。version-1 構造体に存在しないフィールドを使用するには、アプリケーションで、GMVER フィールドを必要なバージョンのバージョン番号に設定する必要があります。

文字セットとエンコード

MQGMO のデータは、CodedCharSetId キュー・マネージャー属性で指定された文字セットと、ENNAT で指定されたローカル・キュー・マネージャーのエンコードで記述されていなければなりません。ただし、アプリケーションが IBM MQ クライアントとして実行されている場合、構造体はクライアントの文字セットとエンコードに従っている必要があります。

- [1081 ページの『フィールド』](#)
- [1101 ページの『初期値』](#)
- [1101 ページの『RPG 宣言』](#)

フィールド

MQGMO 構造体には、以下のフィールドが含まれます。フィールドはアルファベット順に説明されています。

GMGST (1 バイトの文字ストリング)

取り出されたメッセージが 1 つのグループに属しているかどうかを示すフラグ。

以下の値がどれか 1 つ含まれています。

GSNIG

メッセージは 1 つのグループに属していない。

GSMIG

メッセージは 1 つのグループに属しているが、グループの最後にあるものではない。

GSLMIG

メッセージはグループの最後にあるものである。

この値は、グループに属しているメッセージが 1 つしかない場合にも戻されます。

このフィールドは出力フィールドです。このフィールドの初期値は GSNIG です。GMVER が GMVER2 より小さい場合、このフィールドは無視されます。

GMMH (10 桁の符号付き整数)

メッセージ・ハンドル

GMPPRAQ オプションを指定し、PRPCTL キュー属性が PRPRFH に設定されていない場合は、これはキューから取り出されるメッセージのプロパティにデータを追加する、メッセージに対するハンドルです。このハンドルは、MQCRTMH 呼び出しによって作成されます。ハンドルに既に関連付けられているプロパティは、メッセージを取り出す前にすべてクリアされます。

以下のような値も指定することができます。

MQHM_NONE

メッセージ・ハンドルは提供されません。

MQGET 呼び出し上でメッセージ記述子は必須ではありません。有効なメッセージ・ハンドルが提供され、出力上でメッセージ・プロパティを組み込むのに使用される場合は、メッセージ・ハンドルに関連付けられているメッセージ記述子が入力フィールド用に使用されます。

メッセージ記述子を MQGET 呼び出しに指定すると、メッセージ・ハンドルに関連付けられているメッセージ記述子より常に優先します。

GMPPRRF を指定する場合か、GMPPRAQ を指定して PRPCTL キュー属性が PRPRFH の場合は、メッセージ記述子パラメーターが指定されていなければ呼び出しは理由コード RC2026 で失敗します。

MQGET 呼び出しから戻る際に、このメッセージ・ハンドルに関連付けられているプロパティとメッセージ記述子は更新され(メッセージ記述子が MQGET 呼び出し上で指定されている場合はメッセージ記述子も)、取り出されたメッセージの状態を反映します。その後、MQINQMP 呼び出しを使用して、メッセージのプロパティを照会できます。

拡張メッセージ記述子(ある場合)を除いて、MQINQMP 呼び出しを使用して照会できるプロパティはメッセージ・データには含まれません。キュー上のメッセージがメッセージ・データ中のプロパティに含まれていた場合は、データがアプリケーションに戻る前にメッセージ・データから除去されます。

メッセージ・ハンドルが提供されていないか、バージョンが GMVER4 より前の場合は、MQGET 呼び出し上で有効なメッセージ記述子を提供しなければなりません。メッセージ・プロパティ(メッセージ記述子に含まれているプロパティを除く)は、MQGMO 構造および PRPCTL キュー属性中のプロパティ・オプションの値を対象としたメッセージ・データ中に戻されます。

このフィールドは、常に入力フィールドです。このフィールドの初期値は HMNONE です。GMVER が GMVER4 より小さい場合、このフィールドは無視されます。

GMMO (10 桁の符号付き整数)

MQGET に使用される選択基準を制御するオプション。

これらのオプションにより、アプリケーションは、MQGET 呼び出しによって戻されるメッセージを選択するために使用する **MSGDSC** パラメーター内のフィールドを選択することができます。アプリケーションは、このフィールドに必要なオプションを設定してから、**MSGDSC** パラメーター内の対応するフィールドを、それらのフィールドに必要な値に設定します。MQMD にこれらの値が入っているメッセージのみが、MQGET 呼び出しで **MSGDSC** パラメーターを使用した場合の検索の候補になります。戻すメッセージの選択時には、対応した一致オプションが指定されていないフィールドは無視されます。MQGET 呼び出しで選択基準を使用しない場合（つまり、任意のメッセージを受け入れることができる場合）は、**GMMO** を **MONONE** に設定する必要があります。

GMLOGO を指定すると、特定のメッセージのみが次の MQGET 呼び出しで戻されます。

- 現行のグループまたは論理メッセージがない場合は、**MDSEQ** が 1 に等しく、**MDOFF** が 0 に等しいメッセージのみが戻されます。この場合に、以下のオプションを 1 つ以上使用すると、戻されるメッセージのうちのどれが戻されるかを選定できます。
 - **MOMSGI**
 - **MOCORI**
 - **MOGRPI**
- 現行のグループまたは論理メッセージが存在している場合は、グループ内の次のメッセージまたは論理メッセージ内の次のセグメントのみが戻されます。これは、**MO*** オプションを指定しても変更できません。

どちらの場合も、該当しないマッチング・オプションを指定できますが、**MSGDSC** パラメーター内の関連フィールドの値は、返されるメッセージ内の対応するフィールドの値と一致する必要があります。呼び出しは失敗し、理由コード **RC2247** が返されます。

GMMUC または **GMBRWC** のいずれかが指定されている場合、**GMMO** は無視されます。

次のオプションのうちの 1 つ以上を指定できます。

MOMSGI

指定されたメッセージ ID を持つメッセージを取り出します。

このオプションは、取り出されるメッセージのメッセージ ID が MQGET 呼び出しの **MSGDSC** パラメーターの **MDMID** フィールドの値と一致していなければならないことを指定します。これは、適用される他の一致（例えば、関連 ID）に加えて一致している必要があります。

このオプションを指定しない場合、**MSGDSC** パラメーターの **MDMID** フィールドは無視され、すべてのメッセージ ID が一致します。

注: **MINONE** は、メッセージの MQMD 内のどのメッセージ ID とも一致する特殊なメッセージ ID です。したがって、**MOMSGI** を **MINONE** と組み合わせて指定しても、**MOMSGI** を指定しなくても同じことです。

MOCORI

指定された関連 ID を持つメッセージを取り出します。

このオプションは、取り出されるメッセージの関連 ID が MQGET 呼び出しの **MSGDSC** パラメーター内の **MDCID** フィールドの値と一致していなければならないことを指定します。これは、適用される他の一致（例えば、メッセージ ID）に加えて一致している必要があります。

このオプションを指定しない場合、**MSGDSC** パラメーターの **MDCID** フィールドは無視され、すべての関連 ID が一致します。

注: **CINONE** は、メッセージの MQMD 内のどの関連 ID とも一致する特殊な関連 ID です。したがって、**MOCORI** を **CINONE** と組み合わせて指定しても、**MOCORI** を指定しなくても同じことです。

MOGRPI

指定されたグループ ID を持つメッセージを取り出します。

このオプションは、取り出されるメッセージのグループ ID が MQGET 呼び出しの **MSGDSC** パラメーターの **MDGID** フィールドの値と一致していなければならないことを指定します。これは、適用される他の一致（例えば、関連 ID）に加えて一致している必要があります。

このオプションを指定しない場合、**MSGDSC** パラメーターの *MDGID* フィールドは無視され、すべてのグループ ID が一致します。

注: GINONE は、メッセージの MQMD 内のどのグループ ID とも一致する特殊なグループ ID です。したがって、MOGRPI を GINONE と組み合わせて指定しても、MOGRPI を指定しなくても同じことです。

MOSEQN

指定されたメッセージ順序番号を持つメッセージを取り出します。

このオプションは、取り出されるメッセージのメッセージ・シーケンス番号が、MQGET 呼び出しの **MSGDSC** パラメーターの *MDSEQ* フィールドの値と一致している必要があることを指定します。これは、適用される他の一致 (例えば、グループ ID) に加えて一致している必要があります。

このオプションを指定しない場合、**MSGDSC** パラメーターの *MDSEQ* フィールドは無視され、すべてのメッセージ・シーケンス番号が一致します。

MOOFFS

指定されたオフセットを持つメッセージを取り出します。

このオプションは、取り出されるメッセージのオフセットが MQGET 呼び出しの **MSGDSC** パラメーターの *MDOFF* フィールドの値と一致していなければならないことを指定します。これは、適用される他の一致 (例えば、メッセージ・シーケンス番号) に加えて一致している必要があります。

このオプションを指定しない場合、**MSGDSC** パラメーターの *MDOFF* フィールドは無視され、すべてのオフセットが一致します。

説明されたオプションを指定しない場合、以下のオプションが使用できます。

MONONE

一致なし。

このオプションを指定すると、戻すメッセージの選択時にどの一致も使用されません。したがって、キューに入っているメッセージがすべて取り出せるようになります (ただし、GMAMSA、GMASGA、および GMCMPM オプションで制御できます)。

MONONE は、プログラム文書化を支援するために定義されています。このオプションは、他の MO* オプションと組み合わせて使用するオプションではありません。ただし、このオプションの値はゼロと等価なため、他のオプションと組み合わせて使用しても、エラーとして検出されることはありません。

このフィールドは入力フィールドです。このフィールドの初期値は、MOMSGI と MOCORI を組み合わせたものです。GMVER が GMVER2 より小さい場合、このフィールドは無視されます。

注: GMMO フィールドの初期値は、以前のバージョンのキュー・マネージャーとの互換性のために定義されています。ただし、選択基準を使用せずにキューから一連のメッセージを読み取る場合、この初期値では、各 MQGET 呼び出しの前に、アプリケーションが *MDMID* および *MDCID* フィールドを MINONE および CINONE にリセットする必要があります。MDMID および MDCID をリセットする必要性を回避するには、GMVER を GMVER2 に設定し、GMMO を MONONE に設定します。

GMOPT (10 桁の符号付き整数)

MQGET のアクションを制御するオプション。

下記のオプションをいくつか指定できます (または、なにも指定しなくても構いません)。2 つ以上指定が必要な場合は、それらの値を加算します (同じ定数を複数回加算しないでください)。無効なオプションの組み合わせについては注記されています。それ以外の組み合わせは有効です。

待機オプション: 以下のオプションは、メッセージがキューに到着するまでの待機に関連するオプションです。

GMWT

メッセージが到着するのを待機します。

アプリケーションは、適切なメッセージが到着するまで待機します。アプリケーションが待機する最大時間は、GMWI に指定されています。

MQGET 要求が禁止されているか、MQGET 要求が待機中に禁止になる場合は、キューに適切なメッセージがあるかどうかにかかわらず、待機状態は取り消され、呼び出しは CCFAIL で完了し、理由コード RC2016 が戻ります。

このオプションは、GMBRWF オプションまたは GMBRWN オプションと共に使用することができます。

複数のアプリケーションが同じ共有キューで待機している場合、該当するメッセージが到着した時に活動化されるアプリケーション (1 つ以上) については、このセクションで後述します。

注: 以下の説明では、ブラウズ MQGET 呼び出しは、ブラウズ・オプションの 1 つを指定する呼び出しですが、GMLK ではありません。GMLK オプションを指定する MQGET 呼び出しは、非ブラウズ呼び出しとして扱われます。

- 1 つ以上の非ブラウズの MQGET 呼び出しが待機しているが、ブラウズの MQGET 呼び出しは待機していない場合は、1 つが活動化されます。
- 1 つ以上のブラウズの MQGET 呼び出しが待機しているが、非ブラウズの MQGET 呼び出しは待機していないという場合は、すべてが活動化されます。
- 1 つ以上の非ブラウズの MQGET 呼び出し、および 1 つ以上のブラウズの MQGET 呼び出しが待機しているという場合は、1 つの非ブラウズの MQGET 呼び出しが活動化されますが、ブラウズの MQGET 呼び出しについては、一切活動化されない、一部活動化される、あるいはすべてが活動化されるかのいずれかになります。(活動化されるブラウズの MQGET 呼び出しの数は、オペレーティング・システムのスケジューリングの考慮事項と、その他の要素によって決まるため、予測することができません。)

複数の非ブラウズの MQGET 呼び出しが同じキューで待機している場合は、1 つのみが活動化されます。このような状況では、キュー・マネージャーは、待機中の非ブラウズ呼び出しに対して以下のような順序で優先順位を与えます。

1. 特定のメッセージによってのみ満たすことができる特定の get-wait 要求。例えば、特定の *MDMID* または *MDCID* (あるいはその両方) を持つ要求。
2. どのメッセージによっても満たすことができる一般的な get-wait 要求。

以下の点に注意してください。

- 最初のカテゴリでは、より具体的な get-wait 要求 (例えば、*MDMID* と *MDCID* の両方を指定する要求) に追加の優先順位は与えられません。
- いずれのカテゴリでも、どのアプリケーションが選択されるか予測はできません。特に、待機時間が長いアプリケーションから選択されるとは限りません。
- オペレーティング・システムのパス長および優先順位スケジューリングが考慮された結果、待機中のアプリケーションのうち、予期された優先順位より低いオペレーティング・システムの優先順位を持つアプリケーションがメッセージを取得する場合があります。
- また、待機中でないアプリケーションが、待機中のアプリケーションより先にメッセージを取り出すこともあります。

GMBRWC または GMMUC を指定した場合、GMWT は無視され、エラーは発生しません。

GMNWT

適切なメッセージがなければ、ただちに戻ります。

適切なメッセージを使用できない場合、アプリケーションは待機しません。これは、GMWT オプションの反対で、プログラム文書化を支援するために定義されています。いずれも指定されていないときは、これがデフォルト値になります。

GMFIQ

キュー・マネージャーが静止している場合は、失敗します。

このオプションを指定すると、キュー・マネージャーが静止状態にある場合、MQGET 呼び出しが強制的に失敗します。

このオプションを GMWT と共に指定し、かつキュー・マネージャーが静止状態になった時点で待機状態が未解決である場合、

- ・待機状態は取り消され、呼び出しは理由コード RC2161 と共に完了コード CCFAIL を戻します。

GMFIQ が指定されておらず、キュー・マネージャーが静止状態に入った場合には、待機状態は取り消されません。

同期点オプション: 以下のオプションは、作業単位内での MQGET 呼び出しの実行に関連したオプションです。

GMSYP

同期点制御を使用してメッセージを読み取ります。

この要求は、通常の作業単位プロトコルの中で操作することです。メッセージには、他のアプリケーションでは使用できないものとしてマークが付けられますが、作業単位がコミットされたときにのみ、キューから削除されます。作業単位がバックアウトされると、メッセージは再び使用可能になります。

このオプションまたは GMNSYP が指定されていない場合、読み取り要求は作業単位内にありません。

このオプションと組み合わせて使用できないオプションは、以下のとおりです。

- ・ GMBRWF
- ・ GMBRWC
- ・ GMBRWN
- ・ GMLK
- ・ GMNSYP
- ・ GMPSYP
- ・ GMUNLK

GMPSYP

メッセージが持続する場合、同期点制御を使用してメッセージを読み取ります。

この要求は、取り出されたメッセージが持続する場合に限り、標準の作業単位プロトコル内で機能します。持続メッセージには、MQMD の *MDPER* フィールドの値 PEPER があります。

- ・メッセージが持続的である場合、キュー・マネージャーは、アプリケーションで GMSYP が指定されている場合と同じように呼び出しを処理します。
- ・メッセージが持続しない場合、キュー・マネージャーは、アプリケーションで GMNSYP (以下のセクションを参照) が指定されている場合と同じように呼び出しを処理します。

このオプションと組み合わせて使用できないオプションは、以下のとおりです。

- ・ GMBRWF
- ・ GMBRWC
- ・ GMBRWN
- ・ GMCMPM
- ・ GMNSYP
- ・ GMSYP
- ・ GMUNLK

GMNSYP

同期点制御なしでメッセージを取得します。

この要求は、通常の作業単位プロトコルの外部で動作することになります。メッセージは即時にキューから削除されます (ブラウザー要求の場合を除く)。作業単位をバックアウトすることによって、メッセージを再度使用可能にすることはできません。

GMBRWF または GMBRWN が指定されている場合は、このオプションが想定されます。

このオプションと GMSYP が指定されていない場合、取得要求は作業単位内にはありません。

このオプションと組み合わせて使用できないオプションは、以下のとおりです。

- GMSYP
- GMPSYP

ブラウズ・オプション: 以下のオプションは、キュー上のメッセージのブラウズに関連したオプションです。

GMBRWF

キューの先頭からブラウズします。

OOBRW オプションを指定してキューを開くと、キュー上の最初のメッセージの前に、ブラウズ・カーソルが置かれ、論理的に位置付けられます。GMBRWF、GMBRWN、またはGMBRWC オプションを指定して後続の MQGET 呼び出しを使用すると、キューからメッセージを非破壊的に取り出すことができます。ブラウズ・カーソルは、キュー上のメッセージ内の位置にマークを付け、GMBRWN の次の MQGET 呼び出しで適切なメッセージが検索されるようにします。

GMBRWF を指定した MQGET 呼び出しは、ブラウズ・カーソルの直前の位置が無視されます。メッセージ記述子に指定された条件を満たすキュー上の最初のメッセージが検索されます。メッセージはキューに残り、ブラウズ・カーソルはこのメッセージの上に置かれます。

この呼び出しの後、ブラウズ・カーソルは、戻されたメッセージ上に置かれます。GMBRWN を指定した次の MQGET 呼び出しの前にメッセージがキューから除去された場合、ブラウズ・カーソルは、メッセージが占めていたキュー内の位置にあります。ただし、その位置が空になっている場合でも、その位置には変わりません。

次に、GMMUC オプションを非ブラウズの MQGET 呼び出しで使用して、キューからメッセージを除去することができます。

ブラウズ・カーソルは、同じ *HOB*J ハンドルを使用する非ブラウズ MQGET 呼び出しによって移動されません。また、完了コード *CC*FAIL または理由コード *RC*2080 を戻すブラウズ MQGET 呼び出しによって移動することはありません。

GMLK オプションをこのオプションと一緒に指定すると、ブラウズされたメッセージがロックされるようになります。

GMBRWF は、グループ内のメッセージの処理および論理メッセージのセグメントを制御する GM* および MO* オプションと、任意の有効な組み合わせを使用して指定することができます。

GMLOGO が指定されている場合、メッセージは論理順序でブラウズされます。このオプションを省略すると、メッセージは物理的な順序で参照されます。GMBRWF を指定すると、論理順序と物理順序を切り替えることができますが、後続の GMBRWN を使用する MQGET 呼び出しでは、GMBRWF を指定した最新のキュー・ハンドルと同じ順序でキューをブラウズする必要があります。

キュー・マネージャーがキュー上のメッセージをブラウズする MQGET 呼び出し用に保持するグループおよびセグメント情報は、キューからメッセージを除去する MQGET 呼び出しのためにキュー・マネージャーが保持するグループとセグメント情報とは分離されます。GMBRWF を指定した場合、キューマネージャはブラウズのためのグループとセグメント情報を無視して、あたかも現在のグループと現在の論理メッセージがないかのようにキューをスキャンします。MQGET 呼び出しが正常に行われた場合 (完了コード *CC*OK または *CC*WARN)、ブラウズのためのグループおよびセグメントの情報は、戻されたメッセージの情報に設定されます。呼び出しが失敗した場合、グループおよびセグメント情報は、呼び出しの前と同じままです。

このオプションと組み合わせて使用できないオプションは、以下のとおりです。

- GMBRWC
- GMBRWN
- GMMUC
- GMSYP
- GMPSYP
- GMUNLK

キューがブラウズ用にオープンされていなかった場合もエラーになります。

GMBRWN

キューの現在位置からブラウズします。

ブラウズ・カーソルは、MQGET 呼び出しで指定された選択基準を満たすキュー上の次のメッセージに進みます。メッセージはアプリケーションに戻されますが、キューに残ります。

ブラウズのためにキューがオープンされた後、ハンドルを使用する最初のブラウズ呼び出しは、GMBRWF オプションまたは GMBRWN オプションを指定しているかどうかに関係なく、同じ効果があります。

GMBRWN を指定した次の MQGET 呼び出しが発行される前にメッセージがキューから除去された場合、ブラウズ・カーソルは、メッセージが占めていたキュー内の位置が空であっても、論理的にその位置に留まります。

メッセージは、以下の 2 つの方法のいずれかによってキューに入れられます。

- 優先順位による FIFO (MSPRIO)
- 優先順位に関係しない FIFO (MSFIFO)

MsgDeliverySequence キュー属性は、適用されるメソッドを示します (詳しくは、[1380 ページ](#)の『[キューの属性](#)』を参照してください)。

キューの *MsgDeliverySequence* が MSPRIO であり、ブラウズ・カーソルによって現在指し示されているものより高い優先順位のメッセージがキューに到着した場合、そのメッセージは GMBRWN を使用するキューの現行スリーブ中には検出されません。そのメッセージは、GMBRWF で (またはキューの再オープンによって) ブラウズ・カーソルをリセットしないと、検出されません。

その後、必要に応じて非ブラウズ MQGET 呼び出しで GMMUC オプションを使用することにより、キューからメッセージを削除することができます。

ブラウズ・カーソルは、同じ *HOB*J ハンドルを使用する非ブラウズ MQGET 呼び出しによって移動されません。

GMLK オプションをこのオプションと一緒に指定すると、ブラウズされたメッセージがロックされるようになります。

GMBRWN は、論理メッセージのグループおよびセグメントでのメッセージ処理を制御する GM* および MO* オプションの有効な任意の組み合わせで指定することができます。

GMLOGO が指定されている場合、メッセージは論理順序でブラウズされます。このオプションを省略すると、メッセージは物理的な順序で参照されます。GMBRWF を指定すると、論理順序と物理順序を切り替えることができますが、後続の GMBRWN を使用する MQGET 呼び出しでは、GMBRWF を指定した最新のキュー・ハンドルと同じ順序でキューをブラウズする必要があります。この条件が満たされないと、呼び出しは失敗し、理由コード RC2259 が戻ります。

注: GMLOGO が指定されていない場合、MQGET 呼び出しを使用してメッセージ・グループ (またはグループ内にない論理メッセージ) の末尾以降をブラウズする場合には、特別な注意が必要です。例えば、グループ内の最後のメッセージが、キュー上のグループの最初のメッセージの前にある場合、GMBRWN を使用してグループの終わりを超えてブラウズし、MDSEQ を 1 に設定して (次のグループの最初のメッセージを見つけるために) MOSEQN を指定すると、既にブラウズされているグループの最初のメッセージが再び戻されます。これは、即時に発生する可能性があります。あるいは、(介入グループがある場合) 何度かの MQGET 呼び出し後に発生するかもしれません。

無限ループの発生を避けるには、ブラウズの際にキューを 2 回オープンして次のように操作します。

- 各グループの最初のメッセージだけをブラウズするには、最初のハンドルを使用します。
- 特定のグループのメッセージだけをブラウズするには、2 番目のハンドルを使用します。
- MO* オプションを使用して、グループ内のメッセージをブラウズする前に、2 番目のブラウズ・カーソルを最初のブラウズ・カーソルの位置に移動する。
- GMBRWN を使用しないで、1 つのグループが終了した後ろをブラウズする。

MQGET 呼び出しでキューに入っているメッセージをブラウズするためにキュー・マネージャーが保持しているグループおよびセグメント情報は、MQGET 呼び出しでキューからメッセージを除去するために保持しているグループおよびセグメント情報とは異なります。

このオプションと組み合わせて使用できないオプションは、以下のとおりです。

- GMBRWF
- GMBRWC
- GMMUC
- GMSYP
- GMPSYP
- GMUNLK

キューがブラウズ用にオープンされていなかった場合もエラーになります。

GMBRWC

ブラウズ・カーソルの下のメッセージをブラウズします。

このオプションを指定すると、ブラウズ・カーソルが指すメッセージは、MQGMO の *GMMO* フィールドに指定された MO* オプションに関係なく、非破壊的に取り出されます。

ブラウズ・カーソルが指すメッセージは、GMBRWF または GMBRWN のいずれかのオプションを使用して最後に検索されたものです。このキューがオープンされてから、このキューに対しこれらの呼び出しのいずれも発行されなかった場合、またはブラウズ・カーソルが指すメッセージが破壊されて取り出された場合、呼び出しは失敗します。

ブラウズ・カーソルの位置は、この呼び出しでは変更されません。

次に、GMMUC オプションを非ブラウズの MQGET 呼び出しで使用して、キューからメッセージを除去することができます。

ブラウズ・カーソルは、同じ *HOB*J ハンドルを使用する非ブラウズ MQGET 呼び出しによって移動されません。また、完了コード CCFAIL または理由コード RC2080 を戻すブラウズ MQGET 呼び出しによって移動することはありません。

GMBRWC が GMLK と共に指定される場合

- すでにロックされているメッセージは、カーソルの下になければなりません。この場合、ロック解除や再ロックをしないで戻されます。メッセージは、ロックされたままです。
- ロックされたメッセージがないときは、ブラウズ・カーソルの下にあるメッセージはロックされ、アプリケーションに戻されます。ブラウズ・カーソルの下にメッセージがなければ、呼び出しは失敗します。

GMBRWC が GMLK なしで指定される場合

- 既にロックされているメッセージがある場合、そのメッセージがカーソルの下になければなりません。このメッセージはアプリケーションに戻され、その後ロック解除されます。メッセージは現在アンロックされているため、メッセージを再度ブラウズすることができない場合や、破壊取り出しができない場合があります (キューからメッセージを読み込む別のアプリケーションなら破壊取り出しができます)。
- ロックされたメッセージがないときは、ブラウズ・カーソルの下にあるメッセージがアプリケーションに戻されます。ブラウズ・カーソルの下にメッセージが何もない場合は、呼び出しは失敗に終わります。

GMCMPPM が GMBRWC と共に指定されている場合、ブラウズ・カーソルは MQMD 内の *MDOFF* フィールドがゼロのメッセージを識別する必要があります。この条件を満たさないと、この呼び出しは失敗し、理由コード RC2246 が戻ります。

MQGET 呼び出しでキューに入っているメッセージをブラウズするためにキュー・マネージャーが保持しているグループおよびセグメント情報は、MQGET 呼び出しでキューからメッセージを除去するために保持しているグループおよびセグメント情報とは異なります。

このオプションと組み合わせて使用できないオプションは、以下のとおりです。

- GMBRWF
- GMBRWN
- GMMUC
- GMSYP
- GMPSYP
- GMUNLK

キューがブラウザ用にオープンされていなかった場合もエラーになります。

GMMUC

ブラウザ・カーソルが置かれているメッセージを読み取ります。

このオプションを指定すると、MQGMO の *GMMO* フィールドに指定された MO* オプションに関係なく、ブラウザ・カーソルが指すメッセージが取り出されます。メッセージはキューから削除されません。

ブラウザ・カーソルが指すメッセージは、GMBRWF または GMBRWN のいずれかのオプションを使用して最後に検索されたものです。

GMCMPPM が GMMUC とともに指定されている場合、ブラウザ・カーソルは、MQMD 内の *MDOFF* フィールドがゼロであるメッセージを識別する必要があります。この条件を満たさないと、この呼び出しは失敗し、理由コード RC2246 が戻ります。

このオプションと組み合わせて使用できないオプションは、以下のとおりです。

- GMBRWF
- GMBRWC
- GMBRWN
- GMUNLK

キューがブラウザ用および入力用にオープンされていなかった場合もエラーになります。ブラウザ・カーソルが取り出し可能なメッセージを現在指し示していない場合は、エラーが MQGET 呼び出しで戻されます。

ロック・オプション: 以下のオプションは、キュー上のメッセージのロックに関連したオプションです。

GMLK

メッセージをロックします。

このオプションは、ブラウザされるメッセージをロックするため、ロックされたメッセージは、このキューに対してオープンされた他のハンドルには、見えなくなります。このオプションを指定できるのは、以下のオプションのいずれか 1 つが指定されている場合に限りです。

- GMBRWF
- GMBRWN
- GMBRWC

キュー・ハンドルごとに 1 つのメッセージしかロックできません。ただし、メッセージは、次に示す論理メッセージまたは物理メッセージに限りです。

- GMCMPPM を指定すると、論理メッセージを構成しているメッセージ・セグメントがすべて、キュー・ハンドルに対してロックされます (これらのセグメントすべてが、そのキューに含まれ、取り出し可能になっている場合)。
- GMCMPPM を指定しないと、1 つの物理メッセージのみがキュー・ハンドルに対してロックされます。ロックされたメッセージが論理メッセージのセグメントだった場合は、そのセグメントがロックされていると、他のアプリケーションで GMCMPPM を指定しても、論理メッセージの取り出しやブラウザはできません。

ロックされたメッセージは常にブラウザ・カーソルのもとにあり、後で GMMUC オプションを指定した MQGET 呼び出しを発行するとキューから除去できます。キュー・ハンドルを使用する他の

MQGET 呼び出し (例えば、ロックされたメッセージのメッセージ ID を指定した呼び出し) でも、メッセージを削除できます。

呼び出しが完了コード CCFAIL または CCWARN を理由コード RC2080 と共に戻した場合、メッセージはロックされません。

キューからメッセージを除去しないようにアプリケーションで決定した場合は、以下のようにしてロックを解除します。

- このハンドルに対して別の MQGET 呼び出しを発行する。このとき、GMBRWF または GMBRWN のいずれかを指定します (GMLK はあってもなくても構いません)。呼び出しが CCOK または CCWARN で完了すると、メッセージはアンロックされますが、CCFAIL で完了したときは、ロックしたままになります。ただし、次の場合は例外になります。

- CCWARN が理由コード RC2080 と共に戻される場合、メッセージはアンロックされません。
- CCFAIL が理由コード RC2033 と共に戻される場合、メッセージはアンロックされます。

GMLK も指定されている場合、戻されるメッセージはロックされます。GMLK が指定されない場合は、呼び出し後、ロックされるメッセージはありません。

GMWT が指定され、ただちにメッセージを利用できない場合、待機の開始前 (ただし呼び出しでエラーが発生しない) に元のメッセージがアンロックされます。

- このハンドルに対して別の MQGET 呼び出しを発行する。このとき、GMBRWC を指定します (GMLK は指定しません)。呼び出しが CCOK または CCWARN で完了すると、メッセージはアンロックされますが、CCFAIL で完了したときは、ロックしたままになります。ただし、次のような例外があります。

- CCWARN が理由コード RC2080 と共に戻される場合、メッセージはアンロックされません。

- このハンドルに対して、GMUNLK を指定した別の MQGET 呼び出しを発行する。
- このハンドルに対して MQCLOSE 呼び出しを発行する (明示的に、またはアプリケーションの終了によって暗黙的に)。

このオプションを指定するとき、随伴するブラウズ・オプションを指定するために必要な OOBROW 以外に特別なオープン・オプションは必要ありません。

このオプションと組み合わせて使用できないオプションは、以下のとおりです。

- GMSYP
- GMPSYP
- GMUNLK

GMUNLK

メッセージをアンロックします。

アンロックされるメッセージは、GMLK オプションを指定した MQGET 呼び出しにより前もってロックされていなければなりません。このハンドルにロックされたメッセージがない場合は、呼び出しが CCWARN および理由コード RC2209 で完了します。

GMUNLK が指定されている場合、MSGDSC、BUFLN、BUFFER、および DATLEN パラメーターは検査も変更もされません。BUFFER にメッセージは返されません。

(最初にロック要求を出すために OOBROW が必要ですが) このオプションを指定するのに特別なオープン・オプションは必要ありません。

このオプションは、以下のオプション以外のオプションとは組み合わせて使用できません。

- GMNWT
- GMNSYP

これらのオプションは、指定されているかどうかに関わらず、どちらも想定されています。

メッセージ・データ・オプション: 以下のオプションは、メッセージがキューから読み取られるときのメッセージ・データの処理に関連しています。

GMATM

メッセージ・データの切り捨てを許可します。

メッセージ・バッファが小さいためにメッセージ全体を収容できない場合に、このオプションを使用すると、MQGET 呼び出しでは、バッファに収容できる分だけメッセージを入れ、警告完了コードを出して、その処理を終了できます。これは、以下のことを意味します。

- メッセージをブラウズすると、ブラウズ・カーソルが、戻されたメッセージに進みます。
- メッセージを削除すると、戻されたメッセージがキューから削除されます。
- 他のエラーが発生しない場合は、理由コード RC2079 が返されます。

このオプションを指定しないと、バッファは、収容できる分だけのメッセージで満たされても、警告完了コードが戻りますが、処理は完了しません。これは、以下のことを意味します。

- メッセージをブラウズしても、ブラウズ・カーソルは進みません。
- メッセージを削除しても、メッセージはキューから削除されません。
- 他のエラーが発生しない場合は、理由コード RC2080 が返されます。

GMCONV

メッセージ・データを変換します。

このオプションは、データが **BUFFER** パラメーターにコピーされる前に、MQGET 呼び出しの **MSGDSC** パラメーターに指定された **MDCSI** および **MDENC** 値に準拠するように、メッセージ内のアプリケーション・データを変換することを要求します。

メッセージが書き込まれたときに指定された **MDFMT** フィールドは、メッセージ内のデータの性質を識別するために変換プロセスによって想定されます。メッセージ・データの変換は、組み込み形式の場合はキュー・マネージャーによって行われ、他の形式の場合はユーザー作成出口によって行われます。

- 変換が正常に実行された場合、MQGET 呼び出しからの戻り時に、**MSGDSC** パラメーターに指定された **MDCSI** および **MDENC** フィールドは変更されません。
- 変換を正常に実行できない場合 (ただし、それ以外の場合は MQGET 呼び出しがエラーなしで完了する場合)、メッセージ・データは変換されずに戻され、**MSGDSC** の **MDCSI** および **MDENC** フィールドは未変換のメッセージの値に設定されます。この場合、完了コードは CCWARN です。

したがって、どちらの場合も、これらのフィールドは、**BUFFER** パラメーターで返されるメッセージ・データの文字セット ID とエンコードを記述します。

キュー・マネージャーが変換を実行する形式名のリストについては、[1114 ページの『IBM iでのMQMD \(メッセージ記述子\)』](#)で説明されている **MDFMT** フィールドを参照してください。

グループおよびセグメント・オプション: 以下のオプションは、論理メッセージのグループおよびセグメントに含まれるメッセージの処理に関連しています。これらの定義を理解しておく、オプションを把握するのに役に立ちます。

物理メッセージ

このメッセージは、キューに入れたりキューから除去できる最小単位の情報です。多くの場合、1つのMQPUT、MQPUT1、またはMQGET 呼び出しで指定された情報や取り出された情報に相当します。すべての物理メッセージには、固有のメッセージ記述子 (MQMD) があります。通常、物理メッセージは、メッセージ ID (MQMD の **MDMID** フィールド) の異なる値によって区別されます。ただし、これはキュー・マネージャーによって強制されるものではありません。

論理メッセージ

これは、1単位のアプリケーション情報です。システムに制約がない場合には、1つの論理メッセージが1つの物理メッセージになることもあります。ただし、論理メッセージが大きい場合、システムの制約により、1つの論理メッセージをセグメントと呼ばれる複数の物理メッセージに分割することが必要になる場合があります。

セグメント化された論理メッセージは、同じ非ヌル・グループ ID (MQMD の **MDGID** フィールド) を持つ複数の物理メッセージと、同じメッセージ・シーケンス番号 (MQMD の **MDSEQ** フィールド) で構成されます。セグメントは、セグメント・オフセット (MQMD の **MDOFF** フィールド) の固有の値

によって区別されます。この値は、論理メッセージ内のデータの先頭からの物理メッセージ内のデータのオフセットを示します。各セグメントは1つの物理メッセージなので、論理メッセージ内のセグメントには通常、それぞれ固有のメッセージ ID があります。

セグメント化されていない論理メッセージにも、送信側のアプリケーションでセグメント化が許可されている場合は、NULL でないグループ ID があります。ただし、この場合、そのグループ ID を持つのは、論理メッセージが1つのメッセージ・グループに属していないと、1つの物理メッセージのみです。論理メッセージが1つのメッセージ・グループに属していない場合、送信側のアプリケーションによってセグメント化が禁止されている論理メッセージのグループ ID はヌルとなります (GINONE)。

メッセージ・グループ

非空文字の同じグループ ID をもつ1つ以上の論理メッセージから構成される集合です。グループ内のそれぞれの論理メッセージは、メッセージ順序番号に指定された固有の値で区別されます。指定される値は1からnまでの整数で、nはグループ内の論理メッセージの数です。1つ以上の論理メッセージをセグメント化すると、グループ内の物理メッセージの数はn個を超えます。

GMLOGO

グループ内のメッセージおよび論理メッセージのセグメントが、論理順序で戻されます。

このオプションにより、キュー・ハンドルに対する連続した MQGET 呼び出しに対して、戻されるメッセージの順序を制御します。有効に活用するには、必ずそれぞれの呼び出しごとにこのオプションを指定してください。

キュー・ハンドルに対する連続した MQGET 呼び出しで GMLOGO を指定すると、グループ内のメッセージはそのメッセージ順序番号に指定された順序で戻され、論理メッセージのセグメントはそのセグメント・オフセットに指定された順序で戻されます。この順序は、これらのメッセージやセグメントのキューでの出現順序とは異なる場合があります。

注: GMLOGO を指定しても、グループに属さないメッセージおよびセグメントでないメッセージには悪影響を及ぼしません。そのようなメッセージは、事実上、1つのメッセージのみで構成されるメッセージ・グループに属するものとして扱われます。したがって、グループに属すメッセージ、メッセージ・セグメント、グループに属さないセグメント化されていないメッセージが混在している可能性があるキューからメッセージを取り出すため GMLOGO を指定しても、まったく安全です。

必要な順序でメッセージを戻すように、キュー・マネージャーが、連続した MQGET 呼び出し間でのグループおよびセグメント情報を保持します。この情報により、キュー・ハンドルに対する現行のメッセージ・グループと現行の論理メッセージ、グループおよび論理メッセージ内の現行の位置、およびメッセージが1つの作業単位内で取り出されているかどうかが特定されます。キュー・マネージャーがこの情報を保持するので、アプリケーションでは、それぞれの MQGET 呼び出しの前にグループおよびセグメント情報を設定する必要はありません。具体的には、アプリケーションが MQMD の MDGID、MDSEQ、および MDOFF フィールドを設定する必要がないことを意味します。ただし、アプリケーションでは、それぞれの呼び出しに GMSYP オプションまたは GMNSYP オプションのいずれかを正しく設定する必要があります。

キューがオープンしているときは、現行のメッセージ・グループも現行の論理メッセージも存在しません。1つのメッセージ・グループが現行のメッセージ・グループとなるのは、MFMIG フラグ付きのメッセージが MQGET 呼び出しで戻されたときです。以降の呼び出しで GMLOGO を指定すると、以下のいずれかを含むメッセージが戻されるまで、現行のグループは存続します。

- MFSEG の付いていない MFLMIG (グループ内の最後の論理メッセージがセグメント化されていない)
- MFLSEG の付いた MFLMIG (戻されたメッセージがグループ内の最後の論理メッセージの最後のセグメントである)

このようなメッセージが戻されると、メッセージ・グループが終了し、MQGET 呼び出しが正常に完了すると現行のグループはなくなります。同様に、MFSEG フラグ付きのメッセージが MQGET 呼び出しで戻されると、1つの論理メッセージが現行の論理メッセージとなり、MFLSEG フラグ付きのメッセージが戻されると、その論理メッセージが終了します。

選択基準が指定されていない場合、後続の MQGET 呼び出しは、キュー上の最初のメッセージ・グループのメッセージを正しい順序で戻します。次に2番目のメッセージ・グループのメッセージが戻されます。これは、使用できるメッセージがなくなるまで続きます。GMMO フィールドに以下の

オプションを1つ以上指定することによって、戻される特定のメッセージ・グループを選択することができます。

- MOMSGI
- MOCORI
- MOGRPI

ただし、これらのオプションは、現行のメッセージ・グループまたは論理メッセージがない場合にのみ有効です。このトピックで説明されている *GMMO* フィールドを参照してください。

1093 ページの表 741 は、MQGET 呼び出しで戻すメッセージを見つけるときにキュー・マネージャーが検索する *MDMID*、*MDCID*、*MDGID*、*MDSEQ*、および *MDOFF* フィールドの値を示しています。これは、キューからメッセージを除去する場合にもキューに入っているメッセージをブラウズする場合にも適用されます。この表の列の意味は、次のとおりです。

LOG ORD

GMLOGO オプションが呼び出しで指定されているかどうかを示します。

Cur grp

現行のメッセージ・グループが呼び出しの前に存在するかどうかを示します。

Cur log msg

現行の論理メッセージが呼び出しの前に存在するかどうかを示します。

その他の列

キュー・マネージャーが検索する値を示します。「前の」という表現は、キュー・ハンドルに対して前のメッセージのフィールドに戻された値を示します。

表 741. 論理メッセージのグループおよびセグメントに関連した MQGET オプション							
指定するオプション	呼び出しの前のグループおよび論理メッセージの状況		キュー・マネージャーが検索する値				
LOG ORD	Cur grp	Cur log msg	<i>MDMID</i>	<i>MDCID</i>	<i>MDGID</i>	<i>MDSEQ</i>	<i>MDOFF</i>
はい	いいえ	いいえ	統制活動担当者 <i>GMMO</i>	統制活動担当者 <i>GMMO</i>	統制活動担当者 <i>GMMO</i>	1	0
はい	いいえ	はい	任意のメッセージ ID	任意の相関 ID	前のグループ ID	1	前のオフセット + 前のセグメント長
はい	はい	いいえ	任意のメッセージ ID	任意の相関 ID	前のグループ ID	前の順序番号 + 1	0
はい	はい	はい	任意のメッセージ ID	任意の相関 ID	前のグループ ID	前の順序番号	前のオフセット + 前のセグメント長
いいえ	どちらも	どちらも	統制活動担当者 <i>GMMO</i>	統制活動担当者 <i>GMMO</i>	統制活動担当者 <i>GMMO</i>	統制活動担当者 <i>GMMO</i>	統制活動担当者 <i>GMMO</i>

キューの中に複数のメッセージ・グループが存在し、戻されるのに適格である場合、それらのグループが戻される順序は、各グループ内の最初の論理メッセージの最初のセグメントがキューの中のどの位置にあるかによって決定されます(つまり、メッセージ順序番号が 1、オフセットが 0 の物理メッセージにより、適格なグループが戻される順序が決定されます)。

GMLOGO オプションが作業単位に及ぼす影響は、以下のとおりです。

- 1 つのグループ内の最初の論理メッセージまたはセグメントが 1 つの作業単位内で取得された場合には、同じキュー・ハンドルが使用されていれば、そのグループ内の他の論理メッセージおよびセグメントはすべて 1 つの作業単位内で取得する必要があります。ただし、これらは同じ作業

単位内で取得する必要はありません。これにより、多くの物理メッセージからなるメッセージ・グループを、キュー・ハンドルに対する 2 つ以上の連続した作業単位にまたがって分割できます。

- 1 つのグループ内の最初の論理メッセージまたはセグメントが 1 つの作業単位内で取り出されていない場合に、同じキュー・ハンドルが使用されているときは、そのグループ内のその他の論理メッセージおよびセグメントはどれも 1 つの作業単位内で取り出すことはできません。

これらの条件が満たされないと、MQGET 呼び出しは失敗し、理由コード RC2245 が戻ります。

GMLOGO を指定した場合には、MQGET 呼び出しで提供された MQGMO が GMVER2 より下位であったり、MQMD が MDVER2 より下位であったりしてはなりません。この条件が満たされないと、呼び出しは失敗し、理由コード RC2256 または RC2257 の該当する方が戻ります。

キュー・ハンドルに対する連続した MQGET 呼び出しで GMLOGO を指定しないと、メッセージ・グループに属していなくても、また論理メッセージのセグメントでなくてもメッセージは戻されます。これにより、あるグループ内のメッセージや論理メッセージのセグメントが正しくない順序で戻されたり、他のグループ内のメッセージ、他の論理メッセージのセグメント、またはグループにも属さずセグメントでもないメッセージとが混在したりする場合があります。この場合、連続する MQGET 呼び出しによって戻される特定のメッセージは、それらの呼び出しで指定された MO* オプションによって制御されます(これらのオプションの詳細については、1080 ページの『IBM i での MQGMO (読み取りメッセージ・オプション)』で説明されている GMMO フィールドを参照してください)。

この手法を用いると、システム障害が発生した後に、メッセージ・グループまたは論理メッセージを途中から再開することができます。システムの再始動時に、アプリケーションは、MDGID、MDSEQ、MDOFF、および GMMO フィールドを適切な値に設定し、GMLOGO を指定せずに、必要に応じて GMSYP または GMNSYP を設定した MQGET 呼び出しを発行することができます。この呼び出しが成功した場合、キュー・マネージャーはグループとセグメントの情報を保存し、そのキュー・ハンドルを使用する後続の MQGET 呼び出しで通常どおり GMLOGO を指定できます。

MQGET 呼び出しのためにキュー・マネージャーが保持しているグループおよびセグメント情報は、MQPUT 呼び出しのためにキュー・マネージャーが保持しているグループおよびセグメント情報とは異なります。また、キュー・マネージャーは、以下についての情報もそれぞれ保持しています。

- キューからメッセージを削除する MQGET 呼び出し。
- キュー上のメッセージをブラウズする MQGET 呼び出し。

キュー・ハンドルが指定されている場合には、アプリケーションでは、GMLOGO を指定した MQGET 呼び出しと GMLOGO を指定していない MQGET 呼び出しを自由に組み合わせて使用できます。ただし、以下の点に注意してください。

- GMLOGO を指定していない場合は、MQGET 呼び出しが成功するたびに、キュー・マネージャーが、保存しているグループおよびセグメント情報を、戻されたメッセージに対応する値に設定します。これにより、キュー・ハンドルに対してキュー・マネージャーで保存されていた既存のグループおよびセグメント情報が置換されます。呼び出しのアクション(ブラウズまたは削除)に該当する情報だけが変更されます。
- GMLOGO を指定していない場合、現行のメッセージ・グループまたは論理メッセージがあれば、呼び出しは失敗しません。ただし、CCWARN 完了コードで呼び出しが成功する場合もあります。1095 ページの表 742 に、発生する可能性のあるいくつかのケースを示しています。これらの場合に、完了コードが CCOK 以外であれば、理由コードは以下のいずれかになります。

- RC2241
- RC2242
- RC2245

注: キュー・マネージャーは、キューをブラウズする場合や、入力ではなくブラウズ用にオープンされているキューをクローズする場合には、グループおよびセグメント情報をチェックしません。このような場合、完了コードは常に CCOK です(他にエラーがないものと見なされます)。

表 742. グループおよびセグメント情報と整合性のない MQGET または MQCLOSE 呼び出しの結果		
現行の呼び出し	直前の呼び出しは GMLOGO を指定している MQGET だった	直前の呼び出しは GMLOGO を指定していない MQGET だった
GMLOGO を指定している MQGET	CCFAIL	CCFAIL
GMLOGO を指定していない MQGET	CCWARN	CCOK
終了していないグループまたは論理メッセージを指定している MQCLOSE	CCWARN	CCOK

単にメッセージおよびセグメントを論理順序で取り出すアプリケーションでは、最も簡単に使えるオプション **GMLOGO** を指定するようにしてください。このオプションを指定すると、キュー・マネージャーがグループおよびセグメント情報を管理するので、アプリケーションでこの情報を管理する必要はなくなります。しかし、**GMLOGO** オプションで提供される制御以外の制御が必要となる特殊なアプリケーションもあります。このようなアプリケーションでは、このオプションを指定しないようにしてください。これを行う場合、アプリケーションは、それぞれの **MQGET** 呼び出しの前に、**MQMD** 内の **MDMID**、**MDCID**、**MDGID**、**MDSEQ**、および **MDOFF** フィールド、および **MQGMO** 内の **GMMO** 内の **MO*** オプションが正しく設定されていることを確認する必要があります。

例えば、受信した物理メッセージがグループに属していなくても、また論理メッセージのセグメントでなくても、そのメッセージを転送するアプリケーションでは、**GMLOGO** を指定しないようにしてください。これは、送信側のキュー・マネージャーと受信側のキュー・マネージャーの間にパスがいくつもあるような複雑なネットワークの場合に、物理メッセージが正しくない順序で到着することがあるからです。**GMLOGO** およびこれに対応する **MQPUT** 呼び出し上の **PMLOGO** を両方とも指定しないようにすると、転送側のアプリケーションでは、論理順序で次にあるメッセージが到着するのを待たなくても、それぞれの物理メッセージの到着と同時にそのメッセージを取り出し転送することができます。

GMLOGO は、その他の **GM*** オプションと組み合わせても、また状況に応じて各種 **MO*** オプションと組み合わせても指定できます。

GMCMPPM

完全な論理メッセージのみが取り出し可能です。

このオプションを指定すると、完全な論理メッセージのみを **MQGET** 呼び出しで戻すことができます。論理メッセージがセグメント化されていると、キュー・マネージャーはそれらのセグメントの再組み立てを行い、完全な論理メッセージをアプリケーションに戻します。論理メッセージがセグメント化されていたことは、それを受け取るアプリケーションには分かりません。

注: キュー・マネージャーがメッセージ・セグメントの再組み立てを行うように指定するのは、このオプションだけです。このオプションを指定しないと、キューに入っている (および **MQGET** 呼び出しで指定された他の選択基準を満たしている) セグメントはそれぞれ個別にアプリケーションに戻されます。セグメントを個別に受け取らない場合は、必ずアプリケーションで **GMCMPPM** を指定してください。

このオプションを使用するには、完全なメッセージを収容できるバッファがアプリケーションに提供されているか、**GMATM** オプションがアプリケーションに指定されている必要があります。

セグメント化されたメッセージがキューに入っている、セグメントのいくつかが足りない (ネットワーク内が混雑していてまだ到着していないことが考えられる) 場合、**GMCMPPM** を指定していると、完全でない論理メッセージに属しているセグメントが取り出されることはありません。ただし、これらのメッセージ・セグメントは引き続き **CurrentQDepth** キュー属性の値に寄与します。これは、**CurrentQDepth** がゼロより大きい場合でも、検索可能な論理メッセージが存在しない可能性があることを意味します。

持続メッセージの場合、キュー・マネージャーがセグメントの再組み立てを実行できるのは、1つの作業単位内だけです。

- MQGET 呼び出しがユーザー定義の作業単位内で実行されていれば、その作業単位が使用されます。呼び出しが再組み立て処理の途中で失敗した場合、キュー・マネージャーは、再組み立て中に除去されたすべてのセグメントを再度キューに配置します。ただし、このように失敗しても、作業単位は正常にコミットされます。
- 呼び出しがユーザー定義の作業単位以外で実行されていて、またユーザー定義の作業単位が存在していない場合、キュー・マネージャーは、この呼び出しの間だけの作業単位を作成します。呼び出しが成功すると、キュー・マネージャーは作業単位を自動的にコミットします (アプリケーションでこれを行う必要はありません)。呼び出しが失敗すると、キュー・マネージャーは作業単位をバックアウトします。
- 呼び出しがユーザー定義の作業単位以外で実行されていて、さらにユーザー定義の作業単位も存在している場合、キュー・マネージャーは再組み立てを実行できません。再組み立てが必要でないメッセージでは、呼び出しが成功することもあります。しかし、再組み立てが必要なメッセージでは、呼び出しは失敗し、理由コード RC2255 が戻ります。

非持続メッセージの場合、キュー・マネージャーが再組み立てを実行するのに、1つの作業単位が使用可能になっている必要はありません。

1つのセグメントであるそれぞれの物理メッセージには、固有のメッセージ記述子があります。単一の論理メッセージを構成するセグメントの場合、メッセージ記述子内のほとんどのフィールドは、論理メッセージ内のすべてのセグメントで同じです。通常、論理メッセージ内のセグメント間で異なるのは、*MDMID*、*MDOFF*、および *MDMFL* フィールドのみです。ただし、セグメントが中間キュー・マネージャーの送達不能キューの中にある場合、DLQ ハンドラーは *GMCONV* オプションを指定してメッセージを検索するため、そのセグメントの文字セットまたはエンコードが変更されてしまう可能性があります。それで、この途中で DLQ ハンドラーがセグメントの送信に成功した場合、そのセグメントが最終的に宛先キュー・マネージャーに到着した時点で、そのセグメントの文字セットまたはエンコードは論理メッセージ内の他のセグメントと異なっていることがあります。

MDCSI、*MDENC*、または両方のフィールドが異なるセグメントで構成される論理メッセージを、キュー・マネージャーが単一の論理メッセージに再組み立てすることはできません。その代わりに、キュー・マネージャーは、論理メッセージの先頭にあり、同じ文字セット ID とエンコードを持つ連続したセグメントをいくつか再組み立てして戻します。このとき、MQGET 呼び出しは完了し、完了コード *CCWARN* と、理由コード *RC2243* または *RC2244* の該当する方が戻ります。これは、*GMCONV* が指定されていなくても実行されます。残りのセグメントを取り出すには、アプリケーションで *GMCPMP* オプションを指定せずに MQGET 呼び出しを再発行する必要があります。これによって、セグメントが1つずつ取り出せるようになります。GMLOGO を使用すれば、残りのセグメントを順番に取り出すことができます。

セグメントの書き込みを行うアプリケーションでは、メッセージ記述子内の他のフィールドに、セグメント間で異なる値を設定できます。しかし、受信側のアプリケーションで論理メッセージの取り出しに *GMCPMP* が使用されている場合は、これを実行しても利点はありません。キュー・マネージャーは、論理メッセージの再組み立て時に、最初のセグメントのメッセージ記述子からの値をメッセージ記述子に戻します。唯一の例外は *MDMFL* フィールドで、再組み立てされたメッセージが唯一のセグメントであることを示すためにキュー・マネージャーが設定します。

GMCPMP をレポート・メッセージ用に指定すると、キュー・マネージャーは特殊な処理を実行します。キュー・マネージャーは、キューをチェックして、論理メッセージ内のさまざまなセグメントに関連するそのレポート・タイプのすべてのレポート・メッセージがキュー上に存在しているかどうかを確認します。入っている場合には、*GMCPMP* を指定することでそれらのメッセージを1つのメッセージとして取り出すことができます。これには、セグメント化をサポートしているキュー・マネージャーまたは *MCA* を使用してレポート・メッセージを生成するか、発信側のアプリケーションで 100 バイト以上のメッセージ・データを要求する必要があります (つまり、*RO*D* または *RO*F* オプションの該当する方を指定する必要があります)。1つのセグメントに入るアプリケーション・データの一部に欠落があると、戻されるレポート・メッセージでは、足りないバイト分はヌルで置き換えられます。

GMCPMP を *GMMUC* または *GMBRWC* と一緒に指定する場合、MQMD の *MDOFF* フィールドの値が 0 のメッセージにブラウズ・カーソルを置く必要があります。この条件を満たさないと、この呼び出しは失敗し、理由コード *RC2246* が戻ります。

GMCMPPM は、GMASGA に暗黙的に含まれているため指定する必要はありません。

GMCMPPM は、GMPSYP 以外の GM* オプションおよび MOOFFS 以外の MO* オプションと組み合わせで指定できます。

GMAMSA

グループ内のメッセージはすべて使用可能でなければなりません。

このオプションを指定すると、グループ内のメッセージがすべて使用可能な場合に限り、そのグループ内のメッセージを取得できるようになります。メッセージ・グループがキューに入っているも、メッセージのいくつかが足りない(ネットワーク内が混雑していてまだ到着していないことが考えられる)場合、GMAMSA を指定していると、完全でないグループに属しているメッセージが取り出されることはありません。ただし、これらのメッセージは引き続き **CurrentQDepth** キュー属性の値に寄与します。これは、**CurrentQDepth** がゼロより大きい場合でも、検索可能なメッセージ・グループが存在しない可能性があることを意味します。取得可能なメッセージが他にない場合は、指定の待機間隔(指定されていれば)が終了した後で理由コード RC2033 が戻されます。

GMAMSA の処理は、GMLOGO が指定されているかどうかで異なります。

- これらのオプションを両方とも指定している場合、GMAMSA が有効となるのは、現行のグループまたは論理メッセージが存在していない場合のみです。現行のグループまたは論理メッセージが存在している場合には、GMAMSA は無視されます。つまり、メッセージが論理順序で処理されているときは、GMAMSA はオンになっています。
- GMLOGO を指定せずに GMAMSA を指定している場合、GMAMSA は常に有効です。つまり、このオプションは、グループ内の最初のメッセージがキューから削除されたときにオフにする必要があります。これによって、そのグループ内の残りのメッセージを削除できるようになります。

GMAMSA を指定した MQGET 呼び出しが正常に完了するということは、MQGET 呼び出しが発行された時点で、グループ内のすべてのメッセージがキュー上にあるという意味です。ただし、その場合も、他のアプリケーションがグループからメッセージを除去することができます(グループは、グループ内の最初のメッセージを取り出したアプリケーションに対してロックされません)。

このオプションを指定しないと、不完全なグループに属しているメッセージが取り出されることがあります。

GMAMSA は GMASGA を暗黙的に含んでいるため、指定する必要はありません。

GMAMSA は、他のすべての GM* オプションおよびすべての MO* オプションと組み合わせで指定できます。

GMASGA

論理メッセージ内のセグメントはすべて使用可能でなければなりません。

このオプションを指定すると、論理メッセージ内のセグメントがすべて使用可能な場合に限り、その論理メッセージ内のセグメントを取得できるようになります。セグメント化されたメッセージがキューに入っているも、セグメントのいくつかが足りない(ネットワーク内が混雑していてまだ到着していないことが考えられる)場合、GMASGA を指定していると、完全でない論理メッセージに属しているセグメントが取り出されることはありません。ただし、これらのセグメントは引き続き **CurrentQDepth** キュー属性の値に寄与します。これは、**CurrentQDepth** がゼロより大きい場合でも、検索可能な論理メッセージが存在しない可能性があることを意味します。取得可能なメッセージが他にない場合は、指定の待機間隔(指定されていれば)が終了した後で理由コード RC2033 が戻されます。

GMASGA の処理は、GMLOGO が指定されているかどうかで異なります。

- これらのオプションを両方とも指定している場合、GMASGA が有効となるのは、現行の論理メッセージが存在していない場合のみです。現行の論理メッセージが存在している場合には、GMASGA は無視されます。つまり、メッセージが論理順序で処理されているときは、GMASGA はオンになっています。
- GMLOGO を指定せずに GMASGA のみを指定している場合は、GMASGA は常に有効です。つまり、このオプションは、論理メッセージ内の最初のセグメントがキューから削除されたときにオフにする必要があります。これによって、その論理メッセージ内の残りのセグメントを削除できるようになります。

このオプションを指定しないと、論理メッセージが不完全な場合でも、メッセージ・セグメントが取得されることがあります。

GMCMPPM および GMASGA を両方とも指定している場合は、すべてのセグメントを取り出す前に、まずそれらを使用可能にする必要があります。前者のみの場合は、完全なメッセージが戻され、後者のみの場合は、セグメントを1つずつ取り出すことができます。

GMASGA をレポート・メッセージ用に指定すると、キュー・マネージャーは特殊な処理を実行します。キュー・マネージャーは、キューをチェックして、完全な論理メッセージを構成しているそれぞれのセグメントに対して少なくとも1つのレポート・メッセージが入っているかどうかを確認します。入っている場合は、GMASGA 条件は満たされます。しかし、キュー・マネージャーは、キューに入っているレポート・メッセージのタイプはチェックしないので、論理メッセージのセグメントに対応したレポート・メッセージにはさまざまなタイプのレポートが混在している可能性があります。結果として、GMASGA が成功しても、GMCMPPM が成功するとは限りません。ある論理メッセージのセグメントについてさまざまなレポート・タイプが混在している場合には、これらのレポート・メッセージは1つずつ取得する必要があります。

GMASGA は、他のすべての GM* オプションおよびすべての MO* オプションと組み合わせて指定できます。

デフォルト・オプション: 上記で説明されたオプションがいずれも必要でない場合、以下のオプションを使用できます。

GMNONE

指定されるオプションはありません。

この値は、他のオプションが指定されなかったことを示すために使用できます。すべてのオプションはデフォルト値を取ります。GMNONE は、プログラムの文書化をサポートするために定義します。したがって、このオプションは、他のオプションと同時に使用するものではありません。しかしこのオプションの値はゼロのため、他のオプションと同時に使用してもそれを検出できません。

GMOPT フィールドの初期値は GMNWT です。

GMRE1 (1 バイトの文字ストリング)

予約済み。

これは予約フィールドです。このフィールドの初期値は、ブランク文字です。GMVER が GMVER2 より小さい場合、このフィールドは無視されます。

GMRL (10 桁の符号付き整数)

戻されたメッセージ・データの長さ (バイト数)。

これは、MQGET 呼び出しの **BUFFER** パラメーターによって返されるメッセージ・データの長さ (バイト単位) にキュー・マネージャーが設定する出力フィールドです。キュー・マネージャーがこの機能をサポートしない場合、GMRL は値 RLUNDF に設定されます。

メッセージがエンコードや文字セット間で変換された場合、メッセージ・データのサイズが変わることがあります。MQGET 呼び出しからの戻り値は、次のようになります。

- GMRL が RLUNDF でない場合、返されるメッセージ・データのバイト数は GMRL によって指定されます。
- GMRL に値 RLUNDF がある場合、戻されるメッセージ・データのバイト数は、通常、BUFLen と DATLEN の小さい方になりますが、MQGET 呼び出しが理由コード RC2079 で終了した場合は、それよりも小さくなる場合があります。この場合、**BUFFER** パラメーター内の重要でないバイトはヌルに設定されます。

以下のような特殊値が定義されます。

RLUNDF

戻されたデータの長さが定義されていない。

このフィールドの初期値は RLUNDF です。GMVER が GMVER3 より小さい場合、このフィールドは無視されます。

GMRQN (48 バイトの文字ストリング)

宛先キューの解決された名前。

これは、メッセージが取り出されたキューのローカル名に対してキュー・マネージャーが設定した出力フィールドであり、ローカル・キュー・マネージャーに対して定義されるものです。次の場合には、キューをオープンするのに使用された名前とは異なります。

- 別名キューがオープンされた。この場合、別名が解決したローカル・キューの名前が戻されます。
- モデル・キューがオープンされた。この場合、動的なローカル・キューの名前が戻されます。

このフィールドの長さはLNQNによって指定されます。このフィールドの初期値は48個のブランク文字です。

GMRS2 (1 バイトの文字ストリング)

予約済み。

これは予約フィールドです。このフィールドの初期値は、ブランク文字です。GMVERがGMVER4より小さい場合、このフィールドは無視されます。

GMSEG (1 バイトの文字ストリング)

取り出されたメッセージに対して、さらにセグメント化が可能かどうかを示すフラグ。

以下の値がどれか1つ含まれています。

SEGIHB

セグメント化できない。

SEGALW

セグメント化できます。

これは出力フィールドです。このフィールドの初期値はSEGIHBです。GMVERがGMVER2より小さい場合、このフィールドは無視されます。

MSG1 (10 桁の符号付き整数)

シグナル。

これは、予約フィールドです。したがって、値に意味はありません。このフィールドの初期値は0です。

MSG2 (10 桁の符号付き整数)

シグナル ID。

これは、予約フィールドです。したがって、値に意味はありません。

GMSID (4 バイトの文字ストリング)

構造体 ID

値は次のものでなければなりません。

GMSIDV

読み取りメッセージ・オプション構造体の ID。

このフィールドは常に入力フィールドです。このフィールドの初期値はGMSIDVです。

GMSST (1 バイトの文字ストリング)

取り出されたメッセージが論理メッセージの1つのセグメントであるかどうかを示すフラグ。

以下の値がどれか1つ含まれています。

SSNSEG

メッセージは1つのセグメントではない。

SSSEG

メッセージは1つのセグメントであるが、論理メッセージの最後のセグメントではない。

SSLSEG

メッセージは論理メッセージの最後のセグメントである。

この値は、論理メッセージを構成しているセグメントが1つしかない場合にも戻されます。

このフィールドは出力フィールドです。このフィールドの初期値は SSNSEG です。GMVER が GMVER2 より小さい場合、このフィールドは無視されます。

GMTOK (16 バイトのビット・ストリング)

メッセージ・トークン。

これは、予約フィールドです。したがって、値に意味はありません。以下のような特殊値が定義されます。

MTKNON

メッセージ・トークンなし。

値は、フィールドの長さについては2進ゼロです。

このフィールドの長さは LNMTOK によって指定されます。このフィールドの初期値は MTKNON です。GMVER が GMVER3 より小さい場合、このフィールドは無視されます。

GMVER (10 桁の符号付き整数)

構造体のバージョン番号。

値は次のいずれかでなければなりません。

GMVER1

バージョン1の読み取りメッセージ・オプション構造体。

GMVER2

バージョン2の読み取りメッセージ・オプション構造体。

GMVER3

バージョン3の読み取りメッセージ・オプション構造体。

GMVER4

バージョン4の読み取りメッセージ・オプション構造体。

これより新しいバージョンの構造体にのみ存在するフィールドは、そのフィールドの説明にその旨記載されています。以下の定数は、現行バージョンのバージョン番号を指定しています。

GMVERC

読み取りメッセージ・オプション構造体の現行バージョン。

このフィールドは常に入力フィールドです。このフィールドの初期値は GMVER1 です。

GMVER (10 桁の符号付き整数)

構造体のバージョン番号。

値は次のいずれかでなければなりません。

GMVER1

バージョン1の読み取りメッセージ・オプション構造体。

GMVER2

バージョン2の読み取りメッセージ・オプション構造体。

GMVER3

バージョン3の読み取りメッセージ・オプション構造体。

GMVER4

バージョン4の読み取りメッセージ・オプション構造体。

これより新しいバージョンの構造体にのみ存在するフィールドは、そのフィールドの説明にその旨記載されています。以下の定数は、現行バージョンのバージョン番号を指定しています。

GMVERC

読み取りメッセージ・オプション構造体の現行バージョン。

このフィールドは常に入力フィールドです。このフィールドの初期値は GMVER1 です。

GMWI (10 桁の符号付き整数)

待機間隔。

これは、MQGET 呼び出しが適切なメッセージ (つまり、MQGET 呼び出しの **MSGDSC** パラメーターで指定された選択基準を満たすメッセージ) の到着を待機するおおよその時間 (ミリ秒単位) です。詳しくは、1114 ページの『IBM i での MQMD (メッセージ記述子)』で説明されている **MDMID** フィールドを参照してください。この時間が経過しても、適切なメッセージが到達しない場合、呼び出しは理由コード RC2033 と CCFAIL で完了します。

GMWI は GMWT オプションと共に使用されます。このオプションが指定されていない場合は無視されます。これを指定する場合、GMWI はゼロ以上か、または以下の特殊値でなければなりません。

WIULIM

無制限の待機間隔。

このフィールドの初期値は 0 です。

初期値

表 743. MQGMO のフィールド		
フィールド名	定数の名前	定数の値
GMSID	GMSIDV	'GMO-'
GMVER	GMVER1	1
GMOPT	GMNWT	0
GMWI	なし	0
GMSG1	なし	0
GMSG2	なし	0
GMRQN	なし	ブランク
GMMO	MOMSGI + MOCORI	3
GMGST	GSNIG	' '
GMSST	SSNSEG	' '
GMSEG	SEGIHB	' '
GMRE1	なし	' '
GMTOK	MTKNON	Null
GMRL	RLUNDF	-1
GMRS2	なし	' '
GMMH	HMNONE	0

注：

1. 記号-は、単一のブランク文字を表します。

RPG 宣言

```
D*..1.....2.....3.....4.....5.....6.....7..  
D*  
D* MQGMO Structure  
D*  
D* Structure identifier
```

```

D  GMSID                1      4      INZ('GMO ')
D* Structure version number
D  GMVER                5      8I 0 INZ(1)
D* Options that control the action ofMQGET
D  GMOPT                9      12I 0 INZ(0)
D* Wait interval
D  GMWI                13     16I 0 INZ(0)
D* Signal
D  MSG1                17     20I 0 INZ(0)
D* Signal identifier
D  MSG2                21     24I 0 INZ(0)
D* Resolved name of destination queue
D  GMRQN               25      72      INZ
D* Options controlling selection criteriaused for MQGET
D  GMMO               73     76I 0 INZ(3)
D* Flag indicating whether messageretrieved is in a group
D  GMGST              77      77      INZ(' ')
D* Flag indicating whether messageretrieved is a segment of a
D* logicalmessage
D  GMSST              78      78      INZ(' ')
D* Flag indicating whether furthersegmentation is allowed for themessage
D* retrieved
D  GMSEG              79      79      INZ(' ')
D* Reserved
D  GMRE1              80      80      INZ
D* Message token
D  GMTOK              81      96      INZ(X'0000000000000000-
D                                     0000000000000000')
D* Length of message data returned(bytes)
D  GMRL              97     100I 0 INZ(-1)
D* Reserved
D  GMRS2             101     104I 0 INZ(0)
D* Message handle
D  GMMH             105     112I 0 INZ(0)

```

IBM i MQIIH (IMS 情報ヘッダー) について IBM i

MQIIH 構造体は、IBM MQ for z/OS を介して IMS ブリッジに送信されるメッセージの先頭に組み込む必要のある情報を記述します。

概要

形式名: FMIMS。

文字セットおよびエンコード: MQIIH 構造体およびアプリケーション・メッセージ・データに使用する文字セットとエンコードに関しては、以下の特別条件に従うものとします。

- IMS ブリッジ・キューを所有するキュー・マネージャーに接続するアプリケーションは、キュー・マネージャーの文字セットとエンコードで記述されている MQIIH 構造体を提供する必要があります。その理由は、この場合には MQIIH 構造体のデータ変換を実行されないからです。
- 他のキュー・マネージャーに接続するアプリケーションでは、サポートされている文字セットとエンコードで記述されている MQIIH 構造体を提供することができます。MQIIH の変換は、IMS ブリッジ・キューを所有するキュー・マネージャーに接続している受信側のメッセージ・チャネル・エージェントによって実行されます。

注: この条件には例外が 1 つあります。IMS ブリッジ・キューを所有するキュー・マネージャーが CICS を分散キューイングに使用している場合、MQIIH 構造体は、IMS ブリッジ・キューを所有するキュー・マネージャーの文字セットとエンコードで記述されていなければなりません。

- MQIIH 構造体の後続くアプリケーション・メッセージ・データは MQIIH 構造体と同じ文字セットとエンコードでなければなりません。MQIIH 構造体の *IICSI* フィールドおよび *IIENC* フィールドを使用して、そのアプリケーション・メッセージ・データの文字セットおよびエンコードを指定することはできません。

データがキュー・マネージャーのサポートする組み込み形式ではない場合、アプリケーション・メッセージ・データを変換するには、ユーザーがデータ変換出口を提供する必要があります。

- [1103 ページの『IMS ブリッジ・アプリケーションでのパスチケットの認証』](#)
- [1103 ページの『フィールド』](#)

- [1106 ページの『初期値』](#)
- [1106 ページの『RPG 宣言』](#)

IMS ブリッジ・アプリケーションでのパスチケットの認証

IBM MQ 管理者は、IMS ブリッジ・アプリケーションでパスチケットを認証するために使用するアプリケーション名を指定できるようになりました。この場合、アプリケーション名は STGCLASS オブジェクト定義の新しい属性 PTKTAPPL として 1 から 8 文字の英数字ストリングで指定されます。

ブランク値は、以前のリリースの IBM MQ と同じように認証が行われることを意味します。つまり、認証要求でアプリケーション名が送られることはなく、代わりに MVSxxxx 値が使用されます。

1 から 8 文字の英数字の値は、RACF の資料で説明されているように、パスチケット・アプリケーション名の規則に従う必要があります。

IBM MQ 管理者および RACF 管理者の両方が、有効なアプリケーション名を使用することに同意する必要があります。RACF 管理者は、PTKTDATA クラスにプロファイルを作成し、アクセス権限を付与されるすべてのアプリケーションのユーザー ID に READ アクセス権限を与える必要があります。IBM MQ 管理者は、パスチケット認証に使われるアプリケーション名を指定する、必要な STGCLASS 定義を作成または変更する必要があります。

関連情報については、「MQSC コマンド・リファレンス」を参照してください。

フィールド

MQIIH 構造体には、以下のフィールドが含まれます。フィールドはアルファベット順に説明されています。

IIAUT (8 バイトの文字ストリング)

RACF パスワードまたはパスチケット。

このフィールドはオプションです。指定した場合は、MQMD セキュリティー・コンテキスト内のユーザー ID と共に、セキュリティ・コンテキストを提供するために IMS に送られる UTOKEN の作成に使用されます。指定しなかった場合は、ユーザー ID が検証なしで使用されます。これは、RACF スイッチの設定に左右されます。スイッチを設定するには、オーセンティケーターが必要な場合があります。

最初のバイトがブランクまたはヌルの場合は、フィールドは無視されます。以下に示す特別な値が使用されることがあります。

IAUNON

認証なし。

フィールドの長さは LNAUTH で指定されます。このフィールドの初期値は IAUNON です。

IICMT (1 バイトの文字ストリング)

コミット・モード。

コミットモード IMS の詳細については、OTMA リファレンスを参照してください。値は次のいずれかでなければなりません。

ICMCTS

コミット後に送信。

このモードは、出力の二重キューイングを暗黙指定しますが、領域占有時間は短くなります。高速パス・トランザクションおよび会話型トランザクションは、実行できません。

ICMSTC

送信後コミット。

このフィールドの初期値は ICMCTS です。

IICSI (10 桁の符号付き整数)

予約済み。

これは、予約フィールドです。したがって、値に意味はありません。このフィールドの初期値は 0 です。

IIENC (10 桁の符号付き整数)

予約済み。

これは、予約フィールドです。したがって、値に意味はありません。このフィールドの初期値は 0 です。

IIFLG (10 桁の符号付き整数)

フラグ。

値は次のものでなければなりません。

IINONE

フラグなし。

フィールドの初期値は IINONE です。

IIFMT (8 バイトの文字ストリング)

MQIIH に続くデータの IBM MQ 形式名。

ここでは、MQIIH 構造体の後に続くデータの IBM MQ 形式名を指定します。

MQPUT または MQPUT1 呼び出しでは、アプリケーションは、このフィールドをデータに適切な値に設定する必要があります。このフィールドのコーディングの規則は、MQMD の *MDFMT* フィールドの場合と同じです。

このフィールドの長さは LNFMT によって指定されます。このフィールドの初期値は FMNONE です。

IILEN (10 桁の符号付き整数)

MQIIH 構造体の長さ。

値は次のものでなければなりません。

IILEN1

IMS 情報ヘッダー構造体の長さ。

フィールドの初期値は IILEN1 です。

IILTO (8 バイトの文字ストリング)

論理端末の指定変更。

このフィールドは、IO PCB フィールド内に配置されます。フィールドはオプションです。指定されていない場合は、TPIPE 名が使用されます。最初のバイトがブランクまたはヌルの場合は、無視されます。

このフィールドの長さは LNLTOV によって指定されます。このフィールドの初期値は 8 個のブランク文字です。

IIMMN (8 バイトの文字ストリング)

メッセージ形式のサービス・マップ名。

このフィールドは、IO PCB フィールド内に配置されます。フィールドはオプションです。入力では MID を表し、出力では MOD を表します。最初のバイトがブランクまたはヌルの場合は、無視されます。

フィールドの長さは LNMFMN で指定されます。このフィールドの初期値は 8 個のブランク文字です。

IIRFM (8 バイトの文字ストリング)

応答メッセージの IBM MQ 形式名。

このフィールドは、現行メッセージに応答して送信される応答メッセージの IBM MQ 形式名です。コーディングの規則は、MQMD の *MDFMT* フィールドの場合と同じです。

このフィールドの長さは LNFMT によって指定されます。このフィールドの初期値は FMNONE です。

IIRSV (1 バイトの文字ストリング)

予約済み。

これは予約フィールドです。フィールドはブランクでなければなりません。

IISEC (1 バイトの文字ストリング)

セキュリティ有効範囲。

これは、必要な IMS セキュリティ処理を示します。以下の値が定義されます。

ISSCHK

セキュリティ有効範囲のチェック。

ACEE は、従属領域でなく、制御領域に作成されます。

ISSFUL

全セキュリティ有効範囲。

キャッシュ ACEE は制御領域に作成され、キャッシュ以外の ACEE は従属領域に作成されます。

ISSFUL を使用する場合は、ACEE を作成するユーザー ID から従属領域で使用するリソースにアクセスできるようにする必要があります。

このフィールドの ISSCHK および ISSFUL が指定されていない場合は、ISSCHK と見なされます。

フィールドの初期値は ISSCHK です。

IISID (4 バイトの文字ストリング)

構造体 ID

値は次のものでなければなりません。

IISIDV

IMS 情報ヘッダー構造体の ID。

フィールドの初期値は IISIDV です。

IITID (16 バイトのビット・ストリング)

トランザクション・インスタンス ID

このフィールドは IMS からの出力メッセージが使用するものであり、したがって最初の入力時には無視されます。IITST が ITSIC に設定されている場合は、IMS がメッセージを正しい会話に対応付けられるように、次回以降のすべての後続入力で、フィールドを提供する必要があります。以下に示す特別な値が使用されることがあります。

ITINON

トランザクション・インスタンス ID なし。

フィールドの長さは、LNTIID で指定されます。このフィールドの初期値は ITINON です。

IITST (1 バイトの文字ストリング)

トランザクション状態。

これは IMS 会話の状態を示します。最初の入力時には会話は存在しないので、このフィールドは最初の入力では無視されます。後続の入力では、会話が活動状態か否かを示します。出力では、IMS により設定されます。値は次のいずれかでなければなりません。

ITSIC

会話中。

ITSNIC

会話中でない。

ITSARC

トランザクション状態のデータを構造化形式で戻す。

この値が使用できるのは、IMS /DISPLAY TRAN コマンドの場合に限られます。これを指定すると、トランザクション状態のデータが、文字形式ではなく IMS 構造化形式で戻されます。詳細については、[トランザクション・IMSIBM MQ プログラムの作成](#)を参照してください。

フィールドの初期値は ITSNIC です。

IIVER (10 桁の符号付き整数)

構造体のバージョン番号。

値は次のものでなければなりません。

IIVER1

IMS 情報ヘッダー構造体のバージョン番号。

以下の定数は、現行バージョンのバージョン番号を指定しています。

IIVERC

IMS 情報ヘッダー構造体の現行バージョン。

フィールドの初期値は IIVER1 です。

初期値

表 744. MQIIH のフィールド		
フィールド名	定数の名前	定数の値
IISID	IISIDV	'IIH~'
IIVER	IIVER1	1
IILEN	IILEN1	84
IIENC	なし	0
IICSI	なし	0
IIFMT	FMNONE	ブランク
IIFLG	IINONE	0
IILTO	なし	ブランク
IIMMN	なし	ブランク
IIRFM	FMNONE	ブランク
IIAUT	IAUNON	ブランク
IITID	ITINON	Null
IITST	ITSNIC	' '
IICMT	ICMCTS	'0'
IISEC	ISSCHK	'C'
IIRSV	なし	' '

注：

1. 記号~は、単一のブランク文字を表します。

RPG 宣言

```
D*..1.....2.....3.....4.....5.....6.....7..
D*
D* MQIIH Structure
D*
```

```

D* Structure identifier
D IISID 1 4 INZ('IIH ')
D* Structure version number
D IIVER 5 8I 0 INZ(1)
D* Length of MQIIH structure
D IILEN 9 12I 0 INZ(84)
D* Reserved
D IIENC 13 16I 0 INZ(0)
D* Reserved
D IICSI 17 20I 0 INZ(0)
D* MQ format name of data that followsMQIIH
D IIFMT 21 28 INZ(' ')
D* Flags
D IIFLG 29 32I 0 INZ(0)
D* Logical terminal override
D IILTO 33 40 INZ
D* Message format services map name
D IIMMN 41 48 INZ
D* MQ format name of reply message
D IIRFM 49 56 INZ(' ')
D* RACF password or passticket
D IIAUT 57 64 INZ(' ')
D* Transaction instance identifier
D IITID 65 80 INZ('0000000000000000-0000000000000000')
D
D* Transaction state
D IITST 81 81 INZ(' ')
D* Commit mode
D IICMT 82 82 INZ('0')
D* Security scope
D IISEC 83 83 INZ('C')
D* Reserved
D IIRSV 84 84 INZ

```

IBM i

IBM i での MQIMPO (メッセージ・プロパティ照会オプション)

MQIMPO 構造体を使用すると、アプリケーションで、メッセージのプロパティを照会する方法を制御するオプションを指定できます。

概要

目的: この構造体は、MQINQMP 呼び出しの入力パラメーターです。

文字セットとエンコード: MQIMPO 内のデータは、アプリケーションの文字セットおよびアプリケーションのエンコードでなければなりません (ENNAT)。

- [1107 ページの『フィールド』](#)
- [1113 ページの『初期値』](#)
- [1113 ページの『RPG 宣言』](#)

フィールド

MQIMPO 構造体には、以下のフィールドが含まれます。フィールドは**アルファベット順**に説明されています。

IPOPT (10 桁の符号付き整数)

以下のオプションは、MQINQMP のアクションを制御します。以下に説明する 1 つ以上のオプションを指定できます。複数のオプションを指定するには、値を加算する (同じ定数を複数回加算しない) か、ビット単位 OR 演算を使用して値を結合 (プログラミング言語でビット演算がサポートされる場合) します。無効なオプションの組み合わせについては、本文中で指示しています。指示のない組み合わせはすべて有効です。

値データ・オプション: 以下のオプションは、プロパティがメッセージから取り出されときの値データの処理と関係しています。

IPCVAL

このオプションは、MQINQMP 呼び出しが *Value* 領域のプロパティ値を戻す前に指定された *IPREQCSI* および *IPREQENC* 値に、プロパティの値が適合するように変換することを要求します。

- 変換が正常に実行されると、*IPRETCSI* および *IPRETENC* フィールドは、MQINQMP 呼び出しからの戻り時に *IPREQCSI* および *IPREQENC* と同じ内容に設定されます。
- 変換は失敗したものの、MQINQMP 呼び出しがそれ以外についてはエラーなしで完了した場合、プロパティ値は変換されないまま返されます。

プロパティがストリングである場合、*IPRETCSI* および *IPRETENC* フィールドは、変換されないストリングの文字セットおよびエンコードに設定されます。

この場合、完了コードは CCWARN であり、理由コードは RC2466 です。プロパティ・カーソルは、返されたプロパティに進みます。

プロパティ値が変換中に拡張し、**Value** パラメーターのサイズを超える場合は、値は未変換のまま返され、完了コード CCFAIL が出されます。理由コードは RC2469 に設定されます。

MQINQMP 呼び出しの **DataLength** パラメーターは、変換されたプロパティ値を収容するために必要なバッファのサイズをアプリケーションが判別できるようにするために、プロパティ値が変換された場合の長さを返します。プロパティ・カーソルは変更されません。

このオプションは、以下のことも要求します。

- プロパティ名にワイルドカードが含まれているかどうか。および
- *IPRETNAMECHRP* フィールドが、戻される名前のアドレスまたはオフセットを使用して初期設定される。

その後、戻された名前は、*IPREQCSI* および *IPREQENC* の値に合うように変換されます。

- 変換が正常に実行されると、*IPRETNAMECHRP* の *VSCCSID* フィールドおよび返される名前のエンコードは、*IPREQCSI* および *IPREQENC* の入力値に設定されます。
- 変換は失敗したものの、MQINQMP 呼び出しがそれ以外についてはエラーまたは警告なしで完了した場合、返される名前は未変換のままです。この場合、完了コードは CCWARN であり、理由コードは RC2492 です。

プロパティ・カーソルは、返されたプロパティに進みます。値と名前の両方が未変換の場合は、RC2466 が返されます。

返される名前が変換中に拡張し、*RequestedName* の *VSBufsize* フィールドのサイズを超える場合、返されるストリングは未変換のままであり、完了コード CCFAIL が出され、理由コードは RC2465 に設定されます。

変換されたプロパティ値を収容するのに必要なバッファのサイズをアプリケーションが判別できるように、MQCHARV 構造の *VSLength* フィールドは、プロパティ値の変換結果の長さを返します。プロパティ・カーソルは変更されません。

IPCTYP

このオプションは、プロパティの値を、現行のデータ・タイプから、MQINQMP 呼び出しの **Type** パラメーターで指定したデータ・タイプに変換するように要求します。

- 変換が正常に実行されると、MQINQMP 呼び出しから戻る際に、**Type** パラメーターは変更されません。
- 変換は失敗したものの、MQINQMP 呼び出しがそれ以外についてはエラーなしで完了した場合、呼び出しは失敗し、理由コード RC2470 が出されます。プロパティ・カーソルは変更されません。

データ・タイプの変換により変換中に値が拡張し、変換された値が **Value** パラメーターのサイズを超える場合は、値は未変換のまま返され、完了コード CCFAIL が出されます。理由コードは RC2469 に設定されます。

MQINQMP 呼び出しの **DataLength** パラメーターは、変換されたプロパティ値を収容するために必要なバッファのサイズをアプリケーションが判別できるようにするために、プロパティ値が変換された場合の長さを返します。プロパティ・カーソルは変更されません。

MQINQMP 呼び出しの **Type** パラメーターの値が無効の場合、呼び出しは失敗し、理由コード RC2473 が返されます。

要求されたデータ・タイプ変換がサポートされていない場合、呼び出しは失敗し、理由コード RC2470 が返されます。以下のデータ・タイプ変換がサポートされています。

表 745. サポートされているデータ・タイプ変換	
プロパティのデータ・タイプ	サポートされているターゲット・データ・タイプ
TYPBOL	TYPSTR、TYPI8、TYPI16、TYPI32、TYPI64
TYPBST	TYPSTR
TYPI8	TYPSTR、TYPI16、TYPI32、TYPI64
TYPI16	TYPSTR、TYPI32、TYPI64
TYPI32	TYPSTR、TYPI64
TYPI64	TYPSTR
TYPF32	TYPSTR、TYPF64
TYPF64	TYPSTR
TYPSTR	TYPBOL、TYPI8、TYPI16、TYPI32、TYPI64、TYPF32、TYPF64
TYPNUL	なし

サポートされる変換を制御する一般規則は、以下のとおりです。

- 数値のプロパティ値は、変換中にデータが失われなければ、データ・タイプ間で変換できる。
例えば、データ・タイプ TYPI32 のプロパティの値はデータ・タイプ TYPI64 の値に変換できますが、データ・タイプ TYPI16 の値には変換できません。
- どのデータ・タイプのプロパティ値でも、ストリングに変換できる。
- ストリング・プロパティ値は、ストリングが変換用に正しくフォーマットされていれば、任意の他のデータ・タイプに変換できます。アプリケーションが正しくフォーマットされていないストリング・プロパティ値に変換しようとすると、IBM MQ は理由コード RC2472 を返します。
- サポートされていない変換をアプリケーションが試行すると、IBM MQ は理由コード RC2470 を返します。

プロパティ値のデータ・タイプを変換する場合の具体的な規則は、以下のとおりです。

- TYPBOL プロパティ値をストリングに変換する場合、値 TRUE はストリング "TRUE" に変換され、値 FALSE はストリング "FALSE" に変換されます。
- TYPBOL プロパティ値を数値データ・タイプに変換する場合、値 TRUE は 1 に変換され、値 FALSE はゼロに変換されます。
- ストリング・プロパティ値を TYPBOL 値に変換する場合、ストリング "TRUE"、または "1" は "TRUE" に変換され、ストリング "FALSE"、または "0" は "FALSE" に変換されます。

用語「TRUE」および「FALSE」には大/小文字の区別がないことに注意してください。

その他のストリングは変換できません。IBM MQ は理由コード RC2472 を返します。

- ストリング・プロパティ値を、データ・タイプ TYPI8、TYPI16、TYPI32、または TYPI64 の値に変換する場合、ストリングは以下のフォーマットでなければなりません。

```
[blanks][sign]digits
```

ストリングの構成要素の意味は以下のとおりです。

blanks

オプションの先行空白文字

sign

オプションの正符号 (+) または負符号 (-) 文字。

digits

数字 (0 から 9 まで) の連続シーケンス。少なくとも 1 つの数字が存在している必要があります。

数字のシーケンスの後のストリングには数字以外の文字を含めることができますが、それらの文字の最初のものに達するとすぐに変換は停止します。ストリングは 10 進整数を表すと想定されます。

ストリングが正しくフォーマットされていない場合、IBM MQ は理由コード RC2472 を戻します。

- ストリング・プロパティ値を、データ・タイプ TYPF32 または TYPF64 の値に変換する場合、ストリングは以下のフォーマットでなければなりません。

```
[blanks][sign]digits[.digits][e_char[e_sign]e_digits]
```

ストリングの構成要素の意味は以下のとおりです。

blanks

オプションの先行空白文字

sign

オプションの正符号 (+) または負符号 (-) 文字。

digits

数字 (0 から 9 まで) の連続シーケンス。少なくとも 1 つの数字が存在している必要があります。

e_char

指数文字。「E」か「e」のどちらかです。

e_sign

指数用の、オプションの正符号 (+) または負符号 (-) 文字。

e_digits

指数用の、数字 (0-9) の連続シーケンス。ストリングに指数文字がある場合、1 つ以上の数字がなければなりません。

数字のシーケンスの後、または指数を表すオプションの文字の後のストリングには数字以外の文字を含めることができますが、それらの文字の最初のものに達するとすぐに変換は停止します。ストリングは、10 の累乗の指数を持つ 10 進浮動小数点数を表すと想定されます。

ストリングが正しくフォーマットされていない場合、IBM MQ は理由コード RC2472 を戻します。

- 数値のプロパティ値をストリングに変換する際には、値は、この値に関する ASCII 文字を含むストリングではなく、この値の 10 進数のストリング表現に変換されます。例えば、整数 65 はストリング「A」ではなくストリング「65」に変換されます。
- バイト・ストリングのプロパティ値をストリングに変換する際には、各バイトは、そのバイトを表す 2 つの 16 進文字に変換されます。例えば、バイト配列 {0xF1, 0x12, 0x00, 0xFF} はストリング「F11200FF」に変換されます。

IPQLEN

プロパティ値のタイプと長さを照会します。長さは、MQINQMP 呼び出しの **DataLength** パラメーターで返されます。プロパティ値は戻されません。

ReturnedName バッファーを指定すると、MQCHARV 構造の *VSLength* フィールドは、プロパティ名で埋められます。プロパティ名は戻されません。

反復オプション: 以下は、ワイルドカード文字がある名前を使用した、プロパティの反復に関するオプションです。

IPINQF

指定された名前に一致する最初のプロパティを照会します。この呼び出しの後に、カーソルは返されるプロパティに設定されます。

これがデフォルト値です。

その後、必要に応じて MQINQMP 呼び出しで IPINQC オプションを使用することにより、同じプロパティを再び照会することができます。

プロパティ・カーソルは 1 つしかないことに注意してください。したがって、MQINQMP 呼び出しに指定したプロパティ名を変更すると、カーソルはリセットされます。

このオプションは次のいずれかのオプションが指定されている場合には、無効です。

IPINQN
IPINQC

IPINQN

プロパティ・カーソルからの検索を続行しながら、指定された名前に一致する次のプロパティを照会します。カーソルは、返されたプロパティに進みます。

指定された名前に関する最初の MQINQMP 呼び出しの場合は、指定された名前と一致する最初のプロパティが戻されます。

その後、必要に応じて MQINQMP 呼び出しで IPINQC オプションを使用することにより、同じプロパティを再び照会することができます。

カーソルの下のプロパティが削除されている場合は、MQINQMP は削除されたプロパティより後で次に一致するプロパティを戻します。

反復の進行中に、ワイルドカードと一致するプロパティが追加された場合、そのプロパティは反復の完了までに返される場合もあれば、返されない場合もあります。プロパティは、反復が IPINQF を使用して再始動すると返されます。

反復の進行中に、ワイルドカードと一致するプロパティで削除されたものは、削除後には返されません。

このオプションは次のいずれかのオプションが指定されている場合には、無効です。

IPINQF
IPINQC

IPINQC

プロパティ・カーソルによって指し示されるプロパティの値を取り出します。プロパティ・カーソルによってポイントされるプロパティとは、IPINQF または IPINQN オプションを使って最後に照会されたプロパティです。

メッセージ・ハンドルを再利用する際、MQGET 呼び出しの MQGMO の *MsgHandle* フィールド中でメッセージ・ハンドルを指定する際、または MQPUT 呼び出しの MQPMO 構造体の *OriginalMsgHandle* または *NewMsgHandle* フィールド中でメッセージ・ハンドルを指定する際には、プロパティ・カーソルはリセットされます。

プロパティ・カーソルが未設定の時点で、あるいはプロパティ・カーソルによってポイントされるプロパティが削除された後でこのオプションを使用した場合には、呼び出しが失敗して、完了コード CCFAIL と理由コード RC2471 が戻されます。

このオプションは次のいずれかのオプションが指定されている場合には、無効です。

IPINQF
IPINQN

上記のオプションがどれも必要でない場合には、以下のオプションを使用できます。

IPNONE

この値は、他のオプションが指定されなかったことを示すために使用します。すべてのオプションはデフォルト値であるとみなされます。

IPNONE は、プログラムの文書化を支援します。このオプションは、他のオプションと同時に使用するものではありません。ただしこのオプションの値はゼロであるため、他のオプションと同時に使用してもそれを検出できません。

これは常に入力フィールドです。このフィールドの初期値は IPINQF です。

IPREQCSI (10 桁の符号付き整数)

値が文字ストリングの場合に、照会されるプロパティー値の変換結果の文字セット。これは IPCVAL または IPCTYP の指定時に *ReturnedName* の変換先となる文字セットにもなります。

このフィールドの初期値は CSAPL です。

IPREQENC (10 桁の符号付き整数)

これは IPCVAL または IPCTYP の指定時に、照会されたプロパティー値の変換先となるエンコードです。

このフィールドの初期値は ENNAT です。

IPRE1 (10 桁の符号付き整数)

これは予約フィールドです。このフィールドの初期値は、ブランク文字です。

IPRETCSI (10 桁の符号付き整数)

出力では、MQINQMP 呼び出しの **Type** パラメーターが TYPSTR の場合に、これが戻される値の文字セットです。

IPCVAL オプションが指定され、変換が正常に実行された場合、戻り時に、*ReturnedCCSID* フィールドは渡された値と同じ値になります。

フィールドの初期値は、0 です。

IPRETENC (10 桁の符号付き整数)

出力上に戻される値のエンコードです。

IPCVAL オプションが指定され、変換が正常に実行された場合、戻り時に、*ReturnedEncoding* フィールドは渡された値と同じ値になります。

このフィールドの初期値は ENNAT です。

IPRETNAMCHRP (10 桁の符号付き整数)

照会されるプロパティーの実際の名前。

入力時に、ストリング・バッファーは MQCHARV 構造体の *VSPtr* または *VSOffset* フィールドを使用して渡すことができます。ストリング・バッファーの長さは、MQCHARV 構造の *VSBufsize* フィールドを使用して指定されます。

MQINQMP 呼び出しから戻る際に、ストリング・バッファーが、照会されたプロパティーの名前を完全に入れられる長さである場合は、ストリング・バッファーはこの名前ですべて完了します。MQCHARV 構造の *VSLength* フィールドは、プロパティー名の長さで埋められます。名前の変換が失敗したかどうかにかかわらず、MQCHARV 構造の *VSCCSID* フィールドは埋められ、返される名前の文字セットが示されます。

これは入出力フィールドです。このフィールドの初期値は MQCHARV_DEFAULT です。

IPSID (10 桁の符号付き整数)

これは構造体 ID です。値は次のものでなければなりません。

IPSIDV

メッセージ・プロパティ照会オプション構造の ID。

これは常に入力フィールドです。このフィールドの初期値は IPSIDV です。

IPTYP (10 桁の符号付き整数)

プロパティのデータ・タイプのストリング表現。

MQRFH2 ヘッダー中にプロパティが指定されていて、MQRFH2 dt 属性が認識されない場合は、このフィールドを使用してプロパティのデータ・タイプを判別できます。*TypeString* はコード化文字セット 1208 (UTF-8) で戻され、認識に失敗したプロパティの dt 属性の値の先頭 8 バイトになります。

これは、常に出力フィールドです。このフィールドの初期値は、C プログラミング言語ではヌル・ストリングですが、その他のプログラミング言語では 8 桁のブランク文字です。

IPVER (10 桁の符号付き整数)

これは構造体のバージョン番号です。値は次のものでなければなりません。

IPVER1

メッセージ・プロパティ照会オプション構造のバージョン番号。

以下の定数は、現行バージョンのバージョン番号を指定しています。

IPVERC

メッセージ・プロパティ照会オプション構造の現行バージョン。

これは常に入力フィールドです。このフィールドの初期値は IPVER1 です。

初期値

表 746. MQIPMO のフィールド		
フィールド名	定数の名前	定数の値
IPSID	IPSIDV	'IMPO'
IPVER	IPVER1	1
IPOPT	IPINQF	
IPREQENC	ENNAT	
IPREQCSI	CSAPL	
IPRETENC	ENNAT	
IPRETCSI	0	
IPRE1	0	
IPRETNAMCHRP		
IPTYP		ブランク

RPG 宣言

```
D* MQIMPO Structure
D*
D*
D* Structure identifier
D IPSID      1   4  INZ('IMPO')
D*
D* Structure version number
D IPVER      5   8I 0 INZ(1)
D*
```

```

** Options that control the action of
D* MQINQMP
D IPOPT          9   12I 0 INZ(0)
D*
D* Requested encoding of Value
D IPREQENC       13   16I 0 INZ(273)
D*
** Requested character set identifier
D* of Value
D IPREQCSI       17   20I 0 INZ(-3)
D*
D* Returned encoding of Value
D IPRETENC       21   24I 0 INZ(273)
D*
** Returned character set identifier of
D* Value
D IPRETCSI       25   28I 0 INZ(0)
D*
D* Reserved
D IPRE1          29   32I 0 INZ(0)
D*
D* Returned property name
D* Address of variable length string
D IPRETNAMCHRP   33   48* INZ(*NULL)
D* Offset of variable length string
D IPRETNAMCHRO   49   52I 0 INZ(0)
D* Size of buffer
D IPRETNAMVSBS   53   56I 0 INZ(-1)
D* Length of variable length string
D IPRETNAMCHRL   57   60I 0 INZ(0)
D* CCSID of variable length string
D IPRETNAMCHRC   61   64I 0 INZ(-3)
D*
D* Property data type as a string
D IPTYP         65   72 INZ

```

IBM i

IBM i での MQMD (メッセージ記述子)

概要

目的: MQMD 構造体には、送信側アプリケーションと受信側アプリケーションとの間でメッセージがやり取りされるとき、アプリケーション・データに付随する制御情報が入れます。この構造体は、MQGET、MQPUT、および MQPUT1 呼び出しに指定する入出力パラメーターです。

バージョン: MQMD の現行バージョンは MDVER2 です。これより新しいバージョンの構造体にはのみ存在するフィールドについては、そのフィールドの説明にその旨を記載しています。

提供される COPY ファイルには環境によってサポートされている 最新バージョンの MQMD が含まれます。ただし、MDVER フィールドの初期値は MDVER1 に設定されています。version-1 構造体に存在しないフィールドを使用するには、アプリケーションで、MDVER フィールドを必要なバージョンのバージョン番号に設定する必要があります。

バージョン 1 の構造体の宣言は、MQMD1 という名前で使用できます。

文字セットおよびエンコード: MQMD のデータは、**CodedCharSetId** キュー・マネージャー属性で指定された文字セットと、ENNAT で指定されたローカル・キュー・マネージャーのエンコードになっていなければなりません。ただし、アプリケーションを IBM MQ MQI client として実行する場合は、構造体はクライアントの文字セットとエンコードに従っている必要があります。

送信側と受信側のキュー・マネージャーで使用する文字セットまたはエンコードが違う場合、MQMD のデータは自動的に変換されます。アプリケーションで MQMD を変換する必要はありません。

- [1115 ページの『MQMD の異なるバージョンの使用』](#)
- [1115 ページの『メッセージ・コンテキスト』](#)
- [1116 ページの『メッセージ有効期限』](#)
- [1116 ページの『フィールド』](#)
- [1157 ページの『初期値』](#)
- [1158 ページの『RPG 宣言』](#)

MQMD の異なるバージョンの使用

一般的には、バージョン 1 の MQMD でメッセージ・データの前に MQMDE 構造体を付けると、バージョン 2 の MQMD と同等になります。ただし、MQMDE 構造体のすべてのフィールドにデフォルト値が設定されている場合には MQMDE を省略できます。バージョン 1 の MQMD に MQMDE を加えた場合は、このセクションで後述するように使用します。

- MQPUT 呼び出しおよび MQPUT1 呼び出しでアプリケーションからバージョン 1 の MQMD を提供する場合、オプションとして、メッセージ・データに接頭部 MQMDE を付けることができます。その場合は、MQMD の MDFMT フィールドに FMMDE を設定して、MQMDE が存在することを示します。アプリケーションが MQMDE を提供しない場合、キュー・マネージャーは MQMDE の各フィールドにデフォルト値が設定されたものと見なします。

注：バージョン 2 の MQMD に存在して、バージョン 1 の MQMD に存在しないフィールドのいくつかは、MQPUT 呼び出しおよび MQPUT1 呼び出しに対する入出力フィールドです。ただし、キュー・マネージャーは、MQPUT 呼び出しおよび MQPUT1 呼び出しで出力が生成されても MQMDE の該当するフィールドに値を戻しません。出力値が必要な場合には、そのアプリケーションでバージョン 2 の MQMD を使用する必要があります。

- MQGET 呼び出しでアプリケーションからバージョン 1 の MQMD を提供した場合は、MQMDE の 1 つ以上のフィールドがデフォルト以外の値の場合に限り、キュー・マネージャーから返されるメッセージの先頭に MQMDE が付けられます。MQMD の MDFMT フィールドには、MQMDE が存在することを示す FMMDE という値が設定されます。

キュー・マネージャーが MQMDE の各フィールドに使用するデフォルト値は、[1157 ページの表 748](#) に示す各フィールドの初期値と同じです。

メッセージが伝送キュー上にある場合、MQMD 内のフィールドの一部が特定の値に設定されます。詳細については、[1255 ページの『IBM i での MQXQH \(伝送キュー・ヘッダー\)』](#)を参照してください。

メッセージ・コンテキスト

MQMD の特定のフィールドにはメッセージ・コンテキストが含まれます。一般に、

- ID コンテキスト は、最初にメッセージを書き込んだアプリケーションに関連したものです。
- 起点コンテキスト は、そのメッセージを最後に入れたアプリケーションに関係しています。
- ユーザー・コンテキスト は、そのメッセージを最初に書き込んだアプリケーションに関係しています。

これら 2 つのアプリケーションは同じアプリケーションのこともありますが、異なるアプリケーションであるというケースもあります (例えば、メッセージが 1 つのアプリケーションから別のアプリケーションに転送された場合)。

識別コンテキストと起点コンテキストには通常、前述のような意味がありますが、実際には MQMD 内のいずれのタイプのコンテキスト・フィールドの内容も、メッセージが入れられるときに指定された PM* オプションによって決まります。そのため、識別コンテキストは必ずしもメッセージを最初に入れたアプリケーションに関係しているわけではなく、起点コンテキストも必ずしもメッセージを最後に入れたアプリケーションに関係しているわけではありません。むしろこれは、アプリケーション群の設計によって決まります。

アプリケーションのクラスには、メッセージのコンテキストをまったく変更しない、メッセージ・チャンネル・エージェント (MCA) があります。リモート・キュー・マネージャーからメッセージを受け取る MCA は、MQPUT または MQPUT1 呼び出しでコンテキスト・オプション PMSETA を使用します。これによって受信側の MCA は、メッセージとともに送信側の MCA から送られてきたメッセージ・コンテキストを、変更が加えられていない状態で保存できます。ただし、その結果として、起点コンテキストはメッセージを最後に入れたアプリケーション (受信側の MCA) には関係付けられず、それ以前にメッセージを入れたアプリケーション (発信側のアプリケーション自身であることが多い) に関係付けられることになります。

詳細については、[メッセージのコンテキスト](#)を参照してください。

メッセージ有効期限

ロードされたキュー (オープンされていたキュー) で満了したメッセージは、満了から妥当な時間内に自動的にキューから除去されます。このリリースの IBM MQ の他の新しいフィーチャーにより、以前の製品バージョンよりもロードされたキューをスキャンする頻度が低くなります。しかし、ロードされたキューにある満了したメッセージは、常に妥当な期間内に除去されます。

フィールド

MQMD 構造体には、以下のフィールドが含まれます。フィールドはアルファベット順に説明されています。

MDACC (32 バイトのビット・ストリング)

アカウンティング・トークン。

これは、メッセージの *ID* コンテキストの一部です。メッセージ・コンテキストについての詳細は、[メッセージ・コンテキストおよびコンテキスト情報の制御](#)を参照してください。

MDACC を指定すると、アプリケーションは、メッセージの結果として行われた作業に適切な課金を行うことができます。キュー・マネージャーはこの情報をビット・ストリングとして処理し、その内容は検査しません。

キュー・マネージャーは、この情報を生成するときに、以下のように設定します。

- このフィールドの最初のバイトを、フィールドの後続バイトにある会計情報の長さに設定します。長さは 0 から 30 までの範囲であり、2 進整数として最初のバイトに保管されます。
- 2 番目以降のバイト (長さフィールドに指定) を、環境に応じた会計情報に設定します。
 -  **z/OS** z/OS では、会計情報は以下に設定されます。
 - z/OS バッチでは、JES JOB カードからの会計情報、または EXEC カード内の JES ACCT ステートメントからの会計情報 (コンマの区切り文字は 'X'FF' に変更されます)。この情報は、必要に応じて 31 バイトに切り捨てられます。
 - TSO の場合は、ユーザーのアカウント番号。
 - CICS の場合は、LU 6.2 作業単位 ID (UEPUOWDS) (26 バイト)。
 - IMS の場合は、16 文字の IMS リカバリー・トークンと連結した 8 文字の PSB 名。
 -  **IBM i** IBM i では、会計情報は、ジョブの会計コードに設定されます。
 -  **Linux**  **AIX** AIX and Linux では、会計情報は ASCII 文字による数値のユーザー ID に設定されます。
 -  **Windows** Windows では、会計情報は圧縮形式の Windows NT セキュリティー ID (SID) に設定されます。SID は、*MDUID* フィールドに保管されたユーザー ID を一意的に識別します。SID が *MDACC* フィールドに格納されている場合、6 バイトの Identifier Authority (SID の 3 バイト目以降にあります) は省略されます。例えば、Windows NT SID の長さが 28 バイトである場合は、22 バイトの SID 情報が *MDACC* フィールドに保管されます。
- 最後のバイトを、会計トークン・タイプ (以下の値のいずれか) に設定します。

ATTCIC

CICS LUOW ID。

ATTDOS

PC DOS のデフォルトの会計トークン。

ATTWNT

Windows セキュリティー ID。

ATT400

IBM i 会計トークン。

ATTUNIX

AIX and Linux の数値 ID。

ATTUSR

ユーザー定義の会計トークン。

ATTUNK

不明な会計トークン・タイプ。

会計トークン・タイプは以下の環境でのみ、明示的な値に設定されます。

-  AIX
-  IBM i
-  Windows

および、これらのシステムに接続された IBM MQ MQI clients。

これ以外の環境では、会計トークン・タイプは値 ATTUNK に設定されます。これらの環境では、MDPAT フィールドを使用して、受け取った会計トークンのタイプを推測できます。

- その他のすべてのバイトを 2 進ゼロに設定します。

MQPUT および MQPUT1 呼び出しの場合、**PMO** パラメーターに PMSETI または PMSETA が指定されていれば、これは入出力フィールドです。PMSETI または PMSETA が指定されていない場合、入力においてこのフィールドは無視され、出力専用フィールドになります。メッセージ・コンテキストについての詳細は、[メッセージ・コンテキストおよびコンテキスト情報の制御](#)を参照してください。

MQPUT または MQPUT1 呼び出しが正常に完了すると、メッセージがキューに書き込まれた場合、このフィールドには、そのメッセージと共に送信された MDACC が入ります。これは、メッセージが保存された場合 (保存パブリケーションについて詳しくは、[1180 ページの『IBM i での MQPMO \(メッセージ書き込みオプション\)』](#)にある PMRET の説明を参照してください)、そのメッセージと共に保持された MDACC の値になりますが、メッセージがパブリケーションとしてサブスクライバーに送信された場合には MDACC として使用されません。これは、サブスクライバーが提供する値が、サブスクライバーに送信されるすべてのパブリケーションにおいて MDACC をオーバーライドするためです。メッセージにコンテキストがない場合、フィールドは完全に 2 進ゼロになります。

これは、MQGET 呼び出しの出力フィールドです。

このフィールドは、キュー・マネージャーの文字セットに基づいた変換の対象ではありません。このフィールドは、文字ストリングとしてではなく、ビット・ストリングとして扱われます。

キュー・マネージャーは、フィールドにある情報については、何も処理しません。アプリケーションは、会計の目的で情報を使用する場合に、情報を解釈する必要があります。

MDACC フィールドには、以下に示す特別な値が使用されることがあります。

ACNONE

アカウンティング・トークンが指定されていません。

値は、フィールドの長さについては 2 進ゼロです。

このフィールドの長さは、LNACCT で指定されます。このフィールドの初期値は ACNONE です。

MDAID (32 バイトの文字ストリング)

ID に関連するアプリケーション・データ。

これは、メッセージの ID コンテキストの一部です。メッセージ・コンテキストについての詳細は、[メッセージ・コンテキストおよびコンテキスト情報の制御](#)を参照してください。

MDAID は、アプリケーション・スイートによって定義される情報で、メッセージやその発信元に関する追加の情報を提供するために使用することができます。キュー・マネージャーはこの情報を文字データとして扱いますが、そのフォーマットの定義はしません。キュー・マネージャーは、この情報を生成するときに、全体をブランクにします。

MQPUT および MQPUT1 呼び出しの場合、**PMO** パラメーターに PMSETI または PMSETA が指定されていれば、これは入出力フィールドです。ヌル文字がある場合は、ヌル文字およびその後に続く文字は、キュー・マネージャーによってブランクに変換されます。PMSETI または PMSETA が指定されていない

い場合、入力においてこのフィールドは無視され、出力専用フィールドになります。メッセージ・コンテキストについての詳細は、[メッセージ・コンテキストおよびコンテキスト情報の制御](#)を参照してください。

MQPUT または MQPUT1 呼び出しが正常に完了すると、メッセージがキューに書き込まれた場合、このフィールドには、そのメッセージと共に送信された MDAID が入ります。これは、メッセージが保存された場合 (保存パブリケーションについて詳しくは、PMRET の説明を参照してください)、そのメッセージと共に保持された MDAID の値になりますが、メッセージがパブリケーションとしてサブスクライバーに送信された場合には MDAID として使用されません。これは、サブスクライバーが提供する値が、サブスクライバーに送信されるすべてのパブリケーションにおいて MDAID をオーバーライドするためです。メッセージがコンテキストを持っていない場合、フィールドは完全にブランクになります。

これは、MQGET 呼び出しの出力フィールドです。このフィールドの長さは LNAIDD によって指定されます。このフィールドの初期値は 32 個のブランク文字です。

MDAOD (4 バイトの文字ストリング)

発生元に関係するアプリケーション・データ。

これは、メッセージの起点コンテキストの一部です。メッセージ・コンテキストについての詳細は、[メッセージ・コンテキストおよびコンテキスト情報の制御](#)を参照してください。

MDAOD は、アプリケーション・スイートにより定義される情報で、メッセージの発信元についての追加情報を提供するのに使用できます。例えば、ID データが信頼できるかどうかを示すために、適切なユーザー権限で実行されているアプリケーションにより設定が可能です。

キュー・マネージャーはこの情報を文字データとして扱いますが、そのフォーマットの定義はしません。キュー・マネージャーは、この情報を生成するときに、全体をブランクにします。

MQPUT および MQPUT1 呼び出しの場合、**PMO** パラメーターに または **PMSETA** が指定されていれば、これは入出力フィールドです。フィールド内でヌル文字より後の情報はすべて破棄されます。ヌル文字およびその後に続く文字は、キュー・マネージャーによってブランクに変換されます。**PMSETA** が指定されていない場合、入力においてこのフィールドは無視され、出力専用フィールドになります。

MQPUT または MQPUT1 呼び出しが正常に完了すると、メッセージがキューに書き込まれた場合、このフィールドには、そのメッセージと共に送信された MDAOD が入ります。これは、メッセージが保存された場合 (保存パブリケーションについて詳しくは、PMRET の説明を参照してください)、そのメッセージと共に保持された MDAOD の値になりますが、メッセージがパブリケーションとしてサブスクライバーに送信された場合には MDAOD として使用されません。これは、サブスクライバーが提供する値が、サブスクライバーに送信されるすべてのパブリケーションにおいて MDAOD をオーバーライドするためです。メッセージがコンテキストを持っていない場合、フィールドは完全にブランクになります。

これは、MQGET 呼び出しの出力フィールドです。このフィールドの長さは LNAORD によって指定されます。このフィールドの初期値は 4 個のブランク文字です。

MDBOC (10 桁の符号付き整数)

バックアウトのカウンター。

これは、メッセージが、作業単位の一部として MQGET 呼び出しから事前に戻されて、その後バックアウトされた回数のカウントです。メッセージの内容に基づいて処理エラーを検出する際に、アプリケーションへの補助機能として提供されます。カウントには、GMBRW* オプションのいずれかを指定する MQGET 呼び出しは含まれません。

カウントの正確度は、**HardenGetBackout** キュー属性の影響を受けます。[1380 ページの『キューの属性』](#)を参照してください。

これは、MQGET 呼び出しの出力フィールドです。MQPUT および MQPUT1 呼び出しでは、無視されます。このフィールドの初期値は 0 です。

MDCID (24 バイトのビット・ストリング)

関連 ID。

これは、バイト・ストリングで、1 つのメッセージを別のメッセージと関連付けたり、メッセージをアプリケーションが実行している他の作業と関連付けたりするために、アプリケーションによって使用で

きます。相関 ID はメッセージの永続的なプロパティであり、キュー・マネージャーを再始動しても保持されます。相関 ID は文字ストリングではなくバイト・ストリングであるため、あるキュー・マネージャーから別のキュー・マネージャーにメッセージが転送され、文字セットが異なっている場合でも、相関 ID は変換されません。

MQPUT 呼び出しおよび MQPUT1 呼び出しの場合、アプリケーションは任意の値を指定できます。キュー・マネージャーはこの値をメッセージと一緒に送信し、メッセージの取得要求を出したアプリケーションに配信します。

アプリケーションで PMNCID を指定した場合、キュー・マネージャーは固有の相関 ID を生成します。この相関 ID は、メッセージと共に送信されるほか、MQPUT 呼び出しまたは MQPUT1 呼び出しからの出力で送信側アプリケーションに返されます。

生成されたこの相関 ID がメッセージと共に保持されるのは、MQSUB 呼び出しで渡される MQSD の SDCID フィールドで CINONE を指定するサブスクライバーに、メッセージがパブリケーションとして送信されるときに、メッセージが保存され、相関 ID として使用される場合です。

保存パブリケーションの詳細については、[1180 ページの『IBM i での MQPMO \(メッセージ書き込みオプション\)』](#)を参照してください。

キュー・マネージャーまたはメッセージ・チャネル・エージェントは、レポート・メッセージを生成するとき、元のメッセージの MDREP フィールド、すなわち、ROCMTC または ROPCI のいずれかによって指定された方法で MDCID フィールドを設定します。レポート・メッセージを生成するアプリケーションも、上記の指定を行います。

MQGET 呼び出しの場合、MDCID は、キューから取得する特定のメッセージを選択するために使用できる 5 つのフィールドの 1 つです。このフィールドに値を指定する方法の詳細については、MDMID フィールドを参照してください。

相関 ID として CINONE を指定すると、MOCORI を指定しなかった場合と同じ結果になります。つまり、すべての相関 ID が一致することになります。

GMMUC オプションが MQGET 呼び出しの **GMO** パラメーターで指定されている場合、このフィールドは無視されます。

MQGET 呼び出しから戻った時点で、MDCID フィールドには、返されたメッセージ (それがあつた場合) の相関 ID が設定されます。

次の特別な値を使用できます。

CINONE

相関 ID は指定されません。

値は、フィールドの長さについては 2 進ゼロです。

CINEWS

メッセージは、新しいセッションの先頭です。

この値は、新規セッションの開始、つまり、メッセージの新規シーケンスの始まりとして CICS bridge により認識されます。

MQGET 呼び出しの場合、これは入出力フィールドです。MQPUT および MQPUT1 呼び出しでは、PMNCID を指定しなかった場合は入力フィールド、PMNCID を指定した場合は出力フィールドです。このフィールドの長さは、LNCID で指定されます。このフィールドの初期値は CINONE です。

MDCSI (10 桁の符号付き整数)

これは、メッセージにある文字データの文字セット ID を指定します。

注: 呼び出しのパラメーターとして指定する MQMD 構造体およびその他の IBM MQ データ構造体の文字データは、キュー・マネージャーの文字セットでなければなりません。これは、キュー・マネージャーの **CodedCharSetId** 属性によって定義されます。この属性の詳細については、[1412 ページの『IBM i でのキュー・マネージャーの属性』](#)を参照してください。

以下のような特別な値を使用することができます。

CSQM

キュー・マネージャーの文字セット ID。

メッセージ内の文字データは、キュー・マネージャーの文字セットになります。

MQPUT および MQPUT1 呼び出しで、キュー・マネージャーは、メッセージで送信された MQMD 内のこの値を、キュー・マネージャーの実際の文字セット ID に変更します。したがって、値 CSQM が MQGET 呼び出しによって戻されることはありません。

CSINHT

この構造体の文字セット ID を継承する。

メッセージ内の文字データは、この構造体と同じ文字セットです。つまり、キュー・マネージャーの文字セットです。(MQMD の場合のみ、CSINHT は CSQM と同じ意味を持ちます。)

キュー・マネージャーは、メッセージで送信された MQMD 内のこの値を、MQMD の実際の文字セット ID に変更します。エラーが発生しない限り、値 CSINHT が MQGET 呼び出しによって返されることはありません。

MQMD の MDPAT フィールドの値が ATBRKR の場合、CSINHT は使用できません。

CSEMBD

組み込み文字セット ID。

メッセージ内の文字データの文字セットは、メッセージ・データそのものに ID が含まれている文字セットになります。メッセージのデータに組み込まれていて、そのデータの別の部分に適用される文字セット ID はいくつあっても構いません。この値は、いくつかの文字セットによるデータが混在した PCF メッセージでは必ず使用します。PCF メッセージの形式名は FMPCF です。

この値は、MQPUT 呼び出しおよび MQPUT1 呼び出しに対してのみ指定してください。この値を MQGET 呼び出しに指定すると、メッセージが変換されなくなります。

MQPUT 呼び出しおよび MQPUT1 呼び出しで、キュー・マネージャーは、メッセージと一緒に送信された MQMD 内の値 CSQM および CSINHT を前述のように変更しますが、MQPUT 呼び出しまたは MQPUT1 呼び出しで指定された MQMD は変更しません。指定された値について、それ以外の検査は行われません。

メッセージを検索するアプリケーションは、このフィールドとアプリケーションの予期する値とを比較することが必要です。値が異なる場合、アプリケーションは、メッセージ内の文字データを変換することが必要になる場合があります。

GMCONV オプションが MQGET 呼び出しで指定されている場合、このフィールドは入出力フィールドです。アプリケーションが指定した値は、必要に応じてメッセージ・データを変換しなければならないコード化文字セット ID の値です。変換が正しく行われるか、変換が不要であれば、この値は変更されません(ただし、値 CSQM または CSINHT は、実際の値に変換されます)。変換が失敗したときは、MQGET 呼び出しの後の値は、アプリケーションに戻されるメッセージのうち変換されなかったメッセージのコード化文字セット ID を表しています。

それ以外の場合、MQGET 呼び出しでは出力フィールド、MQPUT および MQPUT1 呼び出しでは入力フィールドです。このフィールドの初期値は CSQM です。

MDENC (10 桁の符号付き整数)

メッセージ・データの数値エンコード。

ここでは、メッセージ内の数値データの数値エンコードを指定します。これは、MQMD 構造体自体の数値データには適用されません。数値エンコード方式では、2 進整数、パック 10 進整数、および浮動小数点数の表記が定義されています。

MQPUT または MQPUT1 呼び出しでは、アプリケーションは、このフィールドをデータに適切な値に設定する必要があります。キュー・マネージャーは、フィールドが有効かどうかをチェックしません。以下のような特殊値が定義されます。

ENNAT

マシン固有のエンコード。

このコード・エンコードは、アプリケーションが実行されているプログラミング言語およびマシンのデフォルトになります。

注: この定数の値は、プログラム言語と環境によって異なります。このため、アプリケーションは、実行する環境に適したヘッダー・ファイル、マクロ・ファイル、COPY ファイル、および INCLUDE ファイルを使用してコンパイルする必要があります。

メッセージを書き込んだアプリケーションは、通常 ENNAT を指定する必要があります。メッセージを検索するアプリケーションは、このフィールドと値 ENNAT を比較することが必要です。値が異なる場合、アプリケーションは、メッセージ内の数値データを変換することが必要になる場合があります。GMCONV オプションを使用すると、MQGET 呼び出しの処理の一部としてメッセージを変換することをキュー・マネージャーに要求できます。

GMCONV オプションが MQGET 呼び出しで指定されている場合、このフィールドは入出力フィールドです。アプリケーションで指定する値は、必要に応じてメッセージ・データが変換された後のコードです。変換が成功したか不要の場合、値は変化しません。変換が失敗したときは、MQGET 呼び出しの後の値は、アプリケーションに戻された未変換メッセージのエンコードを表しています。

それ以外の場合、MQGET 呼び出しでは出力フィールド、MQPUT および MQPUT1 呼び出しでは入力フィールドです。このフィールドの初期値は ENNAT です。

MDEXP (10 桁の符号付き整数)

メッセージ存続期間。

これは、メッセージを書き込むアプリケーションで設定される時間で、10 分の 1 秒単位で表されます。この時間が経過するまでに宛先キューからメッセージが除去されなかった場合、そのメッセージは廃棄の対象となります。

メッセージが宛先キューに存在した時間に応じて値は減少します。書き込みがリモート・キューに対するものであるとき、値は中間の伝送キューに存在した時間に対応します。伝送にかなりの時間がかかった場合は、メッセージ・チャネル・エージェントがその時間に応じて減算する可能性もあります。同様に、このメッセージを別のキューに送るアプリケーションも、長時間にわたってメッセージを保持した場合には、必要に応じて値を減分することがあります。しかし、満了時間は概数として扱われるので、短い時間間隔の調節のためにこの値を減分する必要はありません。

アプリケーションが MQGET 呼び出しを用いてメッセージを取り出したとき、MDEXP フィールドは、元の有効期限の残りの時間を表します。

メッセージの有効期限を過ぎると、メッセージはキュー・マネージャーによって廃棄される対象となります。このメッセージは、現在の実装では、ブラウズまたは非ブラウズ MQGET 呼び出しが行われると廃棄されます (まだ満了していなければメッセージが返されます)。例えば、MQGMO の GMMO フィールドが MONONE に設定された非ブラウズ MQGET 呼び出しが FIFO 方式のキューから読み取りを行う場合、満了していないメッセージが最初に出現するまで、満了したメッセージはすべて削除されます。優先順位方式のキューで同じ呼び出しを発行した場合は、満了していない最初のメッセージより先にキューに到着した満了したメッセージのうち、優先順位が等しいメッセージとそれより優先順位が高いメッセージが廃棄されます。

ブラウズまたは非ブラウズのどちらの MQGET 呼び出しでも、満了したメッセージがアプリケーションに返されることはないので、MQGET 呼び出しが正常に終了した後、メッセージ記述子の MDEXP フィールドの値はゼロより大きい、特殊値 EIULIM になります。

メッセージをリモート・キューに書き込む場合、メッセージは、宛先キューに到達する前の中間伝送キューにある間に満了してしまう (そして廃棄される) 可能性もあります。

メッセージが ROEXP* レポート・オプションのうちいずれかを指定した場合は、満了したメッセージが廃棄されるときにレポートが生成されます。オプションがまったく指定されていない場合、そうしたレポートは生成されません。指定時間の経過後は、このメッセージは関係がなくなったと見なされます (後のメッセージに置き換わったと考えられるため)。

有効期限に基づいてメッセージを廃棄する他のプログラムはいずれも、要求に応じて、該当のレポート・メッセージを送らなければなりません。

注:

1. メッセージがゼロの MDEXP 時間を指定して書き込まれる場合、MQPUT または MQPUT1 呼び出しは、理由コード RC2013 を戻して失敗します。この場合、レポート・メッセージは生成されません。
2. 有効期限を過ぎたメッセージが、実際に廃棄されないこともあるので、有効期限を過ぎても取り出しの対象にならないメッセージがキューに入っている可能性もあります。それにもかかわらず、これらのメッセージがキュー内のメッセージとしてカウントされる目的は、キュー・サイズによるトリガーなどです。
3. 満了レポートは、廃棄の対象となったときでなく、要求に応じてメッセージが実際に廃棄されるときに生成されます。
4. 要求に応じて、満了メッセージの廃棄および満了レポートを生成するのは、アプリケーションの作業単位の一部ではありません。メッセージが、作業単位で操作される MQGET 呼び出しの結果、廃棄されるようにスケジュールされていたとしても同じです。
5. ほとんど満了しているメッセージが作業単位の中で MQGET 呼び出しによって取り出され、その後でその作業単位がバックアウトされると、メッセージが廃棄の対象になり、再び取り出すことができないようになることもあります。
6. ほぼ満了したメッセージが GMLK を指定した MQGET 呼び出しによりロックされる場合、そのメッセージは、GMMUC を指定した MQGET 呼び出しにより検索される前に廃棄対象にすることができます。これが起きる場合は、その後の MQGET 呼び出しにおいて、理由コード RC2034 が戻されます。
7. 有効期限時刻がゼロよりも大きい要求メッセージを取り出す場合、アプリケーションは応答メッセージを送信するときに以下のいずれかのアクションを実行できます。
 - 有効期限までの残り時間を、要求メッセージから応答メッセージにコピーする。
 - 応答メッセージ中の有効期限を、ゼロより大きい明示的な値に設定する。
 - 応答メッセージ中の満了時間を EIULIM に設定する。

実行されるアクションは、アプリケーション群の設計によって決まります。ただし、送達不能 (未配布メッセージ) キューにメッセージを書き込むときのデフォルト・アクションは、メッセージの残りの満了期間から引き続き減算することです。
8. トリガー・メッセージは常に EIULIM で生成されます。
9. FMXQH の MDFMT 名があるメッセージ (通常は伝送キュー上) には、MQXQH 内に 2 番目のメッセージ記述子があります。したがって、メッセージに関連した 2 つの MDEXP フィールドがあります。この場合、以下の点に注意してください。
 - アプリケーションがリモート・キューにメッセージを書き込む場合、キュー・マネージャーは、メッセージを最初にローカル伝送キューに入れ、MQXQH 構造を用いてアプリケーション・メッセージ・データに接頭部を付加します。キュー・マネージャーは、2 つの MDEXP フィールドを、アプリケーションによる指定値と同じ値に設定します。

アプリケーションがローカル伝送キューにメッセージを直接書き込む場合は、メッセージ・データは常に MQXQH 構造体で始まり、形式名は FMXQH でなければなりません (ただし、キュー・マネージャーがこの設定を実行するわけではありません)。この場合、アプリケーションは 2 つの MDEXP フィールドを同じ値に設定する必要はありません。 (キュー・マネージャーは、MQXQH 内のフィールド MDEXP が有効な値を含むかどうか、あるいはメッセージデータがそのフィールドを含めるのに十分な長さであるかどうかさえもチェックしません。)
 - FMXQH の MDFMT 名があるメッセージがキュー (これが通常のキューであるか伝送キューであるかにかかわらず) から検索されるとき、キュー・マネージャーは、キュー上で待機する時間を指定したこれらの MDEXP フィールドを両方とも減分します。メッセージ・データに MQXQH の MDEXP フィールドを入れるのに十分な長さが無い場合、エラーは発生しません。
 - キュー・マネージャーは、別個のメッセージ記述子 (つまり、MQXQH 構造体内部に埋め込まれているメッセージ記述子以外のもの) の MDEXP フィールドを使用して、メッセージが廃棄するものとして適格かどうかをテストします。
 - 2 つの MDEXP フィールドの初期値が異なっていた場合、別個のメッセージが取り出された時点でそのメッセージ記述子の内 MDEXP 時間は、ゼロより大きい (したがってメッセージは廃棄の対象にならない) 可能性があります。ただし MQXQH 内の MDEXP フィールドで規定された時間は経過しています。この場合、MQXQH の MDEXP フィールドはゼロに設定されます。

以下のような特殊値が認識されます。

EIULIM

制限なしの存続時間。

メッセージは、無制限の満了時間を指定されています。

これは、MQGET 呼び出しでは出力フィールド、MQPUT および MQPUT1 呼び出しでは入力フィールドです。このフィールドの初期値は EIULIM です。

MDFB (10 桁の符号付き整数)

フィードバックまたは理由コード。

これは、レポートの性質を示すためにメッセージのタイプ MTRPRT と共に使用されるもので、そのタイプのメッセージと共に使用する場合にのみ意味をなします。このフィールドには、値 FB* のいずれか、または値 RC* のいずれかを指定できます。フィードバック・コードは、以下のようにグループ化されています。

FBNONE

フィードバックが提供されていない。

FBSFST

システム生成のフィードバックの最低値。

FBSLST

システム生成のフィードバックの最高値。

システム生成のフィードバック・コードの範囲 FBSFST から FBSLST には、このセクションで後述する一般的なフィードバック・コード (FB*) および、メッセージを宛先キューに書き込めないときに戻る可能性のある理由コード (RC*) も含まれています。

FBAFST

アプリケーション生成のフィードバックの最低値。

FBALST

アプリケーション生成のフィードバックの最高値。

レポート・メッセージを生成するアプリケーションでは、キュー・マネージャーまたはメッセージ・チャンネル・エージェントが生成するレポート・メッセージをシミュレートしない場合は、システム範囲 (FBQUIT 以外) のフィードバック・コードを使用しないでください。

MQPUT または MQPUT1 呼び出しでは、指定される値は FBNONE であるか、またはその値がシステム範囲かアプリケーション範囲のいずれかになければなりません。これは、MDMT の値には関係なく検査されます。

一般的なフィードバック・コード:

FBCOA

宛先キューでの到達の確認 (ROCOA 参照)。

FBCOD

受信側アプリケーションへの送達確認 (ROCOD 参照)。

FBEXP

メッセージが満了しました。

メッセージは、有効期限が切れる前に宛先キューから除去されなかったため、廃棄されました。

FBPAN

アクションの正常終了通知 (ROPAN 参照)。

FBNAN

アクションの異常終了通知 (RONAN 参照)。

FBQUIT

アプリケーションを終了してください。

実行中のアプリケーション・プログラムのインスタンス数を制御するために、ワークロード・スケジューリング・プログラムによってのみ使用されます。このフィードバック・コードと共に

MTRPRT メッセージをアプリケーション・プログラムのインスタンスに送信することにより、そのインスタンスに処理を停止する必要があることを指示します。しかし、この規則の順守はアプリケーション側の問題であり、キュー・マネージャーでは強制しません。

IMS ブリッジのフィードバック・コード: IMS ブリッジは、ゼロ以外の IMS-OTMA センス・コードを受け取ると、そのセンス・コードを 16 進から 10 進に変換して、値 FBIERR (300) を追加したものを、応答メッセージの MDFB フィールドに入れます。このため、IMS-OTMA エラーが発生した場合は、フィードバック・コードの値は FBIFST (301) から FBILST (399) の範囲になります。

IMS ブリッジが生成するフィードバック・コードは次のとおりです。

FBDLZ

データ長がゼロ。

セグメント長が、メッセージのアプリケーション・データにおいてゼロであった。

FBDLN

データ長が負の値。

セグメント長が、メッセージのアプリケーション・データにおいて負であった。

FBDLTB

データ長が大きすぎます。

メッセージのアプリケーション・データのセグメント長が大きすぎます。

FBBUFO

バッファ・オーバーフロー。

長さフィールドのどれか 1 つの値が原因で、データがメッセージ・バッファからオーバーフローしました。

FBLOB1

1 バイト分の長さエラー。

長さフィールドのどれか 1 つの値が、1 バイト短すぎます。

FBIIH

MQIIH 構造体が無効、または欠落しています。

MQMD の MDFMT フィールドには FMIMS が指定されていますが、メッセージは有効な MQIIH 構造体で始まっていません。

FBNAFI

ユーザー ID のユーザーには、IMS を使用する許可がありません。

メッセージ記述子 MQMD に含まれているユーザー ID、または MQIIH 構造体の IIAUT フィールドに含まれているパスワードが、IMS ブリッジで実行された検証に失敗しました。その結果、メッセージは IMS に渡されませんでした。

FBIERR

IMS から予期しないエラーが戻されました。

予期しないエラーが IMS から戻されました。エラーについては詳しくは、IMS ブリッジが存在しているシステムの IBM MQ エラー・ログを調べてください。

FBIFST

IMS 生成のフィードバックの最低値。

IMS 生成のフィードバック・コードは、FBIFST (300) から FBILST (399) の範囲を占めています。IMS-OTMA センス・コード自体は、MDFB から FBIERR を引いた値です。

FBILST

IMS 生成のフィードバックの最高値。

CICS ブリッジのフィードバック・コード: CICS bridge が生成するフィードバック・コードは次のとおりです。

FBCAAB

アプリケーションが異常終了しました。

メッセージ内で指定したアプリケーション・プログラムが異常終了しました。このフィードバック・コードは、MQDLH 構造体の DLREA フィールドのみに戻されます。

FBCANS

アプリケーションを開始できません。

メッセージ内で指定したアプリケーション・プログラムに関する EXEC CICS LINK が失敗しました。このフィードバック・コードは、MQDLH 構造体の DLREA フィールドのみに戻されます。

FBCBRF

通常のエラー処理を完了せずに CICS bridge が異常終了しました。

FBCCSE

文字セット ID が無効です。

FBCIHE

CICS 情報ヘッダー構造体が欠落しているか、または無効です。

FBCCAE

CICS commarea の長さが無効です。

FBCCIE

相関 ID が無効です。

FB CDLQ

送達不能キューが使用不可です。

CICS bridge・タスクが、この要求に対する応答を送達不能キューにコピーできませんでした。要求はバックアウトされました。

FBCENE

エンコードが無効です。

FBCINE

CICS bridge で予期しないエラーが発生しました。

このフィードバック・コードは、MQDLH 構造体の DLREA フィールドのみに戻されます。

FBCNTA

ユーザー ID が許可されないか、パスワードが無効です。

このフィードバック・コードは、MQDLH 構造体の DLREA フィールドのみに戻されます。

FBCUBO

作業単位がバックアウトされました。

以下のいずれかの理由により、作業単位がバックアウトされました。

- 同じ作業単位内の他の要求の処理中に障害が検出された。
- 作業単位の処理中に CICS の異常終了が発生した。

FBCUWE

作業単位制御フィールド CIUOW が無効です。

MQ 理由コード: 例外レポート・メッセージの場合は、MDFB に MQ 理由コードが格納されます。代表的な理由コードは次のとおりです。

RC2051

(2051, X'803') このキューでは書き込み呼び出しが使用禁止になっています。

RC2053

(2053, X'805') キューには既に最大数のメッセージが入っています。

RC2035

(2035, X'7F3') アクセスは許可されません。

RC2056

(2056, X'808') ディスク上にキューのためのスペースがありません。

RC2048

(2048, X'800') キューは永続的なメッセージをサポートしていません。

RC2031

(2031, X'7EF') メッセージ長がキュー・マネージャーの最大許容長より大きいです。

RC2030

(2030, X'7EE') メッセージの長さが、キューの最大許容数より大きいです。

これは、MQGET 呼び出しでは出力フィールド、MQPUT および MQPUT1 呼び出しでは入力フィールドです。このフィールドの初期値は FBNONE です。

MDFMT (8 バイトの文字ストリング)

メッセージ・データの形式名。

これは、メッセージの送信側が受信側に対してメッセージ内のデータの性質を示すために使用できる名前です。この名前には、キュー・マネージャーの文字セットにある文字はすべて指定できますが、指定する名前を次のものに制限することをお勧めします。

- A から Z までの英大文字
- 0 から 9 までの数字

上記以外の文字が使用されている場合は、送信側および受信側のキュー・マネージャーの文字セットの間で名前を変換できないこともあります。

名前は、フィールドの長さまでブランクを埋め込む必要があります。あるいは、フィールドの終わりになる前に名前を終了させるためにヌル文字を使用します。ヌル文字およびそれに続く文字は、ブランクと見なされます。名前の前または途中にはブランクを指定しないでください。MQGET 呼び出しの場合は、キュー・マネージャーは、フィールドの長さまでブランクを埋め込んだ名前を戻します。

キュー・マネージャーは、名前が上述の推奨事項に準じているかどうかは検査しません。

大文字、小文字、および大文字小文字混合の「MQ」で始まる名前には、キュー・マネージャーで定義された意味があるため、ユーザー独自の形式にはこの文字で始まる名前を使用しないでください。キュー・マネージャーの組み込み形式は次のとおりです。

FMNONE

形式名がありません。

データの性質が未定義です。これは、GMCONV オプションを使用してメッセージをキューから取り出すときにデータを変換できないことを意味します。

GMCONV が MQGET 呼び出しで指定されており、メッセージ内のデータの文字セットまたはエンコードが **MSGDSC** パラメーターで指定されているものと異なる場合、メッセージは次の完了コードと理由コードで戻されます (その他のエラーが発生しないと想定します)。

- FMNONE データがメッセージの先頭にある場合は、完了コード CCWARN および理由コード RC2110。
- FMNONE データがメッセージの終わりにある (つまり、1 つ以上の MQ ヘッダー構造体が前にある) 場合は、完了コード CCOK および理由コード RCNONE。この場合、MQ ヘッダー構造体は、要求されている文字セットとエンコード方式に変換されます。

FMADMN

コマンド・サーバー要求/応答メッセージ。

メッセージは、プログラマブル・コマンド・フォーマット (PCF) のコマンド・サーバー要求または応答メッセージです。GMCONV オプションを MQGET 呼び出しに指定している場合は、この形式のメッセージを変換することができます。プログラマブル・コマンド・フォーマット・メッセージの使い方について詳しくは、[プログラマブル・コマンド・フォーマットの使用](#)を参照してください。

FMCICS

CICS 情報ヘッダー。

メッセージ・データの先頭には CICS 情報ヘッダーの MQCIH があり、その後にアプリケーション・データがあります。アプリケーション・データの形式名は、MQCIH 構造体の CIFMT フィールドに指定されています。

FMCMND1

タイプ 1 のコマンド応答メッセージ。

このメッセージは、オブジェクト・カウント、完了コード、および理由コードが入っている MQSC コマンド・サーバー応答メッセージです。GMCONV オプションを MQGET 呼び出しに指定している場合は、この形式のメッセージを変換することができます。

FMCMND2

タイプ 2 のコマンド応答メッセージ。

このメッセージは、要求されたオブジェクト (複数も可) についての情報が入っている MQSC コマンド・サーバー応答メッセージです。GMCONV オプションを MQGET 呼び出しに指定している場合は、この形式のメッセージを変換することができます。

FMDLH

送達不能ヘッダー。

メッセージ・データは送達不能ヘッダー MQDLH で始まります。元のメッセージからのデータは、MQDLH 構造体のすぐ後に続きます。元のメッセージ・データの形式名は、MQDLH 構造体の DLFMT フィールドで指定されています。この構造体について詳しくは、[1068 ページの『MQDLH \(デッドレターヘッダー\) on IBM i』](#)を参照してください。GMCONV オプションを MQGET 呼び出しに指定している場合は、この形式のメッセージを変換することができます。

FMDLH の MDFMT を持つメッセージの場合、COA および COD レポートは生成されません。

FMDH

配布リスト・ヘッダー。

メッセージ・データは、配布リスト・ヘッダー MQDH で始まります。このデータの中には、MQOR レコードおよび MQPMR レコードの配列などがあります。配布リスト・ヘッダーの後に補足データが続くこともあります。補足データ (存在する場合) の形式は、MQDH 構造体の DHFMT フィールドで指定されています。この構造体について詳しくは、[1063 ページの『IBM i での MQDH \(配布ヘッダー\)』](#)を参照してください。MQGET 呼び出しで GMCONV オプションが指定されていれば、FMDH 形式のメッセージを変換することができます。

FMEVNT

イベント・メッセージ。

メッセージは、発生したイベントを報告する MQ イベント・メッセージです。イベント・メッセージの構造は、プログラマブル・コマンドの構造と同じです。この構造について詳しくは、[コマンドおよび応答の構造](#)を参照してください。イベントの詳細については、[イベント・モニター](#)を参照してください。

GMCONV オプションが MQGET 呼び出しで指定されていれば、バージョン 1 のイベント・メッセージを変換することができます。

FMIMS

IMS 情報ヘッダー。

メッセージ・データは IMS 情報ヘッダー MQIIH で始まり、その後にアプリケーション・データが続きます。アプリケーション・データの形式名は、MQIIH 構造体の IIFMT フィールドに指定されています。GMCONV オプションを MQGET 呼び出しに指定している場合は、この形式のメッセージを変換することができます。

FMIMVS

IMS 可変長ストリング。

メッセージは IMS 可変長ストリングで、形式は 11zzccc です。各部分の詳細は次のとおりです。

11

IMS 可変長ストリング項目の合計長を指定する長さ 2 バイトのフィールドです。合計長は、「11 (2 バイト) + zz (2 バイト) + 文字ストリングの長さ」となります。11 は、MDENC フィールドに指定されたエンコードで表された 2 バイトの 2 進整数を示します。

zz

IMS にとって有効なフラグを含む 2 バイト・フィールドです。zz は 2 つの 1 バイトのビット・ストリング・フィールドから成るバイト・ストリングを示し、送信しても送信側と受信側の間でその内容が変わることはありません (つまり、zz は変換の影響を受けません)。

ccc

可変長文字ストリングを示します。長さは 11-4 文字です。ccc は、MDCSI フィールドで指定された文字セットで表されます。

GMCONV オプションを MQGET 呼び出しに指定している場合は、この形式のメッセージを変換することができます。

FMMDE

拡張メッセージ記述子。

メッセージ・データは、拡張メッセージ記述子 MQMDE で始まります。このデータの後に他のデータ (通常はアプリケーション・メッセージ・データ) が続くこともあります。MQMDE の後に続くデータの形式名、文字セット、およびエンコードは、MQMDE の MEFMT フィールド、MECSI フィールド、および MEENC フィールドでそれぞれ与えられます。この構造体の詳細については、[1159 ページの『IBM i での MQMDE \(拡張メッセージ記述子\)』](#)を参照してください。GMCONV オプションを MQGET 呼び出しに指定している場合は、この形式のメッセージを変換することができます。

FMPCF

プログラマブル・コマンド・フォーマット (PCF) でのユーザー定義のメッセージ。

このメッセージは、プログラマブル・コマンド・フォーマット (PCF) メッセージの構造体に適合するユーザー定義のメッセージです。GMCONV オプションを MQGET 呼び出しに指定している場合は、この形式のメッセージを変換することができます。プログラマブル・コマンド・フォーマット・メッセージの使用の詳細については、[プログラマブル・コマンド・フォーマットの使用](#)を参照してください。

FMRMH

参照メッセージ・ヘッダー。

メッセージ・データは、参照メッセージ・ヘッダー MQRMH で始まります。このデータの後に他のデータが続くこともあります。データのフォーマット名、文字セット、およびエンコーディングは、MQRMH 内の、RMCSI、および RMENC フィールドによって指定 RMFMT される。この構造体の詳細については、[1208 ページの『IBM i での MQRMH \(参照メッセージ・ヘッダー\)』](#)を参照してください。GMCONV オプションを MQGET 呼び出しに指定している場合は、この形式のメッセージを変換することができます。

FMRFH

規則およびフォーマット・ヘッダー。

メッセージ・データは、規則およびフォーマット・ヘッダー MQRFH で始まります。このデータの後に他のデータが続くこともあります。データの形式名、文字セット、およびエンコードは、MQRFH の RFFMT、RFCSI、および RFENC フィールドでそれぞれ与えられます。GMCONV オプションを MQGET 呼び出しに指定している場合は、この形式のメッセージを変換することができます。

FMRFH2

規則およびフォーマット・ヘッダー・バージョン 2。

メッセージ・データは、バージョン 2 の規則およびフォーマット・ヘッダー MQRFH2 で始まります。このデータの後に他のデータが続くこともあります。オプションのデータの形式名、文字セット、およびエンコードは、MQRFH2 の RF2FMT、RF2CSI、および RF2ENC フィールドで指定されます。GMCONV オプションを MQGET 呼び出しに指定している場合は、この形式のメッセージを変換することができます。

FMSTR

全体が文字で構成されているメッセージ。

アプリケーション・メッセージ・データは SBCS スtring (1 バイト文字セット)、または DBCS スtring (2 バイト文字セット) のいずれかにすることができます。GMCONV オプションを MQGET 呼び出しに指定している場合は、この形式のメッセージを変換することができます。

FMTM

トリガー・メッセージ。

このメッセージは、MQTM 構造によって記述されるトリガー・メッセージです。この構造の詳細については、[1245 ページの『MQTM - トリガー・メッセージ』](#)を参照してください。GMCONV オプションを MQGET 呼び出しに指定している場合は、この形式のメッセージを変換することができます。

FMWIH

作業情報ヘッダー。

メッセージ・データは、作業情報ヘッダー MQWIH で始まり、その後にアプリケーション・データが続きます。アプリケーション・データの形式名は、MQWIH 構造体の WIFMT フィールドで指定します。

FMXQH

伝送キュー・ヘッダー。

メッセージ・データは伝送キュー・ヘッダー MQXQH で始まります。元のメッセージからのデータは、MQXQH 構造体のすぐ後に続きます。元のメッセージ・データの形式名は、伝送キュー・ヘッダー MDFMT の一部である MQMD 構造体の MDFMT フィールドによって指定されます。この構造体の詳細については、[1255 ページの『IBM i での MQXQH \(伝送キュー・ヘッダー\)』](#)を参照してください。

FMXQH の MDFMT を持つメッセージには、COA および COD レポートは生成されません。

これは、MQGET 呼び出しでは出力フィールド、MQPUT および MQPUT1 呼び出しでは入力フィールドです。このフィールドの長さは LNFMT によって指定されます。このフィールドの初期値は FMNONE です。

MDGID (24 バイトのビット・ストリング)

グループ ID。

物理メッセージが属する特定のメッセージ・グループまたは論理メッセージを識別するために使用されるバイト・ストリングです。MDGID フィールドは、メッセージのセグメント化が許可されている場合にも使用します。いずれの場合についても、MDGID フィールドにはヌル以外の値を設定し、MDMFL フィールドには以下に示すフラグのうち 1 つ以上を設定します。

- MFMIG
- MFLMIG
- MFSEG
- MFLSEG
- MFSEGA

上記のフラグを設定しなかった場合、MDGID の値は特殊なヌル値である GINONE となります。

MQPUT または MQGET 呼び出しでは、以下の場合には、アプリケーションがこのフィールドを設定する必要はありません。

- MQPUT 呼び出しで、PMLOGO を指定した場合
- MQGET 呼び出しで、MOGRPI が指定されていない場合。

これらの呼び出しをレポート・メッセージ以外のメッセージに使用する場合について考えてみます。ただし、アプリケーションがさらに制御を要求する場合、または呼び出しが MQPUT1 の場合、アプリケーションは、MDGID に適切な値が設定されていることを確認する必要があります。

メッセージ・グループおよびメッセージ・セグメントは、グループ ID が重複していない場合にのみ正しく処理できます。そのため、アプリケーションごとに固有のグループ ID を生成しないでください。アプリケーションでは次のいずれかの処理を行ってください。

- PMLOGO を指定した場合は、キュー・マネージャーが、グループにまとめられているメッセージまたは論理メッセージのセグメントであるメッセージの中の最初のメッセージに対して固有のグループ ID を自動的に生成し、残りのメッセージにそのグループ ID を使用します。したがって、アプリケーションで特別なアクションをとる必要はありません。この手順を使用することを検討してください。
- PMLOGO を指定しなかった場合は、アプリケーションからキュー・マネージャーにグループ ID を生成するよう要求してください。要求するには、グループにまとめられているメッセージまたは論理メッセージのセグメントであるメッセージに対して発行する最初の MQPUT または MQPUT1 呼び出しで MDGID に GINONE を設定してください。そして、その呼び出しの出力時にキュー・マネージャーから戻されるグループ ID をグループの残りのメッセージ、あるいは論理メッセージのセグメントに使用してください。メッセージ・グループの中にセグメント分割されたメッセージがある場合は、そのグループのすべてのセグメントおよびメッセージに、同じグループ ID を使用する必要があります。

PMLOGO を指定しなかった場合は、グループに属するメッセージおよび論理メッセージのセグメントであるメッセージを任意の順序 (逆順など) で書き込むことができますが、このようなメッセージに対して発行する最初の MQPUT 呼び出しまたは MQPUT1 呼び出しでグループ ID を割り振る必要があります。

MQPUT 呼び出しおよび MQPUT1 呼び出しの入力時にキュー・マネージャーが使用する値については、[PMOPT](#) で詳しく説明します。MQPUT 呼び出しおよび MQPUT1 呼び出しの出力時に、オープンされたオブジェクトが単一キューであり、配布リストではない場合、キュー・マネージャーはこのフィールドに、メッセージと共に送信された値を設定します。オープンされたオブジェクトが配布リストである場合、このフィールドの値は変わりません。後者の場合、生成されたグループ ID をアプリケーションで認識するには、アプリケーションは PRGID フィールドのある MQPMR レコードを提供する必要があります。

MQGET 呼び出しへの入力では、キュー・マネージャーは [表 1](#) で詳述されている値を使用します。MQGET 呼び出しの出力時に、キュー・マネージャーは、このフィールドに、取り出されたメッセージの値を設定します。

以下のような特殊値が定義されます。

GINONE

グループ ID は指定されません。

値は、フィールドの長さについては 2 進ゼロです。この値は、グループに含まれていないメッセージ (論理メッセージのセグメントではない) で、かつセグメント化が許可されていないメッセージに使用されます。

このフィールドの長さは、LNGID で指定されます。このフィールドの初期値は GINONE です。MDVER 値が MDVER2 未満の場合、このフィールドは無視されます。

MDMFL (10 桁の符号付き整数)

メッセージ・フラグ。

メッセージの属性を指定したり、メッセージの処理を制御したりするフラグを指定します。フラグには次の 2 種類があります。

- セグメント化フラグ
- 状況フラグ

以降に、これらのフラグについて説明します。

セグメント化フラグ: メッセージが大きすぎてキューに入らない場合、メッセージをキューに書き込もうとすると通常は失敗します。セグメント化とは、キュー・マネージャーまたはアプリケーションがメッセージをセグメントといういくつかの小さな単位に分割して、各セグメントを別個の物理メッセージとしてキューに入れるための手法を指します。メッセージを取り出すアプリケーションは、セグメントを 1 つずつ取り出すか、またはキュー・マネージャーに対してセグメントを再組み立てして 1 つのメッセージにするよう要求することができます。後者の場合、メッセージは MQGET 呼び出しで返されます。後者の方法を行うには、MQGET 呼び出しで GMCMPM オプションを指定して、メッセージ全体

を収容できるだけの長さのバッファを提供します。(GMCMPM オプションの詳細については、[1080 ページの『IBM i での MQGMO \(読み取りメッセージ・オプション\)』](#)を参照してください。)メッセージのセグメント化は、送信側のキュー・マネージャー、中間キュー・マネージャー、または宛先キュー・マネージャーで実行できます。

以下のいずれかのオプションを指定すると、メッセージのセグメント化を制御できます。

MFSEGI

セグメント化を禁止します。

このオプションを指定した場合、キュー・マネージャーはメッセージをセグメントに分割することはできません。すでにセグメントになっているメッセージに対して、このオプション指定すると、そのセグメントはさらに小さいセグメントに分割されることはありません。

このフラグの値は 2 進ゼロです。これはデフォルトです。

MFSEGA

セグメント化できます。

このオプションを指定した場合、キュー・マネージャーはメッセージをセグメントに分割することができます。すでにセグメントになっているメッセージにこのオプションを指定した場合は、そのセグメントをさらにいくつかの小さなセグメントに分割することができます。MFSEGA は、MFSEG または MFLSEG を設定していなくても設定できます。

キュー・マネージャーは、メッセージをセグメント化するときに、各セグメントとともに送信される MQMD のコピーの MFSEG フラグをオンにします。ただし、アプリケーションが MQPUT または MQPUT1 呼び出しで提供する MQMD 内のこれらのフラグの設定は変更しません。論理メッセージ内の最後のセグメントである場合には、キュー・マネージャーは、セグメントとともに送信される MQMD 内の MFLSEG フラグもオンにします。

注: MFSEGA を指定して、PMLOGO を指定せずにメッセージを書き込むときには注意が必要です。メッセージが次の条件に該当する場合、

- セグメントではない
- グループに属していない
- 転送されない

この場合、アプリケーションでは、キュー・マネージャーがメッセージごとに固有のグループ ID を生成するように、それぞれの MQPUT 呼び出しまたは MQPUT1 呼び出しの発行前に MDGID フィールドを必ず GINONE にリセットしなければなりません。この操作を行わないと、互いに関連のないメッセージに誤って同じグループ ID が割り当てられ、それ以降の処理で問題が発生することがあります。MDGID フィールドをリセットしなければならない場合の詳細については、MDGID フィールドおよび PMLOGO オプションを参照してください。

キュー・マネージャーは、セグメント (必要な場合はセグメントにヘッダー・データを加えたもの) がキューに収まるように、必要に応じてメッセージをいくつかのセグメントに分割します。ただし、キュー・マネージャーが生成する 1 つのセグメントのサイズには下限があり、この下限より小さくできるのは、あるメッセージから作成された最後のセグメントのみです。アプリケーションが生成するセグメントのサイズの下限は 1 バイトです。キュー・マネージャーが生成する各セグメントのサイズは、均等にならないことがあります。キュー・マネージャーは、メッセージを次のように処理します。

- ユーザー定義の形式の場合は、16 バイトの倍数の境界で分割する。つまり、キュー・マネージャーでは、(最後のセグメントを除いて) 16 バイトより小さいセグメントは生成されません。
- 組み込み形式の場合は、FMSTR 形式を除き、存在するデータの性質に合った箇所で分割する。ただし、MQ ヘッダー構造体の途中ではメッセージを分割しません。つまり、MQ ヘッダー構造体が 1 つ含まれているセグメントは、それ以上分割されず、結果として、そのメッセージの最小セグメント・サイズは 16 バイトより大きくなります。

キュー・マネージャーが生成する 2 番目以降のセグメントは、次のいずれかで開始されます。

- MQ ヘッダー構造
- アプリケーション・メッセージ・データの先頭

– アプリケーション・メッセージ・データの途中

- FMSTR 形式の場合は、存在するデータの性質 (SBCS、DBCS、またはこれらが混在した SBCS/DBCS) に関係なく分割します。ストリングが DBCS または SBCS/DBCS である場合は、文字セットの変換ができないセグメントになることがあります。キュー・マネージャーは、(最後のセグメントを除いて) FMSTR メッセージを 16 バイトより小さいセグメントに分割することはありません。
- 各セグメントの先頭のデータを正しく記述するため、各セグメントの MQMD の MDFMT フィールド、MDCSI フィールド、および MDENC フィールドをキュー・マネージャーによって設定します。形式名は、組み込み形式名またはユーザー定義の形式名になります。
- MDOFF の値がゼロより大きいセグメントについて、MQMD の MDREP フィールドを以下のように変更します。
 - 各レポート・タイプについて、レポート・オプションが RO*D であるのに、セグメント内にユーザー・データ (MQ ヘッダー構造体が存在する場合にそのあとに続くデータ) の最初の 100 バイトが入っていない場合は、レポート・オプションを RO* に変更します。

キュー・マネージャーは上記の規則に従いますが、それ以外でも予期せずにメッセージを分割することがあります。そのため、メッセージがどこで分割されるかについては推測しないでください。

持続メッセージの場合、キュー・マネージャーは作業単位の範囲内でのみセグメント化を実行できます。ただし、次のような点に注意してください。

- MQPUT または MQPUT1 呼び出しがユーザー定義の作業単位内で実行されていれば、その作業単位が使用されます。セグメント分割の処理中に呼び出しが失敗した場合、キュー・マネージャーはその呼び出しによりキューに入れられたセグメントを除去します。ただし、このように失敗しても、作業単位は正常にコミットされます。
- 呼び出しがユーザー定義の作業単位以外で実行されていて、またユーザー定義の作業単位が存在していない場合、キュー・マネージャーは、この呼び出しの間だけの作業単位を作成します。呼び出しが成功すると、キュー・マネージャーは作業単位を自動的にコミットします (アプリケーションでこれを行う必要はありません)。呼び出しが失敗すると、キュー・マネージャーは作業単位をバックアウトします。
- 呼び出しがユーザー定義の作業単位の範囲外で動作しているときに、ユーザー定義の作業単位が存在する場合、キュー・マネージャーはセグメント化を実行できません。メッセージをセグメントに分割する必要がない場合には、呼び出しは成功します。しかし、メッセージをセグメントに分割する必要がある場合には、理由コード RC2255 を戻して、呼び出しは失敗します。

非持続メッセージの場合、キュー・マネージャーはセグメント化を実行するために作業単位を使用できるようにする必要はありません。

セグメントに分割するメッセージのデータ変換に関して、以下に示す特別な考慮事項を定める必要があります。

- MQGET 呼び出しで受信側のアプリケーションのみがデータ変換を実行し、かつそのアプリケーションで GMCMPM オプションを指定した場合は、データ変換出口で変換を行うために、データ変換出口に完全なメッセージが渡されます。しかし、メッセージがセグメントに分割されたことはデータ変換出口では認識しません。
- 受信側のアプリケーションが 1 回につき 1 つのセグメントを取り出す場合は、データ変換出口を呼び出して 1 回につき 1 つのセグメントを変換します。したがって、データ変換出口は各セグメント内のデータを他のセグメント内のデータと関係なく変換できることが必要となります。

メッセージのデータの性質上、16 バイト境界でデータをセグメントに分割すると、データ変換出口で変換できないセグメントになるような場合、または形式が FMSTR で文字セットが DBCS または SBCS/DBCS である場合は、送信側のアプリケーション自体がセグメントを作成して書き込み、MFSEGI を指定してこれらのセグメントがさらにいくつかのセグメントに分割されないようにします。これにより、送信側のアプリケーションでは、データ変換出口が各セグメントを正常に変換できるだけの十分な情報が各セグメントに含まれるようにすることができます。

- 送信側のメッセージ・チャンネル・エージェント (MCA) に対して送信側での変換を指定した場合、MCA は論理メッセージのセグメントでないメッセージのみを変換し、論理メッセージのセグメントであるメッセージは変換しません。

このフラグは、MQPUT 呼び出しおよび MQPUT1 呼び出しでは入力フラグであり、MQGET 呼び出しでは出力フラグです。後者の呼び出しの場合、キュー・マネージャーはこのフラグの値を MQGMO の GMSEG フィールドにも反映します。

このフラグの初期値は MFSEGI です。

状況フラグ: 物理メッセージの状況を示すフラグです。この状況は、メッセージ・グループに属する、論理メッセージのセグメントである、メッセージ・グループに属し、かつ論理メッセージのセグメントである、メッセージ・グループに属さず、かつ論理メッセージのセグメントでもない、のいずれかです。MQPUT 呼び出し、または MQPUT1 呼び出しでは、以下のオプションのうち 1 つ以上を指定できます。MQGET 呼び出しでは以下のオプションのうち 1 つ以上が戻されます。

MF MIG

メッセージは特定のグループのメンバーである。

MFL MIG

メッセージはグループ内の最後の論理メッセージです。

このフラグを設定すると、キュー・マネージャーはメッセージで送信される MQMD のコピーの MF MIG をオンにしますが、MQPUT または MQPUT1 呼び出しでアプリケーションで提供される MQMD のフラグの設定は変更しません。

このフラグは 1 つの論理メッセージのみで構成されるグループにも有効です。この場合にも、このフラグは設定されますが、MDSEQ フィールドの値は 1 になります。

MF SEG

メッセージは論理メッセージのセグメントです。

MF SEG を指定して MFL SEG を指定しなかった場合、セグメント内のアプリケーション・メッセージ・データの長さ (MQ ヘッダー構造体が存在する場合にはその長さを除く) は 1 以上でなければなりません。長さがゼロの場合、MQPUT または MQPUT1 呼び出しは、理由コード RC2253 で失敗します。

MFL SEG

メッセージは論理メッセージの最後のセグメントです。

このフラグを設定すると、キュー・マネージャーはメッセージで送信される MQMD のコピーの MF SEG をオンにしますが、MQPUT または MQPUT1 呼び出しでアプリケーションで提供される MQMD のフラグの設定は変更しません。

このフラグは 1 つのセグメントのみで構成される論理メッセージにも有効です。この場合にも、このフラグは設定されますが、MDOFF フィールドの値はゼロになります。

MFL SEG を指定した場合、セグメント内のアプリケーション・メッセージ・データの長さ (ヘッダー構造体が存在する場合にはその長さを除く) がゼロでも構いません。

アプリケーションでは、メッセージの書き込み時にこれらのフラグが正しく設定されるようにする必要があります。PMLOGO を指定した場合、または前に行ったキュー・ハンドルに対する MQPUT 呼び出しで MQPMO_LOGICAL_ORDER を指定した場合は、このフラグの設定値はキュー・マネージャーがキュー・ハンドル用に保存するグループ情報およびセグメント情報が矛盾してはなりません。PMLOGO を指定した場合、キュー・ハンドルに対して連続した MQPUT 呼び出しを行うときには以下の条件があります。

- 現行のグループおよび論理メッセージがない場合は、上記のすべてのフラグ (およびこれらのフラグを組み合わせたもの) が有効です。
- MF MIG を指定した場合には、MFL MIG を指定するまで MF MIG をオンにしておいてください。この条件が満たされないと、呼び出しは失敗し理由コード RC2241 が戻ります。
- MF SEG を指定した場合には、MFL SEG を指定するまで MF MIG をオンにしておいてください。この条件が満たされないと、呼び出しは失敗し理由コード RC2242 が戻ります。
- MF MIG を指定しないで MF SEG を指定した場合には、MFL SEG が指定されるまで MF MIG をオフにしておいてください。この条件が満たされないと、呼び出しは失敗し理由コード RC2242 が戻ります。

表 1 に、これらのフラグの有効な組み合わせと各種のフィールドに使用する値を示します。

これらのフラグは、MQPUT 呼び出し、および MQPUT1 呼び出しでは入力フラグであり、MQGET 呼び出しでは出力フラグです。後者の呼び出しの場合、キュー・マネージャーはフラグの値を MQGMO の GMGST フィールドおよび GMSST フィールドにも反映します。

デフォルト・フラグ: メッセージにデフォルト属性が設定されていることを示すために、以下のフラグを指定できます。

MFNONE

メッセージ・フラグはありません (デフォルトのメッセージ属性)。

これはセグメンテーションを禁止し、メッセージがグループに属していないこと、およびメッセージが論理メッセージのセグメントではないことを示します。MFNONE は、プログラムの文書化を支援するために定義されています。このフラグを他の目的で使用することは意図されていませんが、値がゼロであるため、そのように使用しても検出されることはありません。

MDMFL フィールドはいくつかのサブフィールドに分かれています。詳細については、[1433 ページの『IBM iでのレポート・オプションおよびメッセージ・フラグ』](#)を参照してください。

このフィールドの初期値は MFNONE です。MDVER 値が MDVER2 未満の場合、このフィールドは無視されます。

MDMID (24 バイトのビット・ストリング)

メッセージ ID。

あるメッセージを他のメッセージと区別するために使用されるバイト・ストリングです。2つのメッセージが同じメッセージ ID を持つことは、キュー・マネージャーによって禁止されませんが、通常は避けてください。メッセージ ID は、メッセージの永続的なプロパティであり、キュー・マネージャーを再始動しても存続します。メッセージ ID は、文字ストリングではなくバイト・ストリングなので、あるキュー・マネージャーから別のキュー・マネージャーへメッセージが流れても、文字セット間の変換は行われません。

MQPUT 呼び出しと MQPUT1 呼び出しでは、アプリケーションが MINONE または PMNMID を指定した場合、キュー・マネージャーは、メッセージが書き込まれるときに固有のメッセージ ID を生成し、メッセージとともに送信されるメッセージ記述子の中にそれを入れます。また、キュー・マネージャーは、送信側のアプリケーションに属するメッセージ記述子の中にこのメッセージ ID を返します。アプリケーションは、この値を使用して、特定のメッセージに関する情報を記録し、アプリケーションの他の部分からの照会に応答することができます。

キュー・マネージャーによって生成 MDMID される識別子は、4 バイトの製品識別子 (ASCII または AMQ-CSQ-EBCDIC 形式の または、ここで `␣` は単一の空白文字を表す) で始まり、それに続いて製品固有の実装による一意の文字列が続く。IBM MQ の場合は、これには、キュー・マネージャー名の最初の 12 文字と、システム・クロックからとられた値が入っています。したがって、メッセージ ID を必ず固有の値にするためには、相互通信可能なすべてのキュー・マネージャーで、プログラム名の最初の 12 文字が異なっている必要があります。また、固有のストリングを生成する機能は、システム・クロックが逆方向に変更されないことを前提としています。キュー・マネージャーにより生成されたメッセージ ID が、アプリケーションにより生成されたメッセージ ID と重複しないようにするには、アプリケーションは、最初の文字が ASCII または EBCDIC の A から I の範囲 (X'41' から X'49' および X'C1' から X'C9') である ID を生成しないようにする必要があります。ただし、アプリケーションでこれらの範囲の先頭文字をもつ ID を生成しないように回避措置がとられることはありません。

メッセージをトピックに書き込む場合、キュー・マネージャーは、パブリッシュされるメッセージごとに必要に応じて固有のメッセージ ID を生成します。PMNMID がアプリケーションによって指定されている場合、キュー・マネージャーは固有のメッセージ ID を生成して出力に戻します。MINONE がアプリケーションによって指定される場合、MQMD 内の MDMID フィールドの値は呼び出しからの戻り時に変更されません。

保存パブリケーションについて詳しくは、[PMOPT](#) の PMRET の説明を参照してください。

メッセージを配布リストに書き込む場合は、キュー・マネージャーが必要に応じて固有のメッセージ ID を生成しますが、MINONE または PMNMID を指定したとしても、呼び出しからの戻り時に、MQMD の MDMID フィールドの値が変更されることはありません。キュー・マネージャーが生成するメッセージ

ID をアプリケーションで認識するには、アプリケーションは PRMID フィールドのある MQPMR レコードを提供する必要があります。

また、送信側のアプリケーションは、MINONE 以外の特定の値をメッセージ ID に指定することができません。これによって、キュー・マネージャーは固有のメッセージ ID の生成を停止します。メッセージを転送するアプリケーションでは、この機能を使用して元のメッセージのメッセージ ID を伝搬することができます。

キュー・マネージャー自体は、以下の目的以外に、このフィールドを使用することはありません。

- 要求があれば、上述のように固有値を生成する。
- メッセージの取得要求を発行したアプリケーションに値を配布する。
- このメッセージについて生成されたレポート・メッセージがあれば、(MDREP オプションに応じて) そのレポート・メッセージの MDCID フィールドに値をコピーする。

キュー・マネージャーまたはメッセージ・チャネル・エージェントは、レポート・メッセージを生成するとき、元のメッセージの MDREP フィールド、すなわち RONMI または ROPMI のいずれかによって指定された方法で MDMID フィールドを設定します。レポート・メッセージを生成するアプリケーションも、上記の指定を行います。

MQGET 呼び出しの場合、MDMID は、キューから取得する特定のメッセージを選択するために使用できる 5 つのフィールドの 1 つです。通常、MQGET 呼び出しでは、キューにある次のメッセージを取り出しますが、特定のメッセージが必要な場合には、5 つの選択基準の 1 つ以上を任意の組み合わせで指定することによって目的のメッセージを取得できます。このメッセージの選択には、次のフィールドを使用できます。

- MDMID
- MDCID
- MDGID
- MDSEQ
- MDOFF

アプリケーションでは、これらのフィールドの 1 つ以上を必要な値に設定した上で、MQGMO の GMMO フィールドに対応する MO* 一致オプションを設定して、これらのフィールドを選択基準として使用するよう指定します。これらのフィールドに値が指定されたメッセージだけが取り出しの対象になります。(アプリケーション側で変更しない場合は) GMMO フィールドのデフォルトでは、メッセージ ID および相関 ID の 2 つが選択基準として使用されます。

通常は、選択基準を満たす最初のメッセージがキューから戻されます。ただし、GMBRWN を指定した場合は、選択基準を満たす次のメッセージが戻されます。このメッセージの走査は、現行カーソル位置の後から開始されます。

注: 選択基準を満たすメッセージがキューから順次走査されるため、選択基準を指定しなかった場合よりも、取り出し時間は遅くなります。特に、条件を満たすメッセージを見付けるまでに多数のメッセージを走査する必要がある場合は、この傾向が強くなります。

各種の状況で選択基準を使用する方法については、[表 1](#) を参照してください。

メッセージ ID として MINONE を指定すると、MOMSGI を指定しなかった場合と同じ結果になります。つまり、すべてのメッセージ ID が一致することになります。

GMMUC オプションが、MQGET 呼び出しの **GMO** パラメーターで指定されている場合、このフィールドは無視されます。

MQGET 呼び出しから戻ったとき、MDMID フィールドは、戻されたメッセージ (それがあある場合) のメッセージ ID に設定されます。

以下に示す特別な値が使用されることがあります。

MINONE

メッセージ ID が指定されていません。

値は、フィールドの長さについては2進ゼロです。

これは、MQGET、MQPUT、およびMQPUT1呼び出しの入出力フィールドです。このフィールドの長さはLN MIDによって指定されます。このフィールドの初期値はMINONEです。

MDMT (10桁の符号付き整数)

メッセージ・タイプ。

これは、メッセージ・タイプを示します。メッセージ・タイプは、以下のようにグループ化されています。

MTSFST

システム定義のメッセージ・タイプに関する最低値。

MTSLST

システム定義のメッセージ・タイプに関する最高値。

現在、以下の値がシステム範囲内で定義されています。

MTDGRM

応答を必要としないメッセージ。

メッセージは、応答が不要なメッセージです。

MTRQST

応答を必要とするメッセージ。

メッセージは、応答が必要なメッセージです。

応答の送信先のキューの名前がMDRQフィールドで指定されていることが必要です。MDREPフィールドは、応答のMDMIDおよびMDCIDが設定される方法を示すものです。

MTRPLY

以前の要求メッセージに対する応答。

このメッセージは、以前の要求メッセージ(MTRQST)に対する応答です。このメッセージは、要求メッセージのMDRQフィールドが示すキューに送信される必要があります。要求のMDREPフィールドを使用して、応答のMDMIDおよびMDCIDの設定方法を制御することが必要です。

注: キュー・マネージャーは、要求と応答の関係を強制することはありません。これはアプリケーションが担当します。

MTRPRT

レポート・メッセージ。

このメッセージは何らかの予期したオカレンスまたは予期しないオカレンスについて報告するもので、通常は一部のその他のメッセージと関連します(例えば、無効なデータの入った要求メッセージを受信した、など)。このメッセージは、元のメッセージのメッセージ記述子のMDRQフィールドが示すキューに送信される必要があります。レポートの性質を示すためにMDFBフィールドの設定が必要です。元のメッセージのMDREPフィールドを使用して、レポートのMDMIDおよびMDCIDの設定方法を制御することが必要です。

キューマネージャまたはメッセージチャネルエージェントによって生成されたレポートメッセージは、常にキューMDRQに送信され、前述のように`MDFB`および`MDCID`フィールドが設定されます。

システム範囲内にあるその他の値も、MQIの将来のバージョンで定義される可能性があります。定義後は、MQPUTおよびMQPUT1呼び出しによって、エラーなしで受け入れられます。

アプリケーション定義の値を使用することもできます。ただし、以下の範囲内でなければなりません。

MTAFST

アプリケーション定義のメッセージ・タイプに関する最低値。

MTALST

アプリケーション定義のメッセージ・タイプに関する最高値。

MQPUT 呼び出しおよび MQPUT1 呼び出しでは、MDMT 値は、システム定義の範囲かアプリケーション定義の範囲内でなければなりません。この範囲内でない場合、呼び出しは理由コード RC2029 で失敗します。

これは、MQGET 呼び出しでは出力フィールド、MQPUT および MQPUT1 呼び出しでは入力フィールドです。このフィールドの初期値は MTDGRM です。

MDOFF (10 桁の符号付き整数)

論理メッセージの先頭を起点とする、物理メッセージ中のデータのオフセット。

物理メッセージのデータの (そのデータを一部として含んでいる) 論理メッセージの先頭からのオフセットをバイト単位で指定します。このようなデータをセグメント といいます。オフセットは、0 から 999 999 999 までの範囲です。論理メッセージのセグメントでない物理メッセージのオフセット値は 0 です。

MQPUT または MQGET 呼び出しでは、以下の場合には、アプリケーションがこのフィールドを設定する必要はありません。

- MQPUT 呼び出しで、PMLOGO を指定した場合
- MQGET 呼び出しで、MOOFFS を指定しなかった場合。

以下、これらの呼び出しをレポート・メッセージ以外のメッセージに使用する場合の推奨方法について説明します。ただし、アプリケーションがそれらの設定を行わなかった場合、または呼び出しが MQPUT1 の場合、アプリケーションは、MDOFF に適切な値が設定されていることを確認する必要があります。

MQPUT 呼び出しおよび MQPUT1 呼び出しへの入力では、キュー・マネージャーは表 1 で詳述されている値を使用します。MQPUT および MQPUT1 呼び出しの出力時に、キュー・マネージャーは、このフィールドにメッセージと共に送信された値を設定します。

論理メッセージのセグメントに関するレポート・メッセージの場合は、MDOLN フィールドを使用して (OLUNDF 以外の値であること)、キュー・マネージャーが保存するセグメント情報の中のオフセット値を更新します。

MQGET 呼び出しへの入力では、キュー・マネージャーは表 1 で詳述されている値を使用します。MQGET 呼び出しの出力時に、キュー・マネージャーは、このフィールドに、取り出されたメッセージの値を設定します。

フィールドの初期値は、0 です。MDVER 値が MDVER2 未満の場合、このフィールドは無視されます。

MDOLN (10 桁の符号付き整数)

元のメッセージの長さ。

これは、セグメントとしてのレポート・メッセージにのみ関連するフィールドです。これは、レポート・メッセージが関連するメッセージ・セグメントの長さを指定します。セグメントが属する論理メッセージの長さやレポート・メッセージ内のデータの長さは指定しません。

注: セグメントであるメッセージのレポート・メッセージを生成する場合、キュー・マネージャーとメッセージ・チャネル・エージェントは、元のメッセージの MDGID、MDSEQ、MDOFF、および MDMFL の各フィールドをレポート・メッセージの MQMD にコピーします。その結果、そのレポート・メッセージも 1 つのセグメントです。レポート・メッセージを生成するアプリケーションで、同じことを実行し、MDOLN フィールドが正しく設定されていることを確認することを推奨します。

以下のような特殊値が定義されます。

OLUNDF

元のメッセージの長さが定義されていません。

MDOLN は、MQPUT 呼び出しおよび MQPUT1 呼び出しの入力フィールドですが、アプリケーションが提供する値は、以下に示す特定の場合にのみ受け入れられます。

- 書き込んでいるメッセージがセグメントであると同時にレポート・メッセージでもある場合、キュー・マネージャーは指定された値を受け入れます。値は次のものでなければなりません。
 - セグメントが最後のセグメントではない場合は、ゼロより大きい値

- セグメントが最後のセグメントである場合は、ゼロ以上の値
- メッセージに含まれるデータの長さ以上の値

これらの条件を満たさない場合は、理由コード RC2252 が戻り、呼び出しは失敗します。

- 書き込んでいるメッセージがセグメントであるが、レポート・メッセージではない場合、キュー・マネージャーはこのフィールドを無視して、アプリケーション・メッセージ・データの長さを使用します。
- 上記以外の場合、キュー・マネージャーはこのフィールドを無視して、OLUNDF を使用します。

これは、MQGET 呼び出し用の出力フィールドです。

このフィールドの初期値は OLUNDF です。MDVER 値が MDVER2 未満の場合、このフィールドは無視されます。

MDPAN (28 バイトの文字ストリング)

メッセージを書き込むアプリケーションの名前。

これは、メッセージの起点コンテキストの一部です。メッセージ・コンテキストについての詳細は、[メッセージ・コンテキストおよびコンテキスト情報の制御](#)を参照してください。

MDPAN の形式は、MDPAT の値によって決まります。

このフィールドがキュー・マネージャーにより設定されると (つまり、PMSETA を除くすべてのオプションについて設定されると)、環境によって決定される値に設定されます。

-  **z/OS** z/OS では、キュー・マネージャーは次のものを使用します。
 - z/OS バッチの場合は、JES JOB カードからの 8 文字のジョブ名
 - TSO の場合は、7 文字の TSO ユーザー ID
 - CICS の場合は、8 文字のアプリケーション ID とそれに続く 4 文字のトランザクション ID
 - IMS の場合は、8 文字の IMS システム ID とそれに続く 8 文字の PSB 名
 - XCF の場合は、8 文字の XCF グループ名とそれに続く 16 文字の XCF メンバー名
 - キュー・マネージャーが生成したメッセージの場合は、キュー・マネージャー名の最初の 28 文字
 - CICS のない分散キューイングの場合は、チャンネル・イニシエーターの 8 文字のジョブ名とそれに続く送達不能キューに書き込まれた 8 文字のモジュール名、およびそれに続く 8 文字のタスク ID。
 - 環境 z/OS UNIX System Services 用に作成されたアドレス空間の 8 文字のジョブ名 IBM MQ for z/OS を用いた言語バインディング MQSeriesJava 処理。通常、これは単一の数字が付加された TSO ユーザー ID です。

それぞれの名前について、フィールドの残り部分にスペースがあれば右側にブランクが埋め込まれます。複数の名前がある場合、名前と名前の間に区切り文字はありません。

-  **Windows** PC DOS、および Windows システムでは、キュー・マネージャーは以下を使用します。
 - CICS アプリケーションの場合は、CICS トランザクション名
 - CICS アプリケーション以外の場合は、実行可能なジョブの完全修飾名の右端から 28 文字
-  **IBM i** IBM i では、キュー・マネージャーは完全修飾ジョブ名を使用します。
-  **Linux**  **AIX** AIX and Linux では、キュー・マネージャーは次のものを使用します。
 - CICS アプリケーションの場合は、CICS トランザクション名
 - 非 CICS アプリケーションでは、キュー・マネージャーは、実行可能ファイルの完全修飾名の右端の 14 文字を使用できる場合にはそれを使用し、使用できない場合 (AIX の場合など) にはブランクを使用します。
- VSE/ESA では、キュー・マネージャーは、8 文字のアプリケーション ID とそれに続く 4 文字の tranid を使用します。

MQPUT および MQPUT1 呼び出しの場合、**PMO** パラメーターに または PMSETA が指定されていれば、これは入出力フィールドです。フィールド内でヌル文字より後の情報はすべて破棄されます。ヌル文

字およびその後続く文字は、キュー・マネージャーによってブランクに変換されます。PMSETA が指定されていない場合、入力においてこのフィールドは無視され、出力専用フィールドになります。

これは、MQGET 呼び出しの出力フィールドです。このフィールドの長さは LNPAN によって指定されます。このフィールドの初期値は 28 個のブランク文字です。

MDPAT (10 桁の符号付き整数)

メッセージを書き込むアプリケーションのタイプ。

これは、メッセージの**起点コンテキスト**の一部です。メッセージ・コンテキストについての詳細は、[メッセージ・コンテキストおよびコンテキスト情報の制御](#)を参照してください。

MDPAT は、以下の標準タイプのいずれかにすることができます。ユーザー定義のタイプを使用することもできますが、値は ATUFST から ATULST までの範囲内に制限する必要があります。

ATAIX

AIX アプリケーション (ATUNIX と同じ値)。

ATBRKR

ブローカー。

ATCICS

CICS トランザクション。

ATCICB

CICS bridge.

ATVSE

CICS/VSE トランザクション。

ATDOS

PC DOS 上の IBM MQ MQI client アプリケーション。

ATDQM

分散キュー・マネージャー・エージェント。

ATGUAR

Tandem Guardian アプリケーション (ATNSK と同じ値)。

ATIMS

IMS アプリケーション。

ATIMSB

IMS ブリッジ。

ATJAVA

Java.

ATMVS

MVS または TSO アプリケーション (ATZOS と同じ値)。

ATNOTE

Lotus Notes エージェント・アプリケーション。

ATNSK

Tandem NonStop Kernel アプリケーション。

AT390

OS/390 アプリケーション (ATZOS と同じ値)。

AT400

IBM i アプリケーション。

ATQM

キュー・マネージャー。

ATUNIX

UNIX アプリケーション。

ATVOS

Stratus VOS アプリケーション。

ATWIN

16 ビットの Windows アプリケーション。

ATWINT

32 ビットの Windows アプリケーション。

ATXCF

XCF。

ATZOS

z/OS アプリケーション。

ATDEF

デフォルトのアプリケーション・タイプ。

この値は、アプリケーションが実行中のプラットフォームの、デフォルト・アプリケーション・タイプです。

注: この定数の値は環境によります。

ATUNK

アプリケーション・タイプが不明。

この値は、他のコンテキスト情報が存在しているのに、アプリケーション・タイプが分からないことを示すために使用することができます。

ATUFST

ユーザー定義のアプリケーション・タイプの最低値。

ATULST

ユーザー定義のアプリケーション・タイプの最高値。

以下のような特殊値が出されることもあります。

ATNCON

メッセージの中にコンテキスト情報がありません。

メッセージがコンテキストを伴わずに書き込まれる場合 (つまり、PMNOC コンテキスト・オプションが指定される場合)、キュー・マネージャーによってこの値が設定されます。

メッセージが検索されると、MDPAT がこの値かどうかテストされ、メッセージにコンテキストがあるかどうかを調べます (その他のコンテキスト・フィールドの中にブランクでないものがある場合、PMSETA を使用するアプリケーションでは、MDPAT を ATNCON には決して設定しないようにしてください)。

ATSIB

メッセージが別の IBM MQ メッセージング製品で作成され、SIB (サービス統合バス) ブリッジを介して届いたことを示します。

アプリケーションの書き込みの結果として、キュー・マネージャーがこの情報を生成するとき、フィールドは、環境により求められる値に設定されます。

 このフィールドは、IBM i では AT400 に設定されることに注意してください。キュー・マネージャーが IBM i で ATCICS を使用することはありません。

MQPUT および MQPUT1 呼び出しの場合、**PMO** パラメーターに または PMSETA が指定されていれば、これは入出力フィールドです。PMSETA が指定されていない場合、入力においてこのフィールドは無視され、出力専用フィールドになります。

MQPUT または MQPUT1 呼び出しが正常に完了すると、メッセージがキューに書き込まれた場合、このフィールドには、そのメッセージと共に送信された MDPAT が入ります。これは、メッセージが保存された場合 (保存パブリケーションについて詳しくは、PMRET の説明を参照してください)、そのメッセージと共に保持された MDPAT の値になりますが、メッセージがパブリケーションとしてサブスクライバーに送信された場合には MDPAT として使用されません。これは、サブスクライバーが提供する値が、サブスクライバーに送信されるすべてのパブリケーションにおいて MDPAT をオーバーライドするためです。メッセージにコンテキストがない場合、このフィールドは ATNCON に設定されます。

これは、MQGET 呼び出しの出力フィールドです。このフィールドの初期値は ATNCON です。

MDPD (8 バイトの文字ストリング)

メッセージが書き込まれた日付。

これは、メッセージの起点コンテキストの一部です。メッセージ・コンテキストについての詳細は、[メッセージ・コンテキストおよびコンテキスト情報の制御](#)を参照してください。

キュー・マネージャーがこのフィールドを生成する際に使用する日付の形式は、以下のとおりです。

• YYYYMMDD

文字は、以下のものを表します。

YYYY

年 (4 桁の数字)

MM

月 (01 から 12 まで)

DD

日 (01 から 31 まで)

MDPD と MDPT フィールドでは、グリニッジ標準時 (GMT) に正確に設定されているシステム・クロックに従って、GMT が使用されます。

メッセージが作業単位の一部として書き込まれた場合は、作業単位がコミットされた日付ではなく、メッセージが書き込まれた日付です。

MQPUT および MQPUT1 呼び出しの場合、**PMO** パラメーターに または **PMSETA** が指定されていれば、これは入出力フィールドです。フィールドの内容は、キュー・マネージャーによって検査されませんが、フィールド内のヌル文字のあとにある情報はいずれも廃棄されます。ヌル文字およびその後続く文字は、キュー・マネージャーによってブランクに変換されます。**PMSETA** が指定されていない場合、入力においてこのフィールドは無視され、出力専用フィールドになります。

MQPUT または MQPUT1 呼び出しが正常に完了すると、メッセージがキューに書き込まれた場合、このフィールドには、そのメッセージと共に送信された MDPD が入ります。これは、メッセージが保存された場合 (保存パブリケーションについて詳しくは、**PMRET** の説明を参照してください)、そのメッセージと共に保持された MDPD の値になりますが、メッセージがパブリケーションとしてサブスクライバーに送信された場合には MDPD として使用されません。これは、サブスクライバーが提供する値が、サブスクライバーに送信されるすべてのパブリケーションにおいて MDPD をオーバーライドするためです。メッセージがコンテキストを持っていない場合、フィールドは完全にブランクになります。

これは、MQGET 呼び出しの出力フィールドです。このフィールドの長さは LNPDAT によって指定されます。このフィールドの初期値は 8 個のブランク文字です。

MDPER (10 桁の符号付き整数)

メッセージの持続性。

このフィールドは、システム障害が発生してキュー・マネージャーを再始動してもメッセージが残るかどうかを示します。MQPUT および MQPUT1 呼び出しでは、値は次のいずれかでなければなりません。

PEPER

メッセージは持続します。

これは、システム障害が発生してキュー・マネージャーを再始動してもメッセージが残ることを意味します。メッセージが書き込まれ、その書き込みの作業単位がコミットされると (つまり、メッセージが作業単位の一部として書き込まれると)、メッセージは補助記憶域に保存されます。メッセージがキューから除去され、受信者の作業単位がコミットされるまで (つまり、メッセージが作業単位の一部として取り出されるまで)、そのメッセージは補助記憶域に残ります。

持続メッセージがリモート・キューに送信される場合は、蓄積交換機能を使用して、宛先への経路にそって各キュー・マネージャーでメッセージが保持されます。これは、次のキュー・マネージャーにメッセージが到達したことが分かるまで続きます。

持続メッセージを以下に書き込むことはできません。

- 一時動的キュー
- カップリング・ファシリティ構造体レベルが 3 未満の場合や カップリング・ファシリティ構造体がリカバリー不能である場合の共有キュー

持続メッセージは、永続動的キュー、事前定義キュー、およびカップリング・ファシリティ構造体がレベル 3 で リカバリー可能である場合の共用キューに書き込むことができます。

PENPER

メッセージは持続しません。

これは、システム障害が発生してキュー・マネージャーを再始動すると、通常はメッセージが残らないことを意味します。これは、キュー・マネージャーの再始動中にメッセージの完全なコピーが補助記憶域で 見つかった場合でも、適用されます。

共有キューの特殊な事例として、キュー共有グループでキュー・マネージャーを再始動しても非永続メッセージが残る が、共有キューでメッセージの保管に使用したカップリング・ファシリティに障害が発生するとメッセージは残らないという場合があります。

PEQDEF

メッセージの持続性はデフォルト時のものです。

- キューがクラスター・キューである場合、メッセージの持続性は、メッセージが書き込まれるキューの特定のインスタンスを所有する宛先キュー・マネージャーで定義された **DefPersistence** 属性から取られます。通常、クラスター・キューのインスタンスの **DefPersistence** 属性の値はすべて同じですが、これは必須ではありません。

DefPersistence の値は、メッセージが宛先キューに置かれるときに *MDPER* フィールドにコピーされます。その後、**DefPersistence** が変更されても、既にキューに置かれているメッセージは影響を受けません。

- キューがクラスター・キューではない場合、メッセージの持続性は、ローカルのキュー・マネージャーで定義された **DefPersistence** 属性から取られます。これは、宛先キュー・マネージャーがリモートの場合も同じです。

キュー名の解決パスに複数の定義がある場合、デフォルトの持続性は、パスの最初の定義にあるこの属性の値から取られます。これには以下のものが考えられます。

- 別名キュー
- ローカル・キュー
- リモート・キューのローカル定義
- キュー・マネージャーの別名
- 伝送キュー (例えば、DefXmitQName キューなど)

DefPersistence の値は、メッセージが書き込まれるときに *MDPER* フィールドにコピーされます。その後、**DefPersistence** が変更されても、既に書き込まれているメッセージは影響を受けません。

持続メッセージと非持続メッセージが同一キューにあっても構いません。

メッセージに応答する場合、アプリケーションは通常、応答メッセージに対して、要求メッセージの持続性を使用します。

MQGET 呼び出しの場合、戻り値は PEPER または PENPER のいずれかです。

これは、MQGET 呼び出しでは出力フィールド、MQPUT および MQPUT1 呼び出しでは入力フィールドです。このフィールドの初期値は PEQDEF です。

MDPRI (10 桁の符号付き整数)

メッセージの優先度。

MQPUT および MQPUT1 呼び出しでは、値はゼロ以上でなければなりません。ゼロは、最低の優先順位です。以下のような特殊値も使用することができます。

PRQDEF

キューのデフォルト優先順位。

- キューがクラスター・キューの場合、メッセージの優先順位は、メッセージが書き込まれたキューの特定のインスタンスを所有する宛先キュー・マネージャーで定義された **DefPriority** 属性から取られます。通常、クラスター・キューのインスタンスの **DefPriority** 属性の値はすべて同じですが、これは必須ではありません。

DefPriority の値は、メッセージが宛先キューに置かれるときに MDPRI フィールドにコピーされます。その後、**DefPriority** が変更されても、既にキューに置かれているメッセージは影響を受けません。

- キューがクラスター・キューではない場合、メッセージの優先順位は、ローカルのキュー・マネージャーで定義された **DefPriority** 属性から取られます。これは、宛先キュー・マネージャーがリモートの場合も同じです。

キュー名の解決パスに複数の定義がある場合、デフォルトの優先順位は、パスの最初の定義にあるこの属性の値から取られます。これには以下のものが考えられます。

- 別名キュー
- ローカル・キュー
- リモート・キューのローカル定義
- キュー・マネージャーの別名
- 伝送キュー (例えば、DefXmitQName キューなど)

DefPriority の値は、メッセージが書き込まれるときに MDPRI フィールドにコピーされます。その後、**DefPriority** が変更されても、既に書き込まれているメッセージは影響を受けません。

MQGET 呼び出しの戻り値は、常にゼロ以上です。値 PRQDEF が返されることはありません。

ローカル・キュー・マネージャーでサポートされている最大の優先順位 (この最大値は **MaxPriority** キュー・マネージャー属性で指定される) より高い優先順位でメッセージが書き込まれた場合、メッセージはキュー・マネージャーで受け入れられますが、キュー・マネージャーの最大優先順位でキューに入れられます。MQPUT 呼び出しまたは MQPUT1 呼び出しの完了時には、CCWARN および理由コード RC2049 が出力されます。しかし、MDPRI フィールドの値は、メッセージを書き込んだアプリケーションが指定した値のままです。

メッセージに応答する場合は、アプリケーションは通常、応答メッセージに対して、要求メッセージの優先順位を使用します。他の状況では、PRQDEF を設定しても、アプリケーションを変更することなく優先順位の調整を行うことができます。

これは、MQGET 呼び出しでは出力フィールド、MQPUT および MQPUT1 呼び出しでは入力フィールドです。このフィールドの初期値は PRQDEF です。

MDPT (8 バイトの文字ストリング)

メッセージが書き込まれた時間。

これは、メッセージの**起点コンテキスト**の一部です。メッセージ・コンテキストについての詳細は、[メッセージ・コンテキストおよびコンテキスト情報の制御](#)を参照してください。

キュー・マネージャーがこのフィールドを生成する際に使用する時刻の形式は、以下のとおりです。

- HHMMSSSTH

文字は、順に以下のものを表します。

HH

時間 (00 から 23 まで)

MM

分 (00 から 59 まで)

SS

秒 (00 から 59 まで、[注](#)を参照)

T

10 分の 1 秒 (0 から 9 まで)

H

100 分の 1 秒 (0 から 9 まで)

注: システム・クロックが非常に正確な時間標準に同期しているときは、ごくまれですが、MDPT に秒数として 60 または 61 が戻ることがあります。これは、グローバル時間標準にうるう秒が挿入されたときに発生します。

MDPD と MDPT フィールドでは、グリニッジ標準時 (GMT) に正確に設定されているシステム・クロックに従って、GMT が使用されます。

メッセージが作業単位の一部として書き込まれた場合は、作業単位がコミットされた時刻ではなく、メッセージが書き込まれた時刻です。

MQPUT および MQPUT1 呼び出しの場合、**PMO** パラメーターに または **PMSETA** が指定されていれば、これは入出力フィールドです。フィールドの内容は、キュー・マネージャーによって検査されませんが、フィールド内のヌル文字のあとにある情報はいずれも廃棄されます。ヌル文字およびその後に続く文字は、キュー・マネージャーによってブランクに変換されます。**PMSETA** が指定されていない場合、入力においてこのフィールドは無視され、出力専用フィールドになります。

MQPUT または MQPUT1 呼び出しが正常に完了すると、メッセージがキューに書き込まれた場合、このフィールドには、そのメッセージと共に送信された MDPT の値が入ります。これは、メッセージが保存された場合 (保存パブリケーションについて詳しくは、**PMRET** の説明を参照してください)、そのメッセージと共に保持された MDPT の値になりますが、メッセージがパブリケーションとしてサブスクライバーに送信された場合には MDPT として使用されません。これは、サブスクライバーが提供する値が、サブスクライバーに送信されるすべてのパブリケーションにおいて MDPT をオーバーライドするためです。メッセージがコンテキストを持っていない場合、フィールドは完全にブランクになります。

これは、MQGET 呼び出しの出力フィールドです。このフィールドの長さは LNPTIM によって指定されます。このフィールドの初期値は 8 個のブランク文字です。

MDREP (10 桁の符号付き整数)

レポート・メッセージのオプション。

レポート・メッセージは、他のメッセージに関するメッセージであり、元のメッセージに関係する予定されたイベントまたは予定されていないイベントについてアプリケーションに知らせるために使用されます。MDREP フィールドにより、元のメッセージを送信したアプリケーションが、どのレポートが必要か、そのレポートにアプリケーション・メッセージ・データを含めるかどうか、また、(レポートおよび応答の両方に関して) レポート・メッセージまたは応答メッセージの内のメッセージおよび関連 ID を設定する方法について指定することができます。以下のタイプのレポート・メッセージのいずれかまたはすべてを要求する (またはどれも要求しない) ことができます。

- 例外
- 有効期限
- 到着確認 (COA)
- 送達時に確認 (COD)
- 肯定アクション通知 (PAN)
- 否定アクション通知 (NAN)

複数のタイプのレポート・メッセージが必要な場合、またはその他のレポート・オプションが必要な場合、値を全部加えることができます (同じ定数は 2 回以上加えないでください)。

レポート・メッセージを受け取ったアプリケーションでは、MQMD の MDFB フィールドを調べることで、報告が生成された理由を判別できます。詳細については、MDFB フィールドを参照してください。

メッセージをトピックに書き込むときにレポート・オプションを使用すると、ゼロ個以上のレポート・メッセージが生成され、アプリケーションに送信される可能性があります。これは、パブリケーション・メッセージが、ゼロ個以上のサブスクライブ・アプリケーションに送信される可能性があるためです。

例外オプション: 次のオプションのいずれかを指定して、例外レポート・メッセージを要求することができます。

ROACTIVITY

必要なアクティビティ報告書

このレポート・オプションを使用すると、サポートされるアプリケーションがこのレポート・オプションが設定されたメッセージを処理する場合に、アクティビティ報告書を生成することが可能になります。

このレポート・オプションが設定されたメッセージは、そのオプションが「認識」されない場合でも、任意のキュー・マネージャーによって受け入れられる必要があります。これによって、従来のキュー・マネージャーによって処理される場合でも、レポート・オプションを任意のユーザー・メッセージで設定することが可能となります。これを行うため、レポート・オプションは ROAUM サブフィールドに置かれます。

ROACT が設定されたメッセージでプロセス (キュー・マネージャーまたはユーザー・プロセスのいずれか) がアクティビティを実行する場合、プロセスはアクティビティ報告書を生成して書き込むよう選択することができます。

アクティビティ報告書オプションを使用すると、キュー・マネージャー・ネットワークを通して任意のメッセージの経路をトレースできるようになります。レポート・オプションは、任意の現行ユーザー・メッセージで指定でき、メッセージのネットワーク経路の計算を即時に開始できるようになります。メッセージを生成するアプリケーションがアクティビティ・レポートの生成を使用可能にすることができない場合は、キュー・マネージャー管理者が提供する API 交差点を使用することにより、それを使用可能にすることができます。

以下のいくつかの条件がアクティビティ報告書に適用されます。

1. ネットワーク内でアクティビティ報告書を生成できるキュー・マネージャーが少ない場合、経路は大まかになります。
2. 取られる経路を判別するために、アクティビティ報告書を簡単に「オーダーできない」可能性があります。
3. アクティビティ報告書は、その要求された宛先への経路を検索できない可能性があります。

ROEXC

例外レポートが必要です。

このタイプのレポートは、メッセージが別のキュー・マネージャーに送信されたのに、指定された宛先キューに送達することができない場合に、メッセージ・チャネル・エージェントによって生成されます。例えば、ターゲット・キューまたは中間伝送キューがいっぱいである場合や、メッセージが大きすぎてキューに入らない場合などがあります。

例外レポート・メッセージが生成されるかどうかは、元のメッセージの持続性と、元のメッセージが経由するメッセージ・チャネルの速度 (通常または高速) によって次のように異なります。

- すべての持続性メッセージと、通常メッセージ・チャネルを経由する非持続性メッセージについては、送信側アプリケーションでエラー条件に対して指定されたアクションが正常に完了した場合にのみ、例外レポートが生成されます。送信側アプリケーションでは、エラー条件が発生したときの元のメッセージの後処理を制御するために、次のアクションのいずれかを指定できます。

- RODLQ (元のメッセージを送達不能キューに格納します)。
- RODISC (元のメッセージを廃棄します)。

送信側アプリケーションで指定したアクションを正常に完了できなかった場合は、元のメッセージは伝送キューに残され、例外レポート・メッセージは生成されません。

- 高速メッセージ・チャネルを経由する非持続性メッセージについては、エラー条件に対して指定されたアクションを正常に完了できない場合でも、元のメッセージが伝送キューから削除され、例外レポートが生成されます。例えば、RODLQ が指定されている場合に、(例えば) 送達不能キューが満杯であるなどの理由で、元のメッセージを送達不能キューに格納できなかったときには、例外レポート・メッセージが生成され、元のメッセージが廃棄されます。

通常メッセージ・チャンネルおよび高速メッセージ・チャンネルについて詳しくは、[メッセージの持続性](#)を参照してください。

MQPUT または MQPUT1 呼び出しから戻された理由コードを用いて、元のメッセージを書き込んだアプリケーションに問題を同期的に通知できる場合、例外レポートは生成されません。

また、アプリケーションは、例外レポートを送信することにより、受け取ったメッセージを処理できないということを知らせることもできます (例えばクレジット払いのときに、引き落とし額が預金額を超えてしまうような場合です)。

元のメッセージのメッセージ・データは、レポート・メッセージに組み込まれません。

ROEXC、ROEXCD、ROEXCF のうち 2 つ以上を指定しないでください。

ROEXCD

データ付きの例外レポートが必要です。

元のメッセージからアプリケーション・メッセージ・データの最初の 100 バイトがレポート・メッセージ内に組み込まれることを除けば、ROEXC と同じです。元のメッセージに 1 つ以上の MQ ヘッダー構造が含まれている場合、それらはアプリケーション・データの 100 バイトに加えて、レポート・メッセージに組み込まれます。

ROEXC、ROEXCD、ROEXCF のうち 2 つ以上を指定しないでください。

ROEXCF

全データの例外レポートが必要です。

元のメッセージからアプリケーション・メッセージ・データのすべてがレポート・メッセージ内に組み込まれることを除けば、ROEXC と同じです。

ROEXC、ROEXCD、ROEXCF のうち 2 つ以上を指定しないでください。

満了オプション: 次のオプションのいずれかを指定して、満了レポート・メッセージを要求することができます。

ROEXP

満了レポートが必要です。

このタイプのレポートは、満了時間がすぎたためにアプリケーションに送達する前にメッセージが廃棄される場合、キュー・マネージャーにより生成されます (MDEXP フィールドを参照してください)。この理由のため廃棄されたメッセージがあっても (ROEXC* オプションのいずれかが指定された場合であっても)、このオプションが設定されていない場合は、レポート・メッセージは生成されません。

元のメッセージのメッセージ・データは、レポート・メッセージに組み込まれません。

ROEXP、ROEXPD、ROEXPF のうち 2 つ以上を指定しないでください。

ROEXPD

満了レポート (データ付き) が必要です。

元のメッセージからアプリケーション・メッセージ・データの最初の 100 バイトがレポート・メッセージ中に組み込まれることを除けば、ROEXP と同じです。元のメッセージに 1 つ以上の MQ ヘッダー構造が含まれている場合、それらはアプリケーション・データの 100 バイトに加えて、レポート・メッセージに組み込まれます。

ROEXP、ROEXPD、ROEXPF のうち 2 つ以上を指定しないでください。

ROEXPF

満了レポート (全データ付き) が必要です。

元のメッセージからアプリケーション・メッセージ・データのすべてがレポート・メッセージ内に組み込まれることを除けば、ROEXP と同じです。

ROEXP、ROEXPD、ROEXPF のうち 2 つ以上を指定しないでください。

到着時確認オプション: 次のオプションのいずれかを指定して、到着時確認レポート・メッセージを要求することができます。

ROCOA

到着時確認レポートが必要です。

このタイプのレポートは、メッセージが宛先キューに置かれる時、宛先キューを所有するキュー・マネージャーにより生成されます。元のメッセージのメッセージ・データは、レポート・メッセージに組み込まれません。

メッセージが作業単位の一部として書き込まれ、しかも宛先キューがローカル・キューであるという場合、キュー・マネージャーによって生成された COA レポート・メッセージは、その作業単位がコミットされているときのみ取り出すことができます。

メッセージ記述子の MDFMT フィールドが FMXQH または FMDLH の場合、COA レポートは生成されません。これにより、メッセージが伝送キューに書き込まれる場合、またはメッセージが配信不能で送達不能キューに書き込まれる場合に、COA レポートは生成されなくなります。

ROCOA、ROCOAD、ROCOAF のうち 2 つ以上を指定しないでください。

ROCOAD

到着時確認レポート (データ付き) が必要です。

元のメッセージからアプリケーション・メッセージ・データの最初の 100 バイトがレポート・メッセージ内に組み込まれることを除けば、ROCOA と同じです。元のメッセージに 1 つ以上の MQ ヘッダー構造が含まれている場合、それらはアプリケーション・データの 100 バイトに加えて、レポート・メッセージに組み込まれます。

ROCOA、ROCOAD、ROCOAF のうち 2 つ以上を指定しないでください。

ROCOAF

到着時確認レポート (全データ付き) が必要です。

元のメッセージからアプリケーション・メッセージ・データのすべてがレポート・メッセージ内に組み込まれることを除けば、ROCOA と同じです。

ROCOA、ROCOAD、ROCOAF のうち 2 つ以上を指定しないでください。

廃棄および満了オプション: 以下のオプションを指定して、レポート・メッセージ用の満了時間および廃棄フラグを設定できます。

ROPDAE

レポート・メッセージの満了時間および廃棄フラグを設定します。

このオプションを使用すると、レポート・メッセージおよび応答メッセージは、元のメッセージから満了時間および廃棄フラグ (廃棄するかどうか) を継承します。このオプションを設定すると、レポート・メッセージおよび応答メッセージは以下のようになります。

1. RODISC フラグを継承します (設定されている場合)。
2. メッセージが満了レポートではない場合、メッセージの残りの満了時間を継承します。メッセージが満了レポートである場合、満了時間は 60 秒に設定されます。

このオプションを設定すると、以下が適用されます。

注:

1. レポートおよび応答メッセージは、廃棄フラグおよび満了値と共に生成され、システム内に残しておくことはできません。
2. トレース経路メッセージは、トレース経路が有効ではないキュー・マネージャーの宛先キューには到達できなくなります。
3. 通信リンクが壊れている場合でも、送達できないレポートでキューがいっぱいになることはありません。
4. コマンド・サーバー応答は要求の残りの満了時間を継承します。

配布時確認オプション: 次のオプションのいずれかを指定して、配布時確認レポート・メッセージを要求することができます。

ROCOD

配布時確認レポートが必要です。

このタイプのレポートは、メッセージをキューから削除する方法でアプリケーションが宛先キューからのメッセージを検索する時に、キュー・マネージャーにより生成されます。元のメッセージのメッセージ・データは、レポート・メッセージに組み込まれません。

メッセージが作業単位の一部として検索される場合、レポート・メッセージは同じ作業単位内に生成され、作業単位がコミットされるまではそのレポートが利用できなくなります。作業単位がバックアウトされる場合、レポートは送信されません。

メッセージ記述子の MDFMT フィールドが FMDLH の場合、COD レポートは生成されません。これにより、メッセージが配布できなくなり、送達不能キューに書き込まれる場合は、COD レポートが生成されなくなります。

宛先キューが XCF キューの場合は、ROCOD は無効です。

ROCOD、ROCODD、ROCODF のうち 2 つ以上を指定しないでください。

ROCODD

配布時確認レポート (データ付き) が必要です。

元のメッセージからアプリケーション・メッセージ・データの最初の 100 バイトがレポート・メッセージ内に組み込まれることを除けば、ROCOD と同じです。元のメッセージに 1 つ以上の MQ ヘッダー構造が含まれている場合、それらはアプリケーション・データの 100 バイトに加えて、レポート・メッセージに組み込まれます。

元のメッセージについての MQGET 呼び出しで GMATM を指定し、戻されたメッセージに切り捨てが行われた場合、レポート・メッセージに置かれる アプリケーション・メッセージ・データの量は、以下のうち最小のものになります。

- 元のメッセージの長さ
- 100 バイト

宛先キューが XCF キューの場合は、ROCODD は無効です。

ROCOD、ROCODD、ROCODF のうち 2 つ以上を指定しないでください。

ROCODF

配布時確認レポート (全データ付き) が必要です。

元のメッセージからアプリケーション・メッセージ・データのすべてがレポート・メッセージ内に組み込まれることを除けば、ROCOD と同じです。

宛先キューが XCF キューの場合は、ROCODF は無効です。

ROCOD、ROCODD、ROCODF のうち 2 つ以上を指定しないでください。

アクション通知オプション: 次のいずれか (または両方) のオプションを指定して、受信側のアプリケーションからアクションの正常終了または異常終了を通知するレポート・メッセージを送信するよう要求できます。

ROPAN

アクション正常終了通知レポートが必要です。

このタイプのレポートは、メッセージを取り出し、そのメッセージに従ってアクションを実行するアプリケーションにより生成されます。これは、メッセージで要求されたアクションが正常に実行されたことを示します。レポートを生成するアプリケーションは、レポートにデータを含めるかどうかを決定します。

メッセージを取り出すアプリケーションにこの要求を送らないと、キュー・マネージャーはこのオプションに基づくアクションを実行しません。ただし、レポートの生成が必要な場合には、メッセージを取り出すアプリケーションが行います。

RONAN

アクション異常終了通知レポートが必要です。

このタイプのレポートは、メッセージを取り出し、そのメッセージに従ってアクションを実行するアプリケーションにより生成されます。これは、メッセージで要求されたアクションが正常に実行されなかったことを示します。レポートを生成するアプリケーションは、レポートにデータを含め

るかどうかを決定します。例えば、要求を実行できなかった理由を示すデータを付けた方が望ましい場合もあります。

メッセージを取り出すアプリケーションにこの要求を送らないと、キュー・マネージャーはこのオプションに基づくアクションを実行しません。ただし、レポートの生成が必要な場合には、メッセージを取り出すアプリケーションが行います。

アクションの正常終了通知時および異常終了通知時、それぞれの条件の指定はアプリケーションが行います。要求が一部実行された場合には PAN 報告ではなく NAN 報告を生成する (要求された場合) ことをお勧めします。また、すべての可能な条件を、アクションの正常終了通知時と異常終了通知時の両方ではなく、いずれか一方に対応付けることをお勧めします。

メッセージ ID オプション: 次のオプションのいずれかを指定して、レポート・メッセージ (または応答メッセージ) の MDMID を設定する方法を制御することができます。

RONMI

新しいメッセージ ID。

これはデフォルトの処置であり、このメッセージの結果としてレポートまたは応答が生成された場合に、レポート・メッセージまたは応答メッセージの MDMID を新たに生成することを示します。

ROPMI

メッセージ ID を渡します。

レポートまたは応答がこのメッセージの結果として生成される場合、このメッセージの MDMID をレポート・メッセージまたは応答メッセージの MDMID にコピーすることになります。

パブリケーション・メッセージの MsgId は、パブリケーションのコピーを受け取る各サブスクライバーごとに異なるため、レポートまたは応答メッセージにコピーされる MsgId もそれぞれ異なります。

このオプションを指定しない場合は、RONMI を指定したものと見なされます。

相関 ID オプション: 次のオプションのいずれかを指定して、レポート・メッセージ (または応答メッセージ) の MDCID を設定する方法を制御することができます。

ROCMTC

メッセージ ID を相関 ID にコピーします。

これはデフォルトの処置であり、このメッセージの結果としてレポートまたは応答が生成された場合に、このメッセージの MDMID をレポート・メッセージまたは応答メッセージの MDCID にコピーすることを示します。

パブリケーション・メッセージの MsgId は、パブリケーションのコピーを受け取る各サブスクライバーごとに異なるため、レポートまたは応答メッセージの CorrelId にコピーされる MsgId もそれぞれ異なります。

ROPCI

相関 ID を渡します。

レポートまたは応答がこのメッセージの結果として生成される場合、このメッセージの MDCID をレポート・メッセージまたは応答メッセージの MDCID にコピーすることになります。

パブリケーション・メッセージの MDCID は、それが SOSCID オプションを使用し、MQSD の SCDIC フィールドを CINONE に設定する場合を除き、サブスクライバー固有のものになります。このため、レポート・メッセージまたは応答メッセージの MDCID にコピーされる MDCID は、各サブスクライバーによって異なる可能性があります。

このオプションを指定しない場合、ROCMTC を指定したものと見なされます。

要求に応答するサーバー、またはレポート・メッセージを生成するサーバーは、元のメッセージの中に ROPCI または ROPMI オプションが設定されていたかどうかを調べるようにしてください。設定されていた場合、サーバーは、これらのオプションで説明されている処置を取るはずで、いずれも設定されていない場合、サーバーは、対応するデフォルトの処置を取るはずで、

: 次のオプションのいずれかを指定して、元のメッセージを宛先キューに配布できない場合のそのメッセージの後処理を制御することができます。これらのオプションが適用されるのは、送信側アプリケーションが例外レポート・メッセージを要求した場合に、そのメッセージが生成中となるような状況の場合のみです。アプリケーションは処理オプションを、例外レポートの要求とは無関係に設定できません。

RODLQ

送達不能キューにメッセージを入れます。

これは、デフォルトの処置であり、メッセージを宛先キューに送達できない場合、メッセージが送達不能キューに入れられることを示します。これは次のような場合に行われます。

- MQPUT または MQPUT1 呼び出しによって戻された理由コードを用いて、元のメッセージを書き込んだアプリケーションに問題を同期的に通知できない場合。送信側によって要求される場合は、例外レポート・メッセージが生成されます。
- 元のメッセージを書き込むアプリケーションがトピックに書き込んでいた場合。

送信側からの要求があった場合には、例外レポート・メッセージが生成されます。

RODISC

メッセージを廃棄します。

これは、宛先キューに配布できないメッセージは廃棄する必要があることを示します。これは次のような場合に行われます。

- MQPUT または MQPUT1 呼び出しによって戻された理由コードを用いて、元のメッセージを書き込んだアプリケーションに問題を同期的に通知できない場合。送信側によって要求される場合は、例外レポート・メッセージが生成されます。
- 元のメッセージを書き込むアプリケーションがトピックに書き込んでいた場合。

送信側からの要求があった場合には、例外レポート・メッセージが生成されます。

元のメッセージを送達不能キューに入れずに、元のメッセージを送信側に返す必要がある場合、送信側は ROEXCF を付けて RODISC を指定する必要があります。

デフォルト・オプション: レポート・オプションが不要な場合には、次の値を指定できます。

RONONE

レポートは必要ありません。

この値は、他のオプションを指定しなかったことを示します。RONONE は、プログラム文書化を支援するために定義されています。このオプションを他のオプションと組み合わせて使用することは意図されていませんが、値がゼロであるため、そのような使い方をしても検出できません。

一般情報:

1. 必要とされるすべてのレポート・タイプは、元のメッセージを送信するアプリケーション具体的に指定されている必要があります。例えば、COA レポートは要求されているが、例外レポートは要求されないという場合は、メッセージが宛先キューに入れられるときに COA レポートが生成されますが、メッセージが宛先キューに到着したときに宛先キューがいっぱいであっても、例外レポートは生成されません。MDREP オプションが設定されていない場合、キュー・マネージャーまたはメッセージ・チャンネル・エージェント (MCA) によって、レポート・メッセージが生成されることはありません。

一部のレポート・オプションは、ローカル・キュー・マネージャーで認識できなくても指定できます。これは、宛先キュー・マネージャーでオプションを処理する場合に有用です。詳しくは、[1433 ページの『IBM iでのレポート・オプションおよびメッセージ・フラグ』](#)を参照してください。

レポート・メッセージが要求される場合、レポートの送信先のキューの名前を MDRQ フィールドで指定する必要があります。レポート・メッセージが受信されると、メッセージ記述子の MDFB フィールドを調べることで、レポートの性質を調べることができます。

2. レポート・メッセージを生成するキュー・マネージャーまたは MCA が、レポート・メッセージを応答キューに入れることができない場合 (例えば、応答キューまたは伝送キューが満杯になっているため)、レポート・メッセージは、代わりに送達不能キューに入ります。これにも失敗した場合、ある

いは送達不能キューがない場合に、取るべきアクションは、レポート・メッセージのタイプによって決まります。

- レポート・メッセージが例外レポートである場合は、その例外レポートを生成させるメッセージは伝送キュー上に残ったままになります。これにより、メッセージが失われなくなります。
- 他のすべてのレポート・タイプについては、レポート・メッセージは廃棄され、処理は引き続き正常に行われます。処理を実行する理由は、元のメッセージがすでに無事に送達されているか (COA または COD レポート・メッセージの場合)、元のメッセージがすでに対象外になっているか (満了レポート・メッセージの場合) のいずれかです。

レポート・メッセージが無事にキュー (宛先キューまたは中間伝送キューのいずれか) に入ると、そのメッセージには特別な処理は行われなくなります (他のメッセージと全く同じ扱いになります)。

3. レポートが生成されると、MDRQ キューがオープンされ、レポートを生成させたメッセージの MQMD の MDUID 許可を使用して、レポート・メッセージが書き込まれます。ただし、以下の場合には例外です。

- 受信側の MCA によって生成される例外レポートは、レポートを生成させたメッセージを書き込むために MCA が使用したのと同じ許可を使って書き込まれます。使用されるユーザー ID は、CDPA チャンネル属性によって決まります。
- キュー・マネージャーによって生成される COA レポートは、レポートを生成させたメッセージをそのキュー・マネージャーに書き込むときに使用されたのと同じ権限を使って書き込まれます。例えば、受信側の MCA がその MCA のユーザー ID を使ってメッセージを書き込んだ場合、キュー・マネージャーは MCA のユーザー ID を使って COA レポートを書き込みます。

レポートを生成するアプリケーションは、通常、応答を生成する場合と同じ許可を持っています。これは、通常、元のメッセージのユーザー ID に与えられた許可と同じです。

レポートがリモート宛先に移動しなければならない場合は、他のメッセージに対する場合と同じ方式で、それを受け取るかどうかを、送信側と受信側が決めることができます。

4. データ付きのレポート・メッセージが要求される場合は、以下のことが行われます。

- レポート・メッセージは、常に、元のメッセージの送信側が要求したデータ量で生成されます。レポート・メッセージが応答先キューに対して大きすぎると、上記の処理が行われます。レポート・メッセージは、応答キューに適合するように切り捨てられることはありません。
- 元のメッセージの MDFMT が FMXQH の場合、レポート内のデータは、MQXQH を含んでいません。レポート・データは、元のメッセージにある MQXQH の後にあるデータの最初のバイトから始まります。これは、キューが伝送キューかどうかに関係なく起こります。

5. COA、COD、または満了レポート・メッセージが、応答キューで受け取られた場合は、元のメッセージが、適切に到着した、送達された、または満了したことが保証されます。ただし、これらのレポート・メッセージの 1 つ以上が要求され、受信されていない場合は、以下に示すいずれかが発生した可能性があるため、逆を想定することはできません。

- a. リンクがダウンしたために、レポート・メッセージは中止された。
- b. 中間伝送キューまたは応答キューに、ブロッキング条件 (キューがいっぱいである、書き込みが禁止されている、など) が存在しているため、レポート・メッセージは中止された。
- c. レポート・メッセージが送達不能キューに入っている。
- d. キュー・マネージャーがレポート・メッセージを生成しようとしたとき、該当するキューにメッセージを書き込まず、また送達不能キュー上にも書き込めなかったため、レポート・メッセージを生成することができませんでした。
- e. 報告される処置 (到着、送達、または満了) と該当するレポート・メッセージの生成との間に、キュー・マネージャーの障害が発生しました。 (COD レポート・メッセージは同じ作業単位内で生成されるので、COD レポート・メッセージではアプリケーションが作業単位内で元のメッセージを取り出す場合は上記のことは起こりません。)

例外レポート・メッセージも、上記の 1、2 および 3 の理由から、同様に中止されることがあります。しかし、MCA が、例外レポート・メッセージを生成できない (応答キューまたは送達不能キューのいずれかにレポート・メッセージを書き込むことができない) 場合は、元のメッセージは送信側

の伝送キューにとどまり、チャンネルがクローズされます。このような状況は、レポート・メッセージがチャンネルの送信側、受信側のどちらで生成されても起こります。

6. 元のメッセージが一時的にブロックされていても (その結果として例外レポート・メッセージが生成され、元のメッセージは送達不能キューに書き込まれる)、そのブロックがクリアされ、アプリケーションが元のメッセージを送達不能キューから読み込み、その宛先に再度メッセージを書き込む場合は、以下に示す動作が発生する場合があります。

- 例外レポート・メッセージは生成されたが、最終的に元のメッセージは、その宛先に正しく到着する。
- 元のメッセージは、あとで別のブロック状態になる可能性があるため、1つの元のメッセージに対して、例外レポート・メッセージが2つ以上生成されます。

トピックに書き込むときのレポート・メッセージ:

1. メッセージをトピックに書き込むときに、レポートを生成することができます。このメッセージはトピックのすべてのサブスクライバーに送信されます。ゼロの場合もあれば、1またはそれ以上の場合もあります。結果として多数のレポート・メッセージが生成される場合があるので、レポート・オプションを使用するように選択する場合に、これを考慮に入れる必要があります。
2. メッセージをトピックに書き込むときに、メッセージのコピーを置く宛先キューが多数存在する場合があります。そのうちのいくつかの宛先キューに問題 (例えばキュー・フル) がある場合、MQPUTの実行が成功するかどうかは (メッセージの持続性に応じて) NPMSGDLV または PMSGDLV の設定に依存します。宛先キューへのメッセージ送達が必ず成功するという設定 (例えばそれが永続サブスクライバーへの持続メッセージで、PMSGDLV が ALL または ALLDUR に設定されている) の場合、成功とは、次の基準のいずれかが満たされる場合として定義されます。

- サブスクライバー・キューへの書き込みが成功する
- サブスクライバー・キューがメッセージを受け取ることができない場合、RODLQ を使用して送達不能キューへの書き込みが正常に行われる
- サブスクライバー・キューがメッセージを受け取ることができない場合に RODISC を使用する

メッセージ・セグメントのレポート・メッセージ:

1. セグメント化が許可されているメッセージの報告メッセージを要求できます (MFSEGA フラグを参照してください)。キュー・マネージャーがメッセージをいくつかのセグメントに分割する必要があると判断した場合には、関連する条件に応じてセグメントごとにレポート・メッセージが生成されます。そのため、アプリケーションでは、要求された各タイプの複数のレポート・メッセージを受信できるようにしてください。レポート・メッセージの MDGID フィールドを使用することによって、元のメッセージのグループ ID を複数の報告と関連付けることができ、MDFB フィールドを使用することによって、各レポート・メッセージのタイプを識別できます。
2. GMLOGO を使用してセグメントに関するレポート・メッセージを取り出す場合には、MQGET 呼び出しを続けて発行したときに異なるタイプのレポートが戻されることがあるので注意してください。例えば、キュー・マネージャーによってセグメントに分割されたメッセージに対して COA レポートおよび COD レポートを要求されている場合、それらのレポート・メッセージに対して MQGET 呼び出しを発行したときに COA レポート・メッセージと COD レポート・メッセージが順不同で混ざり合っていることがあります。この問題を回避するには、GMCMPM オプションを使用します (GMATM を組み合わせても構いません)。GMCMPM を使用すると、キュー・マネージャーは同じレポート・タイプのレポート・メッセージを組み立てます。例えば、最初の MQGET 呼び出しで元のメッセージに関連するすべての COA メッセージの再組み立てを実行し、2 番目の MQGET 呼び出しですべての COD メッセージの再組み立てを実行する場合などが考えられます。いずれかが先に組み立てられるかは、どちらのタイプのレポート・メッセージが先にキューに入るかによります。
3. アプリケーション自体がセグメントを書き込む場合、そのアプリケーションではセグメントごとに異なるレポート・オプションを指定できます。ただし、以下に示す点に注意してください。

- GMCMPM オプションを使用してセグメントを取り出している場合、キュー・マネージャーに格納されるのは、最初のセグメント内のレポート・オプションのみです。
- セグメントを一度に1つずつ取り出している場合に、大部分のセグメントに ROCOD* オプションが指定されていても、このオプションが指定されていないセグメントが1つでもあれば、GMCMPM オプションを使用して一度の MQGET 呼び出しでレポート・メッセージを取り出したり、GMASGA

オプションを使用してすべてのレポート・メッセージが到着した時期を検出したりすることができなくなります。

4. MQ ネットワークでは、個々のキュー・マネージャーが異なる機能を持つことができます。セグメント化をサポートしていないキュー・マネージャーまたは MCA によりセグメントに関するレポート・メッセージが生成された場合、そのキュー・マネージャーまたは MCA は、デフォルト時には、レポート・メッセージの中に必要なセグメント情報を含めません。そのため、このレポート・メッセージのもとになったメッセージを識別することが困難な場合があります。この問題を回避するには、レポート・メッセージと共にデータを要求 (RO*D オプションと RO*F オプションのうちいずれか該当する方を指定) します。ただし、RO*D を指定した場合には、アプリケーションがレポート・メッセージを取り出すときに 100 バイト未満のアプリケーション・メッセージ・データがアプリケーションに戻されることがあるので注意してください (レポート・メッセージがセグメント化をサポートしていないキュー・マネージャーまたは MCA により生成された場合)。

レポート・メッセージのメッセージ記述子の内容: キュー・マネージャーまたはメッセージ・チャネル・エージェント (MCA) は、レポート・メッセージを生成するとき、メッセージ記述子の中のフィールドを以下の値に設定してから、通常の方法でメッセージを書き込みます。

表 747. レポート・メッセージがシステム生成される場合に MQMD フィールドに使用される値

MQMD のフィールド	使用される値
MDSID	MDSIDV
MDVER	MDVER2
MDREP	RONONE
MDMT	MTRPRT
MDEXP	EIULIM
MDFB	レポートの性質に適した値 (FBCOA、FBCOD、FBEXP、または RC* のいずれかの値)
MDENC	元のメッセージ記述子からコピーされます。
MDCSI	元のメッセージ記述子からコピーされます。
MDFMT	元のメッセージ記述子からコピーされます。
MDPRI	元のメッセージ記述子からコピーされます。
MDPER	元のメッセージ記述子からコピーされます。
MDMID	元のメッセージ記述子のレポート・オプションで指定されたもの
MDCID	元のメッセージ記述子のレポート・オプションで指定されたもの
MDBOC	0
MDRQ	ブランク
MDRM	キュー・マネージャーの名前。
MDUID	PMPASI オプションが設定したとおり
MDACC	PMPASI オプションが設定したとおり
MDAID	PMPASI オプションが設定したとおり
MDPAT	ATQM、またはメッセージ・チャネル・エージェントに対応するもの
MDPAN	キュー・マネージャー名またはメッセージ・チャネル・エージェント名の最初の 28 バイト。IMS ブリッジで生成されたレポート・メッセージの場合、このフィールドには、メッセージに関連する IMS システムの XCF グループ名と XCF メンバー名が入っています。
MDPD	レポート・メッセージが送信された日付
MDPT	レポート・メッセージが送信された時刻

表 747. レポート・メッセージがシステム生成される場合に MQMD フィールドに使用される値 (続き)

MQMD のフィールド	使用される値
MDA0D	ブランク
MDGID	元のメッセージ記述子からコピーされます。
MDSEQ	元のメッセージ記述子からコピーされます。
MDOFF	元のメッセージ記述子からコピーされます。
MDMFL	元のメッセージ記述子からコピーされます。
MDOLN	OLUNDF 以外の場合は、元のメッセージ記述子からコピーされる。 OLUNDF の場合は、元のメッセージ・データの長さが設定されます。

レポートを生成するアプリケーションでは、以下を除いて、同様の値を設定することをお勧めします。

- MDRM フィールドをブランクに設定できます (メッセージが書き込まれる時に、キュー・マネージャーは、この値をローカル・キュー・マネージャーの名前に変更します)。
- コンテキスト・フィールドは、応答に使用されたオプション、通常は PMPASI を使用して設定する必要があります。

レポート・フィールドの分析: MDREP フィールドにはサブフィールドが含まれます。そのため、メッセージの送信側が特定のレポートを要求したかどうかを調べる必要があるアプリケーションでは、[1435 ページの『IBM iでのレポート・フィールドの分析』](#)で説明する技法のうちの 1 つを使用してください。

これは、MQGET 呼び出しでは出力フィールド、MQPUT および MQPUT1 呼び出しでは入力フィールドです。このフィールドの初期値は RONONE です。

MDRM (48 バイトの文字ストリング)

応答キュー・マネージャーの名前。

これは、応答メッセージまたはレポート・メッセージを送信されるキュー・マネージャーの名前です。MDRQ は、このキュー・マネージャーで定義されるキューのローカル名です。

MDRM フィールドがブランクである場合、ローカル・キュー・マネージャーは、そのキュー定義自体の **MDRQ** 名を調べます。この名前を持つリモート・キューのローカル定義が存在している場合は、伝送されたメッセージの中の **MDRM** 値が、リモート・キューの定義内の **RemoteQMGrName** 属性の値によって置換されます。受信側のアプリケーションがメッセージの MQGET 呼び出しを出すと、メッセージ記述子にこの値が返されます。リモート・キューのローカル定義が存在しない場合、メッセージと共に伝送される MDRM は、ローカル・キュー・マネージャーの名前になります。

名前を指定するときは、末尾ブランクを付けることができます。最初のヌル文字およびその後続く文字は、ブランクとして扱われます。しかし、それ以外の場合は、名前がキューの命名規則を満たすかどうかを調べる検査は行われません。伝送されたメッセージ内の **MDRM** が置換される場合、このことは伝送された名前についても同様です。

応答先キューが必要でない場合は、MDRM フィールドをブランクに初期化してください (ただし、その検査はありません)。

MQGET 呼び出しの場合、キュー・マネージャーは、常にフィールドの長さまでブランクを埋め込んだ名前を返します。

これは、MQGET 呼び出しでは出力フィールド、MQPUT および MQPUT1 呼び出しでは入力フィールドです。このフィールドの長さは LNQMNM によって指定されます。このフィールドの初期値は 48 個のブランク文字です。

MDRQ (48 バイトの文字ストリング)

応答キューの名前。

これは、メッセージ読み取り要求を発行したアプリケーションが送信する MTRPLY および MTRPRT メッセージの送信先のメッセージ・キューの名前です。この名前は、MDRM で指定されているキュー・マ

ネージャーで定義されているキューのローカル名です。このキューをモデル・キューにすることはできません。ただし、送信側のキュー・マネージャーは、メッセージが書き込まれたときに、これについて確認はしません。

MQPUT および MQPUT1 呼び出しでは、MDMT フィールドに値 MTRQST がある場合、または MDREP フィールドによりいずれかのレポート・メッセージが要求される場合に、このフィールドをブランクにしてはなりません。ただし、メッセージ・タイプにかかわらず、指定された値 (または置き換えられた値) が、メッセージの読み取り要求を発行したアプリケーションに渡されます。

MDRM フィールドがブランクの場合、ローカル・キュー・マネージャーは、独自のキュー定義内で MDRQ 名を探索します。リモート・キューのローカル定義でこの名前が存在する場合、伝送されたメッセージの MDRQ の値は、リモート・キューの定義から取られた **RemoteQName** 属性値で置き換えられ、これらの値は受信側アプリケーションがメッセージの MQGET 呼び出しを発行する際にメッセージ記述子内に戻されます。リモート・キューのローカル定義が存在しない場合、MDRQ は変更されません。

名前を指定するときは、末尾ブランクを付けることができます。最初のヌル文字およびその後続く文字は、ブランクとして扱われます。しかし、名前がそれ以外の点でキュー命名規則を満たすかどうかを調べる検査は行われません。MDRQ が伝送されたメッセージ中に再度挿入される場合も、伝送された名前は同様の扱いとなります。必要に応じて名前が指定されたかどうかのみ検査されます。

応答先キューが必要でない場合は、MDRQ フィールドをブランクに初期化してください (ただし、その検査はありません)。

MQGET 呼び出しの場合、キュー・マネージャーは、常にフィールドの長さまでブランクを埋め込んだ名前を返します。

レポート・メッセージを必要とするメッセージを配布できず、指定されたキューにレポート・メッセージを配布することもできない場合は、元のメッセージとレポート・メッセージの両方が送達不能 (未配布メッセージ) キューに送られます。[1412 ページの『IBM iでのキュー・マネージャーの属性』の DeadLetterQName 属性を参照してください。](#)

これは、MQGET 呼び出しでは出力フィールド、MQPUT および MQPUT1 呼び出しでは入力フィールドです。このフィールドの長さは LNQN によって指定されます。このフィールドの初期値は 48 個のブランク文字です。

MDSEQ (10 桁の符号付き整数)

グループ中の論理メッセージの順序番号。

順序番号は 1 から始まり、グループに新しい論理メッセージが追加されるたびに 1 つずつ大きくなり、最大で 999 999 999 です。グループにまとめられていない物理メッセージの順序番号は 1 です。

MQPUT または MQGET 呼び出しでは、以下の場合には、アプリケーションがこのフィールドを設定する必要はありません。

- MQPUT 呼び出しで、PMLOGO を指定した場合
- MQGET 呼び出しで、MOSEQN が指定されていない場合。

以下、これらの呼び出しをレポート・メッセージ以外のメッセージに使用する場合の推奨方法について説明します。ただし、アプリケーションがさらに制御を要求する場合、または呼び出しが MQPUT1 の場合、アプリケーションは、MDSEQ に適切な値が設定されていることを確認する必要があります。

MQPUT 呼び出しおよび MQPUT1 呼び出しへの入力では、キュー・マネージャーは [表 1](#) で詳述されている値を使用します。MQPUT および MQPUT1 呼び出しの出力時に、キュー・マネージャーは、このフィールドにメッセージと共に送信された値を設定します。

MQGET 呼び出しへの入力では、キュー・マネージャーは [表 1](#) で詳述されている値を使用します。MQGET 呼び出しの出力時に、キュー・マネージャーは、このフィールドに、取り出されたメッセージの値を設定します。

このフィールドの初期値は 1 です。MDVER 値が MDVER2 未満の場合、このフィールドは無視されます。

MDSID (4 バイトの文字ストリング)

構造体 ID

値は次のものでなければなりません。

MDSIDV

メッセージ記述子構造体の ID。

これは常に入力フィールドです。このフィールドの初期値は MDSIDV です。

MDUID (12 バイトの文字ストリング)

ユーザー ID。

これは、メッセージの ID コンテキストの一部です。メッセージ・コンテキストについての詳細は、[メッセージ・コンテキストおよびコンテキスト情報の制御](#)を参照してください。

MDUID には、メッセージを発信したアプリケーションのユーザー ID を指定します。キュー・マネージャーはこの情報を文字データとして扱いますが、そのフォーマットの定義はしません。

メッセージ受信後、後続の MQOPEN または MQPUT1 呼び出しの **OBJDSC** パラメータの ODAU フィールドで使用可能となり MDUID、これにより開く操作を実行するアプリケーションではなく、ユーザー MDUID に対する認証チェックが実行される。

キュー・マネージャーは、MQPUT または MQPUT1 呼び出しについてこの情報を生成する際、環境によって決定されるユーザー ID を使用します。

ユーザー ID は、各環境で次のように決定されます。

-  **z/OS** z/OS では、キュー・マネージャーは次のものを使用します。
 - バッチでは、JES JOB カードまたは開始済みタスクのユーザー ID。
 - TSO では、ログオン・ユーザー ID。
 - CICS の場合、タスクに関連付けられたユーザー ID
 - IMS の場合、ユーザー ID はアプリケーションのタイプによって決まります。

- 回数:

- 非メッセージ BMP 領域
- 非メッセージ IFP 領域
- 成功した GU 呼び出しを発行しなかったメッセージ BMP 領域およびメッセージ IFP 領域

キュー・マネージャーは、領域 JES JOB カードのユーザー ID か、TSO ユーザー ID を使用します。それらの値がブランクまたはヌルの場合には、プログラム仕様ブロック (PSB) の名前を使用します。

- 回数:

- 成功した GU 呼び出しを発行したメッセージ BMP 領域およびメッセージ IFP 領域
- MPP 領域

キュー・マネージャーは、以下のうち 1 つを使用します。

- メッセージに関連付けられたサインオン・ユーザー ID
- 論理端末 (LTERM) 名
- 領域 JES JOB カードのユーザー ID
- TSO のユーザー ID
- PSB 名

-  **IBM i** IBM i では、キュー・マネージャーは、アプリケーション・ジョブに関連付けられたユーザー・プロファイルの名前を使用します。
-  **Linux**  **AIX** AIX and Linux では、キュー・マネージャーは次のものを使用します。

- アプリケーションのログオン名
- ログオンを利用できない場合は、プロセスの実効ユーザー ID
- アプリケーションが CICS トランザクションである場合は、トランザクションに関連付けられたユーザー ID
- VSE/ESA では、これは予約フィールドです。
-  Windows では、キュー・マネージャーはログオン・ユーザー名の最初の 12 文字を使用します。

MQPUT および MQPUT1 呼び出しの場合、**PMO** パラメーターに PMSETI または PMSETA が指定されていれば、これは入出力フィールドです。フィールド内でヌル文字より後の情報はすべて破棄されます。ヌル文字およびその後に続く文字は、キュー・マネージャーによってブランクに変換されます。PMSETI または PMSETA が指定されていない場合、入力においてこのフィールドは無視され、出力専用フィールドになります。

MQPUT または MQPUT1 呼び出しが正常に完了すると、メッセージがキューに書き込まれた場合、このフィールドには、そのメッセージと共に送信された MDUID が入ります。これは、メッセージが保存された場合 (保存パブリケーションについて詳しくは、PMRET の説明を参照してください)、そのメッセージと共に保持された MDUID の値になりますが、メッセージがパブリケーションとしてサブスクライバーに送信された場合には MDUID として使用されません。これは、サブスクライバーが提供する値が、サブスクライバーに送信されるすべてのパブリケーションにおいて MDUID をオーバーライドするためです。メッセージがコンテキストを持っていない場合、フィールドは完全にブランクになります。

これは、MQGET 呼び出しの出力フィールドです。このフィールドの長さは LNUID によって指定されます。このフィールドの初期値は 12 個のブランク文字です。

MDVER (10 桁の符号付き整数)

構造体のバージョン番号。

値は次のいずれかでなければなりません。

MDVER1

バージョン 1 のメッセージ記述子の構造体。

MDVER2

バージョン 2 のメッセージ記述子の構造体。

注: バージョン 2 の MQMD が使用されているときには、アプリケーション・メッセージ・データの先頭に存在する可能性のある MQ ヘッダー構造体に対してキュー・マネージャーが追加の検査を実行します。詳細については、MQPUT 呼び出しの使用上の注意を参照してください。

これより新しいバージョンの構造体にのみ存在するフィールドは、そのフィールドの説明にその旨記載されています。以下の定数は、現行バージョンのバージョン番号を指定しています。

MDVERC

メッセージ記述子の構造体の現行バージョン。

これは常に入力フィールドです。このフィールドの初期値は MDVER1 です。

初期値

表 748. MQMD のフィールド		
フィールド名	定数の名前	定数の値
MDSID	MDSIDV	'MD- - -'
MDVER	MDVER1	1
MDREP	RONONE	0
MDMT	MTDGRM	8

表 748. MQMD のフィールド (続き)

フィールド名	定数の名前	定数の値
MDEXP	EIULIM	-1
MDFB	FBNONE	0
MDENC	ENNAT	環境に依存
MDCSI	CSQM	0
MDFMT	FMNONE	ブランク
MDPRI	PRQDEF	-1
MDPER	PEQDEF	2
MDMID	MINONE	Null
MDCID	CINONE	Null
MDBOC	なし	0
MDRQ	なし	ブランク
MDRM	なし	ブランク
MDUID	なし	ブランク
MDACC	ACNONE	Null
MDAID	なし	ブランク
MDPAT	ATNCON	0
MDPAN	なし	ブランク
MDPD	なし	ブランク
MDPT	なし	ブランク
MDAOD	なし	ブランク
MDGID	GINONE	Null
MDSEQ	なし	1
MDOFF	なし	0
MDMFL	MFNONE	0
MDOLN	OLUNDF	-1
注:		
1. 記号-は、単一のブランク文字を表します。		

RPG 宣言

```

D*.1.....2.....3.....4.....5.....6.....7..
D*
D* MQMD Structure
D*
D* Structure identifier
D MDSID          1          4    INZ('MD ')
D* Structure version number
D MDVER          5          8I 0 INZ(1)
D* Options for report messages
D MDREP          9         12I 0 INZ(0)
D* Message type

```

```

D MDMT 13 16I 0 INZ(8)
D* Message lifetime
D MDEXP 17 20I 0 INZ(-1)
D* Feedback or reason code
D MDFB 21 24I 0 INZ(0)
D* Numeric encoding of message data
D MDENC 25 28I 0 INZ(273)
D* Character set identifier of message data
D MDCSI 29 32I 0 INZ(0)
D* Format name of message data
D MDFMT 33 40 INZ(' ')
D* Message priority
D MDPRI 41 44I 0 INZ(-1)
D* Message persistence
D MDPER 45 48I 0 INZ(2)
D* Message identifier
D MDMID 49 72 INZ(X'00000000000000-
00000000000000000000-
000000000000')
D* Correlation identifier
D MDCID 73 96 INZ(X'00000000000000-
00000000000000000000-
000000000000')
D* Backout counter
D MDBOC 97 100I 0 INZ(0)
D* Name of reply queue
D MDRQ 101 148 INZ
D* Name of reply queue manager
D MDRM 149 196 INZ
D* User identifier
D MDUID 197 208 INZ
D* Accounting token
D MDACC 209 240 INZ(X'00000000000000-
00000000000000000000-
000000000000000000-
000000')
D* Application data relating to identity
D MDAID 241 272 INZ
D* Type of application that put the message
D MDPAT 273 276I 0 INZ(0)
D* Name of application that put the message
D MDPAN 277 304 INZ
D* Date when message was put
D MDPD 305 312 INZ
D* Time when message was put
D MDPT 313 320 INZ
D* Application data relating to origin
D MDAOD 321 324 INZ
D* Group identifier
D MDGID 325 348 INZ(X'00000000000000-
00000000000000000000-
000000000000')
D* Sequence number of logical message within group
D MDSEQ 349 352I 0 INZ(1)
D* Offset of data in physical message from start of logical message
D MDOFF 353 356I 0 INZ(0)
D* Message flags
D MDMFL 357 360I 0 INZ(0)
D* Length of original message
D MDOLN 361 364I 0 INZ(-1)

```

IBM i での MQMDE (拡張メッセージ記述子)

概要

目的: MQMDE 構造体には、アプリケーション・メッセージ・データの前にあるデータ (存在する場合) を記述します。この構造体には、バージョン 2 の MQMD には存在するが、バージョン 1 の MQMD には存在しない MQMD フィールドがあります。

形式名: FMMDE。

文字セットとエンコード: MQMDE 内のデータは、C プログラミング言語の場合、**CodedCharSetId** キュー・マネージャー属性で指定した文字セットと ENNAT で指定したローカル・キュー・マネージャーのエンコードで記述されていなければなりません。

MQMDE の文字セットおよびエンコードは、以下の *MDCSI* および *MDENC* フィールドで設定されます。

- MQMD (MQMDE 構造体がメッセージ・データの開始点にある場合)
- MQMDE 構造体に先行するヘッダー構造体 (その他のすべての場合)

MQMDE がキュー・マネージャーの文字セットとエンコードにない場合、MQMDE は無効にはなりません
が、参照もされません。つまり、MQMDE はメッセージ・データとして扱われます。

使用法: 通常のアプリケーションでは、バージョン 2 の MQMD を使用してください。この場合、アプリケーションでは MQMDE 構造体は検出されません。しかし、特殊な用途のアプリケーションおよびバージョン 1 の MQMD を引き続き使用するアプリケーションでは、ある状況において MQMDE が検出されます。MQMDE 構造体が存在するのは以下のような場合です。

- MQPUT 呼び出しおよび MQPUT1 呼び出しで指定された場合
- MQGET 呼び出しから戻された場合
- 伝送キューのメッセージ内
- [1160 ページの『MQPUT および MQPUT1 呼び出しで指定された MQMDE』](#)
- [1161 ページの『MQGET 呼び出しで MQMDE が戻される場合』](#)
- [1161 ページの『伝送キューのメッセージ内に MQMDE がある場合』](#)
- [1161 ページの『フィールド』](#)
- [1163 ページの『初期値』](#)
- [1164 ページの『RPG 宣言』](#)

MQPUT および MQPUT1 呼び出しで指定された MQMDE

MQPUT 呼び出しおよび MQPUT1 呼び出しでアプリケーションからバージョン 1 の MQMD を提供する場合には、オプションとして、メッセージ・データに接頭部 MQMDE を付けることができます。その場合は、MQMD の *MDFMT* フィールドに FMMDE を設定して、MQMDE が存在することを示します。アプリケーションが MQMDE を提供しない場合、キュー・マネージャーは MQMDE の各フィールドにデフォルト値が設定されたものと見なします。キュー・マネージャーが使用するデフォルト値は、この構造体の初期値と同じです ([1163 ページの表 750](#) を参照)。

アプリケーションがバージョン 2 の MQMD を提供し、かつアプリケーション・メッセージ・データの前に MQMDE を設定した場合、これらの構造体は [1160 ページの表 749](#) に示すように処理されます。

表 749. MQPUT または MQPUT1 で MQMDE が指定された場合のキュー・マネージャーのアクション			
MQMD Version	バージョン 2 のフィールドの値	MQMDE 内の対応するフィールドの値	キュー・マネージャーによるアクション
1	-	有効	MQMDE を参照する
2	デフォルト	有効	MQMDE を参照する
2	デフォルト値以外	有効	MQMDE をメッセージ・データとして扱う
1 または 2	任意	無効	呼び出しは失敗し、該当する理由コードが戻る
1 または 2	任意	MQMDE が無効な文字セットまたはエンコードで記述されているか、サポートされていないバージョンである	MQMDE をメッセージ・データとして扱う

ただし、特殊な場合が 1 つあります。アプリケーションがバージョン 2 の MQMD を使用してセグメントである (MFSEG フラグまたは MFLSEG フラグが設定された) メッセージを書き込むときに、MQMD での形式名が FMDLH である場合、キュー・マネージャーは MQMDE 構造体を生成して、MQDLH 構造体とそのあと

に続くデータの間 に挿入します。キュー・マネージャーがメッセージと共に保存する MQMD 内のバージョン 2 のフィールドにはデフォルト値が設定されます。

バージョン 2 の MQMD には存在するが、バージョン 1 の MQMD には存在しないフィールドの中には、MQPUT および MQPUT1 で入出力フィールドになるものもあります。ただし、キュー・マネージャーは、MQPUT 呼び出しおよび MQPUT1 呼び出しで出力が生成されても MQMDE の該当するフィールドに値を戻しません。出力値が必要な場合には、そのアプリケーションでバージョン 2 の MQMD を使用する必要があります。

MQGET 呼び出しで MQMDE が戻される場合

MQGET 呼び出しでアプリケーションからバージョン 1 の MQMD を提供した場合は、MQMDE の 1 つ以上のフィールドがデフォルト以外の値の場合に限り、キュー・マネージャーから返されるメッセージの先頭に MQMDE が付けられます。MQMD の *MDFMT* フィールドには、MQMDE が存在することを示す FMMDE という値がキュー・マネージャーにより設定されます。

アプリケーションが **BUFFER** パラメーターの先頭に MQMDE を設定しても、その MQMDE は無視されます。MQGET 呼び出しからの戻り時には、MQMDE はメッセージの MQMDE に置換されている (MQMDE が必要な場合) か、またはアプリケーション・メッセージ・データで上書きされています (MQMDE が不要な場合)。

MQGET 呼び出しから MQMDE が戻される場合、MQMDE のデータは通常、キュー・マネージャーの文字セットおよびエンコードで記述されます。ただし以下のような場合は、MQMDE が別の文字セットおよびエンコードの形になっていることがあります。

- MQMDE が MQPUT 呼び出しまたは MQPUT1 呼び出しのデータとして扱われた (これが生じる状況については、[1160 ページの表 749](#) を参照)。
- TCP 接続で接続されたリモート・キュー・マネージャーからメッセージを受け取ったが、受信側のメッセージ・チャンネル・エージェント (MCA) が正しくセットアップされていなかった (詳しくは、[IBM MQ for IBM i オブジェクトのセキュリティ](#)を参照)。

伝送キューのメッセージ内に MQMDE がある場合

伝送キュー内のメッセージには、接頭部として MQXQH 構造体が付いています。この構造体の中には、バージョン 1 の MQMD が格納されています。伝送キューのメッセージ内には MQMDE も存在する場合があります。存在する場合は、MQXQH 構造体とアプリケーション・メッセージ・データの間にあります。MQMDE が存在するのは、通常、MQMDE のフィールドのうち 1 つ以上にデフォルト値以外の値が設定されている場合に限ります。

また、他の IBM MQ ヘッダー構造体が MQXQH 構造体とアプリケーション・メッセージ・データの間に存在する場合もあります。例えば、送達不能ヘッダー MQDLH があり、メッセージがセグメントでない場合は、次のような順序で配置されます。

- MQXQH (バージョン 1 の MQMD を格納)
- MQMDE
- MQDLH
- アプリケーション・メッセージ・データ

フィールド

MQMDE 構造体には、以下のフィールドが含まれます。フィールドはアルファベット順に説明されています。

MECSI (10 桁の符号付き整数)

MQMDE に続くデータの文字セット ID。

ここでは、MQMDE 構造体の後に続くデータの文字セット ID を指定します。これは、MQMDE 構造体自体の文字データには適用されません。

MQPUT または MQPUT1 呼び出しでは、アプリケーションは、このフィールドをデータに適切な値に設定する必要があります。キュー・マネージャーは、このフィールドが有効かどうかをチェックしません。以下のような特別な値を使用することができます。

CSINHT

この構造体の文字セット ID を継承する。

この構造体の後続くデータの文字データは、この構造体に設定されているのと同じ文字セットになります。

キュー・マネージャーは、メッセージで送信される構造体の中のこの値を、構造体の実際の文字セット ID に変更します。エラーが発生しない限り、値 CSINHT が MQGET 呼び出しによって返されることはありません。

MQMD の MDPAT フィールドの値が ATBRKR の場合、CSINHT は使用できません。

このフィールドの初期値は CSUNDF です。

MEENC (10 桁の符号付き整数)

MEENC (10 桁の符号付き整数)

これは、MQMDE 構造体の後続くデータの数値エンコードを指定します。MQMDE 構造体自体の数値データには適用されません。

MQPUT または MQPUT1 呼び出しでは、アプリケーションは、このフィールドをデータに適切な値に設定する必要があります。キュー・マネージャーは、フィールドが有効かどうかをチェックしません。データ・エンコードの詳細については、[1114 ページの『IBM i での MQMD \(メッセージ記述子\)』](#)の MDENC フィールドを参照してください。

このフィールドの初期値は ENNAT です。

MEFLG (10 桁の符号付き整数)

汎用フラグ。

以下のフラグを指定できます。

MEFNON

フラグなし。

このフィールドの初期値は MEFNON です。

MEFMT (8 バイトの文字ストリング)

MQMDE に続くデータの形式名。

ここでは、MQMDE 構造体の後続くデータの形式名を指定します。

MQPUT または MQPUT1 呼び出しでは、アプリケーションは、このフィールドをデータに適切な値に設定する必要があります。キュー・マネージャーは、このフィールドが有効かどうかをチェックしません。形式名の詳細については、[1114 ページの『IBM i での MQMD \(メッセージ記述子\)』](#)の MDFMT フィールドを参照してください。

このフィールドの初期値は FMNONE です。

MEGID (24 バイトのビット・ストリング)

グループ ID。

[1114 ページの『IBM i での MQMD \(メッセージ記述子\)』](#)の MDGID フィールドを参照してください。このフィールドの初期値は GINONE です。

MELEN (10 桁の符号付き整数)

MQMDE 構造体の長さ。

以下の値が定義されます。

MELEN2

バージョン 2 の拡張メッセージ記述子の構造体の長さ。

このフィールドの初期値は MELEN2 です。

MEMFL (10 桁の符号付き整数)

メッセージ・フラグ。

[1114 ページの『IBM iでの MQMD \(メッセージ記述子\)』の MDMFL フィールド](#)を参照してください。このフィールドの初期値は MFNONE です。

MEOFF (10 桁の符号付き整数)

論理メッセージの先頭を起点とする、物理メッセージ中のデータのオフセット。

[1114 ページの『IBM iでの MQMD \(メッセージ記述子\)』の MDOFF フィールド](#)を参照してください。このフィールドの初期値は 0 です。

MEOLN (10 桁の符号付き整数)

元のメッセージの長さ。

[1114 ページの『IBM iでの MQMD \(メッセージ記述子\)』の MDOLN フィールド](#)を参照してください。このフィールドの初期値は OLUNDF です。

MESEQ (10 桁の符号付き整数)

グループ中の論理メッセージの順序番号。

[1114 ページの『IBM iでの MQMD \(メッセージ記述子\)』の MDSEQ フィールド](#)を参照してください。このフィールドの初期値は 1 です。

MESID (4 バイトの文字ストリング)

構造体 ID

値は次のものでなければなりません。

MESIDV

拡張メッセージ記述子の構造体の ID。

このフィールドの初期値は MESIDV です。

MEVER (10 桁の符号付き整数)

構造体のバージョン番号。

値は次のものでなければなりません。

MEVER2

バージョン 2 の拡張メッセージ記述子の構造体。

以下の定数は、現行バージョンのバージョン番号を指定しています。

MEVERC

拡張メッセージ記述子の構造体の現行バージョン。

このフィールドの初期値は MEVER2 です。

初期値

表 750. MQMDE のフィールド		
フィールド名	定数の名前	定数の値
MESID	MESIDV	'MDE↵'
MEVER	MEVER2	2
MELEN	MELEN2	72
MEENC	ENNAT	環境に依存

表 750. MQMDE のフィールド (続き)

フィールド名	定数の名前	定数の値
MECSI	CSUNDF	0
MEFMT	FMNONE	ブランク
MEFLG	MEFNON	0
MEGID	GINONE	Null
MESEQ	なし	1
MEOFF	なし	0
MEMFL	MFNONE	0
MEOLN	OLUNDF	-1
注:		
1. 記号-は、単一のブランク文字を表します。		

RPG 宣言

```

D*.1.....2.....3.....4.....5.....6.....7..
D*
D* MQMDE Structure
D*
D* Structure identifier
D  MESID             1         4      INZ('MDE ')
D* Structure version number
D  MEVER             5         8I 0 INZ(2)
D* Length of MQMDE structure
D  MELEN             9        12I 0 INZ(72)
D* Numeric encoding of data that followsMQMDE
D  MEENC            13        16I 0 INZ(273)
D* Character-set identifier of data thatfollows MQMDE
D  MECSI            17        20I 0 INZ(0)
D* Format name of data that followsMQMDE
D  MEFMT            21        28      INZ('      ')
D* General flags
D  MEFLG            29        32I 0 INZ(0)
D* Group identifier
D  MEGID            33        56      INZ(X'00000000000000-
D                                     0000000000000000000000-
D                                     000000000000')
D* Sequence number of logical messagewithin group
D  MESEQ            57        60I 0 INZ(1)
D* Offset of data in physical messagefrom start of logical message
D  MEOFF            61        64I 0 INZ(0)
D* Message flags
D  MEMFL            65        68I 0 INZ(0)
D* Length of original message
D  MEOLN            69        72I 0 INZ(-1)

```

IBM i IBM i での MQMHBO (メッセージ・ハンドルからバッファーへの変換オプション)

メッセージ・ハンドルからバッファーへの変換オプションを定義する構造

概要

目的: MQMHBO 構造を使用すると、アプリケーションで、メッセージ・ハンドルからバッファーを作成する方法を制御するオプションを指定できます。この構造体は、MQMHBUF 呼び出しの入力パラメーターです。

文字セットとエンコード: MQMHBO 内のデータは、アプリケーションの文字セットおよびアプリケーションのエンコードでなければなりません (ENNAT)。

- [1165 ページの『フィールド』](#)
- [1165 ページの『初期値』](#)
- [1166 ページの『RPG 宣言』](#)

フィールド

MQMHBO 構造体には、以下のフィールドが含まれます。フィールドはアルファベット順に説明されています。

MBOPT (10 桁の符号付き整数)

メッセージ・ハンドルからバッファへの変換オプション構造体 - MBOPT フィールド。

これらのオプションは、MQMHBUF のアクションを制御します。

以下のオプションを指定しなければなりません。

MBPRRF

プロパティをメッセージ・ハンドルからバッファに変換する際に、MQRFH2 形式に変換します。

オプションで、以下のオプションも指定できます。複数のオプションを指定するには、値を加算する (同じ定数を複数回加算しない) か、ビット単位 OR 演算を使用して値を結合 (プログラミング言語でビット演算がサポートされる場合) します。

MBDLPR

バッファに追加されるプロパティが、メッセージ・ハンドルから削除される。呼び出しが失敗すると、プロパティは削除されません。

これは常に入力フィールドです。このフィールドの初期値は MBPRRF です。

MBSID (10 桁の符号付き整数)

メッセージ・ハンドルからバッファへの変換オプション構造体 - MBSID フィールド。

これは構造体 ID です。値は次のものでなければなりません。

MBSIDV

メッセージ・ハンドルからバッファへの変換オプション構造の ID。

これは常に入力フィールドです。このフィールドの初期値は isMBSIDV です。

MBVER (10 桁の符号付き整数)

これは構造体のバージョン番号です。値は次のものでなければなりません。

MBVER1

メッセージ・ハンドルからバッファへの変換オプション構造のバージョン番号。

以下の定数は、現行バージョンのバージョン番号を指定しています。

MBVERC

メッセージ・ハンドルからバッファへの変換オプション構造の現行バージョン。

これは常に入力フィールドです。このフィールドの初期値は MBVER1 です。

初期値

表 751. MQMHBO のフィールド		
フィールド名	定数の名前	定数の値
MVSID	MBSIDV	'MHBO'
MBVER	MBVER1	1

表 751. MQMHBO のフィールド (続き)		
フィールド名	定数の名前	定数の値
MBOPT	MBPRRF	

注:

1. nul・stringまたは空白の値は、blank文字を表します。

RPG 宣言

```

D* MQMHBO Structure
D*
D*
D* Structure identifier
D MBSID          1      4      INZ('MHBO')
D*
D* Structure version number
D MBVER          5      8I 0 INZ(1)
D*
D* Options that control the action of MQMHBUF
D MBOPT          9      12I 0 INZ(1)

```

IBM i IBM i での MQOD (オブジェクト記述子)

MQOD 構造体は、オブジェクトを名前で指定するために使用されます。

概要

目的: 次のタイプのオブジェクトが有効です。

- キューまたは配布リスト
- 名前リスト
- プロセス定義
- キュー・マネージャー
- トピック

この構造体は、MQOPEN および MQPUT1 呼び出しの入出力パラメーターです。

バージョン: MQOD の現行バージョンは ODVER4 です。これより新しいバージョンの構造体にのみ存在するフィールドについては、そのフィールドの説明にその旨を記載しています。

提供される COPY ファイルには環境によってサポートされている 最新バージョンの MQOD が含まれます。ただし、ODVER フィールドの初期値は ODVER1 に設定されています。version-1 構造体に存在しないフィールドを使用するには、アプリケーションで、ODVER フィールドを必要なバージョンのバージョン番号に設定する必要があります。

配布リストをオープンするには、ODVER が ODVER2 以上でなければなりません。

文字セットとエンコード: MQOD 内のデータは、CodedCharSetId キュー・マネージャー属性で指定された文字セットと、ENNAT で指定されたローカル・キュー・マネージャーのエンコードで記述されていなければなりません。ただし、アプリケーションが IBM MQ クライアントとして実行されている場合、構造体はクライアントの文字セットとエンコードに従っている必要があります。

- [1166 ページの『フィールド』](#)
- [1174 ページの『初期値』](#)
- [1175 ページの『RPG 宣言』](#)

フィールド

MQOD 構造体には、以下のフィールドが含まれます。フィールドはアルファベット順に説明されています。

ODASI (40 バイトのビット・ストリング)

代替セキュリティ ID。

これは、適切な許可検査を実行できるようにするために、*ODAU* と共に許可サービスに渡されるセキュリティ ID です。*ODASI* は、次の場合のみ使用されます。

- MQOPEN 呼び出しで OOALTU が指定されている
- MQPUT1 呼び出しで PMALTU が指定されている

上記のいずれかの場合で、かつ、*ODAU* フィールドが最初のヌル文字またはフィールドの終わりまでの全体がブランクでない場合。

ODASI フィールドは、以下の構造体を持っています。

- 最初のバイトは、後続の有効データの長さを示す 2 進整数です。値には、このバイト自体は含まれません。セキュリティ ID がない場合、長さはゼロになります。
- 2 番目のバイトは、存在するセキュリティ ID のタイプを示します。可能な値は次のとおりです。

SITWNT

Windows セキュリティ ID。

SITNON

セキュリティ ID なし。

- 3 番目のバイトから、最初のバイトで定義された長さまでは、セキュリティ ID 自体が含まれています。
- フィールドの残りのバイトは、2 進ゼロに設定されます。

以下に示す特別な値が使用されることがあります。

SINONE

セキュリティ ID が指定されていない。

値は、フィールドの長さについては 2 進ゼロです。

これは入力フィールドです。このフィールドの長さは LNSCID によって指定されます。このフィールドの初期値は SINONE です。*ODVER* が *ODVER3* より小さい場合は、このフィールドは無視されます。

ODAU (12 バイトの文字ストリング)

代替ユーザー ID。

MQOPEN 呼び出しで OOALTU が指定されている場合、または MQPUT1 呼び出しで PMALTU が指定されている場合、このフィールドには、現在アプリケーションを実行しているユーザー ID の代わりに、オープン権限を検査するために使用される代替ユーザー ID が入ります。ただし、検査によっては、現行のユーザー ID を使って実行されます (例えば、コンテキストの検査など)。

OOALTU および PMALTU が指定されておらず、このフィールドの最初のヌル文字までブランク、またはフィールドの最後まですべてブランクの場合、オープンが正常に行われるのは、オプションを指定してこのオブジェクトをオープンする際にユーザー権限が必要ない場合のみです。

OOALTU と PMALTU がいずれも指定されていない場合、このフィールドは無視されます。

これは入力フィールドです。このフィールドの長さは LNUID によって指定されます。このフィールドの初期値は 12 個のブランク文字です。

ODDN (48 バイトの文字ストリング)

動的キューの名前。

これは、MQOPEN 呼び出しによって作成される動的キューの名前です。これが関係してくるのは、*ODON* にモデル・キューが指定されている場合のみであり、それ以外の場合はすべて *ODDN* は無視されます。

この名前では有効な文字は、*ODON* の場合と同じです。さらに、アスタリスクも有効です。*ODON* にモデル・キューの名前が指定されている場合、ブランクである名前 (または最初のヌル文字の前にブランクのみが指定されている名前) は無効です。

名前の最後の空白でない文字がアスタリスク (*) の場合、キューマネージャはアスタリスクを、ローカルキューマネージャ内で生成されるキュー名が一意であることを保証する文字列に置換します。これを保証できるだけの文字数を確保するためには、アスタリスクの位置がカラム 1 から 33 までの範囲でなければなりません。アスタリスクの後に、ブランクまたはヌル文字以外の文字があってはなりません。

名前がキュー・マネージャによって生成された文字だけで構成される場合は、最初の文字にアスタリスクを指定できます。

これは入力フィールドです。このフィールドの長さは LNQN によって指定されます。このフィールドの初期値は 'AMQ.*' で、ブランクが埋め込まれます。

ODIDC (10 桁の符号付き整数)

オープンに失敗したキューの数。

これは配布リスト中のキューの数で、オープンに失敗したキューの数です。このフィールドは、配布リストにはない 1 つのキューをオープンするときにも設定されます。

注: このフィールドは、MQOPEN 呼び出しまたは MQPUT1 呼び出しの **CMPCOD** パラメーターが CCOK または CCWARN の場合に限り設定されます。**CMPCOD** パラメーターが CCFAIL の場合は、設定されません。

これは出力フィールドです。このフィールドの初期値は 0 です。ODVER が ODVER2 より小さい場合は、このフィールドは無視されます。

ODKDC (10 桁の符号付き整数)

オープンに成功したローカル・キューの数。

これは配布リスト中のキューの数で、ローカル・キューに解決し、オープンに成功したキューの数です。この数にはリモート・キューに解決するキューの数は含まれません。ローカル伝送キューを使用して最初にメッセージを格納する場合でも同様です。このフィールドは、配布リストにはない 1 つのキューをオープンするときにも設定されます。

これは出力フィールドです。このフィールドの初期値は 0 です。ODVER が ODVER2 より小さい場合は、このフィールドは無視されます。

ODMN (48 バイトの文字ストリング)

オブジェクト・キュー・マネージャ名。

これは、ODON オブジェクトが定義されているキュー・マネージャの名前です。この名前で有効な文字は、ODON の場合と同じです (上記を参照)。最初のヌル文字またはフィールドの終わりまで名前をすべてブランクにすると、アプリケーションが接続されているキュー・マネージャ (ローカル・キュー・マネージャ) を指定したと見なされます。

以下に示す点は、記されているオブジェクトのタイプに適用されます。

- ODOT が OTTOP、OTNLST、OTPRO、または OTQM である場合、ODMN はブランクまたはローカル・キュー・マネージャの名前である必要があります。
- ODON にモデル・キューの名前が指定されている場合、キュー・マネージャがモデル・キューの属性を使用して動的キューを作成し、キューを作成したキュー・マネージャの名前を ODMN フィールドに戻します。これはローカル・キュー・マネージャの名前です。モデル・キューは MQOPEN 呼び出しでのみ指定されます。したがって MQPUT1 呼び出しでは無効です。
- ODON がクラスター・キューの名前であり、ODMN がブランクである場合、MQOPEN 呼び出しが戻したキュー・ハンドルを使用して送信されるメッセージの宛先は、次のようにキュー・マネージャによって (または、クラスター・ワークロード出口がインストールされている場合はそれによって) 選択されます。
 - OOBNDG が指定された場合、キュー・マネージャは MQOPEN 呼び出しの処理時にクラスター・キューのインスタンスを選択し、そのキュー・ハンドルを使用して書き込まれるすべてのメッセージは、そのインスタンスへ送信されます。

- OOBNDN が指定された場合、キュー・マネージャーはキュー・ハンドルを使用する連続した MQPUT 呼び出しで、その宛先キューの (クラスター内の別のキュー・マネージャー上にある) 別のインスタンスを選択する場合があります。

アプリケーションからクラスター・キューの特定の インスタンス (つまり、クラスターの特定のキュー・マネージャー上にあるキュー・インスタンス) ヘメッセージを送信する必要がある場合は、アプリケーションで *ODMN* フィールドにそのキュー・マネージャーの名前を指定しなければなりません。これにより、ローカル・キュー・マネージャーは指定された宛先キュー・マネージャーヘメッセージを送信することを強制されます。

- オープン中のオブジェクトが配布リストである場合 (すなわち、*ODREC* がゼロより大きい場合)、*ODMN* は空白またはヌル・ストリングでなければなりません。この条件を満たされないと、この呼び出しは失敗し、理由コード RC2153 が戻ります。

ODON がモデル・キューの名前である場合は、MQOPEN 呼び出しの入出力フィールドです。それ以外の場合は、入力専用フィールドです。このフィールドの長さは LNQMNMN によって指定されます。このフィールドの初期値は 48 個の空白文字です。

ODON (48 バイトの文字ストリング)

オブジェクト名

ODMN で識別されるキュー・マネージャーで定義されているオブジェクトのローカル名。この名前には、以下に示す文字を使用できます。

- 英大文字 (A から Z まで)
- 英小文字 (a から z まで)
- 数字 (0 から 9 まで)
- ピリオド (.), スラッシュ (/), 下線 (_), パーセント (%)

名前に先行空白または組み込み空白を含めることはできませんが、後続の空白を含めることは可能です。ヌル文字を使用して、名前の中における有効なデータの末尾を示すことができます。ヌル文字とそれに続く文字はすべて空白として扱われます。以下に示す制約事項は、それぞれ明記している環境に適用されます。

- EBCDIC カタカナを使用するシステムでは、小文字を使用できません。
- IBM i で英小文字、スラッシュ、パーセントの各文字が含まれている名前をコマンドに指定する場合は、それを引用符で囲む必要があります。構造体内のフィールドまたは呼び出しのパラメーターとして指定する名前には、引用符を使用してはなりません。

以下に示す点は、記されているオブジェクトのタイプに適用されます。

- *ODON* にモデル・キューの名前が指定された場合は、キュー・マネージャーがモデル・キューの属性を使用して動的キューを作成し、作成したキューの名前を *ODON* フィールドに戻します。モデル・キューは MQOPEN 呼び出しでのみ指定されます。したがって MQPUT1 呼び出しでは無効です。
- オープン中のオブジェクトが配布リストである場合 (すなわち、*ODREC* が存在し、ゼロより大きい場合)、*ODON* は空白またはヌル・ストリングでなければなりません。この条件を満たさないと、この呼び出しは失敗し、理由コード RC2152 が戻ります。
- *ODOT* が OTQM である場合には、特別な規則が適用されます。その場合、名前は、最初のヌル文字まですべて空白か、またはフィールドの最後まですべて空白でなければなりません。
- *ODON* が TARGTYPE(TOPIC) を使用する別名キューの名前である場合、指定された別名キューに対して最初にセキュリティ検査が行われますが、これは別名キューの使用法として普通なことです。このセキュリティ検査が正常に行われると、この MQOPEN 呼び出しは続行され、OTTOP の MQOPEN と同じように振る舞います。これには、管理トピック・オブジェクトに対するセキュリティ検査が含まれます。

ODON がモデル・キューの名前である場合は、MQOPEN 呼び出しの入出力フィールドです。それ以外の場合は、入力専用フィールドです。このフィールドの長さは LNQN によって指定されます。このフィールドの初期値は 48 個の空白文字です。

完全トピック名は、*ODON* および *ODOS* の 2 つのフィールドから作成できます。これらの 2 つのフィールドの使用の詳細については、「[トピック文字列の結合](#)」を参照してください。

ODORO (10 桁の符号付き整数)

最初のオブジェクト・レコードの MQOD の先頭からのオフセット。

これは、MQOD 構造体の先頭からの MQOR オブジェクト・レコードのオフセットをバイト数で表したものです。オフセットの値は、正負どちらの値にもなります。*ODORO* は、配布リストがオープン中の場合にのみ使用されます。*ODREC* がゼロの場合、このフィールドは無視されます。

配布リストがオープン中の場合、1 つ以上の MQOR オブジェクト・レコードは、配布リスト中の宛先キューの名前を指定するために提供されなければなりません。これは次の 2 つのうちいずれかの方法で行うことができます。

- *ODORO* オフセット・フィールドを使用する

この場合、アプリケーションは (必要なだけ多くの配列要素のある) MQOR レコードの配列で始まる MQOD を含む、独自の構造体を宣言する必要があります。さらに *ODORO* を、MQOD の先頭からその配列で最初の要素のオフセットに設定する必要があります。このオフセットが正しく設定されるように注意してください。

- *ODORP* ポインター・フィールドを使用する

この場合、アプリケーションは MQOD 構造体とは別個に MQOR 構造体の配列を宣言でき、その配列のアドレスに *ODORP* を設定できます。

どちらの手法を選択しても、*ODORO* または *ODORP* のいずれか一方を使用しなければなりません。両方ともゼロである場合、またはいずれもゼロでない場合は、呼び出しは失敗し、理由コード RC2155 が戻ります。

これは入力フィールドです。このフィールドの初期値は 0 です。*ODVER* が *ODVER2* より小さい場合は、このフィールドは無視されます。

ODORP (ポインター)

最初のオブジェクト・レコードのアドレス。

これは、最初の MQOR オブジェクト・レコードのアドレスです。*ODORP* は、配布リストがオープン中の場合にのみ使用されます。*ODREC* がゼロの場合、このフィールドは無視されます。

これは入力フィールドです。このフィールドの初期値は、ヌル・ポインターです。*ODORP* または *ODORO* のいずれか一方が使用されます。両方とも使用することはできません。詳細については、上述の *ODORO* フィールドを参照してください。*ODORP* を使用しない場合は、ヌル・ポインターまたはヌル・バイトに設定する必要があります。*ODVER* が *ODVER2* より小さい場合は、このフィールドは無視されます。

ODOS (MQCHARV)

ODOS は、使用するロング・オブジェクト名を指定します。

このフィールドは、*ODOT* の特定の値に対してのみ参照されます。このフィールドが使用されることをどの値が示すかは、[ODOT](#) の説明を参照してください。

ODOS が *MQCHARV* 構造体の使用法の説明にあるとおりに正しく指定されていない場合、または最大長を超過した場合は、呼び出しは失敗し、理由コード RC2441 が戻ります。

これは入力フィールドです。この構造体のフィールドの初期値は、*MQCHARV* 構造体のものと同じです。

完全トピック名は、*ODON* および *ODOS* の 2 つのフィールドから作成できます。これらの 2 つのフィールドの使用の詳細については、「[トピック文字列の結合](#)」を参照してください。*ODVER* が *ODVER4* より小さい場合は、このフィールドは無視されます。

ODOT (10 桁の符号付き整数)

オブジェクト・タイプ

ODON で名前が指定されているオブジェクトのタイプ。指定可能な値は以下のとおりです。

OTQ

列。オブジェクトの名前は *ODON* にあります。

OTNLST

名前リスト。オブジェクトの名前は *ODON* にあります。

OTPRO

プロセス定義。オブジェクトの名前は *ODON* にあります。

OTQM

キュー・マネージャー。オブジェクトの名前は *ODON* にあります。

OTTOP

トピック。完全トピック名は、*ODON* および *ODOS* の 2 つのフィールドから作成できます。

これら 2 つのフィールドの使用方法的詳細については、「[トピック文字列の結合](#)」を参照してください。

ODON フィールドによって識別されるオブジェクトが見つからない場合、*ODOS* で指定されたストリングが存在する場合でも、呼び出しは失敗し、理由コード RC2425 が戻ります。

これは常に入力フィールドです。このフィールドの初期値は OTQ です。

ODREC (10 桁の符号付き整数)

存在するオブジェクト・レコードの数。

これは、アプリケーションが提供した MQOR オブジェクト・レコードの数です。この数がゼロより大きい場合は配布リストがオープンされており、*ODREC* がリスト内の宛先キューの数になっていることを示しています。配布リストに宛先が 1 つしかない場合は有効です。

ODREC の値はゼロ未満であってはなりません。また、この値がゼロより大きい場合、*ODOT* は OTQ でなければなりません。これらの条件を満たさないと、その呼び出しは失敗し、理由コード RC2154 が戻ります。

これは入力フィールドです。このフィールドの初期値は 0 です。*ODVER* が *ODVER2* より小さい場合は、このフィールドは無視されます。

ODRMN (48 バイトの文字ストリング)

解決済みのキュー・マネージャーの名前。

これは、ローカル・キュー・マネージャーが名前の解決を実行した後の宛先キュー・マネージャーの名前です。戻される名前は、*ODRQN* によって識別されるキューを所有するキュー・マネージャーの名前です。*ODRMN* は、ローカル・キュー・マネージャーの名前にすることができます。

ODRQN が、ローカル・キュー・マネージャーが属するキュー共有グループが所有する共有キューである場合、*ODRMN* はそのキュー共有グループの名前です。キューが他のキュー共有グループによって所有されている場合、*ODRQN* は、キュー共有グループの名前またはキュー共有グループのメンバーであるキュー・マネージャーの名前にすることができます (返される値の性質は、ローカル・キュー・マネージャーに存在するキュー定義によって異なります)。

非ブランク値は、オブジェクトがブラウズ、入力、または出力 (あるいはこれらの組み合わせ) を目的としてオープンされた単一のキューである場合にだけ戻されます。オープンされたオブジェクトが以下のいずれかである場合、*ODRMN* はブランクに設定されます。

- キューでない
- キューだが、オープンの目的がブラウズ、入力、および出力のいずれでもない
- OOBNDN が指定されたクラスター・キュー (**DefBind** キュー属性の値が BNDNOT のときは OOBNDQ が有効なキュー)
- 配布リスト

これは出力フィールドです。このフィールドの長さは LNQN によって指定されます。このフィールドの初期値は、C 言語ではヌル・ストリングであり、他のプログラミング言語では 48 桁の空白文字です。ODVER が ODVER3 より小さい場合は、このフィールドは無視されます。

ODRO (MQCHARV)

ODRO は、キュー・マネージャーが ODon に指定された名前を解決した後のロング・オブジェクト名です。

このフィールドは、トピック・オブジェクトを参照する特定のタイプのオブジェクト、トピックおよびキュー別名の場合にのみ戻されます。

ロング・オブジェクト名が ODOS に指定されており、ODON には何も指定されていない場合、このフィールドに戻される値は、ODOS で指定されている名前と同じです。

このフィールドが省略されている (つまり ODRV.VSBufSize がゼロである) 場合、ODRO は戻されませんが、長さが ODRV.VSLength に戻されます。長さが全体の ODRV よりも短い場合、これは切り捨てられ、指定された長さに入る限り右端の文字が最大限戻されます。

ODRO が MQCHARV 構造体の使用法の説明にあるとおりに正しく指定されていない場合、または最大長を超過した場合は、呼び出しは失敗し、理由コード RC2520 が戻ります。ODVER が ODVER4 より小さい場合は、このフィールドは無視されます。

ODRQN (48 バイトの文字ストリング)

解決済みのキューの名前。

これは、ローカル・キューが名前の解決を実行した後の宛先キュー・マネージャーの名前です。戻される名前は、ODRMN によって識別されるキュー・マネージャー上に存在するキューの名前です。

非空白値は、オブジェクトがブラウズ、入力、または出力 (あるいはこれらの組み合わせ) を目的としてオープンされた単一のキューである場合にだけ戻されます。オープンされたオブジェクトが以下のいずれかである場合、ODRQN は空白に設定されます。

- キューでない
- キューだが、オープンの目的がブラウズ、入力、および出力のいずれでもない
- 配布リスト
- トピック・オブジェクトを参照する別名キュー (代わりに、[1172 ページの『ODRO \(MQCHARV\)』](#)を参照)

これは出力フィールドです。このフィールドの長さは LNQN によって指定されます。このフィールドの初期値は、C 言語ではヌル・ストリングであり、他のプログラミング言語では 48 桁の空白文字です。ODVER が ODVER3 より小さい場合は、このフィールドは無視されます。

ODRRO (10 桁の符号付き整数)

MQOD の先頭から最初の応答レコードのオフセット。

これは、MQOD 構造体の先頭から最初の MQRR 応答レコードのオフセットをバイト数で表したものです。オフセットの値は、正負どちらの値にもなります。ODRRO は、配布リストがオープン中の場合にのみ使用されます。ODREC がゼロの場合、このフィールドは無視されます。

配布リストがオープン中の場合、1 つ以上の MQRR 応答レコードの配列が提供されます。これは、オープンに失敗したキューを判別するため (この場合、MQRR 内の RRCC フィールドに入ります)、および失敗した理由をそれぞれ判別するためです (この場合、MQRR 内の RRREA フィールドに入ります)。データは、応答レコードの配列に、キューの名前がオブジェクト・レコードの配列に発生したのと同じ順番で戻ります。キュー・マネージャーは、呼び出しの結果が混在したときのみ応答レコードを設定します。つまり、オープンに成功したキューもあれば失敗したキューもある場合、または全部失敗したが理由が異なる場合などです。呼び出しから理由コード RC2136 が戻るのはこの場合です。すべてのキューに同じ理由コードが該当する場合は、その理由コードが MQOPEN または MQPUT1 呼び出しの REASON パラメーター内に戻され、応答レコードは設定されません。応答レコードはオプションですが、指定する場合はこれらの ODREC が必要です。

応答レコードは、*ODRRO* にオフセットを指定するか、*ODRRP* にアドレスを指定することにより、オブジェクト・レコードと同様に提供されます。この方法の詳細については、上述の *ODORO* を参照してください。ただし、*ODRRO* および *ODRRP* の両方を使用することはできません。両方ともゼロでない場合、呼び出しは失敗し、理由コード RC2156 が戻ります。

MQPUT1 呼び出しの場合、これらの応答レコードはエラーについての情報を返すのに使用されます。このエラーは、キューがオープンされる場合や、メッセージが配布リスト中のキューに送られる場合に発生します。あるキューに対するオープン操作から出る完了コードおよび理由コードは、そのキューに対する書き込み操作から出るコードに置き換えられます。ただし、これは後者から戻った完了コードが CCOK または CCWARN であった場合に限られます。

これは入力フィールドです。このフィールドの初期値は 0 です。*ODVER* が *ODVER2* より小さい場合は、このフィールドは無視されます。

ODRRP (ポインター)

最初の応答レコードのアドレス。

これは、最初の MQRR 応答レコードのアドレスです。*ODRRP* は、配布リストがオープン中の場合にのみ使用されます。*ODREC* がゼロの場合、このフィールドは無視されます。

応答レコードの指定には、*ODRRP* または *ODRRO* のいずれか一方を使用できます。両方とも使用することはできません。詳しくは、上述の *ODRRO* フィールドの説明を参照してください。*ODRRP* を使用しない場合は、ヌル・ポインターまたはヌル・バイトに設定する必要があります。

これは入力フィールドです。このフィールドの初期値は、ヌル・ポインターです。*ODVER* が *ODVER2* より小さい場合は、このフィールドは無視されます。

ODSID (4 バイトの文字ストリング)

構造体 ID

値は次のものでなければなりません。

ODSIDV

オブジェクト記述子構造体の ID。

これは常に入力フィールドです。このフィールドの初期値は *ODSIDV* です。

ODSS (MQCHARV)

ODSS には、キューからメッセージを取り出す時に使用する選択基準を指定するために使用されるストリングが含まれています。

以下の場合には、*ODSS* は指定しないでください。

- *ODOT* が OTQ ではない場合。
- オープンされたキューが、OOINP* 入力オプションの 1 つを使用してオープンされていない場合。

これらの場合に *ODSS* が指定されると、呼び出しは失敗し、理由コード RC2516 が戻ります。

ODSS が *MQCHARV* 構造体の使用法の説明にあるとおりに正しく指定されていない場合、または最大長を超過した場合は、呼び出しは失敗し、理由コード RC2519 が戻ります。*ODVER* が *ODVER4* より小さい場合は、このフィールドは無視されます。

ODUDC (10 桁の符号付き整数)

オープンに成功したリモート・キューの数

これは配布リスト中のキューの数で、リモート・キューに解決し、オープンに成功したキューの数です。このフィールドは、配布リストにはない 1 つのキューをオープンするときにも設定されます。

これは出力フィールドです。このフィールドの初期値は 0 です。*ODVER* が *ODVER2* より小さい場合は、このフィールドは無視されます。

ODVER (10 桁の符号付き整数)

構造体のバージョン番号。

値は次のいずれかでなければなりません。

ODVER1

バージョン 1 のオブジェクト記述子構造体。

ODVER2

バージョン 2 のオブジェクト記述子構造体。

ODVER3

バージョン 3 のオブジェクト記述子構造体。

ODVER4

バージョン 4 のオブジェクト記述子構造体。

これより新しいバージョンの構造体にもみ存在するフィールドは、そのフィールドの説明にその旨記載されています。以下の定数は、現行バージョンのバージョン番号を指定しています。

ODVERC

現行バージョンのオブジェクト記述子構造体。

これは常に入力フィールドです。このフィールドの初期値は ODVER1 です。

初期値

表 752. MQOD のフィールド		
フィールド名	定数の名前	定数の値
ODSID	ODSIDV	'OD- -'
ODVER	ODVER1	1
ODOT	OTQ	1
ODON	なし	ブランク
ODMN	なし	ブランク
ODDN	なし	'AMQ.*'
ODAU	なし	ブランク
ODREC	なし	0
ODKDC	なし	0
ODUDC	なし	0
ODIDC	なし	0
ODORO	なし	0
ODRRRO	なし	0
ODORP	なし	ヌル・ポインターまたはヌル・バイト
ODRRP	なし	ヌル・ポインターまたはヌル・バイト
ODASI	SINONE	Null
ODRQN	なし	ブランク
ODRMN	なし	ブランク
ODOS	MQCHARV で定義されているとおり	MQCHARV で定義されているとおり
ODRO	ODOS と同じ	ODOS と同じ

表 752. MQOD のフィールド (続き)

フィールド名	定数の名前	定数の値
ODSS	なし	ブランク
注:		
1. 記号-は、単一のブランク文字を表します。		

RPG 宣言

```

D*..1.....2.....3.....4.....5.....6.....7..
D*
D* MQOD Structure
D*
D*
D* Structure identifier
D  ODSID              1          4    INZ('OD ')
D*
D* Structure version number
D  ODVER              5          8I 0 INZ(1)
D*
D* Object type
D  ODOT              9         12I 0 INZ(1)
D*
D* Object name
D  ODON             13         60    INZ
D*
D* Object queue manager name
D  ODMN             61        108    INZ
D*
D* Dynamic queue name
D  ODDN            109        156    INZ('AMQ.*')
D*
D* Alternate user identifier
D  ODAU            157        168    INZ
D*
** Number of object records
D* present
D  ODREC            169       172I 0 INZ(0)
D*
** Number of local queues opened
D* successfully
D  ODKDC            173       176I 0 INZ(0)
D*
** Number of remote queues opened
D* successfully
D  ODUDC            177       180I 0 INZ(0)
D*
** Number of queues that failed to
D* open
D  ODIDC            181       184I 0 INZ(0)
D*
** Offset of first object record
D* from start of MQOD
D  ODORO            185       188I 0 INZ(0)
D*
** Offset of first response record
D* from start of MQOD
D  ODRRO            189       192I 0 INZ(0)
D*
D* Address of first object record
D  ODORP            193       208*   INZ(*NULL)
D*
** Address of first response
D* record
D  ODRRP            209       224*   INZ(*NULL)
D*
D* Alternate security identifier
D  ODASI            225       264    INZ(X'0000000000000000-
D                                000000000000000000000000-
D                                000000000000000000000000-
D                                000000000000')
D*
D* Resolved queue name
D  ODRQN            265       312    INZ

```

```

D*
D* Resolved queue manager name
D  ODRMN          313      360      INZ
D*
D* reserved field
D  ODRE1          361      364I 0 INZ(0)
D*
D* reserved field
D  ODRS2          365      368I 0 INZ(0)
D*
D* Object long name
D* Address of variable length string
D  ODOSCHRP       369      384*      INZ(*NULL)
D* Offset of variable length string
D  ODOSCHRO       385      388I 0 INZ(0)
D* Size of buffer
D  ODOSVSBS       389      392I 0 INZ(-1)
D* Length of variable length string
D  ODOSCHRL       393      396I 0 INZ(0)
D* CCSID of variable length string
D  ODOSCHRC       397      400I 0 INZ(-3)
D*
D* Message Selector
D* Address of variable length string
D  ODSSCHRP       401      416*      INZ(*NULL)
D* Offset of variable length string
D  ODSSCHRO       417      420I 0 INZ(0)
D* Size of buffer
D  ODSSVSBS       421      424I 0 INZ(-1)
D* Length of variable length string
D  ODSSCHRL       425      428I 0 INZ(0)
D* CCSID of variable length string
D  ODSSCHRC       429      432I 0 INZ(-3)
D*
D* Resolved long object name
D* Address of variable length string
D  ODRSOCHRP      433      448*      INZ(*NULL)
D* Offset of variable length string
D  ODRSOCHRO      449      452I 0 INZ(0)
D* Size of buffer
D  ODRSOVSBS      453      456I 0 INZ(-1)
D* Length of variable length string
D  ODRSOCHRL      457      460I 0 INZ(0)
D* CCSID of variable length string
D  ODRSOCHRC      461      464I 0 INZ(-3)
D*
D* Alias queue resolved object type
D  ODRT           465      468I 0 INZ(0)

```

IBM i での MQOR (オブジェクト・レコード)

MQOR 構造体は、単一宛先キューのキュー名およびキュー・マネージャーの名前を指定するのに使われます。

概要

目的: MQOR は、MQOPEN 呼び出しおよび MQPUT1 呼び出しのための入力構造体です。

文字セットとエンコード: MQOR 内のデータは、**CodedCharSetId** キュー・マネージャー属性で指定された文字セットと、ENNAT で指定されたローカル・キュー・マネージャーのエンコードで記述されていなければなりません。ただし、アプリケーションが IBM MQ クライアントとして実行されている場合、構造体はクライアントの文字セットとエンコードに従っている必要があります。

使用法: MQOPEN 呼び出しでこれらの構造体の配列を提供することによって、キューのリストをオープンすることができます。このリストを配布リストと呼びます。各メッセージの書き込みは、そのキューが正常にオープンした場合、その MQOPEN 呼び出しによって戻されたキュー・ハンドルを使用して、リスト中の各キューに置かれます。

- [1177 ページの『フィールド』](#)
- [1177 ページの『初期値』](#)
- [1177 ページの『RPG 宣言』](#)

フィールド

MQOR 構造体には、以下のフィールドが含まれます。フィールドはアルファベット順に説明されています。

ORMN (48 バイトの文字ストリング)

オブジェクト・キュー・マネージャー名。

これは、MQOD 構造体の ODMN フィールドと同じです (詳細については、MQOD を参照してください)。

これは常に入力フィールドです。このフィールドの初期値は 48 個のブランク文字です。

ORON (48 バイトの文字ストリング)

オブジェクト名

これは、以下の点を除き、MQOD 構造体の ODON フィールドと同じです (詳細は MQOD を参照)。

- キューの名前でなければならない。
- モデル・キューの名前であってはならない。

これは常に入力フィールドです。このフィールドの初期値は 48 個のブランク文字です。

初期値

表 753. MQOR のフィールド		
フィールド名	定数の名前	定数の値
ORON	なし	ブランク
ORMN	なし	ブランク

RPG 宣言

```
D*..1.....2.....3.....4.....5.....6.....7..
D*
D* MQOR Structure
D*
D* Object name
D  ORON                1      48    INZ
D* Object queue manager name
D  ORMN                49     96    INZ
```

MQPD - プロパティ記述子

MQPD は、プロパティの属性を定義するために使用されます。

概要

目的: この構造体は、MQSETMP 呼び出しの入出力パラメーターであり、MQINQMP 呼び出しの出力パラメーターです。

文字セットとエンコード: MQPD 内のデータは、アプリケーションの文字セットおよびアプリケーションのエンコードでなければなりません (ENNAT)。

- [1177 ページの『フィールド』](#)
- [1180 ページの『初期値』](#)
- [1180 ページの『RPG 宣言』](#)

フィールド

MQPD 構造体には、以下のフィールドが含まれます。フィールドはアルファベット順に説明されています。

PDCT (10 桁の符号付き整数)

ここでは、プロパティが属しているメッセージ・コンテキストについて説明します。

キューマネージャが、自身が不正と認識する-定義 IBM MQ プロパティを含むメッセージを受信した場合。キュー管理者はフィールド *PDCT* の値を修正する。

次のようなオプションを指定できます。

PDUSC

プロパティは *user* コンテキストに関連付けられます。

MQSETMP 呼び出しを使用してユーザー・コンテキストに関連付けたプロパティを設定するのに、特別な権限は必要ありません。

上記で説明されたオプションが必要ない場合、以下のオプションを使用できます。

PDNOC

プロパティはメッセージ・コンテキストに関連付けられません。

認識されない値は、*PDREA* コード RC2482 で拒否されます。

これは、MQSETMP 呼び出しの入出力フィールドおよび MQINQMP 呼び出しからの出力フィールドです。このフィールドの初期値は PDNOC です。

PDCPYOPT (10 桁の符号付き整数)

これは、プロパティのコピー先となるメッセージ・タイプについて説明します。

これは、認識される IBM MQ 定義のプロパティの出力専用フィールドです。IBM MQ が適切な値を設定します。

キューマネージャが、自身が不正と認識する-定義 IBM MQ プロパティを含むメッセージを受信した場合。キュー管理者はフィールド *CopyOptions* の値を修正する。

以下に説明する 1 つ以上のオプションを指定できます。複数のオプションを指定するには、値を加算する (同じ定数を複数回加算しない) か、ビット単位 OR 演算を使用して値を結合 (プログラミング言語でビット演算がサポートされる場合) します。

COPFOR

このプロパティは、転送されるメッセージにコピーされます。

COPPUB

このプロパティは、メッセージのパブリッシュ中にサブスクライバーが受信したメッセージにコピーされます。

COPREP

このプロパティは応答メッセージにコピーされます。

COPRP

このプロパティはレポート・メッセージにコピーされます。

COPALL

このプロパティはすべてのタイプの後続メッセージにコピーされます。

COPNON

このプロパティは、メッセージにはコピーされません。

デフォルト・オプション: コピー・オプションのデフォルト・セットを提供するには、以下のオプションを指定できます。

COPDEF

このプロパティは、転送中のメッセージ、レポート・メッセージ、またはメッセージのパブリッシュ中にサブスクライバーが受信したメッセージにコピーされます。

これは、オプション COPFOR、COPRP、および COPPUB を組み合わせて指定するのと同様です。

上記のオプションのいずれも必要ではない場合、以下のオプションを使用します。

COPNON

この値は、他のコピー・オプションが指定されなかったことを示すために使用します。このプロパティと後続のメッセージの間にプログラマチックな関連付けは行われません。これは、メッセージ記述子プロパティの場合は常に返されます。

これは、MQSETMP 呼び出しの入出力フィールドおよび MQINQMP 呼び出しからの出力フィールドです。このフィールドの初期値は COPDEF です。

PDOPT (10 桁の符号付き整数)

値は次のものでなければなりません。

PDNONE

指定されるオプションはありません。

これは常に入力フィールドです。このフィールドの初期値は PDNONE です。

PDSID (10 桁の符号付き整数)

これは構造体 ID です。値は以下のものでなければなりません。

PSIDV

プロパティ記述子構造体の ID。

これは常に入力フィールドです。このフィールドの初期値は **PSIDV** です。

PDSUP (10 桁の符号付き整数)

このフィールドは、メッセージ・プロパティが含まれるメッセージをキューに書き込むために、キュー・マネージャーでこのプロパティについてどのレベルのサポートが必要かを記述します。これは、IBM MQ 定義のプロパティにのみ適用され、他のすべてのプロパティに対するサポートはオプションです。

このフィールドは、IBM MQ 定義のプロパティがキュー・マネージャーにより認知された時点で、正しい値に自動的に設定されます。プロパティが認識されない場合、PDSUPO が割り当てられます。キューマネージャが、自身が不正と認識する-定義 IBM MQ プロパティを含むメッセージを受信した場合、キュー管理者はフィールド *PDSUP* の値を修正する。

CMNOVA オプションが設定されたメッセージ・ハンドルで MQSETMP 呼び出しを使用して IBM MQ 定義のプロパティを設定する場合、*PDSUP* は入力フィールドになります。これにより、アプリケーションは、接続しているキュー・マネージャーではサポートされない IBM MQ 定義のプロパティに正しい値を設定する (メッセージの処理は別のキュー・マネージャーで行う) ことができます。

IBM MQ 定義のプロパティではないプロパティには、常に値 PDSUPO が割り当てられます。

CMNOVA オプションが設定されているメッセージ・ハンドルで MQSETMP 呼び出しを使用すると、以下に示す値のいずれかが MQINQMP 呼び出しによって戻されるか、値のいずれかを指定できます。

PDSUPO

プロパティはサポートされていなくても、キュー・マネージャーに受け入れられます。メッセージ・プロパティをサポートしていないキュー・マネージャーにメッセージをフローするために、このプロパティは破棄される場合があります。この値は、IBM MQ 定義ではないプロパティにも割り当てられます。

PDSUPR

プロパティに対するサポートは必須です。メッセージは、IBM MQ 定義のプロパティをサポートしないキュー・マネージャーによりリジェクトされます。MQPUT 呼び出しまたは MQPUT1 呼び出しは、完了コード CCFAIL および理由コード RC2490 で失敗します。

PDSUPL

メッセージの宛先がローカル・キューになっている場合、メッセージは、IBM MQ 定義のプロパティをサポートしないキュー・マネージャーによりリジェクトされます。MQPUT 呼び出しまたは MQPUT1 呼び出しは、完了コード CCFAIL および理由コード RC2490 で失敗します。

メッセージの宛先がリモート・キュー・マネージャーである場合、MQPUT 呼び出しまたは MQPUT1 呼び出しは成功します。

これは、メッセージ・ハンドルの作成時に CMNOVA オプションが設定されている場合、MQINQMP 呼び出しの出力フィールドであり、MQSETMP 呼び出しの入力フィールドです。このフィールドの初期値は PDSUPO です。

PDVER (10 桁の符号付き整数)

これは構造体のバージョン番号です。値は以下のものでなければなりません。

PDVER1

バージョン 1 のプロパティ記述子構造体。

以下の定数は、現行バージョンのバージョン番号を指定しています。

PDVERC

プロパティ記述子構造体の現行バージョン。

これは常に入力フィールドです。このフィールドの初期値は **PDVER1** です。

初期値

表 754. MQPD のフィールド		
フィールド名	定数の名前	定数の値
PDSID	PDSIDV	'PD'
PDVER	PDVER1	1
PDOPT	PDNONE	0
PDSUP	PDSUPO	0
PDCT	PDNOC	0
PDCPYOPT	COPDEF	0

RPG 宣言

```

D* MQDMHO Structure
D*
D*
D* Structure identifier
D  DMSID              1      4      INZ('DMHO')
D*
D* Structure version number
D  DMVER              5      8I 0 INZ(1)
D*
D* Options that control the action of MQDLTMH
D  DMOPT              9      12I 0 INZ(0)

```

IBM i IBM i での MQPMO (メッセージ書き込みオプション)

MQPMO 構造体により、アプリケーションは、メッセージがキューに配置される方法およびトピックに公開される方法を制御するオプションを指定できます。

概要

目的

この構造体は、MQPUT および MQPUT1 呼び出しの入出力パラメーターです。

バージョン

MQPMO の現行バージョンは PMVER2 です。これより新しいバージョンの構造体にのみ存在するフィールドについては、そのフィールドの説明にその旨を記載しています。

提供されている COPY ファイルには、環境でサポートされている最新バージョンの MQPMO が含まれていますが、*PMVER* フィールドの初期値は *PMVER1* に設定されています。 *version-1* 構造体に存在しないフィールドを使用するには、アプリケーションで、*PMVER* フィールドを必要なバージョンのバージョン番号に設定する必要があります。

文字セットとエンコード

MQPMO 内のデータは、**CodedCharSetId** キュー・マネージャー属性で指定された文字セットと **ENNAT** で指定されたローカル・キュー・マネージャーのエンコードで記述されていなければなりません。ただし、アプリケーションが IBM MQ クライアントとして実行されている場合、構造体はクライアントの文字セットとエンコードに従っている必要があります。

- [1181 ページの『フィールド』](#)
- [1195 ページの『初期値』](#)
- [1195 ページの『RPG 宣言』](#)

フィールド

MQPMO 構造体には、以下のフィールドが含まれます。フィールドはアルファベット順に説明されています。

PMCT (10 桁の符号付き整数)

入力キューのオブジェクト・ハンドル。

PMPASI または PMPASA が指定されている場合、このフィールドに入るのは、書き込まれるメッセージに関連付けるコンテキスト情報が取られる元の入力キュー・ハンドルです。

PMPASI と PMPASA が指定されていない場合、このフィールドは無視されます。

これは入力フィールドです。このフィールドの初期値は 0 です。

PMIDC (10 桁の符号付き整数)

送信できなかったメッセージの数。

これは配布リスト中のキューに送信できなかったメッセージの数です。この数にはオープンに失敗したキューの数、およびオープンには成功したが PUT 操作に失敗したキューの数も含まれます。このフィールドは、配布リストにはない単一のキューにメッセージを書き込むときも設定されます。

注: このフィールドは、MQPUT または MQPUT1 呼び出しの **CMPCOD** パラメーターが CCOK または CCWARN の場合にのみ設定されます。**CMPCOD** パラメーターが CCFAIL の場合には設定されません。

これは出力フィールドです。このフィールドの初期値は 0 です。*PMVER* が *PMVER2* より小さい場合、このフィールドは設定されません。

PMKDC (10 桁の符号付き整数)

ローカル・キューへの送信が成功したメッセージの数。

これは、現在の MQPUT 呼び出しまたは MQPUT1 呼び出しがローカル・キューである配布リスト中のキューへの送信に成功したメッセージの数です。この数にはリモート・キューを解決するキューへ送信されたメッセージの数は含まれません。ローカル伝送キューを使用して最初にメッセージを格納する場合でも同様です。このフィールドは、配布リストにはない単一のキューにメッセージを書き込むときも設定されます。

これは出力フィールドです。このフィールドの初期値は 0 です。*PMVER* が *PMVER2* より小さい場合、このフィールドは設定されません。

PMOPT (10 桁の符号付き整数)

MQPUT および MQPUT1 のアクションを制御するオプション。

以下のいずれかを指定しても、または何も指定しなくても構いません。2 つ以上指定が必要な場合は、それらの値を加算します (同じ定数を複数回加算しないでください)。有効でない組み合わせについては、注記されています。それ以外の組み合わせは有効です。

パブリッシュ・オプション: 以下のオプションは、メッセージをトピックにパブリッシュする方法を制御します。

PMSRTO

このパブリケーションの MQMD の MDRQ および MDRM フィールドに入力される情報はサブスクライバーに渡されません。このオプションが、ReplyToQ を必要とするレポート・オプションと同時に使用されると呼び出しは失敗し、RC2027 が戻されます。

PMRET

送信されたパブリケーションがキュー・マネージャーによって保存されます。これにより、サブスクライバーはこのパブリケーションが公開された後、MQSUBRQ 呼び出しを使用することにより、そのコピーを要求することができます。さらに、このパブリケーションが作成された後に、パブリケーションをそのサブスクリプションを行うアプリケーションに送信する (オプション SONEWP を使用して送信しないように選択した場合を除く) こともできます。保存されたパブリケーションがアプリケーションに送られると、そのパブリケーションの mq.IsRetained メッセージ・プロパティによって示されます。

トピック・ツリーの各ノードに保存できるパブリケーションは1つだけです。つまり、他のすべてのアプリケーションによってパブリッシュされた、このトピック用の保存パブリケーションが既に存在する場合、このパブリケーションが置き換えてしまいます。そのため、同じトピックに関するメッセージを保存するパブリッシャーを複数持つことは避けたほうがよいでしょう。

保存パブリケーションがサブスクライバーによって要求される場合、使用されるサブスクリプションのトピックにワイルドカードが含まれていることがあります。その場合、(トピック・ツリーのさまざまなノードの) いくつかの保存パブリケーションがマッチングする可能性があり、複数のパブリケーションが要求側のアプリケーションに送られる場合があります。詳細については、[788 ページの『MQSUBRQ - サブスクリプション要求』](#) 呼び出しの説明を参照してください。

このオプションが使用され、パブリケーションを保存できない場合、メッセージは公開されずに呼び出しが失敗し、RC2479 が戻されます。

同期点オプション: 以下のオプションは、作業単位内での MQPUT または MQPUT1 呼び出しの実行に関連したオプションです。

PMSYP

同期点制御を持つ書き込みメッセージ。

この要求は、通常の作業単位プロトコルの中で操作することです。メッセージは、作業単位がコミットされるまで、作業単位の外側には表示されません。作業単位がバックアウトされると、メッセージは除去されます。

このオプションおよび PMNSYP が指定されていない場合、書き込み要求は作業単位内にありません。

PMSYP と PMNSYP を同時に指定しないでください。

PMNSYP

同期点制御を持たない書き込みメッセージ。

この要求は、通常の作業単位プロトコルの外部で動作することになります。メッセージは即時に使用可能になり、作業単位をバックアウトしても削除できません。

このオプションおよび PMSYP が指定されていない場合、書き込み要求は作業単位内にありません。

PMNSYP と PMSYP を同時に指定しないでください。

メッセージ ID と相関 ID のオプション: 次のオプションは、新しいメッセージ ID または相関 ID を生成することをキュー・マネージャーに要求します。

PMNMID

新しいメッセージ ID を生成します。

このオプションを指定すると、キュー・マネージャーは MQMD の MDMID フィールドの内容を新しいメッセージ ID に置き換えます。このメッセージ ID はメッセージと共に送信され、MQPUT 呼び出しまたは MQPUT1 呼び出しからの出力時にアプリケーションに戻ります。

このオプションは、メッセージが配布リストに書き込まれるときにも指定できます。詳細については、MQPMR 構造体の *PRMID* フィールドの説明を参照してください。

このオプションを使用すると、各 MQPUT または MQPUT1 呼び出しの前に *MDMID* フィールドを MINONE にリセットする必要がなくなります。

PMNCID

新しい相関 ID を生成します。

このオプションを指定すると、キュー・マネージャーは MQMD の *MDCID* フィールドの内容を新しい相関 ID に置き換えます。この相関 ID はメッセージと共に送信され、MQPUT 呼び出しまたは MQPUT1 呼び出しからの出力時にアプリケーションに戻ります。

このオプションは、メッセージが配布リストに書き込まれるときにも指定できます。詳細については、MQPMR 構造体の *PRCID* フィールドの説明を参照してください。

PMNCID は、アプリケーションに固有の相関 ID が必要な状況で役に立ちます。

グループおよびセグメントのオプション: 以下は、論理メッセージのグループおよびセグメント内のメッセージの処理に関するオプションです。これらの定義を理解しておくと、オプションを把握するのに役に立ちます。

物理メッセージ

このメッセージは、キューに入れたりキューから除去できる最小単位の情報です。多くの場合、1 つの MQPUT、MQPUT1、または MQGET 呼び出しで指定された情報や取り出された情報に相当します。すべての物理メッセージには、固有のメッセージ記述子 (MQMD) があります。通常、物理メッセージは、メッセージ ID (MQMD の *MDMID* フィールド) の異なる値によって区別されます。ただし、これはキュー・マネージャーによって強制されるものではありません。

論理メッセージ

これは、1 単位のアプリケーション情報です。システムに制約がない場合には、1 つの論理メッセージが 1 つの物理メッセージになることもあります。ただし、論理メッセージが大きい場合、システムの制約により、1 つの論理メッセージをセグメントと呼ばれる複数の物理メッセージに分割することが必要になる場合があります。

セグメント化された論理メッセージは、同じ非ヌル・グループ ID (MQMD の *MDGID* フィールド) を持つ複数の物理メッセージと、同じメッセージ・シーケンス番号 (MQMD の *MDSEQ* フィールド) で構成されます。セグメントは、セグメント・オフセット (MQMD の *MDOFF* フィールド) の固有の値によって区別されます。この値は、論理メッセージ内のデータの先頭からの物理メッセージ内のデータのオフセットを示します。各セグメントは 1 つの物理メッセージなので、論理メッセージ内のセグメントには通常、それぞれ固有のメッセージ ID があります。

セグメント化されていない論理メッセージにも、送信側のアプリケーションでセグメント化が許可されている場合は、NULL でないグループ ID があります。ただし、この場合、そのグループ ID を持つのは、論理メッセージが 1 つのメッセージ・グループに属していないと、1 つの物理メッセージのみです。論理メッセージが 1 つのメッセージ・グループに属していない場合、送信側のアプリケーションによってセグメント化が禁止されている論理メッセージのグループ ID はヌルとなります (GINONE)。

メッセージ・グループ

非空文字の同じグループ ID をもつ 1 つ以上の論理メッセージから構成される集合です。グループ内のそれぞれの論理メッセージは、メッセージ順序番号に指定された固有の値で区別されます。指定される値は 1 から n までの整数で、n はグループ内の論理メッセージの数です。1 つ以上の論理メッセージをセグメント化すると、グループ内の物理メッセージの数は n 個を超えます。

PMLOGO

グループ内のメッセージおよび論理メッセージのセグメントが、論理順序で書き込まれます。

このオプションは、キュー・マネージャーに、アプリケーションがグループ内のメッセージと論理メッセージのセグメントを書き込む方法を指示します。このオプションは、MQPUT 呼び出しでのみ指定できます。MQPUT1 呼び出しでは無効です。

PMLOGO が指定されると、アプリケーションは MQPUT 呼び出しを続けて使用して、以下のことを行います。

- ・各論理メッセージ内のセグメントを、0 からセグメント・オフセットの小さい順に間を空けずに書き込む。
- ・論理メッセージ内のセグメントをすべて書き込んでから、その次の論理メッセージのセグメントを書き込みます。
- ・各メッセージ・グループ内の論理メッセージを、1 からメッセージ順序番号の小さい順に間を空けずに書き込む。
- ・メッセージ・グループ内の論理メッセージをすべて書き込んでから、その次のメッセージ・グループの論理メッセージを書き込みます。

この順序を「論理順序」といいます。

アプリケーションはキュー・マネージャーにグループ内のメッセージと論理メッセージのセグメントを書き込む方法を指示したため、MQPUT を呼び出すたびにグループの情報やセグメントの情報を維持および更新する必要はありません。これについてはキュー・マネージャーが代わりに行います。具体的には、キュー・マネージャーが *MDGID*、*MDSEQ*、および *MDOFF* フィールドを適切な値に設定するため、アプリケーションが MQMD 内のこれらのフィールドを設定する必要がないことを意味します。アプリケーションが設定する必要があるのは、MQMD 内の *MDMFL* フィールドのみです。これは、メッセージがグループに属しているか、論理メッセージのセグメントであるかを示し、グループ内の最後のメッセージまたは論理メッセージの最後のセグメントを示します。

メッセージ・グループまたは論理メッセージが開始されると、後続の MQPUT 呼び出しでは、MQMD 内の *MDMFL* に適切な MF* フラグを指定する必要があります。終了していないメッセージ・グループがあるときにアプリケーションがグループ内にないメッセージ、または終了していない論理メッセージがあるときに、セグメントではないメッセージを書き込もうとすると、その呼び出しは失敗します。また、状況に応じて理由コード RC2241 または RC2242 が表示されます。ただし、キュー・マネージャーは現在のメッセージ・グループまたは現在の論理メッセージの情報を保存し、アプリケーションはメッセージを送信してこれらを終了します (アプリケーション・メッセージ・データなしも可能です)。このメッセージでは状況に応じて *MFLMIG* または *MFLSEG* を指定します。その後 MQPUT 呼び出しを再発行して、グループまたはセグメントにないメッセージを書き込みます。

[1185 ページの表 755](#) は、有効なオプションとフラグの組み合わせ、および各ケースでキュー・マネージャーが使用する *MDGID*、*MDSEQ*、および *MDOFF* フィールドの値を示しています。この表に示されていないオプションとフラグの組み合わせは無効です。この表の列の意味は、次のとおりです。

LOG ORD

PMLOGO オプションが呼び出しで指定されているかどうかを示します。

MIG

MFmig または MFLMIG オプションが呼び出しで指定されているかどうかを示します。

SEG

MFseg または MFLSEG オプションが呼び出しで指定されているかどうかを示します。

SEG OK

MFSEGA オプションが呼び出しで指定されているかどうかを示します。

Cur grp

現行のメッセージ・グループが呼び出しの前に存在するかどうかを示します。

Cur log msg

現行の論理メッセージが呼び出しの前に存在するかどうかを示します。

その他の列

キュー・マネージャーが使用する値を示しています。「前の」という表現は、キュー・ハンドルに対して前のメッセージのフィールドで使用された値を示します。

PMRLOC

MQPMO 構造体の PMRQN に、メッセージを実際に書き込むローカル・キューの名前を設定する必要があることを指定します。ResolvedQMGrName には、同様にローカル・キューをホストするローカル・キュー・マネージャーの名前が入ります。これが意味することについては OORLOQ を参照してください。キューへの書き込みが許可されている場合、ユーザーはこのフ

ラグを MQPUT 呼び出しで指定するために必要な許可を持っています。特殊権限は必要ありません。

表 755. 論理メッセージのグループおよびセグメントに関連した MQPUT オプション								
指定するオプション				呼び出しの前のグループおよび論理メッセージの状況		キュー・マネージャーが使用する値		
LOG ORD	MIG	SEG	SEG OK	Cur grp	Cur log msg	MDGID	MDSEQ	MDOFF
はい	いいえ	いいえ	いいえ	いいえ	いいえ	GINONE	1	0
はい	いいえ	いいえ	はい	いいえ	いいえ	新しいグループ ID	1	0
はい	いいえ	はい	はい/いいえ	いいえ	いいえ	新しいグループ ID	1	0
はい	いいえ	はい	はい/いいえ	いいえ	はい	前のグループ ID	1	前のオフセット + 前のセグメント長
はい	はい	はい/いいえ	はい/いいえ	いいえ	いいえ	新しいグループ ID	1	0
はい	はい	はい/いいえ	はい/いいえ	はい	いいえ	前のグループ ID	前の順序番号 + 1	0
はい	はい	はい	はい/いいえ	はい	はい	前のグループ ID	前の順序番号	前のオフセット + 前のセグメント長
いいえ	いいえ	いいえ	いいえ	はい/いいえ	はい/いいえ	GINONE	1	0
いいえ	いいえ	いいえ	はい	はい/いいえ	はい/いいえ	GINONE の場合は新しいグループ ID、その他はフィールド内の値	1	0
いいえ	いいえ	はい	はい/いいえ	はい/いいえ	はい/いいえ	GINONE の場合は新しいグループ ID、その他はフィールド内の値	1	フィールド内の値
いいえ	はい	いいえ	はい/いいえ	はい/いいえ	はい/いいえ	GINONE の場合は新しいグループ ID、その他はフィールド内の値	フィールド内の値	0
いいえ	はい	はい	はい/いいえ	はい/いいえ	はい/いいえ	GINONE の場合は新しいグループ ID、その他はフィールド内の値	フィールド内の値	フィールド内の値

注:

- MQPUT1 呼び出しでは、PMLOGO は無効です。
- MDMID フィールドについては、PMNMID または MINONE が指定されている場合、キュー・マネージャーは新しいメッセージ ID を生成し、それ以外の場合はフィールドの値を使用します。
- MDCID フィールドについては、PMNCID が指定されている場合はキュー・マネージャーが新しい相関 ID を生成し、それ以外の場合はフィールドの値を使用します。

PMLOGO が指定されている場合、キュー・マネージャーは、グループ内のすべてのメッセージと論理メッセージ内のセグメントが MQMD の *MDPER* フィールドに同じ値で書き込まれることを必要とします。つまり、すべてが持続であるか、またはすべてが非持続でなければなりません。この条件を満たさないと、MQPUT 呼び出しは失敗し、理由コード RC2185 が戻ります。

PMLOGO オプションが作業単位に及ぼす影響は、以下のとおりです。

- グループ内または論理メッセージ内の最初の物理メッセージが 1 つの作業単位に書き込まれた場合、そのグループ内または論理メッセージ内の他の物理メッセージも、同じキュー・ハンドルが使用されていれば、すべて 1 つの作業単位に書き込む必要があります。ただし、これらは同じ作業単位内で書き込む必要はありません。これにより、複数の物理メッセージから成る 1 つのメッセージ・グループまたは論理メッセージを、キュー・ハンドルに対する 2 つ以上の連続した作業単位にまたがって分割できます。
- グループ内または論理メッセージ内の最初の物理メッセージが 1 つの作業単位に書き込まれていない場合、同じキュー・ハンドルが使用されていれば、そのグループ内または論理メッセージ内の他の物理メッセージはどれも 1 つの作業単位に書き込むことができません。

これらの条件が満たされないと、MQPUT 呼び出しは失敗し、理由コード RC2245 が戻ります。

PMLOGO を指定した場合には、MQPUT 呼び出しで供給された MQMD が、MDVER2 より下位であってはなりません。この条件を満たさないと、この呼び出しは失敗し、理由コード RC2257 が戻ります。

PMLOGO を指定しないと、グループ内のメッセージおよび論理メッセージ内のセグメントは任意の順序で書き込まれます。また、完全なメッセージ・グループまたは完全な論理メッセージを書き込む必要はありません。*MDGID*、*MDSEQ*、*MDOFF*、および *MDMFL* フィールドが適切な値を持つようにするのは、アプリケーションの責任です。

この手法を用いると、システム障害が発生した後に、メッセージ・グループまたは論理メッセージを途中から再開することができます。システムが再始動すると、アプリケーションは *MDGID*、*MDSEQ*、*MDOFF*、*MDMFL*、および *MDPER* の各フィールドを適切な値に設定し、PMSYP または PMNSYP を必要に応じて設定して、PMLOGO を指定せずに MQPUT 呼び出しを発行することができます。この呼び出しが成功した場合、キュー・マネージャーはグループとセグメントの情報を保存し、そのキュー・ハンドルを使用する後続の MQPUT 呼び出しで通常どおり PMLOGO を指定できます。

MQPUT 呼び出しのためにキュー・マネージャーが保持しているグループおよびセグメント情報は、MQGET 呼び出しのためにキュー・マネージャーが保持しているグループおよびセグメント情報とは異なります。

キュー・ハンドルが指定されている場合には、アプリケーションでは、PMLOGO を指定した MQPUT 呼び出しと PMLOGO を指定していない MQPUT 呼び出しを自由に組み合わせて使用できます。ただし、以下の点に注意してください。

- PMLOGO を指定していない場合は、MQPUT 呼び出しが成功するたびに、キュー・マネージャーが、キュー・ハンドルのグループおよびセグメント情報を、アプリケーションによって指定された値に設定します。これにより、キュー・ハンドルに対してキュー・マネージャーで保持されていた既存のグループおよびセグメント情報が置換されます。
- PMLOGO を指定していない場合、現行のメッセージ・グループまたは論理メッセージがあれば、呼び出しは失敗しません。ただし、CCWARN 完了コードで呼び出しが成功する場合もあります。[1187 ページの表 756](#) に、発生する可能性のあるいくつかのケースを示しています。これらの場合に、完了コードが CCOK 以外であれば、理由コードは以下のいずれか (該当するもの) になります。

- RC2241
- RC2242
- RC2185
- RC2245

注: キュー・マネージャーは、MQPUT1 呼び出しのためにグループ情報およびセグメント情報を確認しません。

表 756. グループおよびセグメント情報と整合性のない MQPUT または MQCLOSE 呼び出しの結果		
現行の呼び出し	直前の呼び出しは PMLOGO を指定している MQPUT だった	直前の呼び出しは PMLOGO を指定していない MQPUT だった
PMLOGO を指定している MQPUT	CCFAIL	CCFAIL
PMLOGO を指定していない MQPUT	CCWARN	CCOK
終了していないグループまたは論理メッセージを指定している MQCLOSE	CCWARN	CCOK

単にメッセージおよびセグメントを論理順序で書き込むアプリケーションでは、最も簡単に使える PMLOGO を指定するようにしてください。このオプションを指定すると、キュー・マネージャーがグループおよびセグメント情報を管理するので、アプリケーションでこの情報を管理する必要はなくなります。しかし、PMLOGO オプションで提供される制御以外の制御が必要となる 特殊なアプリケーションもあります。このようなアプリケーションでは、このオプションを指定しないようにしてください。これを行う場合、アプリケーションは、それぞれの MQPUT または MQPUT1 呼び出しの前に、MQMD 内の MDGID、MDSEQ、MDOFF、および MDMFL フィールドが正しく設定されていることを確認する必要があります。

例えば、受信した物理メッセージがグループに属していなくても、また論理メッセージのセグメントでなくても、そのメッセージを転送するアプリケーションでは、PMLOGO を指定しないようにしてください。これには次の 2 つの理由があります。

- メッセージが検索されて順番に書き込まれる場合、PMLOGO を指定するとメッセージに新しいグループ ID が割り当てられ、これによりメッセージの発信元がメッセージ・グループから得られた応答メッセージまたはレポート・メッセージとの相関をとることが困難になります。場合によっては相関をとることが不可能になります。
- 送信側のキュー・マネージャーと受信側のキュー・マネージャーの間にパスが複数あるような複雑なネットワークの場合には、物理メッセージが正しくない順序で到達することがあります。PMLOGO およびこれに対応する MQGET 呼び出し上の GMLOGO を指定しないようにすると、転送側のアプリケーションでは、論理順序で次にあるメッセージが到着するのを待たなくても、それぞれの物理メッセージの到着と同時にそのメッセージを取り出し、転送することができます。

グループ内のメッセージまたは論理メッセージのセグメントに関するレポート・メッセージを生成するアプリケーションでも、レポート・メッセージを書き込むときには PMLOGO を指定してはなりません。

PMLOGO は、他のすべての PM* オプションと組み合わせて指定できます。

コンテキスト・オプション: 以下のオプションは、メッセージ・コンテキストの処理を制御します。

PMNOC

このメッセージに関連するコンテキストはありません。

コンテキストが存在しないことを示すために、識別コンテキストと起点コンテキストの両方が設定されます。つまり、MQMD のコンテキスト・フィールドは次のように設定されます。

- 文字フィールドの場合はブランク
- バイト・フィールドの場合はヌル
- 数値フィールドの場合はゼロ

PMDEFC

デフォルトのコンテキストを使用します。

識別および発信元の両方のデフォルトのコンテキスト情報がメッセージに関連付けられます。キュー・マネージャーは、メッセージ記述子のコンテキスト・フィールドを以下のように設定します。

表 757. MQMD フィールドのデフォルトのコンテキスト情報の値

MQMD のフィールド	使用される値
MDUID	環境から決定できる場合はその値。それ以外のときは、ブランクに設定される。
MDACC	可能な場合は、環境から判別される。判別できない場合は ACNONE に設定。
MDAID	ブランクに設定されます。
MDPAT	環境から決定される。
MDPAN	環境から決定できる場合はその値。それ以外のときは、ブランクに設定される。
MDPD	メッセージが書き込まれる日付に設定。
MDPT	メッセージが書き込まれる時刻に設定。
MDAOD	ブランクに設定されます。

メッセージ・コンテキストについて詳しくは、[メッセージ・コンテキスト](#) および [コンテキスト情報の制御](#)を参照してください。

これは、コンテキスト・オプションが設定されていないときのデフォルト・アクションです。

PMPASI

入力キュー・ハンドルから識別コンテキストを渡します。

メッセージには、識別コンテキストに関連付けられているコンテキスト情報が含まれます。識別コンテキストは、PMCT フィールドで指定されたキュー・ハンドルから取得されます。起点コンテキスト情報は、PMDEFC の場合と同様にキュー・マネージャーによって生成されます (値については上記の表を参照)。メッセージ・コンテキストについての詳細は、[メッセージ・コンテキスト](#) および [コンテキスト情報の制御](#)を参照してください。

MQPUT 呼び出しの場合、キューは OOPASI オプション (またはそれを暗黙的に指定するオプション) を指定してオープンされていることが必要です。MQPUT1 呼び出しの場合、OOPASI オプションを指定した MQOPEN 呼び出しの場合と同様の許可検査が行われます。

PMPASA

入力キュー・ハンドルからすべてのコンテキストを渡します。

メッセージには、識別コンテキストに関連付けられているコンテキスト情報が含まれます。識別コンテキストと起点コンテキストの両方が、PMCT フィールドに指定されたキュー・ハンドルから取得されます。メッセージ・コンテキストについての詳細は、[メッセージ・コンテキスト](#) および [コンテキスト情報の制御](#)を参照してください。

MQPUT 呼び出しの場合、キューは OOPASA オプション (またはそれを暗黙的に指定するオプション) を指定してオープンされていることが必要です。MQPUT1 呼び出しの場合、OOPASA オプションを指定した MQOPEN 呼び出しの場合と同様の許可検査が行われます。

PMSETI

アプリケーションからすべての識別コンテキストを設定します。

メッセージには、識別コンテキストに関連付けられているコンテキスト情報が含まれます。アプリケーションでは、MQMD 構造体の識別コンテキストを指定します。起点コンテキスト情報は、PMDEFC の場合と同様にキュー・マネージャーによって生成されます (値については上記の表を参照)。メッセージ・コンテキストについて詳しくは、[メッセージ・コンテキスト](#) および [コンテキスト情報の制御](#)を参照してください。

MQPUT 呼び出しの場合、キューは OOSSETI オプション (またはそれを暗黙的に指定するオプション) を指定してオープンされていることが必要です。MQPUT1 呼び出しの場合、OOSSETI オプションを指定した MQOPEN 呼び出しの場合と同様の許可検査が行われます。

PMSETA

アプリケーションから、すべてのコンテキストを設定します。

メッセージには、識別コンテキストに関連付けられているコンテキスト情報が含まれます。アプリケーションでは、MQMD 構造体の識別コンテキストと起点コンテキストを指定します。メッセージ・コンテキストについて詳しくは、[メッセージ・コンテキスト](#) および [コンテキスト情報の制御](#) を参照してください。

MQPUT 呼び出しの場合、キューは OOSSETA オプションを指定してオープンされていることが必要です。MQPUT1 呼び出しの場合、OOSSETA オプションを指定した MQOPEN 呼び出しの場合と同様の許可検査が行われます。

PM* コンテキスト・オプションのうち、1 つのみを指定できます。このオプションを指定しないと、PMDEFC を指定したと見なされます。

応答タイプを書き込みます。以下のオプションは、MQPUT または MQPUT1 呼び出しに戻される応答を制御します。これらのオプションのどちらか 1 つだけを指定できます。PMARES および PMSRES を指定しない場合は、PMRASQ または PMRAST を指定したものと見なされます。

PMARES

PMARES オプションは、アプリケーションがキュー・マネージャーによる呼び出しの完了を待機せずに、MQPUT または MQPUT1 操作を完了するように要求します。このオプションを使用すると、メッセージング・パフォーマンスが改善される可能性があります。クライアント・バインディングを使用するアプリケーションの場合は特にそうです。アプリケーションは、MQSTAT verb を使って、前の非同期呼び出し中にエラーが発生したかどうかを定期的に検査することができます。

このオプションの場合、MQMD の以下のフィールドだけに値が入れられることが保証されます。

- MDAID
- MDPAT
- MDPAN
- MDAOD

さらに、PMNMID または PMNCID のいずれかまたは両方がオプションとして指定された場合、戻される MDMID および MDCID の値も入れられます。(ブランクの MDMID フィールドを指定することによって、PMNMID を暗黙的に指定することができます)。

あらかじめ指定済みのフィールドのみが完了されます。通常、MQMD または MQPMO 構造体で戻されるその他の情報は未定義となっています。

MQPUT または MQPUT1 用に非同期書き込み応答を要求する場合、CCOK および RCNONE の CMPCOD および REASON は、必ずしもメッセージがキューに正常に書き込まれたことを意味してはわけではありません。非同期書き込み応答を使用し、メッセージがキューに書き込まれたことを確認する必要がある MQI アプリケーションを開発する場合、書き込み操作から戻った CMPCOD および REASON コードの両方をチェックし、さらに MQSTAT を使用して非同期エラー情報を照会しなければなりません。

各 MQPUT/MQPUT1 呼び出しの成功または失敗はすぐには戻されないかもしれませんが、非同期呼び出しの下で発生した最初のエラーは、後で MQSTAT を呼び出すことによって判別できます。

同期点にある持続メッセージを非同期書き込み応答を使用して送達できなかった場合、トランザクションをコミットしようとしても、コミットは失敗し、トランザクションは完了コード CCFAIL および理由 RC2003 でバックアウトされます。アプリケーションは MQSTAT を呼び出して、前の MQPUT または MQPUT1 が失敗した原因を判別することができます。

同期ポイント下の非永続メッセージが非同期プット応答による配信に失敗し、トランザクションのコミットを試みた場合、コミットは成功するが非永続メッセージは失われる。この状況を回避するには、アプリケーションは MQCMIT を呼び出す前に、MQSTAT を呼び出して、以前に MQPUT または MQPUT1 の失敗があったかどうかを判断する必要があります。失敗が発生した場合、アプリケーションは MQBACK を呼び出すべきである。

PMSRES

この値を MQPMO 構造体の書き込みオプションに指定すると、MQPUT または MQPUT1 操作は常に同期して実行されます。操作が正常に行われると、MQMD および MQPMO のすべてのフィールドに値が入れられます。これは、同期応答がキューまたはトピック・オブジェクトで定義されるデフォルトの書き込み応答の値とは関係なくするためのものです。

PMRASQ

MQPUT 呼び出しでこの値が指定されている場合、使用される書き込み応答タイプは、アプリケーションによって開かれたときにキューで指定された DEFPRESP 値からとられます。IBM WebSphere MQ 7.0 より前のレベルのキュー・マネージャーに接続したクライアント・アプリケーションは、PMSRES が指定された場合と同じように動作します。

このオプションが MQPUT1 呼び出しで指定されると、キュー定義からの DEFPRESP 値は使用されません。MQPUT1 呼び出しが PMSYP を使用する場合には PMARES と同じように動作し、PMNSYP を使用する場合には PMSRES と同じように動作します。

PMRAST

これは、トピック・オブジェクトで使用するための PMRASQ と同義語です。

その他のオプション: 以下のオプションは、許可検査を制御するほか、キュー・マネージャーが静止しているときに発生するイベントを制御します。

PMALTU

指定されたユーザー ID を用いて妥当性検査を行います。

これは、MQPUT1 呼び出しの **OBJDSC** パラメーターの *ODAU* フィールドに、キューにメッセージを書き込む権限の妥当性検査に使用されるユーザー ID が含まれていることを示します。呼び出しが成功するのは、この *ODAU* が、指定されたオプションを使用してキューをオープンすることを許可されている場合のみです。これは、アプリケーションを実行しているユーザー ID が許可されているかどうかには関係ありません。(ただし、これは、指定されたコンテキスト・オプションには適用されず、検査は常に、アプリケーションが実行されているユーザー ID に対して行われます。)

このオプションは、MQPUT1 呼び出しの場合にのみ有効です。

PMFIQ

キュー・マネージャーが静止している場合は、失敗します。

このオプションを指定すると、キュー・マネージャーが静止状態にある場合、MQPUT または MQPUT1 呼び出しが強制的に失敗します。

呼び出しは、理由コード RC2161 と共に完了コード CCFAIL を戻します。

デフォルト・オプション: 上記のオプションのいずれも必要がない場合は、以下のオプションを使用することができます。

PMNONE

指定されるオプションはありません。

この値は、他のオプションが指定されなかったことを示すために使用できます。すべてのオプションはデフォルト値を取ります。PMNONE は、プログラムの文書化をサポートするために定義します。したがって、このオプションは、他のオプションと同時に使用するものではありません。しかしこのオプションの値はゼロのため、他のオプションと同時に使用してもそれを検出できません。

これは入力フィールドです。PMOPT フィールドの初期値は PMNONE です。

PMPRF (10 桁の符号付き整数)

どの MQPMR フィールドが存在しているかを示すフラグ。

このフィールドには、アプリケーションが提供する書き込みメッセージ・レコードにどの MQPMR フィールドが存在するかを示すように設定しなければならないフラグが入っています。PMPRF は、メッセージが配布リストに書き込まれている間だけ使用されます。PMREC がゼロの場合、または PMPRO と PMPRP の両方がゼロの場合、このフィールドは無視されます。

書き込みメッセージ・レコードにあるフィールドについては、キュー・マネージャーは宛先ごとに対応する書き込みメッセージ・レコードのフィールドにある値を使用します。書き込みメッセージ・レコードにないフィールドについては、キュー・マネージャーは MQMD 構造体にある値を使用します。

以下のフラグを 1 つ以上指定して、書き込みメッセージ・レコードにどのフィールドがあるのか表示できます。

PFMID

メッセージ ID フィールドがある。

PFCID

相関 ID フィールドがある。

PFGID

グループ ID フィールドがある。

PFFB

フィードバック・フィールドがある。

PFACC

会計トークン・フィールドがある。

このフラグを指定しない場合は、*PMOPT* フィールドに *PMSETI* または *PMSETA* を指定する必要があります。この条件を満たさないと、その呼び出しは失敗し、理由コード RC2158 が戻ります。

MQPMR フィールドがない場合は、以下を指定できます。

PFNONE

書き込みメッセージ・レコードのフィールドがない。

この値を指定する場合は、*PMREC* をゼロにするか、*PMPRO* と *PMPRP* の両方をゼロにする必要があります。

PFNONE は、プログラムの文書化を支援するために定義します。この定数は他の定数と組み合わせて使用する実数ではありません。ただし、この定数の値はゼロと等価なので、ほかの実数と組み合わせて使用しても、エラーとして検出されることはありません。

PMPRF に無効なフラグが入っている場合、または書き込みメッセージ・レコードが提供されたが *PMPRF* の値が *PFNONE* である場合、その呼び出しは失敗し、理由コード RC2158 が戻ります。

これは入力フィールドです。このフィールドの初期値は *PFNONE* です。*PMVER* が *PMVER2* より小さい場合、このフィールドは無視されます。

PMPRO (10 桁の符号付き整数)

MQPMO の先頭から最初の書き込みメッセージ・レコードのオフセット。

これは、MQPMO 構造体の先頭から最初の MQPMR 書き込みメッセージ・レコードのオフセットをバイト数で表したものです。オフセットの値は、正負どちらの値にもなります。*PMPRO* は、メッセージが配布リストに書き込まれている間だけ使用されます。*PMREC* がゼロの場合、このフィールドは無視されます。

メッセージが配布リストに書き込まれている間、1 つまたは複数の MQPMR 書き込みメッセージ・レコードの配列は、そのメッセージに特定の特性を、宛先に応じて個別に指定するために提供されます。ここで言う特性とは、以下のとおりです。

- メッセージ ID
- 相関 ID
- グループ ID
- フィードバック値
- accounting token

上記のプロパティをすべて指定する必要はありません。しかし、どのサブセットを選択する場合でも、フィールドは正しい順序で指定する必要があります。詳細については、MQPMR 構造体を参照してください。

配布リストをオープンすると、通常はMQODで指定されたオブジェクト・レコードと同じ数のメッセージ・レコードがあります。したがって、各書き込みメッセージ・レコードは、対応するオブジェクト・レコードで識別されたキューにメッセージ・プロパティを供給します。オープンに失敗した配布リストのキューでは、割り当てられた書き込みメッセージ・レコードが配列の該当する位置にまだ残っています。ただし、この場合メッセージ・プロパティは無視されます。

書き込みメッセージ・レコードの数は、オブジェクト・レコードの数とは異なる可能性があります。メッセージ・レコードがオブジェクト・レコードよりも少ない場合は、書き込みメッセージ・レコードのない宛先に対するメッセージ・プロパティは、メッセージ記述子MQMDの対応するフィールドから取得します。書き込みメッセージ・レコードがオブジェクト・レコードよりも多い場合は、超過分は使用されません(それでも、超過分へのアクセスは可能です)。書き込みメッセージ・レコードはオプションですが、指定する場合は、メッセージ・レコードのPMRECが必要です。

書き込みメッセージ・レコードは、PMPROでオフセットを指定するか、PMPRPでアドレスを指定することによって、MQODのオブジェクト・レコードと同様の方法で提供できます。これを行う方法について詳しくは、1166ページの『IBM iでのMQOD (オブジェクト記述子)』で説明しているODOROフィールドを参照してください。

PMPROおよびPMPRPの両方は使用できません。両方ともゼロでない場合、その呼び出しは失敗し、理由コードRC2159を戻します。

これは入力フィールドです。このフィールドの初期値は0です。PMVERがPMVER2より小さい場合、このフィールドは無視されます。

PMPRP (ポインター)

最初の書き込みメッセージ・レコードのアドレス。

これは、最初のMQPMR書き込みメッセージのアドレスです。PMPRPは、メッセージが配布リストに書き込まれている間だけ使用されます。PMRECがゼロの場合、このフィールドは無視されます。

PMPRPまたはPMPROのいずれかを使用して、書き込みメッセージ・レコードを指定できますが、両方を指定することはできません。詳しくは、PMRROフィールドの説明を参照してください。PMPRPを使用しない場合は、ヌル・ポインターまたはヌル・バイトに設定する必要があります。

これは入力フィールドです。このフィールドの初期値は、ヌル・ポインターです。PMVERがPMVER2より小さい場合、このフィールドは無視されます。

PMREC (10桁の符号付き整数)

書き込みメッセージ・レコードの数または存在する応答レコードの数。

これは、アプリケーションが提供したMQPMR書き込みメッセージ・レコードまたはMQRR応答レコードの数です。この数はメッセージが配布リストに書き込まれる場合に限り、ゼロを超えることができます。書き込みメッセージ・レコードおよび応答レコードはオプションです。アプリケーションはレコードを提供する必要はありませんが、いずれか一方のタイプのレコードだけを提供することができます。ただし、アプリケーションが両方のタイプのレコードを提供する場合は、各タイプのPMRECレコードを提供する必要があります。

PMRECの値は、配布リスト内の宛先の数と同じである必要はありません。提供されるレコードの数が多すぎる場合、超過分は使用されません。提供されるレコードの数が少なすぎる場合、書き込みメッセージ・レコードのない宛先のメッセージ・プロパティにはデフォルト値が使用されます(このトピックで後述するPMPROを参照してください)。

PMRECがゼロより小さい場合、またはゼロより大きいメッセージが配布リストに書き込まれなかった場合、その呼び出しは失敗し、理由コードRC2154を戻します。

これは入力フィールドです。このフィールドの初期値は0です。PMVERがPMVER2より小さい場合、このフィールドは無視されます。

PMRMN (48バイトの文字ストリング)

宛先キュー・マネージャーの解決された名前。

これは、ローカル・キュー・マネージャーが名前の解決を実行した後の宛先キュー・マネージャーの名前です。戻される名前は、*PMRQN* によって識別されるキューを所有するキュー・マネージャーの名前であり、ローカル・キュー・マネージャーの名前にすることができます。

PMRQN が、ローカル・キュー・マネージャーが属するキュー共有グループが所有する共有キューである場合、*PMRMN* はそのキュー共有グループの名前です。キューが他のキュー共有グループによって所有されている場合、*PMRQN* は、キュー共有グループの名前またはキュー共有グループのメンバーであるキュー・マネージャーの名前にすることができます (返される値の性質は、ローカル・キュー・マネージャーに存在するキュー定義によって異なります)。

非ブランクの値が戻されるのは、オブジェクトが単一キューである場合のみです。オブジェクトが配布リストまたはトピックである場合、戻される値は未定義です。

これは出力フィールドです。このフィールドの長さは *LNQMN* によって指定されます。このフィールドの初期値は 48 個のブランク文字です。

PMRQN (48 バイトの文字ストリング)

宛先キューの解決された名前。

これは、ローカル・キューが名前の解決を実行した後の宛先キュー・マネージャーの名前です。戻される名前は、*PMRMN* によって識別されるキュー・マネージャー上に存在するキューの名前です。

非ブランクの値が戻されるのは、オブジェクトが単一キューである場合のみです。オブジェクトが配布リストまたはトピックである場合、戻される値は未定義です。

これは出力フィールドです。このフィールドの長さは *LNQN* によって指定されます。このフィールドの初期値は 48 個のブランク文字です。

PMRRO (10 桁の符号付き整数)

MQPMO の先頭から最初の応答レコードのオフセット。

これは、*MQPMO* 構造体の先頭から最初の *MQRR* 応答レコードのオフセットをバイト数で表したものです。オフセットの値は、正負どちらの値にもなります。*PMRRO* は、メッセージが配布リストに書き込まれている間だけ使用されます。*PMREC* がゼロの場合、このフィールドは無視されます。

メッセージが配布リストに書き込まれるときに、メッセージが正常に送信されなかったキュー (*MQRR* の *RRCC* フィールド)、および各障害の理由 (*MQRR* の *RRREA* フィールド) を識別するために、1 つ以上の *MQRR* 応答レコードの配列を提供できます。メッセージの送信が失敗する原因としては、キューのオープン失敗や *PUT* 操作の失敗などが考えられます。キュー・マネージャーが応答コードを設定するのは、呼び出しの結果が一定でない場合のみです。結果が一定でない場合とは、送信できたメッセージと送信できなかったメッセージが混在している場合や、どのメッセージの送信も失敗したが、失敗した理由がそれぞれ異なる場合などです。後者の場合は、呼び出しの結果として理由コード *RC2136* が戻ります。すべてのキューに同じ理由コードが適用される場合は、その理由コードが *MQPUT* または *MQPUT1* 呼び出しの **REASON** パラメーターに返され、応答レコードは設定されません。

配布リストをオープンすると、通常は *MQOD* で指定されたオブジェクト・レコードと同じ数の応答レコードがあります。したがって各応答レコードは、対応するオブジェクト・レコードで識別されたキューへ書き込むために、必要に応じて完了コードおよび理由コードを設定します。オープンに失敗した配布リストのキューには、割り当てられた応答レコードが配列の適切な位置にまだ残っています。ただし、この場合応答レコードには書き込み操作ではなくオープン操作で生じる完了コードおよび理由コードが設定されます。

応答レコードの数は、オブジェクト・レコードの数とは異なる可能性があります。応答レコードの数がオブジェクト・レコードの数より少ない場合、アプリケーションは書き込み操作に失敗したすべての宛先や失敗の理由を識別することができない可能性があります。応答レコードがオブジェクト・レコードよりも多い場合は、超過分は使用されません。ただし、依然として超過分へのアクセスは可能です。応答レコードはオプションですが、指定する場合は、応答レコードの *PMREC* が必要です。

応答レコードは、*PMRRO* でオフセットを指定するか、*PMRRP* でアドレスを指定することによって、*MQOD* のオブジェクト・レコードと同様の方法で提供できます。これを行う方法について詳しくは、[1166 ページの『IBM i での MQOD \(オブジェクト記述子\)』](#)で説明している *ODORO* フィールドを参照し

てください。ただし、*PMRRO* および *PMRRP* の両方を使用することはできません。両方ともゼロでない場合、呼び出しは失敗し、理由コード RC2156 が戻ります。

MQPUT1 呼び出しの場合は、このフィールドをゼロにする必要があります。これは必要があれば、応答情報がオブジェクト記述子 *MQOD* で指定された応答レコードに戻るからです。

これは入力フィールドです。このフィールドの初期値は 0 です。*PMVER* が *PMVER2* より小さい場合、このフィールドは無視されます。

PMRRP (ポインター)

最初の応答レコードのアドレス。

これは、最初の *MQRR* 応答レコードのアドレスです。*PMRRP* は、メッセージが配布リストに書き込まれている間だけ使用されます。*PMREC* がゼロの場合、このフィールドは無視されます。

PMRRP または *PMRRO* のいずれかを使用して応答レコードを指定できますが、両方を指定することはできません。詳しくは、*PMRRO* フィールドの説明を参照してください。*PMRRP* を使用しない場合は、ヌル・ポインターまたはヌル・バイトに設定する必要があります。

MQPUT1 呼び出しの場合は、このフィールドをヌル・ポインターまたはヌル・バイトにする必要があります。これは必要があれば、応答情報がオブジェクト記述子 *MQOD* で指定された応答レコードに戻るからです。

これは入力フィールドです。このフィールドの初期値は、ヌル・ポインターです。*PMVER* が *PMVER2* より小さい場合、このフィールドは無視されます。

PMSID (4 バイトの文字ストリング)

構造体 ID

値は次のものでなければなりません。

PMSIDV

書き込みメッセージ・オプション構造体の ID。

これは常に入力フィールドです。このフィールドの初期値は *PMSIDV* です。

PMSL (MQLONG)

このパブリケーションのターゲットになるサブスクリプションのレベル。

このパブリケーションを受け取るのは、*PMSL* がこの値以下の最も高いサブスクリプションのみです。この値はゼロから 9 までの範囲内でなければなりません。ゼロは最低レベルです。

このフィールドの初期値は 9 です。

PMT0 (10 桁の符号付き整数)

予約済み。

これは、予約フィールドです。したがって、値に意味はありません。このフィールドの初期値は -1 です。

PMUDC (10 桁の符号付き整数)

リモート・キューへの送信が成功したメッセージの数。

これは、現在の *MQPUT* 呼び出しまたは *MQPUT1* 呼び出しがリモート・キューを解決する配布リスト中のキューへの送信に成功したメッセージの数です。キュー・マネージャーが配布リストの形式中に一時的に保存したメッセージは、これらの配布リストが含む個々の宛先の数として数えられます。このフィールドは、配布リストにはない単一のキューにメッセージを書き込むときも設定されます。

これは出力フィールドです。このフィールドの初期値は 0 です。*PMVER* が *PMVER2* より小さい場合、このフィールドは設定されません。

PMVER (10 桁の符号付き整数)

構造体のバージョン番号。

値は次のいずれかでなければなりません。

PMVER1

バージョン 1 の書き込みメッセージ・オプション構造体。

PMVER2

バージョン 2 の書き込みメッセージ・オプション構造体。

これより新しいバージョンの構造体にのみ存在するフィールドは、そのフィールドの説明にその旨記載されています。以下の定数は、現行バージョンのバージョン番号を指定しています。

PMVERC

書き込みメッセージ・オプション構造体の現行バージョン。

これは常に入力フィールドです。このフィールドの初期値は PMVER1 です。

初期値

表 758. MQPMO のフィールド		
フィールド名	定数の名前	定数の値
PMSID	PMSIDV	'PMO '
PMVER	PMVER1	1
PMOPT	PMNONE	0
PMT0	なし	-1
PMCT	なし	0
PMKDC	なし	0
PMUDC	なし	0
PMIDC	なし	0
PMRQN	なし	ブランク
PMRMN	なし	ブランク
PMREC	なし	0
PMPRF	PFNONE	0
PMPRO	なし	0
PMRRO	なし	0
PMPRP	なし	ヌル・ポインターまたはヌル・バイト
PMRRP	なし	ヌル・ポインターまたはヌル・バイト
注:		
1. 記号-は、単一のブランク文字を表します。		

RPG 宣言

```
D*..1.....2.....3.....4.....5.....6.....7..
D* MQPMO Structure
D*
D* Structure identifier
D  PMSID              1      4    INZ('PMO ')
D* Structure version number
D  PMVER              5      8I 0  INZ(1)
```

```

D* Options that control the action of MQPUT and MQPUT1
D PMOPT          9      12I 0 INZ(0)
D* Reserved
D PMTO           13      16I 0 INZ(-1)
D* Object handle of input queue
D PMCT           17      20I 0 INZ(0)
D* Number of messages sent successfully to local queues
D PMKDC          21      24I 0 INZ(0)
D* Number of messages sent successfully to remote queues
D PMUDC          25      28I 0 INZ(0)
D* Number of messages that could not be sent
D PMIDC          29      32I 0 INZ(0)
D* Resolved name of destination queue
D PMRQN          33      80    INZ
D* Resolved name of destination queue manager
D PMRMN          81     128    INZ
D* Number of put message records or response records present
D PMREC         129     132I 0 INZ(0)
D* Flags indicating which MQPMR fields are present
D PMPRF         133     136I 0 INZ(0)
D* Offset of first put message record from start of MQPMO
D PMPRO         137     140I 0 INZ(0)
D* Offset of first response record from start of MQPMO
D PMRRO         141     144I 0 INZ(0)
D* Address of first put message record
D PMPRP         145     160*   INZ(*NULL)
D* Address of first response record
D PMRRP         161     176*   INZ(*NULL)
D* Original message handle
D PMOMH         177     184I 0
D* New message handle
D PMNMH         185     190I 0
D* The action being performed
D PMACT         191     194I 0
D* Reserved
D PMRE1         195     198I 0

```

IBM i での MQPMR (書き込みメッセージ・レコード)

MQPMR 構造体は、メッセージを配布リストに書き込んでいるときに、単一宛先用の種々のメッセージ・プロパティを指定するのに使用します。

概要

目的: MQPMR は、MQPUT 呼び出しおよび MQPUT1 呼び出しのための入出力構造体です。

文字セットとエンコード: MQPMR 内のデータは、**CodedCharSetId** キュー・マネージャー属性で指定した文字セットと ENNAT で指定したローカル・キュー・マネージャーのエンコードで記述されていなければなりません。ただし、アプリケーションが IBM MQ クライアントとして実行されている場合、構造体はクライアントの文字セットとエンコードに従っている必要があります。

使用法: MQPUT 呼び出しまたは MQPUT1 呼び出しにこうした構造体の配列を与えることにより、配布リスト中の宛先キューごとに異なる値を指定できます。入力だけのフィールドもあれば、入出力とも可能なフィールドもあります。

注: この構造体は、固定の配置がないという点で通常とは異なっています。この構造体中のフィールドはオプションであり、各フィールドの有無は MQPMO 中の *PMPRF* フィールドのフラグによって表示されます。ここにあるフィールドは **必ず、次の順序で指定する必要があります**。

- *PRMID*
- *PRCID*
- *PRGID*
- *PRFB*
- *PRACC*

ここにはないフィールドはレコードのスペースをとりません。

MQPMR には固定のレイアウトがないため、COPY ファイルにその定義はありません。したがってアプリケーションのプログラム開発者は、そのアプリケーションに必要なフィールドを含む宣言を作成し、*PMPRF* のフラグを立ててそこにあるフィールドを表示する必要があります。

- [1197 ページの『フィールド』](#)
- [1198 ページの『初期値』](#)
- [1198 ページの『RPG 宣言』](#)

フィールド

MQPMR 構造体には、以下のフィールドが含まれます。フィールドはアルファベット順に説明されています。

PRACC (32 バイトのビット・ストリング)

アカウントリング・トークン。

これはキューに送るメッセージに使用される会計トークンです。このキューの名前は、MQOPEN 呼び出しまたは MQPUT1 呼び出しで与えられた MQOR 構造体の配列内の対応する要素に指定されていた名前です。メッセージ ID は、単一キューに書き込むために MQMD 内の *MDACC* フィールドと同様に処理されます。このフィールドの内容については、[1114 ページの『IBM i での MQMD \(メッセージ記述子\)』](#)の *MDACC* を参照してください。

このフィールドがない場合、MQMD 内の値が使用されます。

これは入力フィールドです。

PRCID (24 バイトのビット・ストリング)

関連 ID。

これは、キューに送るメッセージに使用される関連 ID です。このキューの名前は、MQOPEN 呼び出しまたは MQPUT1 呼び出しで与えられた MQOR 構造体の配列内の対応する要素に指定されていた名前です。メッセージ ID は、単一キューに書き込むために MQMD 内の *MDCID* フィールドと同様に処理されます。

このフィールドが MQPMR レコードにない場合、または宛先と比較して MQPMR レコードが少ない場合、MQMD 内の値は *PRCID* フィールドが入っている MQPMR レコードのない宛先に使用されます。

PMNCID が指定されると、MQPMR レコードの有無にかかわらず、配布リスト中のすべての宛先に新しい単一の関連 ID が生成されます。これは、PMNMID が処理される方法とは異なります (*PRMID* フィールドを参照してください)。

これは入出力フィールドです。

PRFB (10 桁の符号付き整数)

フィードバックまたは理由コード。

これは、キューに送信するメッセージに使用されるフィードバック・コードです。送信対象のキューは、MQOPEN 呼び出しまたは MQPUT1 呼び出しに提供された MQOR 構造体の配列内の対応する要素によって指定された名前を持っているものです。メッセージ ID は、単一キューに書き込むために MQMD 内の *MDFB* フィールドと同様に処理されます。

このフィールドがない場合、MQMD 内の値が使用されます。

これは入力フィールドです。

PRGID (24 バイトのビット・ストリング)

グループ ID。

これは、キューに送るメッセージに使用されるグループ ID です。このキューの名前は、MQOPEN 呼び出しまたは MQPUT1 呼び出しで与えられた MQOR 構造体の配列内の対応する要素に指定されていた名前です。メッセージ ID は、単一キューに書き込むために MQMD 内の *MDGID* フィールドと同様に処理されます。

このフィールドが MQPMR レコードにない場合、または宛先と比較して MQPMR レコードが少ない場合、MQMD 内の値は *PRGID* フィールドが入っている MQPMR レコードのない宛先に使用されます。この値は [1185 ページの表 755](#) に記載されているように処理されますが、以下の点が異なります。

- 新しいグループ ID が使用された場合、キュー・マネージャーは宛先ごとに異なるグループ ID を生成します (つまり、2 つの宛先に同じグループ ID はありません)。
- フィールド内の値が使用された場合、呼び出しは失敗し、理由コード RC2258 が戻ります。

これは入出力フィールドです。

PRMID (24 バイトのビット・ストリング)

メッセージ ID。

これは、キューに送るメッセージに使用されるメッセージ ID です。このキューの名前は、MQOPEN 呼び出しまたは MQPUT1 呼び出し で与えられた MQOR 構造体の配列内の対応する要素に指定されていた名前です。メッセージ ID は、単一キューに書き込むために MQMD 内の *MDMID* フィールドと同様に処理されます。

このフィールドが MQPMR レコードにない場合、または宛先と比較して MQPMR レコードが少ない場合、MQMD 内の値は *PRMID* フィールドが入っている MQPMR レコードのない宛先に使用されます。その値が MINONE の場合、宛先ごとにメッセージ ID が生成されます (つまり、2 つの宛先には同じメッセージ ID はありません)。

PMNMID が指定されると、MQPMR レコードの有無にかかわらず、配布リスト中のすべての宛先に新しいメッセージ ID が生成されます。これは、PMNCID が処理される方法とは異なります (*PRCID* フィールドを参照してください)。

これは入出力フィールドです。

初期値

この構造体には初期値がありません。構造体宣言が提供されていないためです。以下に、すべてのフィールドが必須の場合、アプリケーション・プログラマーが構造体を宣言する方法の例を示します。

RPG 宣言

```
D*.1....:....2....:....3....:....4....:....5....:....6....:....7..  
D* MQPMR Structure  
D*  
D* Message identifier  
D  PRMID              1      24  
D* Correlation identifier  
D  PRCID              25     48  
D* Group identifier  
D  PRGID              49     72  
D* Feedback or reason code  
D  PRFB              73     76I 0  
D* Accounting token  
D  PRACC             77     108
```

IBM i IBM i での MQRFH (規則およびフォーマット・ヘッダー)

MQRFH 構造体は、規則と書式ヘッダーのレイアウトを定義します。

概要

目的: このヘッダーを使用すると、名前と値の組の形式でストリング・データを送信できます。

形式名: FMRFH。

文字セットおよびエンコード: (*RFNVS* を含む) MQRFH 構造体中のフィールドは、文字セットおよび文字のエンコードに従っています。これらは MQRFH の前に来るヘッダー構造体中の *MDCSI* フィールドおよび *MDENC* フィールド、またはアプリケーション・メッセージ・データの先頭に MQRFH がある場合は MQMD 構造体の それらのフィールドによって指定されます。

文字セットは、キュー名に有効な文字用の 1 バイト文字を持つ文字セットでなければなりません。

- [1199 ページの『フィールド』](#)
- [1201 ページの『初期値』](#)
- [1201 ページの『RPG 宣言』](#)

フィールド

MQRFH 構造体には、以下のフィールドが含まれます。フィールドは**アルファベット順**に説明されています。

RFCSI (10 桁の符号付き整数)

RFNVS に続くデータの文字セット ID。

これは、*RFNVS* に続くデータの文字セット ID を指定します。これは、MQRFH 構造体自体の文字データには適用しません。

MQPUT または MQPUT1 呼び出しでは、アプリケーションは、このフィールドをデータに適切な値に設定する必要があります。以下のような特別な値を使用することができます。

CSINHT

この構造体の文字セット ID を継承する。

この構造体の後に続くデータの文字データは、この構造体に設定されているのと同じ文字セットになります。

キュー・マネージャーは、メッセージで送信される構造体の中のこの値を、構造体の実際の文字セット ID に変更します。エラーが発生しない限り、値 CSINHT が MQGET 呼び出しによって返されることはありません。

MQMD の *MDPAT* フィールドの値が ATBRKR の場合、CSINHT は使用できません。

このフィールドの初期値は CSUNDF です。

RFNVS に続くデータの数値エンコード。

これは、*RFNVS* に続くデータの数値エンコードを指定します。これは、MQRFH 構造体自体の数値データには適用しません。

MQPUT または MQPUT1 呼び出しでは、アプリケーションは、このフィールドをデータに適切な値に設定する必要があります。

このフィールドの初期値は ENNAT です。

RFFLG (10 桁の符号付き整数)

フラグ。

以下のように指定できます。

RFNONE

フラグなし。

フィールドの初期値は RFNONE です。

RFFMT (8 バイトの文字ストリング)

RFNVS に続くデータの形式名。

これは、*RFNVS* に続くデータの形式名を指定します。

MQPUT または MQPUT1 呼び出しでは、アプリケーションは、このフィールドをデータに適切な値に設定する必要があります。このフィールドのコーディングの規則は、MQMD の *MDFMT* フィールドの場合と同じです。

このフィールドの初期値は FMNONE です。

RFLEN (10 桁の符号付き整数)

RFNVS を含む MQRFH の全長。

これは、構造体の末尾にある RFNVS フィールドを含む、MQRFH 構造体の長さ (バイト単位) です。長さには、RFNVS フィールドに続くユーザー・データは含まれません。

一部の環境におけるユーザー・データのデータ変換に関する問題を回避するには、RFLEN を 4 の倍数にすることを検討してください。

次の定数は、構造体の固定部分の長さ、つまり RFNVS フィールドを除いた長さを示します。

RFLENV

MQRFH 構造体の長さ。

このフィールドの初期値は RFLENV です。

RFNVS (n バイトの文字ストリング)

名前と値の組を含むストリング。

これは、以下の書式で名前と値の組を含む可変長文字ストリングです。

```
name1 value1 name2 value2 name3 value3 ...
```

各名前および値は、1 つ以上の空白文字で、隣接する名前または値から分離されていなければなりません。これらの空白は意味を持ちません。名前または値では、その名前か値を引用符文字で囲むことによって、有効な空白を含めることができます。開く引用符とそれに対応する閉じる引用符の間のすべての文字は、有効なものとして扱われます。次の例では、名前は FAMOUS_WORDS で、値は Hello World です。

```
FAMOUS_WORDS "Hello World"
```

名前または値には、ヌル文字以外の任意の文字を含めることができます (ヌル文字は RFNVS の区切り文字として機能します)。ただし、インターオペラビリティを支援するため、アプリケーション側で名前を以下の文字に制限できます。

- 先頭文字: 大文字または小文字の英字 (A から Z まで、または a から z まで)、または下線。
- 後続文字: 大文字または小文字の英字、10 進数字 (0 から 9 まで)、下線、ハイフン、またはドット。

名前または値に 1 つ以上の引用符が含まれている場合は、その名前または値を引用符で囲み、ストリング内のそれぞれの引用符を二重にする必要があります。

```
Famous_Words "The program displayed ""Hello World"""
```

名前と値では大文字小文字は区別されます。つまり、小文字は大文字と同じであるとは見なされません。例えば、FAMOUS_WORDS と Famous_Words は 2 つの異なる名前です。

RFNVS のバイト長は、RFLEN - RFLENV に等しくなります。一部の環境におけるユーザー・データのデータ変換に関する問題を回避するには、この長さを 4 の倍数にしてください。RFNVS は、この長さに達するまで空白で埋め込むか、ストリングの最後の文字の次にヌル文字を置いて終了させてください。ヌル文字とそれに続くバイトは、指定された RFNVS の長さまで、無視されます。

注: このフィールドの長さは固定されていないので、このフィールドは、サポートされるプログラミング言語に提供される構造体の宣言から省略されます。

RFSID (4 バイトの文字ストリング)

構造体 ID

値は次のものでなければなりません。

RFSIDV

規則および書式ヘッダー構造体の ID。

このフィールドの初期値は RFSIDV です。

RFVER (10 桁の符号付き整数)

構造体のバージョン番号。

値は次のものでなければなりません。

RFVER1

バージョン 1 の規則および書式ヘッダー構造体。

フィールドの初期値は RFVER1 です。

初期値

表 759. MQRFH のフィールド		
フィールド名	定数の名前	定数の値
RFSID	RFSIDV	'RFH'
RFVER	RFVER1	1
RFLEN	RFLENV	32
RFENC	ENNAT	環境に依存
RFCSI	CSUNDF	0
RFFMT	FMNONE	ブランク
RFFLG	RFNONE	0
注:		
1. 記号-は、単一のブランク文字を表します。		

RPG 宣言

```
D*.1.....2.....3.....4.....5.....6.....7..
D* MQRFH Structure
D*
D* Structure identifier
D RFSID 1 4 INZ('RFH ')
D* Structure version number
D RFVER 5 8I 0 INZ(1)
D* Total length of MQRFH includingNameValueString
D RFLEN 9 12I 0 INZ(32)
D* Numeric encoding of data that followsNameValueString
D RFENC 13 16I 0 INZ(273)
D* Character set identifier of data thatfollows NameValueString
D RFCSI 17 20I 0 INZ(0)
D* Format name of data that followsNameValueString
D RFFMT 21 28 INZ(' ')
D* Flags
D RFFLG 29 32I 0 INZ(0)
```

IBM i IBM i での MQRFH2 (規則および書式ヘッダー 2)

MQRFH2 構造体は、バージョン 2 の規則と書式ヘッダーのフォーマットを定義します。

概要

目的: このヘッダーを使用すると、XML のような構文を使用してエンコードされたデータを送信できます。メッセージには、一連の複数の MQRFH2 構造体を入れることができ、その中の最後の MQRFH2 構造体には、オプションでユーザー・データを続けることができます。

形式名: FMRFH2。

文字セットおよびエンコード: MQRFH2 構造体使用する文字セットとエンコードには、以下の特別な規則が適用されます。

- *RF2NVD* 以外のフィールドは、MQRFH2 の前にあるヘッダー構造体の *MDCSI* および *MDENC* フィールドか、または MQRFH2 がアプリケーション・メッセージ・データの先頭にある場合は MQMD 構造体中の同じフィールドで指定された文字セットとエンコードに従っています。

文字セットは、キュー名に有効な文字用の 1 バイト文字を持つ文字セットでなければなりません。

GMCONV が MQGET 呼び出しで指定されている場合、キュー・マネージャーはこれらのフィールドを、要求された文字セットとエンコードに変換します。

- *RF2NVD* は、*RF2NVC* フィールドで指定された文字セットで表されます。*RF2NVC* に有効なのは特定の Unicode 文字セットのみです (詳細については、*RF2NVC* を参照してください)。

エンコードによって表記が変わる文字セットもあります。*RF2NVC* がそうした文字セットである場合、*RF2NVD* は MQRFH2 内の他のフィールドと同じエンコードである必要があります。

GMCONV が MQGET 呼び出しで指定されている場合、キュー・マネージャーは *RF2NVD* を要求されたエンコードに変換しますが、文字セットは変換しません。

- [1202 ページの『フィールド』](#)
- [1207 ページの『初期値』](#)
- [1207 ページの『RPG 宣言』](#)

フィールド

MQRFH2 構造体には、以下のフィールドが含まれます。フィールドはアルファベット順に説明されています。

RF2CSI (10 桁の符号付き整数)

最後の *RF2NVD* フィールドに続くデータの文字セット ID。

ここでは、最後の *RF2NVD* フィールドの後に続くデータの文字セット ID を指定します。これは MQRFH2 構造体そのものの文字データには適用されません。

MQPUT または MQPUT1 呼び出しでは、アプリケーションは、このフィールドをデータに適切な値に設定する必要があります。以下のような特別な値を使用することができます。

CSINHT

この構造体の文字セット ID を継承する。

この構造体の後に続くデータの文字データは、この構造体に設定されているのと同じ文字セットになります。

キュー・マネージャーは、メッセージで送信される構造体の中のこの値を、構造体の実際の文字セット ID に変更します。エラーが発生しない限り、値 CSINHT が MQGET 呼び出しによって返されることはありません。

MQMD の *MDPAT* フィールドの値が ATBRKR の場合、CSINHT は使用できません。

このフィールドの初期値は CSINHT です。

RF2ENC (10 桁の符号付き整数)

最後の *RF2NVD* フィールドに続くデータの数値エンコード。

ここでは、最後の *RF2NVD* フィールドの後に続くデータの数値エンコードを指定します。これは、MQRFH2 構造体自体の数値データには適用されません。

MQPUT または MQPUT1 呼び出しでは、アプリケーションは、このフィールドをデータに適切な値に設定する必要があります。

このフィールドの初期値は ENNAT です。

RF2FLG (10 桁の符号付き整数)

フラグ。

以下の値を指定する必要があります。

RFNONE

フラグなし。

フィールドの初期値は RFNONE です。

RF2FMT (8 バイトの文字ストリング)

最後の RF2NVD フィールドに続くデータの形式名。

これは、最後の RF2NVD フィールドの後に続くデータの形式名を指定します。

MQPUT または MQPUT1 呼び出しでは、アプリケーションは、このフィールドをデータに適切な値に設定する必要があります。このフィールドのコーディングの規則は、MQMD の MDFMT フィールドの場合と同じです。

このフィールドの初期値は FMNONE です。

RF2LEN (10 桁の符号付き整数)

RF2NVL および RF2NVD フィールドをすべて含む MQRFH2 の合計長。

これは、構造体の終わりにある RF2NVL および RF2NVD フィールドを含む、MQRFH2 構造体の長さ (バイト数) です。構造体の終わりに RF2NVL および RF2NVD フィールドの複数のペアが連続して存在するのは有効です。

```
length1, data1, length2, data2, ...
```

RF2LEN には、構造体の終わりにある最後の RF2NVD フィールドに続くユーザー・データはありません。

一部の環境におけるユーザー・データのデータ変換に関する問題を回避するには、RF2LEN を 4 の倍数にすることを検討してください。

後に続く定数は、構造体の固定部分の長さ、すなわち、RF2NVL および RF2NVD フィールドを除いた長さです。

RFLEN2

MQRFH2 構造体の固定部分の長さ。

このフィールドの初期値は RFLEN2 です。

RF2NVC (10 桁の符号付き整数)

RF2NVD の文字セット ID。

これは、RF2NVD フィールド内のデータのコード化文字セット ID を指定します。これは、MQRFH2 構造体内のその他のストリングの文字セットとは異なります。また、構造体の終わりにある最後の RF2NVD に続くデータの文字セットとも異なる場合があります。

RF2NVC 次のいずれかの CCSID 値を持つ必要があります：

1200

UTF-16 (サポートされる最新バージョンの Unicode)

13488

UTF-16 (Unicode バージョン 2.0 サブセット)

17584

UTF-16 (Unicode バージョン 3.0 サブセット) (ユーロ記号を含む)

1208

UTF-8 (サポートされる最新バージョンの Unicode)

UTF-16 文字セットの場合、*RF2NVD* のエンコード (バイト・オーダー) は、*MQRFH2* 構造体内のその他のフィールドのエンコードと同じである必要があります。代理文字 (X'D800' から X'DFFF') はサポートされていません。

注: *RF2NVC* に前述の値のいずれもない場合、*MQRFH2* 構造体が *MQGET* 呼び出しで変換を要求すると、呼び出しは理由コード RC2111 で完了し、メッセージは変換されずに戻されます。

このフィールドの初期値は 1208 です。

RF2NVD (n バイトの文字ストリング)

名前/値データ。

これは、XML 形式の構文を使ってエンコードされるデータを含む可変長文字ストリングです。このストリングの長さ (バイト単位) は、*RF2NVD* フィールドの前にある *RF2NVL* フィールドによって指定されます。この長さは、4 の倍数である必要があります。

RF2NVL フィールドと *RF2NVD* フィールドはオプションですが、これらのフィールドが存在する場合は、ペアとして隣接している必要があります。フィールドの対は、次のように必要に応じて何回でも繰り返すことができます。

```
length1 data1 length2 data2 length3 data3
```

これらのフィールドはオプションのため、サポートされている種々のプログラム言語用に提供されている構造体の宣言からは省略されています。

RF2NVD は、*GMCONV* オプションを指定してメッセージを取り出すときに、*MQGET* 呼び出しで指定された文字セットに変換されないため、通常とは異なっています。*RF2NVD* は、元の文字セットで記述されたままになります。ただし、*RF2NVD* は、*MQGET* 呼び出しで指定されたエンコード方式には変換されます。

名前/値データの構文: このストリングは、ゼロ個以上のプロパティを含む単一の「フォルダー」からなります。フォルダーは、フォルダーと同じ名前を持つ XML 開始タグと終了タグによって区切られます。

```
<folder> property1 property2 ... </folder>
```

フォルダーの終了タグの後に *RF2NVL* で定義された長さまで続く文字は、空白である必要があります。フォルダー内で、各プロパティは、名前と値、およびオプションのデータ・タイプから構成されます。

```
<name dt="datatype">value</name>
```

これらの例では、

- 区切り文字 (<、=、"、/、および >) は、表示されているとおりに指定する必要があります。
- name* は、プロパティのユーザー指定名です。名前の詳細については、以下の例を参照してください。
- datatype* はプロパティのユーザー指定 (オプション) のデータ・タイプです。有効なデータ・タイプについては、以下の例を参照してください。
- value* は、プロパティのユーザー指定値です。値の詳細については、以下の段落を参照してください。
- 値の前の > 文字と値の後の < 文字の間の空白は意味を持ちます。*dt=* の前には 1 つ以上の空白が必要です。他の場所の空白は、タグの間やタグの前に自由に記述できます (読みやすくするなど)。これらの空白は意味を持ちません。

プロパティが相互に関連している場合、グループと同じ名前を持つ XML 開始タグと終了タグで囲むことによって、それらを一緒にグループ化できます。

```
<folder> <group> property1 property2 ... </group> </folder>
```


グループは、他のグループの中に無制限にネストでき、任意のグループをフォルダー内で 2 回以上記述できます。また、フォルダーで一部のプロパティをグループに含め、他のプロパティを含めなくても構いません。

プロパティ、グループ、およびフォルダーの名前: プロパティ、グループ、およびフォルダーの名前は、有効な XML タグ名でなければなりません。ただし、プロパティ、グループ、およびフォルダーの名前にコロンを含めることはできません。特に、次の点に注意してください。

- 名前は文字か下線で始まらなければなりません。有効な文字は W3C XML 仕様で定義され、基本的に Unicode カテゴリー Ll、Lu、Lo、Lt、および Nl から構成されます。
- 名前の中の残りの文字は、文字、10 進数字、下線、ハイフン、またはドットのどれでも構いません。これらは Unicode カテゴリー Ll、Lu、Lo、Lt、Nl、Mc、Mn、Lm、および Nd に対応します。
- Unicode 互換文字 (X'F900' 以上) は、名前のどの部分でも許可されていません。
- 名前は、ストリング XML (小文字と大文字のすべての組み合わせを含む) で開始することはできません。

さらに、

- 名前は大文字小文字を区別します。例えば、ABC、abc、および Abc は 3 つの異なる名前です。
- フォルダーごとに個別の名前空間があります。その結果、1 つのフォルダー内のグループまたはプロパティは、別のフォルダーにある同じ名前のグループまたはプロパティと競合しません。
- グループおよびプロパティは、フォルダー内の同じ名前空間を占有します。その結果、プロパティはそのプロパティを含むフォルダー内のグループと同じ名前を持つことができません。

一般に、RF2NVD フィールドを解析するプログラムは、そのプログラムで認識できない名前を持つプロパティやグループを (そのプロパティやグループが正しい形式になっている場合は) 無視しなければなりません。

プロパティのデータ・タイプ: 各プロパティは、オプションでデータ・タイプを持つことができます。指定されると、データ・タイプは、大文字、小文字、または大文字小文字混合での次の値のいずれかでなければなりません。

表 760. データ・タイプとその使用法	
データ・タイプ	使用目的
string	文字のシーケンス。エスケープ・シーケンスを使用して、特定の文字を指定する必要があります。
boolean	文字 0 または 1 (1 は TRUE を示します)。
bin.hex	オクテットを表す 16 進数字。
i1	-128 から +127 までの範囲の整数で、10 進数字とオプションの符号だけを使用して表されます。
i2	-32 768 から +32 767 の範囲の整数で、10 進数字とオプションの符号だけを使用して表されます。
i4	-2 147 483 648 から +2 147 483 647 の範囲の整数で、10 進数字とオプションの符号だけを使用して表されます。
i8	-9 223 372 036 854 775 808 から +9 223 372 036 854 775 807 の範囲の整数で、10 進数字とオプションの符号だけを使用して表されます。

表 760. データ・タイプとその使用法 (続き)	
データ・タイプ	使用目的
int	-9 223 372 036 854 775 808 から +9 223 372 036 854 775 807 の範囲の整数で、10 進数字とオプションの符号だけを使用して表されます。送信者が特定の精度を暗示したくない場合、これは i8, i2, i4, または i の代わりに使用 i1 できます。
r4	1.175E-37 から 3.402 823 47E+38 の範囲の大きさの浮動小数点数で、10 進数字、オプションの符号、オプションの小数桁、およびオプションの指数を使用して表されます。
r8	2.225E-307 から 1.797 693 134 862 3E+308 の範囲の大きさの浮動小数点数で、10 進数字、オプションの符号、オプションの小数桁、およびオプションの指数を使用して表されます。

プロパティの値: プロパティの値は、必須のエスケープ・シーケンスが関連付けられている特殊文字を除き、任意の文字で構成できます。以下の表で「必須」と示されている文字の値の場所は、対応するエスケープ・シーケンスで置換しなければなりません。この表には、オプションのエスケープ・シーケンスが関連付けられている文字も示されています。「オプション」と示されている文字の値の場所は、対応するエスケープ・シーケンスで置換できますが、必ずしも置換する必要はありません。

表 761. エスケープ文字とその使用法		
文字	エスケープ・シーケンス	使用法
&	&	必須
<	<	必須
>	>	オプション
"	"	オプション
'	'	オプション

注: エスケープシーケンスの先頭にある文字 & は、に置換されては & ならない。

次の例では、値の中のブランクは有効です。しかし、エスケープ・シーケンスは必要ありません。

```
<Famous_Words>The program displayed "Hello World"</Famous_Words>
```

RF2NVL (10 桁の符号付き整数)

RF2NVD の長さ。

これは、RF2NVD フィールドのデータのバイト長を指定します。RF2NVD フィールドに続くデータ (ある場合) のデータ変換に関する問題を回避するには、RF2NVL を 4 の倍数にしてください。

注: RF2NVL フィールドと RF2NVD フィールドはオプションですが、これらのフィールドが存在する場合は、ペアとして隣接する必要があります。フィールドの対は、次のように必要に応じて何回でも繰り返すことができます。

```
length1 data1 length2 data2 length3 data3
```

これらのフィールドはオプションのため、サポートされている種々のプログラム言語用に提供されている構造体の宣言からは省略されています。

RF2SID (4 バイトの文字ストリング)

構造体 ID

値は次のものでなければなりません。

RFSIDV

規則および書式ヘッダー構造体の ID。

このフィールドの初期値は RFSIDV です。

RF2VER (10 桁の符号付き整数)

構造体のバージョン番号。

値は次のものでなければなりません。

RFVER2

バージョン 2 の規則および書式ヘッダー構造体。

フィールドの初期値は RFVER2 です。

初期値

表 762. MQRFH2 のフィールド		
フィールド名	定数の名前	定数の値
RF2SID	RFSIDV	'RFH-'
RF2VER	RFVER2	2
RF2LEN	RFLEN2	36
RF2ENC	ENNAT	環境に依存
RF2CSI	CSINHT	-2
RF2FMT	FMNONE	ブランク
RF2FLG	RFNONE	0
RF2NVC	なし	1208

注：

1. 記号-は、単一のブランク文字を表します。

RPG 宣言

```
D*..1.....2.....3.....4.....5.....6.....7..
D*
D* MQRFH2 Structure
D*
D* Structure identifier
D RF2SID          1          4      INZ('RFH ')
D* Structure version number
D RF2VER          5          8I 0 INZ(2)
D* Total length of MQRFH2 including allNameValueLength and
D* NameValueDatafields
D RF2LEN          9          12I 0 INZ(36)
D* Numeric encoding of data that followslast NameValueData field
D RF2ENC          13         16I 0 INZ(273)
D* Character set identifier of data thatfollows last NameValueData field
D RF2CSI          17         20I 0 INZ(-2)
D* Format name of data that follows lastNameValueData field
D RF2FMT          21         28      INZ(' ')
D* Flags
D RF2FLG          29         32I 0 INZ(0)
D* Character set identifier ofNameValueData
D RF2NVC          33         36I 0 INZ(1208)
```

MQRMH 構造体は参照メッセージ・ヘッダーの形式を定義します。

概要

目的: このヘッダーは、ユーザー作成のメッセージ・チャンネル出口とともに使用され、ひとつのキュー・マネージャーから別のキュー・マネージャーへの大量のデータ ("バルク・データ" と呼ばれる) を送信します。通常のメッセージングとは異なり、バルク・データはキューに格納されません。キューに格納されるのはバルク・データへの参照のみです。これにより、少数の大規模なメッセージで IBM MQ リソースが使い尽くされる危険性を減らすことができます。

形式名: FMRMH。

文字セットおよびエンコード: MQRMH の文字データ、およびオフセット・フィールドでアドレス指定される文字列は、ローカル・キュー・マネージャーの文字セットで記述されていなければなりません。これは、**CodedCharSetId** キュー・マネージャー属性によって指定されます。MQRMH の数値のデータは、ネイティブ・マシンのエンコードでなければなりません。C プログラミング言語の場合、これは ENNAT の値により指定します。

MQRMH の文字セットおよびエンコードは、以下の *MDCSI* フィールドおよび *MDENC* フィールドに設定しなければなりません。

- MQMD (MQRMH 構造体がメッセージ・データの開始点にある場合)
- MQRMH 構造体に先行するヘッダー構造体 (その他のすべての場合)

使用法: アプリケーションが、MQRMH から構成されるバルク・データを省略したメッセージを書き込みます。伝送キューがメッセージ・チャンネル・エージェント (MCA) によってメッセージを読み込むと、ユーザー定義のメッセージ出口が呼び出されて参照メッセージ・ヘッダーを処理します。この出口は、参照メッセージに MQRMH 構造体によって識別されたバルク・データを追加できます。その後、MCA はそのチャンネルを通じて次のキュー・マネージャーにメッセージを送ります。

受信側には、参照メッセージ待機中のメッセージ出口が必要です。参照メッセージを受け取ると、出口はメッセージ中の MQRMH に続くバルク・データからオブジェクトを作成します。次にバルク・データを除いて参照メッセージを渡します。参照メッセージは、(バルク・データなしで) キューから参照メッセージを読み取るアプリケーションによってあとから取り出すことができます。

通常 MQRMH 構造体は、メッセージ内にあるものがすべてです。ただし、メッセージが伝送キューにある場合は、1 つ以上の追加ヘッダーが MQRMH 構造体の前に付きます。

参照メッセージは、配布リストに送ることもできます。この場合、メッセージが伝送キューにあると、MQDH 構造体およびその関連レコードは MQRMH 構造体の前に付きます。

注: 参照メッセージはセグメント化メッセージとして送信しないでください。メッセージ出口が正常に処理できないためです。

- [1208 ページの『データ変換』](#)
- [1209 ページの『フィールド』](#)
- [1213 ページの『初期値』](#)
- [1214 ページの『RPG 宣言』](#)

データ変換

データ変換を行うため、MQRMH 構造体の変換には、送信側環境データ、送信側オブジェクト名、宛先環境データ、および宛先オブジェクト名の変換が含まれます。構造体の先頭にある *RMLEN* バイト中以外のバイトは、データ変換終了後廃棄されるか、未定義の値が入ります。バルク・データは、以下の記述がすべて該当する場合に変換されます。

- バルク・データがデータ変換実行時にメッセージの中にある。
- MQRMH 内の *RMFMT* フィールドの値が *FMNONE* 以外です。
- ユーザーが作成したデータ変換出口が指定された形式名で存在する。

ただし、メッセージがキューにある場合、通常、そのメッセージにはバルク・データが入らないため、GMCONV オプションではバルク・データは変換されないので注意してください。

フィールド

MQRMH 構造体には、以下のフィールドが含まれます。フィールドはアルファベット順に説明されています。

RMCSI (10 桁の符号付き整数)

バルク・データの文字セット ID。

これは、バルク・データの文字セット ID を指定します。MQRMH 構造体自体の文字データには適用されません。

MQPUT または MQPUT1 呼び出しでは、アプリケーションは、このフィールドをデータに適切な値に設定する必要があります。以下のような特別な値を使用することができます。

CSINHT

この構造体の文字セット ID を継承する。

この構造体の後続くデータの文字データは、この構造体に設定されているのと同じ文字セットになります。

キュー・マネージャーは、メッセージで送信される構造体の中のこの値を、構造体の実際の文字セット ID に変更します。エラーが発生しない限り、値 CSINHT が MQGET 呼び出しによって返されることはありません。

MQMD の MDPAT フィールドの値が ATBRKR の場合、CSINHT は使用できません。

このフィールドの初期値は CSUNDF です。

RMDEL (10 桁の符号付き整数)

宛先環境データの長さ。

このフィールドがゼロの場合、宛先環境データはなく、RMDEO は無視されます。

RMDEO (10 桁の符号付き整数)

宛先環境データのオフセット。

このフィールドは MQRMH 構造体の先頭からの宛先環境データのオフセットを指定します。宛先環境データは、参照メッセージの作成者が認識していれば、その作成者が指定できます。例えば、宛先環境データはバルク・データが格納されるオブジェクトのディレクトリー・パスにできます。ただし、作成者が宛先環境データを認識していない場合は、必要な環境情報を調べるのはユーザーのメッセージ出口の役目です。

宛先環境データの長さはによって与えられる。この長 RMDEL さがゼロの場合、宛先環境データは存在せず、RMDEO は無視される。宛先環境データがある場合は、構造体の先頭から RMLEN バイト以内に完全なデータとして存在する必要があります。

アプリケーションを作成するとき、RMSEO フィールド、RMSNO フィールド、および RMDNO フィールドによってアドレス指定された任意のデータが、宛先環境データと連続していることを前提としないでください。

このフィールドの初期値は 0 です。

RMDL (10 桁の符号付き整数)

バルク・データの長さ。

RMDL フィールドは、MQRMH 構造体によって参照されるバルク・データの長さを指定します。

バルク・データがメッセージ中にある場合、そのデータは MQRMH 構造体の先頭からの RMLEN バイトのオフセットから始まります。メッセージ全体の長さから RMLEN を引いた値が、存在するバルク・データの長さになります。

メッセージにデータがある場合、*RMDL* は関連するデータの量を指定します。通常、*RMDL* は、メッセージに示されているデータの長さと同じ値になります。

MQRMH 構造体が、指定された論理オフセットから開始してオブジェクトに残っているデータを表示する場合、バルク・データがメッセージ中にない場合に限り、*RMDL* の値にゼロを指定できます。

データがない場合は、*MQRMH* の端はメッセージの端に一致します。

このフィールドの初期値は 0 です。

RMDNL (10 桁の符号付き整数)

宛先オブジェクト名の長さ。

このフィールドがゼロの場合、宛先オブジェクト名はなく、*RMDNO* は無視されます。

RMDNO (10 桁の符号付き整数)

宛先オブジェクト名のオフセット。

このフィールドは *MQRMH* 構造体の先頭からの宛先オブジェクト名のオフセットを指定します。宛先オブジェクト名は、参照メッセージの作成者が既知のデータであれば、その作成者が指定できます。ただし、作成者が宛先オブジェクト名を認識していない場合は、作成または変更されるオブジェクトを識別するのはユーザーのメッセージ出口の役目です。

宛先オブジェクト名の長さは、*RMDNL* によって指定されます。この長さがゼロの場合、宛先オブジェクト名はなく、*RMDNO* は無視されます。宛先オブジェクト名が存在する場合は、構造体の先頭から *RMLEN* バイト以内に完全に存在する必要があります。

アプリケーションを作成するとき、*RMSEO* フィールド、*RMSNO* フィールド、および *RMDEO* フィールドによってアドレス指定された任意のデータが、宛先オブジェクト名がと連続していることを前提としないでください。

このフィールドの初期値は 0 です。

RMDO (10 桁の符号付き整数)

バルク・データの低オフセット。

このフィールドは、バルク・データがその一部を形成するオブジェクトの先頭からのバルク・データのオフセットを指定します。オブジェクトの先頭からのバルク・データのオフセットを、論理オフセットと呼びます。これは、*MQRMH* 構造体の先頭からのバルク・データの物理オフセットではありません。物理オフセットは *RMLEN* に指定します。

論理オフセットは、参照メッセージを使用して大きいオブジェクトを送信できるように 2 つに分割されます。実際の論理オフセットは次の 2 つのフィールドの合計によって得られます。

- *RMDO* フィールド。このフィールドは、論理オフセットを 1 000 000 000 で割ったときの余りを表します。したがって、この値の範囲は 0 以上 999 999 999 以下です。
- *RMDO2* フィールド。このフィールドは、論理オフセットを 1 000 000 000 で割ったときの商を表します。したがって、この値は論理オフセット中にある 1 000 000 000 の倍数です。この倍数の範囲は 0 以上 999 999 999 以下です。

このフィールドの初期値は 0 です。

RMDO2 (10 桁の符号付き整数)

バルク・データの高オフセット。

このフィールドは、バルク・データがその一部を形成するオブジェクトの先頭からのバルク・データの高オフセットを指定します。この値の範囲は 0 以上 999 999 999 以下です。詳細は、*RMDO* を参照してください。

このフィールドの初期値は 0 です。

RMENC (10 桁の符号付き整数)

バルク・データの数値エンコード。

これは、バルク・データの数値エンコードを指定します。MQRMH 構造体自体の数値データには適用されません。

MQPUT または MQPUT1 呼び出しでは、アプリケーションは、このフィールドをデータに適切な値に設定する必要があります。

このフィールドの初期値は ENNAT です。

RMFLG (10 桁の符号付き整数)

参照メッセージのフラグ。

以下のフラグが定義されます。

RMLAST

参照メッセージにはオブジェクトの最後の部分が表示されます。

このフラグを指定すると、参照メッセージには参照先オブジェクトの最後の部分が表示されます。

RMNLST

参照メッセージにはオブジェクトの最後の部分は表示されない。

RMNLST は、プログラムの文書化を支援するために定義します。このオプションを他のオプションと組み合わせて使用することは意図されていませんが、値がゼロであるため、そのような使い方をしても検出できません。

このフィールドの初期値は RMNLST です。

RMFMT (8 バイトの文字ストリング)

バルク・データの形式名。

これは、バルク・データの形式名を指定します。

MQPUT または MQPUT1 呼び出しでは、アプリケーションは、このフィールドをデータに適切な値に設定する必要があります。このフィールドのコーディングの規則は、MQMD の *MDFMT* フィールドの場合と同じです。

このフィールドの初期値は FMNONE です。

RMLEN (10 桁の符号付き整数)

MQRMH の全長。固定フィールドの端のストリングを含むものの、バルク・データは含みません。

フィールドの初期値は、0 です。

RMOII (24 バイトのビット・ストリング)

オブジェクトのインスタンス ID。

このフィールドはオブジェクトに特有のインスタンスを識別するのに使用します。このフィールドが必要でない場合は、以下の値に設定してください。

OIINON

指定されたオブジェクトのインスタンス ID がありません。

値は、フィールドの長さについては 2 進ゼロです。

このフィールドの長さは、LNOIID で指定されます。このフィールドの初期値は OIINON です。

RMOT (8 バイトの文字ストリング)

オブジェクト・タイプ

これは、メッセージ出口がサポートする参照メッセージの種類を認識するために使用する名前です。この名前を *RMFMT* フィールドと同じ規則に適合させることを検討してください。

このフィールドの初期値は 8 ブランクです。

RMSEL (10 桁の符号付き整数)

送信側環境データの長さ。

このフィールドがゼロの場合、ソース環境データは存在せず、*RMSEO* は無視されます。

このフィールドの初期値は 0 です。

RMSEO (10 桁の符号付き整数)

送信側環境データのオフセット。

このフィールドは *MQRMH* 構造体の先頭からの発信元環境データのオフセットを指定します。発信元環境データは、参照メッセージの作成者が既知のデータであれば、その作成者が指定できます。例えば、送信側環境データはバルク・データを含むオブジェクトのディレクトリー・パスにすることが可能です。ただし、作成者が送信側環境データを認識していない場合、ユーザーのメッセージ出口が必要な環境情報を調べます。

送信側環境データの長さは *RMSEL* に指定します。この長さがゼロの場合、送信側環境データはなく、*RMSEO* は無視されます。送信側環境データがある場合は、構造体の先頭から *RMLEN* バイト以内に完全なデータとして存在する必要があります。

アプリケーションを作成するとき、環境データが構造体中の最後の固定フィールドの直後から始まることを前提としないでください。また、*RMSNO* フィールド、*RMDEO* フィールド、および *RMDNO* フィールドによってアドレス指定された任意のデータが、環境データと連続していることを前提としないでください。

このフィールドの初期値は 0 です。

RMSID (4 バイトの文字ストリング)

構造体 ID

値は次のものでなければなりません。

RMSIDV

参照メッセージ・ヘッダー構造体の ID。

このフィールドの初期値は *RMSIDV* です。

RMSNL (10 桁の符号付き整数)

送信側オブジェクト名の長さ。

このフィールドがゼロの場合、ソース・オブジェクト名はなく、*RMSNO* は無視されます。

このフィールドの初期値は 0 です。

RMSNO (10 桁の符号付き整数)

送信側オブジェクト名のオフセット。

このフィールドは *MQRMH* 構造体の先頭からの送信側オブジェクト名のオフセットを指定します。送信側オブジェクト名は、参照メッセージの作成者が既知のデータであれば、その作成者が指定できます。ただし、作成者が送信側オブジェクト名を認識していない場合、ユーザーのメッセージ出口がアクセスされるオブジェクトを識別します。

ソース・オブジェクト名の長さは *RMSNL* によって指定されます。この長さがゼロの場合、ソース・オブジェクト名はなく、*RMSNO* は無視されます。ソース・オブジェクト名が存在する場合は、構造体の先頭から *RMLEN* バイト以内に完全に存在する必要があります。

アプリケーションを作成するとき、*RMSEO* フィールド、*RMDEO* フィールド、および *RMDNO* フィールドによってアドレス指定された任意のデータが、送信側オブジェクトの名前と連続していることを前提としないでください。

このフィールドの初期値は 0 です。

RMVER (10 桁の符号付き整数)

構造体のバージョン番号。

値は次のものでなければなりません。

RMVER1

バージョン 1 の参照メッセージ・ヘッダー構造体の ID。

以下の定数は、現行バージョンのバージョン番号を指定しています。

RMVERC

参照メッセージ・ヘッダー構造体の現行バージョン。

このフィールドの初期値は RMVER1 です。

初期値

表 763. MQRMH のフィールド		
フィールド名	定数の名前	定数の値
RMSID	RMSIDV	'RMH␣'
RMVER	RMVER1	1
RMLLEN	なし	0
RMENC	ENNAT	環境に依存
RMCSI	CSUNDF	0
RMFMT	FMNONE	ブランク
RMFLG	RMNLST	0
RMOT	なし	ブランク
RMOII	OIINON	Null
RMSEL	なし	0
RMSEO	なし	0
RMSNL	なし	0
RMSNO	なし	0
RMDEL	なし	0
RMDEO	なし	0
RMDNL	なし	0
RMDNO	なし	0
RMDL	なし	0
RMDO	なし	0
RMDO2	なし	0
注：		
1. 記号␣は、単一のブランク文字を表します。		

```
D*..1.....2.....3.....4.....5.....6.....7..
D*
D* MQRMH Structure
D*
D* Structure identifier
D RMSID          1      4    INZ('RMH ')
D* Structure version number
D RMVER          5      8I 0 INZ(1)
D* Total length of MQRMH, including strings at end of fixed fields, but not
D* the bulk data
D RMLLEN         9     12I 0 INZ(0)
```

```

D* Numeric encoding of bulk data
D RMENC 13 16I 0 INZ(273)
D* Character set identifier of bulkdata
D RMCSI 17 20I 0 INZ(0)
D* Format name of bulk data
D RMFMT 21 28 INZ(' ')
D* Reference message flags
D RMFLG 29 32I 0 INZ(0)
D* Object type
D RMOT 33 40 INZ
D* Object instance identifier
D RMOII 41 64 INZ(X'00000000000000-
D 00000000000000000000-
D 000000000000')
D* Length of source environmentdata
D RMSEL 65 68I 0 INZ(0)
D* Offset of source environmentdata
D RMSEO 69 72I 0 INZ(0)
D* Length of source object name
D RMSNL 73 76I 0 INZ(0)
D* Offset of source object name
D RMSNO 77 80I 0 INZ(0)
D* Length of destination environmentdata
D RMDL 81 84I 0 INZ(0)
D* Offset of destination environmentdata
D RMDEO 85 88I 0 INZ(0)
D* Length of destination objectname
D RMDNL 89 92I 0 INZ(0)
D* Offset of destination objectname
D RMDNO 93 96I 0 INZ(0)
D* Length of bulk data
D RMDL 97 100I 0 INZ(0)
D* Low offset of bulk data
D RMD0 101 104I 0 INZ(0)
D* High offset of bulk data
D RMD02 105 108I 0 INZ(0)

```

RPG 宣言

IBM i IBM i での MQRR (応答レコード)

MQRR 構造体は、宛先が配布リストである場合に、単一宛先キュー用のオープン操作または PUT 操作によって生じる完了コードおよび理由コードを受け取るのに使用します。

概要

目的: MQRR は、MQOPEN 呼び出し、MQPUT 呼び出し、および MQPUT1 呼び出しのための出力構造体です。

文字セットとエンコード: MQRR 内のデータは、**CodedCharSetId** キュー・マネージャー属性で指定した文字セットと ENNAT で指定したローカル・キュー・マネージャーのエンコードで記述されていなければなりません。ただし、アプリケーションが IBM MQ クライアントとして実行されている場合、構造体はクライアントの文字セットとエンコードに従っている必要があります。

使用法: こうした構造体の配列を MQOPEN 呼び出しと MQPUT 呼び出し、または MQPUT1 呼び出しで指定することによって、呼び出しの結果が混在している場合、つまり配布リストの中に呼び出しが成功したキューもあれば失敗したキューもあるという場合に、リストに含まれるすべてのキューについて完了コードおよび理由コードを判別することができます。その呼び出しからの理由コード RC2136 は、(アプリケーションによって与えられた場合) キュー・マネージャーが応答レコードを設定したことを示します。

- [1214 ページの『フィールド』](#)
- [1215 ページの『初期値』](#)
- [1215 ページの『RPG 宣言』](#)

フィールド

MQRR 構造体には、以下のフィールドが含まれます。フィールドは**アルファベット順**に説明されています。

RRCC (10 桁の符号付き整数)

キューの完了コード。

これは、キューに対するオープン操作または PUT 操作の結果生じる完了コードです。このキューの名前は、MQOPEN 呼び出しまたは MQPUT1 呼び出し で与えられた MQOR 構造体の配列内の対応する要素に指定されていた名前です。

これは、常に出力フィールドです。このフィールドの初期値は CCOK です。

RRREA (10 桁の符号付き整数)

キューの理由コード。

これは、キューに対するオープン操作または PUT 操作の結果生じる理由コードです。このキューの名前は、MQOPEN 呼び出しまたは MQPUT1 呼び出し で与えられた MQOR 構造体の配列内の対応する要素に指定されていた名前です。

これは、常に出力フィールドです。このフィールドの初期値は RCNONE です。

初期値

表 764. MQRR のフィールド		
フィールド名	定数の名前	定数の値
RRCC	CCOK	0
RRREA	RCNONE	0

RPG 宣言

```
D*.1.....2.....3.....4.....5.....6.....7..  
D*  
D* MQRR Structure  
D*  
D* Completion code for queue  
D  RRCC              1      4I 0 INZ(0)  
D* Reason code for queue  
D  RRRR              5      8I 0 INZ(0)
```

IBM i IBM i での MQSCO (TLS 構成オプション)

MQSCO 構造体を (MQCD 構造体の TLS フィールドとともに) 使用すると、IBM MQ MQI client として実行されるアプリケーションは、チャンネル・プロトコルが TCP/IP である場合に、クライアント接続の TLS の使用を制御する構成オプションを指定できます。

概要

目的: この構造体は、MQCONN 呼び出しの入力パラメーターです。

クライアント・チャンネルのチャンネル・プロトコルが TCP/IP でない場合、MQSCO 構造体は無視されます。

文字セットとエンコード: MQSCO 内のデータは、**CodedCharSetId** キュー・マネージャー属性で指定された文字セットと、ENNAT で指定されたローカル・キュー・マネージャーのエンコードで記述されていなければなりません。

- [1216 ページの『フィールド』](#)
- [1220 ページの『初期値』](#)
- [1221 ページの『RPG 宣言』](#)

フィールド

MQSCO 構造体には、以下のフィールドが含まれます。フィールドはアルファベット順に説明されています。

SCAIC (10 桁の符号付き整数)

これは、SCAIP または SCAIO フィールドによってアドレス指定された認証情報 (MQAIR) レコードの数です。詳しくは、[1016 ページの『IBM i での MQAIR \(認証情報レコード\)』](#)を参照してください。値はゼロ以上でなければなりません。値が無効の場合、呼び出しは失敗し、理由コード RC2383 が戻されます。

これは入力フィールドです。このフィールドの初期値は 0 です。

SCAIO (10 桁の符号付き整数)

これは、MQSCO 構造体の先頭からの最初の認証情報レコードのオフセットをバイト数で表したものです。オフセットの値は、正負どちらの値にもなります。SCAIC がゼロの場合、このフィールドは無視されます。

SCAIO または SCAIP のいずれかを使用して MQAIR レコードを指定できますが、両方を使用することはできません。詳しくは、SCAIP フィールドの説明を参照してください。

これは入力フィールドです。このフィールドの初期値は 0 です。

SCAIP (10 桁の符号付き整数)

これは、最初の認証情報レコードのアドレスです。SCAIC がゼロの場合、このフィールドは無視されます。

MQAIR レコードの配列を提供するには、次の 2 つの方法があります。

- SCAIP ポインター・フィールドを使用する

この場合、アプリケーションは MQSCO 構造体とは別個の MQAIR レコードの配列を宣言でき、配列のアドレスに SCAIP を設定できます。

他の環境へ移植できる形式のポインター・データ・タイプをサポートするプログラミング言語 (例えば C プログラミング言語など) には、SCAIP の使用を検討してください。

- SCAIO オフセット・フィールドを使用する

この場合、アプリケーションは MQSCO と、その後続く MQAIR レコードの配列を含む、複合構造体を宣言して、SCAIO を MQSCO 構造体の先頭からのその配列の最初のレコードのオフセットに設定する必要があります。この値が正しいこと、および値が MQLONG 内に収まることを確認してください (最も制限の大きいプログラミング言語は COBOL で、有効範囲は -999 999 999 から +999 999 999 です)。

ポインターのデータ・タイプをサポートしていないプログラミング言語や、他の環境に移植できない方式のポインター・データ・タイプをインプリメントしているプログラミング言語 (COBOL プログラミング言語など) の場合には、SCAIO の使用を検討してください。

どちらの方法を選ぶ場合でも、使用できるのは SCAIP および SCAIO のいずれか 1 つだけです。両方がゼロ以外である場合、呼び出しは理由コード RC2384 で失敗します。

これは入力フィールドです。このフィールドの初期値は、ポインターをサポートするプログラミング言語のヌル・ポインターです。それ以外の場合は、すべてヌルのバイトのストリングです。

注: プログラミング言語がポインターのデータ・タイプをサポートしていないプラットフォームでは、このフィールドは適切な長さのバイト・ストリングとして宣言されます。

SCCERLBL (10 桁の符号付き整数)

このフィールドは、使用される証明書ラベルの詳細を示します。

IBM MQ は、SCCERLBL フィールドの値をブランクとして初期化します。必要な値を入力するか、デフォルト値を受け入れてください。

ibmwebspheremquser_id は、この製品のすべてのバージョンでこのフィールドに有効な値です。5.0 未満の MQSCO バージョンでは、これが唯一有効な値です。したがって、このフィールドの値は実行時に解釈され、必要に応じて変更されます。5.0 未満の MQSCO バージョンを指定した場合、または SCCERLBL フィールドのデフォルト値のブランクを受け入れた場合、システムは値 ibmwebspheremquser_id を使用します。

これは入力フィールドです。

SCCERTVPOL (10 桁の符号付き整数)

このフィールドは、どのタイプの証明書妥当性検査ポリシーを使用するかを指定します。このフィールドは、以下のいずれかの値に設定できます。

MQ_CERT_VAL_POLICY_ANY

セキュア・ソケット・ライブラリーでサポートされる各証明書妥当性検査ポリシーを適用します。ポリシーのうちのいずれかにおいて証明書チェーンが有効と見なされる場合、その証明書チェーンを受け入れます。

MQ_CERT_VAL_POLICY_RFC5280

RFC5280 準拠の証明書妥当性検査ポリシーのみ適用します。この設定は、ANY 設定よりも厳密に妥当性検査しますが、一部の旧式のデジタル証明書を拒否します。

このフィールドの初期値は MQ_CERT_VAL_POLICY_ANY です。

SCCH (10 桁の符号付き整数)

このフィールドは、クライアント・システムに接続される暗号ハードウェアの構成の詳細を提供するフィールドです。

このフィールドを以下の形式のストリングに設定するか、ブランクまたはヌルのままにしておきます。

```
GSK_PKCS11=the PKCS #11 driver path and file name;the PKCS #11
token label;the PKCS #11 token password;symmetric cipher setting>;
```

PKCS11 インターフェースに準拠する暗号化ハードウェア (例えば、IBM 4960 または IBM 4963) を使用するには、PKCS11 ドライバー・パス、PKCS11 トークン・ラベル、および PKCS11 トークン・パスワードの各ストリングを指定して、それぞれの終わりにセミコロンを付けます。

PKCS #11 ドライバー・パスは、PKCS #11 カードに対するサポートを提供する共有ライブラリーの絶対パスです。PKCS #11 ドライバー・ファイル名は共有ライブラリーの名前です。PKCS #11 パスおよびファイル名に必要な値の例を以下に示します。

```
/usr/lib/pkcs11/PKCS11_API.so
```

PKCS #11 トークン・ラベルはすべて小文字にする必要があります。ハードウェアを大/小文字混合または大文字のトークン・ラベルで構成した場合は、小文字で構成し直してください。

暗号ハードウェア構成が不要な場合には、このフィールドをブランクまたはヌルにします。

値がフィールドの長さより短い場合、値をヌル文字で終了するか、フィールドの長さまでブランクを埋め込みます。この値が正しくない場合、またはこの値を暗号化ハードウェアを構成するために使用すると失敗する場合、呼び出しは理由コード RC2382 で失敗します。

これは入力フィールドです。このフィールドの長さは LNSSCH によって指定されます。このフィールドの初期値はブランク文字です。

SCEPSUITEB (10 桁の符号付き整数)

このフィールドは、スイート B 準拠の暗号方式が使用されるかどうかと、使用される強度レベルを指定します。値は以下のいずれかです (複数可)。

- SCEPSUITEB0

スイート B 準拠の暗号方式は使用されません。

- SCEPSUITEB1

128 ビットの強度の Suite B セキュリティーを使用します。

- SCEPSUITEB2

192 ビットの強度の Suite B セキュリティーを使用します。

注：このフィールドで SCEPSUITEB0 を他の値と共に使用することは無効です。

SCFR (10 桁の符号付き整数)

IBM MQ 暗号ハードウェアで構成可能であり、使用される暗号モジュールは当該ハードウェア製品が提供するものである。これらは使用中の暗号ハードウェア製品に応じて特定のレベルで認証 FIPS 可能である。

このフィールドを使用して、暗号化機能が提供 IBM MQ ソフトウェアで提供される場合、認証済みアルゴリズムのみ FIPS が使用されることを指定します。

TLSIBM MQ 暗号化の実装がインストールされると、いくつかの FIPS 認証済みモジュールを提供する実装も同時にインストールされます。

値は次のいずれかです。

MQSSL_FIPS_NO

これがデフォルト値です。この値に設定した場合、次のことが保証されます。

- 特定のプラットフォームでサポートされるすべての CipherSpec を使用できます。
- 暗号ハードウェアを使用せずに実行した場合、以下の CipherSpecs は IBM MQ プラットフォーム上で認定暗号 FIPS 140-2 を使用して実行されます：
 - TLS_RSA_WITH_3DES_EDE_CBC_SHA
 - TLS_RSA_WITH_AES_128_CBC_SHA
 - TLS_RSA_WITH_AES_256_CBC_SHA

MQSSL_FIPS_YES

この値に設定した場合、暗号ハードウェアを使って暗号化を実行していない限り、次のことが保証されます。

- このクライアント接続に適用される CipherSpec では、認証済み暗号 FIPS アルゴリズムのみを使用できます。
- インバウンドおよびアウトバウンド TLS チャンネル接続は、CipherSpec として次のいずれかが使用される場合のみ成功します。
 - TLS_RSA_WITH_3DES_EDE_CBC_SHA
 - TLS_RSA_WITH_AES_128_CBC_SHA
 - TLS_RSA_WITH_AES_256_CBC_SHA

注：

1.  CipherSpec TLS_RSA_WITH_3DES_EDE_CBC_SHA は推奨されません。
2. FIPS のみの CipherSpec が構成されている場合、MQI クライアントは FIPS 以外の CipherSpec が指定されている接続を RC2393 で拒否します (可能な場合)。IBM MQ すべてのそのような接続を拒否することを保証するものではなく、設定 IBM MQ が準拠 FIPS しているかどうかを判断するのはあなたの責任です。

SCKEYPWL (10 桁の符号付き整数)

これは TLS キーリポジトリのパスフレーズの長さです。

キーリポジトリのパスフレーズの最大長は 128 文字です。キーリポジトリのパスフレーズが許可される最大長を超える場合、接続は RC2381 エラーで失敗します。

これは入力フィールドです。このフィールドの初期値は 0 です。

SCKEYPWO (10 桁の符号付き整数)

これは TLS キーリポジトリのパスフレーズのオフセット (バイト単位) です。オフセットの値は、正負どちらの値にもなります。

キーリポジトリのパスフレーズを指定するには、`SCKEYPWP -p` または `-r SCKEYPW0` のいずれかを使用できますが、両方同時に使用することはできません。詳細については、フィールド **SCKEYPWP** の説明を参照してください。

これは入力フィールドです。このフィールドの初期値は 0 です。

SCKEYPWP (ポインタ)

これは TLS キーリポジトリのパスフレーズのアドレスです。

これは入力フィールドです。このフィールドの初期値は、ヌル・ポインターです。

キーリポジトリのパスフレーズは、プレーンテキストの文字列として指定することも、`gpssphrase` **runmqicred** ユーティリティを使用して暗号化されたパスフレーズとして指定することもできます。

このフィールドで指定されたキーリポジトリのパスフレーズは、MQKEYRPWD 環境変数またはクライアント設定ファイルの SSL スタンス内のプロパティ SSLKeyRepositoryPassword で指定されたキーリポジトリのパスフレーズを上書きします。

キーリポジトリのパスフレーズを指定するには、`SCKEYPWP -p` または `-r SCKEYPW0` のいずれかを使用できますが、両方同時に使用することはできません。

SCKR (10 桁の符号付き整数)

このフィールドは、鍵と証明書が格納されている鍵データベースファイルの場所を指定します。ファイルの拡張子が指定されていない場合、自動的に という拡張子が **.kdb** 追加されます。

各キーデータベースファイルには、関連付けられたパスワード保管ファイルを持つことができます。このファイルには、鍵データベースへのプログラマチックなアクセスを可能にするために使用される、暗号化されたパスワードが保持されます。パスワード・スタッシュ・ファイルは、キー・データベースと同じディレクトリに存在し、鍵データベースと同じファイル語幹がなければならず、最後にサフィックス **.sth** を付けなければなりません。

例えば、キーデータベースファイルがである場合、パスワード保管 **/xxx/yyy/key.kdb** ファイルはでなければなりません。ここで **xxx** および **yyy/xxx/yyy/key.sth** はディレクトリ名を表す。

キーデータベースのパスワードは、または **SCKEYPW0SCKEYPWP** フィールドを使用して指定することもできます。

値がフィールドの長さより短い場合、値をヌル文字で終了するか、フィールドの長さまでブランクを埋め込みます。値は検査されません。そのため、鍵リポジトリへのアクセスでエラーが発生すると、呼び出しは理由コード **RC2381** で失敗します。

IBM MQ MQI client から TLS 接続を実行するには、**SCKR** に有効な鍵データベース・ファイル名を設定します。

これは入力フィールドです。このフィールドの長さは **LNSSKR** によって指定されます。このフィールドの初期値は、ブランク文字です。

SCSID (10 桁の符号付き整数)

これは構造体 ID です。値は以下のものでなければなりません。

SCSIDV

TLS 構成オプション構造体の ID。

これは常に入力フィールドです。このフィールドの初期値は **SCSIDV** です。

SCVER (10 桁の符号付き整数)

これは構造体のバージョン番号です。値は以下のものでなければなりません。

SCVER1

バージョン 1 の TLS 構成オプション構造体。

SCVER2

バージョン 2 の TLS 構成オプション構造体。

SCVER3

バージョン 3 の TLS 構成オプション構造体。

SCVER4

バージョン 4 の TLS 構成オプション構造体。

SCVER5

バージョン 5 の TLS 構成オプション構造体。

SCVER6

Version-6 TLS 設定オプション構造体。

以下の定数は、現行バージョンのバージョン番号を指定しています。

SCVERC

TLS 構成オプション構造体の現行バージョン。

これは常に入力フィールドです。このフィールドの初期値は SCVER1 です。

初期値

表 765. MQSCO のフィールド		
フィールド名	定数の名前	定数の値
SCSID	SCSIDV	'SC0~'
SCVER	SCVER1	1
SCKR	なし	ヌル・ストリングまたは ブランク
SCCH	なし	ヌル・ストリングまたは ブランク
SCAIC	なし	0
SCAIO	なし	0
SCAIP	なし	ヌル・ポインターまたは ヌル・バイト
SCKRC	なし	ヌル・ポインターまたは ヌル・バイト
SCFR	なし	ヌル・ポインターまたは ヌル・バイト
SCEPSUITEB	なし	ヌル・ポインターまたは ヌル・バイト
SCCERTVPOL	なし	ヌル・ポインターまたは ヌル・バイト
SCCERLBL	なし	ヌル・ポインターまたは ヌル・バイト
SCKEYPWP	なし	ヌル・ポインターまたは ヌル・バイト
SCKEYPWO	なし	0
SCKEYPWL	なし	0

表 765. MQSCO のフィールド (続き)

フィールド名	定数の名前	定数の値
注:		
1. 記号-は、単一のブランク文字を表します。		
2. オプション SCEPSUITEB については、を 1221 ページの『RPG 宣言』 参照してください。		

RPG 宣言

```

D*...1.....2.....3.....4.....5.....6.....7..
D* MQSCO Structure
D*
D* Structure identifier
D  SCSID          1          4    INZ('SCO ')
D* Structure version number
D  SCVER          5          8I 0 INZ(1)
D* Location of TLS key repository
D  SCKR           9         264    INZ
D* Cryptographic hardware configuration string
D  SCCH          265         520    INZ
D* Number of MQAIR records present
D  SCAIC          521         524I 0 INZ(0)
D* Offset of first MQAIR record from start of MQSCO structure
D  SCAIO          525         528I 0 INZ(0)
D* Address of first MQAIR record
D  SCAIP          529         544*   INZ(*NULL)
D* Ver:1 **
D* Number of unencrypted bytes sent/received before secret key is
D* reset
D  SCKRC          545         548I 0 INZ(0)
D* Using FIPS certified algorithms
D  SCFR          549         552I 0 INZ(0)
D* Ver:2 **
* Use only Suite B cryptographic algorithms
D SCEPSUITEB0
D SCEPSUITEB1          553         556I 0 INZ(1)
D SCEPSUITEB2          557         560I 0 INZ(0)
D SCEPSUITEB3          561         564I 0 INZ(0)
D SCEPSUITEB4          565         568I 0 INZ(0)
D SCEPSUITEB          10I 0 DIM(4) OVERLAY(SCEPSUITEB0)
D* Ver:3 **
D* Certificate validation policy
D SCCERTVPOL          569         572I 0 INZ(0)
D* Ver:4 **

```

IBM i IBM i での MQSD (サブスクリプション記述子)

MQSD 構造体を使用して、作成されるサブスクリプションに関する詳細情報を指定します。

概要

目的

この構造体は、MQSUB 呼び出しの入出力パラメーターです。

管理対象サブスクリプション

アプリケーションに、そのサブスクリプションと一致するパブリケーションの宛先として特定のキューを使用する固有の必要がない場合は、管理対象サブスクリプション機能を利用できます。アプリケーションが管理対象サブスクリプションの使用を選ぶと、キュー・マネージャーは、MQSUB 呼び出しからの出力としてオブジェクト・ハンドルを提供して、パブリッシュされるメッセージが送信される宛先をサブスクライバーに通知します。詳しくは、[HOBJ \(10 桁の符号付き整数\) - 入出力](#)を参照してください。

またキュー・マネージャーは、以下の状態で、サブスクリプションの除去時に管理対象宛先から未取り出しのメッセージのクリーンアップに着手します。

- MQCLOSE を CORMSB と共に使用することによりサブスクリプションが除去され、管理対象 Hobj がクローズされた場合。

- 非永続サブスクリプション (SONDUR) を使用しているアプリケーションへの接続が失われた場合に、暗黙的な方法で
- サブスクリプションの除去時に満了していることにより - サブスクリプションが満了していて、管理対象 Hobj がクローズされるため。

管理対象サブスクリプションと非永続サブスクリプションを併用して、クリーンアップを行えるようにすることにより、クローズされた非永続サブスクリプションに関するメッセージによりキュー・マネージャー中のスペースが塞がれないようにする必要があります。永続サブスクリプションも管理対象宛先を使用できます。

文字セットとエンコード

MQSD 内のデータは、**CodedCharSetId** キュー・マネージャー属性で指定した文字セットと ENNAT で指定したローカル・キュー・マネージャーのエンコードで記述されていなければなりません。ただし、アプリケーションが IBM MQ クライアントとして実行されている場合、構造体はクライアントの文字セットとエンコードに従っている必要があります。

- [1222 ページの『フィールド』](#)
- [1235 ページの『初期値』](#)
- [1235 ページの『RPG 宣言』](#)

フィールド

MQSD 構造体には、以下のフィールドが含まれます。フィールドは、アルファベット順にリストされています。

SDAID (32 バイトの文字ストリング)

これは、このサブスクリプションに一致するすべてのパブリケーション・メッセージのメッセージ記述子 (MQMD) の *MDAID* フィールドに入る値です。SDAID は、メッセージ ID コンテキストの一部です。メッセージのコンテキストの詳細については、[メッセージのコンテキスト](#) を参照してください。

MDAID の詳細については、[MDAID](#) を参照してください。

SOSETI オプションを指定しない場合は、デフォルトのコンテキスト情報として、このサブスクリプションに関してパブリッシュされる各メッセージ中で設定される *MDAID* はブランクになります。

SOSETI オプションが指定されている場合、SDAID はユーザーによって生成され、このフィールドは、このサブスクリプションの各パブリケーションに設定される *MDAID* を含む入力フィールドになります。

このフィールドの長さは LNAIDD によって指定されます。このフィールドの初期値は 32 個のブランク文字です。

SOALT オプションを使用して既存のサブスクリプションを変更する場合、将来のパブリケーション・メッセージの *SDAID* は変更できます。

SORES を使用した MQSUB 呼び出しからの戻り時に、このフィールドはサブスクリプションに現在使用されている *MDAID* に設定されます。

SDACC (32 バイトの文字ストリング)

これは、このサブスクリプションに一致するすべてのパブリケーション・メッセージのメッセージ記述子 (MQMD) の *MDACC* フィールドに入る値です。MDACC は、メッセージ ID コンテキストの一部です。メッセージのコンテキストの詳細については、[メッセージのコンテキスト](#) を参照してください。

MDACC の詳細については、[MDACC](#) を参照してください。

SDACC フィールドには、以下の特殊値を使用できます。

ACNONE

アカウンティング・トークンが指定されていません。

値は、フィールドの長さについては 2 進ゼロです。

SOSETI オプションが指定されていない場合、会計トークンはキュー・マネージャーによってデフォルト・コンテキスト情報として生成され、このフィールドは、このサブスクリプションに対してパブリッシュされる各メッセージに設定される **MDACC** を含む出力フィールドになります。

SOSETI オプションが指定されている場合、会計トークンはユーザーによって生成され、このフィールドは、このサブスクリプションの各パブリケーションに設定される **MDACC** を含む入力フィールドになります。

このフィールドの長さは、LNACCT で指定されます。このフィールドの初期値は ACNONE です。

SOALT オプションを使用して既存のサブスクリプションを変更する場合、将来のパブリケーション・メッセージの **MDACC** の値は変更できます。

SORES を使用した MQSUB 呼び出しからの戻り時に、このフィールドはサブスクリプションに現在使用されている **MDACC** に設定されます。

SDASI (40 バイトのビット・ストリング)

これは、適切な許可検査を実行できるようにするために、**SDAU** と共に許可サービスに渡されるセキュリティ ID です。

SDASI は、**SOALTU** が指定されており、かつ **SDAU** フィールドが、最初のヌル文字またはフィールドの終わりまでの全体がブランクでない場合にのみ使用されます。

SORES を使用した MQSUB 呼び出しからの戻り時には、このフィールドは変更されません。

詳しくは、MQOD データ・タイプの **ODASI** の説明を参照してください。

SDAU (12 バイトの文字ストリング)

SOALTU を指定した場合、このフィールドには、現在アプリケーションを実行しているユーザー ID ではなく、サブスクリプションと宛先キュー (MQSUB 呼び出しの **Hobj** パラメーターで指定されている) への出力の許可を検査するために使用される代替ユーザー ID が入れられます。

正常に実行されると、アプリケーションが現在実行されているユーザー ID の代わりに、このフィールドで指定されたユーザー ID がサブスクリプション所有ユーザー ID として記録されます。

SOALTU が指定されて、このフィールドの最初のヌル文字までブランク、またはフィールドの最後まですべてブランクの場合、サブスクリプションが正常に行われるのは、指定のオプションでこのトピックにサブスクライブするために、または出力用宛先キューに、ユーザー許可が必要ない場合のみです。

SOALTU が指定されていない場合、このフィールドは無視されます。

SORES を使用した MQSUB 呼び出しからの戻り時には、このフィールドは変更されません。

これは入力フィールドです。このフィールドの長さは LNUID によって指定されます。このフィールドの初期値は 12 個のブランク文字です。

SDCID (24 バイトのビット・ストリング)

このサブスクリプションと突き合わせるために送信されるすべてのパブリケーションには、メッセージ記述子中にこの関連 ID が含まれます。複数のサブスクリプションが同じキューを使用してパブリケーションを取得する場合、関連 ID で MQGET を使用すると、特定のサブスクリプションに対するパブリケーションのみを取得できます。この関連 ID はキュー・マネージャーまたはユーザーのいずれかによって生成されます。

SOSCID オプションが指定されていない場合、関連 ID はキュー・マネージャーによって生成され、このフィールドは、このサブスクリプションに対してパブリッシュされる各メッセージに設定される関連 ID を含む出力フィールドになります。

SOSCID オプションが指定されている場合、関連 ID はユーザーによって生成され、このフィールドは、このサブスクリプションの各パブリケーションに設定される関連 ID を含む入力フィールドになります。この場合、フィールドの値が CINONE であれば、このサブスクリプションに対してパブリッシュされる各メッセージに設定される関連 ID はメッセージの元の書き込みによって作成された関連 ID です。

SOGRP オプションが指定されており、指定された相関 ID が、同じキューとオーバーラップするトピック・ストリングを使用する既存のグループ化されたサブスクリプションと同じである場合、グループ内で最も有意なサブスクリプションにのみパブリケーションのコピーが提供されます。

このフィールドの長さは、LNCID で指定されます。このフィールドの初期値は CINONE です。

SOALT オプションを使用して既存のサブスクリプションを変更していて、このフィールドが入力フィールドである場合、サブスクリプションが SOGRP オプションを使用して作成されたものでない限り、サブスクリプション相関 ID は変更できます。

SORES を使用した MQSUB 呼び出しからの戻り時には、このフィールドはサブスクリプションの現在の相関 ID に設定されます。

SDEXP (10 桁の符号付き整数)

これは、サブスクリプションの満了後の時間で、10 分の 1 秒単位で表されます。この間隔が渡された後は、パブリケーションはこのサブスクリプションと突き合わせられません。これは、このサブスクライバーへ送信される、パブリケーションの MQMD 内の MDEXP フィールドの値としても使用されます。

以下のような特殊値が認識されます。

EIULIM

サブスクリプションの満了に期限はありません。

SOALT オプションを使用して既存のサブスクリプションを変更する場合、サブスクリプションの満了時間は変更できます。

SORES オプションを使用する MQSUB 呼び出しからの戻り時に、このフィールドは、有効期限の残り時間ではなく、サブスクリプションの元の有効期限に設定されます。

SDON (48 バイトの文字ストリング)

これは、ローカル・キュー・マネージャーに定義されたトピック・オブジェクトの名前です。

この名前には、以下に示す文字を使用できます。

- 英大文字 (A から Z まで)
- 英小文字 (a から z まで)
- 数字 (0 から 9 まで)
- ピリオド (.), スラッシュ (/), 下線 (_), パーセント (%)

名前の先頭をブランクにしたり、名前にブランクを埋め込んだりすることはできませんが、名前の後にブランクを入れることはできます。ヌル文字を使用して、名前の中における有効なデータの末尾を示します。ヌル文字とそれに続く文字はすべてブランクとして扱われます。次の制約事項が適用されます。

- EBCDIC カタカナを使用するシステムでは、小文字を使用できません。
- 英小文字、斜線、パーセントの各文字が含まれている名前をコマンドに指定するときは、それを必ず引用符で囲まなくてはなりません。構造体内のフィールドまたは呼び出しのパラメーターとして指定する名前には、引用符を使用してはなりません。

SDON は、完全トピック名を形成するために使用されます。

完全トピック名は、SDON および SDOS の 2 つのフィールドから作成できます。これらの 2 つのフィールドの使用の詳細については、「[トピック文字列の結合](#)」を参照してください。

SORES オプションを使用した MQSUB 呼び出しからの戻り時には、このフィールドは変更されません。

このフィールドの長さは LNTOPN によって指定されます。このフィールドの初期値は 48 個のブランク文字です。

SDALT オプションを使用して既存のサブスクリプションを変更する場合、サブスクライブ先のトピック・オブジェクトの名前は変更できません。このフィールドおよび SDOS は省略できます。これらの

フィールドが提供される場合は、その解決結果が同じトピック名のフルネームにならなければなりません。そうでない場合、呼び出しは RC2510 で失敗します。

SDOPT (10 桁の符号付き整数)

以下のオプションのうち少なくとも 1 つを指定する必要があります。

- SOALT
- SORES
- SOCRT

値を加えることができます。同じ定数を複数回加算しないでください。表は、これらのオプションの組み合わせ方を示しています。無効な組み合わせには、その旨を示しています。それ以外の組み合わせは有効です。

アクセスまたは作成オプション

アクセスおよび作成オプションは、サブスクリプションを作成するか、または既存のサブスクリプションを返すか、または変更するかを制御します。これらのオプションのうち少なくとも 1 つを指定する必要があります。表は、アクセスまたは作成オプションの有効な組み合わせを表示しています。

表 766. アクセスおよび作成オプションの有効な組み合わせ	
オプションの組み合わせ	注
SOCRT	既存のサブスクリプションがない場合、サブスクリプションを作成し、既に存在する場合には失敗します。
SORES	既存のサブスクリプションを再開します。既存のサブスクリプションがない場合、失敗します。
SOCRT + SORES	サブスクリプションがない場合は作成し、ある場合は一致するものを再開します。数回実行されるアプリケーションで使用する場合に有用な組み合わせです。
SORES + SOALT (注を参照)	MQSD の指定と一致するようフィールドを変更し、既存のサブスクリプションを再開します。既存のサブスクリプションがない場合、失敗します。
SOCRT + SOALT (注を参照)	サブスクリプションがない場合は作成し、ある場合はフィールドを MQSD 中の指定内容と一致するよう変更して、一致するものを再開します。続行する前にサブスクリプションが確実に特定の状態にあるようにしたいアプリケーションで使用するとう有用な組み合わせです。

注:

SOALT を指定するオプションで SORES も指定できますが、この組み合わせは SOALT を単独で指定する場合と比べて追加の効果はありません。SOALT は SORES を暗黙指定します。これは、MQSUB を呼び出してサブスクリプションを変更することが、サブスクリプションの再開をも意味するためです。しかし、その逆は真ではありません。サブスクリプションを再開することは変更を暗黙に示しません。

SOCRT

指定されたトピックに関するサブスクリプションを作成します。同じ SDSN を使用するサブスクリプションが既に存在する場合、呼び出しは失敗して RC2432 が戻ります。この失敗は、SOCRT オプションを SORES と組み合わせることによって回避できます。SDSN は必ずしも必要ではありません。詳しくは、このフィールドの説明を参照してください。

SOCRT を SORES と組み合わせると、まず指定された *SDSN* の既存のサブスクリプションがあるかどうかを検査され、ある場合にはその既存のサブスクリプションへのハンドルが戻されます。既存のサブスクリプションがない場合には、MQSD で指定されたすべてのフィールドを使用して、新規サブスクリプションが作成されます。

SOCRT は、SOALT と組み合わせることができ、類似した効果が得られます (このトピックの以下の SOALT の詳細を参照)。

SORES

SDSN で指定されるサブスクリプションに一致する既存のサブスクリプションへのハンドルを戻します。一致するサブスクリプション属性に対する変更は加えられず、MQSD 構造体中の出力上に戻されます。MQSD の内容のほとんどは使用されません。使用されるフィールドは、*SDSID*、*SDVER*、*SDOPT*、*SDAID*、*SDASI*、および *SDSN* です。

完全サブスクリプション名に一致するサブスクリプションが存在しない場合、呼び出しは失敗し、理由コード RC2428 が戻ります。この失敗は、SOCRT オプションを SORES と組み合わせることによって回避できます。SOCRT の詳細については、[SOCRT](#) を参照してください。

サブスクリプションのユーザー ID は、サブスクリプションを作成したユーザー ID か、またはその後別のユーザー ID によって変更が加えられている場合は、最近正常に変更を加えたユーザー ID です。*SDAID* が使用され、そのユーザーに代替のユーザー ID の使用が許可されている場合、サブスクリプションを作成したユーザー ID の代わりに *SDAID* がサブスクリプションを作成したユーザー ID として記録されます。

SDAU のフィールドが使用され、そのユーザーに代替のユーザー ID の使用が許可されている場合、サブスクリプションを作成したユーザー ID は *SDAU* として記録されます。

SOAUID オプションを使用しないで作成された一致するサブスクリプションが存在し、サブスクリプションのユーザー ID がサブスクリプションへのハンドルを要求しているアプリケーションのユーザー ID と異なる場合、呼び出しは失敗し、理由コード RC2434 が戻ります。

一致するサブスクリプションが存在し、別のアプリケーションによって現在使用されている場合は、呼び出しは失敗し、理由コード RC2429 が戻ります。現在同じ接続によって使用中である場合、呼び出しは失敗せず、サブスクリプションへのハンドルが戻されます。

SubName で指定されているサブスクリプションが、アプリケーションからの再開または変更が有効なサブスクリプションではない場合、呼び出しは失敗し、RC2523 が戻ります。

SORES は SOALT によって暗黙指定されるため、SOALT と組み合わせる必要はありませんが、これら 2 つのオプションを組み合わせてもエラーではありません。

SOALT

SDSN で指定されるものと一致する完全サブスクリプション名を持つ既存のサブスクリプションへのハンドルを戻します。サブスクリプションの属性に MQSD で指定されたものと異なるものがある場合、その属性の変更が禁止されていない限り、サブスクリプション内で変更されます。詳細は、各属性の説明に注記されており、要約は下の表にあります。変更できない属性を変更しようとすると、呼び出しは失敗し、次の表に示す理由コードが戻ります。

完全サブスクリプション名に一致するサブスクリプションが存在しない場合、呼び出しは失敗し、理由コード RC2428 が戻ります。この失敗は、SOCRT オプションを SOALT と組み合わせることによって回避できます。

SOCRT を SOALT と組み合わせると、まず指定された完全サブスクリプション名の既存のサブスクリプションがあるかどうかを検査され、ある場合には、前述したように変更が行われ、その既存のサブスクリプションへのハンドルが戻されます。既存のサブスクリプションがない場合には、MQSD で指定されたすべてのフィールドを使用して、新規サブスクリプションが作成されます。

サブスクリプションのユーザー ID は、サブスクリプションを作成したユーザー ID か、またはその後別のユーザー ID によって変更が加えられている場合は、最近正常に変更を加えたユーザー ID です。*SDAU* が使用されている場合 (かつ、そのユーザーに代替のユーザー ID の使用が許可されてい

る場合)、サブスクリプションを作成したユーザー ID の代わりに、代替のユーザー ID がサブスクリプションを作成したユーザー ID として記録されます。

オプション SOAUID を使用しないで作成された一致するサブスクリプションが存在し、サブスクリプションのユーザー ID がサブスクリプションへのハンドルを要求しているアプリケーションのユーザー ID と異なる場合、呼び出しは失敗し、理由コード RC2434 が戻ります。

一致するサブスクリプションが存在し、別のアプリケーションによって現在使用されている場合は、呼び出しは失敗し、RC2429 が戻ります。現在同じ接続によって使用中である場合、呼び出しは失敗せず、サブスクリプションへのハンドルが戻されます。

SubName で指定されているサブスクリプションが、アプリケーションからの再開または変更が有効なサブスクリプションではない場合、呼び出しは失敗し、RC2523 が戻ります。

以下の表には、SOALT によって変更できるサブスクリプション属性を示します。

表 767. 変更可能な MQSD および MQSUB 中の属性			
データ・タイプ記述子または関数呼び出し	フィールド名	SOALT を使用したこの属性の変更	理由コード
MQSD	耐久性オプション	いいえ	RC2509
MQSD	宛先オプション	はい	なし
MQSD	登録オプション	可 (注釈 1 を参照)	SOGRP を変更しようとした場合、RC2515
MQSD	パブリケーション・オプション	可 (注釈 2 を参照)	なし
MQSD	ワイルドカード・オプション	いいえ	RC2510
MQSD	その他のオプション	不可 (注釈 3 を参照)	なし
MQSD	ObjectName	いいえ	RC2510
MQSD	SDAU	不可 (注釈 4 を参照)	なし
MQSD	SDASI	不可 (注釈 4 を参照)	なし
MQSD	SDEXP	はい	なし
MQSD	SDOS	いいえ	RC2510
MQSD	SDSN	不可 (注釈 5 を参照)	なし
MQSD	SDSUD	はい	なし
MQSD	SDCID	可 (注釈 6 を参照)	グループ化されたサブスクリプション内の場合、RC2515
MQSD	SDPRI	はい	なし
MQSD	SDACC	はい	なし
MQSD	SDAID	はい	なし
MQSD	SDSL	いいえ	RC2512
MQSUB	Hobj	可 (注釈 6 を参照)	グループ化されたサブスクリプション内の場合、RC2515

注：

1. SOGRP は変更できません。
2. SONEWP はサブスクリプションの一部ではないため、変更できません。
3. これらのオプションはサブスクリプションの一部ではありません。
4. この属性はサブスクリプションの一部ではありません。
5. この属性は、変更されるサブスクリプションの ID です。
6. グループ化されたサブスクリプション (SOGRP) の一部である場合を除いて変更可能です。

耐久性オプション: 以下のオプションは、サブスクリプションの耐久性の程度を制御します。これらのオプションのうち 1 つのみ指定できます。SOALT オプションを使用して既存のサブスクリプションを変更する場合、サブスクリプションの耐久性は変更できません。SORES を使用した MQSUB 呼び出しからの戻り時には、適切な耐久性オプションが設定されます。

SODUR

このトピックに対するサブスクリプションが、CORMSB オプションを指定した MQCLOSE を使用して明示的に除去されるまでは、残されるよう要求します。明示的に除去されない場合、キュー・マネージャーへのこのアプリケーションの接続が閉じられた後でもこのサブスクリプションは残ります。

永続サブスクリプションを許可しないように定義されているトピックへ永続サブスクリプションが要求された場合、呼び出しは失敗し、RC2436 が戻ります。

SONDUR

このトピックに対するサブスクリプションがまだ明示的に除去されていない場合は、アプリケーションのキュー・マネージャーに対する接続がクローズされる際に除去されるよう要求します。

SONDUR は、SODUR オプションの反対で、プログラム文書化を支援するために定義されています。いずれも指定されていないときは、これがデフォルト値になります。

宛先オプション: 以下のオプションは、サブスクライブされたトピックに対するパブリケーションが送信される宛先を制御します。SOALT オプションを使用して既存のサブスクリプションを変更する場合、サブスクリプションに対するパブリケーションに使用される宛先は変更できます。SORES を使用した MQSUB 呼び出しからの戻り時に、適切である場合、このオプションは設定されます。

SOMAN

パブリケーションが送信される宛先がキュー・マネージャーによって管理されるように要求します。

HOBJ で戻されるオブジェクト・ハンドルは、キュー・マネージャー管理対象キューを表し、その後の MQGET、MQCB、MQINQ、または MQCLOSE 呼び出しに使用されます。

SOMAN が指定されていない場合、以前の MQSUB 呼び出しから戻されたオブジェクト・ハンドルは **Hobj** パラメーターに指定できません。

登録オプション: 以下のオプションは、キュー・マネージャーに対して行われる、このサブスクリプションに関する登録の詳細を制御します。SOALT オプションを使用して既存のサブスクリプションを変更する場合は、これらの登録オプションを変更できます。SORES を使用した MQSUB 呼び出しからの戻り時には、適切な登録オプションが設定されます。

SOGRP

このサブスクリプションは、同じ SDSL で、同じキューを使用し、同じ相関 ID を指定する他のサブスクリプションと共にグループ化されます。これにより、使用されるトピック・ストリングのセットがオーバーラップするために、サブスクリプションのグループに対して複数のパブリケーション・メッセージが提供されるようなトピックへのパブリケーションがあっても、1 つのメッセージのみがキューに送達されるようになります。このオプションを使用しない場合は、一致する固有の各サブスクリプション (SDSN によって識別される) にパブリケーションのコピーが提供されるので、多数のサブスクリプションによって共用されるキューにパブリケーションの複数のコピーが入れられることがあります。

グループ内で最も有意なサブスクリプションにのみ、パブリケーションのコピーが提供されます。最も有意なサブスクリプションは、ワイルドカードのある位置までトピック名のフルネームに基づ

きます。グループ中でワイルドカードの体系を混合して使用する場合は、ワイルドカードの位置のみ重要になります。同じキューを共有するサブスクリプションのグループ内で異なるワイルドカード方式を組み合わせないことをお勧めします。

新しくグループ化されたサブスクリプションを作成する際にも、固有の *SDSN* がなければなりません。グループ中の既存のサブスクリプションのトピック名のフルネームと一致する場合は、呼び出しは *RC2514* で失敗します。

グループ内で最も有意なサブスクリプションでも *SONOLC* を指定しており、これが同じアプリケーションからのパブリケーションである場合は、キューにパブリケーションが配布されません。

このオプションで作成されたサブスクリプションを変更する場合、グループ化を暗黙指定するフィールド、*MQSUB* 呼び出しの *Hobj* (キューおよびキュー・マネージャー名を表す)、および *SDCID* は変更できません。これらを変更しようとする、呼び出しは失敗し、*RC2515* が戻ります。

このオプションは、*CINONE* に設定されていない *SDCID* と共に *SOSCID* と組み合わせなければならず、*SOMAN* と組み合わせることはできません。

SOAUID

SOAUID を指定した場合、サブスクライバーの ID は単一のユーザー ID に制限されません。そのため、ユーザーは適切な権限を持っていれば、サブスクリプションの変更や再開を行うことができます。サブスクリプションは、一時に単一のユーザーのみが持つことができます。現在別のアプリケーションが使用中であるサブスクリプションの使用を再開しようとする、呼び出しは失敗し、*RC2429* が戻ります。

このオプションを既存のサブスクリプションに追加するには、*SOALT* を使用した *MQSUB* 呼び出しは、元のサブスクリプション自体と同じユーザー ID から行わなければなりません。

MQSUB 呼び出しが *SOAUID* を設定した既存のサブスクリプションを参照し、ユーザー ID が元のサブスクリプションとは異なる場合、このトピックにサブスクライブする権限が新しいユーザー ID にある場合にのみ、呼び出しは成功します。正常終了すると、以後このサブスクライバーに対するパブリケーションは、パブリケーション・メッセージ中に新しいユーザー ID が設定されて、サブスクライバーのキューに書き込まれます。

SOAUID と *SOFUID* を両方とも指定しないでください。どちらも指定されない場合、デフォルトは *SOFUID* です。

SOFUID

SOFUID を指定した場合、サブスクリプションは、最後にサブスクリプションを変更したユーザー ID のみを変更または再開できます。サブスクリプションが変更されていない場合は、サブスクリプションを作成したユーザー ID です。

MQSUB verb が *SOAUID* が設定された既存のサブスクリプションを参照し、*SOALT* を使用してそのサブスクリプションが *SOFUID* を使用するように変更する場合、サブスクリプションのユーザー ID はこの新規ユーザー ID に固定されます。このトピックにサブスクライブする権限が新しいユーザー ID にある場合にのみ、呼び出しは成功します。

サブスクリプションを所有するユーザー ID として記録されている ID 以外のユーザー ID が *SOFUID* サブスクリプションを再開または変更しようとする、呼び出しは失敗し、*RC2434* が戻ります。サブスクリプションの所有ユーザー ID を表示するには、**DISPLAY SBSTATUS** コマンドを使用します。

SOAUID と *SOFUID* を両方とも指定しないでください。どちらも指定されない場合、デフォルトは *SOFUID* です。

パブリケーション・オプション: 以下のオプションは、パブリケーションがこのサブスクライバーに送信される方法を制御します。*SOALT* オプションを使用して既存のサブスクリプションを変更する場合は、これらのパブリケーション・オプションを変更できます。

SONOLC

アプリケーションが独自のパブリケーションを参照しないことを、ブローカーに指示します。接続ハンドルが同じ場合、パブリケーションは同じアプリケーションから発信されたと見なされます。

SORES を使用した MQSUB 呼び出しからの戻り時に、適切である場合、このオプションは設定されます。

SONEWP

このサブスクリプションの作成時に、現在保存されているパブリケーションは送信されません。新しいパブリケーションのみ送信されます。このオプションは、SOCRE が指定されている場合にのみ適用されます。以後のサブスクリプションに対する変更により、パブリケーションのフローは変更されないで、トピック上に保存されているパブリケーションは新しいパブリケーションとしてサブスクライバーにすでに送信されていることになります。

このオプションが SOCRE なしで指定された場合は、呼び出しは失敗し、RC2046 が戻ります。

SORES を使用した MQSUB 呼び出しからの戻り時には、サブスクリプションがこのオプションを使用して作成された場合でも、このオプションは設定されません。

このオプションを使用しないと、以前に保存されたメッセージは、提供された宛先キューに送信されます。このアクションがエラー RC2525 または RC2526 で失敗した場合、サブスクリプションの作成は失敗します。

このオプションは、SOPUBR と組み合わせた場合は無効です。

SOPUBR

このオプションを設定すると、サブスクライバーが必要なときに明確に情報を要求することを示します。キュー・マネージャーは非送信請求メッセージをサブスクライバーに送信しません。前の MQSUB 呼び出しから Hsub ハンドルを使用して MQSUBRQ 呼び出しを行うたびに、保存パブリケーション (トピック中でワイルドカードが指定されている場合は複数のパブリケーションの可能性あり) がサブスクライバーに送信されます。このオプションを使用して MQSUB 呼び出しを行っても、パブリケーションは送信されません。SORES を使用した MQSUB 呼び出しからの戻り時に、適切である場合、このオプションは設定されます。

このオプションは、SONEWP と組み合わせた場合は無効です。

ワイルドカード・オプション: 以下のオプションは、MQSD の SDOS フィールドに指定されるストリングで、ワイルドカードがどのように解釈されるかを制御します。これらのオプションのうち 1 つのみ指定できます。SOALT オプションを使用して既存のサブスクリプションを変更する場合、これらのワイルドカード・オプションは変更できません。SORES を使用した MQSUB 呼び出しからの戻り時には、適切なワイルドカード・オプションが設定されます。

SOWCHR

ワイルドカードは、トピック・ストリング中の文字のみに対して作動します。SOWCHR フィールドは、スラッシュ (/) を、特別な意味のない単なる文字として処理します。

以下の表に、SOWCHR で定義される動作を示します。

表 768. ワイルドカードの解釈	
特殊文字	動作
*	ワイルドカード、ゼロ以上の文字
?	ワイルドカード、1 文字
%	ストリング内で '*'、'? '、または '%' を特殊文字として解釈するのではなく文字として使用できるようにするためのエスケープ文字。例えば '%*', '%?', または '%%' のように使用します。

例えば、以下のトピック上でパブリッシュするとします。

```
/level0/level1/level2/level3/level4
```

このトピックは、以下のトピックを使用するサブスクライバーと一致します。

```
*  
/*  
/ level0/level1/level2/level3/*
```

```
/ level0/level1/* /level3/level4
/ level0/level1/le?el2/level3/level4
```

注: パブリッシュ/サブスクライブに関する MQRFH1 形式のメッセージを使用している場合、このワイルドカードの使用法により、IBM MQ V6 および WebSphere MB V6 で提供される意味が正確に提供されます。この方法は、新しく作成したアプリケーションには使用せず、以前にこのバージョンに対して実行し、SOWTOP で説明されているデフォルトのワイルドカード動作を使用するように変更されていないアプリケーションのみに使用することをお勧めします。

SOWTOP

ワイルドカードは、トピック・ストリング中のトピック・エレメントのみにに対して作動します。デフォルトを選択していない場合は、これがデフォルトの動作になります。

以下の表に、SOWTOP で必要とされる動作を示します。

表 769. ワイルドカードの解釈	
特殊文字	動作
/	トピック・レベルの分離文字
#	ワイルドカード: 複数のトピック・レベル
+	ワイルドカード: 単一のトピック・レベル

注:

1 つのトピック・レベル中で「+」と「#」を他の文字 (これらの文字自体を含む) と混用すると、これらの文字はワイルドカードとして扱われません。以下のストリングでは、「#」および「+」文字は普通の文字として扱われます。

```
level0/level1/#+/level3/level#
```

例えば、以下のトピック上でパブリッシュするとします。

```
/level0/level1/level2/level3/level4
```

このトピックは、以下のトピックを使用するサブスクライバーと一致します。

```
#
/#
/ level0/level1/level2/level3/#
/ level0/level1/+ /level3/level4
```

注: パブリッシュ/サブスクライブに MQRFH2 形式のメッセージを使用している場合、このワイルドカードの使い方は、WebSphere Message Broker 6 で提供される意味になります。

その他のオプション: 以下のオプションは、サブスクリプションではなく API 呼び出しが発行される方法を制御します。SORES を使用した MQSUB 呼び出しからの戻り時には、これらのオプションは変更されません。

SOALTU

SDAU フィールドには、この MQSUB 呼び出しを検証するために使用するユーザー ID が含まれます。アプリケーションを実行しているユーザー ID に対し、指定されたアクセス・オプションでオブジェクトのオープン許可が与えられているかどうかには関係なく、この SDAU にその許可が与えられている場合にのみ、呼び出しは正常に行われます。

SOSCID

サブスクリプションは、SDCID フィールドに指定される相関 ID を使用することになります。このオプションが指定されない場合、相関 ID はサブスクリプション時にキュー・マネージャーが自動的に作成し、SDCID フィールドでアプリケーションに戻します。詳しくは、[SDCID \(24 バイトのビット・ストリング\)](#) を参照してください。

SOSETI

サブスクリプションは、SDACC および SDAID フィールドに指定される会計トークンおよびアプリケーション ID データを使用することになります。

このオプションが指定された場合、宛先キューが 00SETI を使用した MQOPEN 呼び出しでアクセスされた場合と同じ許可検査が行われます。これは、SOMAN オプションも使用されている場合を除きます。この場合には、宛先キューの許可検査は行われません。

このオプションが指定されない場合は、このサブスクライバーへ送信されるパブリケーションには、以下のように、デフォルトのコンテキスト情報が関連付けられます。

表 770. このサブスクライバーに送信されるパブリケーションのデフォルトのコンテキスト情報	
MQMD のフィールド	使用される値
MDUID	サブスクリプションが作成されたときにサブスクリプションに関連付けられたユーザー ID。
MDACC	可能な場合は、環境から判別される。判別できない場合は ACNONE に設定。
MDAID	ブランクに設定される。

このオプションは、SOCRE および SOALT の場合のみ有効です。SORES と併用すると、SDACC および SDAID フィールドは無視されるため、このオプションは無効になります。

以前にサブスクリプションで ID コンテキスト情報が提供された場合に、このオプションを使用しないでそのサブスクリプションを変更すると、変更されたサブスクリプションに関するデフォルトのコンテキスト情報が生成されます。

別のユーザー ID がオプション SOAUID を指定して使用することが認められているサブスクリプションが別のユーザー ID によって再開されると、現在そのサブスクリプションを所有している新しいユーザー ID に対するデフォルトの ID コンテキストが生成され、それ以降はこの新しい ID コンテキストを含むパブリケーションが配信されます。

SOFIQ

MQSUB 呼び出しは、キュー・マネージャーが静止状態になっている場合は失敗します。z/OS では、CICS または IMS アプリケーションについてこのオプションを指定すると、接続が静止状態になっている場合には MQSUB 呼び出しを強制的に失敗させます。

SDAU (12 バイトの文字ストリング)

SOALTU を指定した場合、このフィールドには、現在アプリケーションを実行しているユーザー ID ではなく、サブスクリプションと宛先キュー (MQSUB 呼び出しの **Hobj** パラメーターで指定されている) への出力の許可を検査するために使用される代替ユーザー ID が入れられます。

正常に実行されると、アプリケーションが現在実行されているユーザー ID の代わりに、このフィールドで指定されたユーザー ID がサブスクリプション所有ユーザー ID として記録されます。

SOALTU が指定されて、このフィールドの最初のヌル文字までブランク、またはフィールドの最後まですべてブランクの場合、サブスクリプションが正常に行われるのは、指定のオプションでこのトピックにサブスクライブするために、または出力用宛先キューに、ユーザー許可が必要ない場合のみです。

SOALTU が指定されていない場合、このフィールドは無視されます。

SORES を使用した MQSUB 呼び出しからの戻り時には、このフィールドは変更されません。

これは入力フィールドです。このフィールドの長さは LNUID によって指定されます。このフィールドの初期値は 12 個のブランク文字です。

SDPRI (10 桁の符号付き整数)

これは、このサブスクリプションに一致するすべてのパブリケーション・メッセージのメッセージ記述子 (MQMD) の MQPRI フィールドに入る値です。MQMD 内の MQPRI フィールドについて詳しくは、[MDPRI](#) を参照してください。

値はゼロ以上でなければなりません。ゼロは、最低優先順位です。以下のような特殊値も使用できません。

PRQDEF

MQSUB 呼び出しの Hobj フィールドでサブスクリプション・キューが提供されており、管理対象ハンドルではない場合、メッセージの優先順位はそのキューの **DefPriority** 属性から取られます。そのように識別されたキューがクラスター・キューであるか、またはキュー名の解決パスに定義が 2 つ以上ある場合には、MDPRI の説明のとおり、優先順位はパブリケーション・メッセージがキューに書き込まれるときに決定されます。

MQSUB 呼び出しで管理対象ハンドルを使用した場合、メッセージの優先順位は、サブスクライブするトピックに関連付けられたモデル・キューの **DefPriority** 属性から取得されます。

PRPUB

メッセージの優先順位は、元のパブリケーションの優先順位です。これはフィールドの初期値です。

SOALT オプションを使用して既存のサブスクリプションを変更する場合、将来のパブリケーション・メッセージの *MQPRI* は変更できます。

SORES を使用した MQSUB 呼び出しからの戻り時には、このフィールドはサブスクリプションに現在使用されている優先順位に設定されます。

SDRO (MQCHARV)

SDRO は、キュー・マネージャーが *SDON* に指定された名前を解決した後のロング・オブジェクト名です。

ロング・オブジェクト名が *SDOS* に指定されており、*SDON* には何も指定されていない場合、このフィールドに戻される値は、*SDOS* で指定されている名前と同じです。

このフィールドが省略されている (つまり *SDRO.VSBufSize* がゼロである) 場合、*SDRO* は戻されませんが、長さが *SDRO.VSLength* に戻されます。長さが全体の *SDRO* よりも短い場合、これは切り捨てられ、指定された長さに入る限り右端の文字が最大限戻されます。

SDRO が *MQCHARV* 構造体の使用法の説明にあるとおりに正しく指定されていない場合、または最大長を超過した場合は、呼び出しは失敗し、理由コード *RC2520* が戻ります。

SDSID (4 バイトの文字ストリング)

これは構造体 ID です。値は以下のものでなければなりません。

SDSIDV

サブスクリプション記述子の構造体の ID。

これは常に入力フィールドです。このフィールドの初期値は *SDSIDV* です。

SDSL (10 桁の符号付き整数)

これはサブスクリプションに関連付けられているレベルです。パブリケーションがこのサブスクリプションに配布されるのは、*SDSL* の最高値がパブリケーション時に使用される *PubLevel* 値以下のサブスクリプション・セット中にこのサブスクリプションが含まれている場合のみです。

値は 0 から 9 の範囲でなければなりません。ゼロが最低レベルです。

このフィールドの初期値は 1 です。

SOALT オプションを使用して既存のサブスクリプションを変更する場合、*SDSL* は変更できません。

SDSN (MQCHARV)

SDSN は、サブスクリプション名を指定します。

このフィールドは、*SDOPT* で *SODUR* オプションが指定されている場合にのみ必須ですが、このフィールドの値が提供された場合は、*SONDUR* のキュー・マネージャーによっても使用されます。このフィールドはサブスクリプションを識別するために使用されるため、指定する場合、*SDSN* はキュー・マネージャー内で固有でなければなりません。

SDSN の最大長は 10240 です。

このフィールドは 2 つの目的を果たします。SODUR サブスクリプションの場合、これはサブスクリプションの作成後に、(COKPSB オプションを使用して) サブスクリプションへのハンドルを閉じた場合、またはキュー・マネージャーから切断した場合に、そのサブスクリプションを再開するためにそれを識別する手段です。作成後に除去するサブスクリプションを識別するには、SORES オプションを指定した MQSUB 呼び出しを使用します。SDSN フィールドは、DISPLAY SBSTATUS 内の SDSN フィールドのサブスクリプションの管理ビューにも表示されます。

MQCHARV 構造体の使用方法の説明に従ってが誤って指定 SDSN された場合、または最大長を超えた場合、あるいは必要であるにもかかわらず省略された場合（すなわち SDSN。VCHRL がゼロである）か、または最大長を超えている場合、呼び出しは失敗し、理由コード RC2440 が戻ります。

これは入力フィールドです。この構造体のフィールドの初期値は、MQCHARV 構造体のものと同じです。

SOALT オプションを使用して既存のサブスクリプションを変更する場合、サブスクリプション名はサブスクリプションを識別するのに使用されるフィールドであるため、変更できません。これは、SORES オプションを使用した MQSUB 呼び出しの出力でも変更されません。

SDSS (MQCHARV)

SDSS は、トピックからのメッセージのサブスクライブ時に使用する選択基準を提供するストリングです。

この可変長フィールドは、バッファが提供されており、さらに VSBufSize に正のバッファ長がある場合に、SORES オプションを使用する MQSUB 呼び出しからの出力時に返されます。呼び出しでバッファが提供されていない場合は、選択ストリングの長さだけが MQCHARV の VSLength フィールドで返されます。フィールドを返すのに必要なスペースよりも提供されたバッファが小さい場合、VSBufSize バイトのみがそのバッファに戻されます。

SDSS が MQCHARV 構造体の使用法の説明にあるとおりに正しく指定されていない場合、または最大長を超過した場合は、呼び出しは失敗し、理由コード RC2519 が戻ります。

SDSUD (MQCHARV)

このフィールドでサブスクリプションについて提供されるデータは、このサブスクリプションへ送信される各パブリケーションの mq.SubUserData メッセージ・プロパティとして含まれます。

SDSUD の最大長は 10240 です。

SDSUD が MQCHARV 構造体の使用法の説明にあるとおりに正しく指定されていない場合、または最大長を超過した場合は、呼び出しは失敗し、理由コード RC2431 が戻ります。

これは入力フィールドです。この構造体のフィールドの初期値は、MQCHARV 構造体のものと同じです。

SOALT オプションを使用して既存のサブスクリプションを変更する場合、サブスクリプションのユーザー・データは変更できます。

この可変長フィールドは、バッファが提供されていて、VSBufLen に正のバッファ長が指定されている場合には、SORES オプションを使用する MQSUB 呼び出しからの出力で返されます。呼び出しでバッファが提供されていない場合は、MQCHARV の VCHRL フィールドにサブスクリプション・ユーザー・データの長さのみが戻ります。提供されているバッファがフィールドを返すのに必要なスペースより小さい場合は、提供されているバッファに VSBufLen のバイト数のみ戻されます。

SDVER (10 桁の符号付き整数)

これは構造体のバージョン番号です。値は以下のものでなければなりません。

SDVER1

バージョン 1 のサブスクリプション記述子の構造体。

以下の定数は、現行バージョンのバージョン番号を指定しています。

SDVERC

サブスクリプション記述子の構造体の現行バージョン。

これは常に入力フィールドです。このフィールドの初期値は SDVER1 です。

初期値

表 771. MQSD のフィールド		
フィールド名	定数の名前	定数の値
SDSID	SDSIDV	'SD--'
SDVER	SDVER1	1
SDOPT	SONDUR	0
SDON	なし	ブランク
SDAU	なし	ブランク
SDASI	SINONE	Null
SDEXP	EIULIM	-1
SDOS	MQCHARV 用に定義される名前と値	
SDSN	MQCHARV 用に定義される名前と値	
SDSUD	MQCHARV 用に定義される名前と値	
SDCID	CINONE	Null
SDPRI	PRQDEF	-3
SDACC	ACNONE	Null
SDAID	なし	ブランク
SDSL	なし	1
SDRO	MQCHARV で定義されている名前および値	
注記:		
1. 記号-は、単一のブランク文字を表します。		

RPG 宣言

```
D*.1....:....2....:....3....:....4....:....5....:....6....:....7..  
D* MQSD Structure  
D*  
D* Structure identifier  
D SDSID 1 4  
D* Structure version number  
D SDVER 5 8I 0  
D* Options associated with subscribing  
D SDOPT 9 12I 0  
D* Object name  
D SDON 13 60  
D* Alternate user identifier  
D SDAU 61 72  
D* Alternate security identifier  
D SDASI 73 112  
D* Expiry of Subscription  
D SDEXP 113 116I 0  
D* Object Long name  
D SDOSP 117 132*  
D SDOSO 133 136I 0  
D SDOSS 137 140I 0
```

D SDOSL	141	144I 0
D SDOSC	145	148I 0
D* Subscription name		
D SDSNP	149	164*
D SDSNO	165	168I 0
D SDSNS	169	172I 0
D SDSNL	173	176I 0
D SDSNC	177	180I 0
D* Subscription User data		
D SDSUDP	181	196*
D SDSUDO	197	200I 0
D SDSUDS	201	204I 0
D SDSUDL	205	208I 0
D SDSUDC	209	212I 0
D* Correlation Id related to this subscription		
D SDCID	213	236
D* Priority set in publications		
D SDPRI	237	240I 0
D* Accounting Token set in publications		
D SDACC	241	272
D* Appl Identity Data set in publications		
D SDAID	273	304
D* Message Selector		
D SDSSP	305	320*
D SDSS0	321	324I 0
D SDSSS	325	328I 0
D SDSSL	329	332I 0
D SDSSC	333	336
D* Subscription level		
D SDSL	337	340 0
D* Resolved Long object name		
D SDR0P	341	356*
D SDR00	357	360I 0
D SDR0S	361	364I 0
D SDR0L	365	368I 0
D SDR0C	369	372I 0

IBM i IBM i での MQSMPO (メッセージ・プロパティ設定オプション)

MQSMPO 構造体を使用すると、アプリケーションで、メッセージのプロパティを設定する方法を制御するオプションを指定できます。

概要

目的: この構造体は、**MQSETMP** 呼び出しの入力パラメーターです。

文字セットとエンコード: **MQSMPO** 内のデータは、アプリケーションの文字セットおよびアプリケーションのエンコードでなければなりません (ENNAT)。

- [1236 ページの『フィールド』](#)
- [1238 ページの『初期値』](#)
- [1238 ページの『RPG 宣言』](#)

フィールド

MQSMPO 構造体には、以下のフィールドが含まれます。フィールドは**アルファベット順**に説明されています。

SPOPT (10 桁の符号付き整数)

位置オプション: 以下は、プロパティ・カーソルと比較したプロパティの相対位置に関するオプションです。

SPSETF

指定した名前と一致する最初のプロパティの値を設定します。これが存在しない場合には、階層がこれと一致する他のすべてのプロパティの後に、新しいプロパティを追加します。

SPSETC

プロパティ・カーソルによって指し示されるプロパティの値を設定します。プロパティ・カーソルによってポイントされるプロパティとは、IPINQF または IPINQN オプションを使って最後に照会されたプロパティです。

プロパティ・カーソルは、メッセージ・ハンドルが再使用されるときにリセットされます。あるいは、MQGET 呼び出しの MQGMO 構造体または MQPUT 呼び出しの MQPMO 構造体の HMSG フィールドでメッセージ・ハンドルが指定されている場合にも、リセットされます。

プロパティ・カーソルが未設定の時点で、あるいはプロパティ・カーソルによってポイントされるプロパティが削除された後でこのオプションを使用した場合には、呼び出しが失敗して、完了コード CCFAIL と理由コード RC2471 が戻されます。

SPSETA

プロパティ・カーソルによって指し示されるプロパティの後に新しいプロパティを設定します。プロパティ・カーソルによってポイントされるプロパティとは、IPINQF または IPINQO オプションを使って最後に照会されたプロパティです。

プロパティ・カーソルは、メッセージ・ハンドルが再使用されるときにリセットされます。あるいは、MQGET 呼び出しの MQGMO 構造体または MQPUT 呼び出しの MQPMO 構造体の HMSG フィールドでメッセージ・ハンドルが指定されている場合にも、リセットされます。

プロパティ・カーソルが未設定の時点で、あるいはプロパティ・カーソルによってポイントされるプロパティが削除された後でこのオプションを使用した場合には、呼び出しが失敗して、完了コード CCFAIL と理由コード RC2471 が戻されます。

説明されているオプションを必要としない場合、以下のオプションを使用します。

SPNONE

指定されるオプションはありません。

これは常に入力フィールドです。このフィールドの初期値は SPSETF です。

SPSID (10 桁の符号付き整数)

これは構造体 ID です。値は以下のものでなければなりません。

SPSIDV

メッセージ・プロパティ設定オプション構造の ID。

これは常に入力フィールドです。このフィールドの初期値は **SPSIDV** です。

SPVAKCSI (10 桁の符号付き整数)

値が文字ストリングの場合に設定されるプロパティ値の文字セット。

これは常に入力フィールドです。このフィールドの初期値は **CSAPL** です。

SPVALENC (10 桁の符号付き整数)

値が数値の場合に設定されるプロパティ値のエンコード。

これは常に入力フィールドです。このフィールドの初期値は **ENNAT** です。

SPVER (10 桁の符号付き整数)

これは構造体のバージョン番号です。値は以下のものでなければなりません。

SPVER1

バージョン 1 のメッセージ・プロパティ設定オプション構造。

以下の定数は、現行バージョンのバージョン番号を指定しています。

SPVERC

メッセージ・プロパティ設定オプション構造の現行バージョン。

これは常に入力フィールドです。このフィールドの初期値は **SPVER1** です。

初期値

表 772. MQSMP0 のフィールド		
フィールド名	定数の名前	定数の値
SPSID	SPSIDV	'SMP0'
SPVER	SPVER1	1
SPOPT	SPNONE	0
SPVALENC	ENNAT	環境に依存
SPVALCSI	CSAPL	-3

RPG 宣言

```
D* MQSMP0 Structure
D*
D*
D* Structure identifier
D  SPSID              1      4      INZ('SMP0')
D*
D* Structure version number
D  SPVER              5      8I 0 INZ(1)
D*
D* Options that control the action of
D* MQSETMP
D  SPOPT              9      12I 0 INZ(0)
D*
D* Encoding of Value
D  SPVALENC          13      16I 0 INZ(273)
D*
D* Character set identifier of Value
D  SPVALCSI          17      20I 0 INZ(-3)
```

IBM i IBM i での MQSRO (サブスクリプション要求オプション)

MQSRO 構造体を使用して、サブスクリプションの要求方法を制御するオプションをアプリケーションで指定できます。

概要

目的: この構造体は、MQSUBRQ 呼び出しの入出力パラメーターです。

バージョン: MQSRO の現行バージョンは SRVER1 です。

- [1238 ページの『フィールド』](#)
- [1239 ページの『初期値』](#)
- [1240 ページの『RPG 宣言』](#)

フィールド

MQSRO 構造体には、以下のフィールドが含まれます。フィールドは**アルファベット順**に説明されています。

SRNMP (10 桁の符号付き整数)

これはアプリケーションに戻される出力フィールドで、この呼び出しの結果としてサブスクリプション・キューに送信されるパブリケーションの数を示します。この呼び出しの結果としてこの数のパブリケーションが送信されていますが、これだけ多くのメッセージをアプリケーションが取得できるという保証はありません。非持続メッセージの場合は特にそうです。

サブスクライブ対象のトピックにワイルドカードが含まれていた場合は、パブリケーションが複数ある可能性があります。HSUB で表されるサブスクリプションが作成されたときにトピック・ストリングに

ワイルドカードがなかった場合は、この呼び出しの結果送信されるパブリケーションは多くても 1 つです。

SROPT (10 桁の符号付き整数)

以下のオプションを 1 つ指定する必要があります。オプションは、1 つだけ指定することができます。

その他のオプション: 以下のオプションは、キュー・マネージャーが静止しているときに発生するイベントを制御します。

SRFIQ

MQSUBRQ 呼び出しは、キュー・マネージャーが静止状態にあるときに失敗します。

デフォルト・オプション: 上記で説明されたオプションが必要でない場合、以下のオプションを使用しなければなりません。

SRNONE

この値は、他のオプションが指定されなかったことを示すために使用します。すべてのオプションはデフォルト値であるとみなされます。

SRNONE はプログラムの文書化を支援します。このオプションは、他のオプションと組み合わせて使用するオプションではありませんが、このオプションの値はゼロと等価なので、他のオプションと組み合わせて使用しても、エラーとして検出されることはありません。

SRSID (4 バイトの文字ストリング)

これは構造体 ID です。値は以下のものでなければなりません。

SRSIDV

サブスクリプション要求 SROPT 構造体の ID。

これは常に入力フィールドです。このフィールドの初期値は SRSIDV です。

SRVER (10 桁の符号付き整数)

これは構造体のバージョン番号です。値は以下のものでなければなりません。

SRVER1

バージョン 1 のサブスクリプション要求オプションの構造。

以下の定数は、現行バージョンのバージョン番号を指定しています。

SRVERC

サブスクリプション要求オプションの構造の現行バージョン。

これは常に入力フィールドです。このフィールドの初期値は SRVER1 です。

初期値

表 773. MQSRO のフィールド		
フィールド名	定数の名前	定数の値
SRSID	SRSIDV	'SRO-'
SRVER	SRVER1	1
SROPT	SRNONE	0
SRNMP	なし	0
注:		
1. 記号-は、単一のブランク文字を表します。		
2. ヌル・ストリングまたはブランクの値は、C 言語ではヌル・ストリングを表し、他のプログラミング言語ではブランク文字を表します。		

RPG 宣言

```
D*..1.....2.....3.....4.....5.....6.....7..  
D* MQSRO Structure  
D*  
D* Structure identifier  
D  SRSID          1          4  
D* Structure version number  
D  SRVER          5          8I 0  
D* Options that control the action of MQSUBRQ  
D  SROPT          9         12I 0  
D* Number of publications sent  
D  SRNMP         13         16I 0
```

IBM i IBM i での MQSTS (状況報告構造体)

MQSTS 構造体は、MQSTAT コマンドによって返される状況構造体のデータを記述します。

概要

文字セットおよびエンコード: MQSTS の文字データは、ローカル・キュー・マネージャーの文字セットで記述されます。これは、*CodedCharSetId* キュー・マネージャー属性によって指定されます。MQSTS の数値データはネイティブ・マシンのエンコードで記述され、これは *ENNAT* によって指定されます。

使用法: MQSTAT コマンドは、状況情報を取り出すために使用されます。この情報は MQSTS 構造体に戻れます。MQSTAT について詳しくは、[1370 ページの『IBM i での MQSTAT \(状況情報の取り出し\)』](#)を参照してください。

- [1240 ページの『フィールド』](#)
- [1243 ページの『初期値』](#)
- [1244 ページの『RPG 宣言』](#)

フィールド

MQSTS 構造体には、以下のフィールドが含まれます。フィールドはアルファベット順に説明されています。

STSCC (10 桁の符号付き整数)

これは、MQSTS 構造体に報告される最初のエラーの完了コードです。

これは、常に出力フィールドです。このフィールドの初期値は CCOK です。

STSFC (10 桁の符号付き整数)

これは、失敗した非同期書き込み呼び出しの数です。

これは出力フィールドです。このフィールドの初期値は 0 です。

STSOBJN (48 バイトの文字ストリング)

これは、最初の失敗に関係したオブジェクトのローカル名です。

これは出力フィールドです。このフィールドの初期値は 48 個のブランク文字です。

STSOQMGR (48 バイトの文字ストリング)

これは、*STSOBJN* オブジェクトが定義されているキュー・マネージャーの名前です。最初のヌル文字またはフィールドの終わりまで名前をすべてブランクにすると、アプリケーションが接続されているキュー・マネージャー (ローカル・キュー・マネージャー) を指定したと見なされます。

これは出力フィールドです。このフィールドの初期値は 48 個のブランク文字です。

STS00 (10 桁の符号付き整数)

レポート対象となっているオブジェクトをオープンするのに使用される STS00。現行では、MQSTS のバージョン 2 以上のみ。

STS00 の値は、MQSTAT の **STYPE** パラメーターの値に依存します。

STATAPT

ゼロ。

STATREC

ゼロ。

STATRER

障害が発生したときに使用された STS00。障害の理由は、MQSTS 構造体の STSCC フィールドおよび STSRC フィールドにレポートされます。

STS00 は、出力フィールドです。初期値はゼロです。

STSOS (MQCHARV)

報告対象の失敗オブジェクトの長いオブジェクト名。現行では、MQSTS のバージョン 2 以上のみ。

STSOS は、最大長 10240 の MQCHARV フィールドです。MQCHARV 構造体の使用法についての説明は、[MQCHARV](#) を参照してください。

STSOS の解釈は、MQSTAT の **STYPE** パラメーターの値に依存します。

STATAPT

これは、失敗した MQPUT 操作で使用されたキューまたはトピックの長いオブジェクト名です。

STATREC

長さゼロのストリング

STATRER

これは、再接続が失敗する原因となったオブジェクトの長いオブジェクト名です。

STSOS は、出力フィールドです。その初期値は、長さゼロのストリングです。

STSOT (10 桁の符号付き整数)

ObjectName で名前が指定されているオブジェクトのタイプ。指定可能な値は以下のとおりです。

OTALSQ

別名キュー。

OTLOCQ

ローカル・キュー。

OTMODQ

モデル・キュー。

OTQ

列。

OTREMQ

リモート・キュー。

OTTOP

トピック。

これは、常に出力フィールドです。このフィールドの初期値は OTQ です。

STSRC (10 桁の符号付き整数)

これは、MQSTS 構造体に報告される最初のエラーの理由コードです。

これは、常に出力フィールドです。このフィールドの初期値は RCNONE です。

STSROBJN (48 バイトの文字ストリング)

これは、ローカル・キュー・マネージャーによって名前が解決された後に、*STSROBJN* で指定される宛先キューの名前です。戻される名前は、*STSRQMGR* で示されるキュー・マネージャーに存在するキューの名前です。

非ブランク値は、オブジェクトがブラウズ、入力、または出力 (あるいはこれらの組み合わせ) を目的としてオープンされた単一のキューである場合にだけ戻されます。オープンされたオブジェクトが以下のいずれかである場合、*STSROBJN* はブランクに設定されます。

- トピック
- キューだが、オープンの目的がブラウズ、入力、および出力のいずれでもない

これは出力フィールドです。このフィールドの初期値は 48 個のブランク文字です。

STSRQMGR (48 バイトの文字ストリング)

これは、ローカル・キュー・マネージャーによって名前が解決された後の宛先キュー・マネージャーの名前です。戻される名前は、*STSROBJN* によって識別されるキューを所有するキュー・マネージャーの名前です。*STSRQMGR* は、ローカル・キュー・マネージャーの名前にすることができます。

STSROBJN が、ローカル・キュー・マネージャーが属するキュー共有グループが所有する共有キューである場合、*STSRQMGR* はそのキュー共有グループの名前です。キューが他のキュー共有グループによって所有されている場合、*STSROBJN* は、キュー共有グループの名前またはキュー共有グループのメンバーであるキュー・マネージャーの名前にすることができます (返される値の性質は、ローカル・キュー・マネージャーに存在するキュー定義によって異なります)。

非ブランク値は、オブジェクトがブラウズ、入力、または出力 (あるいはこれらの組み合わせ) を目的としてオープンされた単一のキューである場合にだけ戻されます。オープンされたオブジェクトが以下のいずれかである場合、*STSRQMGR* はブランクに設定されます。

- トピック
- キューだが、オープンの目的がブラウズ、入力、および出力のいずれでもない
- OOBNDN が指定されたクラスター・キュー (**DefBind** キュー属性の値が OOBNDN のときは OOBNDQ が有効なキュー)

これは出力フィールドです。このフィールドの初期値は 48 個のブランク文字です。

STSSC (10 桁の符号付き整数)

これは、成功した非同期書き込み呼び出しの数です。

これは出力フィールドです。このフィールドの初期値は 0 です。

STSSID (4 バイトの文字ストリング)

これは構造体 ID です。値は次のものでなければなりません。

STSSID

状況報告構造体の ID。

このフィールドの初期値は STSSID です。

STSSO (10 桁の符号付き整数)

失敗したサブスクリプションをオープンするのに使用された STSSO。現行では、MQSTS のバージョン 2 以上のみ。

STSSO の解釈は、MQSTAT の **STYPE** パラメーターの値に依存します。

STATAPT

ゼロ。

STATREC

ゼロ。

STATRER

障害が発生したときに使用された STSS0。障害の理由は、MQSTS 構造体の STSCC フィールドおよび STSRC フィールドにレポートされます。障害がトピックへのサブスクリプションに関連していない場合、返される値はゼロです。

STSS0 は、出力フィールドです。初期値はゼロです。

STSSUN (MQCHARV)

失敗しているサブスクリプションの名前。現行では、MQSTS のバージョン 2 以上のみ。

STSSUN は、最大長 10240 の MQCHARV フィールドです。MQCHARV 構造体の使用法についての説明は、[MQCHARV](#) を参照してください。

STSSUN の解釈は、MQSTAT の **STYPE** パラメーターの値に依存します。

STATAPT

長さゼロのストリング。

STATREC

長さゼロのストリング。

STATRER

再接続が失敗する原因となったサブスクリプションの名前。サブスクリプション名が使用可能でない場合、または失敗がサブスクリプションに関連していない場合には、これは長さゼロのストリングです。

STSSUN は、出力フィールドです。その初期値は、長さゼロのストリングです。

STSVER (10 桁の符号付き整数)

これは構造体のバージョン番号です。値は次のものでなければなりません。

STSVR1

状況報告構造体のバージョン番号。

以下の定数は、現行バージョンのバージョン番号を指定しています。

STSVRC

状況報告構造体の現行バージョン。

このフィールドの初期値は STSVR1 です。

STSWC (10 桁の符号付き整数)

これは、警告と共に完了した非同期書き込み呼び出しの数です。

これは出力フィールドです。このフィールドの初期値は 0 です。

初期値

表 774. MQSTS のフィールド		
フィールド名	定数の名前	定数の値
STSSID	STSID	
STSVER	STSVRC	STSVR1
STSCC	CCOK	0
STSRC	RCNONE	0
STSSC	なし	0
STSWC	なし	0

表 774. MQSTS のフィールド (続き)

フィールド名	定数の名前	定数の値
STSF	なし	0
STSOT	なし	0
STSOBJN	なし	ブランク
STSOQMGR	なし	ブランク
STSR	なし	ブランク
STSRQMGR	なし	ブランク
STSOS	MQCHARV 用に定義される名前と値	
STSSUN	MQCHARV 用に定義される名前と値	
STS00	なし	0
STSS0	なし	0

RPG 宣言

```

D*..1.....2.....3.....4.....5.....6.....7..
D* MQSTS Structure
D*
D* Structure identifier
D STSSID 1 4
D* Structure version number
D STSVER 5 8I 0
D* Completion code
D STSCC 9 12I 0
D* Reason code
D STSRC 13 16I 0
D* Success count
D STSSC 17 20I 0
D* Warning count
D STSWC 21 24I 0
D* Failure count
D STSFC 25 28I 0
D* Object type
D STSOT 29 32I 0
D* Object name
D STSOBJN 33 80
D* Object queue manager
D STSOQMGR 81 128
D* Resolved object name
D STSR
```



```

D* Failing open options
D STS00                289    292I 0
D* Failing subscription options
D STSS0                293    296I 0
D* Ver:2 **

```

MQTM - トリガー・メッセージ

MQTM 構造体は、キュー・トリガー・イベントが発生した時に、キュー・マネージャーによりトリガー・モニター・アプリケーションに送信されるトリガー・メッセージ内のデータについて記述します。

概要

目的: この構造体は、IBM MQ トリガー・モニター・インターフェース (TMI) の一部です。TMI は、IBM MQ フレームワーク・インターフェースに含まれています。

形式名: FMTM。

文字セットおよびエンコード: MQTM の文字データは、MQTM を生成するキュー・マネージャーの文字セットにあります。MQTM の数値データは、MQTM を生成するキュー・マネージャーのマシン・エンコードにあります。

MQTM の文字セットおよびエンコードは、以下の *MDCSI* および *MDENC* フィールドで設定されます。

- MQMD (MQTM 構造体がメッセージ・データの開始点にある場合)
- MQTM 構造体に先行するヘッダー構造体 (その他のすべての場合)

使用法: トリガー・モニターのアプリケーションでは、トリガー・メッセージ内の情報の一部またはすべてを、トリガー・モニター・アプリケーションによって開始されるアプリケーションに渡さなければならない場合があります。開始するアプリケーションに必要な情報には、*TMQN*、*TMTD*、および *TMUD* が含まれます。トリガー・モニター・アプリケーションでは、起動したアプリケーションに MQTM 構造体を直接渡すだけでなく、MQTMC2 構造体を渡すこともできます。どちらを渡すかは、起動したアプリケーション側の環境および条件で許可されるもので決まります。MQTMC2 の詳細については、[1249 ページの『IBM iでの MQTMC2 \(トリガー・メッセージ 2 - 文字フォーマット\)』](#)を参照してください。

- IBM i では、IBM MQ が提供するトリガー・モニター・アプリケーションが、MQTMC2 構造体を開始済みアプリケーションに渡します。

トリガーについては、[トリガー操作の前提条件](#)を参照してください。

- [1245 ページの『トリガー・メッセージの MQMD』](#)
- [1246 ページの『フィールド』](#)
- [1248 ページの『初期値』](#)
- [1249 ページの『RPG 宣言』](#)

トリガー・メッセージの MQMD

表 775. キュー・マネージャーによって生成されるトリガー・メッセージの MQMD のフィールドの設定

MQMD のフィールド	使用される値
<i>MDSID</i>	MDSIDV
<i>MDVER</i>	MDVER1
<i>MDREP</i>	RONONE
<i>MDMT</i>	MTDGRM
<i>MDEXP</i>	EIULIM
<i>MDFB</i>	FBNONE
<i>MDENC</i>	ENNAT

表 775. キュー・マネージャーによって生成されるトリガー・メッセージの MQMD のフィールドの設定 (続き)

MQMD のフィールド	使用される値
MDCSI	キュー・マネージャーの CodedCharSetId 属性
MDFMT	FMTM
MDPRI	開始キューの DefPriority 属性
MDPER	PENPER
MDMID	固有の値
MDCID	CINONE
MDBOC	0
MDRQ	ブランク
MDRM	キュー・マネージャーの名前。
MDUID	ブランク
MDACC	ACNONE
MDAID	ブランク
MDPAT	ATQM、またはメッセージ・チャネル・エージェントに対応するもの
MDPAN	キュー・マネージャー名の最初の 28 バイト
MDPD	トリガー・メッセージが送信された日付
MDPT	トリガー・メッセージが送信された時刻
MDAOD	ブランク

トリガー・メッセージを生成するアプリケーションでは、以下のものを除いて、同様の値を設定することをお勧めします。

- **MDPRI** フィールドを **PRQDEF** に設定できます (メッセージが書き込まれる時に、キュー・マネージャーはこの値を開始キューのデフォルトの優先順位に変更します)。
- **MDRM** フィールドをブランクに設定できます (メッセージが書き込まれる時に、キュー・マネージャーはこの値をローカル・キュー・マネージャーの名前に変更します)。
- コンテキスト・フィールドには、アプリケーションに適応する値を設定する必要があります。

フィールド

MQTM 構造体には、以下のフィールドが含まれます。フィールドは**アルファベット順**に説明されています。

TMAI (256 バイトの文字ストリング)

アプリケーション ID。

これは、開始されるアプリケーションを識別する文字ストリングであり、トリガー・メッセージを受け取るトリガー・モニター・アプリケーションで使用されます。キューマネージャはこのフィールドを、フィールド **TMPN** で識別されるプロセスオブジェクトの属性 **App1Id** の値で初期化する。この属性の詳細については [1410 ページの『IBM i でのプロセス定義の属性』](#)を参照のこと。このデータの内容は、キュー・マネージャーにとっては意味のないものです。

TMAI の意味は、トリガー・モニター・アプリケーションによって決まります。IBM MQ によって提供されるトリガー・モニターでは、**TMAI** を実行可能プログラムの名前にすることが必要です。

このフィールドの長さは **LNPROA** によって指定されます。このフィールドの初期値は 256 個のブランク文字です。

TMAT (10 桁の符号付き整数)

アプリケーション・タイプ。

これは、開始するプログラムの性質を識別するもので、トリガー・メッセージを受け取るトリガー・モニター・アプリケーションで使用されます。キューマネージャはこのフィールドを、フィールド *TMPN* で識別されるプロセスオブジェクトの属性 **App1Type** の値で初期化する。この属性の詳細については [1410 ページの『IBM iでのプロセス定義の属性』](#) を参照のこと。このデータの内容は、キュー・マネージャにとっては意味のないものです。

TMAT は、以下の標準値のいずれかにすることができます。ユーザー定義のタイプを使用することもできますが、ATUFST から ATULST の範囲内に値を制限する必要があります。

時刻 CICS

CICS トランザクション。

ATVSE

CICS/VSE トランザクション。

AT400

IBM i アプリケーション。

ATUFST

ユーザー定義のアプリケーション・タイプの最低値。

ATULST

ユーザー定義のアプリケーション・タイプの最高値。

このフィールドの初期値は 0 です。

TMED (128 バイトの文字ストリング)

環境データ。

これは、開始されるアプリケーションに関連する環境関連情報が入っている文字ストリングであり、トリガー・メッセージを受け取るトリガー・モニター・アプリケーションで使用されます。キューマネージャはこのフィールドを、フィールド *TMPN* で識別されるプロセスオブジェクトの属性 **EnvData** の値で初期化する。この属性の詳細については [1410 ページの『IBM iでのプロセス定義の属性』](#) を参照のこと。このデータの内容は、キュー・マネージャにとっては意味のないものです。

このフィールドの長さは LNPROE によって指定されます。このフィールドの初期値は 128 個の空白文字です。

TMPN (48 バイトの文字ストリング)

プロセス・オブジェクトの名前。

これは、起動されたキューに指定されるキュー・マネージャのプロセス・オブジェクトの名前であり、トリガー・メッセージを受け取るトリガー・モニター・アプリケーションで使用します。キューマネージャはこのフィールドを、フィールド *TMQN* で識別されるキューの属性 **ProcessName** 値で初期化する。この属性の詳細については [1380 ページの『キューの属性』](#) を参照のこと。

定義済みフィールド長より短い名前は、常に右側が空白で埋め込まれます。ヌル文字で終了することはありません。

このフィールドの長さは LNPRON によって指定されます。このフィールドの初期値は 48 個の空白文字です。

TMQN (48 バイトの文字ストリング)

起動されたキューの名前。

これは、トリガー・イベントが発生したキューの名前であり、トリガー・モニター・アプリケーションによって開始されたアプリケーションで使用されます。キュー・マネージャは、起動されるキューの **QName** 属性の値でこのフィールドを初期設定します。この属性の詳細については、[1380 ページの『キューの属性』](#) を参照してください。

定義済みフィールド長より短い名前は、右側がブランクで埋め込まれます。ヌル文字で終了することはありません。

このフィールドの長さは LNQN によって指定されます。このフィールドの初期値は 48 個のブランク文字です。

TMSID (4 バイトの文字ストリング)

構造体 ID

値は次のものでなければなりません。

TMSIDV

トリガー・メッセージ構造体の ID。

このフィールドの初期値は TMSIDV です。

TMTD (64 バイトの文字ストリング)

トリガー・データです。

これは、トリガー・メッセージを受け取るトリガー・モニター・アプリケーションで使用する自由形式のデータです。キューマネージャはこのフィールドを、フィールド *TMQN* で識別されるキューの属性 **TriggerData** 値で初期化する。この属性の詳細については [1380 ページの『キューの属性』](#) を参照のこと。このデータの内容は、キュー・マネージャーにとっては意味のないものです。

このフィールドの長さは LNTRGD によって指定されます。このフィールドの初期値は 64 個のブランク文字です。

TMUD (128 バイトの文字ストリング)

ユーザー・データ。

これは、開始されるアプリケーションに関連するユーザー情報が入っている文字ストリングであり、トリガー・メッセージを受け取るトリガー・モニター・アプリケーションで使用されます。キューマネージャはこのフィールドを、フィールド *TMPN* で識別されるプロセスオブジェクトの属性 **UserData** の値で初期化する。この属性の詳細については [1410 ページの『IBM i でのプロセス定義の属性』](#) を参照のこと。このデータの内容は、キュー・マネージャーにとっては意味のないものです。

このフィールドの長さは LNPROU によって指定されます。このフィールドの初期値は 128 個のブランク文字です。

TMVER (10 桁の符号付き整数)

構造体のバージョン番号。

値は次のものでなければなりません。

TMVER1

トリガー・メッセージ構造体のバージョン番号。

以下の定数は、現行バージョンのバージョン番号を指定しています。

TMVERC

トリガー・メッセージ構造体の現行バージョン。

このフィールドの初期値は TMVER1 です。

初期値

表 776. MQTM のフィールド		
フィールド名	定数の名前	定数の値
<i>TMSID</i>	TMSIDV	'TM' 48
<i>TMVER</i>	TMVER1	1
<i>TMQN</i>	なし	ブランク

表 776. MQTM のフィールド (続き)		
フィールド名	定数の名前	定数の値
TMPN	なし	ブランク
TMTD	なし	ブランク
TMAT	なし	0
TMAI	なし	ブランク
TMED	なし	ブランク
TMUD	なし	ブランク
注:		
1. 記号-は、単一のブランク文字を表します。		

RPG 宣言

```

D*.1.....2.....3.....4.....5.....6.....7..
D*
D* MQTM Structure
D*
D* Structure identifier
D TMSID          1          4    INZ('TM ')
D* Structure version number
D TMVER          5          8I 0 INZ(1)
D* Name of triggered queue
D TMQN          9          56    INZ
D* Name of process object
D TMPN         57         104    INZ
D* Trigger data
D TMTD        105         168    INZ
D* Application type
D TMAT        169        172I 0 INZ(0)
D* Application identifier
D TMAI        173         428    INZ
D* Environment data
D TMED        429         556    INZ
D* User data
D TMUD        557         684    INZ

```

IBM i IBM i での MQTMC2 (トリガー・メッセージ 2 - 文字フォーマット)

トリガー・モニター・アプリケーションが、トリガー・メッセージ (MQTM) を開始キューから取り出すとき、トリガー・モニターは、そのトリガー・メッセージ内の一部またはすべての情報を、トリガー・モニターが開始するアプリケーションに渡す必要がある場合もあります。

概要

目的: 開始されるアプリケーションに必要な情報には、*TC2QN*、*TC2TD*、および *TC2UD* があります。トリガー・モニター・アプリケーションでは、起動したアプリケーションに MQTM 構造体を直接渡すだけでなく、MQTMC2 構造体を渡すこともできます。どちらを渡すかは、起動したアプリケーション側の環境および条件で許可されるもので決まります。

この構造体は、IBM MQ トリガー・モニター・インターフェース (TMI) の一部です。TMI は、IBM MQ フレームワーク・インターフェースに含まれています。

文字セットおよびエンコード: MQTMC2 の文字データは、ローカル・キュー・マネージャーの文字セットです。これは、**CodedCharSetId** キュー・マネージャー属性によって指定されます。

使用法: MQTMC2 構造体は、MQTM 構造体の形式に類似しています。相違点は、MQTM 内の非文字フィールドが、MQTMC2 では、同じ長さの文字フィールドに変更される、構造体の終わりにキュー・マネージャー名が追加されることです。

- IBM i では、IBM MQ が提供するトリガー・モニター・アプリケーションが、MQTMC2 構造体を開始済みアプリケーションに渡します。
- [1250 ページの『フィールド』](#)
- [1251 ページの『初期値』](#)
- [1251 ページの『RPG 宣言』](#)

フィールド

MQTMC2 構造体には、以下のフィールドが含まれます。フィールドはアルファベット順に説明されています。

TC2AI (256 バイトの文字ストリング)

アプリケーション ID。

MQTM 構造体の *TMAI* フィールドを参照してください。

TC2AT (4 バイトの文字ストリング)

アプリケーション・タイプ。

このフィールドには、元のトリガー・メッセージの MQTM 構造体の *TMAT* フィールドの値に関係なく、常にブランクが入ります。

TC2ED (128 バイトの文字ストリング)

環境データ。

MQTM 構造体の *TMED* フィールドを参照してください。

TC2PN (48 バイトの文字ストリング)

プロセス・オブジェクトの名前。

MQTM 構造体の *TPMN* フィールドを参照してください。

TC2QMN (48 バイトの文字ストリング)

キュー・マネージャー名。

これは、トリガー・イベントが発生したキュー・マネージャーの名前です。

TC2QN (48 バイトの文字ストリング)

起動されたキューの名前。

MQTM 構造体の *TMQN* フィールドを参照してください。

TC2SID (4 バイトの文字ストリング)

構造体 ID

値は次のものでなければなりません。

TCSIDV

トリガー・メッセージ (文字形式) 構造の ID。

TC2TD (64 バイトの文字ストリング)

トリガー・データです。

MQTM 構造体の *TMTD* フィールドを参照してください。

TC2UD (128 バイトの文字ストリング)

ユーザー・データ。

MQTM 構造体の *TMUD* フィールドを参照してください。

TC2VER (4 バイトの文字ストリング)

構造体のバージョン番号。

値は次のものでなければなりません。

TCVER2

バージョン 2 トリガー・メッセージ (文字形式) 構造体。

以下の定数は、現行バージョンのバージョン番号を指定しています。

TCVERC

トリガー・メッセージ (文字形式) 構造体の現行バージョン。

初期値

表 777. MQTMC2 のフィールド		
フィールド名	定数の名前	定数の値
TC2SID	TCSIDV	'TMC'
TC2VER	TCVER2	'2'
TC2QN	なし	ブランク
TC2PN	なし	ブランク
TC2TD	なし	ブランク
TC2AT	なし	ブランク
TC2AI	なし	ブランク
TC2ED	なし	ブランク
TC2UD	なし	ブランク
TC2QMN	なし	ブランク
注:		
1. 記号'は、単一のブランク文字を表します。		

RPG 宣言

```
D*.1.....2.....3.....4.....5.....6.....7..
D* MQTMC2 Structure
D*
D* Structure identifier
D TC2SID 1 4
D* Structure version number
D TC2VER 5 8
D* Name of triggered queue
D TC2QN 9 56
D* Name of process object
D TC2PN 57 104
D* Trigger data
D TC2TD 105 168
D* Application type
D TC2AT 169 172
D* Application identifier
D TC2AI 173 428
D* Environment data
D TC2ED 429 556
D* User data
D TC2UD 557 684
D* Queue manager name
D TC2QMN 685 732
```


MQWIH 構造体は、z/OS ワークロード・マネージャーで処理するメッセージの最初に組み込む必要のある情報を記述します。

概要

形式名: FMWIH。

文字セットおよびエンコード: MQWIH 構造体中のフィールドは、文字セットおよび文字のエンコードに従っています。これらは MQWIH の前に来るヘッダー構造体中の *MDCSI* フィールドおよび *MDENC* フィールド、またはアプリケーション・メッセージ・データの先頭に MQWIH がある場合は MQMD 構造体のそれらのフィールドによって指定されます。

文字セットは、キュー名に有効な文字用の 1 バイト文字を持つ文字セットでなければなりません。

使用法: メッセージが z/OS ワークロード・マネージャーによって処理される場合、メッセージは MQWIH 構造体で開始しなければなりません。

- [1252 ページの『フィールド』](#)
- [1254 ページの『初期値』](#)
- [1254 ページの『RPG 宣言』](#)

フィールド

MQWIH 構造体には、以下のフィールドが含まれます。フィールドは**アルファベット順**に説明されています。

WICSI (10 桁の符号付き整数)

MQWIH に続くデータの文字セット ID。

これは、MQWIH の後に続くデータの文字セット ID を指定します。MQWIH 構造体自体の文字データには適用されません。

MQPUT または MQPUT1 呼び出しでは、アプリケーションは、このフィールドをデータに適切な値に設定する必要があります。以下のような特別な値を使用することができます。

CSINHT

この構造体の文字セット ID を継承する。

この構造体の後に続くデータの文字データは、この構造体に設定されているのと同じ文字セットになります。

キュー・マネージャーは、メッセージで送信される構造体の中のこの値を、構造体の実際の文字セット ID に変更します。エラーが発生しない限り、値 CSINHT が MQGET 呼び出しによって返されることはありません。

MQMD の *MDPAT* フィールドの値が ATBRKR の場合、CSINHT は使用できません。

このフィールドの初期値は CSUNDF です。

WIENC (10 桁の符号付き整数)

MQWIH に続くデータの数値エンコード。

これは、MQWIH の後に続くデータの数値エンコードを指定します。MQWIH 構造体自体の数値データには適用されません。

MQPUT または MQPUT1 呼び出しでは、アプリケーションは、このフィールドをデータに適切な値に設定する必要があります。

このフィールドの初期値は 0 です。

WIFLG (10 桁の符号付き整数)

フラグ

値は次のものでなければなりません。

WINONE

フラグなし。

このフィールドの初期値は WINONE です。

WIFMT (8 バイトの文字ストリング)

MQWIIH に続くデータの形式名。

これは、MQWIIH 構造体の後に続くデータの形式名を指定します。

MQPUT または MQPUT1 呼び出しでは、アプリケーションは、このフィールドをデータに適切な値に設定する必要があります。このフィールドのコーディングの規則は、MQMD の *MDFMT* フィールドの場合と同じです。

このフィールドの長さは LNFMT によって指定されます。このフィールドの初期値は FMNONE です。

WILEN (10 桁の符号付き整数)

MQWIIH 構造体の長さ。

値は次のものでなければなりません。

WILEN1

バージョン 1 の作業情報ヘッダー構造体の長さ。

以下の定数は、現行バージョンの長さを指定しています。

WILENC

現行バージョンの作業情報ヘッダー構造体の長さ。

このフィールドの初期値は WILEN1 です。

WIRSV (32 バイトの文字ストリング)

予約済み。

これは予約フィールドです。フィールドはブランクでなければなりません。

WISID (4 バイトの文字ストリング)

構造体 ID

値は次のものでなければなりません。

WISIDV

作業情報ヘッダー構造体の ID。

このフィールドの初期値は WISIDV です。

WISNM (32 バイトの文字ストリング)

サービス名。

これは、メッセージを処理するサービスの名前です。

このフィールドの長さは、LNSVNM で指定します。このフィールドの初期値は 32 個のブランク文字です。

WISST (8 バイトの文字ストリング)

サービス・ステップ名。

これは、メッセージが関連する *WISNM* のステップの名前です。

このフィールドの長さは、LNSVST で指定します。このフィールドの初期値は 8 個のブランク文字です。

WITOK (16 バイトのビット・ストリング)

メッセージ・トークン。

これは、メッセージを一意に識別するメッセージ・トークンです。

MQPUT および MQPUT1 呼び出しでは、このフィールドは無視されます。このフィールドの長さは LNMTOK によって指定されます。このフィールドの初期値は MTKNON です。

WIVER (10 桁の符号付き整数)

構造体のバージョン番号。

値は次のものでなければなりません。

WIVER1

バージョン 1 の作業情報ヘッダー構造体。

以下の定数は、現行バージョンのバージョン番号を指定しています。

WIVERC

現行バージョンの作業情報ヘッダー構造体。

このフィールドの初期値は WIVER1 です。

初期値

表 778. MQWIH のフィールド		
フィールド名	定数の名前	定数の値
WISID	WISIDV	'WIH~'
WIVER	WIVER1	1
WILEN	WILEN1	120
WIENC	なし	0
WICSI	CSUNDF	0
WIFMT	FMNONE	ブランク
WIFLG	WINONE	0
WISNM	なし	ブランク
WISST	なし	ブランク
WITOK	MTKNON	Null
WIRSV	なし	ブランク
注:		
1. 記号~は、単一のブランク文字を表します。		

RPG 宣言

```
D*.1.....2.....3.....4.....5.....6.....7..
D*
D* MQWIH Structure
D*
D* Structure identifier
D  WISID          1      4      INZ('WIH ')
D* Structure version number
D  WIVER          5      8I 0 INZ(1)
D* Length of MQWIH structure
D  WILEN          9     12I 0 INZ(120)
D* Numeric encoding of data that followsMQWIH
D  WIENC         13     16I 0 INZ(0)
D* Character-set identifier of data thatfollows MQWIH
D  WICSI         17     20I 0 INZ(0)
D* Format name of data that followsMQWIH
D  WIFMT         21     28      INZ('      ')
```

D* Flags			
D WIFLG	29	32I 0	INZ(0)
D* Service name			
D WISNM	33	64	INZ
D* Service step name			
D WISST	65	72	INZ
D* Message token			
D WITOK	73	88	INZ(X'0000000000000000-0000000000000000')
D			
D* Reserved			
D WIRSV	89	120	INZ

IBM i IBM i での MQXQH (伝送キュー・ヘッダー)

MQXQH 構造体は、伝送キューに入っているメッセージのアプリケーション・メッセージ・データの接頭部に付けられる情報を記述します。

概要

目的: 伝送キューは、特殊なタイプのローカル・キューで、リモート・キューに宛先指定された (つまり、ローカル・キュー・マネージャーに属さないキューに宛先指定された) メッセージを一時的に保持します。伝送キューは、USTRAN の値を持つ **Usage** キュー属性によって示されます。

形式名: FMXQH。

文字セットとエンコード: MQXQH 内のデータは、C プログラミング言語の場合、**CodedCharSetId** キュー・マネージャー属性で指定した文字セットと ENNAT で指定したローカル・キュー・マネージャーのエンコードで記述されていなければなりません。

MQXQH の文字セットおよびエンコードは、以下の **MDCSI** フィールドおよび **MDENC** フィールドに設定しなければなりません。

- 分離 MQMD (MQXQH 構造体がメッセージ・データの開始点にある場合)
- MQXQH 構造体に先行するヘッダー構造体 (その他のすべての場合)

使用法: 伝送キューにあるメッセージには、以下に示す 2 つのメッセージ記述子があります。

- メッセージ・データから独立して保管されるメッセージ記述子。これは独立メッセージ記述子と呼ばれ、メッセージが伝送キューに配置される場合、キュー・マネージャーにより生成されます。独立メッセージ記述子内のフィールドのいくつかは、MQPUT または MQPUT1 呼び出しでアプリケーションが提供するメッセージ記述子からコピーされます。

独立メッセージ記述子は、メッセージが伝送キューから除去されると、MQGET 呼び出しの **MSGDSC** パラメーターにあるアプリケーションに戻されます。

- 2 番目のメッセージ記述子は、メッセージ・データの一部として MQXQH 構造体内に保存されます。これは組み込みメッセージ記述子と呼ばれ、MQPUT または MQPUT1 呼び出し (少しのバリエーションあり) でアプリケーションが提供したメッセージ記述子のコピーです。

組み込みメッセージ記述子は、常にバージョン 1 の MQMD です。アプリケーションが書き込んだメッセージでは、MQMD 内の 1 つ以上のバージョン 2 フィールドにデフォルト値ではない値があると、MQMDE 構造体が MQXQH 構造体の後に続き、さらにアプリケーション・メッセージ・データがあればこれが続きます。この MQMDE 構造体は、次のいずれかです。

- キュー・マネージャーによって生成された (アプリケーションがメッセージを書き込むのにバージョン 2 の MQMD を使用した場合)。
- アプリケーション・メッセージ・データ開始時点からすでにあった (アプリケーションがメッセージを書き込むのにバージョン 1 の MQMD を使用した場合)。

組み込みメッセージ記述子は、メッセージが最終宛先キューから除去されると、MQGET 呼び出しの **MSGDSC** パラメーターにあるアプリケーションに戻されます。

- [1256 ページの『独立メッセージ記述子内のフィールド』](#)
- [1257 ページの『組み込みメッセージ記述子中のフィールド』](#)
- [1257 ページの『リモート・キューへのメッセージの書き込み』](#)

- [1258 ページの『伝送キューにメッセージを直接書き込む場合』](#)
- [1258 ページの『伝送キューからメッセージを読み取る場合』](#)
- [1258 ページの『フィールド』](#)
- [1259 ページの『初期値』](#)
- [1259 ページの『RPG 宣言』](#)

独立メッセージ記述子内のフィールド

独立メッセージ記述子のフィールドは、キュー・マネージャーにより、以下のリストのように設定されます。キュー・マネージャーがバージョン 2 の MQMD をサポートしていない場合、バージョン 1 の MQMD は機能を低下させることなく使用されます。

表 779. 独立メッセージ記述子内のフィールドと使用される値

独立 MQMD のフィールド	使用される値
MDSID	MDSIDV
MDVER	MDVER2
MDREP	組み込みメッセージ記述子からコピーされますが、ROAUXM により識別されるビットをゼロに設定します (これにより、メッセージが伝送キューに入れられる場合や伝送キューから除去される場合に、COA または COD レポート・メッセージが生成されなくなります。)
MDMT	組み込みメッセージ記述子からコピーされます。
MDEXP	組み込みメッセージ記述子からコピーされます。
MDFB	組み込みメッセージ記述子からコピーされます。
MDENC	ENNAT
MDCSI	キュー・マネージャーの CodedCharSetId 属性。
MDFMT	FMXQH
MDPRI	組み込みメッセージ記述子からコピーされます。
MDPER	組み込みメッセージ記述子からコピーされます。
MDMID	新しい値が、キュー・マネージャーにより生成されます。このメッセージ ID は、キュー・マネージャーが組み込みメッセージ記述子に対して生成した可能性がある <i>MDMID</i> とは異なります (上記参照)。
MDCID	組み込みメッセージ記述子からの <i>MDMID</i> 。
MDBOC	0
MDRQ	組み込みメッセージ記述子からコピーされます。
MDRM	組み込みメッセージ記述子からコピーされます。
MDUID	組み込みメッセージ記述子からコピーされます。
MDACC	組み込みメッセージ記述子からコピーされます。
MDAID	組み込みメッセージ記述子からコピーされます。
MDPAT	ATQM
MDPAN	キュー・マネージャー名の最初の 28 バイト。
MDPD	メッセージが伝送キューに書き込まれた日付。
MDPT	メッセージが伝送キューに書き込まれた時刻。
MDAOD	ブランク

表 779. 独立メッセージ記述子内のフィールドと使用される値 (続き)

独立 MQMD のフィールド	使用される値
<i>MDGID</i>	GINONE
<i>MDSEQ</i>	1
<i>MDOFF</i>	0
<i>MDMFL</i>	MFNONE
<i>MDOLN</i>	OLUNDF

組み込みメッセージ記述子中のフィールド

組み込みメッセージ記述子のフィールドの値は、以下の点を除いて、MQPUT または MQPUT1 呼び出しの **MSGDSC** パラメーターにある値と同じになります。

- *MDVER* フィールドの値は、常に MDVER1 です。
- *MDPRI* フィールドの値が PRQDEF の場合、値はキューの **DefPriority** 属性の値に置き換えられます。
- *MDPER* フィールドの値が PEQDEF の場合、値はキューの **DefPersistence** 属性の値に置き換えられます。
- *MDMID* フィールドの値が MINONE であるか、PMNMID オプションが指定されるか、またはメッセージが配布リストのメッセージの場合、*MDMID* は、キュー・マネージャーによって生成された新しいメッセージ ID に置き換えられます。

配布リストのメッセージが異なる伝送キューに置かれた短い配布リストのメッセージに細分化される場合、新しい各組み込みメッセージ記述子の *MDMID* フィールドも、元の配布リストのメッセージのフィールドと同じです。

- *PMNCID* オプションが指定されると、*MDCID* はキュー・マネージャーによって生成された新しい関連 ID に置き換えられます。
- コンテキスト・フィールドは、**PMO** パラメーターに指定された PM* オプションが示すとおりに設定されます。コンテキスト・フィールドは次のとおりです。
 - *MDACC*
 - *MDAID*
 - *MDAOD*
 - *MDPAN*
 - *MDPAT*
 - *MDPD*
 - *MDPT*
 - *MDUID*
- バージョン 2 のフィールドがある場合、1 つまたは複数のバージョン 2 フィールドにデフォルトではない値があると、これは MQMD から取り除かれ、さらに MQMDE 構造体に移動されます。

リモート・キューへのメッセージの書き込み

アプリケーションが (リモート・キューの名前を直接指定する方法か、リモート・キューのローカル定義を使用する方法で) リモート・キューにメッセージを書き込むと、ローカル・キュー・マネージャーは次の操作を実行します。

- 組み込みメッセージ記述子が入っている MQXQH 構造体の作成
- 必要な MQMDE 構造体がまだない場合、その MQMDE の追加
- アプリケーション・メッセージ・データの追加

- 該当する伝送キューへのメッセージの格納

伝送キューにメッセージを直接書き込む場合

アプリケーションから伝送キューにメッセージを直接書き込むこともできます。この場合、アプリケーションは、アプリケーション・メッセージ・データの接頭部に MQXQH 構造体を付け、適切な値でフィールドを初期設定する必要があります。さらに、MQPUT または MQPUT1 呼び出しのパラメータ **MSGDSC** のフィールド *MDFMT* には、FMXQH の値を設定する必要があります。

アプリケーションにより作成された MQXQH 構造体の中の文字データは、ローカル・キュー・マネージャーの文字セットに含まれているもの (**CodedCharSetId** キュー・マネージャー属性で定義されたもの) でなければならず、整数データは固有のマシン・エンコードに含まれているものでなければなりません。さらに、MQXQH 構造体の中にある文字データは、フィールドの定義長までブランクを埋め込む必要があります。ヌル文字を使用してデータを未完了で終了させてはなりません。キュー・マネージャーは、MQXQH 構造体に含まれるヌル文字とその後続の文字をブランクに変換しないためです。

ただし、キュー・マネージャーは、MQXQH 構造体が存在していること、およびそのフィールドに対して有効な値が指定されていることは検査しないため注意してください。

伝送キューからメッセージを読み取る場合

伝送キューからメッセージを読み取るアプリケーションでは、MQXQH 構造体に含まれている情報を適切な方法で処理する必要があります。アプリケーションメッセージデータの先頭に MQXQH 構造体が存在することは、MQGET 呼び出しのパラメータ **MSGDSCMDFMT** のフィールドに FMXQH 値が返されることで示される。パラメータ **MSGDSC** のおよび *MDENC MDCSI* フィールドで返される値は、MQXQH 構造体内の文字データおよび整数データの文字セットとエンコーディングを示します。アプリケーション・メッセージ・データの文字セットおよびエンコードは、組み込みメッセージ記述子内の *MDCSI* フィールドおよび *MDENC* フィールドによって定義されます。

フィールド

MQXQH 構造体には、以下のフィールドが含まれます。フィールドはアルファベット順に説明されています。

XQMD (MQMD1)

元のメッセージの記述子。

これは組み込みメッセージ記述子であり、メッセージが最初にリモート・キューに書き込まれたときに MQPUT 呼び出しまたは MQPUT1 呼び出しで **MSGDSC** パラメーターとして指定されたメッセージ記述子 MQMD とほぼ同じコピーです。

注: これはバージョン 1 の MQMD です。

フィールドの初期値は、MQMD 構造体の中のものと同じです。

XQRQ (48 バイトの文字ストリング)

宛先キューの名前。

これはメッセージの最終的な行き先と思われるメッセージキューの名前です（例えば、このキューが別のリモートキューのローカル定義 *XQRQM* として定義されている場合など、実際の最終的な行き先とは異なる可能性があります）。

メッセージが配布リストのメッセージ（つまり、組み込みメッセージ記述子中の *MDFMT* フィールドが *FMDH*）の場合、*XQRQ* はブランクです。

このフィールドの長さは *LNQN* によって指定されます。このフィールドの初期値は 48 個のブランク文字です。

XQRQM (48 バイトの文字ストリング)

宛先キュー・マネージャーの名前。

これは、メッセージの最終宛先であるキューを所有するキュー・マネージャー、またはキュー共有グループの名前です。

メッセージが配布リスト・メッセージの場合、XQRQM はブランクです。

このフィールドの長さはLNQMN によって指定されます。このフィールドの初期値は 48 個のブランク文字です。

XQSID (4 バイトの文字ストリング)

構造体 ID

値は次のものでなければなりません。

XQSIDV

伝送キュー・ヘッダー構造体の ID。

このフィールドの初期値は XQSIDV です。

XQVER (10 桁の符号付き整数)

構造体のバージョン番号。

値は次のものでなければなりません。

XQVER1

伝送キュー・ヘッダー構造体のバージョン番号。

以下の定数は、現行バージョンのバージョン番号を指定しています。

XQVERC

伝送キューのヘッダー構造体の現行バージョン。

このフィールドの初期値は XQVER1 です。

初期値

表 780. MQXQH のフィールド		
フィールド名	定数の名前	定数の値
XQSID	XQSIDV	'XQH '
XQVER	XQVER1	1
XQRQ	なし	ブランク
XQRQM	なし	ブランク
XQMD	MQMD と同じ名前および値。1157 ページの表 748 を参照	-
注:		
1. 記号-は、単一のブランク文字を表します。		

RPG 宣言

```
D*..1.....2.....3.....4.....5.....6.....7..
D*
D* MQXQH Structure
D*
D* Structure identifier
D  XQSID           1           4    INZ('XQH ')
D* Structure version number
D  XQVER           5           8I 0  INZ(1)
D* Name of destination queue
D  XQRQ            9           56    INZ
D* Name of destination queue manager
D  XQRQM           57          104    INZ
D* Original message descriptor
```

D	XQ1SID	105	108	INZ('MD')
D	XQ1VER	109	112I 0	INZ(1)
D	XQ1REP	113	116I 0	INZ(0)
D	XQ1MT	117	120I 0	INZ(8)
D	XQ1EXP	121	124I 0	INZ(-1)
D	XQ1FB	125	128I 0	INZ(0)
D	XQ1ENC	129	132I 0	INZ(273)
D	XQ1CSI	133	136I 0	INZ(0)
D	XQ1FMT	137	144	INZ('')
D	XQ1PRI	145	148I 0	INZ(-1)
D	XQ1PER	149	152I 0	INZ(2)
D	XQ1MID	153	176	INZ(X'0000000000000000-0000000000000000000000-000000000000')
D				
D	XQ1CID	177	200	INZ(X'0000000000000000-0000000000000000000000-000000000000')
D				
D	XQ1BOC	201	204I 0	INZ(0)
D	XQ1RQ	205	252	INZ
D	XQ1RM	253	300	INZ
D	XQ1UID	301	312	INZ
D	XQ1ACC	313	344	INZ(X'0000000000000000-0000000000000000000000-0000000000000000000000-0000000')
D				
D	XQ1AID	345	376	INZ
D	XQ1PAT	377	380I 0	INZ(0)
D	XQ1PAN	381	408	INZ
D	XQ1PD	409	416	INZ
D	XQ1PT	417	424	INZ
D	XQ1AOD	425	428	INZ

IBM i IBM i での関数呼び出し

この情報は、IBM i プログラミングで使用できる関数呼び出しについて学習するために使用します。

IBM i での呼び出しの記述で使用される規則

この一連のトピックでは、各呼び出しについて、呼び出しのパラメーターおよび使用法について説明します。また、プログラミング言語 RPG における呼び出しの典型的な呼び出し方法、そのパラメーターの通常の宣言方法について示します。

重要：IBM MQ API 呼び出しをコーディングする際は、すべての関連パラメーターを確実に指定する (以下の各セクションで説明されているように) 必要があります。これを行わないと、予測不能の結果になることがあります。

呼び出しの個別説明は、以下の形式で行います。

呼び出し名

呼び出し名。このあとに呼び出しの目的についての簡単な説明が続きます。

パラメーター

各パラメーター別に、そのパラメーター名の後にそのデータ・タイプを括弧 () に入れて示し、さらに入出力を示します。例えば以下のとおりです。

CMPCOD (9 桁の 10 進整数) - 出力

構造体データ・タイプについては、[1002 ページの『基本データ型について IBM i』](#)に詳細を記載してあります。

パラメーターの入出力の指示は、以下に示すとおりです。

入力

このパラメーターには、プログラマーが値を指定する必要があります。

出力

このパラメーターは、呼び出しによって戻されます。

入出力

プログラマーはこのパラメーターを指定する必要がありますが、その値は呼び出しによって変更されます。

パラメーターが取ることのできる値のリストとともに、パラメーターの目的についての簡単な説明もあります。

各呼び出しの最後の2つのパラメーターは、完了コードと理由コードです。完了コードは、呼び出しが正しく完了した、一部だけ完了した、または全く完了しなかった、のいずれかを示します。呼び出しが一部成功したまたは失敗した場合、その詳細は理由コードに示されます。

使用上の注意

呼び出しに関する追加情報。その使用方法や使用上の制約事項について説明します。

RPG での呼び出し

RPG 言語での呼び出しの一般的な呼び出し方法、およびそのパラメーターの宣言方法。

その他、以下に示す表記法を使用しています。

定数

定数の名前は、例えば OOOOUT のように、大文字で表します。

配列

ある種の呼び出しの場合、パラメーターは、サイズが固定していない文字ストリングの配列です。これらのパラメーターの記述の中では、小文字の *n* が数字の定数を表します。そのパラメーターの宣言をエンコードするときは、必要な数値で *n* を置き換えます。

IBM i IBM i での MQBACK (バックアウトの変更)

MQBACK 呼び出しは、最後の同期点以降に発生したメッセージの読み取りと書き込みをすべてバックアウトすることをキュー・マネージャーに示します。作業単位の一部として書き込まれたメッセージは削除されます。作業単位の一部として取り出されたメッセージはキューに戻されます。

- この呼び出しは、次の環境でサポートされます。

-  AIX
-  IBM i
-  Windows

- [1261 ページの『構文』](#)
- [1261 ページの『使用上の注意』](#)
- [1263 ページの『パラメーター』](#)
- [1263 ページの『RPG 宣言』](#)

構文

MQBACK (*Hconn*, *CompCode*, *Reason*)

使用上の注意

MQBACK を使用する際には、以下の使用上の注意を考慮してください。

- この呼び出しは、キュー・マネージャーそのものが作業単位を調整するときのみ使用できます。これはローカル作業単位で、変更内容が IBM MQ リソースに対してのみ影響を及ぼします。
- キュー・マネージャーが作業単位を調整しない環境では、MQBACK ではなく適切なバックアウト呼び出しを使用する必要があります。この環境ではまた、アプリケーションの異常終了を原因とする暗黙的バックアウトをサポートすることもできます。
 - IBM i では、この呼び出しはキュー・マネージャーで調整されるローカル作業単位で使うことができます。これは、コミットメント定義がジョブ・レベルで存在してはならないことを意味します。つまり、**CMTSCOPE(*JOB)** パラメーターを指定した STRCMTCTL コマンドがジョブに対して発行されてはなりません。

3. 作業単位内にあるコミットされていない変更内容でアプリケーションが終了する場合、それらの変更内容の後処理は、そのアプリケーションが正常に終了するか、異常終了するかで異なります。詳細については、[1300 ページの『IBM iでの MQDISC \(キュー・マネージャーの切断\)』](#)の使用上の注意を参照してください。
4. アプリケーションでグループ内のメッセージまたは論理メッセージのセグメントの書き込みまたは読み取りを行う場合、キュー・マネージャーは、最後に MQPUT および MQGET 呼び出しが正常に実行されたメッセージ・グループに関する情報を保存します。この情報は、キュー・ハンドルに関する次のような情報です。

- MQMD 中の MDGID、MDSEQ、MDOFF、および MDMFL フィールドの値。
- そのメッセージが作業単位の一部であるかどうか。
- MQPUT 呼び出しについて、そのメッセージが持続メッセージか、非持続メッセージか。

キュー・マネージャーは、次のものについて 1 つずつ、3 セットのグループおよびセグメント情報を保持しています。

- 最後に正常に実行された MQPUT 呼び出し (これは作業単位の一部である場合があります)。
- 最後に正常に実行された MQGET 呼び出しのうちキューからメッセージを削除したもの (作業単位の一部である場合があります)。
- 最後に正常に実行された MQGET 呼び出しのうちキュー上のメッセージをブラウズしたもの (これが作業単位の一部であることはありません)。

アプリケーションで、作業単位の一部としてそのメッセージの書き込みまたは読み取りを行い、その作業単位をバックアウトすると、そのグループおよびセグメント情報は、その以前の値に復元されます。

- MQPUT 呼び出しについての情報は、現行作業単位内のそのキュー・ハンドルについて最初に正常に実行された MQPUT 呼び出し以前の値に復元されます。
- MQGET 呼び出しについての情報は、現行作業単位内のそのキュー・ハンドルについて最初に正常に実行された MQGET 呼び出し以前の値に復元されます。

作業単位の開始後にアプリケーションによって更新されたが、作業単位の有効範囲外にあるキューは、作業単位がバックアウトされた場合に、そのグループ情報およびセグメント情報を復元しません。

作業単位のバックアウト時にグループおよびセグメント情報を以前の値に復元する機能により、アプリケーションは、数多くのセグメントで構成される大きなメッセージ・グループまたは大きな論理メッセージをいくつかの作業単位にまたがって広げることができます。そして、いずれかの作業単位が失敗しても、そのメッセージ・グループまたは論理メッセージ内の正しい点でアプリケーションを再始動できます。ローカル・キュー・マネージャーのキュー・ストレージが限られている場合には、いくつかの作業単位を使用する方が有効となる場合があります。ただし、システム障害の発生時に各メッセージの書き込みまたは読み取りを正しい時点で再始動できるようにするには、アプリケーションが十分な情報を維持する必要があります。システム障害後に正しい時点から再始動する方法の詳細については、[1180 ページの『IBM iでの MQPMO \(メッセージ書き込みオプション\)』](#)の PMLOGO オプション、および [1080 ページの『IBM iでの MQGMO \(読み取りメッセージ・オプション\)』](#)の GMLOGO オプションを参照してください。

次の『使用上の注意』は、キュー・マネージャーで作業単位を調整する場合にのみ適用されます。

1. 作業単位の 1 つには、1 つの接続ハンドルと同じ有効範囲があります。つまり、特定の作業単位に影響を与える IBM MQ 呼び出しはすべて、同じ接続ハンドルを使用して実行しなければなりません。別の接続ハンドルを用いて呼び出しを発行すると (例えば、別のアプリケーションで呼び出しを発行する)、別の作業単位に影響が及びます。接続ハンドルの有効範囲の詳細については、[1287 ページの『IBM iでの MQCONN \(キュー・マネージャーの接続\)』](#)のセクションで説明している **HCONN** パラメーターを参照してください。
2. この呼び出しで影響を受けるメッセージは、現行の作業単位の一部として書き込まれたメッセージ、または取り出されたメッセージに限られます。
3. 1 つの作業単位内で MQGET、MQPUT、または MQPUT1 呼び出しを発行する長時間実行中のアプリケーションが、コミット呼び出しやバックアウト呼び出しを一度も発行しないと、他のアプリケーションが使用できないメッセージでキューが満杯になる恐れがあります。これを避けるには、管理者は、ランナウェイ・アプリケーションがキューをいっぱいにしない程度に低く、かつ必要なメッセージング・アプ

リケーションが正常に機能する程度に高い値を **MaxUncommittedMsgs** キュー・マネージャー属性に設定しなければなりません。

パラメーター

MQBACK 呼び出しには、以下のパラメーターがあります。

HCONN (10 桁の符号付き整数) - 入力

接続ハンドル。

このハンドルは、キュー・マネージャーに対する接続を表します。HCONN の値は、先行の MQCONN または MQCONNX 呼び出しによって戻されたものです。

CMPCOD (10 桁の符号付き整数) - 出力

完了コード。

これは、以下のいずれかになります。

CCOK

正常終了。

CCFAIL

呼び出し失敗。

REASON (10 桁の符号付き整数) - 出力

COMCOD を限定する理由コード。

COMCOD が CCOK の場合

RCNONE

(0, X'000') レポートする理由コードはありません。

COMCOD が CCFAIL の場合

RC2219

(2219, X'8AB') 前の呼び出しが完了する前に MQI 呼び出しが再入力されました。

RC2009

(2009, X'7D9') キュー・マネージャーとの接続が失われました。

RC2018

(2018, X'7E2') 接続ハンドルが無効です。

RC2101

(2101, X'835') オブジェクトが損傷しました。

RC2123

(2123, X'84B') コミットまたはバックアウト操作の結果が混在している。

RC2162

(2162, X'872') キュー・マネージャーのシャットダウン中です。

RC2102

(2102, X'836') 使用できるシステム・リソースが不足しています。

RC2071

(2071, X'817') ストレージが不足しています。

RC2195

(2195, X'893') 予期しないエラーが発生しました。

RPG 宣言

```
C*.1....:....2....:....3....:....4....:....5....:....6....:....7..  
C          CALLP      MQBACK(HCONN : COMCOD : REASON)
```

呼び出しのプロトタイプ定義は次のようになります。

```

D*..1.....2.....3.....4.....5.....6.....7..
DMQBACK          PR          EXTPROC('MQBACK')
D* Connection handle
D HCONN          10I 0 VALUE
D* Completion code
D COMCOD          10I 0
D* Reason code qualifying COMCOD
D REASON          10I 0

```

IBM i での MQBEGIN (作業単位の開始)

MQBEGIN 呼び出しは、キュー・マネージャーによって調整される作業単位を開始します。また、この作業単位は外部リソース・マネージャーを伴うこともあります。

- この呼び出しは、次の環境でサポートされます。

-  AIX
-  IBM i
-  Windows

- [1264 ページの『構文』](#)
- [1264 ページの『使用上の注意』](#)
- [1265 ページの『パラメーター』](#)
- [1266 ページの『RPG 宣言』](#)

構文

MQBEGIN (*HCONN*, *BEGOP*, *CMPCOD*, *REASON*)

使用上の注意

1. MQBEGIN 呼び出しは、キュー・マネージャーで調整される作業単位の開始に使用します。また、この作業単位で他のリソース・マネージャー所有のリソースへの変更を行うこともあります。キュー・マネージャーは、次の 3 つのタイプの作業単位をサポートします。

キュー・マネージャーで調整されるローカル作業単位

これは、参加するリソース・マネージャーがキュー・マネージャーのみである作業単位であり、したがってキュー・マネージャーが作業単位コーディネーターとして機能します。

- このタイプの作業単位を開始するには、作業単位内の最初の MQPUT、MQPUT1、または MQGET 呼び出しに PMSYP または GMSYP オプションを指定してください。

ローカル作業単位を開始するのに、アプリケーションが MQBEGIN 呼び出しを発行する必要はありません。ただし、MQBEGIN を使用すると、呼び出しは CCWARN および理由コード RC2121 で完了します。

- このタイプの作業単位をコミットまたはバックアウトするには、MQCMIT または MQBACK 呼び出しを使用する必要があります。

キュー・マネージャーで調整されるグローバル作業単位

キュー・マネージャーが、IBM MQ リソースおよび他のリソース・マネージャーに属するリソースの両方に対して作業単位コーディネーターとして機能する作業単位です。これらのリソース・マネージャーは、キュー・マネージャーと連携して、必ず作業単位内のリソースへのすべての変更内容が一度にコミットまたはバックアウトされるようにします。

- このタイプの作業単位を開始するには、MQBEGIN 呼び出しを使用する必要があります。
- このタイプの作業単位をコミットまたはバックアウトするには、MQCMIT および MQBACK 呼び出しを使用する必要があります。

外部調整されるグローバル作業単位

キュー・マネージャーが参加プログラムであるが、作業単位コーディネーターとしては機能しない作業単位です。その代わりに、キュー・マネージャーと連携する外部作業単位コーディネーターが存在します。

- このタイプの作業単位を開始するには、外部作業単位コーディネーターが提供する関連呼び出しを使用する必要があります。

作業単位を開始するために MQBEGIN 呼び出しを使用すると失敗し、理由コード RC2012 が戻ります。

- このタイプの作業単位をコミットまたはバックアウトするには、外部作業単位コーディネーターが提供するコミット呼び出しおよびバックアウト呼び出しを使用しなければなりません。

作業単位のコミットまたはバックアウトをするために MQCMIT または MQBACK 呼び出しを使用すると失敗し、理由コード RC2012 が戻ります。

2. 作業単位内にあるコミットされていない変更内容でアプリケーションが終了する場合、それらの変更内容の後処理は、そのアプリケーションが正常に終了するか、異常終了するかで異なります。詳細については、1300 ページの『[IBM i での MQDISC \(キュー・マネージャーの切断\)](#)』の使用上の注意を参照してください。
3. アプリケーションが一度に参加プログラムとしてかかわることができる作業単位は、1 つだけです。アプリケーションで MQBEGIN 呼び出しを発行する場合、そのアプリケーション用の作業単位がすでに存在していると、その呼び出しは、作業単位のタイプに関係なく失敗し、理由コード RC2128 が戻ります。
4. MQBEGIN 呼び出しは、IBM MQ クライアント環境では無効となります。この呼び出しを使用しようとすると失敗し、理由コード RC2012 が戻ります。
5. キュー・マネージャーが各グローバル作業単位の作業単位コーディネーターとして機能している場合、作業単位に参加プログラムとしてかかわることができるリソース・マネージャーは、キュー・マネージャーの構成ファイル内に定義されます。
6. IBM i では、次の 3 つのタイプの作業単位がサポートされています。
 - キュー管理者が調整するローカル作業単位は、ジョブレベルでコミットメント定義が存在しない場合にのみ使用できます。つまり、そのジョブに対して **CMTSCOPE(*JOB)** パラメータ付き **STRCMTCTL** コマンドが発行されていない必要があります。
 - キュー・マネージャーで調整されるグローバル作業単位は、サポートされていません。
 - 外部調整されるグローバル作業単位は、コミットメント定義がジョブ・レベルで存在する場合、つまり、STRCMTCTL コマンドに **CMTSCOPE(*JOB)** パラメーターを指定したものがジョブに対して発行された場合にのみ使用することができます。それが実行されている場合、IBM i の COMMIT および ROLLBACK 操作が、IBM MQ リソースと他の参加しているリソース・マネージャーのリソースに対して適用されます。

パラメーター

MQBEGIN 呼び出しには、以下のパラメーターがあります。

HCONN (10 桁の符号付き整数) - 入力

接続ハンドル。

このハンドルは、キュー・マネージャーに対する接続を表します。HCONN の値は、先行の MQCONN または MQCONNX 呼び出しによって戻されたものです。

BEGOP (MQBO) - 入出力

MQBEGIN のアクションを制御するオプション。

詳細は 1020 ページの『[IBM i での MQBO \(開始オプション\)](#)』を参照してください。

必須オプションがない場合、C アセンブラーまたは S/390 アセンブラーで作成されたプログラムでは、MQBO 構造体のアドレスを指定せずに、ヌル・パラメーター・アドレスを指定することができます。

CMPCOD (10桁の符号付き整数) - 出力

完了コード。

これは、以下のいずれかになります。

CCOK

正常終了。

CCWARN

警告 (部分完了)。

CCFAIL

呼び出し失敗。

REASON (10桁の符号付き整数) - 出力

CMPCOD を限定する理由コード。

CMPCOD が CCOK の場合

RCNONE

(0, X'000') レポートする理由コードはありません。

CMPCOD が CCWARN の場合

RC2121

(2121, X'849') 参加するリソース・マネージャーが登録されていない。

RC2122

(2122, X'84A') 参加するリソース・マネージャーが利用不能である。

CMPCOD が CCFAIL の場合

RC2134

(2134, X'856') 開始オプション構造体が無効である。

RC2219

(2219, X'8AB') 前の呼び出しが完了する前に MQI 呼び出しが再入力されました。

RC2009

(2009, X'7D9') キュー・マネージャーとの接続が失われました。

RC2012

(2012, X'7DC') この環境では呼び出しが無効です。

RC2018

(2018, X'7E2') 接続ハンドルが無効です。

RC2046

(2046, X'7FE') オプションが無効であるか、矛盾しています。

RC2162

(2162, X'872') キュー・マネージャーのシャットダウン中です。

RC2102

(2102, X'836') 使用できるシステム・リソースが不足しています。

RC2071

(2071, X'817') ストレージが不足しています。

RC2195

(2195, X'893') 予期しないエラーが発生しました。

RC2128

(2128, X'850') 作業単位が開始済みである。

RPG 宣言

```
C*.1.....2.....3.....4.....5.....6.....7..  
C          CALLP      MQBEGIN(HCONN : BEGOP : CMPCOD :  
C                                REASON)
```


呼び出しのプロトタイプ定義は次のようになります。

```
D*..1.....2.....3.....4.....5.....6.....7..
MQBEGIN          PR          EXTPROC('MQBEGIN')
D* Connection handle
D HCONN          10I 0 VALUE
D* Options that control the action of MQBEGIN
D BEGOP          12A
D* Completion code
D CMPCOD          10I 0
D* Reason code qualifying CMPCOD
D REASON          10I 0
```

IBM i IBM i での MQBUFMH (バッファからメッセージ・ハンドルへの変換)

MQBUFMH 関数呼び出しは、バッファをメッセージ・ハンドルに変換するので、MQMHBUF 呼び出しの逆です。

この呼び出しは、メッセージ記述子と、バッファ内の MQRFH2 プロパティを取り、メッセージ・ハンドルを使用してそれらを使用可能にします。メッセージ・データの MQRFH2 プロパティは、オプションで除去されます。メッセージ記述子の *Encoding*、*CodedCharSetId*、および *Format* フィールドは、必要であればプロパティが除去された後に更新され、バッファの内容が正しく記述されます。

- [1267 ページの『構文』](#)
- [1267 ページの『使用上の注意』](#)
- [1267 ページの『パラメーター』](#)
- [1269 ページの『RPG 宣言』](#)

構文

MQBUFMH (*Hconn*, *Hmsg*, *BufMsgHOpts*, *MsgDesc*, *Buffer*, *BufferLength*, *DataLength*, *CompCode*, *Reason*)

使用上の注意

MQBUFMH 呼び出しは、API 出口によってインターセプトできません。アプリケーションのスペース内でバッファはメッセージ・ハンドルに変換されます。この呼び出しはキュー・マネージャーに到達しません。

パラメーター

MQBUFMH 呼び出しには、以下のパラメーターがあります。

HCONN (10 桁の符号付き整数) - 入力

このハンドルは、キュー・マネージャーに対する接続を表します。HCONN の値は、Hmsg パラメーターで指定されているメッセージ・ハンドルを作成するために使用された接続ハンドルと一致していなければなりません。

HCUNAS を使用してメッセージ・ハンドルが作成された場合は、バッファをメッセージ・ハンドルに変換するスレッド上で有効な接続を確立しなければなりません。有効な接続が確立されていない場合、呼び出しは失敗し、RC2009 となります。

HMSG (20 桁の符号付き整数) - 入力

このハンドルは、バッファに必要なメッセージ・ハンドルです。値は、前の MQCRTMH 呼び出しで戻されたものです。

BMHOPT (MQBMHO) - 入力

アプリケーションでは、MQBMHO 構造体を使用することによって、バッファからメッセージ・ハンドルを生成する方法を制御するためのオプションを指定することができます。

詳細は 1019 ページの『IBM i での MQBMHO (バッファからメッセージ・ハンドルへの変換オプション)』を参照してください。

MSGDSC (MQMD) - 入出力

MSGDSC 構造体には、メッセージ記述子プロパティが入れられます。これはバッファ域の内容を記述します。

呼び出しからの出力上で、オプションでプロパティがバッファ域から除去されます。この場合、メッセージ記述子が更新され、バッファ域は正しく記述されます。

この構造体中のデータは、アプリケーションの文字セット内およびエンコード内になければなりません。

BUFLEN (10 桁の符号付き整数) - 入力

BUFLEN は、バッファ域の長さです (バイト単位)。

0 バイトの BUFLEN は有効であり、それは、そのバッファ域にデータが含まれていないことを示します。

BUFFER (1 バイトのビット・ストリング x BUFLEN) - 入出力

BUFFER は、メッセージ・バッファが入れられる領域を定義します。ほとんどのデータの場合、バッファを 4 バイトの境界に位置合わせする必要があります。

BUFFER に文字データまたは数値データが含まれている場合は、MSGDSC パラメーターの *CodedCharSetId* および *Encoding* フィールドを、データに適した値に設定します。これにより、必要に応じてデータを変換することができます。

メッセージ・バッファ中にプロパティがある場合はオプションで除去されます。これらのプロパティは、後で呼び出しから戻る際にメッセージ・ハンドルから使用できるようになります。

C プログラミング言語では、パラメーターは、void を示すポインタとして宣言されます。つまり、どのタイプのデータのアドレスもパラメーターとして指定できます。

BUFLEN パラメーターがゼロの場合、BUFFER は参照されません。この場合、C または System/390 アセンブラで作成されたプログラムによって渡されるパラメーター・アドレスはヌルのこともあります。

DATLEN (10 桁の符号付き整数) - 出力

DATLEN は、プロパティが削除された可能性のあるバッファの長さです (バイト単位)。

CMPCOD (10 桁の符号付き整数) - 出力

CCOK

正常終了。

CCFAIL

呼び出し失敗。

REASON (10 桁の符号付き整数) - 出力

CMPCOD を限定する理由コード。

CMPCOD が CCOK の場合

RCNONE

(0, X'000') レポートする理由コードはありません。

CMPCOD が CCFAIL の場合

RC2204

(2204, X'089C') アダプターが利用できません。

RC2130

(2130, X'852') アダプター・サービス・モジュールをロードできません。

RC2157

(2157, X'86D') 1 次 ASID とホーム ASID が異なっています。

RC2489

(2489, X'09B9') バッファからメッセージ・ハンドルへの変換オプション構造体が無効です。

RC2004

(2004, X'07D4') バッファ・パラメーターが無効である。

RC2005

(2005, X'07D5') バッファ長パラメーターは無効です。

RC2219

(2219, X'08AB') 前の呼び出しが完了する前に MQI 呼び出しが入力された。

RC2009

(2009, X'07D9') キュー・マネージャーとの接続が失われました。

RC2460

(2460, X'099C') メッセージ・ハンドルが無効。

RC2026

(2026, X'07EA') メッセージ記述子が無効である。

RC2499

(2499, X'09C3') メッセージ・ハンドルがすでに使用中。

RC2046

(2046, X'07FE') オプションが無効であるか、矛盾しています。

RC2334

(2334, X'091E') MQRFH2 構造体が無効である。

RC2421

(2421, X'0975') プロパティを含む MQRFH2 フォルダを構文解析できなかった。

RC2195

(2195, X'893') 予期しないエラーが発生しました。

RPG 宣言

```
C*..1.....2.....3.....4.....5.....6.....7..
C                                CALLP      MQBUFMH(HCONN : HMSG : BMHOPT :
                                MSGDSC : BUFLN : BUFFER :
                                DATLEN : CMPCOD : REASON)
```

呼び出しのプロトタイプ定義は次のようになります。

```
DMQBUFMH          PR              EXTPROC('MQBUFMH')
D* Connection handle
D HCONN              10I 0
D* Message handle
D HMSG              10I 0
D* Options that control the action of MQBUFMH
D BMHOPT            12A  VALUE
D* Message descriptor
D MSGDSC              364A
D* Length in bytes of the Buffer area
D BUFLN              10I 0
D* Area to contain the message buffer
D BUFFER              *  VALUE
D* Length of the output buffer
D DATLEN              10I 0
D* Completion code
D CMPCOD              10I 0
D* Reason code qualifying CompCode
D REASON              10I 0
```

IBM i IBM i での MQCB (コールバック管理)

MQCB 呼び出しは、指定されたオブジェクト・ハンドルにコールバックを登録し、そのコールバックの活動化とそのコールバックに対する変更を制御します。

コールバックとは、特定のイベントが発生した時点で IBM MQ によって呼び出されるコードの断片 (動的にリンクできる関数の名前か関数ポインターのいずれかとして指定される) のことです。

V7 クライアントで MQCB および MQCTL を使用するには、V7 サーバーに接続しなければならず、チャンネルの **SHARECNV** パラメーターにゼロ以外の値がなければなりません。

定義できるコールバックのタイプは以下のとおりです。

メッセージ・コンシューマー

メッセージ・コンシューマー・コールバック関数は、指定された選択基準と一致するメッセージがオブジェクト・ハンドル上で使用可能な時点で呼び出されます。

各オブジェクト・ハンドルに対して登録できるコールバック関数は 1 つのみです。複数の選択基準を指定して単一のキューを読み取る場合は、そのキューを複数回オープンしなければならず、各ハンドル上に 1 つのコンシューマー関数が登録されています。

イベント・ハンドラー

コールバック環境全体に影響する条件に関するイベント・ハンドラーが呼び出されます。

キュー・マネージャーや接続の停止や静止などのイベント条件が発生すると、この関数が呼び出されます。

この関数は、単一のメッセージ・コンシューマー固有の条件 (例えば RC2016) では呼び出されません。ただし、コールバック関数が正常に終了しない場合には呼び出されます。

- [1270 ページの『構文』](#)
- [1270 ページの『MQCB の使用上の注意』](#)
- [1271 ページの『MQCB のパラメーター』](#)
- [1278 ページの『RPG 宣言』](#)

構文

MQCB (*HCONN, OPERATN, HOBJ, CBDSC, MSGDSC, GMO, CMPCOD, REASON*)

MQCB の使用上の注意

1. MQCB を使用して、キュー上で使用可能で、指定された基準と一致する、メッセージごとに呼び出されるアクションを定義します。アクションが処理される際には、メッセージがキューから除去されて定義済みのメッセージ・コンシューマーに渡されるか、メッセージ・トークンが提供されてメッセージの取り出しに使用されます。
2. MQCB は、MQCTL を使用したコンシュームを始める前にコールバック・ルーチンを定義するために使用するか、またはコールバック・ルーチン内から使用することができます。
3. コールバック・ルーチン外から MQCB を使用するには、最初に MQCTL を使用したメッセージ・コンシュームを中断し、その後でコンシュームを再開しなければなりません。

メッセージ・コンシューマーのコールバック・シーケンス

コンシューマーを構成して、そのコンシューマーがライフ・サイクル内におけるキーポイントで、コールバックを呼び出すようにすることができます。以下に例を示します。

- コンシューマーが最初に登録される時
- 接続が開始するとき
- 接続が停止するとき
- MQCLOSE により明示的もしくは暗示的に、コンシューマーの登録が解除される時

表 781. MQCTL verb 定義	
動詞	意味
MQCTL(START)	CTLSR 操作を使用した MQCTL 呼び出し

表 781. MQCTL verb 定義 (続き)	
動詞	意味
MQCTL(STOP)	CTLSP 操作を使用した MQCTL 呼び出し
MQCTL(WAIT)	CTLSW 操作を使用した MQCTL 呼び出し

これによりコンシューマーは、コンシューマーに関連した状態を維持できます。コールバックがアプリケーションから要求される場合、コンシューマー呼び出しの規則は以下のようになります。

REGISTER

常にコールバックの最初のタイプの呼び出しになります。

必ず MQCB(CBREG) 呼び出しと同じスレッドで呼び出されます。

START

常に MQCTL(START) verb と同期して呼び出されます。

- MQCTL(START) verb が戻る前にすべての START コールバックは完了します。

CTLTHR が要求される場合はメッセージ配信と同じスレッドにあります。

前のコールバックが MQCTL(START) 中に MQCTL(STOP) を発行した場合などは、START コールは保障されません。

STOP

接続が再開するまで、この呼び出し以降はメッセージやイベントは配信されません。

アプリケーションが START、メッセージ、またはイベントの呼び出しを前に受けていれば、STOP は保障されます。

DEREGISTER

常にコールバックの最後のタイプの呼び出しになります。

アプリケーションでスレッド・ベースの初期化を実行して、必ず START と STOP のコールバックをクリーンアップするようにしてください。非スレッド・ベースの初期化を行って、REGISTER と DEREGISTER のコールバックのクリーンアップができます。

スレッドのライフと可用性に関しては、説明されていること以外の推測を行わないでください。例えば、最後に DEREGISTER の呼び出しがされた後も生きているスレッドを当てにしないでください。同様に、CTLTHR を使用しないことを選択している場合は、接続の開始時には必ずスレッドが存在しているとは想定しないでください。

スレッドの特性に対してアプリケーションに特定の要件がある場合は、必ずそれに合わせてスレッドが作成され、それから MQCTL(WAIT) が使用されます。このステップは、非同期メッセージ配信のためにスレッド IBM MQ をに委譲します。

メッセージ・コンシューマーの接続使用法

通常、1 つの MQI 呼び出しが未解決の間に別の呼び出しをアプリケーションが発行すると、呼び出しは失敗して、理由コード RC2219 が出されます。

ただし、前の呼び出しが完了する前にアプリケーションがさらに MQI 呼び出しを発行しなければならない特殊なケースがあります。例えば、コンシューマーの呼び出しは、CBRE を指定した MQCB 呼び出し中に行うことができます。

このような場合、MQCB か MQCTL verb のいずれかをアプリケーションが発行する結果として、アプリケーションはコールバックされ、次の MQI 呼び出しを出すことができます。この場合、CBCCALLT タイプの CBCTRC で呼び出されたときに、例えば MQOPEN 呼び出しをコンシューマー関数で発行できます。MQDISC を除く任意の MQI 呼び出しを実行できます。

MQCB のパラメーター

MQCB 呼び出しには、以下のパラメーターがあります。

HCONN (10 桁の符号付き整数) - 入力

コールバック管理関数 - HCONN パラメーター。

このハンドルは、キュー・マネージャーに対する接続を表します。HCONN の値は、先行の MQCONN または MQCONNX 呼び出しによって戻されたものです。

OPERATN (10 桁の符号付き整数) - 入力

コールバック管理関数 - OPERATN パラメーター。

指定されたオブジェクト・ハンドルに定義されたコールバックで処理されている操作。以下のオプションのいずれかを指定する必要があります。複数のオプションが必要な場合には、値を追加するか (複数回同じ定数を追加しないでください)、またはビット単位 OR 演算を使用して値を組み合わせることができます (プログラミング言語がビット演算をサポートする場合)。

無効な組み合わせには、その旨を示しています。その他の組み合わせはすべて有効です。

CBREG

指定されたオブジェクト・ハンドルにコールバック関数を定義します。この操作は、呼び出される関数と、使用される選択基準を定義します。

オブジェクト・ハンドルに関するコールバック関数がすでに定義されている場合は、その定義は置き換えられます。コールバックを置き換えている間にエラーが検出されると、関数は登録解除されます。

以前にコールバックが登録解除されたコールバック関数でコールバックが登録される場合は、置き換え操作として扱われます。初期呼び出しまたは最終呼び出しは呼び出されません。

CBREG は CTLSU または CTLRE と一緒に使用できます。

CBUNR

オブジェクト・ハンドルのメッセージのコンシュームを停止し、このハンドルをコールバックに適格なものから除きます。

関連付けられているハンドルがクローズすると、コールバックは自動的に登録解除されます。

CBUNR がコンシューマー内から呼び出され、そのコールバックに呼び出しの停止が定義されている場合、これはコンシューマーからの戻り時に呼び出されます。

この操作がコンシューマーが登録されていない *Hobj* に対して発行されると、この呼び出しは RC2448 で戻されます。

CTLSU

オブジェクト・ハンドルに関するメッセージのコンシュームを中断します。

この操作がイベント・ハンドラーに適用される場合は、中断している間にイベント・ハンドラーはイベントを読み取らず、中断状態の間に失われたイベントは再開時に操作に提供されません。

中断状態の間、コンシューマー関数は制御タイプのコールバックの取得を続行します。

CTLRE

オブジェクト・ハンドルに関するメッセージのコンシュームを再開します。

この操作がイベント・ハンドラーに適用される場合は、中断している間にイベント・ハンドラーはイベントを読み取らず、中断状態の間に失われたイベントは再開時に操作に提供されません。

CBDSC (MQCBD) - 入力

コールバック管理関数 - CBDSC パラメーター。

これは、アプリケーションによって登録されているコールバック関数と、登録時に使用されるオプションを識別する構造です。

この構造体の詳細については、[233 ページの『MQCBD - コールバック記述子』](#)を参照してください。

コールバック記述子は、CBREG オプションにのみ必要です。この記述子が必要とされない場合、渡されるパラメーター・アドレスがヌルである可能性があります。

HOBJ (10 桁の符号付き整数) - 入力

コールバック管理関数 - HOBJ パラメーター。

このハンドルは、メッセージのコンシューム元のオブジェクトに対し設定されたアクセスを表します。これは、以前の **MQOPEN** または **MQSUB** 呼び出しから戻されたハンドルです (**HOBJ** パラメーター内に)。

HOBJ は、イベント・ハンドラー・ルーチン (CBTEH) の定義時には不要です。これは、**HONONE** として指定されていなければなりません。

この **Hobj** が **MQOPEN** 呼び出しから戻された場合、キューは、以下に示す 1 つ以上のオプションでオープンしておく必要があります。

- **OOINPS**
- **OOINPX**
- **OOINPQ**
- **OOBRW**

MSGDSC (MQMD) - 入力

コールバック管理関数 - MSGDSC パラメーター。

この構造体は、必要なメッセージの属性と、取り出されるメッセージの属性を記述します。

MsgDesc パラメーターは、コンシューマーが必要とするメッセージの属性と、メッセージ・コンシューマーに渡される **MQMD** のバージョンを定義します。

MQMD 中で、**MsgId**、**CorrelId**、**GroupId**、**MsgSeqNumber**、および **Offset** が、**GetMsgOpts** パラメーターで指定されたオプションに応じて、メッセージの選択に使用されます。

GMCONV オプションを指定する場合、メッセージの変換に **Encoding** と **CodedCharSetId** が使用されます。

詳細については、**MQMD** を参照してください。

MsgDesc は、任意のフィールドでデフォルト値以外の値を必要とする場合の **CBREG** にのみ使用されます。**MsgDesc** はイベント・ハンドラーには使用されません。

記述子が必須でない場合は、渡されるパラメーター・アドレスをヌルにすることができます。

複数のコンシューマーが、セクターがオーバーラップしている同一のキューに対して登録されている場合は、メッセージごとに選択されるコンシューマーが未定義になることに注意してください。

GMO (MQGMO) - 入力

コールバック管理関数 - GMO パラメーター。

メッセージ・コンシューマーによるメッセージの取得を制御するオプション。

MQGET 呼び出しで使用される場合のすべてのオプションの意味は、1080 ページの『IBM i での MQGMO (読み取りメッセージ・オプション)』に説明されています。例外を以下に示します。

GMSSIG

このオプションは許可されていません。

GMBRWF、GMBRWN、GMMBH、GMMBC

ブラウズしているコンシューマーに配布されるメッセージの順序は、これらのオプションの組み合わせで指示されます。以下の組み合わせが有効です。

GMBRWF

キュー上の最初のメッセージが繰り返しコンシューマーに配布されます。このオプションは、コンシューマーがコールバック中のメッセージを破壊的にコンシュームする場合に便利です。このオプションは注意して使用してください。

GMBRWN

コンシューマーは、現行カーソル位置からキューの終わりに達するまで、キュー上の各メッセージを与えられます。

GMBRWF + GMBRWN

カーソルはキューの先頭にリセットされます。リセット後、カーソルがキューの終わりに達するまで、コンシューマーは各メッセージを与えられます。

GMBRWF + GMMBH または GMMBC

キューの先頭から始まって、コンシューマーはキュー上のマークが付いていない最初のメッセージを与えられ、その後このコンシューマー用にマークが付けられます。この組み合わせにより、コンシューマーは現行カーソル・ポイントの後に追加された新しいメッセージを確実に受け取ることができます。

GMBRWN + GMMBH または GMMBC

コンシューマーには、カーソル位置から始めて、キューの次のマークの付いていないメッセージが送信されます。このメッセージには、このコンシューマーのためにマークが付けられます。メッセージを現行カーソル位置の後のキューに追加できるので、この組み合わせは注意して使用してください。

GMBRWF + GMBRWN + GMMBH または GMMBC

この組み合わせは許可されていません。この呼び出しを使用すると、RC2046 が戻されます。

GMNWT、GMWT、および GMWI

これらのオプションは、コンシューマーを呼び出す方法を制御します。

GMNWT

コンシューマーは、RC2033 では呼び出されません。コンシューマーはメッセージとイベントに関してのみ呼び出されます。

GMWT およびゼロの GMWI

メッセージがなく、さらに以下の場合にのみ、RC2033 コードがコンシューマーに渡されます。

- コンシューマーが開始されている。
- 最後のメッセージがない理由コード以降、少なくとも 1 つのメッセージがコンシューマーに送信されている。

この場合、ゼロの待機間隔が指定されていると、コンシューマーはビジー・ループ中でポーリングできません。

GMWT および正の GMWI

理由コード RC2033 による指定された待機インターバルの後、ユーザーが呼び出されます。この呼び出しは、どのメッセージがコンシューマーに送信されたかに関係なく行われます。これにより、ユーザーは、ハートビートまたはバッチ・タイプの処理を実行できます。

GMWT および WIULIM の GMWI

これは、RC2033 が戻されるまで無制限に待機することを指定します。コンシューマーは、RC2033 では呼び出されません。

GMO は、任意のフィールドでデフォルト値以外の値を必要とする場合の CBREG にのみ使用されます。GMO はイベント・ハンドラーには使用されません。

オプションが必要とされない場合、渡されるパラメーター・アドレスがヌルである可能性があります。

MQGMO 構造中でメッセージ・プロパティ・ハンドルが提供されている場合は、コンシューマー・コールバック中に渡されるコピーが MQGMO 構造中に提供されます。MQCB 呼び出しから戻る際に、アプリケーションはメッセージ・プロパティ・ハンドルを削除できます。

CMPCOD (10 桁の符号付き整数) - 出力

コールバック管理関数 - CMPCOD パラメーター。

完了コード。以下のいずれかです。

CCOK

正常終了。

CCWARN

警告 (部分完了)。

CCFAIL

呼び出し失敗。

REASON (10 桁の符号付き整数) - 出力

コールバック管理関数 - REASON パラメーター。

次の理由コードは、キュー・マネージャーが **REASON** パラメーターに対して戻す理由コードです。

CMPCOD が CCOK の場合

RCNONE

(0, X'000') レポートする理由コードはありません。

CompCode が CCFAIL の場合

RC2204

(2204, X'89C') アダプターが利用できません。

RC2133

(2133, X'855') データ変換サービス・モジュールをロードできない。

RC2130

(2130, X'852') アダプター・サービス・モジュールをロードできません。

RC2374

(2374, X'946') API 出口で障害が発生しました。

RC2183

(2183, X'887') API 出口をロードできません。

RC2157

(2157, X'86D') 1 次 ASID とホーム ASID が異なっています。

RC2005

(2005, X'7D5') バッファ長パラメーターは無効です。

RC2219

(2219, X'8AB') 前の呼び出しが完了する前に MQI 呼び出しが入力されました。

RC2487

(2487, X'9B7') コールバック・タイプ・フィールドが正しくない。

RC2448

(2448, X'990') コールバックが登録されていないので、登録解除、中断、または再開できない。

RC2486

(2486, X'9B6') *CallbackFunction* または *CallbackName* のどちらかを指定しなければならないが、両方指定することはできない。

RC2483

(2483, X'9B3') コールバック・タイプ・フィールドが正しくない。

RC2484

(2484, X'9B4') MQCBD オプション・フィールドが正しくない。

RC2140

(2140, X'85C') 待機要求が CICS により拒否された。

RC2009

(2009, X'7D9') キュー・マネージャーとの接続が失われました。

RC2217

(2217, X'8A9') 接続が許可されていません。

RC2202

(2202, X'89A') 接続が静止しています。

RC2203

(2203, X'89B') 接続がシャットダウン中です。

RC2207

(2207, X'89F') 相関 ID のエラー。

RC2010

(2010, X'7DA') データ長パラメーターが無効である。

RC2016

(2016, X'7E0') キューからの読み取りが禁止されている。

RC2351

(2351, X'92F') グローバル作業単位に矛盾がある。

RC2186

(2186, X'88A') 読み取りメッセージ・オプションの構造体が無効である。

RC2353

(2353, X'931') グローバル作業単位のためのハンドルが使用中。

RC2018

(2018, X'7E2') 接続ハンドルが無効です。

RC2019

(2019, X'7E3') オブジェクト・ハンドルが無効です。

RC2259

(2259, X'8D3') ブラウズの指定が不整合である。

RC2245

(2245, X'8C5') 作業単位の指定が不整合である。

RC2246

(2246, X'8C6') カーソル下のメッセージが取り出し対象として無効である。

RC2352

(2352, X'930') グローバル作業単位とローカル作業単位に矛盾がある。

RC2247

(2247, X'8C7') 突き合わせオプションが無効である。

RC2485

(2485, X'9B4') *MaxMsgLength* フィールドが正しくない。

RC2026

(2026, X'7EA') メッセージ記述子が無効である。

RC2497

(2497, X'9C1') 指定された関数入り口点がモジュール中になかった。

RC2496

(2496, X'9C0') モジュールが見つかったが、タイプが間違っている。32 ビットでも 64 ビットでもない。または有効なダイナミック・リンク・ライブラリーではない。

RC2495

(2495, X'9BF') モジュールが検索パス中にないか、またはロードが許可されていない。

RC2250

(2250, X'8CA') メッセージ順序番号が無効である。

RC2331

(2331, X'91B') メッセージ・トークンについて無効な使い方をしている。

RC2033

(2033, X'7F1') メッセージが使用できない。

RC2034

(2034, X'7F2') ブラウズ・カーソルがメッセージに位置付けされていない。

RC2036

(2036, X'7F4') ブラウズのためにキューがオープンされていない。

RC2037

(2037, X'7F5') 入力のためにキューがオープンされていない。

RC2041

(2041, X'7F9') オープンされた後でオブジェクト定義が変更された。

RC2101

(2101, X'835') オブジェクトが損傷しました。

RC2206

(2206, X'89E') API 呼び出し上の命令コードが正しくない。

RC2046

(2046, X'7FE') オプションが無効であるか、矛盾しています。

RC2193

(2193, X'891') ページ・セット・データ・セットへのアクセス中にエラーが発生しました。

RC2052

(2052, X'804') キューが削除されました。

RC2394

(2394, X'95A') キューの索引タイプが間違っている。

RC2058

(2058, X'80A') キュー・マネージャー名が無効であるか、認識されていません。

RC2059

(2059, X'80B') キュー・マネージャーを接続に使用できません。

RC2161

(2161, X'871') キュー・マネージャーが静止しています。

RC2162

(2162, X'872') キュー・マネージャーのシャットダウン中です。

RC2102

(2102, X'836') 使用できるシステム・リソースが不足しています。

RC2069

(2069, X'815') このハンドルに未解決のシグナルがある。

RC2071

(2071, X'817') ストレージが不足しています。

RC2109

(2109, X'83D') 出口プログラムにより呼び出しが抑止されました。

RC2024

(2024, X'7E8') 現行の作業単位内では、これ以上メッセージを処理できない。

RC2072

(2072, X'818') 同期点サポートが利用できない。

RC2195

(2195, X'893') 予期しないエラーが発生しました。

RC2354

(2354, X'932') グローバル作業単位の参加に失敗した。

RC2355

(2355, X'933') 作業単位呼び出しの混合はサポートされていない。

RC2255

(2255, X'8CF') 作業単位がキュー・マネージャーから使用不可。

RC2090

(2090, X'82A') MQGMO での待機間隔が無効である。

RC2256

(2256, X'8D0') 提供された MQGMO のバージョンが違っている。

RC2257

(2257, X'8D1') 提供された MQMD のバージョンが違っている。

RC2298

(2298, X'8FA') 要求された関数は、現在の環境では使用できない。

RPG 宣言

```
C*..1.....2.....3.....4.....5.....6.....7..  
C                                CALLP      MQCB(HCONN : OPERATN : CBDSC :  
                                HOBJ : MSGDSC : GMO :  
                                DATLEN : CMPCOD : REASON)
```

呼び出しのプロトタイプ定義は次のようになります。

DMQCB	PR	EXTPROC('MQCB')
D* Connection handle		
D HCONN		10I 0 VALUE
D* Operation		
D OPERATN		10I 0 VALUE
D* Callback descriptor		
D CBDSC		180A
D* Object handle		
D HOBJ		10I 0 VALUE
D* Message Descriptor		
D MSGDSC		364A
D* Get options		
D GMO		112A
D* Completion code		
D CMPCOD		10I 0
* Reason code qualifying CompCode		
D REASON		10I 0

IBM i IBM i での MQCLOSE (オブジェクトのクローズ)

MQCLOSE 呼び出しは、オブジェクトへのアクセスを解放するもので、MQOPEN 呼び出しの逆です。

- [1278 ページの『構文』](#)
- [1278 ページの『使用上の注意』](#)
- [1279 ページの『パラメーター』](#)
- [1284 ページの『RPG 宣言』](#)

構文

MQCLOSE (*HCONN*, *HOBJ*, *OPTS*, *CMPCOD*, *REASON*)

使用上の注意

1. アプリケーションが MQDISC 呼び出しを発行するか、正常終了または異常終了すると、このアプリケーションによってオープンされたままになっているすべてのオブジェクトは、CONONE オプションで自動的にクローズされます。
2. クローズされるオブジェクトがキューであるときに、以下の点が適用されます。
 - キューに対する操作が作業単位の一部として実行される場合は、そのキューは、同期点の前後のいずれでも、同期点の結果に影響を与えることなくクローズすることができます。
 - OOBROW オプションを指定してオープンされたキューの場合、ブラウザ・カーソルは破壊されます。その後で OOBROW オプションによってキューを再オープンした場合は、新しいブラウザ・カーソルが作成されます (MQOPEN で説明している OOBROW オプションを参照してください)。
 - MQCLOSE 呼び出しを発行した時点で、該当するハンドルに対してメッセージがロックされている場合は、ロックは解除されます ([1080 ページの『IBM i での MQGMO \(読み取りメッセージ・オプション\)』](#)で説明している GMLK オプションを参照してください)。
3. クローズされるオブジェクトが動的キュー (永続または一時) であるとき、以下の点が適用されます。
 - 動的キューの場合、オプション CODEL または COPURG は、対応する MQOPEN 呼び出し上に指定されたオプションに関係なく指定することができます。

- 動的キューが削除されると、キューに対して未解決になっている GMWT オプションを持つすべての MQGET 呼び出しは取り消され、理由コード RC2052 が戻ります。1080 ページの『IBM i での MQGMO (読み取りメッセージ・オプション)』の GMWT オプションを参照してください。

動的キューが削除された場合、MQCLOSE 以外の呼び出しが前に入手した *HOBJ* ハンドルを使用してキューを参照しようとする失敗し、理由コード RC2052 が戻ります。

削除されたキューにアプリケーションからアクセスできなくても、キューを参照するすべてのハンドルがクローズして、キューに影響を与えるすべての作業単位がコミットされるか、またはバックアウトされるまでは、キューはシステムから除去されず、また関連リソースも解放されない点に注意してください。

- 永続動的キューが削除されると、MQCLOSE 呼び出しで指定した *HOBJ* ハンドルが、キューを作成した MQOPEN 呼び出しによって戻されたハンドルではない場合は、MQOPEN 呼び出しを妥当性検査するために使用されたユーザー ID が、キューを削除する許可を持っているかどうか検査されます。MQOPEN 呼び出し時に OOALTU オプションが指定された場合、検査されたユーザー ID は *ODAU* になります。

この検査は以下のような場合は実行されません。

- 指定されたハンドルが、キューを作成した MQOPEN 呼び出しによって戻されたハンドルである場合。
- 削除されるキューが一時動的キューの場合。
- 一時動的キューがクローズされると、MQCLOSE 呼び出しで指定した *HOBJ* ハンドルが、キューを作成した MQOPEN 呼び出しによって戻されたハンドルである場合は、キューは削除されます。これは MQCLOSE 呼び出し時にクローズ・オプションが指定されているか否かにかかわらず起こります。キューにメッセージがある場合、それらは廃棄されます。その際、レポート・メッセージは生成されません。

キューに影響を与える、コミットされていない作業単位がある場合でも、キューとそのメッセージは削除されます。しかし、このことは作業単位が失敗する原因にはなりません。ただし、前述のとおり、各作業単位がコミットされるか、またはバックアウトされるまでは、作業単位に関連するリソースは解放されません。

4. クローズされるオブジェクトが配布リストであるとき、以下の点が適用されます。

- 配布リストに有効なクローズ・オプションは **CONONE** のみです。この呼び出しは、他のオプションが指定されている場合には失敗し、理由コード RC2046 または RC2045 が戻ります。
- 配布リストがクローズされると、リスト内のキューについて個々の完了コードおよび理由コードは戻されません。診断目的に利用できるのは、この呼び出しの **CMPCOD** および **REASON** パラメーターのみです。

いずれかのキューのクローズ時に障害が起こっても、キュー・マネージャーは処理を継続し、配布リスト内の残りのキューをクローズしようとします。次に、この呼び出しの **CMPCOD** および **REASON** パラメーターが、その障害を記述する情報を戻すよう設定されます。このため、キューのほとんどが正常にクローズされた場合でも、完了コードが **CCFAIL** になる場合があります。クローズ中にエラーが発生したキューは、識別されません。

複数のキュー内に障害がある場合、いずれの障害が **CMPCOD** および **REASON** パラメーターに報告されるかは定義されません。

パラメーター

MQCLOSE 呼び出しには、以下のパラメーターがあります。

HCONN (10 桁の符号付き整数) - 入力

接続ハンドル。

このハンドルは、キュー・マネージャーに対する接続を表します。*HCONN* の値は、先行の *MQCONN* または *MQCONN* 呼び出しによって戻されたものです。

HOBJ (10 桁の符号付き整数) - 入出力

オブジェクト・ハンドル

このハンドルは、クローズするオブジェクトを表します。オブジェクトは、どのタイプでも構いません。*HOBJ* の値は、前の MQOPEN 呼び出しで戻されたものです。

呼び出しが正常に完了すると、キュー・マネージャーは、環境に対して有効なハンドルでない値にこのパラメーターを設定します。値は、以下のとおりです。

HOUNUH

使用できないオブジェクト・ハンドル。

OPTS (10 桁の符号付き整数) - 入力

MQCLOSE のアクションを制御するオプション。

OPTS パラメーターのオブジェクトのクローズ方法を制御します。複数の方法でクローズできるのは、永続動的キューおよびサブスクリプションのみです。永続動的キューは保存することも削除することもできます。永続動的キューとは、**DefinitionType** 属性の値が QDPERM であるキューを指します (1380 ページの『キューの属性』の **DefinitionType** 属性を参照してください)。クローズ・オプションは、このトピックの後の表に要約されています。

永続サブスクリプションは保持することも除去することもできます。これらは、SODUR オプションを指定した MQSUB 呼び出しを使用して作成されます。

管理対象の宛先 (SOMAN オプションを使用した MQSUB 呼び出しで戻される **Hobj** パラメーター) へのハンドルを閉じる場合、関連したサブスクリプションも除去されたときには、キュー・マネージャーは取得されていないパブリケーションをすべてクリーンアップします。これは、MQSUB 呼び出しで戻される **Hsub** パラメーターで CORMSB オプションを使用することによって行われます。非永続サブスクリプションの場合、CORMSB は MQCLOSE でのデフォルトの振る舞いであることに注意してください。

非管理対象の宛先へのハンドルを閉じる場合、パブリケーションが送信されるキューのクリーンアップはユーザーの責任で行います。まず CORMSB を使用してサブスクリプションを閉じてから、キューに何も残らなくなるまでメッセージを処理することをお勧めします。

以下のいずれかを 1 つのみ指定する必要があります。

動的キュー・クローズ・オプション

これらのオプションは、永続動的キューを閉じる方法を制御します。

CODEL

キューを削除します。

以下の条件のいずれかが真の場合、キューは削除されます。

- 前の MQOPEN 呼び出しによって作成された永続動的キューであり、メッセージ、およびキューに対して未解決になっているコミットされていない読み取り要求または書き込み要求がない (現行タスクまたは任意のタスクのための)。
- *HOBJ* を戻す MQOPEN 呼び出しにより作成された一時動的キューである。この場合、キューに入っているすべてのメッセージは消去されます。

その他の場合 (MQSUB 呼び出しで *Hobj* が返された場合を含む) はすべて、呼び出しは理由コード RC2045 で失敗し、オブジェクトは削除されません。

COPURG

キューを削除し、そこに入っているすべてのメッセージを除去します。

以下の条件のいずれかが真の場合、キューは削除されます。

- 前の MQOPEN 呼び出しで作成された永続動的キューであり、キューに対して未解決になっているコミットされていない読み取り要求または書き込み要求がない (現行タスクまたはそれ以外の任意のタスクのための)。
- *HOBJ* を戻す MQOPEN 呼び出しにより作成された一時動的キューである。

その他の場合 (MQSUB 呼び出しで *Hobj* が返された場合を含む) はすべて、呼び出しは理由コード RC2045 で失敗し、オブジェクトは削除されません。

次の表は、どのクローズ・オプションが有効であるか、およびオブジェクトを残すか削除するかを示しています。

表 782. 保存または削除されたオブジェクトで使用するための有効なクローズ・オプション			
オブジェクトまたはキューのタイプ	CONONE	CODEL	COPURG
キュー以外のオブジェクト	保存	無効	無効
事前定義されたキュー	保存	無効	無効
永続動的キュー	保存	空で、保留中の更新がない場合は削除	メッセージを削除。保留中の更新がないキューを削除
一時動的キュー (キューの作成者から発行された呼び出し)	削除	削除	削除
一時動的キュー (キューの作成者から発行されていない呼び出し)	保存	無効	無効
配布リスト	保存	無効	無効
管理対象サブスクリプション宛先	保存	無効	無効
配布リスト (サブスクリプションは除去済み)	メッセージを削除。キューを削除。	無効	無効

サブスクリプションのクローズ・オプション

これらのオプションは、ハンドルのクローズ時に永続サブスクリプションを除去するかどうか、およびアプリケーションによる読み取りを待機中のパブリケーションをクリーンアップするかどうかを制御します。これらのオプションは、MQSUB 呼び出しの **Hsub** パラメーターで戻されるオブジェクト・ハンドルで使用する場合にのみ有効です。

COKPSB

サブスクリプションに対するハンドルはクローズされますが、作成されたサブスクリプションは保持されます。パブリケーションは引き続き、サブスクリプションで指定された宛先に送られます。このオプションは、オプション **SODUR** を指定してサブスクリプションが行われた場合のみ有効です。サブスクリプションが永続的である場合、**COKPSB** がデフォルトです。

CORMSB

サブスクリプションは除去され、サブスクリプションに対するハンドルはクローズされます。

MQSUB 呼び出しの **Hobj** パラメーターは **Hsub** パラメーターの閉止によって無効にされず、残りのパブリケーションを受け取るために引き続き **MQGET** または **MQCB** で使用することができます。MQSUB 呼び出しの **Hobj** パラメーターもクローズされると、それが管理対象宛先だった場合、取り出されていないパブリケーションはすべて除去されます。

サブスクリプションが非永続的である場合、**CORMSB** がデフォルトです。

これらのサブスクリプションのクローズ・オプションは、以下の表に要約されます。

永続サブスクリプションのハンドルをクローズしてサブスクリプションをそのままにしておくには、以下のサブスクリプションのクローズ・オプションを使用します。

表 783. 永続サブスクリプション・ハンドルを閉じて、そのサブスクリプションを終了するための作業オプション	
タスク	サブスクリプション閉止オプション
MQOPENed ハンドルのパブリケーションを保持する	COKPSB
MQOPENed ハンドルのパブリケーションを除去する	アクションは許可されていない
SOMAN を使用するハンドルのパブリケーションを保持する	COKPSB
SOMAN を使用するハンドルのパブリケーションを除去する	アクションは許可されていない

永続サブスクリプション・ハンドルを閉じてアンサブスクライブするか、または非永続サブスクリプション・ハンドルを閉じることによってアンサブスクライブを行うには、以下のサブスクリプションのクローズ・オプションを使用します。

表 784. アンサブスクライブの作業オプション	
タスク	サブスクリプション閉止オプション
MQOPENed ハンドルのパブリケーションを保持する	CORMSB
MQOPENed ハンドルのパブリケーションを除去する	アクションは許可されていない
SOMAN を使用するハンドルのパブリケーションを保持する	CORMSB
SOMAN を使用するハンドルのパブリケーションを除去する	COPGSB

先読みオプション

以下のオプションは、アプリケーションが要求する前にクライアントに送信されたものの、アプリケーションによってまだ消費されていない非永続メッセージに実行されることを制御します。これらのメッセージは、クライアント先読みバッファに格納されてアプリケーションによる要求を待機し、MQCLOSE が完了する前にキューから廃棄または消費することができます。

COIMM

オブジェクトは即時にクローズされ、アプリケーションが要求する前にクライアントに送信されたメッセージはすべて廃棄されます。アプリケーションがこのメッセージを消費することはできません。これがデフォルト値です。

COQSC

オブジェクトのクローズ要求は行われますが、アプリケーションが要求する前にクライアントに送信されたメッセージがあれば、引き続きクライアント先読みバッファに存在し、MQCLOSE 呼び出しは警告コード RC2458 で戻りますが、オブジェクト・ハンドルは有効なままです。

アプリケーションは引き続きそのオブジェクト・ハンドルを使用することができ、メッセージがなくなるまで取り出しを続行します。その後、オブジェクトを再びクローズします。アプリケーションの要求前にクライアントに送信されるメッセージがなくなると、先読みはオフになります。

COIMM が使用された場合に廃棄されるメッセージが最後の MQGET 呼び出しと続く MQCLOSE の間に着信する可能性があるので、アプリケーションでは、クライアント先読みバッファ内にメッセージがないという点に到達しようとするよりも、COQSC を使用することが推奨されています。

COQSC を使用して MQCLOSE が非同期コールバック関数内から発行される場合は、先読みメッセージの同じ動作が適用されます。警告コード RC2458 が返される場合、コールバック関数は少なくとももう一度呼び出されます。先読みされた最後に残ったメッセージがコールバック関数に渡されると、CBCFLG フィールドは CBCFBE に設定されます。

デフォルト・オプション

上記のいずれのオプションも必要ない場合、次のオプションを使用できます。

CONONE

オプションのクローズ処理は不要である。

これは、次のものに対して指定しなければなりません。

- キュー以外のオブジェクト
- 事前定義キュー
- 一時動的キュー (ただし、*HOBJ* が、キューを作成した *MQOPEN* 呼び出しにより戻されるハンドルではない場合のみ)
- 配布リスト

上記の場合はすべて、オブジェクトは残され、削除されません。

このオプションが一時動的キューに対して指定されていると、次のようになります。

- *HOBJ* を戻した *MQOPEN* 呼び出しにより作成されたキューは、削除されます。そして、そのキュー内にあるメッセージはすべて除去されます。
- 上記以外の場合、キュー (およびキュー内のすべてのメッセージ) は保存されます。

永続動的キューに対してこのオプションが指定されていると、キューは残され、削除されません。

CMPCOD (10 桁の符号付き整数) - 出力

完了コード。

これは、以下のいずれかになります。

CCOK

正常終了。

CCWARN

警告 (部分完了)。

CCFAIL

呼び出し失敗。

REASON (10 桁の符号付き整数) - 出力

CMPCOD を限定する理由コード。

CMPCOD が *CCOK* の場合

RCNONE

(0, X'000') レポートする理由コードはありません。

CMPCOD が *CCWARN* の場合

RC2241

(2241, X'8C1') メッセージ・グループが不完全である。

RC2242

(2242, X'8C2') 論理メッセージが不完全である。

CMPCOD が *CCFAIL* の場合

RC2219

(2219, X'8AB') 前の呼び出しが完了する前に *MQI* 呼び出しが再入力されました。

RC2009

(2009, X'7D9') キュー・マネージャーとの接続が失われました。

RC2018

(2018, X'7E2') 接続ハンドルが無効です。

RC2019

(2019, X'7E3') オブジェクト・ハンドルが無効です。

RC2035

(2035, X'7F3') アクセスは許可されません。

RC2101

(2101, X'835') オブジェクトが損傷しました。

RC2045

(2045, X'7FD') オプションが、オブジェクト・タイプとして無効です。

RC2046

(2046, X'7FE') オプションが無効であるか、矛盾しています。

RC2058

(2058, X'80A') キュー・マネージャー名が無効であるか、認識されていません。

RC2059

(2059, X'80B') キュー・マネージャーを接続に使用できません。

RC2162

(2162, X'872') キュー・マネージャーのシャットダウン中です。

RC2055

(2055, X'807') メッセージ、またはコミットされていない 書き込み要求か取得要求が、1 つ以上キューに入っています。

RC2102

(2102, X'836') 使用できるシステム・リソースが不足しています。

RC2063

(2063, X'80F') セキュリティー・エラーが発生しました。

RC2071

(2071, X'817') ストレージが不足しています。

RC2195

(2195, X'893') 予期しないエラーが発生しました。

RPG 宣言

```
C*..1.....2.....3.....4.....5.....6.....7..
C                                CALLP      MQCLOSE(HCONN : HOBJ : OPTS :
C                                CMPCOD : REASON)
```

呼び出しのプロトタイプ定義は次のようになります。

```
D*..1.....2.....3.....4.....5.....6.....7..
DMQCLOSE          PR          EXTPROC('MQCLOSE')
D* Connection handle
D HCONN              10I 0 VALUE
D* Object handle
D HOBJ              10I 0
D* Options that control the action of MQCLOSE
D OPTS              10I 0 VALUE
D* Completion code
D CMPCOD            10I 0
D* Reason code qualifying CMPCOD
D REASON            10I 0
```

IBM i IBM i での MQCMIT (変更のコミット)

MQCMIT 呼び出しは、アプリケーションが同期点に達したこと、および最後の同期点以降に発生したメッセージの読み取りと書き込みをすべて永続化することをキュー・マネージャーに示します。作業単位の一部として書き込まれたメッセージは、他のアプリケーションで使用できるようになります。作業単位の一部として取り出されたメッセージは削除されます。

- [1285 ページの『構文』](#)
- [1285 ページの『使用上の注意』](#)
- [1286 ページの『パラメーター』](#)
- [1287 ページの『RPG 宣言』](#)

構文

MQCMIT (HCONN, COMCOD, REASON)

使用上の注意

MQCMIT を使用する際には、以下の使用上の注意を考慮してください。

1. この呼び出しは、キュー・マネージャーそのものが作業単位を調整するときのみ使用できます。これはローカル作業単位で、変更内容が IBM MQ リソースに対してのみ影響を及ぼします。
2. キュー・マネージャーが作業単位を調整しない環境では、MQCMIT ではなく適切なコミット呼び出しを使用する必要があります。この環境ではまた、アプリケーションの正常終了を原因とする暗黙的コミットをサポートしている場合もあります。
 - IBM i では、この呼び出しはキュー・マネージャーで調整されるローカル作業単位で使うことができます。これは、コミットメント定義がジョブ・レベルで存在してはならないことを意味します。つまり、**CMTSCOPE(*JOB)** パラメーターを指定した STRCMTCTL コマンドがジョブに対して発行されていることはありません。
3. 作業単位内にあるコミットされていない変更内容でアプリケーションが終了する場合、それらの変更内容の後処理は、そのアプリケーションが正常に終了するか、異常終了するかで異なります。詳細については、[1300 ページの『IBM i での MQDISC \(キュー・マネージャーの切断\)』](#)の使用上の注意を参照してください。
4. アプリケーションでグループ内のメッセージまたは論理メッセージのセグメントの書き込みまたは読み取りを行う場合、キュー・マネージャーは、最後に MQPUT および MQGET 呼び出しが正常に実行されたメッセージ・グループに関する情報を保存します。この情報は、キュー・ハンドルに関する次のような情報です。
 - MQMD 中の MDGID、MDSEQ、MDOFF、および MDMFL フィールドの値。
 - そのメッセージが作業単位の一部であるかどうか。
 - MQPUT 呼び出しについて、そのメッセージが持続メッセージか、非持続メッセージか。

作業単位がコミットされると、キュー・マネージャーは、グループおよびセグメント情報を保持し、アプリケーションは現行メッセージ・グループまたは論理メッセージの書き込みまたは読み取りを継続することができます。

1 つの作業単位のコミット時にグループおよびセグメント情報を保持することにより、アプリケーションは、大きなメッセージ・グループまたは数多くのセグメントで構成される大きな論理メッセージを、いくつかの作業単位にスプレッドできます。ローカル・キュー・マネージャーのキュー・ストレージが限られている場合には、いくつかの作業単位を使用の方が有効となる場合があります。ただし、システム障害の発生時に各メッセージの書き込みまたは読み取りを正しい時点で再始動できるようにするには、アプリケーションが十分な情報を維持する必要があります。システム障害後に正しい時点から再始動する方法の詳細については、[1180 ページの『IBM i での MQPMO \(メッセージ書き込みオプション\)』](#)の PMLOGO オプション、および [1080 ページの『IBM i での MQGMO \(読み取りメッセージ・オプション\)』](#)の GMLOGO オプションを参照してください。

5. アプリケーションが作業単位内に永続メッセージのプットを含み、MQPUT または MQPUT1 呼び出しで MQPMO_ASYNC_RESPONSE オプションが使用され、かつプットが失敗した場合、MQCMIT は MQCC_FAILED、MQRC_BACKED_OUT を返します。[MQPMO のフィールド詳細](#)にある MQPMO_ASYNC_RESPONSE の説明を参照してください。

同様のシナリオにおいて、作業単位内の非永続メッセージに対して、MQCMIT は MQCC_OK、MQRC_NONE を返します。

複数のプットが存在する場合で、MQPMO_ASYNC_RESPONSE が使用されていた場合、いずれかの永続メッセージのプットに失敗すると、MQCMIT 呼び出しは MQCC_FAILED、MQRC_BACKED_OUT を返します。

次の『使用上の注意』は、キュー・マネージャーで作業単位を調整する場合にのみ適用されます。

1. 作業単位の 1 つには、1 つの接続ハンドルと同じ有効範囲があります。つまり、特定の作業単位に影響を与える IBM MQ 呼び出しはすべて、同じ接続ハンドルを使用して実行しなければなりません。別の接

続ハンドルを用いて呼び出しを発行すると (例えば、別のアプリケーションで呼び出しを発行する)、別の作業単位に影響が及びます。接続ハンドルの有効範囲については、MQCONN の項で説明している **HCONN** パラメーターを参照してください。

2. この呼び出しで影響を受けるメッセージは、現行の作業単位の一部として書き込まれたメッセージ、または取り出されたメッセージに限られます。
3. 長時間実行しているアプリケーションで、1つの作業単位に対して MQGET、MQPUT、または MQPUT1 呼び出しを発行している場合、コミット呼び出しまたはバックアウト呼び出しを一度も発行しないと、キューが他のアプリケーションでは使用できないメッセージで満杯になることがあります。これを避けるには、管理者は、ランナウェイ・アプリケーションがキューをいっぱいにしない程度に低く、かつ必要なメッセージング・アプリケーションが正常に機能する程度に高い値を **MaxUncommittedMsgs** キュー・マネージャー属性に設定しなければなりません。

パラメーター

MQCMIT 呼び出しには、以下のパラメーターがあります。

HCONN (10 桁の符号付き整数) - 入力

接続ハンドル。

このハンドルは、キュー・マネージャーに対する接続を表します。HCONN の値は、先行の MQCONN または MQCONNX 呼び出しによって戻されたものです。

COMCOD (10 桁の符号付き整数) - 出力

完了コード。

これは、以下のいずれかになります。

CCOK

正常終了。

CCWARN

警告 (部分完了)。

CCFAIL

呼び出し失敗。

REASON (10 桁の符号付き整数) - 出力

COMCOD を限定する理由コード。

COMCOD が CCOK の場合

RCNONE

(0, X'000') レポートする理由コードはありません。

COMCOD が CCWARN の場合

RC2003

(2003, X'7D3') 作業単位がバックアウトされた。

RC2124

(2124, X'84C') コミット操作の結果が保留状態である。

COMCOD が CCFAIL の場合

RC2219

(2219, X'8AB') 前の呼び出しが完了する前に MQI 呼び出しが再入力されました。

RC2009

(2009, X'7D9') キュー・マネージャーとの接続が失われました。

RC2018

(2018, X'7E2') 接続ハンドルが無効です。

RC2101

(2101, X'835') オブジェクトが損傷しました。

RC2123

(2123, X'84B') コミットまたはバックアウト操作の結果が混在している。

RC2162

(2162, X'872') キュー・マネージャーのシャットダウン中です。

RC2102

(2102, X'836') 使用できるシステム・リソースが不足しています。

RC2071

(2071, X'817') ストレージが不足しています。

RC2195

(2195, X'893') 予期しないエラーが発生しました。

RPG 宣言

```
C*..1.....2.....3.....4.....5.....6.....7..  
C                                CALLP      MQCMIT(HCONN : COMCOD : REASON)
```

呼び出しのプロトタイプ定義は次のようになります。

```
D*..1.....2.....3.....4.....5.....6.....7..  
DMQCMIT      PR      EXTPROC('MQCMIT')  
D* Connection handle  
D HCONN                                10I 0 VALUE  
D* Completion code  
D COMCOD                                10I 0  
D* Reason code qualifying COMCOD  
D REASON                                10I 0
```

IBM i IBM i での MQCONN (キュー・マネージャーの接続)

MQCONN 呼び出しは、アプリケーション・プログラムをキュー・マネージャーに接続します。この呼び出しは、キュー・マネージャー接続ハンドルを提供します。この接続ハンドルは後続のメッセージ・キューイング呼び出しでアプリケーションによって使用されます。

- アプリケーションは、キュー・マネージャーに接続するには MQCONN 呼び出しまたは MQCONNX 呼び出しを使用し、キュー・マネージャーから切断するには MQDISC 呼び出しを使用する必要があります。

IBM MQ for Multiplatforms では、アプリケーションの各スレッドが異なるキュー・マネージャーに接続できます。他のシステムの場合、プロセス内のすべての同時接続は同じキュー・マネージャーに対するものでなければなりません。

- [1287 ページの『構文』](#)
- [1287 ページの『使用上の注意』](#)
- [1288 ページの『パラメーター』](#)
- [1291 ページの『RPG 宣言』](#)

構文

MQCONN (QMNAME, HCONN, CMPCOD, REASON)

使用上の注意

1. MQCONN 呼び出しを使用して接続が行われるキュー・マネージャーを、ローカル・キュー・マネージャーと呼びます。
2. ローカル・キュー・マネージャーが所有するキューは、アプリケーションでは、ローカル・キューとして扱われます。これらのキューに対しては、メッセージの書き込みと読み取りが可能です。

ローカル・キュー・マネージャーが所属するキュー共有グループが所有する共有キューは、アプリケーションでは、ローカル・キューとして扱われます。これらのキューに対しては、メッセージの書き込みと読み取りが可能です。

リモート・キュー・マネージャーが所有するキューは、リモート・キューとして扱われます。これらキューに対しては、メッセージに書き込み可能ですが、メッセージの読み取りはできません。

3. アプリケーションの実行中にキュー・マネージャーで障害が発生した場合、アプリケーションは後続の IBM MQ 呼び出しで使用する新しい接続ハンドルを取得するために、再び MQCONN 呼び出しを発行する必要があります。アプリケーションは、呼び出しが成功するまで定期的に MQCONN 呼び出しを発行することができます。

アプリケーションはキュー・マネージャーに接続されているかどうか分からない場合でも、接続ハンドルを取得するために MQCONN 呼び出しを発行しても問題ありません。アプリケーションが既に接続されている場合、返されるハンドルは前に発行した MQCONN 呼び出しによって返されたハンドルと同じですが、完了コード CCWARN と理由コード RC2002 が一緒に返されます。

4. アプリケーションによる IBM MQ 呼び出しの使用が終了した場合、アプリケーションで MQDISC 呼び出しを使用してキュー・マネージャーから切断する必要があります。
5. IBM i では、異常終了するプログラムは、キュー・マネージャーから自動的に切断されません。したがって、アプリケーションは、MQCONN または MQCONNEX 呼び出しが完了コード CCWARN および理由コード RC2002 を戻す可能性を考慮に入れて作成しなければなりません。この場合に戻された接続ハンドルは、正常なものとして使用できます。

パラメーター

MQCONN 呼び出しには、以下のパラメーターがあります。

QMNAME (48 バイトの文字ストリング) - 入力

キュー・マネージャーの名前。

これは、アプリケーションが接続先にしたいキュー・マネージャーの名前です。この名前には、以下に示す文字を使用できます。

- 英大文字 (A から Z まで)
- 英小文字 (a から z まで)
- 数字 (0 から 9 まで)
- ピリオド (.), スラッシュ (/), 下線 (_), パーセント (%)

名前の先頭を空白にしたり、名前に空白を埋め込んだりすることはできませんが、名前の後に空白を入れることはできます。ヌル文字を使用して、名前の中における有効なデータの末尾を示すことができます。ヌル文字とそれに続く文字はすべて空白として扱われます。以下に示す制約事項は、それぞれ明記している環境に適用されます。

- IBM i で英小文字、スラッシュ、パーセントの各文字が含まれている名前をコマンドに指定する場合は、それを引用符で囲む必要があります。これらの引用符は、QMNAME パラメーターでは指定しないでください。

名前全体が空白で構成されている場合は、デフォルトのキュー・マネージャーの名前が使用されます。

QMNAME に指定される名前は、接続可能なキュー・マネージャーの名前でなければなりません。

キュー共有グループ:複数のキュー・マネージャーが存在し、キュー共有グループを構成するように設定されているシステムでは、キュー・マネージャーの名前の代わりに QMNAME にキュー共有グループの名前を指定できます。これにより、アプリケーションは、キュー共有グループで使用可能な任意のキュー・マネージャーに接続できます。また、空白の QMNAME によってデフォルト・キュー・マネージャーの代わりにキュー共有グループに接続するようにシステムを構成することもできます。

QMNAME がキュー共有グループの名前を指定している場合に、その名前のキュー・マネージャーもシステムに存在する場合は、前者よりも後者を優先して接続を行います。その接続が失敗した場合のみ、キュー共有グループ内のキュー・マネージャーの 1 つへの接続が試行されます。

接続に成功した場合は、MQCONN または MQCONNX 呼び出しによって戻されたハンドルを使用し、接続先の特定のキュー・マネージャーに属するすべてのリソース (共有と非共有の両方) にアクセスできます。これらのリソースへのアクセスは、通常の許可制御に従います。

アプリケーションが、並行接続を確立するために MQCONN または MQCONNX 呼び出しを 2 回発行し、一方または両方の呼び出しでキュー共有グループの名前を指定する場合は、2 番目の呼び出しで完了コード CCWARN および理由コード RC2002 が戻される場合があります。これは、2 番目の呼び出しで最初の呼び出しと同じキュー・マネージャーに接続する場合に発生します。

キュー共有グループは、z/OS でのみサポートされています。キュー共有グループへの接続は、バッチ、RRS バッチ、および TSO 環境でのみサポートされます。

IBM MQ クライアント・アプリケーション: IBM MQ MQI client・アプリケーションの場合、指定されたキュー・マネージャー名を含む各クライアント接続チャネル定義を使用して、成功するまで接続を試行します。ただし、キュー・マネージャーは、指定された名前と同じでなければなりません。名前全体がブランクで指定されている場合は、すべてブランクのキュー・マネージャー名を持つ各クライアント接続チャネルが、成功するまで試行されます。この場合、キュー・マネージャーの実際の名前についての検査は行われません。

IBM MQ クライアント・キュー・マネージャー・グループ: 指定の名前がアスタリスク (*) で始まっている場合、接続される実際のキュー・マネージャーの名前は、アプリケーションで指定された名前とは異なる場合があります。指定された名前 (アスタリスクなし) は、接続可能なキュー・マネージャーのグループを定義します。その実施では、1 個ずつ順に英字順で試行しながら、接続できるものが見つかるまで、各グループから 1 つずつ選択していきます。グループ内のキュー・マネージャーがどれも接続に使用できない場合、呼び出しは失敗します。各キュー・マネージャーは、1 回のみ試行されます。名前にアスタリスクのみが指定されている場合は、実施によって定義されたデフォルトのキュー・マネージャー・グループが使用されます。

キュー・マネージャー・グループは MQ クライアント環境で実行されるアプリケーションについてのみサポートされます。非クライアント・アプリケーションがアスタリスクで始まるキュー・マネージャー名を指定すると、この呼び出しは失敗します。グループ内の各キュー・マネージャーと通信するために、いくつかのクライアント接続チャネル定義に同じキュー・マネージャー名 (アスタリスクなしで指定された名前) を指定することによって、グループが定義されます。デフォルト・グループの定義は、1 つ以上のクライアント接続チャネル定義に、それぞれブランクのキュー・マネージャー名を付けることによって行われます (したがって、ブランクのみの名前を定義することは、クライアント・アプリケーションの名前に単一アスタリスクを指定するのと同様です)。

グループの 1 つのキュー・マネージャーに接続したあと、アプリケーションは、メッセージおよびオブジェクト記述子の中のキュー・マネージャー名フィールドに、通常の方法でブランクを指定して、アプリケーションが実際に接続されたキュー・マネージャーの名前 (ローカル・キュー・マネージャー) を意味することができます。アプリケーションがこの名前を認識する必要がある場合は、MQINQ 呼び出しを発行して、QMGrName キュー・マネージャー属性を照会することができます。

接続名の先頭にアスタリスクを付けることは、アプリケーションがグループ内の特定のキュー・マネージャーと接続されていることを前提としないことを意味します。適切なアプリケーションは以下のとおりです。

- メッセージは書き込むが、メッセージを読み取らないアプリケーション。
- 要求メッセージを書き込んでから、一時動的 キューから応答メッセージを取得するアプリケーション。

不適当なアプリケーションは、特定のキュー・マネージャーの特定のキューからメッセージを読み取らなければならないアプリケーションなどです。このようなアプリケーションでは、名前の先頭にはアスタリスクを付けないでください。

アスタリスクが指定されている場合、名前の残りの最大の長さは、47 文字です。

このパラメーターの長さは、LNQMNM で指定されます。

HCONN (10 桁の符号付き整数) - 出力

接続ハンドル。

このハンドルは、キュー・マネージャーに対する接続を表します。アプリケーションで発行されるその後のすべてのメッセージ・キューイング呼び出しで、指定する必要があります。MQDISC 呼び出しが発行されたとき、またはハンドルの有効範囲を定義する処理の単位が終了したときに、有効でなくなります。

ハンドルの有効範囲は、以下の最小単位に制限されます。アプリケーションが実行されているプラットフォームによってサポートされる並列処理。ハンドルは、MQCONN 呼び出しが発行された並列処理の単位の外側では無効です。

- IBM i の場合、ハンドルの有効範囲は、呼び出しを発行するジョブです。

CMPCOD (10 桁の符号付き整数) - 出力

完了コード。

これは、以下のいずれかになります。

CCOK

正常終了。

CCWARN

警告 (部分完了)。

CCFAIL

呼び出し失敗。

REASON (10 桁の符号付き整数) - 出力

CMPCOD を限定する理由コード。

CMPCOD が CCOK の場合

RCNONE

(0, X'000') レポートする理由コードはありません。

CMPCOD が CCWARN の場合

RC2002

(2002, X'7D2') アプリケーションはすでに接続されています。

CMPCOD が CCFAIL の場合

RC2219

(2219, X'8AB') 前の呼び出しが完了する前に MQI 呼び出しが再入力されました。

RC2267

(2267, X'8DB') クラスター・ワークロード出口をロードできません。

RC2009

(2009, X'7D9') キュー・マネージャーとの接続が失われました。

RC2018

(2018, X'7E2') 接続ハンドルが無効です。

RC2035

(2035, X'7F3') アクセスは許可されません。

RC2137

(2137, X'859') オブジェクトが正常にオープンされていません。

RC2058

(2058, X'80A') キュー・マネージャー名が無効であるか、認識されていません。

RC2059

(2059, X'80B') キュー・マネージャーを接続に使用できません。

RC2161

(2161, X'871') キュー・マネージャーが静止しています。

RC2162

(2162, X'872') キュー・マネージャーのシャットダウン中です。

RC2102

(2102, X'836') 使用できるシステム・リソースが不足しています。

RC2063

(2063, X'80F') セキュリティー・エラーが発生しました。

RC2071

(2071, X'817') ストレージが不足しています。

RC2195

(2195, X'893') 予期しないエラーが発生しました。

RPG 宣言

```
C*..1.....2.....3.....4.....5.....6.....7..  
C                                CALLP      MQCONN(QMNAME : HCONN : CMPCOD :  
C                                REASON)
```

呼び出しのプロトタイプ定義は次のようになります。

```
D*..1.....2.....3.....4.....5.....6.....7..  
DMQCONN          PR          EXTPROC('MQCONN')  
D* Name of queue manager  
D QMNAME          48A  
D* Connection handle  
D HCONN          10I 0  
D* Completion code  
D CMPCOD          10I 0  
D* Reason code qualifying CMPCOD  
D REASON          10I 0
```

IBM i IBM i での MQCONN (キュー・マネージャーの接続 (拡張))

MQCONN 呼び出しは、アプリケーション・プログラムをキュー・マネージャーに接続します。この呼び出しによって取得したキュー・マネージャー接続ハンドルを使用して、アプリケーションは、それ以降の、IBM MQ 呼び出しを実行します。

MQCONN 呼び出しは、MQCONN 呼び出しに似ていますが、MQCONN では、呼び出しの動作を制御するオプションを指定できます。

IBM MQ for Multiplatforms では、アプリケーションの各スレッドが異なるキュー・マネージャーに接続できます。他のシステムの場合、プロセス内のすべての同時接続は同じキュー・マネージャーに対するものでなければなりません。

- [1291 ページの『構文』](#)
- [1291 ページの『パラメーター』](#)
- [1292 ページの『RPG 宣言』](#)

構文

MQCONN (QMNAME, CNOPT, HCONN, CMPCOD, REASON)

パラメーター

MQCONN 呼び出しには、以下のパラメーターがあります。

QMNAME (48 バイトの文字ストリング) - 入力

キュー・マネージャーの名前。

詳しくは、[1287 ページの『IBM i での MQCONN \(キュー・マネージャーの接続\)』](#) で説明されている **QMNAME** パラメーターを参照してください。

CNOPT (MQCNO) - 入出力

MQCONN のアクションを制御するオプション。

詳細は 1050 ページの『[IBM i での MQCNO \(接続オプション\)](#)』を参照してください。

HCONN (10 桁の符号付き整数) - 出力

接続ハンドル。

詳しくは、1287 ページの『[IBM i での MQCONN \(キュー・マネージャーの接続\)](#)』で説明されている **HCONN** パラメーターを参照してください。

CMPCOD (10 桁の符号付き整数) - 出力

完了コード。

詳しくは、1287 ページの『[IBM i での MQCONN \(キュー・マネージャーの接続\)](#)』で説明されている **CMPCOD** パラメーターを参照してください。

REASON (10 桁の符号付き整数) - 出力

CMPCOD を限定する理由コード。

戻される理由コードの詳細については、1287 ページの『[IBM i での MQCONN \(キュー・マネージャーの接続\)](#)』の **REASON** パラメーターを参照してください。

MQCONN 呼び出しから返される追加の理由コードは、以下のとおりです。

CMPCOD が CCFAIL の場合

RC2278

(2278, X'8E6') クライアント接続フィールドが無効です。

RC2139

(2139, X'85B') 接続オプション構造体が無効です。

RC2046

(2046, X'7FE') オプションが無効であるか、矛盾しています。

RPG 宣言

```
C*.1.....2.....3.....4.....5.....6.....7..
C                                CALLP      MQCONN(QMNAME : HCONN : CMPCOD :
C                                REASON)
```

呼び出しのプロトタイプ定義は次のようになります。

```
D*.1.....2.....3.....4.....5.....6.....7..
DMQCONN          PR          EXTPROC('MQCONN')
D* Name of queue manager
D QMNAME          48A
D* Options that control the action of MQCONN
D HCONN          224A
D* Connection handle
D HCONN          10I 0
D* Completion code
D CMPCOD          10I 0
D* Reason code qualifying CMPCOD
D REASON          10I 0
```

IBM i IBM i での MQCRTMH (メッセージ・ハンドルの作成)

MQCRTMH 呼び出しは、メッセージ・ハンドルを戻します。

アプリケーションが後続のメッセージ・キューイング呼び出し上でこの呼び出しを使用できます。

- [MQSETMP](#) 呼び出しを使用して、メッセージ・ハンドルのプロパティを設定します。
- [MQINQMP](#) 呼び出しを使用して、メッセージ・ハンドルのプロパティの値を照会します。
- [MQDLTMP](#) 呼び出しを使用して、メッセージ・ハンドルのプロパティを削除します。

メッセージ・ハンドルを MQPUT および MQPUT1 呼び出しで使用する、メッセージ・ハンドルのプロパティを、書き込まれるメッセージのプロパティと関連付けることができます。同様に、メッセージ・ハンドルを MQGET 呼び出しに指定することによって、MQGET 呼び出しの完了時に、メッセージ・ハンドルを使用して、取得されるメッセージのプロパティにアクセスできます。

[MQDLTMH](#) を使用してメッセージ・ハンドルを削除します。

- [1293 ページの『構文』](#)
- [1293 ページの『パラメーター』](#)
- [1295 ページの『RPG 宣言』](#)

構文

MQCRTMH (*Hconn*, *CrtMsgHOpts*, *Hmsg*, *CompCode*, *Reason*)

パラメーター

MQCRTMH 呼び出しには、以下のパラメーターがあります。

HCONN (10 桁の符号付き整数) - 入力

このハンドルは、キュー・マネージャーに対する接続を表します。*HCONN* の値は、先行の MQCONN または MQCONNX 呼び出しによって戻されたものです。キュー・マネージャーへの接続が無効になり、メッセージ・ハンドル上で IBM MQ 呼び出しが作動していない場合は、[MQDLTMH](#) が暗黙的に呼び出されてメッセージを削除します。

あるいは、以下の値を指定することができます。

HCUNAS

接続ハンドルは特定のキュー・マネージャーに対する接続を表しません。

この値を使用する場合、メッセージ・ハンドルに割り振られたストレージを解放するために、[MQDLTMH](#) を明示的に呼び出してメッセージ・ハンドルを削除する必要があります。IBM MQ が暗黙的にメッセージ・ハンドルを削除することはありません。

メッセージ・ハンドルを作成するスレッドでは、キュー・マネージャーへの有効な接続が少なくとも 1 つ確立されている必要があります。接続がないと、呼び出しは RC2018 で失敗します。

CRTOPT (MQCMHO) - 入力

MQCRTMH のアクションを制御するオプション。詳細については、[MQCMHO](#) を参照してください。

HMSG (20 桁の符号付き整数) - 出力

出力で、メッセージ・ハンドルのプロパティの設定、照会、および削除に使用できるメッセージ・ハンドルが戻されます。当初、メッセージ・ハンドルにプロパティは含まれていません。

メッセージ・ハンドルには、メッセージ記述子も関連付けられます。最初は、このメッセージ記述子にはデフォルト値が含まれています。MQSETMP 呼び出しと MQINQMP 呼び出しを使用して、関連付けられたメッセージ記述子フィールドの値を設定したり、照会したりできます。MQDLTMP 呼び出しは、メッセージ記述子のフィールドをリセットして、デフォルト値に戻します。

HCONN パラメーターの値として HCUNAS が指定されている場合には、処理単位内の接続で、返されたメッセージ・ハンドルを MQGET、MQPUT、または MQPUT1 呼び出しで使えます。ただし、メッセージ・ハンドルを使用できる IBM MQ 呼び出しは、一度に 1 つに限られます。2 番目の IBM MQ 呼び出しで同じメッセージ・ハンドルを使用しようとした際に、そのハンドルが使用中の場合は、2 番目の IBM MQ 呼び出しは失敗し、理由コード RC2499 が戻ります。

HCONN パラメーターが HCUNAS でない場合、返されるメッセージ・ハンドルは指定された接続でのみ使えます。

このメッセージ・ハンドルが使用される以下に示す MQI 呼び出しでは、同じ *HCONN* パラメーター値を使用する必要があります。

- MQDLTMH

- MQSETMP
- MQINQMP
- MQDLTMP
- MQMHBUFF
- MQBUFMH

戻されるメッセージ・ハンドルは、このメッセージ・ハンドルに MQDLTMH 呼び出しが発行されたとき、またはハンドルの有効範囲を定義する処理の単位が終了したときに無効になります。メッセージ・ハンドルの作成時に特定の接続が提供され、キュー・マネージャーに対するこの接続が無効になった場合 (例えば、MQDBC が呼び出された場合)、MQDLTMH が暗黙的に呼び出されます。

CMPCOD (10 桁の符号付き整数) - 出力

完了コード。以下のいずれかです。

CCOK

正常終了。

CCFAIL

呼び出し失敗。

REASON (10 桁の符号付き整数) - 出力

CMPCOD を限定する理由コード。

CMPCOD が CCOK の場合

RCNONE

(0, X'000') レポートする理由コードはありません。

CMPCOD が CCFAIL の場合

RC2204

(2204, X'089C') アダプターが利用できません。

RC2130

(2130, X'852') アダプター・サービス・モジュールをロードできません。

RC2157

(2157, X'86D') 1 次 ASID とホーム ASID が異なっています。

RC2219

(2219, X'08AB') 前の呼び出しが完了する前に MQI 呼び出しが入力された。

RC2461

(2461, X'099D') メッセージ・ハンドル作成オプションの構造が無効です。

RC2273

(2273, X'7D9') キュー・マネージャーへの接続が失われました。

RC2017

(2017, X'07E1') 使用可能なハンドルがなくなりました。

RC2018

(2018, X'7E2') 接続ハンドルが無効です。

RC2460

(2460, X'099C') メッセージ・ハンドル・ポインターが無効。

RC2046

(2046, X'07FE') オプションが無効であるか、矛盾しています。

RC2071

(2071, X'817') ストレージが不足しています。

RC2195

(2195, X'893') 予期しないエラーが発生しました。

詳しくは、[1426 ページの『IBM i の戻りコード \(ILE RPG\)』](#)を参照してください。

RPG 宣言

```
C*..1.....2.....3.....4.....5.....6.....7..  
C                                CALLP      MQCRTMH(HCONN : CRTOPT : HMSG :  
                                CMPCOD : REASON)
```

呼び出しのプロトタイプ定義は次のようになります。

```
DMQCRTMH          PR              EXTPROC('MQCRTMH')  
D* Connection handle  
D HCONN              10I 0 VALUE  
D* Options that control the action of MQCRTMH  
D CRTOPT              12A  
D* Message handle  
D HMSG              20I 0  
D* Completion code  
D CMPCOD              10I 0  
D* Reason code qualifying CompCode  
D REASON              10I 0
```



IBM i での MQCTL (コールバック制御)

MQCTL 呼び出しは、接続に対してオープンされたオブジェクト処理に対する制御アクションを実行します。

- [1295 ページの『構文』](#)
- [1295 ページの『使用上の注意』](#)
- [1295 ページの『パラメーター』](#)
- [1300 ページの『RPG 宣言』](#)

構文

MQCTL (*Hconn, Operation, ControlOpts, CompCode, Reason*)

使用上の注意

1. コールバック・ルーチンは、呼び出すすべてのサービスからの応答を検査しなければなりません。また解決できない条件をルーチンが検出した場合は、MQCB(CBREG) コマンドを発行してコールバック・ルーチンに対する呼び出しが繰り返されないようにしなければなりません。

パラメーター

MQCTL 呼び出しには、以下のパラメーターがあります。

HCONN (10 桁の符号付き整数) - 入力

このハンドルは、キュー・マネージャーに対する接続を表します。HCONN の値は、先行の MQCONN または MQCONNX 呼び出しによって戻されたものです。

OPERATN (10 桁の符号付き整数) - 入力

指定されたオブジェクト・ハンドルに定義されたコールバックで処理されている操作。以下のオプションのうち、いずれか 1 つだけを指定する必要があります。

CTLSR

指定された接続ハンドルについて定義されているすべてのメッセージ・コンシューマー関数のためのメッセージのコンシュームを開始します。

コールバックは、システムによって開始されるスレッド上で実行されます。それはアプリケーション・スレッドのいずれとも異なります。

この操作は、提供された接続ハンドルの制御をシステムに渡します。コンシューマー・スレッド以外のスレッドから発行できる MQI 呼び出しは、以下のものだけです。

- MQCTL、CTLSP 操作

- MQCTL、CTLSU 操作
- MQDISC - HConn を切断する前に MQCTL、CTLSP 操作を実行します。

RC2500 は、接続ハンドルの開始中に IBM MQ API 呼び出しが発行され、この呼び出しがメッセージ・コンシューマー関数から発信されていない場合に返されます。

接続が失敗すると、可能な限り早い段階で会話が停止されます。したがって、メイン・スレッドで発行された IBM MQ API 呼び出しは、一定の期間、戻りコードとして RC2500 を受け取った後に、接続が停止状態に変化すると戻りコード RC2009 を受け取る可能性があります。

この呼び出しは、コンシューマー関数で発行できます。コールバック・ルーチンと同じ接続の場合、その目的は、それ以前に発行された CTLSP 操作を取り消すということだけです。

アプリケーションがスレッドに対応していない IBM MQ ライブラリーにバインドされている場合には、このオプションはサポートされていません。

CTLSW

指定された接続ハンドルについて定義されているすべてのメッセージ・コンシューマー関数のためのメッセージのコンシュームを開始します。

メッセージ・コンシューマーは同じスレッド上で実行されます。以下のことが発生する時点まで、制御は MQCTL の呼び出し側に戻されません。

- MQCTL CTLSP または CTLSU の操作を使用することにより解放される場合、または
- すべてのコンシューマー・ルーチンが登録解除されたか中断された時点。

すべてのコンシューマーが登録解除または中断状態になった場合、暗黙のうちに CTLSP 操作が実行されます。

現行接続ハンドルまたはその他の接続ハンドルのいずれかについて、コールバック・ルーチン内からこのオプションを使用することはできません。そのような呼び出しを行おうとすると、RC2012 が戻されます。

CTLSW 操作中のいずれかの時点で、登録も中断もしていないコンシューマーがある場合は、呼び出しは理由コード RC2446 で失敗します。

CTLSW 操作中に接続が中断されると、MQCTL 呼び出しは警告の理由コード RC2521 を返しますが、接続は「開始済み」のままです。

アプリケーション側で CTLSP または CTLRE を発行することは可能です。この場合、CTLRE 操作は待機状態になります。

このオプションは、単一スレッド・クライアントではサポートされていません。

CTLSP

メッセージのコンシュームを停止し、すべてのコンシューマーがそれぞれの操作を完了するのを待機します。その後、このオプションが完了します。この操作は、接続ハンドルを解放します。

コールバック・ルーチン内から発行した場合、そのルーチンが終了する時点までこのオプションは有効になりません。既に読んだメッセージのコンシューマー・ルーチンが完了し、コールバック・ルーチンの停止呼び出しが要求されて実行された後は、それ以上メッセージ・コンシューマー・ルーチンは呼び出されません。

コールバック・ルーチン外から発行された場合、既に読んだメッセージのコンシューマー・ルーチンが完了し、コールバックの停止呼び出しが要求されて実行された後、制御は呼び出し元に戻されません。しかし、コールバック自体は登録済みのままです。

この関数は、先読みメッセージに対しては何の効果もありません。コールバック関数内から、コンシューマーが MQCLOSE(COQSC) を実行することによって、送達するためのメッセージがさらにあるかどうかを判別する必要があります。

CTLSU

メッセージのコンシュームを休止します。この操作は、接続ハンドルを解放します。

この機能は、アプリケーションに関するメッセージの先読みに対しては影響しません。長期間メッセージのコンシュームを停止する場合は、キューをクローズし、コンシュームを続行する必要がある時点で再オープンすることを考慮してください。

コールバック・ルーチン内から発行した場合、そのルーチンが終了する時点までは有効になりません。現在のルーチンが終了すると、その後、メッセージ・コンシューマー・ルーチンは呼び出されなくなります。

コールバック外から発行された場合、現在のコンシューマー・ルーチンが完了して、それ以降にルーチンが呼び出されなくなるまで、制御は呼び出し元に戻されません。

CTLRE

メッセージのコンシュームを再開します。

通常このオプションはメイン・アプリケーション・スレッドから発行されますが、コールバック・ルーチン内から、同じルーチン内でそれより前に発行された中断要求を取り消す目的で使用することも可能です。

CTLRE を使用して CTLSW を再開すると、操作はブロックします。

PCTLOP (MQCTLO) - 入力

MQCTL のアクションを制御するオプション

この構造の詳細については、[MQCTLO](#) を参照してください。

CMPCOD (10 桁の符号付き整数) - 出力

完了コード。以下のいずれかです。

CCOK

正常終了。

CCWARN

警告 (部分完了)。

CCFAIL

呼び出し失敗。

REASON (10 桁の符号付き整数) - 出力

次の理由コードは、キュー・マネージャーが **Reason** パラメーターに対して戻す理由コードです。

CMPCOD が CCOK の場合

RCNONE

(0, X'000') レポートする理由コードはありません。

CMPCOD が CCFAIL の場合

RC2133

(2133, X'855') データ変換サービス・モジュールをロードできない。

RC2204

(2204, X'89C') アダプターが利用できません。

RC2130

(2130, X'852') アダプター・サービス・モジュールをロードできません。

RC2374

(2374, X'946') API 出口で障害が発生しました。

RC2183

(2183, X'887') API 出口をロードできません。

RC2157

(2157, X'86D') 1 次 ASID とホーム ASID が異なります。

RC2005

(2005, X'7D5') バッファー長パラメーターは無効です。

RC2487

(2487, X'9B7') コールバック・ルーチンを呼び出せない

RC2448

(2448, X'990') コールバックが登録されていないので、登録解除、中断、または再開できない

RC2486

(2486, X'9B6') CBREG 呼び出しで CallbackFunction と CallbackName の両方が指定されました。
または、CallbackFunction か CallbackName のいずれか一方が指定されましたが、現在登録されているコールバック関数と一致していません。

RC2483

(2483, X'9B3') CallBackType フィールドが正しくない。

RC2219

(2219, X'8AB') 前の呼び出しが完了する前に MQI 呼び出しが入力されました。

RC2444

(2444, X'98C') オプション・ブロックが正しくない。

RC2484

(2484, X'9B4') MQCBD オプション・フィールドが正しくない。

RC2140

(2140, X'85C') 待機要求が CICS により拒否された。

RC2009

(2009, X'7D9') キュー・マネージャーとの接続が失われました。

RC2217

(2217, X'8A9') 接続が許可されていません。

RC2202

(2202, X'89A') 接続が静止しています。

RC2203

(2203, X'89B') 接続がシャットダウン中です。

RC2207

(2207, X'89F') 相関 ID のエラー。

RC2016

(2016, X'7E0') キューからの読み取りが禁止されている。

RC2351

(2351, X'92F') グローバル作業単位に矛盾がある。

RC2186

(2186, X'88A') 読み取りメッセージ・オプションの構造体が無効である。

RC2353

(2353, X'931') グローバル作業単位のためのハンドルが使用中。

RC2018

(2018, X'7E2') 接続ハンドルが無効です。

RC2019

(2019, X'7E3') オブジェクト・ハンドルが無効です。

RC2259

(2259, X'8D3') ブラウズの指定が不整合である。

RC2245

(2245, X'8C5') 作業単位の指定が不整合である。

RC2246

(2246, X'8C6') カーソル下のメッセージが取り出し対象として無効である。

RC2352

(2352, X'930') グローバル作業単位とローカル作業単位に矛盾がある。

RC2247

(2247, X'8C7') 突き合わせオプションが無効である。

RC2485

(2485, X'9B5') MaxMsgLength フィールドが正しくない

RC2026

(2026, X'7EA') メッセージ記述子が無効である。

RC2497

(2497, X'9C1') 指定された関数入り口点が見つからなかった。

RC2496

(2496, X'9C0') モジュールが見つかったが、タイプが間違っている (32 ビットまたは 64 ビット) か、有効な DLL ではない。

RC2495

(2495, X'9BF') モジュールが検索パス中にないか、またはロードが許可されていない。

RC2206

(2206, X'89E') メッセージ ID のエラー。

RC2250

(2250, X'8CA') メッセージ順序番号が無効である。

RC2331

(2331, X'91B') メッセージ・トークンについて無効な使い方をしている。

RC2036

(2036, X'7F4') ブラウズのためにキューがオープンされていない。

RC2037

(2037, X'7F5') 入力のためにキューがオープンされていない。

RC2041

(2041, X'7F9') オープンされた後でオブジェクト定義が変更された。

RC2101

(2101, X'835') オブジェクトが損傷しました。

RC2488

(2488, X'9B8') API 呼び出し上の命令コードが正しくない。

RC2046

(2046, X'7FE') オプションが無効であるか、矛盾しています。

RC2193

(2193, X'891') ページ・セット・データ・セットへのアクセス中にエラーが発生しました。

RC2052

(2052, X'804') キューが削除されました。

RC2394

(2394, X'95A') キューの索引タイプが間違っている。

RC2058

(2058, X'80A') キュー・マネージャー名が無効であるか、認識されていません。

RC2059

(2059, X'80B') キュー・マネージャーを接続に使用できません。

RC2161

(2161, X'871') キュー・マネージャーが静止しています。

RC2162

(2162, X'872') キュー・マネージャーのシャットダウン中です。

RC2102

(2102, X'836') 使用できるシステム・リソースが不足しています。

RC2069

(2069, X'815') このハンドルに未解決のシグナルがある。

RC2071

(2071, X'817') ストレージが不足しています。

RC2109

(2109, X'83D') 出口プログラムにより呼び出しが抑止されました。

RC2072

(2072, X'818') 同期点サポートが利用できない。

RC2195

(2195, X'893') 予期しないエラーが発生しました。

RC2354

(2354, X'932') グローバル作業単位の参加に失敗した。

RC2355

(2355, X'933') 作業単位呼び出しの混合はサポートされていない。

RC2255

(2255, X'8CF') 作業単位がキュー・マネージャーから使用不可。

RC2090

(2090, X'82A') MQGMO での待機間隔が無効である。

RC2256

(2256, X'8D0') 提供された MQGMO のバージョンが違っている。

RC2257

(2257, X'8D1') 提供された MQMD のバージョンが違っている。

RC2298

(2298, X'8FA') 要求された関数は、現在の環境では使用できない。

RPG 宣言

```
C*.1.....2.....3.....4.....5.....6.....7..
C                                CALLP      MQCTL(HCONN : OPERATN : PCTLOP :
                                           CMPCOD : REASON)
```

呼び出しのプロトタイプ定義は次のようになります。

```
DMQCTL          PR          EXTPROC('MQCTL')
D* Connection handle
D HCONN          10I 0 VALUE
D* Operation
D OPERATN        10I 0 VALUE
D* Control options
D PCTLOP          32A
D* Completion code
D CMPCOD          10I 0
D* Reason code qualifying CompCode
D REASON          10I 0
```

IBM i IBM i での MQDISC (キュー・マネージャーの切断)

MQDISC 呼び出しはキュー・マネージャーとアプリケーション・プログラムとの接続を切断します。MQCONN および MQCONNX 呼び出しの逆の操作にあたります。

- [1300 ページの『構文』](#)
- [1301 ページの『使用上の注意』](#)
- [1301 ページの『パラメーター』](#)
- [1302 ページの『RPG 宣言』](#)

構文

MQDISC (HCONN, CMPCOD, REASON)

使用上の注意

1. アプリケーションがまだオブジェクトをオープンしているときに MQDISC 呼び出しが発行されると、これらのオブジェクトはキュー・マネージャーによってクローズされます。このときのクローズ・オプションは CONONE です。
2. 作業単位内にあるコミットされていない変更内容でアプリケーションが終了する場合、それらの変更内容の後処理は、そのアプリケーションの終了の仕方によって異なります。
 - a. アプリケーションが終了前に MQDISC 呼び出しを発行する場合、
 - ・ キュー・マネージャーが調整する作業単位の場合、キュー・マネージャーがアプリケーションの代わりに MQCMIT 呼び出しを出します。可能であれば作業単位がコミットされ、そうでなければバックアウトされます。
 - ・ 外部的に調整された作業単位の場合、作業単位の状態には変更がありません。しかし、キュー・マネージャーは、作業単位コーディネーターに求められると、作業単位がコミットされなければならないことを示します。
 - b. アプリケーションが正常終了しても MQDISC 呼び出しを発行しない場合は、作業単位はバックアウトされます。
 - c. アプリケーションが MQDISC 呼び出しを出さずに異常終了する場合、作業単位はバックアウトされます。

パラメーター

MQDISC 呼び出しには、以下のパラメーターがあります。

HCONN (10-digit signed integer) - 入出力

接続ハンドル。

このハンドルは、キュー・マネージャーに対する接続を表します。HCONN の値は、先行の MQCONN または MQCONNX 呼び出しによって戻されたものです。

呼び出しが正常に完了すると、キュー・マネージャーは、HCONN をその環境で無効なハンドルを表す値に設定します。値は、以下のとおりです。

HCUNUH

使用できない接続ハンドル。

CMPCOD (10 桁の符号付き整数) - 出力

完了コード。

これは、以下のいずれかになります。

CCOK

正常終了。

CCWARN

警告 (部分完了)。

CCFAIL

呼び出し失敗。

REASON (10 桁の符号付き整数) - 出力

CMPCOD を限定する理由コード。

CMPCOD が CCOK の場合

RCNONE

(0, X'000') レポートする理由コードはありません。

CMPCOD が CCFAIL の場合

RC2219

(2219, X'8AB') 前の呼び出しが完了する前に MQI 呼び出しが再入力されました。

RC2009

(2009, X'7D9') キュー・マネージャーとの接続が失われました。

RC2018

(2018, X'7E2') 接続ハンドルが無効です。

RC2058

(2058, X'80A') キュー・マネージャー名が無効であるか、認識されていません。

RC2059

(2059, X'80B') キュー・マネージャーを接続に使用できません。

RC2162

(2162, X'872') キュー・マネージャーのシャットダウン中です。

RC2102

(2102, X'836') 使用できるシステム・リソースが不足しています。

RC2071

(2071, X'817') ストレージが不足しています。

RC2195

(2195, X'893') 予期しないエラーが発生しました。

RPG 宣言

```
C*..1.....2.....3.....4.....5.....6.....7..
C                                CALLP      MQDISC(HCONN : CMPCOD : REASON)
```

呼び出しのプロトタイプ定義は次のようになります。

```
D*..1.....2.....3.....4.....5.....6.....7..
DMQDISC      PR      EXTPROC('MQDISC')
D* Connection handle
D HCONN                      10I 0
D* Completion code
D CMPCOD                    10I 0
D* Reason code qualifying CMPCOD
D REASON                      10I 0
```

IBM i IBM i での MQDLTMH (メッセージ・ハンドルの削除)

MQDLTMH 呼び出しは、メッセージ・ハンドルを削除するので、MQCRTMH 呼び出しの逆です。

- [1302 ページの『構文』](#)
- [1302 ページの『使用上の注意』](#)
- [1304 ページの『パラメーター』](#)
- [1305 ページの『RPG 宣言』](#)

構文

MQDLTMH ((Hconn, Hmsg, DltMsgHOpts, CompCode, Reason)

使用上の注意

1. この呼び出しは、キュー・マネージャーそのものが作業単位を調整するときのみ使用できます。次のタイプがあります。
 - ローカル作業単位 (変更内容は IBM MQ リソースにのみ影響を及ぼす)。
 - グローバル作業単位 (変更内容は、IBM MQ リソースだけでなく、他のリソース・マネージャーに属するリソースにも影響を及ぼす場合がある)。

ローカル作業単位およびグローバル作業単位の詳細については、[1264 ページの『IBM i での MQBEGIN \(作業単位の開始\)』](#)を参照してください。

2. キュー・マネージャーが作業単位を調整しない環境では、MQBACK ではなく適切なバックアウト呼び出しを使用してください。この環境ではまた、アプリケーションの異常終了を原因とする暗黙的バックアウトをサポートすることもできます。

• z/OS では、以下の呼び出しを使用してください。

- バッチ・プログラム (IMS バッチ DL/I プログラムを含む) は、作業単位が IBM MQ リソースにのみ影響する場合に、MQBACK 呼び出しを使用できます。一方、作業単位が IBM MQ リソースと、その他のリソース・マネージャー (Db2 など) に属するリソースの両方に影響を及ぼす場合は、z/OS Recoverable Resource Service (RRS) が提供する SRRBACK 呼び出しを使用できます。SRRBACK 呼び出しを実行すると、RRS 調整対応のリソース・マネージャーに属するリソースに対する変更がバックアウトされます。
- CICS アプリケーションは、EXEC CICS SYNCPOINT ROLLBACK コマンドを使用して作業単位をバックアウトする必要があります。CICS アプリケーションには MQBACK 呼び出しを使用しないでください。
- IMS アプリケーション (バッチ DL/I プログラム以外) は、ROLB などの IMS 呼び出しを使用して、作業単位をバックアウトする必要があります。IMS アプリケーション (バッチ DL/I プログラム以外) には、MQBACK 呼び出しを使用しないでください。
- IBM i では、この呼び出しはキュー・マネージャーで調整されるローカル作業単位で使用してください。これは、コミットメント定義がジョブ・レベルで存在してはならないことを意味します。つまり、**CMTSCOPE(*JOB)** パラメーターを指定した STRCMTCTL コマンドがジョブに対して発行されているわけではありません。

3. 作業単位内にあるコミットされていない変更内容でアプリケーションが終了する場合、それらの変更内容の後処理は、そのアプリケーションが正常に終了するか、異常終了するかで異なります。詳細については、[1300 ページの『IBM i での MQDISC \(キュー・マネージャーの切断\)』](#)の使用上の注意を参照してください。

4. アプリケーションでグループ内のメッセージまたは論理メッセージのセグメントの書き込みまたは読み取りを行う場合、キュー・マネージャーは、最後に MQPUT および MQGET 呼び出しが正常に実行されたメッセージ・グループに関する情報を保存します。この情報は、キュー・ハンドルに関する次のような情報です。

- MQMD 中の *GroupId*、*MsgSeqNumber*、*Offset*、および *MsgFlags* フィールドの値。
- そのメッセージが作業単位の一部であるかどうか。
- MQPUT 呼び出しについて、そのメッセージが持続メッセージか、非持続メッセージか。

キュー・マネージャーは、次のものについて 1 つずつ、3 セットのグループおよびセグメント情報を保持しています。

- 最後に正常に実行された MQPUT 呼び出し (これは作業単位の一部である場合があります)。
- 最後に正常に実行された MQGET 呼び出しのうちキューからメッセージを削除したもの (作業単位の一部である場合があります)。
- 最後に正常に実行された MQGET 呼び出しのうちキュー上のメッセージをブラウズしたもの (これが作業単位の一部であることはありません)。

アプリケーションが作業単位の一部としてメッセージの書き込みまたは読み取りを行った後でその作業単位をバックアウトした場合、グループとセグメントの情報は、次に示すように以前の値に復元されます。

- MQPUT 呼び出しについての情報は、現行作業単位内のそのキュー・ハンドルについて最初に正常に実行された MQPUT 呼び出し以前の値に復元されます。
- MQGET 呼び出しについての情報は、現行作業単位内のそのキュー・ハンドルについて最初に正常に実行された MQGET 呼び出し以前の値に復元されます。

作業単位の開始後にアプリケーションによって更新されたキューであっても、それが作業単位の有効範囲外である場合は、その作業単位がバックアウトされても、グループおよびセグメント情報は復元されません。

作業単位のバックアウト時にグループおよびセグメント情報を以前の値に復元する機能により、アプリケーションは、数多くのセグメントで構成される大きなメッセージ・グループまたは大きな論理メッセージをいくつかの作業単位にまたがって広げることができます。そして、いずれかの作業単位が失敗しても、そのメッセージ・グループまたは論理メッセージ内の正しい点でアプリケーションを再始動できます。ローカル・キュー・マネージャーのキュー・ストレージが限られている場合には、いくつかの作業単位を使用する方が有効となる場合があります。ただし、システム障害の発生時に各メッセージの書き込みまたは読み取りを正しい時点で再始動できるようにするには、アプリケーションが十分な情報を維持している必要があります。

システム障害後に正しい時点から再始動する方法の詳細については、[PMOPT \(10 桁の符号付き整数\)](#) の [PMLOGO](#) オプション、および [GMOPT \(10 桁の符号付き整数\)](#) の [GMLOGO](#) オプションを参照してください。

次の『使用上の注意』は、キュー・マネージャーで作業単位を調整する場合にのみ適用されます。

5. 作業単位の 1 つには、1 つの接続ハンドルと同じ有効範囲があります。特定の作業単位に影響を与えるすべての IBM MQ 呼び出しは、同じ接続ハンドルを使用して実行しなければなりません。別の接続ハンドルを用いて呼び出しを発行すると (例えば、別のアプリケーションで呼び出しを発行する)、別の作業単位に影響が及びます。接続ハンドルの有効範囲については、[HCONN \(10 桁の符号付き整数\)](#) - 出力を参照してください。
6. この呼び出しで影響を受けるメッセージは、現行の作業単位の一部として書き込まれたメッセージ、または取り出されたメッセージに限られます。
7. 長時間実行しているアプリケーションが、1 つの作業単位に対して MQGET、MQPUT、または MQPUT1 呼び出しを発行する一方、コミット呼び出しまたはバックアウト呼び出しを一度も発行しない場合には、キューが他のアプリケーションでは使用できないメッセージで満杯になることがあります。この可能性を回避するために、管理者は、**MaxUncommittedMsgs** キュー・マネージャー属性を、ランナウェイ・アプリケーションがキューを満杯にしないように十分に低い値に設定する必要がありますが、予期されるメッセージング・アプリケーションが正しく機能するように十分に高い値に設定する必要があります。

パラメーター

MQDLTMH 呼び出しには、以下のパラメーターがあります。

HCONN (10 桁の符号付き整数) - 入力

このハンドルは、キュー・マネージャーに対する接続を表します。

値は、**HMSG** パラメーターで指定されているメッセージ・ハンドルを作成するために使用された接続ハンドルと一致していなければなりません。

メッセージ・ハンドルが、HCUNAS を使用して作成されている場合、メッセージ・ハンドルを削除するスレッドで、有効な接続が確立されている必要があります。接続がないと、呼び出しは RC2009 で失敗します。

HMSG (20 桁の符号付き整数) - 入出力

これは削除されるメッセージ・ハンドルです。値は、前の MQCRTMH 呼び出しで戻されたものです。

呼び出しが正常に完了すると、ハンドルは環境に対して無効な値に設定されます。値は、以下のとおりです。

HMUNUH

使用できないメッセージ・ハンドル。

同じメッセージ・ハンドルを渡した別の IBM MQ 呼び出しが進行中の場合は、メッセージ・ハンドルを削除できません。

DLTOPT (MQDMHO) - 入力

詳細については、[MQDMHO](#) を参照してください。

CMPCOD (10 桁の符号付き整数) - 出力

完了コード。以下のいずれかです。

CCOK

正常終了。

CCFAIL

呼び出し失敗。

REASON (10 桁の符号付き整数) - 出力

CMPCOD を限定する理由コード。

CMPCOD が CCOK の場合

RCNONE

(0, X'000') レポートする理由コードはありません。

CMPCOD が CCFAIL の場合

RC2204

(2204, X'089C') アダプターが利用できません。

RC2130

(2130, X'852') アダプター・サービス・モジュールをロードできません。

RC2157

(2157, X'86D') 1 次 ASID とホーム ASID が異なっています。

RC2219

(2219, X'08AB') 前の呼び出しが完了する前に MQI 呼び出しが入力された。

RC2009

(2009, X'07D9') キュー・マネージャーとの接続が失われました。

RC2462

(2462, X'099E') メッセージ・ハンドル削除オプションの構造が無効である。

RC2460

(2460, X'099C') メッセージ・ハンドル・ポインターが無効。

RC2499

(2499, X'09C3') メッセージ・ハンドルがすでに使用中。

RC2046

(2046, X'07FE') オプションが無効であるか、矛盾しています。

RC2071

(2071, X'817') ストレージが不足しています。

RC2195

(2195, X'893') 予期しないエラーが発生しました。

詳しくは、[1426 ページの『IBM i の戻りコード \(ILE RPG\)』](#)を参照してください。

RPG 宣言

```
C*..1.....2.....3.....4.....5.....6.....7..
C                                CALLP      MQDLTMH(HCONN : HMSG : DLTOPT :
                                CMPCOD : REASON)
```

呼び出しのプロトタイプ定義は次のようになります。

```
DMQDLTMH          PR          EXTPROC('MQDLTMH')
D* Connection handle
D HCONN              10I 0 VALUE
D* Message handle
D HMSG              20I 0
D* Options that control the action of MQDLTMH
D DLTOPT            12A
D* Completion code
D CMPCOD            10I 0
D* Reason code qualifying CompCode
D REASON            10I 0
```

MQDLTMP - メッセージ・プロパティの削除

MQDLTMP 呼び出しは、メッセージ・ハンドルからプロパティを削除するので、MQSETMP 呼び出しの逆です。

- [1306 ページの『構文』](#)
- [1306 ページの『パラメーター』](#)
- [1307 ページの『RPG 宣言』](#)

構文

MQDLTMP (*Hconn, Hmsg, DltPropOpts, Name, CompCode, Reason*)

パラメーター

MQDLTMP 呼び出しには、以下のパラメーターがあります。

HCONN (10 桁の符号付き整数) - 入力

このハンドルは、キュー・マネージャーに対する接続を表します。値は、**HMSG** パラメーターで指定されているメッセージ・ハンドルを作成するために使用された接続ハンドルと一致していなければなりません。

メッセージ・ハンドルが、HCUNAS を使用して作成されている場合、メッセージ・ハンドルを削除するスレッドで、有効な接続が確立されている必要があります。接続がないと、呼び出しは RC2009 で失敗します。

HMSG (20 桁の符号付き整数) - 入力

これは、削除されるプロパティを含むメッセージ・ハンドルです。値は、前の MQCRTMH 呼び出しで戻されたものです。

DLTOPT (MQDMPO) - 入力

詳細については、[MQDMPO](#) データ・タイプを参照してください。

PRNAME (MQCHARV) - 入力

削除するプロパティの名前。プロパティ名の詳細については、[プロパティ名](#)を参照してください。

プロパティ名にワイルドカードを使用することはできません。

CMPCOD (10 桁の符号付き整数) - 出力

完了コード。以下のいずれかです。

CCOK

正常終了。

CCWARN

警告 (部分完了)。

CCFAIL

呼び出し失敗。

REASON (10 桁の符号付き整数) - 出力

CMPCOD を限定する理由コード。

CMPCOD が CCOK の場合

RCNONE

(0, X'000') レポートする理由コードはありません。

CMPCOD が CCWARN の場合

RC2471

(2471, X'09A7') プロパティが使用できない。

RC2421

(2421, X'0975') プロパティを含む MQRFH2 フォルダーを構文解析できなかった。

CMPCOD が CCFAIL の場合

RC2204

(2204, X'089C') アダプターが利用できません。

RC2130

(2130, X'0852') アダプター・サービス・モジュールをロードできない。

RC2157

(2157, X'086D') 1 次 ASID とホーム ASID とが異なっている。

RC2219

(2219, X'08AB') 前の呼び出しが完了する前に MQI 呼び出しが入力された。

RC2009

(2009, X'07D9') キュー・マネージャーとの接続が失われました。

RC2481

(2481, X'09B1') メッセージ・プロパティ削除のオプション構造体が無効です。

RC2460

(2460, X'099C') メッセージ・ハンドルが無効。

RC2499

(2499, X'09C3') メッセージ・ハンドルがすでに使用中。

RC2046

(2046, X'07FE') オプションが無効であるか、矛盾しています。

RC2442

(2442, X'098A') プロパティ名が無効である。

RC2111

(2111, X'083F') プロパティ名エンコード文字セット ID が無効である。

RC2195

(2195, X'0893') 予期しないエラーが発生した。

これらのコードについて詳しくは、[API 完了コードと理由コード](#)を参照してください。

RPG 宣言

```
C*..1.....2.....3.....4.....5.....6.....7..
C                                CALLP      MQDLTMP(HCONN : HMSG : DLTOPT :
                                PRNAME : CMPCOD : REASON)
```

呼び出しのプロトタイプ定義は次のようになります。

```
DMQDLTMP          PR          EXTPROC('MQDLTMP')
D* Connection handle
D HCONN            10I 0 VALUE
D* Message handle
D HMSG             20I 0 VALUE
D* Options that control the action of MQDLTMP
D DLTOPT           12A
D* Property name
D PRNAME           32A
D* Completion code
D CMPCOD           10I 0
D* Reason code qualifying CompCode
D REASON           10I 0
```


MQGET 呼び出しは、MQOPEN 呼び出しを使用してオープンされたローカル・キューからメッセージを取り出します。

- [1308 ページの『構文』](#)
- [1308 ページの『使用上の注意』](#)
- [1311 ページの『パラメーター』](#)
- [1315 ページの『RPG 宣言』](#)

構文

MQGET (*HCONN, HOBJ, MSGDSC, GMO, BUFLN, BUFFER, DATLEN, CMPCOD, REASON*)

使用上の注意

1. 取り出されたメッセージは、通常、キューから削除されます。削除は、MQGET 呼び出し自体の一部、または同期点の一部として行われる可能性があります。 **GMO** パラメーターで **GMBRWF** または **GMBRWN** オプションが指定されている場合は、メッセージの削除は行われません ([1080 ページの『IBM i での MQGMO \(読み取りメッセージ・オプション\)』](#)に記載されている **GMOPT** フィールドを参照してください)。
2. **GMLK** オプションが、ブラウズ・オプションの 1 つを使用して指定される場合、ブラウズしたメッセージはロックされ、このハンドルにのみ表示されます。

GMUNLK オプションが指定されていると、以前にロックされたメッセージがロック解除されます。この場合、メッセージは検索されません。 **MSGDSC**、**BUFLN**、**BUFFER**、および **DATLEN** パラメーターを調べたり、変更したりしません。

3. MQGET 呼び出しを発行するアプリケーションを IBM MQ MQI client として実行する場合、MQGET 呼び出しの処理中に IBM MQ MQI client が異常終了したり、クライアント接続が切断されたりすると、取得するメッセージが失われる可能性があります。これは、キュー・マネージャーのプラットフォーム上で実行されている、クライアントに代わって MQGET 呼び出しを発行するサロゲートが、メッセージをクライアントに戻す時点まで、クライアントが切断されていることを検出できないために生じます。検出されるのは、メッセージがキューから除去された後です。このことは、持続メッセージの場合でも、非持続メッセージの場合でも起こります。

常に作業単位内でメッセージを検索することにより、このような仕方ではメッセージを失うリスクをなくすることができます (つまり、MQGET 呼び出しで **GMSYP** オプションを指定し、**MQCMIT** または **MQBACK** 呼び出しを使用して、メッセージの処理が完了するときに作業単位をコミットまたはバックアウトします)。 **GMSYP** が指定されている場合、クライアントが異常終了する、または接続が切断されると、サロゲートはキュー・マネージャー上の作業単位をバックアウトして、メッセージはキューに再び入れられます。

基本的に、キュー・マネージャーのプラットフォームで実行しているアプリケーションでも同じ状況が生じる可能性があります。ただし、この場合、メッセージが失われる期間は短いものです。しかし、IBM MQ MQI clients と同じように、作業単位内でメッセージを取得することによってリスクをなくすることができます。

4. もしもアプリケーションがメッセージのシーケンスを特定の 1 つの作業単位内のキューを処理し、その作業単位を正常にコミットすると、メッセージは次のように検索できるようになります。
 - キューが非共有キュー (つまりローカル・キュー) である場合は、作業単位のメッセージはすべて同時に利用できます。
 - キューが共有キューである場合、作業単位のメッセージは、書き込まれた順番で利用できます。すべてを同時に利用することはできません。システムの負荷が重い場合は、作業単位内の最初のメッセージを正常に取得できても、作業単位内の 2 番目以降のメッセージに対する MQGET 呼び出しが RC2033 で失敗する可能性があります。このことが生じた場合、アプリケーションは少し待ってから、操作を再試行しなければなりません。

5. アプリケーションがメッセージ・グループを使用せずにメッセージ・シーケンスを同じキューに書き込んだ場合、特定の条件が満たされていれば、それらのメッセージの順序は保持されます。詳細については、MQPUT 呼び出しの説明の『使用上の注意』を参照してください。条件が満たされていて、さらに以下の条件も満たされていれば、メッセージは送信された順序で受信側のアプリケーションに提示されます。

- キューからメッセージを読み取る受信側は 1 つだけである。

キューからメッセージを読み取るアプリケーションが 2 つ以上ある場合は、シーケンスに属するメッセージを識別するために使用するメカニズムについて、アプリケーションが送信側と合意している必要があります。例えば、送信側は、一連のメッセージの中のすべての MDCID フィールドを、その一連のメッセージのみに固有の値に設定することができます。

- 受信側は、例えば特定の MDMID または MDCID を指定することによって、検索の順序を意図的に変更することはありません。

送信側のアプリケーションがメッセージ・グループとしてメッセージを書き込む場合、受信側のアプリケーションが MQGET 呼び出しで GMLOGO オプションを指定していれば、メッセージは正しい順序で受信側のアプリケーションに提示されます。メッセージ・グループの詳細については、以下を参照してください。

- MQMD の MDMFL フィールド
- MQPMO の PMLOGO オプション
- MQGMO の GMLOGO オプション

6. アプリケーションは、**MSGDSC** パラメーターの MDFB フィールドにフィードバック・コード FBQUIT があるかどうかをテストします。この値が検出されると、アプリケーションが終了します。詳細については、[1114 ページの『IBM i での MQMD \(メッセージ記述子\)』の MDFB フィールド](#)を参照してください。

7. HOBJS によって識別されたキューが OOSAVA オプションでオープンされ、MQGET 呼び出しからの完了コードが CCOK または CCWARN である場合は、キュー・ハンドル HOBJS と関連付けられたコンテキストが、取り出されたメッセージのコンテキストに設定されます (ただし、GMBRWF または GMBRWN オプションが設定されていて、コンテキストに使用不可のマークが付いている場合を除く)。このコンテキストは、PMPASI または PMPASA オプションを指定することにより、後続の MQPUT または MQPUT1 呼び出しで使用することができます。これにより、受信したメッセージのコンテキストの全体または一部を別のメッセージに転送することが可能になります (例えば、メッセージを別のキューに転送する場合など)。メッセージ・コンテキストについての詳細は、[メッセージ・コンテキストおよびコンテキスト情報の制御](#)を参照してください。

8. GMCONV オプションが **GMO** パラメーターに指定されている場合は、アプリケーション・メッセージ・データは、受信側アプリケーションで要求される表現に変換されてから **BUFFER** パラメーターに入ります。

- メッセージ中の制御情報の MDFMT フィールドが、アプリケーション・データの構造を識別し、メッセージ中の制御情報の MDCSI および MDENC フィールドが、その文字セット ID とエンコードを指定します。
- MQGET 呼び出しを発行するアプリケーションは、**MSGDSC** パラメーターの MDCSI および MDENC フィールドに、アプリケーション・メッセージ・データを変換する必要がある文字セット ID とエンコードを指定します。

メッセージ・データの変換が必要な時は、メッセージの中の制御情報の MDFMT フィールドの値に応じて、キュー・マネージャー自体またはユーザー作成出口で変換が実行されます。

- 以下に示す形式は、キュー・マネージャーによって自動的に変換されます。これらは、「組み込み」形式と呼ばれています。

FMADMN

FMMDE

FMCICS

FMPCF

FMCMDD1

FMRMH

FMCMD2	FMRFH
FMDLH	FMRFH2
FMDH	FMSTR
FMEVNT	FMTM
FMIMS	FMXQH
FMIMVS	

- **FMNONE** という形式名は、メッセージ内のデータの特性が未定義であることを示す特殊な値です。その結果、キュー・マネージャーは、メッセージがキューから取り出されるときには、変換を行いません。

注: **FMNONE** という形式名のメッセージに対する **MQGET** 呼び出しで **GMCONV** を指定したときに、メッセージの文字セットまたはエンコードが **MSGDSC** パラメーターに指定された文字セットまたはエンコードと異なる場合、それでもメッセージは **BUFFER** パラメーターに戻されますが (ただし、他のエラーはないものとする)、完了コード **CCWARN** と理由コード **RC2110** を戻して、呼び出しは完了します。

メッセージ・データの特性が変換の必要がないことを表す時、または送信側/受信側アプリケーションが、メッセージ・データを送信する形式に互いに合意した時、**FMNONE** を使用することができます。

- それ以外の形式名を使用すると、メッセージは、ユーザー作成出口に渡され、変換が行われます。その出口は、環境固有の追加とは別に、形式と同一の名前を持ちます。ユーザー指定の形式名に、「MQ」という文字で始まる名前は使用しないでください。これは、この文字で始まる名前は将来サポートされる形式名と競合する可能性があるためです。

メッセージ内のユーザー・データは、サポートされているすべての文字セットおよびエンコードとの間で変換することができます。ただし、メッセージに 1 つ以上の **IBM MQ** ヘッダー構造体が含まれている場合、キュー名内の有効な文字について 2 バイト文字またはマルチバイト文字を備えている文字セットとの間で、そのメッセージを変換することはできません。そのような変換を試みた場合、**RC2111** と **RC2115** のいずれかの理由コードが発生し、メッセージは変換されずに戻されます。**UNICODE** 文字セット **UTF-16** は、そのような文字セットの一例です。

MQGET から戻る際の、以下の理由コードは、メッセージが正常に変換されたことを示しています。

- **RCNONE**

以下の理由コードは、メッセージが正常に変換された可能性があることを示しています。アプリケーションは、**MSGDSC** パラメーターの **MDCSI** フィールドと **MDENC** フィールドを調べて確認する必要があります。

- **RC2079**

その他の理由コードはすべて、メッセージが変換されなかったことを表します。

注: この例で説明した理由コードの解釈が成立するのは、ユーザー作成出口により変換が実行され、その出口が処理のガイドラインに準拠している場合に限りです。

9. 上記の組み込み形式では、キュー・マネージャーに、**GMCONV** オプションが指定されていると、メッセージの文字ストリングのデフォルト変換を実行する場合があります。デフォルト変換の場合、キュー・マネージャーによるストリング・データの変換時に、実際の文字セットに近似するインストール指定デフォルト文字セットが使用されます。その結果、**MQGET** 呼び出しは終了し、完了コード **CCOK** が戻ります。**CCWARN** および理由コード **RC2111** または **RC2115** が戻ることはありません。

注: 近似する文字セットを使用してストリング・データを変換すると、文字が不正確に変換される場合があります。実際の文字セットおよびデフォルト文字セットの両方に共通する文字のみをストリング内に使用するようにすれば、これを回避することができます。

デフォルト変換は、アプリケーション・メッセージ・データにも、また **MQMD** および **MQMDE** 構造体内の各文字フィールドにも適用されます。

- アプリケーション・メッセージ・データのデフォルト変換は、次のすべての記述が該当する場合にだけ生じます。
 - アプリケーションが GMCONV を指定しています。
 - メッセージに、サポートされていない文字セットからの変換、またはその文字セットへの変換が必要なデータが含まれています。
 - キュー・マネージャーがインストールまたは再始動され、デフォルト変換が使用可能になりました。
- MQMD および MQMDE 構造体の文字フィールドのデフォルト変換は、キュー・マネージャーでデフォルト変換が使用可能になっている場合に、必要に応じて行われます。この変換は、アプリケーションで GMCONV オプションが MQGET 呼び出しに指定されていない場合でも実行されます。

10. RPG プログラミング例に示す **BUFFER** パラメーターは、ストリングとして宣言されています。このため、パラメーターの最大長は 256 バイトに制限されます。これより大きいバッファが必要な場合は、このパラメーターをストリングではなく構造体として宣言するか、あるいは物理ファイルのフィールドとして宣言する必要があります。

このパラメーターを構造体として宣言すると、最大長は 9999 バイトまで増加します。このパラメーターを物理ファイルのフィールドとして宣言すると、最大長は約 32 KB まで増加します。

パラメーター

MQGET 呼び出しには、以下のパラメーターがあります。

HCONN (10 桁の符号付き整数) - 入力

接続ハンドル。

このハンドルは、キュー・マネージャーに対する接続を表します。HCONN の値は、先行の MQCONN または MQCONNX 呼び出しによって戻されたものです。

HOBJ (10 桁の符号付き整数) - 入力

オブジェクト・ハンドル

このハンドルは、メッセージが取り出されるキューを表します。HOBJ の値は、前の MQOPEN 呼び出しで戻されたものです。このキューは、次のオプションを 1 つ以上指定してオープンしておく必要があります (詳しくは、[1332 ページの『IBM i での MQOPEN \(オブジェクトのオープン\)』](#)を参照してください)。

- OOINPS
- OOINPX
- OOINPQ
- OOBROW

MSGDSC (MQMD) - 入出力

メッセージ記述子。

この構造体は、必要なメッセージの属性と、取り出されるメッセージの属性を記述します。詳細は [1114 ページの『IBM i での MQMD \(メッセージ記述子\)』](#)を参照してください。

BUFLEN がメッセージ長より短い場合でも、GMATM が **GMO** パラメーターで指定されているかどうかにかかわらず、キュー・マネージャーによって MSGDSC が入力されます ([1080 ページの『IBM i での MQGMO \(読み取りメッセージ・オプション\)』](#)で説明されている GMOPT フィールドを参照してください)。

アプリケーションがバージョン 1 の MQMD を提供している場合、戻されるメッセージでは、アプリケーション・メッセージ・データに MQMDE の接頭部が付いていますが、これは MQMDE 内の 1 つ以上のフィールドがデフォルト以外の値を持つ場合のみです。MQMDE 内のすべてのフィールドがデフォルト値を持つ場合、この MQMDE は省略されます。MQMD 内の MDFMT フィールドにある FMMDE という形式名があれば、MQMDE が存在することを意味します。

GMO (MQGMO) - 入出力

MQGET のアクションを制御するオプション。

詳細は 1080 ページの『[IBM i での MQGMO \(読み取りメッセージ・オプション\)](#)』を参照してください。

BUFLEN (10 桁の符号付き整数) - 入力

BUFFER 域の長さ (バイト単位)。

メッセージにデータがない場合、またはメッセージがキューから削除され、データが廃棄される (この場合には GMATM の指定が必要) 場合には、ゼロを指定できます。

注: キューから読み取り可能な最長メッセージの長さは、キュー属性 **MaxMsgLength** によって示されます (1380 ページの『[キューの属性](#)』を参照)。

BUFFER (1 バイトのビット・ストリング x BUFLEN) - 出力

メッセージ・データが入れられる区域。

バッファは、メッセージのデータの性質に適した境界に位置合わせされなければなりません。IBM MQ ヘッダー構造になっているメッセージを含む、ほとんどのメッセージには 4 バイトの位置合わせが適していますが、メッセージによってはより厳しい位置合わせを必要とする場合があります。例えば、64 ビット・バイナリー整数を含むメッセージは 8 バイト境界に合わせる必要がある場合があります。

BUFLEN がメッセージ長より短い場合でも、可能な限り多くのメッセージが BUFFER へ移動されます。この処理は、**GMO** パラメーターに GMATM が指定されているかどうかにかかわらず行われます (詳細については、1080 ページの『[IBM i での MQGMO \(読み取りメッセージ・オプション\)](#)』の **GMOPT** フィールドを参照してください)。

BUFFER 内のデータの文字セットとエンコードは、**MSGDSC** パラメーターで返される **MDCSI** フィールドと **MDENC** フィールドによって指定されます。これらの値が受信側で必要とされている値と異なる場合、受信側はアプリケーション・メッセージ・データを必要な文字セットとエンコードに変換する必要があります。ユーザー作成の出口で **GMCONV** オプションを使用して、メッセージ・データの変換を行うことができます (このオプションの詳細については、1080 ページの『[IBM i での MQGMO \(読み取りメッセージ・オプション\)](#)』を参照してください)。

注: MQGET 呼び出しのその他のパラメーターはすべて、ローカル・キュー・マネージャーの文字セットとエンコードに従っています (**CodedCharSetId** キュー・マネージャー属性と **ENNAT** で指定します)。

呼び出しが失敗した場合は、バッファの内容が変更されてしまっていることもあります。

DATLEN (10 桁の符号付き整数) - 出力

メッセージの長さ。

これは、メッセージ内のアプリケーション・データの長さ (バイト数) です。このメッセージ長が **BUFLEN** よりも長い場合には、**BUFLEN** のバイト数分だけが **BUFFER** パラメーター中に戻されます (すなわち、メッセージは切り捨てられます)。この値がゼロの場合は、メッセージにアプリケーション・データがないことを意味します。

BUFLEN がメッセージ長より短い場合でも、**GMATM** が **GMO** パラメーターで指定されているかどうかに関係なく、**DATLEN** はキュー・マネージャーによって入力されます (詳しくは、1080 ページの『[IBM i での MQGMO \(読み取りメッセージ・オプション\)](#)』で説明されている **GMOPT** フィールドを参照してください)。これにより、アプリケーションは、メッセージ・データを収容するのに必要なバッファのサイズを判別して、適切なサイズのバッファを用いて呼び出しを再発行することができます。

ただし、**GMCONV** オプションが指定され、変換されたメッセージ・データが **BUFFER** に入れるには長すぎる場合には、**DATLEN** に戻される値は以下のものになります。

- 未変換データの長さ (キュー・マネージャー定義の形式の場合)。

この場合、データの特性により、変換中にデータが拡張する場合、アプリケーションは、**DATLEN** にキュー・マネージャーが戻した値より大きいバッファを割り当てなければなりません。

- データ変換出口により戻される値 (アプリケーション定義の形式の場合)

CMPCOD (10 桁の符号付き整数) - 出力

完了コード。

これは、以下のいずれかになります。

CCOK

正常終了。

CCWARN

警告 (部分完了)。

CCFAIL

呼び出し失敗。

REASON (10 桁の符号付き整数) - 出力

CMPCOD を限定する理由コード。

次の理由コードは、キュー・マネージャーが **REASON** パラメーターに対して戻す理由コードです。アプリケーションが GMCONV オプションを指定し、ユーザー作成出口を起動してメッセージ・データの一部またはすべてを変換する場合、その出口が、**REASON** パラメーターに戻される値を決定します。このため、このセクションで後述する値以外が戻ることがあります。

CMPCOD が CCOK の場合

RCNONE

(0, X'000') レポートする理由コードはありません。

CMPCOD が CCWARN の場合

RC2120

(2120, X'848') 変換されたデータが、バッファーには大きすぎる。

RC2190

(2190, X'88E') 変換されたストリングが、フィールドには大きすぎる。

RC2150

(2150, X'866') DBCS ストリングが無効である。

RC2110

(2110, X'83E') メッセージ形式が無効である。

RC2243

(2243, X'8C3') 各メッセージ・セグメントが異なる CCSID をもつ。

RC2244

(2244, X'8C4') 各メッセージ・セグメントが異なるエンコードをもつ。

RC2209

(2209, X'8A1') ロックされているメッセージがない。

RC2119

(2119, X'847') メッセージ・データが変換されなかった。

RC2272

(2272, X'8E0') メッセージ・データが一部変換されなかった。

RC2145

(2145, X'861') ソース・バッファー・パラメーターが無効。

RC2111

(2111, X'83F') ソース・エンコード文字セット ID が無効である。

RC2113

(2113, X'841') メッセージ内のパック 10 進数のエンコードが認識できない。

RC2114

(2114, X'842') メッセージ内の浮動小数点のエンコードが認識できない。

RC2112

(2112, X'840') ソース整数エンコードが認識できない。

RC2143

(2143, X'85F') ソース長パラメーターが無効である。

RC2146

(2146, X'862') ターゲット・バッファー・パラメーターが無効である。

RC2115

(2115, X'843') ターゲット・エンコード文字セット ID が無効である。

RC2117

(2117, X'845') 受信側で指定されたパック 10 進数のエンコードが認識できない。

RC2118

(2118, X'846') 受信側で指定された浮動小数点のエンコードが認識できない。

RC2116

(2116, X'844') ターゲット整数エンコードが認識できない。

RC2079

(2079, X'81F') 切り捨てられたメッセージが戻された (処理は完了している)。

RC2080

(2080, X'820') 切り捨てられたメッセージが戻された (処理は完了していない)。

CMPCOD が CCFAIL の場合

RC2004

(2004, X'7D4') バッファー・パラメーターが無効である。

RC2005

(2005, X'7D5') バッファー長パラメーターは無効です。

RC2219

(2219, X'8AB') 前の呼び出しが完了する前に MQI 呼び出しが再入力されました。

RC2009

(2009, X'7D9') キュー・マネージャーとの接続が失われました。

RC2010

(2010, X'7DA') データ長パラメーターが無効である。

RC2016

(2016, X'7E0') キューからの読み取りが禁止されている。

RC2186

(2186, X'88A') 読み取りメッセージ・オプションの構造体が無効である。

RC2018

(2018, X'7E2') 接続ハンドルが無効です。

RC2019

(2019, X'7E3') オブジェクト・ハンドルが無効です。

RC2241

(2241, X'8C1') メッセージ・グループが不完全である。

RC2242

(2242, X'8C2') 論理メッセージが不完全である。

RC2259

(2259, X'8D3') ブラウズの指定が不整合である。

RC2245

(2245, X'8C5') 作業単位の指定が不整合である。

RC2246

(2246, X'8C6') カーソル下のメッセージが取り出し対象として無効である。

RC2247

(2247, X'8C7') 突き合わせオプションが無効である。

RC2026

(2026, X'7EA') メッセージ記述子が無効である。

RC2250

(2250, X'8CA') メッセージ順序番号が無効である。

RC2033

(2033, X'7F1') メッセージが使用できない。

RC2034

(2034, X'7F2') ブラウズ・カーソルがメッセージに位置付けされていない。

RC2036

(2036, X'7F4') ブラウズのためにキューがオープンされていない。

RC2037

(2037, X'7F5') 入力のためにキューがオープンされていない。

RC2041

(2041, X'7F9') オープンされた後でオブジェクト定義が変更された。

RC2101

(2101, X'835') オブジェクトが損傷しました。

RC2046

(2046, X'7FE') オプションが無効であるか、矛盾しています。

RC2052

(2052, X'804') キューが削除されました。

RC2058

(2058, X'80A') キュー・マネージャー名が無効であるか、認識されていません。

RC2059

(2059, X'80B') キュー・マネージャーを接続に使用できません。

RC2161

(2161, X'871') キュー・マネージャーが静止しています。

RC2162

(2162, X'872') キュー・マネージャーのシャットダウン中です。

RC2102

(2102, X'836') 使用できるシステム・リソースが不足しています。

RC2071

(2071, X'817') ストレージが不足しています。

RC2024

(2024, X'7E8') 現行の作業単位内では、これ以上メッセージを処理できない。

RC2072

(2072, X'818') 同期点サポートが利用できない。

RC2195

(2195, X'893') 予期しないエラーが発生しました。

RC2255

(2255, X'8CF') 作業単位がキュー・マネージャーから使用不可。

RC2090

(2090, X'82A') MQGMO での待機間隔が無効である。

RC2256

(2256, X'8D0') 提供された MQGMO のバージョンが違っている。

RC2257

(2257, X'8D1') 提供された MQMD のバージョンが違っている。

RPG 宣言

```

C*.1.....2.....3.....4.....5.....6.....7..
C                                CALLP      MQGET(HCONN : HOBJ : MSGDSC : GMO :
C                                BUFLN : BUFFER : DATLEN :
C                                CMPCOD : REASON)

```


呼び出しのプロトタイプ定義は次のようになります。

```
D*..1.....2.....3.....4.....5.....6.....7..
DMQGET          PR          EXTPROC('MQGET')
D* Connection handle
D HCONN          10I 0 VALUE
D* Object handle
D HOBJ          10I 0 VALUE
D* Message descriptor
D MSGDSC          364A
D* Options that control the action of MQGET
D GMO          112A
D* Length in bytes of the Buffer area
D BUFLen          10I 0 VALUE
D* Area to contain the message data
D BUFFER          * VALUE
D* Length of the message
D DATLEN          10I 0
D* Completion code
D CMPCOD          10I 0
D* Reason code qualifying CMPCOD
D REASON          10I 0
```

IBM i IBM i での MQINQ (オブジェクト属性の照会)

MQINQ 呼び出しは、オブジェクトの属性が入っている整数の配列と一連の文字ストリングを戻します。

次のタイプのオブジェクトが有効です。

- キュー
- 名前リスト
- プロセス定義
- キュー・マネージャー
- [1316 ページの『構文』](#)
- [1316 ページの『使用上の注意』](#)
- [1317 ページの『パラメーター』](#)
- [1324 ページの『RPG 宣言』](#)

構文

MQINQ (*HCONN*, *HOBJ*, *SELCNT*, *SELS*, *IACNT*, *INTATR*, *CALEN*, *CHRRATR*, *CMPCOD*, *REASON*)

使用上の注意

1. 戻される値は、選択された属性のスナップショットです。返された値にアプリケーションが対処するまでに、属性が変更されないという保証はありません。
2. モデル・キューをオープンする場合は、動的ローカル・キューが作成されます。このことは、属性を照会するためにモデル・キューをオープンした場合であっても該当します。
動的キューの属性は、(特定の例外はありますが) その動的キューを作成した時のモデル・キューの属性と同じです。その後、このキューに対して MQINQ 呼び出しを使用すると、キュー・マネージャーは、モデル・キューの属性ではなく動的キューの属性を返します。動的キューに継承されるモデル・キューの属性の詳細については、[表 1](#) を参照してください。
3. 照会対象のオブジェクトが別名キューの場合、MQINQ 呼び出しから返される属性値は、それらの別名キューの属性であって、別名から解決される基本キューの属性ではありません。
4. 照会するオブジェクトがクラスター・キューの場合、以下のように、照会できる属性は、そのキューをどのようにオープンしたかによって決まります。
 - 照会に加えて、入力、ブラウズ、または設定のうち 1 つ以上の作業を目的としてクラスター・キューをオープンする場合、正常にオープンするためには、そのクラスター・キューのローカル・インスタンスが存在していなければなりません。この場合、照会できる属性は、ローカル・キューに有効な属性です。

- ・ 照会のみ、または照会と出力を目的としてクラスター・キューをオープンする場合、照会できる属性は、以下の属性のみです。この場合、**QType** 属性の値は QTCLUS です。

- CAQD
- CAQN
- IADBND
- IADPER
- IADPRI
- IAIPUT
- IAQTYP

バインドを固定せずにクラスター・キューをオープンした場合 (つまり、MQOPEN 呼び出しで OOBNDN を指定した場合、**DefBind** 属性の値が BNDNOT のときに OOBNDQ を指定した場合) は、そのキューに対する後続の MQINQ 呼び出しで、クラスター・キューの別のインスタンスが照会されることがあります。ただし、一般的にはすべてのインスタンスが同じ属性値を持っています。

クラスター・キューの詳細については、[キュー・マネージャー・クラスターの構成](#)を参照してください。

5. いくつかの属性を照会し、その後に MQSET 呼び出しを使用してそれらの一部を設定する場合は、設定する属性をセレクター配列の先頭に配置し、同じ配列を (カウントを減らして) MQSET で使用できるようにすると便利です。
6. 複数の警告状態が発生した場合 (**CMPCOD** パラメーターを参照)、返される理由コードは、以下のリストで該当するもののうち、最初の理由コードです。
 - a. RC2068
 - b. RC2022
 - c. RC2008
7. オブジェクト属性の詳細については、以下を参照してください。
 - ・ [1380 ページの『キューの属性』](#)
 - ・ [1408 ページの『名前リストの属性』](#)
 - ・ [1410 ページの『IBM i でのプロセス定義の属性』](#)
 - ・ [1412 ページの『IBM i でのキュー・マネージャーの属性』](#)
8. 新規ローカル・キュー SYSTEM.ADMIN.COMMAND.EVENT は、コマンドが発行されるときに毎回生成されるメッセージをキューイングするために使用されます。以下に示すように、CMDEV キュー・マネージャー属性の設定に応じて、ほとんどのコマンドに対するメッセージはこのキューに書き込まれます。
 - ・ **ENABLED** - すべての成功コマンドについて、コマンド・イベント・メッセージが生成され、このキューに書き込まれます。
 - ・ **NODISPLAY** - DISPLAY (MQSC) コマンド以外のすべての成功コマンドと Inquire (PCF) コマンドについて、コマンド・イベント・メッセージが生成され、このキューに書き込まれます。
 - ・ **DISABLED** - コマンド・イベント・メッセージは生成されません (これはキュー・マネージャーの初期デフォルト値です)。

パラメーター

MQINQ 呼び出しには、以下のパラメーターがあります。

HCONN (10 桁の符号付き整数) - 入力

接続ハンドル。

このハンドルは、キュー・マネージャーに対する接続を表します。**HCONN** の値は、先行の MQCONN または MQCONNX 呼び出しによって戻されたものです。

HOBJ (10 桁の符号付き整数) - 入力

オブジェクト・ハンドル

このハンドルが、必要な属性を備えているオブジェクト (任意のタイプ) を表します。このハンドルは、OOINQ オプションを指定した先行の MQOPEN 呼び出しによって戻されたものでなければなりません。

SELCNT (10 桁の符号付き整数) - 入力

セレクトーのカウンタ。

これは、SELS 配列で提供されるセレクトーのカウンタです。これは戻される属性の数です。ゼロは有効な値です。許可される最大数は 256 です。

SELS (10 桁の符号付き整数 x SELCNT) - 入力

属性セレクトーの配列。

これは、**SELCNT** 個の属性セレクトーの配列です。各セレクトーは、必要な値を持つ属性 (整数または文字) を識別します。

各セレクトーは、HOBJ が表すオブジェクト・タイプに対して有効でなければなりません。そうでない場合、その呼び出しは完了コード CCFAIL および理由コード RC2067 で失敗します。

特殊なケースのキューの場合は、以下のようになります。

- セレクトーが、いずれのタイプのキューについても無効である場合、その呼び出しは完了コード CCFAIL および理由コード RC2067 で失敗します。
- セレクトーが、オブジェクトのタイプを除くタイプ (複数の場合あり) のキューにしか当てはまらない場合、その呼び出しは完了コード CCWARN および理由コード RC2068 で成功します。
- 照会対象のキューがクラスター・キューである場合、有効なセレクトーは、そのキューがどのように解決されるかによって異なります。詳細については、使用上の注意 4 を参照してください。

選択子は任意の順序で指定できます。整数属性セレクトー (IA* セレクトー) に対応する属性値は、SELS でのこれらのセレクトーの出現順序と同じ順序で INTATR に返されます。文字属性セレクトー (CA* セレクトー) に対応する属性値は、CHRATR に、セレクトーの出現順序と同じ順序で返されます。IA* セレクトーが CA* セレクトーに挟まれていても構いません。重要なのは、それぞれのタイプにおける相対順序のみです。

注:

- 整数および文字属性セレクトーは、2 つの異なる範囲内に割り当てられます。IA* セレクトーは、IAFRST から IALAST までの範囲に、CA* セレクトーは、CAFRST から CALAST までの範囲にあります。

各範囲について、定数 IALSTU および CALSTU によって、キュー・マネージャーが受け入れる最大値を定義しています。

- すべての IA* セレクトーが先頭にある場合は、SELS および INTATR 配列で同じエレメント番号を使用して、対応するエレメントのアドレスを示すことができます。

照会できる属性が、以下の表にリストされています。CA* セレクトーの場合、CHRATR 内の結果ストリングの長さ (バイト単位) を定義する定数は、括弧内に指定します。

表 785. キュー用の MQINQ 属性セレクトー		
セレクトー	説明	注記
CAALTD	最新の変更の日付 (LNDATE)。	1
CAALTT	最新の変更の時刻 (LNTIME)。	1
CABRQN	超過バックアウト・リキュー名 (LNQN)。	5
CABASQ	別名が解決されるキューの名前 (LNQN)。	
CACFSN	カップリング・ファシリティーの構造体名 (LNCFSN)。	3

表 785. キュー用の MQINQ 属性セレクトター (続き)		
セレクトター	説明	注記
CACLN	クラスター名 (LNCLUN)。	1
CACLNL	クラスターの名前リスト (LNNLN)。	1
CACRTD	キュー作成日付 (LNCRTD)。	
CACRTT	キュー作成時刻 (LNCRTT)。	
CAINIQ	開始キュー名 (LNQN)。	
CAPRON	プロセス定義の名前 (LNPRON)。	
CAQD	キューの説明 (LNQD)。	
CAQN	キュー名 (LNQN)。	
CARQMN	リモート・キュー・マネージャーの名前 (LNQMN)。	
CARQN	リモート・キュー・マネージャー上で認識されているリモート・キューの名前 (LNQN)。	
CATRGD	トリガー・データ (LNTRGD)。	5
CAXQN	伝送キュー名 (LNQN)。	
IABTHR	バックアウトしきい値。	5
IACDEP	キュー上のメッセージの数。	
IADBND	デフォルトのバインディング。	1
IADINP	デフォルトの入力用オープンのオプション。	5
IADPER	デフォルトのメッセージ持続性。	
IADPRI	デフォルトのメッセージ優先度。	5
IADEFT	キュー定義タイプ。	
IADIST	配布リスト・サポート。	2
IAHGB	バックアウト・カウントを強化するかどうか。	5
IAIGET	読み取り操作が許されているかどうかを判別します。	
IAIPUT	書き込み操作が許されているかどうかを判別します。	
IAMLEN	最大メッセージ長。	
IAMDEP	キューに許可するメッセージの最大数。	
IAMDS	メッセージ優先順位が関係あるかどうかを判別します。	5
IAOIC	キューを入力用にオープンしている MQOPEN 呼び出しの数。	
IAOOC	キューを出力用にオープンしている MQOPEN 呼び出しの数。	
IAQDHE	キューのサイズ上位イベントの制御属性。	4、5
IAQDHL	キュー項目数の上限。	4、5
IAQDLE	キューのサイズ下位イベントの制御属性。	4、5
IAQDLL	キュー項目数の下限。	4、5
IAQDME	キュー・サイズ最大イベントの制御属性。	4、5
IAQSI	キュー・サービス間隔の制限。	4、5

表 785. キュー用の MQINQ 属性セレクトター (続き)		
セレクトター	説明	注記
IAQSIE	キュー・サービス間隔イベントの制御属性。	4、5
IAQTYP	キュー・タイプ。	
IAQSGD	キュー共有グループ後処理。	3
IARINT	キュー保持期間間隔。	5
IASCOP	キュー定義の有効範囲。	4、5
IASHAR	キューを入力用に共有できるかどうか。	
IATRGC	トリガー制御。	
IATRGD	トリガー項目数。	5
IATRGP	トリガーのしきい値メッセージ優先順位。	5
IATRGT	トリガー・タイプ。	
IAUSAG	使用法	
CLWLUSEQ	リモート・キューを使用します。	

注：

1. 以下のプラットフォームでサポートされます。

- ▶ **AIX** AIX
- ▶ **IBM i** IBM i
- ▶ **Windows** Windows
- ▶ **z/OS** z/OS

および、これらのシステムに接続された IBM MQ MQI clients。

2. 以下のプラットフォームでサポートされます。

- ▶ **AIX** AIX
- ▶ **IBM i** IBM i
- ▶ **Windows** Windows

および、これらのシステムに接続された IBM MQ クライアント。

3. ▶ **z/OS** z/OS でサポートされます。

4. ▶ **z/OS** z/OS ではサポートされません。

5. VSE/ESA ではサポートされません。

表 786. 名前リストに関する MQINQ 属性セレクトター		
セレクトター	説明	注記
CAALTD	最新の変更の日付 (LNDATE)	1
CAALTT	最新の変更の時刻 (LNTIME)	1
CALSTD	名前リストの説明 (LNNLD)	1
CALSTN	名前リスト・オブジェクトの名前 (LNNLN)	1

表 786. 名前リストに関する MQINQ 属性セクター (続き)		
セクター	説明	注記
CANAMS	名前リスト内の名前 (LNQN x リスト内の名前の数)	1
IANAMC	名前リスト内の名前の数	1
IAQSGD	キュー共有グループ後処理	3

表 787. プロセス定義用 MQINQ 属性セクター		
セクター	説明	注記
CAALTD	最新の変更の日付 (LNDATE)	1
CAALTT	最新の変更の時刻 (LNTIME)	1
CAAPPI	アプリケーション ID (LNPROA)	5
CAENVD	環境データ (LNPROE)	5
CAPROD	プロセス定義の説明 (LNPROD)	5
CAPRON	プロセス定義の名前 (LNPRON)	5
CAUSRD	ユーザー・データ (LNPROU)	5
IAAPPT	アプリケーション・タイプ	5
IAQSGD	キュー共有グループ後処理	3

表 788. キュー・マネージャー用の MQINQ 属性セクター		
セクター	説明	注記
CAALTD	最新の変更の日付 (LNDATE)	1
CAALTT	最新の変更の時刻 (LNTIME)	1
CACADX	自動チャンネル定義出口名 (LNEXN)	1
CACLWD	クラスター・ワークロード出口に渡されるデータ (LNEXDA)	1
CACLWX	クラスター・ワークロード出口名 (LNEXN)	1
CACMDQ	システム・コマンド入力キュー名 (LNQN)	5
CADLQ	送達不能キュー名 (LNQN)	5
CADXQN	デフォルトの伝送キュー名 (LNQN)	5
CAQMD	キュー・マネージャーの説明 (LNQMD)	5
CAQMID	キュー・マネージャー ID (LNQMID)	1
CAQMN	ローカル・キュー・マネージャー名 (LNQMN)	5
CAQSGN	キュー共有グループ名 (LNQSGN)	3
CARPN	キュー・マネージャーがリポジトリ・サービスを提供するクラスターの名前 (LNQMN)	1
CARPNL	キュー・マネージャーがリポジトリ・サービスを提供するクラスターの名前が入った 名前リスト・オブジェクトの名前 (LNNLN)	1
CMDEV	コマンドの発行時に生成されたメッセージを キューに書き込むかどうかを決める 制御属性	8

表 788. キュー・マネージャー用の MQINQ 属性セクター (続き)		
セクター	説明	注記
IAAUTE	権限イベントの制御属性	4、5
IACAD	自動チャネル定義の制御属性	2
IACADE	自動チャネル定義イベントの制御属性	2
IACLXQ	デフォルトのクラスタ伝送キュータイプ。	4
IACLWL	クラスター・ワークロード長	1
IACCSI	コード化文字セット ID	5
IACMDL	キュー・マネージャーでサポートされるコマンド・レベル	5
IACFGE	構成イベントの制御属性	3
IADIST	配布リスト・サポート	2
IAINHE	禁止イベントの制御属性	4、5
IALCLE	ローカル・イベントの制御属性	4、5
IAMHND	ハンドルの最大数	5
IAMLEN	最大メッセージ長	5
IAMPRI	最高の優先順位	5
IAMUNC	1 つの作業単位内のコミットされていないメッセージの最大数	5
IAPFME	パフォーマンス・イベントの制御属性	4、5
IAPLAT	キュー・マネージャーがあるプラットフォーム	5
IARMTE	リモート・イベントの制御属性	4、5
IASSE	開始 / 停止イベントの制御属性	4、5
IASYNC	同期点の使用可能性	5
IATRLFT	未使用の非管理トピックの存続時間	
IATRGI	トリガー間隔	5

IACNT (10 桁の符号付き整数) - 入力

整数属性のカウント。

これは、INTATR 配列内のエレメントの数です。ゼロは有効な値です。

これが **SELS** パラメーター内の IA* セクターの数以上であれば、要求されたすべての整数属性が返されます。

INTATR (10 桁の符号付き整数 x IACNT) - 出力

整数属性の配列。

これは、IACNT 個の整数属性値の配列です。

整数属性値は、**SELS** パラメーター内の IA* セクターと同じ順序で返されます。この配列に IA* セクターより多くの数のエレメントが入っている場合、それらの余分なエレメントは変更されません。

HOBJS がキューを表しているが、属性セクターがそのキューのタイプに当てはまらない場合は、INTATR 配列の対応するエレメントに特定の値 IAVNA が返されます。

CALEN (10 桁の符号付き整数) - 入力

文字属性バッファの長さ。

これは、**CHRATR** パラメーターの長さ (バイト単位) です。

これは、少なくとも要求された文字属性 (**SELS** を参照) の長さの合計でなければなりません。ゼロは有効な値です。

CHRATR (1 バイト文字ストリング x CALEN) - 出力

文字属性。

これは、各文字属性が連結されて返されるバッファです。バッファの長さは、**CALEN** パラメーターによって指定されます。

文字属性は、**SELS** パラメーター内の CA* セレクターと同じ順序で返されます。各属性ストリングの長さは、属性 (**SELS** を参照) ごとに固定されており、必要に応じて値の右側にブランクが埋め込まれます。バッファが、要求された文字属性 (埋め込み文字も含む) をすべて含めるのに必要なバッファより大きい場合、戻された最後の属性値を超えるバイトは変更されません。

HOBJS がキューを表しているが、属性セレクターがそのタイプのキューに当てはまらない場合は、すべてがアスタリスク (*) で構成された文字ストリングが、その属性の値として **CHRATR** に返されます。

CMPCOD (10 桁の符号付き整数) - 出力

完了コード。

これは、以下のいずれかになります。

CCOK

正常終了。

CCWARN

警告 (部分完了)。

CCFAIL

呼び出し失敗。

REASON (10 桁の符号付き整数) - 出力

CMPCOD を限定する理由コード。

CMPCOD が CCOK の場合

RCNONE

(0, X'000') レポートする理由コードはありません。

CMPCOD が CCWARN の場合

RC2008

(2008, X'7D8') 文字属性用に十分なスペースがありません。

RC2022

(2022, X'7E6') 整数属性用に十分なスペースがありません。

RC2068

(2068, X'814') セレクターがキュー・タイプに適用できません。

CMPCOD が CCFAIL の場合

RC2219

(2219, X'8AB') 前の呼び出しが完了する前に MQI 呼び出しが再入力されました。

RC2006

(2006, X'7D6') 文字属性の長さが無効である。

RC2007

(2007, X'7D7') 文字属性ストリングが無効である。

RC2009

(2009, X'7D9') キュー・マネージャーとの接続が失われました。

RC2018

(2018, X'7E2') 接続ハンドルが無効です。

RC2019

(2019, X'7E3') オブジェクト・ハンドルが無効です。

RC2021

(2021, X'7E5') 整数属性のカウントが無効です。

RC2023

(2023, X'7E7') 整数属性の配列が無効です。

RC2038

(2038, X'7F6') キューが照会用にオープンされていません。

RC2041

(2041, X'7F9') オープンされた後でオブジェクト定義が変更された。

RC2101

(2101, X'835') オブジェクトが損傷しました。

RC2052

(2052, X'804') キューが削除されました。

RC2058

(2058, X'80A') キュー・マネージャー名が無効であるか、認識されていません。

RC2059

(2059, X'80B') キュー・マネージャーを接続に使用できません。

RC2162

(2162, X'872') キュー・マネージャーのシャットダウン中です。

RC2102

(2102, X'836') 使用できるシステム・リソースが不足しています。

RC2065

(2065, X'811') セレクターのカウントが無効である。

RC2067

(2067, X'813') 属性選択子が無効です。

RC2066

(2066, X'812') セレクターのカウントが大きすぎる。

RC2071

(2071, X'817') ストレージが不足しています。

RC2195

(2195, X'893') 予期しないエラーが発生しました。

RPG 宣言

```

C*..1.....2.....3.....4.....5.....6.....7..
C                                CALLP      MQINQ(HCONN : HOBJ : SELCNT :
C                                SELS(1) : IACNT : INTATR(1) :
C                                CALEN : CHRATR : CMPCOD :
C                                REASON)

```

呼び出しのプロトタイプ定義は次のようになります。

```

D*..1.....2.....3.....4.....5.....6.....7..
DMQINQ      PR      EXTPROC('MQINQ')
D* Connection handle
D HCONN      10I 0 VALUE
D* Object handle
D HOBJ      10I 0 VALUE
D* Count of selectors
D SELCNT      10I 0 VALUE
D* Array of attribute selectors
D SELS      10I 0
D* Count of integer attributes
D IACNT      10I 0 VALUE

```



```

D* Array of integer attributes
D INTATR                                10I 0
D* Length of character attributes buffer
D CALEN                                10I 0 VALUE
D* Character attributes
D CHRATR                                *   VALUE
D* Completion code
D CMPCOD                                10I 0
D* Reason code qualifying CMPCOD
D REASON                                10I 0

```

IBM i IBM i での MQINQMP (メッセージ・プロパティの照会)

MQINQMP 呼び出しは、メッセージのプロパティの値を戻します。

- [1325 ページの『構文』](#)
- [1325 ページの『パラメーター』](#)
- [1329 ページの『RPG 宣言』](#)

構文

MQINQMP (*Hconn*, *Hmsg*, *InqPropOpts*, *Name*, *PropDesc*, *Type*, *ValueLength*, *Value*, *DataLength*, *CompCode*, *Reason*)

パラメーター

MQINQMP 呼び出しには、以下のパラメーターがあります。

HCONN (10 桁の符号付き整数) - 入力

このハンドルは、キュー・マネージャーに対する接続を表します。 *Hconn* の値は、**Hmsg** パラメーターで指定されているメッセージ・ハンドルを作成するために使用された接続ハンドルと一致していなければなりません。

メッセージ・ハンドルが、HCUNAS を使用して作成されている場合、メッセージ・ハンドルのプロパティを照会するスレッドで、有効な接続が確立されている必要があります。接続がないと、呼び出しは RC2009 で失敗します。

HMSG (20 桁の符号付き整数) - 入力

これは照会されるメッセージ・ハンドルです。 値は、前の **MQCRTMH** 呼び出しで戻されたものです。

INQOPT (MQIMPO) - 入力

詳細については、[MQIMPO](#) データ・タイプを参照してください。

PRNAME (MQCHARV) - 入力

これは、照会するプロパティの名前を記述します。

この名前プロパティが見つからない場合、呼び出しは失敗し、理由は RC2471 となります。

プロパティ名の末尾にパーセント記号 (%) の文字を使用できます。このワイルドカードは、ピリオド (.) を含むゼロ個以上の文字と一致します。そのため、アプリケーションが多数のプロパティの値を照会できます。MQINQMP にオプション IPINQF を指定して呼び出すことにより、最初にマッチングするプロパティを取得します。その後、オプション IPINQN を指定して再び呼び出し、次にマッチングするプロパティを取得します。それ以上マッチングするプロパティがなくなると、呼び出しは失敗して RC2471 となります。InqPropOpts 構造体の *ReturnedName* フィールドが、プロパティに返された名前アドレスまたはオフセットで初期化されると、MQINQMP から戻った時点で、一致したプロパティの名前がこのフィールドに取り込まれます。InqPropOpts 構造体の *ReturnedName* の *VSBufSize* フィールドが、戻されるプロパティ名の長さより小さい値の場合、完了コードは CCFAIL に設定され、理由は RC2465 になります。

既知の同義語のあるプロパティは以下のように戻されます。

1. 接頭部「mqps」が付いたプロパティ。IBM MQ プロパティ名とともに返されます。例えば "mqps.Top" ではなく、"MQTopicString" という名前で返されます。
2. 接頭部が「jms」のプロパティ。または「mcd」JMS ヘッダー・フィールド名として返されます。例えば "jms.Exp" ではなく、"JMSEExpiration" という名前で返されます。
3. 接頭部が「usr」のプロパティ。は、その接頭部なしで返されます。例えば "usr.Color" ではなく、"Color" という名前で返されます。

同義語のあるプロパティは 1 回のみ返されます。

RPG プログラミング言語では、すべてのプロパティを照会する場合と、"usr." で始まるすべてのプロパティを照会する場合のために、以下のマクロ変数が定義されています。

INQALL

メッセージのすべてのプロパティに対する照会。

INQUSR

先頭が「usr.」の、メッセージのすべてのプロパティに対する照会。返される名前は、「usr」なしで返されます。接頭部。

IPINQN が指定されていても、前の呼び出し以来に名前が変更されている場合、または初回呼び出しの場合には、暗黙のうちに IPINQF が指定されたものとみなされます。

プロパティ名の使用については、[プロパティ名およびプロパティ名に関する制約事項](#)を参照してください。

PRPDSC (MQPD) - 出力

この構造体を使用して、プロパティの属性を定義します。その中には、プロパティがサポートされていない場合に起きること、プロパティが属するメッセージ・コンテキスト、およびプロパティのコピー先のメッセージが含まれます。この構造体の詳細については、[MQPD](#) を参照してください。

TYPE (10 桁の符号付き整数) - 入出力

MQINQMP 呼び出しから戻される際に、このパラメーターは *Value* のデータ・タイプに設定されます。データ・タイプは次のいずれかです。

TYPBOL

ブール値。

TYPBST

バイト・ストリング。

TYPI8

8 ビットの符号付き整数。

TYPI16

16 ビットの符号付き整数。

TYPI32

32 ビットの符号付き整数。

TYPI64

64 ビットの符号付き整数。

TYPF32

32 ビットの浮動小数点数。

TYPF64

64 ビットの浮動小数点数。

TYPSTR

文字ストリング。

TYPNUL

プロパティは存在しますがヌル値です。

プロパティ値のデータ・タイプが認識されていないものである場合には、TYPSTR が戻され、値のストリング表記が *Value* 領域に入れられます。データ・タイプのストリング表記は、*ILOPT* パラメーターの *IPTYP* フィールドに含まれています。警告完了コードが戻され、理由は RC2467 になります。

さらに、オプション IPCTYP が指定されている場合には、プロパティ値の変換が要求されます。プロパティを戻す際のデータ・タイプを指定するには、*Type* を入力として使用します。データ・タイプ変換について詳しくは、[1107 ページの『IBM i での MQIMPO \(メッセージ・プロパティ照会オプション\)』](#)の IPCTYP オプションの説明を参照してください。

タイプ変換を要求しない場合、入力で以下の値を使用できます。

TYPAST

プロパティの値は、そのデータ・タイプを変換せずに戻されます。

VALLEN (10 桁の符号付き整数) - 入力

Value 域のバイト単位の長さ。

値を戻す必要のないプロパティの場合は、ゼロを指定します。これらのプロパティには、アプリケーションによってヌル値または空ストリングを持つように設計されているものがあります。また、IPQLEN オプションが指定された場合にもゼロを指定してください。その場合、値は戻されません。

VALUE (1 バイトのビット・ストリング x VALLEN) - 出力

これは、照会プロパティ値を含む領域です。バッファは、戻される値に適した境界に位置合わせされなければなりません。この処理に失敗すると、後で値にアクセスする際にエラーが発生する可能性があります。

VALLEN がプロパティ値の長さより小さい場合、プロパティ値のうち可能な限り多くの部分が VALUE に移され、呼び出しは、完了コード CCFAIL、理由 RC2469 で失敗します。

VALUE 中のデータの文字セットは、INQOPT パラメーターの IPRETCSI フィールドによって示されます。VALUE 中のデータのエンコード方式は、INQOPT パラメーターの IPRETENC フィールドによって示されます。

VALLEN パラメーターがゼロである場合、VALUE は参照されません。

DATLEN (10 桁の符号付き整数) - 出力

これは、Value 域に返される実際のプロパティ値の長さ (バイト数) です。

DataLength がプロパティ値の長さより小さい場合も、MQINQMP の呼び出しから戻った時点で DataLength にはデータが入れられます。これにより、アプリケーションは、プロパティ値を入れるのに必要なバッファのサイズを判別して、適切なサイズのバッファを用いて呼び出しを再発行することができます。

以下の値が戻される場合もあります。

Type パラメーターが TYPSTR または TYPBST に設定されている場合、

VLEMP

プロパティは存在しますが、文字やバイトが含まれていません。

CMPCOD (10 桁の符号付き整数) - 出力

完了コード。以下のいずれかです。

CCOK

正常終了。

CCWARN

警告 (部分完了)。

CCFAIL

呼び出し失敗。

REASON (10 桁の符号付き整数) - 出力

CompCode を限定する理由コード。

CMPCOD が CCOK の場合

RCNONE

(0, X'000') レポートする理由コードはありません。

CompCode が CCWARN の場合

RC2492

(2492, X'09BC') 戻されたプロパティ名が変換されなかった。

RC2466

(2466, X'09A2') プロパティ値が変換されなかった。

RC2467

(2467, X'09A3') プロパティのデータ・タイプがサポートされていない。

RC2421

(2421, X'0975') プロパティを含む MQRFH2 フォルダを構文解析できなかった。

CMPCOD が CCFAIL の場合

RC2204

(2204, X'089C') アダプターが利用できません。

RC2130

(2130, X'0852') アダプター・サービス・モジュールをロードできない。

RC2157

(2157, X'086D') 1 次 ASID とホーム ASID とが異なっている。

RC2004

(2004, X'07D4') 値パラメーターが無効である。

RC2005

(2005, X'07D5') 値長パラメーターが無効である。

RC2219

(2219, X'08AB') 前の呼び出しが完了する前に MQI 呼び出しが入力された。

RC2009

(2009, X'07D9') キュー・マネージャーとの接続が失われました。

RC2010

(2010, X'07DA') データ長パラメーターが無効である。

RC2464

(2464, X'09A0') メッセージ・プロパティ照会オプションの構造体が無効である。

RC2460

(2460, X'099C') メッセージ・ハンドルが無効。

RC2499

(2499, X'09C3') メッセージ・ハンドルがすでに使用中。

RC2064

(2046, X'07F8') オプションが無効、または整合性がない。

RC2482

(2482, X'09B2') プロパティ記述子の構造体が無効である。

RC2470

(2470, X'09A6') 実際のデータ・タイプから要求されたデータ・タイプへの変換がサポートされていない。

RC2442

(2442, X'098A') プロパティ名が無効である。

RC2465

(2465, X'09A1') 戻される名前バッファにとってプロパティ名が大きすぎる。

RC2471

(2471, X'09A7') プロパティが使用できない。

RC2469

(2469, X'09A5') Value 域にとってプロパティ値が大きすぎる。

RC2472

(2472, X'09A8') 値データ中に数字フォーマット・エラーが発生した。

RC2473

(2473, X'09A9') 要求されたプロパティ・タイプが無効である。

RC2111

(2111, X'083F') プロパティ名エンコード文字セット ID が無効である。

RC2071

(2071, X'0871') 使用できるストレージが十分でない。

RC2195

(2195, X'0893') 予期しないエラーが発生した。

これらのコードの詳細については、以下を参照してください。

- [IBM MQ for z/OS メッセージ、完了状況、および理由コードの IBM MQ for z/OS](#)
- [メッセージとエラーコード](#) (その他のすべての IBM MQ プラットフォームの場合)

RPG 宣言

```
C*..1.....2.....3.....4.....5.....6.....7..  
C                                CALLP      MQINQMP(HCONN : HMSG : INQOPT :  
                                           PRNAME : PRPDSC : TYPE :  
                                           VALLEN : VALUE : DATLEN :  
                                           CMPCOD : REASON)
```

呼び出しのプロトタイプ定義は次のようになります。

```
DMQINQMP      PR      EXTPROC('MQINQMP')  
D* Connection handle  
D HCONN              10I 0 VALUE  
D* Message handle  
D HMSG              20I 0 VALUE  
D* Options that control the action of MQINQMP  
D INQOPT            72A  
D* Property name  
D PRNAME            32A  
D* Property descriptor  
D PRPDSC            24A  
D* Property data type  
D TYPE              10I 0  
D* Length in bytes of the Value area  
D VALLEN            10I 0 VALUE  
D* Property value  
D VALUE              *    VALUE  
D* Length of the property value  
D DATLEN            10I 0  
D* Completion code  
D CMPCOD            10I 0  
D* Reason code qualifying CompCode  
D REASON            10I 0
```

IBM i IBM i での MQMHBUFF (メッセージ・ハンドルのバッファーへの変換)

MQMHBUFF はメッセージ・ハンドルをバッファーに変換するので、MQBUFMH 呼び出しの逆です。

- [1329 ページの『構文』](#)
- [1330 ページの『使用上の注意』](#)
- [1330 ページの『パラメーター』](#)
- [1332 ページの『RPG 宣言』](#)

構文

MQMHBUFF (*Hconn*, *Hmsg*, *MsgHBufOpts*, *Name*, *MsgDesc*, *BufferLength*, *Buffer*, *DataLength*, *CompCode*, *Reason*)

使用上の注意

MQMHBUF はメッセージ・ハンドルをバッファーに変換します。

MQGET API 出口と併用して、メッセージ・プロパティ API を使って特定のプロパティにアクセスしてから、これらのプロパティをバッファーに入れて、メッセージ・ハンドルではなく MQRFH2 ヘッダーを使用するように設計されているアプリケーションに渡すことができます。

この呼び出しは MQBUFMH 呼び出しの逆です。MQBUFMH 呼び出しを使用すると、バッファーからメッセージ・ハンドルにメッセージ・プロパティを構文解析できます。

パラメーター

MQMHBUF 呼び出しには、以下のパラメーターがあります。

HCONN (10 桁の符号付き整数) - 入力

このハンドルは、キュー・マネージャーに対する接続を表します。

HCONN の値は、HMSG パラメーターで指定されているメッセージ・ハンドルを作成するために使用された接続ハンドルと一致していなければなりません。

HCUNAS を使用してメッセージ・ハンドルが作成された場合は、メッセージ・ハンドルを削除することによって、スレッドで有効な接続を確立する必要があります。有効な接続が確立されていない場合、呼び出しは失敗し、RC2009 となります。

HMSG (20 桁の符号付き整数) - 入力

このハンドルは、バッファーに必要なメッセージ・ハンドルです。

値は、前の MQCRTMH 呼び出しで戻されたものです。

MHBOPT (MQMHBO) - 入力

アプリケーションでは、MQMHBO 構造体を使用することによって、メッセージ・ハンドルからバッファーを生成する方法を制御するためのオプションを指定することができます。

詳細は 1019 ページの『IBM i での MQBMHO (バッファーからメッセージ・ハンドルへの変換オプション)』を参照してください。

PRNAME (MQCHARV) - 入力

バッファーに書き込む 1 つ以上のプロパティの名前。

この名前に一致するプロパティが見つからない場合、呼び出しは失敗し、理由は RC2471 となります。

ワイルドカード

ワイルドカードを使用して、複数のプロパティをバッファーに書き込むことができます。それには、プロパティ名の末尾にパーセント記号 (%) を使用します。このワイルドカードは、0 個以上の文字 (ピリオド文字 '.' を含む) にマッチングします。

プロパティ名の使用については、[プロパティ名およびプロパティ名に関する制約事項](#)を参照してください。

MSGDSC (MQMD) - 入出力

MSGDSC 構造体は、バッファー域の内容を記述します。

出力上では、バッファー域のエンコード方式、文字セット ID、およびデータの形式を、呼び出しによって書き込まれるとおりに正しく記述するように、*Encoding*、*CodedCharSetId*、および *Format* フィールドが設定されます。

この構造体中のデータは、アプリケーションの文字セット内およびエンコード内にあります。

BUFLEN (10 桁の符号付き整数) - 入力

BUFLEN は、バッファー域の長さです (バイト単位)。

BUFFER (1 バイトのビット・ストリング x BUFLen) - 入出力

BUFFER は、メッセージ・バッファが入れられる領域を定義します。ほとんどのデータの場合、バッファを 4 バイトの境界に位置合わせする必要があります。

BUFFER に文字データまたは数値データが含まれている場合は、**MSGDSC** パラメーターの *CodedCharSetId* および *Encoding* フィールドを、データに適した値に設定します。これにより、必要に応じてデータを変換することができます。

メッセージ・バッファ中にプロパティがある場合はオプションで除去されます。これらのプロパティは、後で呼び出しから戻る際にメッセージ・ハンドルから使用できるようになります。

C プログラミング言語では、パラメーターは、**void** を示すポインターとして宣言されます。つまり、どのタイプのデータのアドレスもパラメーターとして指定できます。

BUFLen パラメーターがゼロの場合、**BUFFER** は参照されません。この場合、C または System/390 アセンブラで作成されたプログラムによって渡されるパラメーター・アドレスはヌルのこともあります。

DATLEN (10 桁の符号付き整数) - 出力

DATLEN は、バッファに入れて戻されるプロパティの長さです (バイト単位)。その値がゼロの場合、*PRNAME* で指定される値にマッチングするプロパティがなかったということであり、呼び出しは理由コード **RC2471** で失敗します。

BUFLen が、バッファにプロパティを格納するのに必要な長さよりも小さい場合、**MQMHBUFF** 呼び出しは失敗し、**RC2469** となります。しかし、**DATLEN** には値が入れられます。これにより、アプリケーションは、プロパティを入れるために必要なバッファのサイズを判別して、必要とされる値を **BUFLen** に指定して呼び出しを再発行することができます。

CMPCOD (10 桁の符号付き整数) - 出力

完了コード。以下のいずれかです。

CCOK

正常終了。

CCFAIL

呼び出し失敗。

REASON (10 桁の符号付き整数) - 出力

CMPCOD を限定する理由コード。

CMPCOD が **CCOK** の場合

RCNONE

(0, X'000') レポートする理由コードはありません。

CMPCOD が **CCFAIL** の場合

RC2204

(2204, X'089C') アダプターが利用できません。

RC2130

(2130, X'852') アダプター・サービス・モジュールをロードできません。

RC2157

(2157, X'86D') 1 次 ASID とホーム ASID が異なります。

RC2501

(2501, X'095C') メッセージ・ハンドルからバッファへの変換オプション構造体が無効です。

RC2004

(2004, X'07D4') バッファ・パラメーターが無効である。

RC2005

(2005, X'07D5') バッファ長パラメーターは無効です。

RC2219

(2219, X'08AB') 前の呼び出しが完了する前に MQI 呼び出しが入力された。

RC2009

(2009, X'07D9') キュー・マネージャーとの接続が失われました。

RC2010

(2010, X'07DA') データ長パラメーターが無効である。

RC2460

(2460, X'099C') メッセージ・ハンドルが無効。

RC2026

(2026, X'07EA') メッセージ記述子が無効である。

RC2499

(2499, X'09C3') メッセージ・ハンドルがすでに使用中。

RC2046

(2046, X'07FE') オプションが無効であるか、矛盾しています。

RC2442

(2442, X'098A') プロパティ名が無効である。

RC2471

(2471, X'09A7') プロパティが使用できない。

RC2469

(2469, X'09A5') BufferLength の値が小さすぎるため、指定されたプロパティを入れることができません。

RC2195

(2195, X'893') 予期しないエラーが発生しました。

RPG 宣言

```
C*..1.....2.....3.....4.....5.....6.....7..
C                                CALLP      MQMHBUFF(HCONN : HMSG : MHBOP :
                                           PRNAME : MSGDSC : BUFLN :
                                           BUFFER : DATLEN :
                                           CMPCOD : REASON)
```

呼び出しのプロトタイプ定義は次のようになります。

```
DMQMHBUFF      PR      EXTPROC('MQMHBUFF')
D* Connection handle
D HCONN          10I 0 VALUE
D* Message handle
D HMSG          20I 0 VALUE
D* Options that control the action of MQMHBUFF
D MHBOP          12A
D* Property name
D PRNAME        32A
D* Message descriptor
D MSGDSC        364A
D* Length in bytes of the Buffer area
D BUFLN         10I 0 VALUE
D* Area to contain the properties
D BUFFER        *   VALUE
D* Length of the properties
D DATLEN        10I 0
D* Completion code
D CMPCOD        10I 0
D* Reason code qualifying CompCode
D REASON        10I 0
```

IBM i IBM i での MQOPEN (オブジェクトのオープン)

MQOPEN の呼び出しはオブジェクトへのアクセスを確立します。

次のタイプのオブジェクトが有効です。

- キュー (配布リストを含む)
- 名前リスト
- プロセス定義
- キュー・マネージャー
- トピック

索引

- [1333 ページの『構文』](#)
- [1333 ページの『使用上の注意』](#)
- [1337 ページの『パラメーター』](#)
- [1344 ページの『RPG 宣言』](#)

構文

MQOPEN (*HCONN, OBJDSC, OPTS, HOBJ, CMPCOD, REASON*)

使用上の注意

1. 開かれるオブジェクトは、以下のいずれかです。

- 以下の目的を持つキュー。
 - メッセージを取得またはブラウズする (MQGET 呼び出しを使用)。
 - メッセージを書き込む (MQPUT 呼び出しを使用)。
 - キューの属性について照会する (MQINQ 呼び出しを使用)。
 - キューの属性を設定する (MQSET 呼び出しを使用)。

指定されたキューがモデル・キューである場合は、動的ローカル・キューが作成されます。

配布リストは、キューのリストを格納する特殊なタイプのキュー・オブジェクトです。これを開いてメッセージを書き込むことはできますが、メッセージの取得やブラウズ、あるいは属性の照会や設定を行うことはできません。詳しくは、使用上の注意 8 を参照してください。

QSGDISP(GROUP) があるキューは特別なタイプのキュー定義であり、MQOPEN または MQPUT1 呼び出しでは使用できません。

- 以下の目的を持つ名前リスト。
 - リスト内のキューの名前について照会する (MQINQ 呼び出しを使用)。
 - 以下の目的を持つプロセス定義。
 - プロセス属性について照会する (MQINQ 呼び出しを使用)。
 - 以下の目的を持つキュー・マネージャー。
 - ローカル・キュー・マネージャーの属性について照会する (MQINQ 呼び出しを使用)。
2. 1 つのアプリケーションで同じオブジェクトを 2 回以上オープンするのは、有効です。オープンするたびに異なるオブジェクト・ハンドルが返されます。返されるそれぞれのハンドルは、対応するオープンの実行対象となる関数において使用できます。
 3. オープンするオブジェクトが、キューではあるがクラスター・キューでない場合、ローカル・キュー・マネージャー内の名前解決はすべて、MQOPEN 呼び出しの時点で行われます。特定の MQOPEN 呼び出しに関して以下の処理のうちの 1 つ以上を実行できます。
 - 別名を基本キューの名前に解決する。
 - リモート・キューのローカル定義の名前を、リモート・キュー・マネージャーの名前と、そのキューがリモート・キュー・マネージャーで認識されている名前に解決する。
 - リモート・キュー・マネージャーの名前をローカル伝送キューの名前に解決する。

ただし、そのハンドルに対する後続の MQINQ 呼び出しまたは MQSET 呼び出しは、オープンされている名前に関連するものであり、ネーム・レゾリューションが行われた後の結果のオブジェクトとは関連がない点に注意してください。例えば、オープンされたオブジェクトが別名である場合、MQINQ 呼び出しによって戻される属性は別名の属性であり、別名が解決される先の基本キューの属性ではありません。しかし、対応する MQOPEN の **OPTS** パラメーターに指定された内容には関係なく、名前解決の検査が行われます。

オープンするオブジェクトがクラスター・キューの場合、ネーム・レゾリューションは MQOPEN 呼び出しの時点で行うことも、据え置くこともできます。ネーム・レゾリューションをいつ行うかは、MQOPEN 呼び出しで指定された OOBND* オプションで制御されます。

- OOBND0
- OOBNDN
- OOBNDQ

クラスター・キューのネーム・レゾリューションの詳細については、[ネーム・レゾリューション](#)を参照してください。

4. アプリケーションでオブジェクトがオープンされている間に、そのオブジェクトの属性が変わることもあります。多くの場合、アプリケーションでは属性の変化を通知しませんが、特定の属性についてキュー・マネージャーがハンドルに「無効」としてマーク付けます。次のとおりです。

- オブジェクトの名前の解決に影響するすべての属性。これは使用されるオープン・オプションに関係なく当てはまります。以下のものが含まれます。
 - 開いている別名キューの **BaseQName** 属性に対する変更。
 - **RemoteQName** または **RemoteQMgrName** キュー属性に対する変更。このキューに対してオープンされているハンドル、またはこの定義を介してキュー・マネージャー別名として解決されるキューに対する変更。
 - リモート・キュー用に現在開いているハンドルの解決先が別の伝送 キューになるような変更、あるいはまったく解決できなくなるような変更。例えば、次のものが含まれます。
 - リモート・キューのローカル定義の **XmitQName** 属性に対する変更 (その定義がキュー用に使用されるかキュー・マネージャーの別名用に使用されているかには無関係)。

この条件には例外が 1 つあります。その例外とは、新規伝送キューの作成です。ハンドルを開くとき、このキューが存在していればそれが解決先となっていたものの、実際にはそれがなかったためにデフォルトの伝送キューが解決先となった場合、このハンドルは無効にはなりません。

- **DefXmitQName** キュー・マネージャーに対する変更。この場合は、以前に指定されたキューに解決されたすべてのオープン・ハンドル (デフォルトの伝送キューであるというだけの理由でそれに解決されたオープン・ハンドル) に、無効のマークが付けられます。その他の理由でこのキューに解決されたハンドルは影響を受けません。
- **Shareability** キュー属性 (ローカル・キュー、またはローカル・キューに解決されるキューに対して、現在 OOINPS アクセスを提供しているハンドルが 2 つ以上ある場合)。この場合は、オープン・オプションとは関係なく、このキュー、またはこのキューに解決されるキューの、開かれているすべてのハンドルに無効のマークが付けられます。
- オープン・オプションとは関係なく、このキュー、またはこのキューに解決されるキューに対して開かれているすべてのハンドルの **Usage** キュー属性。

ハンドルに無効のマークが付けられると、このハンドルを使用したそのあとの呼び出し (MQCLOSE 以外) はすべて失敗し、理由コード RC2041 が戻ります。アプリケーションで、MQCLOSE 呼び出し (発信元ハンドルを使用) を発行してから、キューを再オープンしてください。以前に成功した呼び出しで使用した古いハンドルに対するコミットされていない更新は、この時点でもアプリケーション・ロジックの必要に応じてコミットまたはバックアウトが可能です。

属性を変更すると、これが発生する可能性がある場合は、特殊な "force" バージョンのコマンドを使用する必要があります。

5. キュー・マネージャーは、MQOPEN 呼び出しが発行されるときにセキュリティ検査を行い、アプリケーションの実行に使用されるユーザー ID に適切なレベルの権限があることを確認してからアクセ

スが許可されます。許可検査は、オープンされるオブジェクトの名前に対して行われ、名前が解決された後の結果の名前 (1 つ以上) に対しては行われません。

オープンされるオブジェクトがモデル・キューの場合、キュー・マネージャーは、モデル・キューの名前と作成された動的キューの名前の両方に対して、完全セキュリティ検査を行います。作成される動的キューが後で明示的に開かれると、さらにリソース・セキュリティ検査が動的キューの名前に対して実行されます。

6. この呼び出しの **OBJDSC** パラメーターには、リモート・キューを次の 2 つの方法のいずれかで指定できます (1166 ページの『**IBM i**での **MQOD** (オブジェクト記述子)』で説明している **ODON** フィールドおよび **ODMN** フィールドを参照してください)。

- **ODON** に、リモート・キューのローカル定義の名前を指定する方法。この場合、**ODMN** はローカル・キュー・マネージャーを参照するものであり、ブランクとして指定することができます。

ローカル・キュー・マネージャーで実行されるセキュリティ妥当性検査では、そのユーザーが、リモート・キューのローカル定義をオープンする許可を持っているかどうかを検査されます。

- **ODON** に、リモート・キュー・マネージャーに認識されているリモート・キューの名前を指定する方法。この場合、**ODMN** はリモート・キュー・マネージャーの名前です。

ローカル・キュー・マネージャーによって実行されるセキュリティ妥当性検査では、ユーザーが、名前解決プロセスからの結果の伝送キューにメッセージを送る許可を持っているかどうかを検査されます。

いずれの場合も、次のようになります。

- ユーザーがキューにメッセージを書き込む許可を持っているかどうかを検査するために、ローカル・キュー・マネージャーからリモート・キュー・マネージャーへメッセージが送られることはありません。
 - メッセージがリモート・キュー・マネージャーに届くとき、リモート・キュー・マネージャーは、メッセージを発信しているユーザーが許可を持っていないため、それを拒否することがあります。
7. **OOWB** オプションを用いた **MQOPEN** 呼び出しは、オブジェクト・ハンドルといずれか 1 つのブラウザ・オプションを指定する **MQGET** 呼び出しで使用するために、ブラウザ・カーソルを確立します。これにより、内容を変更せずにキューをスキャンすることができます。ブラウザによって検出されたメッセージは、後で **GMMUC** オプションを使用することにより、キューから除去することができます。

同一のキューに対していくつかの **MQOPEN** 要求を出すと、1 つのアプリケーションに対して複数のブラウザ・カーソルをアクティブにすることができます。

8. 次の注意事項は、配布リストの使用に適用されます。

- **MQOD** 構造内の各フィールドは、配布リストを開くときに、次のように設定しなければなりません。
 - **ODVER** は、**ODVER2** 以上にする。
 - **ODOT** は、**OTQ** にする。
 - **ODON** は、ブランクまたはヌル・ストリングにする。
 - **ODMN** は、ブランクまたはヌル・ストリングにする。
 - **ODREC** は、ゼロより大きな値にする。
 - **ODORO** と **ODORP** のうちの片方をゼロ、もう片方をゼロ以外にする。
 - **ODRRO** および **ODRRP** は、ゼロ以外にしない。
 - **ODORO** または **ODORP** のいずれかによりアドレス指定される **ODREC** オブジェクト・レコードが存在する。これらのオブジェクト・レコードには、開かれる宛先キューの名前を設定しなければなりません。
 - **ODRRO** および **ODRRP** のうちの片方がゼロ以外であるとき、**ODREC** 応答レコードが存在する。これらは、呼び出しが理由コード **RC2136** を出して終了する場合、キュー・マネージャーにより設定されます。

ODREC がゼロであることを確認すれば、バージョン 2 の **MQOD** でも、配布リストにない単一キューをオープンできます。

- **OPTS** パラメーターでは、次のオープン・オプションのみが有効です。
 - OOOOUT
 - OOPAS*
 - OOSSET*
 - OOALTU
 - OOFIQ
 - 配布リスト内の宛先キューとして、ローカル・キュー、別名キュー、またはリモート・キューを指定することは可能ですが、モデル・キューを指定することはできません。モデル・キューが指定されている場合は、オープンに失敗し、理由コード RC2057 が戻ります。ただし、このようになって、リスト内の他のキューは正常に開かれます。
 - 完了コード・パラメーターおよび理由コード・パラメーターは、次のように設定されます。
 - 配布リスト内のキューに対するオープン操作がすべて同じ結果になった (すべて成功または失敗した) 場合、完了コード・パラメーターおよび理由コード・パラメーターは、この共通の結果を示す値に設定されます。MQRR 応答レコード (アプリケーションにより提供されている場合) は、この場合には設定されません。
 例えば、すべてのオープンが成功すると、完了コードは CCOK に設定され、理由コードは RCNONE に設定されます。いずれのキューも存在しないためにすべてのオープンが失敗すると、それらのパラメーターは CCFAIL および RC2085 に設定されます。
 - 配布リスト内のキューに対するオープン操作が同じ結果にならなかった (すべて成功でもすべて失敗でもない) 場合には、次のようになります。
 - 完了コード・パラメーターは、少なくとも 1 つのオープンが成功した場合、CCWARN に設定され、すべて失敗した場合には CCFAIL に設定されます。
 - 理由コード・パラメーターは、RC2136 に設定されます。
 - 応答レコード (アプリケーションにより提供されている場合) は、配布リスト内のキューごとに、個別の完了コードおよび理由コードに設定されます。
 - 配布リストが正常にオープンされた場合、呼び出しにより返された *HOB*J ハンドルを後続の MQPUT 呼び出しで使用して、配布リスト内のキューにメッセージを書き込むことができます。さらにこのハンドルを MQCLOSE 呼び出しで使用して、配布リストへのアクセスを解放することもできます。配布リストに有効なクローズ・オプションは、CONONE のみです。
 配布リストにメッセージを書き込むために、MQPUT1 呼び出しを使用することもできます。このリスト内のキューを定義する MQOD 構造は、その呼び出しのパラメーターとして指定されます。
 - アプリケーションが最大許容ハンドル数を越えたかどうかを検査するとき、配布リスト内の正常にオープンされた宛先ごとに、別のハンドルとしてカウントされます (**MaxHandles** キュー・マネージャー属性を参照)。これは、配布リスト内の複数の宛先が実際に同一の物理キューに解決されるときにも当てはまります。配布リストについて発行された MQOPEN または MQPUT1 呼び出しによって、アプリケーションが使用しているハンドルの数が *MaxHandles* を超える場合、呼び出しは理由コード RC2017 を戻して失敗します。
 - 宛先が正常に開かれるたびに、**OpenOutputCount** 属性の値が 1 つずつ加算されます。配布リスト内の複数の宛先が実際に同一の物理キューに解決される場合、キューの **OpenOutputCount** 属性の値はそのキューに解決される配布リスト内の宛先の数だけ増加します。
 - 各キューを個々にオープンするとハンドルが無効になるようなキュー定義の変更 (例えば、解決パスの変更) があっても、配布リスト・ハンドルは無効にはなりません。しかし、後続の MQPUT 呼び出しで配布リスト・ハンドルが使用される際、その特定のキューについては失敗します。
 - 配布リストに宛先が 1 つしかない場合は有効です。
9. 以下の注意事項は、クラスター・キューの使用に適用されます。
- 初めてクラスター・キューが開かれたとき、ローカル・キュー・マネージャーがフル・リポジトリ・キュー・マネージャーでなければ、ローカル・キュー・マネージャーは、フル・リポジトリ・キュー・マネージャーからそのクラスター・キューに関する情報を取得します。ネットワークが使用中の場合は、ローカル・キュー・マネージャーがリポジトリ・キュー・マネージャーから必要な

情報を受信するまでに何秒か要する場合があります。その結果、MQOPEN 呼び出しを発行したアプリケーションは最大 10 秒間待機しなければならない場合があります、その後、MQOPEN 呼び出しから制御が戻ります。この時間内にローカル・キュー・マネージャーがクラスター・キューに関して必要な情報を受信できなかった場合、呼び出しは理由コード RC2189 を戻して失敗します。

- あるクラスター・キューが開いており、クラスター内にそのキューのインスタンスが複数存在する場合、実際に開かれるインスタンスは、MQOPEN 呼び出しで指定されたオプションによって決まります。

– 指定したオプションに、次のいずれかが含まれている場合:

- OOBROW
- OOINPQ
- OOINPX
- OOINPS
- OOSSET

オープンされるクラスター・キューのインスタンスは、ローカル・インスタンスでなければなりません。そのキューにローカル・インスタンスがない場合は、MQOPEN 呼び出しは失敗します。

– 指定したオプションに上記のどれも含まれておらず、次の 1 つまたは両方が含まれている場合

- OOINQ
- OOOOUT

オープンされるインスタンスは、ローカル・インスタンスがあればローカル・インスタンスになり、なければリモート・インスタンスになります。ただし、キュー・マネージャーによって選択されたインスタンスを、クラスター・ワークロード出口によって変更することもできます (そのような出口がある場合)。

クラスター・キューの詳細については、[クラスター・キュー](#)を参照してください。

10. トリガー・モニターにより開始されるアプリケーションには、そのアプリケーションに関連付けられているキューの名前が、開始時に渡されます。このキュー名を **OBJDSC** パラメーターに指定すると、そのキューをオープンできます。詳細については、MQTMC 構造体を参照してください。
11. OORLOQ オプションを使用する場合、ローカル、別名またはモデル・キューがオープンされていればローカル・キューは既に戻されていますが、例えば、リモート・キューまたは非ローカル・クラスター・キューがオープンされている場合には、そうではありません。ResolvedQName および ResolvedQMgrName には、リモート・キュー定義にある RemoteQName および RemoteQMgrName が設定されます。選択されたリモート・クラスター・キューの場合も同様です。例えば、リモート・キューのオープン時に OORLOQ が指定されている場合、ResolvedQName がメッセージが書き込まれる伝送キューになります。ResolvedQMgrName には、伝送キューをホストするローカル・キュー・マネージャーの名前が入ります。キューでの参照、入出力が許可されている場合、ユーザーはこのフラグを MQOPEN 呼び出しで指定するために必要な許可を持っています。特殊権限は必要ありません。

パラメーター

MQOPEN 呼び出しには、以下のパラメーターがあります。

HCONN (10 桁の符号付き整数) - 入力

接続ハンドル。

このハンドルは、キュー・マネージャーに対する接続を表します。HCONN の値は、先行の MQCONN または MQCONNXX 呼び出しによって戻されたものです。

OBJDSC (MQOD) - 入出力

オブジェクト記述子。

これは、開くオブジェクトを識別する構造です。詳細については、[1166 ページの『IBM i での MQOD \(オブジェクト記述子\)』](#)を参照してください。

OBJDSC パラメーターの *ODON* フィールドがモデル・キューの名前である場合は、動的ローカル・キュー これは、モデル・キューの属性を使用して作成されます。これは、**OPTS** パラメーターで指定されたオープン・オプションに関係なく行われます。MQOPEN 呼び出しによって返される *HOB*J を使用する後続の操作は、モデル・キューではなく新しい動的キューで行われます。MQINQ 呼び出しおよび MQSET 呼び出しの場合でも同じです。**OBJDSC** パラメーターのモデル・キューの名前は、作成された動的キューの名前と置き換わります。動的キューのタイプは、モデル・キューの **DefinitionType** 属性の値によって決まります (1380 ページの『キューの属性』を参照してください)。動的キューに適用されるクローズ・オプションの詳細については、MQCLOSE 呼び出しの記述を参照してください。

OPTS (10 桁の符号付き整数) - 入力

MQOPEN のアクションを制御するオプション。

以下のオプションを少なくとも 1 つ指定する必要があります。

- OOBWR
- OOINP* (このうちの 1 つのみ)
- OOINQ
- OOOOT
- OOSSET
- OORLQ

それ以外のオプションも必要に応じて指定できます。複数のオプションが必要な場合は、それらの値を追加します (同じ定数を複数回追加しないでください)。無効な組み合わせには、その旨を示しています。その他の組み合わせはすべて有効です。**OBJDSC** によって指定されたオブジェクトのタイプに適用されるオプションだけが許可されます (各キュー・タイプに有効な MQOPEN オプションを参照してください)。

アクセス・オプション: 以下のオプションは、オブジェクトに対して実行できる操作のタイプを制御します。

OOINPQ

キュー定義のデフォルトを使用してメッセージを取得するためにキューを開きます。

後続の MQGET 呼び出しで使用するために、キューが開かれます。アクセスのタイプは、**DefInputOpenOption** キュー属性の値に応じて、共用または排他的のいずれかになります。詳細については、1380 ページの『キューの属性』を参照してください。

このオプションは、ローカル・キュー、別名キュー、およびモデル・キューに関してのみ有効です。リモート・キューや配布リスト、さらにキューでないオブジェクトに関しては無効です。

OOINPS

共有アクセスによりメッセージを読み取るためにキューをオープンする。

後続の MQGET 呼び出しで使用するために、キューが開かれます。キューが、実行中のアプリケーションまたは他のアプリケーションによって OOIINPS を指定して現在オープンされている場合は、この呼び出しは成功しますが、キューが OOIINPX を指定して現在オープンされている場合は、理由コード RC2042 で失敗します。

このオプションは、ローカル・キュー、別名キュー、およびモデル・キューに関してのみ有効です。リモート・キューや配布リスト、さらにキューでないオブジェクトに関しては無効です。

OOINPX

メッセージを読み取るためにキューを排他アクセス・モードでオープンする。

後続の MQGET 呼び出しで使用するために、キューが開かれます。キューが、実行中のアプリケーションまたは他のアプリケーションによって、いずれかのタイプ (OOINPS または OOIINPX) の入力用に現在オープンされている場合、理由コード RC2042 で呼び出しは失敗します。

このオプションは、ローカル・キュー、別名キュー、およびモデル・キューに関してのみ有効です。リモート・キューや配布リスト、さらにキューでないオブジェクトに関しては無効です。

以下の注は、次のオプションに適用されます。

- これらのオプションは、1つだけ指定できます。
- これらのオプションのいずれかを指定した MQOPEN 呼び出しは、**InhibitGet** キュー属性が QAGETI に設定されていても成功します (ただし、その後の MQGET 呼び出しは、属性がこの値に設定されていると失敗します)。
- キューが共用可能でないものとして定義される場合 (つまり、**Shareability** キュー属性の値が QANSHR)、共有アクセスのためにキューをオープンすると、排他的アクセスでキューをオープンしようとしたものとして扱われます。
- 別名キューがこれらのオプションのいずれかで開かれている場合は、排他的使用をしているかどうか (または別のアプリケーションで排他的使用をしているか) のテストが、別名解決先の基本キューに対して行われます。
- ODMN がキュー・マネージャー別名の名前の場合は、これらのオプションは無効です。キュー・マネージャー別名に使用されたリモート・キューのローカル定義内の **RemoteQMGrName** 属性値がローカル・キュー・マネージャーの名前である場合でも同様です。

OOBRW

メッセージをブラウズするためにキューを開きます。

以下のいずれかのオプションを使用する後続の MQGET 呼び出しで使用するために、キューが開かれます。

- GMBRWF
- GMBRWN
- GMBRWC

これは、キューが現在 OOINPX でオープンされている場合でも可能です。OOBRW オプションを指定して MQOPEN 呼び出しを発行すると、ブラウズ・カーソルが設定され、論理的にはキューにある最初のメッセージの前にそのブラウズ・カーソルが置かれます。詳細については、[1080 ページの『IBM iでの MQGMO \(読み取りメッセージ・オプション\)』の GMOPT フィールドを参照してください。](#)

このオプションは、ローカル・キュー、別名キュー、およびモデル・キューについてのみ有効です。リモート・キュー、配布リスト、およびキュー以外のオブジェクトには無効です。また、ODMN がキュー・マネージャーの別名である場合、有効ではありません。キュー・マネージャーに別名を付けるために使用されるリモート・キューのローカル定義内の **RemoteQMGrName** 属性の値がローカル・キュー・マネージャーの名前であるときも同様です。

OOOUT

キューを開いてメッセージを書き込んだり、トピックまたはトピック・ストリングを開いてメッセージを公開したりします。

後続の MQPUT 呼び出しで使用するために、キューが開かれます。

このオプションを用いた MQOPEN 呼び出しは、**InhibitPut** キュー属性が QAPUTI に設定されていても成功します (ただし、属性がこの値に設定されていると、後続の MQPUT 呼び出しは失敗します)。

このオプションは、配布リストおよびトピックをはじめ、すべてのタイプのキューに有効です。

OOINQ

属性を照会するためにオブジェクトを開きます。

後続の MQINQ 呼び出しで使用するために、キュー、名前リスト、プロセス定義、またはキュー・マネージャーが開かれます。

このオプションは、配布リスト以外のすべてのタイプのオブジェクトで有効です。ODMN がキュー・マネージャーの別名である場合、有効ではありません。キュー・マネージャーに別名を付けるために使用されるリモート・キューのローカル定義内の **RemoteQMGrName** 属性の値がローカル・キュー・マネージャーの名前であるときも同様です。

OOSSET

属性を設定するためにキューを開きます。

後続の MQSET 呼び出しで使用するために、キューが開かれます。

このオプションは、配布リスト以外のすべてのタイプのキューで有効です。ODMN がリモート・キューのローカル定義の名前である場合は無効です。キュー・マネージャー別名に使用されたりリモート・キューのローカル定義内の **RemoteQMGrName** 属性値がローカル・キュー・マネージャーの名前である場合でも同様です。

バインディング・オプション: 以下のオプションは、開かれるオブジェクトがクラスター・キューである場合に適用されます。これらのオプションは、クラスター・キューのインスタンスへのキュー・ハンドルのバインディングを制御します。

OOBNDQ

キューがオープンされたときに、ハンドルを宛先にバインドします。

このオプションを指定すると、キューがオープンされたときに、ローカル・キュー・マネージャーが、キュー・ハンドルを宛先キューのインスタンスにバインドします。その結果、このハンドルを使って書き込まれるすべてのメッセージが、宛先キューの同じインスタンスに、同じ経路で送信されます。

このオプションは、キューの場合にのみ有効であり、クラスター・キューにのみ影響します。クラスター・キューではないキューに対して指定された場合、このオプションは無視されます。

OOBNDN

特定の宛先にバインドしません。

このオプションを指定すると、ローカル・キュー・マネージャーは、宛先キューのインスタンスへのキュー・ハンドルのバインドを停止します。その結果、同じハンドルを使用したその後の MQPUT 呼び出しでは、メッセージが宛先キューのさまざまなインスタンスに送信される場合や、同じインスタンスに送信はされるがさまざまな経路を経由する場合があります。また、このオプションを使用すると、選択されたインスタンスを、ネットワーク条件に従って、ローカル・キュー・マネージャー、リモート・キュー・マネージャー、またはメッセージ・チャンネル・エージェント (MCA) で後で変更することもできます。

注: トランザクションを完了するために一連のメッセージを交換する必要のあるクライアント・アプリケーションおよびサーバー・アプリケーションでは、OOBNDN (DefBind の値が BNDNOT の場合は OOBNDQ) を使用しないでください。これを使用すると、その後のメッセージがサーバー・アプリケーションの別のインスタンスに送信されることがあります。

クラスター・キューに OOBROW または OOINP* オプションのいずれかを指定すると、キュー・マネージャーはそのクラスター・キューのローカル・インスタンスを選択します。その結果、OOBNDN が指定されていても、キュー・ハンドルのバインディングは固定されます。

OOBNDN と OOINQ を組み合わせて指定した場合、そのハンドルを使用するその後の MQINQ 呼び出しで、クラスター・キューのさまざまなインスタンスが照会される可能性があります。ただし、通常はすべてのインスタンスが同じ属性値を持っています。

OOBNDN はキューにのみ有効であり、クラスター・キューにのみ影響を及ぼします。クラスター・キューではないキューに対して指定された場合、このオプションは無視されます。

OOBNDQ

キューのデフォルトのバインディングを使用します。

このオプションを指定すると、ローカル・キュー・マネージャーは、**DefBind** キュー属性で定義されたとおりにキュー・ハンドルをバインドします。この属性の値は、BNDOPN または BNDNOT のいずれかです。

OOBNDQ と OOBNDN が指定されていない場合は、OOBNDQ がデフォルトです。

OOBNDQ は、プログラム文書を支援するために定義されます。このオプションは、他の 2 つのバインド・オプションのいずれかと組み合わせて使用することを意図して用意されたオプションではありません。しかしその値はゼロであるため、そのように組み合わせて使用しても検出できません。

コンテキスト・オプション: 以下のオプションは、メッセージ・コンテキストの処理を制御します。

OOSAVA

メッセージが取り出されるときにコンテキストを保管します。

コンテキスト情報がこのキュー・ハンドルに関連付けられます。この情報は、このハンドルを使用して取り出されたメッセージのコンテキストから設定されます。メッセージ・コンテキストについて詳しくは、[メッセージ・コンテキストおよびコンテキスト情報の制御](#)を参照してください。

このコンテキスト情報は、後で MQPUT 呼び出しまたは MQPUT1 呼び出しを使用してキューに書き込まれるメッセージに渡すことができます。1180 ページの『[IBM i での MQPMO \(メッセージ書き込みオプション\)](#)』の PMPASI および PMPASA オプションの詳細を参照してください。

メッセージが正常に取り出されるまでは、キューに書き込まれるメッセージにコンテキストを渡すことはできません。

GMBRW* ブラウズ・オプションのいずれかを使用して取り出されるメッセージには、そのコンテキスト情報が保存されません (ただし、MSGDSC パラメーターのコンテキスト・フィールドは、ブラウズの後で設定されます)。

このオプションは、ローカル・キュー、別名キュー、およびモデル・キューについてのみ有効です。リモート・キュー、配布リスト、およびキュー以外のオブジェクトには無効です。OOINP* オプションのいずれか 1 つを指定する必要があります。

OOPASI

識別コンテキストを渡すことができます。

これによって、キューへのメッセージ書き込み時に **PMO** パラメーターに PMPASI オプションを指定できます。これによって、OOSAVA オプションを指定してオープンされた入力キューからメッセージに識別コンテキスト情報を指定します。メッセージ・コンテキストについての詳細は、[メッセージ・コンテキストおよびコンテキスト情報の制御](#)を参照してください。

OOOUT オプションは指定する必要があります。

このオプションは、配布リストをはじめ、すべてのタイプのキューで有効です。

OOPASA

すべてのコンテキストを渡すことができるようにします。

これによって、キューへのメッセージ書き込み時に **PMO** パラメーターに PMPASA オプションを指定できます。これによって、OOSAVA オプションを指定してオープンされた入力キューからメッセージに識別コンテキスト情報と起点コンテキスト情報を指定します。メッセージ・コンテキストについての詳細は、[メッセージ・コンテキストおよびコンテキスト情報の制御](#)を参照してください。

このオプションは、OOPASI を暗黙的に指定するため、指定する必要はありません。OOOUT オプションは指定する必要があります。

このオプションは、配布リストをはじめ、すべてのタイプのキューで有効です。

OOSSETI

識別コンテキストを設定することができるようにします。

このオプションを指定すると、メッセージをキューに書き込むときに PMSETI オプションを **PMO** パラメーターで指定できるようになります。この結果、MQPUT または MQPUT1 呼び出しで指定された **MSGDSC** パラメーターに含まれる識別コンテキスト情報がメッセージに与えられます。メッセージ・コンテキストについての詳細は、[メッセージ・コンテキストおよびコンテキスト情報の制御](#)を参照してください。

このオプションは、OOPASI を暗黙的に指定するため、指定する必要はありません。OOOUT オプションは指定する必要があります。

このオプションは、配布リストをはじめ、すべてのタイプのキューで有効です。

OOSSETA

すべてのコンテキストを設定できるようにします。

このオプションを指定すると、メッセージをキューに書き込むときに PMSETA オプションを **PMO** パラメーターで指定できるようになります。この結果、MQPUT または MQPUT1 呼び出しで指定され

た **MSGDSC** パラメーターに含まれる識別コンテキスト情報および発信元コンテキスト情報がメッセージに与えられます。メッセージ・コンテキストについての詳細は、[メッセージ・コンテキストおよびコンテキスト情報の制御](#)を参照してください。

このオプションでは、以下のオプションが暗黙指定されるため、これらをあらためて指定する必要はありません。

- OOPASI
- OOPASA
- OOSSETI

OOOUT オプションは指定する必要があります。

このオプションは、配布リストをはじめ、すべてのタイプのキューで有効です。

その他のオプション: 以下のオプションは、許可検査を制御するほか、キュー・マネージャーが静止しているときに発生するイベントを制御します。

OOALTU

指定されたユーザー ID を用いて妥当性検査を行います。

これは、**OBJDSC** パラメーターの **ODAU** フィールドに、この **MQOPEN** 呼び出しの妥当性検査に使用されるユーザー ID が含まれていることを示します。アプリケーションを実行しているユーザー ID に対し、指定されたアクセス・オプションでオブジェクトのオープン許可が与えられているかどうかには関係なく、この **ODAU** にその許可が与えられている場合にのみ、呼び出しは正常に行われます。ただし、これは、指定されたコンテキスト・オプションには適用されません。コンテキスト・オプションの場合は常に、そのアプリケーションの実行に使用されているユーザー ID に対して検査されます。

このオプションは、すべてのタイプのオブジェクトで有効です。

OOFIQ

キュー・マネージャーが静止している場合は、失敗します。

このオプションは、キュー・マネージャーが静止状態にあるときに、**MQOPEN** 呼び出しを強制的に失敗させます。

このオプションは、すべてのタイプのオブジェクトで有効です。

OORLQ

オープンされたローカル・キューの名前を入力します。

このオプションは、**MQOD** 構造体の **ResolvedQName** (使用可能な場合) に、オープンされたローカル・キューの名前を入力する必要があることを指定します。**ResolvedQMgrName** には同じように、ローカル・キューをホストするローカル・キュー・マネージャーの名前が入力されます。

表 789. 各キュー・タイプに有効な MQOPEN オプション						
オプション	別名 (1343 ページの『1』)	ローカルおよびモデル	リモート	非ローカル・クラスター	配布リスト	トピック
OOINPQ	✓	✓	-	-	-	-
OOINPS	✓	✓	-	-	-	-
OOINPX	✓	✓	-	-	-	-
OOBRW	✓	✓	-	-	-	-
OOOUT	✓	✓	✓	✓	✓	✓
OOINQ	✓	✓	1343 ページの『2』	✓	-	-

表 789. 各キュー・タイプに有効な MQOPEN オプション (続き)						
オプション	別名 (1343 ページの『1』)	ローカルおよびモデル	リモート	非ローカル・クラスター	配布リスト	トピック
OOSSET	✓	✓	1343 ページの『2』	-	-	-
OOBNDQ (1343 ページの『3』)	✓	✓	✓	✓	✓	-
OOBNDN (1343 ページの『3』)	✓	✓	✓	✓	✓	-
OOBNDQ (1343 ページの『3』)	✓	✓	✓	✓	✓	-
OOSAVA	✓	✓	-	-	-	-
OOPASI	✓	✓	✓	✓	✓	1343 ページの『5』
OOPASA	✓	✓	✓	✓	✓	1343 ページの『5』
OOSSETI	✓	✓	✓	✓	✓	1343 ページの『5』
OOSSETA	✓	✓	✓	✓	✓	1343 ページの『5』
OOALTU	✓	✓	✓	✓	✓	✓
OOFIQ	✓	✓	✓	✓	✓	✓
OORLQ	✓	✓	✓	✓	-	-

注:

1. 別名のオプションの妥当性は、その別名が解決されるキューのオプションの妥当性に依拠して決められます。
2. このオプションは、リモート・キューのローカル定義の場合にのみ有効です。
3. このオプションは、どのタイプのキューについても指定できますが、そのキューがクラスター・キューでない場合は無視されます。
4. この属性はトピックでは無視されます。
5. これらの属性はトピックで使用できますが、サブスクライバーに送信されるコンテキスト・フィールドではなく、保存メッセージに設定されたコンテキストにのみ影響します。

HOBJS (10 桁の符号付き整数) - 出力

オブジェクト・ハンドル

このハンドルは、オブジェクトに対し設定されているアクセスを表します。また、オブジェクトに対して操作される後続のメッセージ・キューイング呼び出しで指定する必要があります。MQCLOSE 呼び出しが発行されたとき、またはハンドルの有効範囲を定義する処理の単位が終了したときに、有効でなくなります。

ハンドルの有効範囲は、以下の最小単位に制限されます。アプリケーションが実行されているプラットフォームによってサポートされる並列処理。このハンドルは、MQOPEN 呼び出しが発行された並列処理の単位外では無効です。

- IBM i の場合、ハンドルの有効範囲は、呼び出しを発行するジョブです。

CMPCOD (10 桁の符号付き整数) - 出力

完了コード。

これは、以下のいずれかになります。

CCOK

正常終了。

CCWARN

警告 (部分完了)。

CCFAIL

呼び出し失敗。

RPG 宣言

```
C*..1.....2.....3.....4.....5.....6.....7..  
C                                CALLP      MQOPEN(HCONN : OBJDSC : OPTS :  
C                                HOBJ : CMPCOD : REASON)
```

呼び出しのプロトタイプ定義は次のようになります。

```
D*..1.....2.....3.....4.....5.....6.....7..  
DMQOPEN          PR              EXTPROC('MQOPEN')  
D* Connection handle  
D HCONN          10I 0 VALUE  
D* Object descriptor  
D OBJDSC          468A  
D* Options that control the action of MQOPEN  
D OPTS           10I 0 VALUE  
D* Object handle  
D HOBJ           10I 0  
D* Completion code  
D CMPCOD          10I 0  
D* Reason code qualifying CMPCOD  
D REASON          10I 0
```

IBM i

IBM i での MQPUT (メッセージの書き込み)

MQPUT 呼び出しは、キュー、配布リスト、またはトピックにメッセージを書き込みます。キュー、配布リスト、またはトピックは、既にオープンされていなければなりません。

- [1344 ページの『構文』](#)
- [1344 ページの『使用上の注意』](#)
 - [1345 ページの『トピック』](#)
 - [1345 ページの『MQPUT および MQPUT1』](#)
 - [1345 ページの『宛先 キュー』](#)
 - [1346 ページの『配布リスト』](#)
 - [1348 ページの『ヘッダー』](#)
 - [1348 ページの『Buffer』](#)
- [1348 ページの『パラメーター』](#)
- [1354 ページの『RPG 宣言』](#)

構文

MQPUT (HCONN, HOBJ, MSGDSC, PMO, BUFLN, BUFFER, CMPCOD, REASON)

使用上の注意

トピック

以下の注意事項は、トピックの使用に適用されます。

1. MQPUT を使用してトピックでメッセージをパブリッシュする場合、サブスクライバー・キューで問題が発生した (例えば、キューが満杯である) ために、そのトピックの 1 つ以上のサブスクライバーにパブリケーションを提供できない場合、MQPUT 呼び出しに戻される理由コードおよび配信時の振る舞いは、TOPIC での PMSGDLV または NPMSGDLV 属性の設定によって異なります。RODLQ が指定されている場合にパブリケーションが送達不能キューに送達されたり、RODISC が指定されている場合にメッセージが廃棄されたりしても、メッセージは正常に送達されたと見なされることに注意してください。パブリケーションが何も送達されなかった場合、MQPUT は RC2502 で戻ります。これは次の場合に起こります。

- PMSGDLV または NPMSGDLV (メッセージの持続性によって異なる) が ALL に設定されている TOPIC にメッセージがパブリッシュされ、いずれかのサブスクリプション (永続的かどうかにかかわらず) にパブリケーションを受け取ることができないキューがある。
- メッセージが PMSGDLV または NPMSGDLV (メッセージのパーススタンスによって異なる) が ALLDUR に設定されている TOPIC にパブリッシュされており、永続サブスクリプションにパブリケーションを受信できないキューが含まれている。

パブリケーションを一部のサブスクライバーに送達できなかった場合でも、以下の場合には、MQPUT は RCNONE で戻ることがあります。

- PMSGDLV または NPMSGDLV (メッセージの持続性によって異なる) が ALLAVAIL に設定されている TOPIC にメッセージがパブリッシュされ、いずれかのサブスクリプション (永続的かどうかにかかわらず) にパブリケーションを受け取ることができないキューがある。
 - PMSGDLV または NPMSGDLV (メッセージの持続性によって異なる) が ALLDUR に設定されている TOPIC にメッセージがパブリッシュされ、非永続サブスクリプションにパブリケーションを受け取ることができないキューがある。
2. 使用されているトピックに対するサブスクライバーが存在しない場合、パブリッシュされるメッセージはどのキューにも送信されずに廃棄されます。このメッセージが永続的か非永続的か、またはメッセージの満了時間が無制限か短時間かは関係ありません。サブスクライバーが存在しない場合、メッセージはいずれにしても廃棄されます。このことの例外となるのは、メッセージが保存される場合です。この場合、メッセージはどのサブスクライバーのキューにも送信されませんが、メッセージはトピックに対して保管され、新規サブスクリプションに対して、または MQSUBRQ を使用して保存パブリケーションを要求するサブスクライバーに対して送達されます。

MQPUT および MQPUT1

MQPUT および MQPUT1 呼び出しを使用して、メッセージをキューに書き込むことができます。どの呼び出しが使用されるかは状況に応じて異なります。

- 複数のメッセージを同じ キューに書き込むときは、MQPUT 呼び出しを使用してください。
MQOPEN オプションを指定する MQOPEN 呼び出しが最初に発行され、その後に 1 つまたは複数の MQPUT 要求が続き、キューにメッセージを追加します。最後に、キューは MQCLOSE 呼び出しでクローズされます。この結果、MQPUT1 呼び出しを繰り返して使用するよりもパフォーマンスが向上します。
- メッセージを 1 つのみキューに書き込むときは、MQPUT1 呼び出しを使用してください。

この呼び出しは、MQOPEN、MQPUT、および MQCLOSE 呼び出しをまとめて単一の呼び出しにカプセル化するので、発行する必要がある呼び出しの数は最小になります。

宛先 キュー

アプリケーションがメッセージ・グループを使用せずにメッセージ・シーケンスを同じキューに書き込んだ場合、以下の条件が満たされていれば、それらのメッセージの順序は保持されます。ローカル宛先キューとリモート宛先キューの両方に適用される条件と、リモート宛先キューだけに適用される条件とがあります。

ローカル宛先キューおよびリモート宛先キューの条件

- すべての MQPUT 呼び出しが同一作業単位内に含まれている、または作業単位内にまったく含まれていません。

メッセージが 1 つの作業単位内の特定のキューに書き込まれると、他のアプリケーションからのメッセージに、そのキューのメッセージ・シーケンスが散在することがあります。

- すべての MQPUT 呼び出しは、同じオブジェクト・ハンドル *HOB* を使用して実行されます。

環境によっては、異なるオブジェクト・ハンドルを使用しても、メッセージ・シーケンスが保存されることがあります。ただし、呼び出しが同じアプリケーションから実行される場合に限りです。「同じアプリケーション」の意味は、環境によって異なります。

– IBM i の場合、アプリケーションはジョブ。

- どのメッセージも同じ優先順位をもっている。

リモート宛先キューの追加条件

- 送信側のキュー・マネージャーから宛先キュー・マネージャーへのパスが 1 つしかない。

シーケンス内の一部のメッセージが別のパスを使用する可能性がある場合 (例えば、再構成のため、または通信量のバランスのため、あるいはメッセージ・サイズに基づくパス選択のために) は、宛先キュー・マネージャーでのメッセージの順番は保証できません。

- 送信側、中間、または宛先キュー・マネージャーで、メッセージが一時的に送達不能キューに置かれません。

1 つ以上のメッセージが一時的に送達不能キューに置かれる場合 (例えば、伝送キューまたは宛先キューが一時的に満杯であるために)、メッセージが宛先キューに順序どおりに到達しない可能性があります。

- メッセージがすべて持続メッセージか、あるいはすべて非持続メッセージかのいずれかである。

送信側のキュー・マネージャーと宛先キュー・マネージャーの間の経路のチャネルの **CDNPM** 属性が **NPF**AST に設定されている場合、非持続メッセージが持続メッセージに先行してしまう可能性があり、その結果、非持続メッセージに対する持続メッセージの相対的順序が保持されない場合があります。ただし、持続メッセージ同士および非持続メッセージ同士の相対的順序は保持されます。

これらの条件が満たされない場合、メッセージ・グループを使用して、メッセージの順序を保持することができます。ただし、これには、送信側のアプリケーションと受信側のアプリケーションの両方がメッセージ・グループ化サポートを使用している必要があることに注意してください。メッセージ・グループの詳細については、以下を参照してください。

- MQMD の *MDMFL* フィールド
- MQPMO の *PMLOGO* オプション
- MQGMO の *GMLOGO* オプション

配布リスト

次の注意事項は、配布リストの使用に適用されます。

1. バージョン 1 の MQPMO のまたはバージョン 2 の MQPMO を使用して、各メッセージを配布リストに書き込むことができます。バージョン 1 の MQPMO が使用される (または *PMREC* を持つバージョン 2 の MQPMO がゼロに等しい) 場合には、書き込みメッセージ・レコードも応答レコードもアプリケーションにより提供されません。つまり、メッセージが配布リスト内のいくつかのキューに正常に送信され、それ以外のキューには正常に送信されない場合、エラーが発生したキューを識別することは不可能になります。

書き込みメッセージ・レコードまたは応答レコードがアプリケーションにより提供される場合には、*PMVER* フィールドを、*PMVER2* に設定する必要があります。

バージョン 2 の MQPMO でも、*PMREC* をゼロにすると、配布リストにない単一キューにメッセージを送信することができます。

2. 完了コード・パラメーターおよび理由コード・パラメーターは、次のように設定されます。

- 配布リスト内のキューへの書き込みがすべて同様に成功または失敗すると、完了コードおよび理由コード・パラメーターがその共通の結果を説明するよう設定されます。MQRR 応答レコード (アプリケーションにより提供されている場合) は、この場合には設定されません。

例えば、すべての書き込みが成功すると、完了コードは CCOK に設定され、理由コードは RCNONE に設定されます。すべてのキューが書き込み用に使用禁止になっているために書き込みに失敗した場合は、各パラメーターは CCFAIL および RC2051 に設定されます。

- 配布リスト内のキューに対する書き込みが一部成功した場合または失敗したがその理由が異なる場合は、次のように設定されます。
 - 少なくとも 1 つの書き込みが成功した場合、完了コード・パラメーターは CCWARN に、そしてすべてが失敗した場合には、CCFAIL に設定されます。
 - 理由コード・パラメーターは、RC2136 に設定されます。
 - 応答レコード (アプリケーションにより提供されている場合) は、配布リスト内のキューごとに、個別の完了コードおよび理由コードに設定されます。

宛先への書き込みが、その宛先のオープンが失敗したために、失敗した場合、応答レコード内の各フィールドは、CCFAIL および RC2137 に設定されます。その宛先は、PMIDC に組み込まれます。

3. 配布リストに指定した宛先のいずれかの解決先がローカル・キューである場合、メッセージは通常形式で (つまり、配布リスト・メッセージとしてではなく) そのキューに登録されます。複数の宛先の解決先が同じローカル・キューである場合は、このローカル・キューには同じメッセージが宛先数分登録されます。

配布リストに指定した宛先がリモート・キューに解決された場合、メッセージは、適切な伝送キュー上に登録されます。いくつかの宛先の解決先が同じ伝送キューである場合、この伝送キューには、これらの宛先が設定された配布リスト・メッセージが 1 つ登録されることがあります。これは、宛先どうしがアプリケーション提供の宛先リスト内で隣り合っているかどうかとは関係ありません。ただし、この処理が行われるのは、伝送キューで配布リスト・メッセージがサポートされている場合のみです ([1380 ページの『キューの属性』](#)で説明している **DistLists** キュー属性を参照してください)。

伝送キューが配布リストをサポートしていない場合、通常形式のメッセージのコピーが、その伝送キューを使用する各宛先の伝送キュー上に配置されます。

アプリケーション・メッセージ・データを持つ配布リストが伝送キューに対して大きすぎる場合、配布リスト・メッセージは、包含する宛先数の少ない小さな配布リスト・メッセージに分割されます。アプリケーション・メッセージ・データのみがキューに保管される場合、配布リスト・メッセージはまったく使用できず、キュー・マネージャーは、その伝送キューを使用する各宛先用にそのメッセージのコピーを通常形式で生成します。

それぞれの宛先が異なるメッセージ優先順位またはメッセージ持続性を持つ場合 (アプリケーションで PRQDEF または PEQDEF を指定すると、このようなことが起こります) は、メッセージが同じ配布リスト・メッセージ内に保持されません。そのため、キュー・マネージャーは、異なる優先順位および持続性値を収容するのに必要な数の配布リスト・メッセージを生成します。

4. 配布リストへメッセージを書き込むと、メッセージは次のいずれかになる場合があります。

- 1 つの配布リスト・メッセージ
- いくつかの小さな配布リスト・メッセージ
- 配布リスト・メッセージと通常メッセージが混在するメッセージ
- 通常メッセージだけ

上記のいずれになるかは、次の内容により異なります。

- リスト内の各宛先がローカル、リモート、またはローカルおよびリモートのいずれであるか。
- 各宛先が同じメッセージ優先順位およびメッセージ持続性を有するかどうか。
- 伝送キューが配布リスト・メッセージを保持できるかどうか。
- 伝送キューの最大メッセージ長が、配布リスト形式でそのメッセージを保管できる長さであるかどうか。

ただし、上記のいずれの場合でも、結果として発生するそれぞれの物理メッセージ(つまり、その書き込みから発生する標準メッセージまたは配布リスト・メッセージ)は、次の場合において、1つのメッセージとしてカウントされます。

- アプリケーションが作業単位内の最大許容メッセージ数を超過したかどうかを検査するとき (**MaxUncommittedMsgs** キュー・マネージャー属性を参照)。
 - トリガー発行条件が満たされているかどうかを検査するとき。
 - キューのサイズを増加させ、各キューの最大サイズが超過するかどうかを検査するとき。
5. 各キューを個々にオープンするとハンドルが無効になるようなキュー定義の変更 (例えば、解決パスの変更)があっても、配布リスト・ハンドルは無効にはなりません。しかし、後続の MQPUT 呼び出しで配布リスト・ハンドルが使用される際、その特定のキューについては失敗します。

ヘッダー

アプリケーション・メッセージ・データの先頭にある1つ以上の IBM MQ ヘッダー構造体を使用してメッセージが書き込まれる場合、キュー・マネージャーはそのヘッダー構造体に対して一定の検査を実行し、それらが有効であるか検証します。キュー・マネージャーがエラーを検出すると、呼び出しは失敗し、該当する理由コードが戻ります。実行される検査は、存在する特定の構造体によって異なります。さらに、バージョン2以降の MQMD が MQPUT または MQPUT1 呼び出し内で使用される場合のみ、検査が実行されます。MQMDE がアプリケーション・メッセージ・データの開始時点で存在していても、バージョン1の MQMD が使用されている場合には、これらの検査は実行されません。

次の IBM MQ ヘッダー構造体 MQDH および MQMDE の妥当性は、キュー・マネージャーによって完全に検証されます。

他の IBM MQ ヘッダー構造体の場合、キュー・マネージャーは、一定の妥当性検査を実行しますが、すべてのフィールドを検査する訳ではありません。ローカル・キュー・マネージャーによってサポートされない構造体、およびメッセージ内の最初の MQDLH の後の構造体には、妥当性検査は行われません。

IBM MQ 構造体内の各フィールドに関する一般検査の他にも、次の条件が満たされている必要があります。

- IBM MQ 構造体は、2つ以上のセグメントに分割してはなりません。この構造体は1つのセグメント内にその全体を含める必要があります。
- PCF メッセージ内の各構造体の長さの合計が、MQPUT または MQPUT1 呼び出し上の **BUFLN** パラメーターで指定される長さと等しくなければなりません。PCF メッセージとは、次のいずれかの形式名を持つメッセージです。
 - FMADMN
 - FMEVNT
 - FMPCF
- IBM MQ 構造体は、切り捨て構造体が許可される次の状況を除いて、切り捨ててはなりません。
 - レポート・メッセージであるメッセージ
 - PCF メッセージ。
 - MQDLH 構造体を含む各メッセージ (最初の MQDLH の後ろの構造体は、切り捨て可能です。この MQDLH の前にある構造体は、切り捨てできません。)

Buffer

RPG プログラミング例に示す **BUFFER** パラメーターは、ストリングとして宣言されています。このため、パラメーターの最大長は 256 バイトに制限されます。これより大きいバッファが必要な場合は、このパラメーターをストリングではなく構造体として宣言するか、あるいは物理ファイルのフィールドとして宣言する必要があります。そうすると、指定できる最大長が約 32 KB まで増加します。

パラメーター

MQPUT 呼び出しには、以下のパラメーターがあります。

HCONN (10 桁の符号付き整数) - 入力

接続ハンドル。

このハンドルは、キュー・マネージャーに対する接続を表します。HCONN の値は、先行の MQCONN または MQCONNX 呼び出しによって戻されたものです。

HOBJ (10 桁の符号付き整数) - 入力

オブジェクト・ハンドル

このハンドルは、メッセージが追加されるキュー、またはメッセージがパブリッシュされるトピックを表します。HOBJ の値は、OOOUT オプションを指定した、前の MQOPEN 呼び出しから戻されたものです。

MSGDSC (MQMD) - 入出力

メッセージ記述子。

この構造体は、送られるメッセージの属性を記述するものであり、書き込み要求が完了した後でメッセージに関する情報を受け取ります。詳細は [1114 ページの『IBM i での MQMD \(メッセージ記述子\)』](#) を参照してください。

アプリケーションがバージョン 1 の MQMD を提供している場合、メッセージ・データの接頭部に MQMDE 構造体を付けると、バージョン 2 の MQMD に存在し、バージョン 1 には存在しない各フィールドの値を指定できます。MQMD 内の MDFMT フィールドは、MQMDE が存在することを示すため、FMMDE に設定しておく必要があります。詳細については、[1159 ページの『IBM i での MQMDE \(拡張メッセージ記述子\)』](#) を参照してください。

PMO (MQPMO) - 入出力

MQPUT のアクションを制御するオプション。

詳細は [1180 ページの『IBM i での MQPMO \(メッセージ書き込みオプション\)』](#) を参照してください。

BUFLEN (10 桁の符号付き整数) - 入力

BUFFER 中のメッセージの長さ。

ゼロは有効であり、メッセージにアプリケーション・データが含まれていないことを示します。BUFLEN の上限はさまざまな要因によって決まります。

- 宛先キューが共有キューの場合は、上限は 63 KB (64 512 バイト) です。
- 宛先がローカル・キューの場合やローカル・キューとして解決される場合 (ただし共有キューでない場合)、上限は、以下の条件を満たすかどうかによって異なります。
 - ローカル・キュー・マネージャーがセグメント化をサポートしている。
 - 送信側アプリケーションが、キュー・マネージャーでメッセージのセグメント化を可能にするフラグを指定している。このフラグは MFSEGA であり、バージョン 2 の MQMD 内で指定できるほか、バージョン 1 の MQMD を使用する場合は MQMDE 内で指定できます。

この 2 つの条件が両方とも満たされている場合は、BUFLEN は 999 999 999 から MQMD 内の MDOFF フィールドの値を引いた値を超えることはできません。したがって、書き込むことのできる最長の論理メッセージは、999 999 999 バイト (MDOFF がゼロの場合) になります。ただし、オペレーティング・システムによるリソースの制約、またはアプリケーションが実行されている環境によるリソースの制約によって下限が課される場合があります。

上記の条件のいずれか一方または両方が満たされない場合、BUFLEN は、キューの MaxMsgLength 属性とキュー・マネージャーの MaxMsgLength 属性のうち小さい方の値以下でなければなりません。

- 宛先がリモート・キューの場合や宛先の解決先がリモート・キューである場合も、ローカル・キューの条件が適用されます。ただし、メッセージが宛先キューに到達するまでに通過するすべてのキュー・マネージャーが適用の対象となります。特に、次のキューに注意してください。
 - ローカル・キュー・マネージャーで一時的にメッセージを保管するために使用されるローカル伝送キュー。

2. ローカルのキュー・マネージャーと宛先のキュー・マネージャーとの間の経路にあるキュー・マネージャーで、メッセージを保管するために使用される中間伝送キュー (それがあある場合)。
3. 宛先キュー・マネージャーでの宛先キュー。

したがって、書き込み可能なメッセージの最大長は、これらのキューやキュー・マネージャーのうち、もっとも制限の厳しいものによって決まります。

メッセージが伝送キューに入れられる場合は、メッセージ・データと共に追加情報があるため、転送できるアプリケーション・データの量は小さくなります。この状況では、*BUFLen* の制限を決定する際に、伝送キューの *MaxMsgLength* 値から LNMHD バイト分を減算してください。

注: メッセージの書き込み時に同時に診断できるのは、1 の条件を満たさない障害 (理由コード RC2030 または RC2031) のみです。条件 2 または 3 が満たされない場合、メッセージは、中間キュー・マネージャーまたは宛先キュー・マネージャーのいずれかの箇所で送達不能 (未配布メッセージ) キューにリダイレクトされます。これが発生した場合、送信側からの要求があれば、レポート・メッセージが生成されます。

BUFFER (1 バイトのビット・ストリング x BUFLen) - 入力

メッセージ・データ。

これは、送信するアプリケーション・データが入っているバッファです。バッファは、メッセージ内のデータの性質に適した境界に合わせなければなりません。ほとんどのメッセージ (MQ ヘッダー構造体を含むメッセージを含む) に対しては 4 バイト境界への位置合わせが適していますが、一部のメッセージにはさらに厳しい位置合わせ条件が課されます。例えば、64 ビット・バイナリー整数を含むメッセージは 8 バイト境界に合わせる必要がある場合があります。

BUFFER に文字データ、数値データ、またはその両方が含まれている場合、**MSGDSC** パラメーターの *MDCSI* および *MDENC* フィールドは、データに適切な値に設定する必要があります。これにより、メッセージの受信側は、(必要に応じて) データを受信側が使用する文字セットおよびエンコードに変換できるようになります。

注: MQPUT 呼び出しにある他のパラメーターはすべて、**CodedCharSetId** キュー・マネージャー属性で指定した文字セットと ENNAT で指定したローカル・キュー・マネージャーのエンコードで記述されていなければなりません。

CMPCOD (10 桁の符号付き整数) - 出力

完了コード。

これは、以下のいずれかになります。

CCOK

正常終了。

CCWARN

警告 (部分完了)。

CCFAIL

呼び出し失敗。

REASON (10 桁の符号付き整数) - 出力

CMPCOD を限定する理由コード。

CMPCOD が CCOK の場合

RCNONE

(0, X'000') レポートする理由コードはありません。

CMPCOD が CCWARN の場合

RC2104

(2104, X'838') メッセージ記述子のレポート・オプションが認識されない。

RC2136

(2136, X'858') 複数の理由コードが返されました。

CMPCOD が CCFAIL の場合

RC2004

(2004, X'7D4') バッファー・パラメーターが無効である。

RC2005

(2005, X'7D5') バッファー長パラメーターは無効です。

RC2009

(2009, X'7D9') キュー・マネージャーとの接続が失われました。

RC2013

(2013, X'7DD') 満了時刻が無効である。

RC2014

(2014, X'7DE') フィードバック・コードが無効である。

RC2018

(2018, X'7E2') 接続ハンドルが無効です。

RC2019

(2019, X'7E3') オブジェクト・ハンドルが無効です。

RC2024

(2024, X'7E8') 現行の作業単位内では、これ以上メッセージを処理できない。

RC2026

(2026, X'7EA') メッセージ記述子が無効である。

RC2027

(2027, X'7EB') 応答先キューがない。

RC2029

(2029, X'7ED') メッセージ記述子のメッセージ・タイプが無効である。

RC2030

(2030, X'7EE') メッセージの長さが、キューの最大許容数より大きいです。

RC2031

(2031, X'7EF') メッセージ長がキュー・マネージャーの最大許容長より大きいです。

RC2039

(2039, X'7F7') キューが出力用にオープンされていない。

RC2041

(2041, X'7F9') オープンされた後でオブジェクト定義が変更された。

RC2046

(2046, X'7FE') オプションが無効であるか、矛盾しています。

RC2047

(2047, X'7FF') 持続性が無効である。

RC2048

(2048, X'800') キューは永続的なメッセージをサポートしていません。

RC2050

(2050, X'802') メッセージ優先順位が無効である。

RC2051

(2051, X'803') このキューでは書き込み呼び出しが使用禁止になっています。

RC2052

(2052, X'804') キューが削除されました。

RC2053

(2053, X'805') キューには既に最大数のメッセージが入っています。

RC2056

(2056, X'808') ディスク上にキューのためのスペースがありません。

RC2058

(2058, X'80A') キュー・マネージャー名が無効であるか、認識されていません。

RC2059

(2059, X'80B') キュー・マネージャーを接続に使用できません。

RC2061

(2061, X'80D') メッセージ記述子のレポート・オプションが無効である。

RC2071

(2071, X'817') ストレージが不足しています。

RC2072

(2072, X'818') 同期点サポートが利用できない。

RC2093

(2093, X'82D') キューが全コンテキスト・パスとしてオープンされていない。

RC2094

(2094, X'82E') キューが識別コンテキスト・パスとしてオープンされていない。

RC2095

(2095, X'82F') キューが全コンテキスト設定用にオープンされていない。

RC2096

(2096, X'830') キューが識別コンテキスト設定用にオープンされていない。

RC2097

(2097, X'831') 参照されたキュー・ハンドルがコンテキストを保存しない。

RC2098

(2098, X'832') 参照されたキュー・ハンドルでコンテキストが使用できない。

RC2101

(2101, X'835') オブジェクトが損傷しました。

RC2102

(2102, X'836') 使用できるシステム・リソースが不足しています。

RC2135

(2135, X'857') 配布ヘッダー構造体が無効である。

RC2136

(2136, X'858') 複数の理由コードが返されました。

RC2137

(2137, X'859') オブジェクトが正常にオープンされていません。

RC2149

(2149, X'865') PCF 構造体が無効である。

RC2154

(2154, X'86A') 存在するレコード数が無効です。

RC2156

(2156, X'86C') 応答レコードが無効です。

RC2158

(2158, X'86E') 書き込みメッセージ・レコード・フラグが無効である。

RC2159

(2159, X'86F') 書き込みメッセージ・レコードが無効である。

RC2161

(2161, X'871') キュー・マネージャーが静止しています。

RC2162

(2162, X'872') キュー・マネージャーのシャットダウン中です。

RC2173

(2173, X'87D') 書き込みメッセージ・オプションの構造体が無効である。

RC2185

(2185, X'889') 持続性の指定が不整合である。

RC2188

(2188, X'88C') クラスター・ワークロード出口によって呼び出しが拒否されました。

RC2189

(2189, X'88D') クラスター名の解決に失敗しました。

RC2195

(2195, X'893') 予期しないエラーが発生しました。

RC2219

(2219, X'8AB') 前の呼び出しが完了する前に MQI 呼び出しが再入力されました。

RC2241

(2241, X'8C1') メッセージ・グループが不完全である。

RC2242

(2242, X'8C2') 論理メッセージが不完全である。

RC2245

(2245, X'8C5') 作業単位の指定が不整合である。

RC2248

(2248, X'8C8') メッセージ記述子の拡張子が無効である。

RC2249

(2249, X'8C9') メッセージ・フラグが無効である。

RC2250

(2250, X'8CA') メッセージ順序番号が無効である。

RC2251

(2251, X'8CB') メッセージ・セグメント・オフセットが無効である。

RC2252

(2252, X'8CC') 元の長さが無効である。

RC2253

(2253, X'8CD') メッセージ・セグメント内のデータの長さがゼロである。

RC2255

(2255, X'8CF') 作業単位がキュー・マネージャーから使用不可。

RC2257

(2257, X'8D1') 提供された MQMD のバージョンが違っている。

RC2258

(2258, X'8D2') グループ ID が無効である。

RC2266

(2266, X'8DA') クラスター・ワークロード出口で障害が発生しました。

RC2269

(2269, X'8DD') クラスター・リソース・エラー。

RC2270

(2270, X'8DE') 使用可能な宛先キューがない。

RC2420

(2420) MQPUT 呼び出しが発行されましたが、メッセージ・データに無効な MQEPH 構造体が含まれます。

RC2479

(2479, X'9AF') パブリケーションを保存できませんでした。

RC2480

(2480, X'9B0') ターゲット・タイプが変更されました。別名キューはキューを参照していましたが、現在ではトピックを参照しています。

RC2502

(2502, X'9C6') パブリケーションが失敗しました。パブリケーションはどのサブスクライバーにも送達されていません。

RC2551

(2551, X'9F7') 指定された選択ストリングを使用できません。

RC2554

(2554, X'9FA') 拡張メッセージ・セレクターを使ってサブスクライバーにメッセージを送達すべきかどうか決定するために、メッセージの内容を解析することができませんでした。

RPG 宣言

```
C*..1.....2.....3.....4.....5.....6.....7..  
C                                CALLP      MQPUT(HCONN : HOBJ : MSGDSC : PMO :  
C                                BUFLN : BUFFER : CMPCOD :  
C                                REASON)
```

呼び出しのプロトタイプ定義は次のようになります。

```
D*..1.....2.....3.....4.....5.....6.....7..  
DMQPUT      PR                      EXTPROC('MQPUT')  
D* Connection handle  
D HCONN                      10I 0 VALUE  
D* Object handle  
D HOBJ                      10I 0 VALUE  
D* Message descriptor  
D MSGDSC                      364A  
D* Options that control the action of MQPUT  
D PMO                        200A  
D* Length of the message in Buffer  
D BUFLN                      10I 0 VALUE  
D* Message data  
D BUFFER                      *    VALUE  
D* Completion code  
D CMPCOD                      10I 0  
D* Reason code qualifying CMPCOD  
D REASON                      10I 0
```

IBM i IBM i での MQPUT1 (1つのメッセージの書き込み)

MQPUT1 呼び出しは、キューまたは配布リスト上に、あるいはトピックに1つのメッセージを書き込みます。キュー、配布リストまたはトピックは、オープンされていなくてもかまいません。

- [1354 ページの『構文』](#)
- [1354 ページの『使用上の注意』](#)
- [1355 ページの『パラメーター』](#)
- [1360 ページの『RPG 宣言』](#)

構文

MQPUT1 (HCONN, OBJDSC, MSGDSC, PMO, BUFLN, BUFFER, CMPCOD, REASON)

使用上の注意

1. MQPUT および MQPUT1 呼び出しを使用して、メッセージをキューに書き込むことができます。どの呼び出しが使用されるかは状況に応じて異なります。

- 複数のメッセージを同じキューに書き込むときは、MQPUT 呼び出しを使用してください。

MQOPEN オプションを指定する MQOPEN 呼び出しが最初に発行され、その後に1つまたは複数の MQPUT 要求が続き、キューにメッセージを追加します。最後に、キューは MQCLOSE 呼び出しでクローズされます。この結果、MQPUT1 呼び出しを繰り返して使用するよりもパフォーマンスが向上します。

- メッセージを1つのみキューに書き込むときは、MQPUT1 呼び出しを使用してください。

この呼び出しは、MQOPEN、MQPUT、および MQCLOSE 呼び出しをまとめて単一の呼び出しにカプセル化するので、発行する必要がある呼び出しの数は最小になります。

2. アプリケーションがメッセージ・グループを使用せずにメッセージ・シーケンスを同じキューに書き込んだ場合、特定の条件が満たされていれば、それらのメッセージの順序は保持されます。ただし、ほとんどの環境では MQPUT1 呼び出しはこれらの条件を満たしていないため、メッセージの順序は保たれません。これらの環境では、MQPUT 呼び出しを使用する必要があります。詳細については、MQPUT 呼び出しの説明の『使用上の注意』を参照してください。
3. MQPUT1 呼び出しは、配布リストへの各メッセージの書き込みに使用できます。この点についての一般情報は、MQOPEN および MQPUT 呼び出しの『使用上の注意』を参照してください。

MQPUT1 呼び出しを使用する場合、次のような相違点があります。

- a. MQRR 応答レコードがアプリケーションによって提供されている場合、MQOD 構造体を使用して提供される必要があります。MQPMO 構造体を使用しても提供されません。
- b. 理由コード RC2137 は、MQPUT1 により応答レコード内に戻されることはありません。キューがオープンに失敗すると、そのキューの応答レコードには、オープン操作の結果発生する実際の理由コードが設定されます。

キューのオープン操作が成功し、完了コード CCWARN が戻る、そのキューの応答レコード内の完了コードおよび理由コードは、書き込み操作の結果発生する完了コードと理由コードにより置き換えられます。

MQOPEN および MQPUT 呼び出しと同様、キュー・マネージャーは、呼び出しの結果が配布リスト内のすべてのキューについて同じでない場合のみ、応答レコード (提供されている場合) を設定します。これは、呼び出しが終了し、理由コード RC2136 が戻ることに示されます。

4. クラスター・キューにメッセージを書き込むために MQPUT1 呼び出しを使用すると、MQOPEN 呼び出しで OOBNDN を指定した場合と同様の結果になります。
5. アプリケーション・メッセージ・データの先頭にある 1 つ以上の IBM MQ ヘッダー構造体を使用してメッセージが書き込まれる場合、キュー・マネージャーはそのヘッダー構造体に対して一定の検査を実行し、それらが有効であるか検証します。これに関する詳細については、MQPUT 呼び出しの『使用上の注意』を参照してください。
6. 複数の警告状態が発生した場合 (**CMPCOD** パラメーターを参照)、返される理由コードは、以下のリストで該当するもののうち、最初の理由コードです。
 - a. RC2136
 - b. RC2242
 - c. RC2241
 - d. RC2049 または RC2104
7. RPG プログラミング例に示す **BUFFER** パラメーターは、ストリングとして宣言されています。このため、パラメーターの最大長は 256 バイトに制限されます。これより大きいバッファが必要な場合は、このパラメーターをストリングではなく構造体として宣言するか、あるいは物理ファイルのフィールドとして宣言する必要があります。そうすると、指定できる最大長が約 32 KB まで増加します。

パラメーター

MQPUT1 呼び出しには、以下のパラメーターがあります。

HCONN (10 桁の符号付き整数) - 入力

接続ハンドル。

このハンドルは、キュー・マネージャーに対する接続を表します。HCONN の値は、先行の MQCONN または MQCONNX 呼び出しによって戻されたものです。

OBJDSC (MQOD) - 入出力

オブジェクト記述子。

これは、メッセージが追加される先のキューを識別する構造です。詳細は [1166 ページの『IBM i での MQOD \(オブジェクト記述子\)』](#)を参照してください。

ユーザーは、出力のためのキューをオープンする権限を持っていないければなりません。このキューは、モデル・キューであってはなりません。

MSGDSC (MQMD) - 入出力

メッセージ記述子。

この構造体は、送信されるメッセージの属性を記述するものであり、書き込み要求が完了した後で、フィードバック情報を受け取ります。詳細は [1114 ページの『IBM iでの MQMD \(メッセージ記述子\)』](#) を参照してください。

アプリケーションがバージョン 1 の MQMD を提供している場合、メッセージ・データの接頭部に MQMDE 構造体を付けると、バージョン 2 の MQMD に存在し、バージョン 1 には存在しない各フィールドの値を指定できます。MQMD 内の MDFMT フィールドは、MQMDE が存在することを示すため、FMMDE に設定しておく必要があります。詳細については、[1159 ページの『IBM iでの MQMDE \(拡張メッセージ記述子\)』](#) を参照してください。

PMO (MQPMO) - 入出力

MQPUT1 のアクションを制御するオプション。

詳細は [1180 ページの『IBM iでの MQPMO \(メッセージ書き込みオプション\)』](#) を参照してください。

BUFLEN (10 桁の符号付き整数) - 入力

BUFFER 中のメッセージの長さ。

ゼロは有効であり、メッセージにアプリケーション・データが含まれていないことを示します。この上限は、さまざまな要因によって変化します。詳細については、MQPUT 呼び出しの **BUFLEN** パラメーターの説明を参照してください。

BUFFER (1 バイトのビット・ストリング x BUFLEN) - 入力

メッセージ・データ。

これは、送信するアプリケーション・メッセージ・データが入っているバッファーです。バッファーは、メッセージ内のデータの性質に適した境界に合わせなければなりません。IBM MQ ヘッダー構造になっているメッセージを含む、ほとんどのメッセージには 4 バイトの位置合わせが適していますが、メッセージによってはより厳しい位置合わせを必要とする場合があります。例えば、64 ビット・バイナリー整数を含むメッセージは 8 バイト境界に合わせる必要がある場合があります。

BUFFER に文字データ、数値データ、またはその両方が含まれている場合、**MSGDSC** パラメーターの **MDCSI** および **MDENC** フィールドは、データに適切な値に設定する必要があります。これにより、メッセージの受信側は、(必要に応じて) データを受信側が使用する文字セットおよびエンコードに変換できるようになります。

注：MQPUT1 呼び出しにある他のパラメーターはすべて、**CodedCharSetId** キュー・マネージャー属性で指定した文字セットと ENNAT で指定したローカル・キュー・マネージャーのエンコードで記述されていなければなりません。

CMPCOD (10 桁の符号付き整数) - 出力

完了コード。

これは、以下のいずれかになります。

CCOK

正常終了。

CCWARN

警告 (部分完了)。

CCFAIL

呼び出し失敗。

REASON (10 桁の符号付き整数) - 出力

CMPCOD を限定する理由コード。

CMPCOD が CCOK の場合

RCNONE

(0, X'000') レポートする理由コードはありません。

CMPCOD が CCWARN の場合

RC2104

(2104, X'838') メッセージ記述子のレポート・オプションが認識されない。

RC2136

(2136, X'858') 複数の理由コードが返されました。

RC2049

(2049, X'801') メッセージ優先順位が、サポートされる最大値を超えている。

RC2241

(2241, X'8C1') メッセージ・グループが不完全である。

RC2242

(2242, X'8C2') 論理メッセージが不完全である。

CMPCOD が CCFAIL の場合

RC2001

(2001, X'7D1') 別名基本キューのタイプは無効です。

RC2004

(2004, X'7D4') バッファ・パラメーターが無効である。

RC2005

(2005, X'7D5') バッファ長パラメーターは無効です。

RC2009

(2009, X'7D9') キュー・マネージャーとの接続が失われました。

RC2013

(2013, X'7DD') 満了時刻が無効である。

RC2014

(2014, X'7DE') フィードバック・コードが無効である。

RC2017

(2017, X'7E1') 使用可能なハンドルがなくなりました。

RC2018

(2018, X'7E2') 接続ハンドルが無効です。

RC2024

(2024, X'7E8') 現行の作業単位内では、これ以上メッセージを処理できない。

RC2026

(2026, X'7EA') メッセージ記述子が無効である。

RC2027

(2027, X'7EB') 応答先キューがない。

RC2029

(2029, X'7ED') メッセージ記述子のメッセージ・タイプが無効である。

RC2030

(2030, X'7EE') メッセージの長さが、キューの最大許容数より大きいです。

RC2031

(2031, X'7EF') メッセージ長がキュー・マネージャーの最大許容長より大きいです。

RC2035

(2035, X'7F3') アクセスは許可されません。

RC2042

(2042, X'7FA') オプションが矛盾するオブジェクトが既に関いています。

RC2043

(2043, X'7FB') オブジェクト・タイプが無効です。

RC2044

(2044, X'7FC') オブジェクト記述子の構造が無効です。

RC2046

(2046, X'7FE') オプションが無効であるか、矛盾しています。

RC2047

(2047, X'7FF') 持続性が無効である。

RC2048

(2048, X'800') キューは永続的なメッセージをサポートしていません。

RC2050

(2050, X'802') メッセージ優先順位が無効である。

RC2051

(2051, X'803') このキューでは書き込み呼び出しが使用禁止になっています。

RC2052

(2052, X'804') キューが削除されました。

RC2053

(2053, X'805') キューには既に最大数のメッセージが入っています。

RC2056

(2056, X'808') ディスク上にキューのためのスペースがありません。

RC2057

(2057, X'809') キュー・タイプが無効です。

RC2058

(2058, X'80A') キュー・マネージャー名が無効であるか、認識されていません。

RC2059

(2059, X'80B') キュー・マネージャーを接続に使用できません。

RC2061

(2061, X'80D') メッセージ記述子のレポート・オプションが無効である。

RC2063

(2063, X'80F') セキュリティー・エラーが発生しました。

RC2071

(2071, X'817') ストレージが不足しています。

RC2072

(2072, X'818') 同期点サポートが利用できない。

RC2082

(2082, X'822') 別名の基本キューが不明です。

RC2085

(2085, X'825') オブジェクト名が不明です。

RC2086

(2086, X'826') オブジェクトのキュー・マネージャーが不明です。

RC2087

(2087, X'827') リモート・キュー・マネージャーが不明です。

RC2091

(2091, X'82B') 伝送キューはローカルではありません。

RC2092

(2092, X'82C') 伝送キューの使用方法が正しくありません。

RC2097

(2097, X'831') 参照されたキュー・ハンドルがコンテキストを保存しない。

RC2098

(2098, X'832') 参照されたキュー・ハンドルでコンテキストが使用できない。

RC2101

(2101, X'835') オブジェクトが損傷しました。

RC2102

(2102, X'836') 使用できるシステム・リソースが不足しています。

RC2135

(2135, X'857') 配布ヘッダー構造体が無効である。

RC2136

(2136, X'858') 複数の理由コードが返されました。

RC2149

(2149, X'865') PCF 構造体が無効である。

RC2154

(2154, X'86A') 存在するレコード数が無効です。

RC2155

(2155, X'86B') オブジェクト・レコードが無効です。

RC2156

(2156, X'86C') 応答レコードが無効です。

RC2158

(2158, X'86E') 書き込みメッセージ・レコード・フラグが無効である。

RC2159

(2159, X'86F') 書き込みメッセージ・レコードが無効である。

RC2161

(2161, X'871') キュー・マネージャーが静止しています。

RC2162

(2162, X'872') キュー・マネージャーのシャットダウン中です。

RC2173

(2173, X'87D') 書き込みメッセージ・オプションの構造体が無効である。

RC2184

(2184, X'888') リモート・キュー名が無効です。

RC2188

(2188, X'88C') クラスター・ワークロード出口によって呼び出しが拒否されました。

RC2189

(2189, X'88D') クラスター名の解決に失敗しました。

RC2195

(2195, X'893') 予期しないエラーが発生しました。

RC2196

(2196, X'894') 伝送キューが不明です。

RC2197

(2197, X'895') デフォルト伝送キューが不明です。

RC2198

(2198, X'896') デフォルト伝送キューはローカルではありません。

RC2199

(2199, X'897') デフォルト伝送キューの使用法エラー。

RC2258

(2258, X'8D2') グループ ID が無効である。

RC2248

(2248, X'8C8') メッセージ記述子の拡張子が無効である。

RC2219

(2219, X'8AB') 前の呼び出しが完了する前に MQI 呼び出しが再入力されました。

RC2249

(2249, X'8C9') メッセージ・フラグが無効である。

RC2250

(2250, X'8CA') メッセージ順序番号が無効である。

RC2251

(2251, X'8CB') メッセージ・セグメント・オフセットが無効である。

RC2252

(2252, X'8CC') 元の長さが無効である。

RC2253

(2253, X'8CD') メッセージ・セグメント内のデータの長さがゼロである。

RC2255

(2255, X'8CF') 作業単位がキュー・マネージャーから使用不可。

RC2257

(2257, X'8D1') 提供された MQMD のバージョンが違っている。

RC2266

(2266, X'8DA') クラスター・ワークロード出口で障害が発生しました。

RC2269

(2269, X'8DD') クラスター・リソース・エラー。

RC2270

(2270, X'8DE') 使用可能な宛先キューがない。

RC2420

(2420) MQPUT1 呼び出しが発行されましたが、メッセージ・データに無効な MQEPH 構造体が含まれます。

RC2551

(2551, X'9F7') 指定された選択ストリングを使用できません。

RC2554

(2554, X'9FA') 拡張メッセージ・セレクターを使ってサブスクライバーにメッセージを送達すべきかどうか決定するために、メッセージの内容を解析することができませんでした。

RPG 宣言

```
C*.1.....2.....3.....4.....5.....6.....7..
C                                CALLP      MQPUT1(HCONN : OBJDSC : MSGDSC :
C                                PMO : BUFLN : BUFFER :
C                                CMPCOD : REASON)
```

呼び出しのプロトタイプ定義は次のようになります。

```
D*.1.....2.....3.....4.....5.....6.....7..
D MQPUT1          PR          EXTPROC('MQPUT1')
D* Connection handle
D HCONN          10I 0 VALUE
D* Object descriptor
D OBJDSC          468A
D* Message descriptor
D MSGDSC          364A
D* Options that control the action of MQPUT1
D PMO            200A
D* Length of the message in BUFFER
D BUFLN          10I 0 VALUE
D* Message data
D BUFFER          *   VALUE
D* Completion code
D CMPCOD          10I 0
D* Reason code qualifying CMPCOD
D REASON          10I 0
```

IBM i IBM i での MQSET (オブジェクト属性の設定)

MQSET 呼び出しは、ハンドルで表されるオブジェクトの属性を変更するために使用します。このオブジェクトはキューでなければなりません。

- [1361 ページの『構文』](#)
- [1361 ページの『使用上の注意』](#)
- [1361 ページの『パラメーター』](#)
- [1365 ページの『RPG 宣言』](#)

構文

MQSET (HCONN, HOBJ, SELCNT, SELS, IACNT, INTATR, CALEN, CHRATR, CMPCOD, REASON)

使用上の注意

1. この呼び出しを使用する場合、アプリケーションは、整数属性の配列、文字属性ストリングの集合、またはその両方を指定できます。エラーが発生しなければ、指定された属性はすべて同時に設定されます。エラーが発生した場合 (セレクターが無効である場合や、属性に無効な値を設定しようとした場合など)、この呼び出しは失敗し、属性は設定されません。
2. 属性の値は、MQINQ 呼び出しを使用して調べることができます。詳細については、[1316 ページの『IBM iでの MQINQ \(オブジェクト属性の照会\)』](#)を参照してください。

注: MQINQ 呼び出しを使用して値を照会できる属性のすべてが、MQSET 呼び出しを使用して値を変更できるわけではありません。例えば、プロセス・オブジェクトもキュー・マネージャー属性も、この呼び出しでは設定できません。

3. 属性の変更は、キュー・マネージャーを再始動しても維持されます (一時動的キューへの変更は例外であり、キュー・マネージャーを再始動すると失われます)。
4. モデル・キューの属性は、MQSET 呼び出しを使用して変更できません。ただし、MQOO_SET オプションを指定した MQOPEN 呼び出しを使用してモデル・キューをオープンした場合は、その MQOPEN 呼び出しで作成した動的ローカル・キューの属性を、MQSET 呼び出しを使用して設定することができます。
5. 設定対象のオブジェクトがクラスター・キューの場合、正常にオープンするには、クラスター・キューのローカル・インスタンスが必要です。

オブジェクト属性の詳細については、以下を参照してください。

- [1380 ページの『キューの属性』](#)
- [1408 ページの『名前リストの属性』](#)
- [1410 ページの『IBM iでのプロセス定義の属性』](#)
- [1412 ページの『IBM iでのキュー・マネージャーの属性』](#)

パラメーター

MQSET 呼び出しには、以下のパラメーターがあります。

HCONN (10 桁の符号付き整数) - 入力

接続ハンドル。

このハンドルは、キュー・マネージャーに対する接続を表します。HCONN の値は、先行の MQCONN または MQCONNX 呼び出しによって戻されたものです。

HOBJ (10 桁の符号付き整数) - 入力

オブジェクト・ハンドル

このハンドルは、属性を設定するキュー・オブジェクトを表します。このハンドルは、OOSSET オプションを指定した先行の MQOPEN 呼び出しによって戻されたものです。

SELCNT (10 桁の符号付き整数) - 入力

セレクターのカウント。

これは、SELS 配列で提供されるセレクトーのカウンタです。設定する属性の数です。ゼロは有効な値です。許可される最大数は 256 です。

SELS (10 桁の符号付き整数 x SELCNT) - 入力

属性セレクトーの配列。

これは、**SELCNT** 個の属性セレクトーで構成される配列です。各セレクトーは、値を設定する属性 (整数または文字) を示します。

各セレクトーは、HOBJ が表すキューのタイプに対して有効でなければなりません。特定の IA* および CA* 値のみが許可されます。これらの値については、このセクションの後半にリストしています。

選択子は任意の順序で指定できます。整数属性セレクトー (IA* セレクトー) に対応する属性値は、INTATR に、SELS でのセレクトーの出現順序と同じ順序で指定されていなければなりません。文字属性セレクトー (CA* セレクトー) に対応する属性値は、CHRATR に、セレクトーの出現順序と同じ順序で指定されていなければなりません。IA* セレクトーが CA* セレクトーに挟まれていても構いません。重要なのは、それぞれのタイプにおける相対順序のみです。

同じセレクトーを複数回指定してもエラーにはなりません。そうした場合は、その特定のセレクトーに対して最後に指定された値が、実際に反映される値となります。

注:

1. 整数および文字属性セレクトーは、2 つの異なる範囲内に割り当てられます。IA* セレクトーは、IAFRST から IALAST までの範囲に、CA* セレクトーは、CAFRST から CALAST までの範囲にあります。

各範囲について、定数 IALSTU および CALSTU によって、キュー・マネージャーが受け入れる最大値を定義しています。

2. すべての IA* セレクトーが先頭にある場合は、SELS および INTATR 配列で同じエレメント番号を使用して、対応するエレメントのアドレスを示すことができます。

設定できる属性を、以下の表にリストします。これ以外の属性は、この呼び出しを使用しても設定できません。CA* 属性セレクトーの場合、CHRATR に必要なストリングの長さ (バイト単位) を定義する定数は、括弧内に指定します。

表 790. キューに関する MQSET 属性セレクトー		
セレクトー	説明	注記
CATRGD	トリガー・データ (LNTRGD)。	1363 ページの『2』
IADIST	配布リスト・サポート。	1363 ページの『1』
IAIGET	読み取り操作が許されているかどうかを判別します。	
IAIPUT	書き込み操作が許されているかどうかを判別します。	
IATRGC	トリガー制御。	1363 ページの『2』
IATRGD	トリガー項目数。	1363 ページの『2』
IATRGP	トリガーのしきい値メッセージ優先順位。	1363 ページの『2』

表 790. キューに関する MQSET 属性セクター (続き)		
セクター	説明	注記
IATRGT	トリガー・タイプ。	1363 ページの『2』

注:

1. 以下のプラットフォームでのみサポートされます。

-  AIX
-  IBM i
-  Windows

および、これらのシステムに接続された IBM MQ クライアント。

2. VSE/ESA ではサポートされません。

INTACT (10 桁の符号付き整数) - 入力

整数属性のカウント。

これは、INTATR 配列の要素数です。SELS パラメーターに指定された IA* セクターの数以上でなければなりません。何もない場合はゼロが有効な値です。

INTATR (10 桁の符号付き整数 x r x IACNT) - 入力

整数属性の配列。

これは、IACNT 個の整数属性値の配列です。これらの属性値は、SELS 配列内の IA* セクターと同じ順序で並んでいなければなりません。

CALEN (10 桁の符号付き整数) - 入力

文字属性バッファの長さ。

これは、CHATTR パラメーターの長さ (バイト単位) であり、少なくとも、SELS 配列に指定されている各文字属性の長さの合計でなければなりません。SELS に CA* セクターが指定されていない場合はゼロが有効な値です。

CHATTR (1 バイト文字ストリング x CALEN) - 入力

文字属性。

これは、各文字属性値が連結されて入っている バッファです。バッファの長さは、CALEN パラメーターによって指定されます。

これらの文字属性は、SELS 配列内の CA* セクターと同じ順序で指定されていなければなりません。各文字属性の長さは固定です (SELS を参照)。属性の設定値に含まれる空白以外の文字数が、属性の定義された長さよりも少ない場合、その値は属性の定義された長さに一致 CHATTR するよう、右側に空白を埋めて調整されなければならない。

CMPCOD (10 桁の符号付き整数) - 出力

完了コード。

これは、以下のいずれかになります。

CCOK

正常終了。

CCFAIL

呼び出し失敗。

REASON (10 桁の符号付き整数) - 出力

CMPCOD を限定する理由コード。

CMPCOD が CCOK の場合

RCNONE

(0, X'000') レポートする理由コードはありません。

CMPCOD が CCFAIL の場合

RC2219

(2219, X'8AB') 前の呼び出しが完了する前に MQI 呼び出しが再入力されました。

RC2006

(2006, X'7D6') 文字属性の長さが無効である。

RC2007

(2007, X'7D7') 文字属性ストリングが無効である。

RC2009

(2009, X'7D9') キュー・マネージャーとの接続が失われました。

RC2018

(2018, X'7E2') 接続ハンドルが無効です。

RC2019

(2019, X'7E3') オブジェクト・ハンドルが無効です。

RC2020

(2020, X'7E4') 取得禁止または書き込み禁止のキュー属性の値が無効です。

RC2021

(2021, X'7E5') 整数属性のカウントが無効です。

RC2023

(2023, X'7E7') 整数属性の配列が無効です。

RC2040

(2040, X'7F8') キューが設定用にオープンされていません。

RC2041

(2041, X'7F9') オープンされた後でオブジェクト定義が変更された。

RC2101

(2101, X'835') オブジェクトが損傷しました。

RC2052

(2052, X'804') キューが削除されました。

RC2058

(2058, X'80A') キュー・マネージャー名が無効であるか、認識されていません。

RC2059

(2059, X'80B') キュー・マネージャーを接続に使用できません。

RC2162

(2162, X'872') キュー・マネージャーのシャットダウン中です。

RC2102

(2102, X'836') 使用できるシステム・リソースが不足しています。

RC2065

(2065, X'811') セレクターのカウントが無効である。

RC2067

(2067, X'813') 属性選択子が無効です。

RC2066

(2066, X'812') セレクターのカウントが大きすぎる。

RC2071

(2071, X'817') ストレージが不足しています。

RC2075

(2075, X'81B') トリガー制御属性の値が無効です。

RC2076

(2076, X'81C') トリガー項目数属性の値が無効です。

RC2077

(2077, X'81D') トリガー・メッセージ優先順位属性の値が無効です。

RC2078

(2078, X'81E') トリガー・タイプ属性の値が無効です。

RC2195

(2195, X'893') 予期しないエラーが発生しました。

RPG 宣言

```
C*.1.....2.....3.....4.....5.....6.....7..  
C                                CALLP      MQSET(HCONN : HOBJ : SELCNT :  
C                                SELS(1) : IACNT : INTATR(1) :  
C                                CALEN : CHRATR : CMPCOD :  
C                                REASON)
```

呼び出しのプロトタイプ定義は次のようになります。

```
D*.1.....2.....3.....4.....5.....6.....7..  
DMQSET                PR                EXTPROC('MQSET')  
D* Connection handle  
D HCONN                10I 0 VALUE  
D* Object handle  
D HOBJ                10I 0 VALUE  
D* Count of selectors  
D SELCNT              10I 0 VALUE  
D* Array of attribute selectors  
D SELS                10I 0  
D* Count of integer attributes  
D IACNT               10I 0 VALUE  
D* Array of integer attributes  
D INTATR              10I 0  
D* Length of character attributes buffer  
D CALEN               10I 0 VALUE  
D* Character attributes  
D CHRATR              *    VALUE  
D* Completion code  
D CMPCOD              10I 0  
D* Reason code qualifying CMPCOD  
D REASON              10I 0
```

IBM i IBM i での MQSETMP (メッセージ・ハンドルのプロパティの設定)

MQSETMP 呼び出しは、メッセージ・ハンドルのプロパティを設定したり変更したりします。

- [1365 ページの『構文』](#)
- [1365 ページの『使用上の注意』](#)
- [1367 ページの『パラメーター』](#)
- [1370 ページの『RPG 宣言』](#)

構文

MQSETMP (*Hconn*, *Hmsg*, *SetPropOpts*, *Name*, *PropDesc*, *Type*, *ValueLength*, *Value*, *CompCode*, *Reason*)

使用上の注意

- この呼び出しは、キュー・マネージャーそのものが作業単位を調整するときのみ使用できます。次のタイプがあります。
 - ローカル作業単位 (変更内容は IBM MQ リソースにのみ影響を及ぼす)。
 - グローバル作業単位 (変更内容は、IBM MQ リソースだけでなく、他のリソース・マネージャーに属するリソースにも影響を及ぼす場合がある)。

ローカル作業単位およびグローバル作業単位の詳細については、[1264 ページの『IBM i での MQBEGIN \(作業単位の開始\)』](#)を参照してください。

- キュー・マネージャーが作業単位を調整しない環境では、MQBACK ではなく適切なバックアウト呼び出しを使用してください。この環境ではまた、アプリケーションの異常終了を原因とする暗黙的バックアウトをサポートすることもできます。
 - z/OS では、以下の呼び出しを使用してください。
 - バッチ・プログラム (IMS バッチ DL/I プログラムを含む) は、作業単位が IBM MQ リソースにのみ影響する場合に、MQBACK 呼び出しを使用できます。一方、作業単位が IBM MQ リソースと、その他のリソース・マネージャー (Db2 など) に属するリソースの両方に影響を及ぼす場合は、z/OS Recoverable Resource Service (RRS) が提供する SRRBACK 呼び出しを使用できます。SRRBACK 呼び出しを実行すると、RRS 調整対応のリソース・マネージャーに属するリソースに対する変更がバックアウトされます。
 - CICS アプリケーションは、EXEC CICS SYNCPOINT ROLLBACK コマンドを使用して作業単位をバックアウトする必要があります。CICS アプリケーションには MQBACK 呼び出しを使用しないでください。
 - IMS アプリケーション (バッチ DL/I プログラム以外) は、ROLB などの IMS 呼び出しを使用して、作業単位をバックアウトする必要があります。IMS アプリケーション (バッチ DL/I プログラム以外) には、MQBACK 呼び出しを使用しないでください。
 - IBM i では、この呼び出しはキュー・マネージャーで調整されるローカル作業単位で使用してください。これは、コミットメント定義がジョブ・レベルで存在してはならないことを意味します。つまり、**CMTSCOPE(*JOB)** パラメーターを指定した STRCMTCTL コマンドがジョブに対して発行されているわけではありません。
- 作業単位内にあるコミットされていない変更内容でアプリケーションが終了する場合、それらの変更内容の後処理は、そのアプリケーションが正常に終了するか、異常終了するかで異なります。詳細については、[1300 ページの『IBM i での MQDISC \(キュー・マネージャーの切断\)』](#)の使用上の注意を参照してください。
- アプリケーションでグループ内のメッセージまたは論理メッセージのセグメントの書き込みまたは読み取りを行う場合、キュー・マネージャーは、最後に MQPUT および MQGET 呼び出しが正常に実行されたメッセージ・グループに関する情報を保存します。この情報は、キュー・ハンドルに関する次のような情報です。

- MQMD 中の *GroupId*、*MsgSeqNumber*、*Offset*、および *MsgFlags* フィールドの値。
- そのメッセージが作業単位の一部であるかどうか。
- MQPUT 呼び出しについて、そのメッセージが持続メッセージか、非持続メッセージか。

キュー・マネージャーは、次のものについて 1 つずつ、3 セットのグループおよびセグメント情報を保持しています。

- 最後に正常に実行された MQPUT 呼び出し (これは作業単位の一部である場合があります)。
- 最後に正常に実行された MQGET 呼び出しのうちキューからメッセージを削除したもの (作業単位の一部である場合があります)。
- 最後に正常に実行された MQGET 呼び出しのうちキュー上のメッセージをブラウズしたもの (これが作業単位の一部であることはありません)。

アプリケーションで、作業単位の一部としてそのメッセージの書き込みまたは読み取りを行い、その作業単位をバックアウトすると、そのグループおよびセグメント情報は、その以前の値に復元されます。

- MQPUT 呼び出しについての情報は、現行作業単位内のそのキュー・ハンドルについて最初に正常に実行された MQPUT 呼び出し以前の値に復元されます。
- MQGET 呼び出しについての情報は、現行作業単位内のそのキュー・ハンドルについて最初に正常に実行された MQGET 呼び出し以前の値に復元されます。

作業単位の開始後にアプリケーションによって更新されたキューであっても、それが作業単位の有効範囲外である場合は、その作業単位がバックアウトされても、グループおよびセグメント情報は復元されません。

作業単位のバックアウト時にグループおよびセグメント情報を以前の値に復元する機能により、アプリケーションは、数多くのセグメントで構成される大きなメッセージ・グループまたは大きな論理メッセージをいくつかの作業単位にまたがって広げることができます。そして、いずれかの作業単位が失敗しても、そのメッセージ・グループまたは論理メッセージ内の正しい点でアプリケーションを再始動できます。

ローカル・キュー・マネージャーのキュー・ストレージが限られている場合には、いくつかの作業単位を使用する方が有効となる場合があります。ただし、システム障害の発生時に各メッセージの書き込みまたは読み取りを正しい時点で再始動できるようにするには、アプリケーションが十分な情報を維持している必要があります。

システム障害後に正しい時点から再始動する方法の詳細については、[PMOPT \(10 桁の符号付き整数\)](#) の [PMLOGO](#) オプション、および [GMOPT \(10 桁の符号付き整数\)](#) の [GMLOGO](#) オプションを参照してください。

次の『使用上の注意』は、キュー・マネージャーで作業単位を調整する場合にのみ適用されます。

- 作業単位の 1 つには、1 つの接続ハンドルと同じ有効範囲があります。特定の作業単位に影響を与えるすべての IBM MQ 呼び出しは、同じ接続ハンドルを使用して実行しなければなりません。別の接続ハンドルを用いて呼び出しを発行すると (例えば、別のアプリケーションで呼び出しを発行する)、別の作業単位に影響が及びます。接続ハンドルの有効範囲については、[HCONN \(10 桁の符号付き整数\) - 出力](#)を参照してください。
- この呼び出しで影響を受けるメッセージは、現行の作業単位の一部として書き込まれたメッセージ、または取り出されたメッセージに限られます。
- 長時間実行しているアプリケーションが、1 つの作業単位に対して MQGET、MQPUT、または MQPUT1 呼び出しを発行する一方、コミット呼び出しまたはバックアウト呼び出しを一度も発行しない場合には、キューが他のアプリケーションでは使用できないメッセージで満杯になることがあります。この可能性を回避するために、管理者は、**MaxUncommittedMsgs** キュー・マネージャー属性を、ランナウェイ・アプリケーションがキューを満杯にしないように十分に低い値に設定する必要がありますが、予期されるメッセージング・アプリケーションが正しく機能するように十分に高い値に設定する必要があります。

パラメーター

MQSETMP 呼び出しには、以下のパラメーターがあります。

HCONN (10 桁の符号付き整数) - 入力

このハンドルは、キュー・マネージャーに対する接続を表します。

値は、**HMSG** パラメーターで指定されているメッセージ・ハンドルを作成するために使用された接続ハンドルと一致していなければなりません。

メッセージ・ハンドルが、HCUNAS を使用して作成されている場合、メッセージ・ハンドルのプロパティを設定するスレッドで、有効な接続が確立されている必要があります。接続がないと、呼び出しは理由コード RC2009 で失敗します。

HMSG (20 桁の符号付き整数) - 入力

これは変更されるメッセージ・ハンドルです。値は、前の MQCRTMH 呼び出しで戻されたものです。

SETOPT (MQSMPO) - 入力

メッセージ・プロパティの設定方法を制御します。

この構造を使用すると、アプリケーションで、メッセージ・プロパティの設定方法を制御するオプションを指定できます。この構造は、MQSETMP 呼び出しの入力パラメーターです。詳細については、[MQSMPO](#) を参照してください。

PRNAME (MQCHARV) - 入力

これは、設定するプロパティの名前です。

プロパティ名の使用については、[プロパティ名およびプロパティ名に関する制約事項](#)を参照してください。

PRPDSC (MQPD) - 入出力

この構造を使用して、以下を含むプロパティの属性を定義します。

- プロパティがサポートされていない場合に発生すること
- プロパティが属しているメッセージ・コンテキスト
- プロパティがフロー時にコピーされるメッセージ

この構造体の詳細については、[MQPD](#) を参照してください。

TYPE (10 桁の符号付き整数) - 入力

設定するプロパティのデータ・タイプ。これは以下のいずれかです。

TYPBOL

ブール値。 *ValueLength* は 4 でなければなりません。

TYPBST

バイト・ストリング。 *ValueLength* はゼロ以上でなければなりません。

TYPI8

8 ビットの符号付き整数。 *ValueLength* は 1 でなければなりません。

TYPI16

16 ビットの符号付き整数。 *ValueLength* は 2 でなければなりません。

TYPI32

32 ビットの符号付き整数。 *ValueLength* は 4 でなければなりません。

TYPI64

64 ビットの符号付き整数。 *ValueLength* は 8 でなければなりません。

TYPF32

32 ビットの浮動小数点数。 *ValueLength* は 4 でなければなりません。

TYPF64

64 ビットの浮動小数点数。 *ValueLength* は 8 でなければなりません。

TYPSTR

文字ストリング。 *ValueLength* はゼロ以上か、特殊値 VLNULL でなければなりません。

TYPNUL

プロパティは存在しますがヌル値です。 *ValueLength* はゼロでなければなりません。

VALLEN (10 桁の符号付き整数) - 入力

Value パラメーターのプロパティ値の長さ (バイト数)。

ヌル値、ストリング、バイト・ストリングの場合のみ、ゼロが有効です。ゼロは、プロパティは存在するものの、値に文字またはバイトが入っていないことを示します。

値はゼロ以上であるか、以下の特殊値 (*Type* パラメーターが TYPSTR に設定されている場合) でなければなりません。

VLNULL

値はストリング内で最初に検出されるヌルで区切られます。ヌルはストリングの一部には含められません。この値は、TYPSTR が設定されていない場合には無効です。

注: VLNULL が設定されている場合にストリングの終端となるヌル文字は、*Value* の文字セットのヌルです。

VALUE (1 バイトのビット・ストリング x VALLEN) - 入力

設定するプロパティの値。バッファーは、値のデータの性質に適した境界に位置合わせされなければなりません。

C プログラミング言語では、パラメーターは、void を示すポインターとして宣言されます。つまり、どのタイプのデータのアドレスもパラメーターとして指定できます。

ValueLength がゼロの場合は、*Value* は参照されません。この場合、C または System/390 アセンブラーで作成されたプログラムによって渡されるパラメーター・アドレスはヌルのこともあります。

CMPCOD (10 桁の符号付き整数) - 出力

完了コード。以下のいずれかです。

CCOK

正常終了。

CCFAIL

呼び出し失敗。

REASON (10 桁の符号付き整数) - 出力

CMPCOD を限定する理由コード。

CMPCOD が CCOK の場合

RCNONE

(0, X'000') レポートする理由コードはありません。

CMPCOD が CCWARN の場合

RC2421

(2421, X'0975') プロパティーを含む MQRFH2 フォルダーを構文解析できなかった。

CMPCOD が CCFAIL の場合

RC2204

(2204, X'089C') アダプターが利用できません。

RC2130

(2130, X'852') アダプター・サービス・モジュールをロードできません。

RC2157

(2157, X'86D') 1 次 ASID とホーム ASID が異なります。

RC2004

(2004, X'07D4') 値パラメーターが無効である。

RC2005

(2005, X'07D5') 値長パラメーターが無効である。

RC2219

(2219, X'08AB') 前の呼び出しが完了する前に MQI 呼び出しが入力された。

RC2460

(2460, X'099C') メッセージ・ハンドル・ポインターが無効。

RC2499

(2499, X'09C3') メッセージ・ハンドルがすでに使用中。

RC2046

(2046, X'07FE') オプションが無効であるか、矛盾しています。

RC2482

(2482, X'09B2') プロパティー記述子の構造体が無効である。

RC2442

(2442, X'098A') プロパティー名が無効である。

RC2473

(2473, X'09A9') プロパティーのデータ・タイプが無効である。

RC2472

(2472, X'09A8') 値データ中に数字フォーマット・エラーが発生した。

RC2463

(2463, X'099F') メッセージ・プロパティー設定オプションの構造が無効である。

RC2111

(2111, X'083F') プロパティー名エンコード文字セット ID が無効である。

RC2071

(2071, X'817') ストレージが不足しています。

RC2195

(2195, X'893') 予期しないエラーが発生しました。

詳しくは、[1426 ページの『IBM i の戻りコード \(ILE RPG\)』](#)を参照してください。

RPG 宣言

```
C*.1.....2.....3.....4.....5.....6.....7..  
C                                CALLP      MQSETMP(HCONN : HMSG : SETOPT :  
                                           PRNAME : PRPDSC :  
                                           TYPE : VALLEN : VALUE :  
                                           CMPCOD : REASON)
```

呼び出しのプロトタイプ定義は次のようになります。

```
DMQSETMP          PR          EXTPROC('MQSETMP')  
D* Connection handle  
D HCONN           10I 0 VALUE  
D* Message handle  
D HMSG           10I 0 VALUE  
D* Options that control the action of MQSETMP  
D SETOPT         20A  
D* Property name  
D PRNAME         32A  
D* Property descriptor  
D PRPDSC         24A  
D* Property data type  
D TYPE          10I 0 VALUE  
D* Length of the Value area  
D VALLEN        10I 0 VALUE  
D* Property value  
D VALUE         *    VALUE  
D* Completion code  
D CMPCOD        10I 0  
D* Reason code qualifying CompCode  
D REASON        10I 0
```

IBM i IBM i での MQSTAT (状況情報の取り出し)

MQSTAT 呼び出しを使用して、状況情報を取り出します。返される状況情報のタイプは、呼び出しで指定される STYPE 値で決定されます。

- [1370 ページの『構文』](#)
- [1370 ページの『使用上の注意』](#)
- [1371 ページの『パラメーター』](#)
- [1372 ページの『RPG 宣言』](#)

構文

MQSTAT (HCONN, STYPE, STAT, CMPCOD, REASON)

使用上の注意

1. STATAPT のタイプを指定して MQSTAT を呼び出すと、前の非同期 MQPUT および MQPUT1 操作に関する情報が戻ります。呼び出しで渡される MQSTAT 構造体には、その接続において最初に記録された非同期の警告またはエラー情報が入れます。この最初の記録に続いてエラーや警告が発生しても、通常これらの値は変更されません。ただし、完了コード CCWARN のエラーが起きると、CCFAIL の完了コードを持つ後続の障害が代わりに返されます。
2. 接続が確立されてから、または最後に MQSTAT を呼び出してからエラーが発生していない場合、CCOK の CMPCOD および RCNONE の REASON が返されます。
3. 接続ハンドルの下で処理された非同期呼び出しのカウンタ数は、STSPSC、STSPWC、および STSPFC という 3 つのカウンターを使用することによって返されます。これらのカウンターは、非同期操作が正常に処理されるか、警告を受けるか、または失敗するたびにキュー・マネージャーによって増分されます。

(会計上の目的で配布リストに挿入する場合、配布リスト当たり 1 回ではなく宛先キュー当たり 1 回ずつカウントされることに注意してください)。

4. MQSTAT への呼び出しが正常に行われると、その前のエラー情報またはカウントはリセットされます。

パラメーター

MQSTAT 呼び出しには、以下のパラメーターがあります。

Hconn (MQHCONN) - 入力

このハンドルは、キュー・マネージャーに対する接続を表します。Hconn の値は、先行の MQCONN または MQCONNX 呼び出しによって戻されたものです。

STYPE (10 桁の符号付き整数) - 入力

要求される状況情報のタイプ。有効な唯一の値は、以下のものです。

STATAPT

以前の非同期 PUT 操作に関する情報を戻します。

STS (MQSTS) - 入出力

状況情報の構造体。詳細は [1240 ページの『IBM i での MQSTS \(状況報告構造体\)』](#) を参照してください。

CMPCOD (10 桁の符号付き整数) - 出力

完了コード。以下のいずれかです。

CCOK

正常終了。

CCFAIL

呼び出し失敗。

REASON (10 桁の符号付き整数) - 出力

CMPCOD を限定する理由コード。

CMPCOD が CCOK の場合

RCNONE

(0, X'000') レポートする理由コードはありません。

CMPCOD が CCFAIL の場合

RC2374

(2374, X'946') API 出口で障害が発生しました

RC2183

(2183, X'887') API 出口をロードできません。

RC2219

(2219, X'8AB') 前の呼び出しが完了する前に MQI 呼び出しが入力されました。

RC2009

(2009, X'7D9') キュー・マネージャーとの接続が失われました。

RC2203

(2203, X'89B') 接続がシャットダウン中です。

RC2018

(2018, X'7E2') 接続ハンドルが無効です。

RC2162

(2162, X'872') キュー・マネージャーは停止しています。

RC2102

(2102, X'836') 使用できるシステム・リソースが不足しています。

RC2430

(2430, X'97E') MQSTAT タイプでエラーが発生しました。

RC2071

(2071, X'817') ストレージが不足しています。

RC2424

(2424, X'978') MQSTS 構造体のエラー

RC2195

(2195, X'893') 予期しないエラーが発生しました。

RC2298

(2298, X'8FA') 要求された関数は、現在の環境では使用できない。

これらのコードの詳細については、以下を参照してください。

- [メッセージとエラーコード](#)

RPG 宣言

```
C*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
C                                     CALLP      MQSTAT(HCONN : ETYPE : ERR :
C                                     CMPCOD : REASON)
```

呼び出しのプロトタイプ定義は次のようになります。

```
D.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
DMQSTAT      PR      EXTPROC('MQSTAT')
D* Connection handle
D HCONN      10I 0 VALUE
D* Status information type
D STYPE      10I 0 VALUE
D* Status information
D STATUS      296A
D* Completion code
D CMPCOD      10I 0
D* Reason code qualifying CompCode
D REASON      10I 0
```

IBM i IBM i での MQSUB (サブスクリプションの登録)

MQSUB 呼び出しは、特定のトピックに対するアプリケーションのサブスクリプションを登録します。

- [1372 ページの『構文』](#)
- [1372 ページの『使用上の注意』](#)
- [1374 ページの『パラメーター』](#)
- [1377 ページの『RPG 宣言』](#)

構文

MQSUB (HCONN, SUBDSC, HOBJ, HSUB, CMPCOD, REASON)

使用上の注意

- サブスクリプションは、事前定義されたトピック・オブジェクトのショート・ネーム、トピック・ストリングのフルネームを使用して命名されるか、[トピック・ストリングの組み合わせ](#)で説明されているように、2つの部分を連結して形成されます。
- キュー・マネージャーは、MQSUB 呼び出しが発行されるときにセキュリティー検査を行い、アクセス許可が出される前に、アプリケーションの実行に使用されるユーザー ID に適切なレベルの権限が付与されていることを確認します。該当するトピック・オブジェクトは、呼び出しで指定されているショート・ネームによって位置指定されます。ロング・ネームが指定されている場合には、検出されたトピック階層における最も近いショート・ネーム・オブジェクトによって位置指定されます。権限検査がこのトピック・オブジェクトに対して行われ、サブスクライブの権限が設定されているか確認します。さらに権限検

査は、宛先キューに対しても行われ、出力に関する権限が設定されているか確認します。SDMAN オプションが使用された場合、これは、権限検査がこのトピック・オブジェクトに関連付けられた管理対象キュー一名に対して行われることを意味します。非管理対象キューが指定された場合、これは、権限検査が **HOB** パラメーターで示されるキューに対して行われることを意味します。

- SOMAN オプションが使用されている場合に MQSUB 呼び出しで返される **HOB** を照会して、バックアウトしきい値や過度のバックアウト再キューの名前などの属性を見つけることができます。管理対象キューの名前も照会できますが、このキューを直接オープンしようとししないでください。
- サブスクリプションをグループ化して、そのグループの複数のサブスクリプションが1つのパブリケーションと一致した場合でもこのパブリケーションのみサブスクリプションのグループに配布できます。サブスクリプションをグループ化するには、SOGRP オプションを使用します。またサブスクリプションをグループ化するには、以下の条件を満たさなければなりません。
 - 同じキュー・マネージャーで同じ名前付きキュー (つまり SOMAN オプションを使用していない) を使用する (これは、MQSUB 呼び出しの **HOB** パラメーターで示される)
 - 同じ **SDCID** を共有する
 - **SDSL** が同じである

これらの属性は、そのグループに含まれると見なされるサブスクリプションのセットを定義します。さらにサブスクリプションがグループ化される場合、これらは変更できない属性です。**SDSL** を変更すると RC2512 が出され、それ以外のいずれか (サブスクリプションがグループ化されていない場合に変更できるもの) を変更すると RC2515 が出されます。

- SORES オプションを使用する MQSUB 呼び出しから戻る際に、MQSD 中のフィールドに値が入れられます。返された MQSD は、サブスクリプションに対して行う必要があるすべての変更を MQSD に適用してから、SOALT オプションを使用する MQSUB 呼び出しに直接渡すことができます。表に示されているように、一部のフィールドには特別な考慮事項があります。

表 791. MQSUB からの MQSD 出力	
MQSD のフィールド名	特別な考慮事項
アクセスまたは作成オプション	MQSUB 呼び出しからの戻り時に、これらのオプションはどれも設定されません。後で MQSUB 呼び出しの中で MQSD を再利用する場合、必要なオプションは明示的に設定する必要があります。
耐久性オプション、宛先オプション、登録オプション、およびワイルドカード・オプション	これらのオプションは、該当する場合に設定されます。
パブリケーション・オプション	これらのオプションは、該当する場合に設定されます。ただし SONEWP は例外で、SOCRE のみに適用できます。
その他のオプション	これらのオプションは、MQSUB 呼び出しからの戻り時に変更されません。これらのオプションは、API 呼び出しが発行される方法を制御し、サブスクリプションと共に格納されません。これらは、MQSD を再利用する後続の MQSUB 呼び出しで必要に応じて設定する必要があります。
ObjectName	この入力専用フィールドは、MQSUB 呼び出しからの戻り時に変更されません。
ObjectString	この入力専用フィールドは、MQSUB 呼び出しからの戻り時に変更されません。バッファが提供されている場合、使用される完全なトピック名が SDRO フィールドで返されます。
AlternateUserId および AlternateSecurityId	これらの入力専用フィールドは、MQSUB 呼び出しからの戻り時に変更されません。これらのオプションは、API 呼び出しが発行される方法を制御し、サブスクリプションと共に格納されません。これらは、MQSD を再利用する後続の MQSUB 呼び出しで必要に応じて設定する必要があります。
SubExpiry	SORES オプションを使用する MQSUB 呼び出しからの戻り時に、このフィールドは、残りの満了時間ではなく、サブスクリプションの元の満了時間に設定されます。その後、SOALT オプションを使用する MQSUB 呼び出しの中で MQSD を再利用すると、サブスクリプションの満了はリセットされ、カウントダウンが再度開始します。
SubName	このフィールドは MQSUB 呼び出しの入力フィールドで、出力時に変更されません。

表 791. MQSUB からの MQSD 出力 (続き)	
MQSD のフィールド名	特別な考慮事項
SubUserData および SelectionString	これらの可変長フィールドは、バッファが提供されていて、さらに VCHRP に正のバッファ長が指定されている場合には、SORES オプションを使用する MQSUB 呼び出しからの出力で返されます。バッファが提供されていない場合には、MQCHARV の VCHRL フィールドで長さだけが返されます。提供されたバッファがフィールドを返すために必要なスペースより小さい場合、提供されたバッファに VCHRP バイト分だけが返されます。 その後で SOALT オプションを使用する MQSUB 呼び出しの中で MQSD を再利用し、バッファが提供されておらず、VCHRL がゼロ以外に設定されている場合、その長さがフィールドの既存の長さとは一致する場合には、フィールドの変更は行われません。
SubCorrelId および PubAccountingToken	SOSCID を使用しない場合、キュー・マネージャーにより SDCID が生成されます。SOSETI を使用しない場合、キュー・マネージャーにより SDACC が生成されます。 これらのフィールドは、SORES オプションを使用する MQSUB 呼び出しからの MQSD で返されます。これらがキュー・マネージャーにより生成された場合、生成値は、SOCRE または SOALT オプションを使用する MQSUB 呼び出しで返されます。
PubPriority, SubLevel & PubApplIdentityData	これらのフィールドは MQSD 中に戻されます。
ResObjectString	この出力専用フィールドは、バッファが提供されている場合には MQSD で返されます。

パラメーター

MQSUB 呼び出しには、以下のパラメーターがあります。

HCONN (10 桁の符号付き整数) - 入力

このハンドルは、キュー・マネージャーに対する接続を表します。HCONN の値は、先行の MQCONN または MQCONNx 呼び出しによって戻されたものです。

SUBDSC (MQSD) - 入出力

これは、アプリケーションによって使用法が登録されているオブジェクトを識別する構造です。詳しくは、[1221 ページの『IBM i での MQSD \(サブスクリプション記述子\)』](#)を参照してください。

HOBJ (10 桁の符号付き整数) - 入出力

このハンドルは、このサブスクリプションに送信されたメッセージを取得するために設定されたアクセスを表します。これらのメッセージを特定のキューに保管することもできますし、それらの保管の管理をキュー・マネージャーに依頼することもできます (後者の場合には特定のキューを必要としません)。

オブジェクト・ハンドル

特定のキューを使用する場合、それは作成時にサブスクリプションに関連付けられる必要があります。次の 2 つの方法でこの作業を行うことができます。

- SDCRT オプションを指定して MQSUB を呼び出すときにこのハンドルを指定する方法。このハンドルが呼び出しの入力パラメーターとして指定される場合、それは、OOINP*、OOOUT (例えばリモート・キューである場合)、または OOBW オプションの少なくとも 1 つを使用したキューに対する前の MQOPEN 呼び出しで返された、有効なオブジェクト・ハンドルである必要があります。そうでない場合、呼び出しは RC2019 で失敗します。このハンドルは、トピック・オブジェクトに解決される別名キューへのオブジェクト・ハンドルにすることはできません。その場合、呼び出しは RC2019 で失敗します。
- DEFINE SUB MQSC コマンドを使用し、そのコマンドにキュー・オブジェクトの名前を提供する。

キュー・マネージャーがこのサブスクリプションに送信されたメッセージの保管を管理するようにする場合、そのことをサブスクリプションの作成時に示す必要があります。これは、SOMAN オプションを

使用し、パラメーター値を HONONE に設定することにより行います。キュー・マネージャーは、呼び出しの出力パラメーターとしてハンドルを返します。返されるハンドルは管理対象ハンドルと呼ばれます。HONONE が指定されても SOMAN は指定されていない場合、呼び出しは RC2019 で失敗します。

キュー・マネージャーにより戻される管理対象ハンドルは、MQGET または MQCB 呼び出し (ブラウズ・オプションありなしの両方)、MQINQ 呼び出し、または MQCLOSE で使用できます。これは MQPUT、MQSET、または後続の MQSUB では使用できません。これらで使用しようとする、MQPUT の場合は RC2039、MQSET の場合は RC2040、MQSUB の場合は RC2038 で失敗します。

MQSD 構造体の OPTS フィールドで SORES オプションを使用してこのサブスクリプションを再開すると、HONONE が指定されている場合には、このパラメーターでハンドルをアプリケーションに返すことができます。このオプションは、サブスクリプションが管理対象ハンドルを使用しているかどうかに関係なく使用できます。サブスクリプション・キューへのハンドルを DEFINE SUB コマンドで定義する場合、DEFINE SUB を使用して作成されたサブスクリプションでこのオプションが役立つ場合があります。管理用に作成されたサブスクリプションが再開される場合には、キューは OOINPQ および OOBWRW でオープンされます。他のオプションが必要である場合、アプリケーションで明示的にサブスクリプション・キューをオープンし、呼び出しでオブジェクト・ハンドルを指定する必要があります。キューのオープンに問題がある場合、呼び出しは RC2522 で失敗します。HOBj が指定された場合、それは元の MQSUB 呼び出しの HOBj と同等でなければなりません。つまり、MQOPEN 呼び出しで返されたオブジェクト・ハンドルが指定される場合、ハンドルは前に使用されたキューと同じキューに対するものでなければなりません。そうでない場合、呼び出しは RC2019 で失敗します。

MQSD 構造体の OPTS フィールドで SOALT オプションを使用することによりこのサブスクリプションが変更される場合、別の HOBj を指定できます。このパラメーターを使用して以前に識別されたキューに送達されたすべてのパブリケーションはそのキューに残るので、HOBj パラメーターが別のキューを示すようになった場合、アプリケーションの側でこれらのメッセージの取得を行う必要があります。

以下の表には、このパラメーターの使用法と、さまざまなサブスクリプション・オプションが要約されています。

表 792. 各種サブスクリプション・オプションでの Hobj の使用		
オプション	Hobj	説明
SOCRT + SOMAN	入力では無視される	キュー・マネージャーでメッセージの保管を管理するサブスクリプションを作成する。
SOCRT	有効なオブジェクト・ハンドル	メッセージの宛先として特定のキューを提供して、サブスクリプションを作成します。
SORES	HONONE	以前に作成されたサブスクリプション (管理対象とそれ以外の両方) を再開し、アプリケーションで使用するためのオブジェクト・ハンドルをキュー・マネージャーが返すようにする。
SORES	有効で一致するオブジェクト・ハンドル	特定のキューをメッセージの宛先として使用する、以前に作成されたサブスクリプションを再開し、オブジェクト・ハンドルを特定のオープン・オプションを指定して使用する。
SOALT + SOMAN	HONONE	以前に特定のキューを使用していた既存のサブスクリプションを管理対象になるように変更する。
SOALT	有効なオブジェクト・ハンドル	既存のサブスクリプションが特定のキューを使用するように変更する (管理対象または別の特定のキューのいずれかから)。

パブリケーションを受け取る以降の MQGET 呼び出しでは、HOBj が提供されたか返されたかにかかわらず、それを指定する必要があります。

HOBj ハンドルは、それに対して MQCLOSE 呼び出しが発行されたとき、またはハンドルの有効範囲を定義する処理の単位が終了したときに、無効になります。戻されるオブジェクト・ハンドルの有効範囲は、呼び出しで指定される接続ハンドルの有効範囲と同じです。ハンドルの有効範囲について詳しくは、[HCONN](#) を参照してください。HOBj ハンドルの MQCLOSE は、HSub ハンドルには影響を与えません。

HSUB (10 桁の符号付き整数) - 出力

このハンドルは、作成されたサブスクリプションを表します。以下の 2 つの後続操作で使用できます。

- 後続の MQSUBRQ 呼び出しで使用して、サブスクリプション作成時に SOPUBR オプションが使用されている場合にパブリケーションを送信するように要求できます。
- 後続の MQCLOSE 呼び出しで使用して、作成されているサブスクリプションを除去できます。HSUB ハンドルは、MQCLOSE 呼び出しが発行されたとき、またはハンドルの有効範囲を定義する処理の単位が終了したときに、無効になります。戻されるオブジェクト・ハンドルの有効範囲は、呼び出しで指定される接続ハンドルの有効範囲と同じです。HSUB ハンドルの MQCLOSE は、HOBJ ハンドルには影響を与えません。

このハンドルを MQGET または MQCB 呼び出しに渡すことはできません。HOBJS パラメーターを使用する必要があります。このハンドルを他の IBM MQ 呼び出しに渡すと、RC2019 になります。

CMPCOD (10 桁の符号付き整数) - 出力

完了コード。以下のいずれかです。

CCOK

正常終了。

CCWARN

警告 (部分完了)

CCFAIL

呼び出し失敗

REASON (10 桁の符号付き整数) - 出力

CMPCOD を限定する理由コード。

CMPCOD が CCOK の場合

RCNONE

(0, X'000') レポートする理由コードはありません。

CMPCOD が CCFAIL の場合

RC2019

(2019 X'07E3') オブジェクト・ハンドルが無効です。

RC2046

(2046 X'07FE') オプションが無効または矛盾しています。

RC2085

(2085 X'0825') 識別されたオブジェクトが見つかりません。

RC2161

(2161 X'0871') キュー・マネージャーが静止しています。

RC2298

(2298 X'08FA') 関数はサポートされていません。

RC2424

(2424 X'0978') サブスクリプション記述子 (MQSD) が無効です。

RC2425

(2441 X'979') トピック・ストリングが無効です。

RC2428

(2428 X'097C') 指定されたサブスクリプション名は既存のサブスクリプションと一致しません。

RC2429

(2429 X'097D') サブスクリプション名が存在し、別のアプリケーションが使用中です。

RC2431

(2431 X'097F') SubUserData フィールドが無効です。

RC2432

(2432 X'0980') サブスクリプションが存在します。

RC2434

(2434 X'0982') サブスクリプション名は既存のサブスクリプションと一致します。

RC2440

(2440 X'0988') SubName フィールドが無効です。

RC2441

(2441 X'0989') Objectstring フィールドが無効です。

RC2435

(2435 X'0983') SDALT を使用して属性を変更できないか、またはサブスクリプションが SDIMM で作成されました。

RC2436

(2436 X'0984') SODUR オプションは無効です。

RC2459

(2459, X'99B') 選択ストリング構文エラー。

RC2503

(2503 X'09C7') 現在 MQSUB 呼び出しはサブスクライブ先のトピックでは使用禁止になっています。

RC2519

(2519, X'9D7') 選択ストリングは、MQCHARV 構造体の使用方法についての記述で指定されているものと異なります。

RC2551

(2551, X'9F7') 指定された選択ストリングを使用できません。

RPG 宣言

```
C*.1.....2.....3.....4.....5.....6.....7..
C                                CALLP      MQSUB(HCONN : SUBDSC : HOBJ :
C                                HSUB : CMPCOD : REASON)
```

呼び出しのプロトタイプ定義は次のようになります。

```
D*.1.....2.....3.....4.....5.....6.....7..
DMQSUB      PR              EXTPROC('MQSUB')
D* Connection handle
D HCONN                      10I 0 VALUE
D* Subscription descriptor
D SUBDSC                      400A
D* Object handle for queue
D HOBJ                      10I 0
D* Subscription object handle
D HSUB                      10I 0
D* Completion code
D CMPCOD                      10I 0
D* Reason code qualifying CompCode
D REASON                      10I 0
```

IBM i IBM i での MQSUBRQ (サブスクリプション要求)

MQSUBRQ 呼び出しはサブスクリプションに対する要求を行います。

- [1377 ページの『構文』](#)
- [1378 ページの『使用上の注意』](#)
- [1378 ページの『パラメーター』](#)
- [1379 ページの『RPG 宣言』](#)

構文

MQSUBRQ (HCONN, HSUB, ACTION, SUBROPT, CMPCOD, REASON)

使用上の注意

次の使用上の注意は、SRAPUB の使用に適用されます。

1. この verb が正常に完了した場合、指定されたサブスクリプションに一致する保存パブリケーションはサブスクリプションへ送信済みであり、サブスクリプションを作成した元の MQSUB verb で戻された HOBJ を使用する MQGET または MQCB を使用して受信できます。
2. サブスクリプションを作成した元の MQSUB verb によってサブスクライブされたトピックにワイルドカードが含まれている場合、複数の保存パブリケーションが送信されることがあります。この呼び出しの結果として送信されたパブリケーションの数は、SBROPT 構造体内の *SRNMP* フィールドに記録されます。
3. この verb が理由コード RC2437 で完了した場合、指定されたトピックには現行の保存パブリケーションはありませんでした。
4. この verb が理由コード RC2525 または RC2526 で完了した場合は、指定されたトピックには現行の保存パブリケーションはありますが、エラーが生じたためそれらが送達不可能であったことになります。
5. この呼び出しを行う前に、アプリケーションにはトピックへの現行のサブスクリプションがなければなりません。サブスクリプションがアプリケーションの以前のインスタンスで作成され、サブスクリプションへの有効なハンドルが使用可能でない場合、アプリケーションはまず SORES オプションで MQSUB を呼び出して、この呼び出しに使用するハンドルを取得する必要があります。
6. パブリケーションは、このアプリケーションの現行のサブスクリプションに使用するために登録された宛先に送信されます。パブリケーションをその他の宛先に送信するには、SOALT オプションで MQSUB 呼び出しを使用してサブスクリプションをまず変更する必要があります。

パラメーター

MQSUBRQ 呼び出しには、以下のパラメーターがあります。

HCONN (10 桁の符号付き整数) - 入力

このハンドルは、キュー・マネージャーに対する接続を表します。*HCONN* の値は、先行の MQCONN または MQCONNX 呼び出しによって戻されたものです。

アプリケーション CICSz/OS では、MQCONN 呼び出しを省略でき、:*HCONN* に以下の値を指定できます：

HCDEFH

デフォルトの接続ハンドル。

HSUB (10 桁の符号付き整数) - 入力

このハンドルは、更新が要求されるサブスクリプションを表します。*HSUB* の値は前の MQSUB 呼び出しで戻されています。

ACTION (10 桁の符号付き整数) - 入力

このパラメーターは、サブスクリプションで要求される特定のアクションを制御します。以下のいずれかを 1 つのみ指定する必要があります。

SRAPUB

このアクションは指定されたトピックに対して更新パブリケーションが送信されることを要求します。これは、通常、サブスクライバーがサブスクリプション作成時に、MQSUB 呼び出しでオプション SOPUBR を指定した場合に使用されます。トピックに関する保存パブリケーションがキュー・マネージャー中にある場合は、サブスクライバーに送信されます。ない場合は、その呼び出しは失敗します。保存されたパブリケーションがアプリケーションに送られると、そのパブリケーションの MQIsRetained メッセージ・プロパティによって示されます。

HSUB パラメーターによって表される既存のサブスクリプション内のトピックにはワイルドカードが含まれる場合があるため、サブスクライバーは複数の保存パブリケーションを受信することがあります。

SBROPT (MQSRO) - 入出力

これらのオプションは、MQSUBRQ のアクションを制御します。詳しくは、[575 ページの『MQSRO - サブスクリプション要求オプション』](#)を参照してください。

CMPCOD (10 桁の符号付き整数) - 出力

完了コード。以下のいずれかです。

CCOK

正常終了。

CCWARN

警告 (部分完了)

CCFAIL

呼び出し失敗

Reason (10 桁の符号付き整数) - 出力

CMPCOD を限定する理由コード。

もし CPMCOD CCOK であるならば：

RCNONE

(0, X'000') レポートする理由コードはありません。

CPMCOD が CCFAIL の場合

RC2298

2298 (X'08FA') 要求された関数は、現在の環境では使用できない。

RC2437

2437 (X'0985') このトピックに関する現在格納中の保存パブリケーションがない。

RC2046

2046 (X'07FE') オプション・パラメーターまたはフィールドに、無効なオプションか、または無効なオプションの組み合わせが含まれている。

RC2161

2161 (X'0871') キュー・マネージャーが静止中

RC2438

2438 (X'0986') MQSUBRQ 呼び出しで、サブスクリプション要求オプション MQSRO が無効である。

RPG 宣言

```
C*.1.....2.....3.....4.....5.....6.....7..
C                                CALLP      MQSUBRQ(HCONN : HSUB : ACTION :
C                                SBROPT : CMPCOD : REASON)
```

呼び出しのプロトタイプ定義は次のようになります。

```
D*.1.....2.....3.....4.....5.....6.....7..
DMQSUBRQ          PR          EXTPROC('MQSUBRQ')
D* Connection handle
D HCONN              10I 0 VALUE
D* Subscription handle
D HSUB              10I 0 VALUE
D* Action requested on the subscription
D ACTION            10I 0 VALUE
D* Subscription Request Options
D SBROPT              16A
D* Completion code
D CMPCOD              10I 0
D* Reason code qualifying CompCode
D REASON              10I 0
```


この一連のトピックでは、MQINQ 関数呼び出しの対象となり得る IBM MQ オブジェクトだけをリストし、照会可能な属性や使用されるセレクターの詳細を示します。

キューの属性

この情報は、様々なタイプのキュー定義と、それぞれのタイプでサポートされている属性について学習するために使用してください。

キューのタイプ: キュー・マネージャーは、以下のタイプのキュー定義をサポートしています。

ローカル・キュー

これは、メッセージを格納する物理キューです。このキューはローカル・キュー・マネージャー上に存在します。

ローカル・キュー・マネージャーに接続しているアプリケーションは、このタイプのキューに対してメッセージを書き込んだり、メッセージを削除したりできます。**QType** キュー属性の値は QTLOC です。

共用キュー

これは、メッセージを格納する物理キューです。このキューは共有リポジトリ内に存在します。その共有リポジトリを所有するキュー共有グループに属するすべてのキュー・マネージャーからアクセス可能です。

キュー共有グループ内のキュー・マネージャーに接続しているアプリケーションは、このタイプのキューに対してメッセージを書き込んだり、メッセージを削除したりできます。このようなキューは、実質的にはローカル・キューと同じです。**QType** キュー属性の値は QTLOC です。

- 共有キューは、z/OS でのみサポートされます。

クラスター・キュー

これは、メッセージを格納する物理キューです。このキューは、ローカル・キュー・マネージャー上か、ローカル・キュー・マネージャーと同じクラスターに所属する 1 つ以上のキュー・マネージャー上の、いずれかに存在します。

ローカル・キュー・マネージャーに接続しているアプリケーションは、キューの場所にかかわらず、このタイプのキューに対してメッセージを書き込んだり、メッセージを削除したりできます。キューのインスタンスがローカル・キュー・マネージャー上にある場合、そのキューの振る舞いはローカル・キューと同じで、ローカル・キュー・マネージャーに接続されたアプリケーションはそのキューからメッセージを削除できます。**QType** キュー属性の値は QTCLUS です。

別名キュー

これは物理キューではなく、ローカル・キューの代替名です。別名を解決したときのローカル・キュー名は、別名キューの定義の一部です。

ローカル・キュー・マネージャーに接続されたアプリケーションは、別名キューにメッセージを置いたり、別名キューからメッセージを削除したりできます。置かれたり削除されたりするメッセージの位置は、別名を解決したときのローカル・キューです。**QType** キュー属性の値は QTALS です。

リモート・キュー

これは物理キューではなく、リモート・キュー・マネージャー上に存在するキューのローカル定義です。リモート・キューのローカル定義には、リモート・キュー・マネージャーにメッセージを経路指定する方法を、ローカル・キュー・マネージャーに示す情報が入っています。

ローカル・キュー・マネージャーに接続されたアプリケーションは、リモート・キューにメッセージを置くことができます。置かれるメッセージの位置は、メッセージをリモート・キュー・マネージャーに送付するために使用されるローカル伝送キューです。アプリケーションは、リモート・キューからメッセージを除去することはできません。**QType** キュー属性の値は QTREM です。

リモート・キュー定義は、以下の目的にも使用できます。

- 応答キューへの別名割り当て

この場合、定義の名前は、応答先のキューの名前です。詳しくは、[応答先キュー別名定義](#)を参照してください。

- キュー・マネージャーの別名割り当て

この場合、定義の名前は、キューの名前ではなく キュー・マネージャーの別名です。詳しくは、[キュー・マネージャー別名定義](#)を参照してください。

モデル・キュー

これは、物理的なキューではありません。キュー属性の 1 セットであり、これを基にしてローカル・キューを作成することができます。

このタイプのキューにはメッセージを保管できません。

キュー属性には、すべてのタイプのキューに適用される属性と、特定のタイプのキューにのみ適用される属性があります。属性が適用されるキューのタイプは、[1381 ページの表 793](#) 以降の表では「X」で示されています。

[1381 ページの表 793](#) には、キューに固有の属性がまとめられています。属性の説明は、アルファベット順に掲載しています。

この表で示されている属性の名前は、MQINQ 呼び出しおよび MQSET 呼び出しで使用する名前です。MQSC コマンドを使用して属性を定義、変更、または表示するときには、代替の短縮名が使用されます。詳細については、[MQSC コマンド](#)を参照してください。

以下の表では、列は次のように適用されます。

- ローカル・キューの列は、共有キューにも適用されます。
- モデル・キューの列は、そのモデル・キューを基にして作成された ローカル・キューに継承される属性を示しています。
- クラスター・キューの列は、照会のみ、または照会と出力を目的としてクラスター・キューがオープンされた場合に照会できる属性を示しています。照会に加えて、入力、ブラウズ、または設定のうち 1 つ以上の作業を目的としてクラスター・キューがオープンされた場合は、代わりにローカル・キューの列が適用されます。

表 793. キューの属性						
属性	説明	ローカル	モデル	別名	リモート	クラスター
AlterationDate	定義が最後に変更された日付	X		X	X	
AlterationTime	定義が最後に変更された時刻	X		X	X	
BackoutRequeueQName	超過バックアウト再キューイング用のキュー名	X	X			
BackoutThreshold	バックアウトしきい値	X	X			
BaseQName	別名が解決されるキュー名			X		
ClusterChannelName	クラスター送信側チャネル名	✓	✓			
ClusterName	キューが属するクラスターの名前	X		X	X	
ClusterNamelist	キューが属するクラスターの名前を含む名前リスト・オブジェクトの名前	X		X	X	

表 793. キューの属性 (続き)

属性	説明	ローカル	モデル	別名	リモート	クラスター
CreationDate	キューが作成された日付	X				
CreationTime	キューが作成された時刻	X				
CurrentQDepth	現行キュー項目数	X				
DefBind	デフォルトのバインド	X		X	X	X
DefinitionType	キュー定義タイプ	X	X			
DefInputOpenOption	デフォルト入力オープン・オプション	X	X			
DefPersistence	デフォルトのメッセージ持続性	X	X	X	X	X
DefPriority	デフォルトのメッセージ優先順位	X	X	X	X	X
DistLists	配布リスト・サポート	X	X			
HardenGetBackout	正確なバックアウト・カウントを維持するかどうか	X	X			
InhibitGet	このキューに対する読み取り操作が可能かどうかを制御する	X	X	X		
InhibitPut	このキューに対する書き込み操作が可能かどうかを制御する	X	X	X	X	X
InitiationQName	開始キューの名前	X	X			
MaxMsgLength	メッセージの最大長 (バイト数)。	X	X			
MaxQDepth	キューの最大長	X	X			
MediaLog	指定されたキューのメディア・リカバリーに必要な最も古いログ・エクステンツ (IBM i では最も古いジャーナル・レシーバー) の ID	✓	✓			
MsgDeliverySequence	メッセージ・デリバリー・シーケンス	X	X			
OpenInputCount	入力のためのオープンの回数	X				
OpenOutputCount	出力のためのオープンの回数	X				
ProcessName	プロセス名	X	X			

表 793. キューの属性 (続き)						
属性	説明	ローカル	モデル	別名	リモート	クラスター
QDepthHighEvent	キュー・サイズ上限イベントが生成されるかどうかを制御する	X	X			
QDepthHighLimit	キュー・サイズの上限	X	X			
QDepthLowEvent	キュー・サイズ下限イベントが生成されるかどうかを制御する	X	X			
QDepthLowLimit	キュー・サイズの下限	X	X			
QDepthMaxEvent	「キュー・フル」イベントを生成するかどうかを制御します	X	X			
QDesc	キューの記述	X	X	X	X	X
QName	キュー名	X		X	X	X
QServiceInterval	キュー・サービス間隔の目標値	X	X			
QServiceIntervalEvent	サービス間隔上限イベントまたはサービス間隔 OK イベントを生成するかどうかを制御	X	X			
QTYPE	キュー・タイプ	X		X	X	X
RemoteQMgrName	リモート・キュー・マネージャーの名前				X	
RemoteQName	リモート・キューの名前				X	
RetentionInterval	保存間隔	X	X			
SCOPE	キューの項目がセル・ディレクトリーにも存在するかどうかを制御	X		X	X	
Shareability	キューの共有可能性	X	X			
TriggerControl	トリガー制御	X	X			
TriggerData	トリガー・データ	X	X			
TriggerDepth	トリガー項目数	X	X			
TriggerMsgPriority	トリガーのしきい値メッセージ優先順位	X	X			
TriggerType	トリガー・タイプ	X	X			
USAGE	キューの使用法	X	X			
XmitQName	伝送キュー名				X	

 **IBM i での AlterationDate (12 バイトの文字ストリング)**
定義が最後に変更された日付。

表 794. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X		X	X	

これは、定義を最後に変更した日付です。この日付の形式は YYYY-MM-DD であり、長さを 12 バイトにするために、末尾に空白を 2 つ埋め込みます (例えば 1992-09-23 のようになり、 は 2 つの空白文字を示しています)。

特定の属性の値 (例えば、*CurrentQDepth*) はキュー・マネージャーが作動すると変更されます。これらの属性を変更しても、*AlterationDate* には影響しません。

この属性の値を判別するには、MQINQ 呼び出しで CAALTD セレクターを使用します。この属性の長さは LNDATE で指定します。

IBM i IBM i での *AlterationTime* (8 バイトの文字ストリング)

定義が最後に変更された時刻。

表 795. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X		X	X	

これは、定義を最後に変更した時刻です。時刻の形式は HH.MM.SS です。24 時間時計を使用し、10 時未満の場合は先頭にゼロを付けます (例えば、09.10.20)。時刻はローカル時間です。

特定の属性の値 (例えば、*CurrentQDepth*) はキュー・マネージャーが作動すると変更されます。これらの属性を変更しても、*AlterationTime* には影響しません。

この属性の値を判別するには、MQINQ 呼び出しで CAALTT セレクターを使用します。この属性の長さは LNTIME で指定します。

IBM i IBM i での *BackoutRequeueQName* (48 バイトの文字ストリング)

超過バックアウト再キューイング用のキュー名。

表 796. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

WebSphere Application Server 内部で実行しているアプリケーション、および IBM MQ Application Server Facilities を使用するアプリケーションは、この属性を使用して、バックアウト済みメッセージの宛先を判別します。その他のすべてのアプリケーションでは、キュー・マネージャーは、この属性値を照会できるようにする以外には、この属性の値に基づいてアクションを取ることはありません。

この属性の値を判別するには、MQINQ 呼び出しで CABRQN セレクターを使用します。この属性の長さは LNQN で指定します。

IBM i IBM i での *BackoutThreshold* (10 桁の符号付き整数)

バックアウトしきい値。

表 797. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

WebSphere Application Server 内部で実行されているアプリケーション、および IBM MQ Application Server Facilities を使用するアプリケーションは、この属性を使用して、メッセージをバックアウトすべきかどうかを判別します。その他のすべてのアプリケーションでは、キュー・マネージャーは、この属性値を照会できるようにする以外には、この属性の値に基づいてアクションを取ることはありません。

この属性の値を判別するには、MQINQ 呼び出しで IABTHR セレクターを使用します。

▶ IBM i IBM i での BaseQName (48 バイトの文字ストリング)

別名が変換されるキュー名。

表 798. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
		X		

ローカル・キュー・マネージャーに対して定義されるキューの名前です。キュー名の詳細については、MQOD の ODON フィールドを参照してください。キューは次のいずれかです。

QTLOC

ローカル・キュー。

QTREM

リモート・キューのローカル定義。

QTCLUS

クラスター・キュー。

この属性の値を判別するには、MQINQ 呼び出しで CABASQ セレクターを使用します。この属性の長さは LNQN で指定します。

▶ IBM i IBM i での BaseType (整数パラメーター構造)

別名の解決先のオブジェクトのタイプ。

表 799. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
		X		

この属性には、以下のいずれかの値を使用できます。

OTQ

基本オブジェクト・タイプはキューです。

OTTOP

基本オブジェクト・タイプはトピックです。

▶ IBM i IBM i での CFStrucName (12 バイトの文字ストリング)

カップリング・ファシリティ構造体名。

表 800. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

これは、キュー上のメッセージが保管されるカップリング・ファシリティ構造体の名前です。名前の最初の文字は A から Z までの範囲にあり、残りの文字は A から Z まで、0 から 9 まで、またはブランクの範囲にあります。

カップリング・ファシリティの構造体の完全な名前は、**QSGName** キュー・マネージャー属性の値に接尾部として **CFStrucName** キュー属性の値を付けたものです。

この属性は共有キューにのみ適用されます。*QSGDisp* に値 *QSGDSH* がない場合、この属性は無視されます。

この属性の値を判別するには、MQINQ 呼び出しで CACFSN セレクターを使用します。この属性の長さは LNCFSN で指定します。

 この属性は、z/OS でのみサポートされます。

ClusterChannelName (20 バイトの文字ストリング)

ClusterChannelName このキューを送信キューとして使用するクラスター送信チャネルの総称です。この属性は、このクラスター伝送キューからクラスター受信側チャネルにメッセージを送信するクラスター送信側チャネルを指定します。

表 801. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

デフォルトのキュー・マネージャー構成では、すべてのクラスター送信側チャネルがメッセージを単一の伝送キュー **SYSTEM.CLUSTER.TRANSMIT.QUEUE** から送信します。デフォルト構成は、キュー・マネージャー属性 **DefClusterXmitQueueType** を変更することによって変更できます。属性のデフォルト値は **SCTQ** です。この値は **CHANNEL** に変更できます。**DefClusterXmitQueueType** 属性を **CHANNEL** に設定すると、各クラスター送信側チャネルは、デフォルトで特定のクラスター伝送キュー **SYSTEM.CLUSTER.TRANSMIT.ChannelName** を使用するようになります。

また、伝送キュー属性である **ClusterChannelName** 属性をクラスター送信側チャネルに手動で設定することもできます。クラスター送信側チャネルによって接続されたキュー・マネージャーを宛先とするメッセージは、クラスター送信側チャネルを識別する伝送キューに保管されます。これらのメッセージがデフォルトのクラスター伝送キューに保管されることはありません。**ClusterChannelName** 属性をブランクに設定すると、チャネルの再始動時に、チャネルはデフォルトのクラスター伝送キューに切り替わります。デフォルト・キューは、キュー・マネージャーの **DefClusterXmitQueueType** 属性の値に応じて、**SYSTEM.CLUSTER.TRANSMIT.ChannelName** または **SYSTEM.CLUSTER.TRANSMIT.QUEUE** のどちらかになります。

アスタリスク "*" を **ClusterChannelName** に指定することにより、伝送キューをクラスター送信側チャネルのセットに関連付けることができます。アスタリスクはチャネル名ストリングの先頭、末尾、またはそれ以外の場所に任意の数だけ使用できます。**ClusterChannelName** は長さ 20 文字に制限されています: **MQ_CHANNEL_NAME_LENGTH**。

IBM i での ClusterName (48 バイトの文字ストリング)

キューが属するクラスターの名前。

表 802. この属性が適用されるキュー・タイプ

ローカル	モデル	別名	リモート	クラスター
X		X	X	

これは、キューが属するクラスターの名前です。キューが複数のクラスターに属している場合、*ClusterNameList* はクラスターを識別する名前リスト・オブジェクトの名前を指定し、*ClusterName* はブランクになります。*ClusterName* および *ClusterNameList* の少なくとも 1 つがブランクでなければなりません。

この属性の値を判別するには、MQINQ 呼び出しで CACLN セレクターを使用します。この属性の長さは LNCLUN で指定します。

IBM i IBM i での *ClusterNameList* (48 バイトの文字ストリング)

キューが属するクラスターの名前を含む名前リスト・オブジェクトの名前。

表 803. この属性が適用されるキュー・タイプ

ローカル	モデル	別名	リモート	クラスター
X		X	X	

これは、このキューが属するクラスターの名前を含む名前リスト・オブジェクトの名前です。キューが 1 つのクラスターのみ属する場合、名前リスト・オブジェクトには 1 つの名前のみ含まれます。代わりに *ClusterName* を使用して、そのクラスターの名前を指定できます。この場合、*ClusterNameList* はブランクにします。*ClusterName* および *ClusterNameList* の少なくとも 1 つがブランクでなければなりません。

この属性の値を判別するには、MQINQ 呼び出しで CACLNL セレクターを使用します。この属性の長さは LNNLN によって指定されます。

IBM i IBM i での *CreationDate* (12 バイトの文字ストリング)

キューが作成された日付。

表 804. この属性が適用されるキュー・タイプ

ローカル	モデル	別名	リモート	クラスター
X				

キューが作成された日付です。この日付の形式は YYYY-MM-DD であり、長さを 12 バイトにするために、末尾にブランクを 2 つ埋め込みます (例えば 1992-09-23-- のようになり、-- は 2 つのブランク文字を示しています)。

- IBM i では、キューの作成日付が、オペレーティング・システムの中でキューを表すエンティティ (ファイルまたはユーザー・スペース) の日付と異なる場合があります。

この属性の値を判別するには、MQINQ 呼び出しで CACRTD セレクターを使用します。この属性の長さは、LNCRTD で指定されます。

IBM i IBM i での *CreationTime* (8 バイトの文字ストリング)

キューが作成された時刻。

表 805. この属性が適用されるキュー・タイプ

ローカル	モデル	別名	リモート	クラスター
X				

キューが作成された時刻です。時刻の形式は HH.MM.SS です。24 時間時計を使用し、10 時未満の場合は先頭にゼロを付けます (例えば、09.10.20)。時刻はローカル時間です。

- IBM i では、キューの作成時刻が、そのキューを表す基礎のオペレーティング・システム・エンティティ (ファイルまたはユーザー・スペース) の作成時刻と異なる場合があります。

この属性の値を判別するには、MQINQ 呼び出しで CACRTT セレクターを使用します。この属性の長さは、LNCRTT で指定されます。

IBM i IBM i での CurrentQDepth (10 桁の符号付き整数)

キューの現行サイズ。

表 806. この属性が適用されるキュー・タイプ

ローカル	モデル	別名	リモート	クラスター
X				

キューに現在入っているメッセージの数。この値は、MQPUT 呼び出しが行われたとき、および MQGET 呼び出しのバックアウトが行われたときに増加します。また、ブラウズのない MQGET 呼び出しが行われたとき、および MQPUT 呼び出しのバックアウトが行われたときに減少します。このカウントには、1 つの作業単位でそのキューに書き込まれたメッセージのうち、MQGET 呼び出しによる取り出しの対象でないものも含めて、まだコミットされていないメッセージの数が含まれることになります。同様に、1 つの作業単位で MQGET 呼び出しによって取り出されたメッセージは、まだコミットされる必要のあるものも、このカウントから除かれることになります。

また、このカウントには、満了期間を過ぎてもまだ廃棄されていない、取り出し対象以外のメッセージも含まれます。1114 ページの『IBM i での MQMD (メッセージ記述子)』の MDEXP フィールドを参照してください。

作業単位処理とメッセージのセグメンテーションの両方が原因で、CurrentQDepth が MaxQDepth を超えることがあります。ただし、このことによってメッセージの検索に影響があるわけではありません。キューにあるすべてのメッセージは、MQGET 呼び出しを使用して通常の方法で検索できます。

属性の値は、キュー・マネージャーの動作に応じて変動します。

この属性の値を判別するには、MQINQ 呼び出しで IACDEP セレクターを使用します。

IBM i IBM i での DefBind (10 桁の符号付き整数)

デフォルトのバインディング。

表 807. この属性が適用されるキュー・タイプ

ローカル	モデル	別名	リモート	クラスター
X		X	X	X

この属性は、MQOPEN 呼び出しで OOBNDQ を指定するときに使用されるデフォルト・バインディングであり、キューはクラスター・キューです。DefBind は、以下の値のうちのいずれかをとることができます。

BNDOPN

MQOPEN 呼び出しで固定されたバインディング。

BNDNOT

固定されていないバインディング。

BNDGRP

バインディングは、MQOPEN 呼び出しでは固定されないが、論理グループのすべてのメッセージに対して、MQPUT で固定されている。

この属性の値を判別するには、MQINQ 呼び出しで IADBND セレクターを使用します。

IBM i IBM i での DefinitionType (10 桁の符号付き整数)

キュー定義タイプ。

表 808. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

これは、キューの定義方法を示します。値は、次のいずれか 1 つです。

QDPRE

事前定義された永続キュー。

システム管理者に作成される永続キューであり、システム管理者だけがそのキューを削除できます。

事前定義されたキューが DEFINE MQSC コマンドを使用して作成された場合、このキューを削除するには、DELETE MQSC コマンドしか使用できません。事前定義されたキューは、モデル・キューからは作成できません。

コマンドは、オペレーターが発行する場合と、コマンド入力キューにコマンド・メッセージを送信する許可ユーザーが発行する場合があります (1412 ページの『IBM i でのキュー・マネージャーの属性』で説明されている **CommandInputQName** 属性を参照してください)。

QDPERM

動的に定義された永続キュー。

オブジェクト記述子 MQOD 内に指定されているモデル・キューの名前を指定して MQOPEN 呼び出しを発行しているアプリケーションによって作成された永続キューです。モデル・キュー定義には、**DefinitionType** 属性の値 QDPERM があります。

このタイプのキューは、MQCLOSE 呼び出しを使用して削除できます。詳しくは、[1278 ページの『IBM i での MQCLOSE \(オブジェクトのクローズ\)』](#)を参照してください。

永続動的キューの **QSGDisp** 属性の値は QSGDQM です。

QDTEMP

動的に定義された一時キュー。

オブジェクト記述子 MQOD 内に指定されているモデル・キューの名前を指定して MQOPEN 呼び出しを発行しているアプリケーションによって作成された一時キューです。モデル・キュー定義には、**DefinitionType** 属性の値 QDTEMP があります。

このタイプのキューは、これを作成したアプリケーションによってクローズされるときに、MQCLOSE 呼び出しで自動的に削除されます。

一時動的キューの **QSGDisp** 属性の値は QSGDQM です。

QDSHAR

動的に定義された共有キュー。

このキューは、オブジェクト記述子 MQOD で指定されているモデル・キューの名前を指定して MQOPEN 呼び出しを発行したアプリケーションによって作成された共有永続キューです。モデル・キュー定義には、**DefinitionType** 属性の値 QDSHAR があります。

このタイプのキューは、MQCLOSE 呼び出しを使用して削除できます。詳しくは、[1278 ページの『IBM iでの MQCLOSE \(オブジェクトのクローズ\)』](#)を参照してください。

共用動的キューの **QSGDisp** 属性の値は QSGDSH です。

モデル・キューは常に事前定義されているため、モデル・キュー定義内のこの属性には、モデル・キューの定義方法は示されていません。代わりに、モデル・キュー内のこの属性の値は、MQOPEN 呼び出しを使用してモデル・キュー定義から作成された各動的キューの *DefinitionType* を決定するために使用されます。

この属性の値を判別するには、MQINQ 呼び出しで IADEFT セレクターを使用します。

IBM i IBM i での DefInputOpenOption (10 桁の符号付き整数)

入力に関するデフォルトのオープン・オプション。

表 809. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

これは、キューを入力用にオープンするときのデフォルトの方法です。これが適用されるのは、キューのオープン時に、MQOPEN 呼び出しに OOINPQ オプションが指定されている場合です。これは、以下の値のいずれかになります。

OOINPX

メッセージを読み取るためにキューを排他アクセス・モードでオープンする。

後続の MQGET 呼び出しで使用するために、キューが開かれます。キューが、実行中のアプリケーションまたは他のアプリケーションによって、いずれかのタイプ (OOINPS または OOIINPX) の入力用に現在オープンされている場合、理由コード RC2042 で呼び出しは失敗します。

OOINPS

共有アクセスによりメッセージを読み取るためにキューをオープンする。

後続の MQGET 呼び出しで使用するために、キューが開かれます。キューが、実行中のアプリケーションまたは他のアプリケーションによって OOIINPS を指定して現在オープンされている場合は、この呼び出しは成功しますが、キューが OOIINPX を指定して現在オープンされている場合は、理由コード RC2042 で失敗します。

この属性の値を判別するには、MQINQ 呼び出しで IADINP セレクターを使用します。

IBM i IBM i での DefPersistence (10 桁の符号付き整数)

デフォルトのメッセージ持続性。

表 810. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X	X	X	X

これは、キューのメッセージのデフォルト持続性です。これが適用されるのは、メッセージが書き込まれているときにメッセージ記述子内に PEQDEF が指定されている場合です。

キュー名解決パス内に定義が 2 つ以上ある場合、デフォルトの持続性は、MQPUT または MQPUT1 呼び出しが行われる時に、パス内にある複数の定義のうちの最初の定義 (これがキュー・マネージャーの別名であっても) 内の属性値からとられます。これには以下のものが考えられます。

- 別名キュー
- ローカル・キュー

- リモート・キューのローカル定義
- キュー・マネージャーの別名
- 伝送キュー (例えば、*DefXmitQName* キューなど)

これは、以下の値のいずれかになります。

PEPER

メッセージは持続します。

これは、システム障害が発生して キュー・マネージャーを再始動してもメッセージが残ることを意味します。持続メッセージを以下に書き込むことはできません。

- 一時動的キュー
- 共有キュー

持続メッセージは、永続動的キュー、および事前定義のキューに書き込むことができます。

PENPER

メッセージは持続しません。

これは、システム障害が発生してキュー・マネージャーを再始動すると、通常はメッセージが残らないことを意味します。これは、キュー・マネージャーの再始動中にメッセージの完全なコピーが補助記憶域で 見つかった場合でも、適用されます。

共有キューの特殊な事例として、キュー共有グループでキュー・マネージャーを再始動しても非永続メッセージが残る が、共有キューでメッセージの保管に使用したカップリング・ファシリティに障害が発生するとメッセージは残らないという場合があります。

持続メッセージと非持続メッセージが同一キューにあっても構いません。

この属性の値を判別するには、MQINQ 呼び出しで IADPER セレクターを使用します。

IBM i IBM i での *DefPriority* (10 桁の符号付き整数)

デフォルトのメッセージ優先度。

表 811. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X	X	X	X

これは、キューのメッセージのデフォルト優先順位です。これが適用されるのは、メッセージがキューに書き込まれているときにメッセージ記述子内に PRQDEF が指定されている場合です。

キュー名解決パス内に定義が 2 つ以上ある場合、そのメッセージに関するデフォルト優先順位は、PUT 操作時にパス内にある複数の定義のうちの最初の 定義内の属性値からとられます。これには以下のものが考えられます。

- 別名キュー
- ローカル・キュー
- リモート・キューのローカル定義
- キュー・マネージャーの別名
- 伝送キュー (例えば、*DefXmitQName* キューなど)

メッセージがキューに入れられる方法は、そのキューの **MsgDeliverySequence** 属性の値によって異なります。

- **MsgDeliverySequence** 属性が MSPRIO であれば、メッセージがキューに入れられる論理位置は、メッセージ記述子内の *MDPRI* フィールドによって決まります。
- **MsgDeliverySequence** 属性が MSFIFO であれば、メッセージは、メッセージ記述子内の *MDPRI* フィールドの値にかかわらず、解決済みキューの *DefPriority* に等しい優先順位を持つように扱われてキ

キューに入れます。ただし、メッセージを書き込むアプリケーションで指定した *MDPRI* フィールドの値はそのまま保存されます。詳細については、1380 ページの『キューの属性』の **MsgDeliverySequence** 属性を参照してください。

優先順位は、ゼロ (最下位) から *MaxPriority* (最上位) の範囲にあります。1412 ページの『IBM i でのキュー・マネージャーの属性』の **MaxPriority** 属性を参照してください。

この属性の値を判別するには、MQINQ 呼び出しで IADPRI セレクターを使用します。

IBM i IBM i での DefReadAhead (10 桁の符号付き整数)

クライアントに配信される非持続メッセージのデフォルトの先読み動作を指定します。

表 812. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X	X		

DefReadAhead は、以下の値のいずれか 1 つに設定できます。

RAHNO

非持続メッセージは、アプリケーションが要求する前にクライアントに送信されません。クライアントが異常終了した場合に失われる非持続メッセージは、最大で 1 つだけです。

RAHYES

非持続メッセージは、アプリケーションで要求される前にクライアントに送信されます。クライアントが異常終了した場合、またはクライアントに送信されたすべてのメッセージをクライアントが消費しない場合に、非持続メッセージは失われることがあります。

RAHDIS

このキューに対して、非持続メッセージの先読みは有効になりません。クライアント・アプリケーションによって先読みが要求されているかどうかに関わりなく、メッセージはクライアントに前もって送信されません。

この属性値を調べるには、MQINQ 呼び出しで IADRAH セレクターを使用します。

IBM i IBM i での DefPResp (10 桁の符号付き整数)

デフォルトの書き込み応答タイプ (DEFPRESP) 属性は、MQPMO 内の PutResponseType が PMRASQ に設定されている場合に、アプリケーションにより使用される値を定義します。この属性は、すべてのキュー・タイプに有効です。

表 813. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X	X	X	X

これは、以下の値のいずれかになります。

SYNC

PUT 操作は同期的に実行され、応答が返されます。

ASYN

PUT 操作は非同期的に実行され、MQMD フィールドのサブセットが返されます。

この属性の値を判別するには、MQINQ 呼び出しで IADPRT セレクターを使用します。

IBM i IBM i での DistLists (10 桁の符号付き整数)

配布リスト・サポート。

表 814. この属性が適用されるキュー・タイプ

ローカル	モデル	別名	リモート	クラスター
X	X			

これは、配布リストのメッセージをキューに書き込むことができるかどうかを示します。この属性はメッセージ・チャンネル・エージェント (MCA) によって設定され、ローカル・キュー・マネージャーに、チャンネルのもう一方のキュー・マネージャーが配布リストをサポートしているかどうかを通知します。後者のキュー・マネージャー (「パートナー・キュー・マネージャー」という) は、送信 MCA によってローカル伝送キューから削除されると、次にメッセージを受信するキュー・マネージャーです。

属性はパートナー・キュー・マネージャーで受信 MCA との接続を確立するたびに、送信 MCA によって設定されます。この方法では、ローカル・キュー・マネージャーは送信 MCA によって、パートナー・キュー・マネージャーが正しく処理できるメッセージのみを送信キューに置くようになります。

この属性の本来の使用方法では、伝送キューと共に使用しますが、説明にある処理は伝送キューに定義された使用方法とは関係なく行われます (**Usage** 属性を参照してください)。

これは、以下の値のいずれかになります。

DLSUPP

配布リストがサポートされています。

これは、配布リストのメッセージがキューに格納でき、その形でパートナー・キュー・マネージャーに転送できることを示しています。これにより、複数の宛先に送信するときの処理量が少なくなります。

DLNSUP

配布リストはサポートされていません。

これは、配布リストのメッセージがキューに格納できないことを示しています。パートナー・キュー・マネージャーで配布リストがサポートしていないためです。アプリケーションによって書き込まれた配布リストのメッセージがこのキューに置かれると、キュー・マネージャーは配布リストのメッセージを分割し、個々のメッセージをキューに書き込みます。これにより、複数の宛先にメッセージを送信するときの処理量は増えますが、メッセージはパートナー・キュー・マネージャーによって正しく処理されるようになります。

この属性の値を判別するには、MQINQ 呼び出しで IADIST セレクターを使用します。この属性の値を変更するには、MQSET 呼び出しを使用します。

IBM i IBM i での HardenGetBackout (10 桁の符号付き整数)

正確なバックアウト・カウントを保守するかどうかを制御する。

表 815. この属性が適用されるキュー・タイプ

ローカル	モデル	別名	リモート	クラスター
X	X			

メッセージごとに、そのメッセージが 1 つの作業単位で MQGET 呼び出しによって取り出される回数、つまりその作業単位内で続いてバックアウトされる回数が保守されます。このカウントは、MQGET 呼び出しが完了したあとで、メッセージ記述子内の MDBOC フィールドに入れられます。

メッセージのバックアウト・カウントは、キュー・マネージャーの再始動後も存続します。ただし、カウントの正確さを期するためには、このキューに関する作業単位内でメッセージが MQGET 呼び出しによって取り出されるたびに、情報を「ハード化しておく」(ディスクまたは他の永続記憶装置上に記録する) 必要があります。これを行わないと、キュー・マネージャーの障害が MQGET 呼び出しのバックアウトを伴って発生した場合に、カウントは増加しない場合があります。

ただし、1 作業単位内で MQGET 呼び出しごとに情報をハード化することには、パフォーマンス・コストがかかり、カウントが正確である必要がある場合のみ、**HardenGetBackout** 属性を QABH に設定しなければなりません。

- IBM i では、この属性の設定値とは無関係に、メッセージのバックアウト・カウントが常にハード化されます。

属性の値は以下のとおりです。

QABH

バックアウト・カウントが保管される。

このキューのメッセージのバックアウト・カウントを正確にするために、ハード化が行われます。

QABNH

バックアウト・カウントは記憶されないことがあります。

このキューのメッセージのバックアウト・カウントを正確にするためのハード化は行われません。したがって、カウントが正しい値よりも小さくなる可能性があります。

この属性の値を判別するには、MQINQ 呼び出しで IAHGB セレクターを使用します。

IBM i IBM i での InhibitGet (10桁の符号付き整数)

このキューのための読み取り操作を認めるかどうかを制御する。

表 816. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X	X		

キューが別名キューである場合、MQGET 呼び出しが成功するためには、取得操作時に別名キューと基本キューの両方に対して取得操作が許可されている必要があります。値は、次のいずれか 1 つです。

QAGETI

取得操作は禁止されています。

MQGET 呼び出しは理由コード RC2016 で失敗します。これには、GMBRWF または GMBRWN を指定する MQGET 呼び出しが含まれます。

注：作業単位内で操作中の MQGET 呼び出しが正常に完了した場合は、**InhibitGet** 属性の値を後で QAGETI に変更しても、作業単位のコミットが妨げられることはありません。

QAGETA

取得操作は許可されています。

この属性の値を判別するには、MQINQ 呼び出しで IAIGET セレクターを使用します。この属性の値を変更するには、MQSET 呼び出しを使用します。

IBM i IBM i での InhibitPut (10桁の符号付き整数)

このキューに対する書き込み操作を認めるかどうかを制御する。

表 817. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X	X	X	X

キュー名解決パス内に定義が複数ある場合、MQPUT 呼び出しまたは MQPUT1 呼び出しが成功するためには、PUT 操作時にそのパス内のすべての定義 (キュー・マネージャーの別名定義を含む) に対して PUT 操作が許可されていなければなりません。これは、以下の値のいずれかになります。

QAPUTI

書き込み操作は使用禁止です。

MQPUT 呼び出しおよび MQPUT1 呼び出しは理由コード RC2051 で失敗します。

注：作業単位内で操作中の MQPUT 呼び出しが正常に完了した場合は、**InhibitPut** 属性の値を後で QAPUTI に変更しても、作業単位のコミットが妨げられることはありません。

QAPUTA

書き込み操作が許可されています。

この属性の値を判別するには、MQINQ 呼び出しで IAIPUT セレクターを使用します。この属性の値を変更するには、MQSET 呼び出しを使用します。

IBM i IBM i での InitiationQName (48 バイトの文字ストリング)

開始キューの名前。

表 818. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

これは、ローカル・キュー・マネージャーで定義されているキューの名前です。そのキューのタイプは QTLOC でなければなりません。この属性を持つキューにメッセージが到着したことにより、アプリケーションの始動が必要になった場合、キュー・マネージャーは、トリガー・メッセージを開始キューに送ります。トリガー・メッセージを受け取ったあとで該当するアプリケーションを開始させるトリガー・モニター・アプリケーションによって、開始キューを監視する必要があります。

この属性の値を判別するには、MQINQ 呼び出しで CAINIQ セレクターを使用します。この属性の長さは LNQN で指定します。

IBM i IBM i での MaxMsgLength (10 桁の符号付き整数)

メッセージの最大長 (バイト数)。

表 819. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

これは、キューに入れることができる最長の物理メッセージの長さの上限です。しかし、**MaxMsgLength** キュー属性は、**MaxMsgLength** キュー・マネージャー属性とは別に設定できるため、キューに置くことのできる物理メッセージの実際の最大長は、これら 2 つの値のうちの小さい方の値になります。

キュー・マネージャーがセグメント化をサポートしている場合は、アプリケーションで MFSEGA フラグを MQMD に指定しているときに限って、これら 2 つの **MaxMsgLength** 属性の小さい方の値より長い論理メッセージをアプリケーションから書き込むことができます。そのフラグを指定した場合、論理メッセージの長さの上限は 999 999 999 バイトですが、オペレーティング・システムによって、またはアプリケーションが実行されている環境によってリソースが制約される結果、通常、これより小さい値になります。

キューに置こうとするメッセージが長すぎると失敗し、以下の理由コードが戻ります。

- RC2030。これは、メッセージがキューにとって長すぎる場合に戻ります。
- RC2031。これは、メッセージがキュー・マネージャーにとって長すぎ、キューにとっては長すぎない場合に戻ります。

MaxMsgLength 属性の下限は、ゼロです。上限は、環境により決定されます。

- IBM i の場合、最大メッセージ長は 100 MB (104 857 600 バイト) です。

詳細については、[1344 ページの『IBM i での MQPUT \(メッセージの書き込み\)』](#)の **BUFLEN** パラメーターを参照してください。

この属性の値を判別するには、MQINQ 呼び出しで IAMLEN セレクターを使用します。

IBM i IBM i での MaxQDepth (10 桁の符号付き整数)

キューの最大長。

表 820. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

これは、一度にキューに入れることのできる物理メッセージの数の上限を定義します。すでに *MaxQDepth* 個のメッセージが入っているキューにメッセージを書き込もうとすると、理由コード RC2053 で失敗します。

作業単位ごとの処理やメッセージのセグメント化を行うと、いずれの場合もキューにある物理メッセージの実際の数が *MaxQDepth* を超えてしまいます。ただし、このことによってメッセージの検索に影響があるわけではありません。キューにあるすべてのメッセージは、MQGET 呼び出しを使用して通常の方法で検索できます。

値はゼロ以上です。上限は、環境により決定されます。

注：キューに入っているメッセージの数が *MaxQDepth* に満たない場合でも、キューのために使用可能な保管スペースを使い切ってしまう可能性があります。

この属性の値を判別するには、MQINQ 呼び出しで IAMDEP セレクターを使用します。

IBM i IBM i での MediaLog (10 桁の符号付き整数)

特定のキューのメディア・リカバリーに必要なログ・エクステンツ (または IBM i 上のジャーナル・レシーバー) の ID。

表 821. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

循環ロギングを使用しているキュー・マネージャーでは、値はヌル・ストリングとして戻されます。

IBM i IBM i での MsgDeliverySequence (10 桁の符号付き整数)

メッセージ・デリバリー・シーケンス。

表 822. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

これは、メッセージが MQGET 呼び出しによってアプリケーションに戻される順番を決定します。

MSFIFO

メッセージは FIFO (先入れ先出し法) の順に戻されます。

MQGET 呼び出しによって戻されるメッセージは、優先順位に関係なく、その呼び出しで指定されている選択基準を満たすメッセージのうちの最初のメッセージであることを意味します。

MSPRIO

メッセージが優先順位順に戻されます。

MQGET 呼び出しによって戻されるメッセージが、その呼び出しで指定されている選択基準を満たすメッセージのうちで、最も優先順位の高いメッセージであることを意味します。各優先順位内では、メッセージは FIFO (先入れ先出し) の順番で返されます。

キューにメッセージが入っているときに関連する属性が変更されると、配布順序は次のようになります。

- MQGET 呼び出しによってメッセージが戻される順番は、そのメッセージがキューに到達したときにそのキューに関して有効な **MsgDeliverySequence** と **DefPriority** の属性値によって、次のように決められます。

- メッセージが到達したときに **MsgDeliverySequence** が MSFIFO であれば、そのメッセージは、優先順位が **DefPriority** である場合と同じ扱いでキューに入れられます。メッセージのメッセージ記述子の **MDPRI** フィールドの値に影響しません。このフィールドには、メッセージが最初に入れられたときの値がそのまま入っています。
- メッセージが到着したときに **MsgDeliverySequence** が MSPRIO であれば、メッセージ記述子内の **MDPRI** フィールドで指定されている優先順位に該当する位置でキューに入れられます。

キュー上にメッセージがあるときに **MsgDeliverySequence** 属性の値が変更された場合、キュー上のメッセージの順序は変更されません。

メッセージがキューにある間に **DefPriority** 属性の値が変更された場合は、**MsgDeliverySequence** 属性が MSFIFO に設定されていても、メッセージは FIFO の順番に配布されるとは限りません。つまり、より高い優先順位でキューに入れられたメッセージが最初に送られます。

この属性の値を判別するには、MQINQ 呼び出しで IAMDS セレクターを使用します。

IBM i IBM i での OpenInputCount (10 桁の符号付き整数)

入力のためのオープン数。

表 823. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X				

MQGET 呼び出しを用いてキューからメッセージを除去するために現在使用されているハンドルの数です。ローカル・キュー・マネージャーに認識されているハンドルの総数です。キューが共有キューである場合、この数には、ローカル・キュー・マネージャーが属しているキュー共有グループにある、他のキュー・マネージャーのキューのために実行された入力のオープンは含まれません。

このカウントには、このキューに解決される別名キューが入力用にオープンされたときのハンドルが含まれます。また、カウントには、入力が行われなかったアクション (例えば、ブラウズのためのみにキューがオープンされた場合) のためにキューがオープンされたときのハンドルは含まれません。

属性の値は、キュー・マネージャーの動作に応じて変動します。

この属性の値を判別するには、MQINQ 呼び出しで IAOIC セレクターを使用します。

IBM i IBM i での OpenOutputCount (10 桁の符号付き整数)

出力のためのオープン数。

表 824. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X				

MQPUT 呼び出しによって、キューにメッセージを追加するために現在使用されているハンドルの数です。ローカル・キュー・マネージャーに認識されているハンドルの総数です。ローカル・キューに対してリモ

ート・キュー・マネージャーで実行された出力のためのオープンは含まれません。キューが共有キューである場合、この数には、ローカル・キュー・マネージャーが属しているキュー共有グループにある、他のキュー・マネージャーのキューのために実行された出力のオープンは含まれません。

このカウントには、このキューに解決される別名キューが出力用にオープンされたときのハンドルが含まれます。また、カウントには、出力が行われなかったアクション (例えば、ブラウズのためのみにキューがオープンされた場合) のためにキューがオープンされたときのハンドルは含まれません。

属性の値は、キュー・マネージャーの動作に応じて変動します。

この属性の値を判別するには、MQINQ 呼び出しで IAQOC セレクターを使用します。

IBM i IBM i での ProcessName (48 バイトの文字ストリング)

プロセス名。

表 825. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

これは、ローカル・キュー・マネージャーについて定義されたプロセス・オブジェクトの名前です。プロセス・オブジェクトは、キューに対するサービスを行うことのできるプログラムを識別します。

この属性の値を判別するには、MQINQ 呼び出しで CAPRON セレクターを使用します。この属性の長さは LNPRON で指定します。

IBM i IBM i での QDepthHighEvent (10 桁の符号付き整数)

キュー・サイズ上限イベントを生成するかどうかを制御します。

表 826. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

キュー・サイズ上限イベントは、アプリケーションがキューにメッセージを書き込んだためキューに入っているメッセージの数がキューのサイズ上限しきい値以上になったことを示します (QDepthHighLimit 属性を参照してください)。

注: この属性の値は動的に変化します。

QDepthHighEvent は、以下の 2 つの値のうちのいずれかをとることができます。

EVRDIS

イベント報告は無効です。

EVRENA

イベント報告は有効です。

イベントの詳細については、[イベント・モニター](#)を参照してください。

この属性の値を判別するには、MQINQ 呼び出しで IAQDHE セレクターを使用します。

IBM i IBM i での QDepthHighLimit (10 桁の符号付き整数)

キュー項目数の上限。

表 827. この属性が適用されるキュー・タイプ

ローカル	モデル	別名	リモート	クラスター
X	X			

キュー・サイズ上限イベントを生成するために、キューのサイズと比較されるしきい値。このイベントは、アプリケーションがキューにメッセージを書き込んだため、キューに入っているメッセージの数がキューのサイズ上限のしきい値以上になったことを示します。**QDepthHighEvent** 属性を参照してください。

値は、キューの最大サイズ (**MaxQDepth** 属性) に対する パーセントで表されます。値の範囲は 0 から 100 です。デフォルト値は 80 です。

この属性の値を判別するには、MQINQ 呼び出しで IAQDHL セレクターを使用します。

IBM i IBM i での QDepthLowEvent (10 桁の符号付き整数)

キュー・サイズ下限イベントを生成するかどうかを制御します。

表 828. この属性が適用されるキュー・タイプ

ローカル	モデル	別名	リモート	クラスター
X	X			

キュー・サイズ下限イベントは、アプリケーションがキューからメッセージを取り出したためキューに入っているメッセージの数がキューのサイズ下限しきい値以下になったことを示します (**QDepthLowLimit** 属性を参照してください)。

注：この属性の値は動的に変化します。

QDepthLowEvent は、以下の値のうちのいずれかをとることができます。

EVRDIS

イベント報告は無効です。

EVRENA

イベント報告は有効です。

イベントの詳細については、[イベント・モニター](#)を参照してください。

この属性の値を判別するには、MQINQ 呼び出しで IAQDLE セレクターを使用します。

IBM i IBM i での QDepthLowLimit (10 桁の符号付き整数)

キュー項目数の下限。

表 829. この属性が適用されるキュー・タイプ

ローカル	モデル	別名	リモート	クラスター
X	X			

キュー・サイズの下限イベントを生成するためにキューのサイズと比較されるしきい値です。このイベントは、アプリケーションがキューからメッセージを取り出したために、キューに入っているメッセージの数がキューのサイズ下限のしきい値以下になったことを示します。**QDepthLowEvent** 属性を参照してください。

値は、キューの最大サイズ (**MaxQDepth** 属性) に対する パーセントで表されます。値の範囲は 0 から 100 です。デフォルト値は 20 です。

この属性の値を判別するには、MQINQ 呼び出しで IAQDLL セレクターを使用します。

▶ IBM i IBM i での QDepthMaxEvent (10 桁の符号付き整数)

キュー満杯イベントを生成するかどうかを制御します。

表 830. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

キュー満杯イベントは、キューが満杯であるために、そのキューに対する書き込みが拒否されたことを示します。つまり、キューのサイズがすでに最大値に達していることになります。

注：この属性の値は動的に変化します。

これは、以下の値のいずれかになります。

EVRDIS

イベント報告は無効です。

EVRENA

イベント報告は有効です。

イベントの詳細については、[イベント・モニター](#)を参照してください。

この属性の値を判別するには、MQINQ 呼び出しで IAQDME セレクターを使用します。

▶ IBM i IBM i での QDesc (64 バイトの文字ストリング)

キューの記述。

表 831. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X	X	X	X

これは、コメントを記述するために使用できるフィールドです。このフィールドの内容は、キュー・マネージャーにとっては特に意味を持ちませんが、キュー・マネージャーが、このフィールドに表示できる文字のみが含まれていることを要件としている場合もあります。フィールドにヌル文字を入れることはできません。また、必要に応じて、右側がブランクで埋められます。DBCS をインストール済みの環境では、このフィールドに DBCS 文字を入れることができます (最大フィールド長として 64 バイトが適用されます)。

注：このフィールドに、(CodedCharSetId キュー・マネージャー属性で定義されている) キュー・マネージャーの文字セットに含まれていない文字が含まれている場合、このフィールドが別のキュー・マネージャーに送信されると、それらの文字が正しく変換されない可能性があります。

この属性の値を判別するには、MQINQ 呼び出しで CAQD セレクターを使用します。この属性の長さは、LNQD で指定されます。

▶ IBM i IBM i での QName (48 バイトの文字ストリング)

キュー名。

表 832. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X		X	X	X

これは、ローカル・キュー・マネージャー上に定義されたキューの名前です。キュー名について詳しくは、[IBM MQ オブジェクトの命名規則](#)を参照してください。キュー・マネージャー上で定義されたキューはすべて、同一のキュー名前空間を共有します。そのため、QTLOC キューと QTALS キューを同じ名前にすることはできません。

この属性の値を判別するには、MQINQ 呼び出しで CAQN セレクターを使用します。この属性の長さは LNQN で指定します。

IBM i IBM i での QServiceInterval (10 桁の符号付き整数)

キュー・サービス間隔の目標値。

表 833. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

サービス間隔は、サービス間隔上位イベントおよびサービス間隔 OK イベントを生成する際の比較用に使われます。**QServiceIntervalEvent** 属性を参照してください。

値は、ミリ秒単位で表されます。値の範囲は、0 から 999 999 999 です。

この属性の値を判別するには、MQINQ 呼び出しで IAQSI セレクターを使用します。

IBM i IBM i での QServiceIntervalEvent (10 桁の符号付き整数)

サービス間隔上限イベントまたはサービス間隔 OK イベントを生成するかどうかを制御します。

表 834. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

- サービス間隔上位イベントが生成されるのは、検査の結果、少なくとも **QServiceInterval** 属性によって示されている期間、このキューから取り出されたメッセージがなかったことが分かった場合です。
- サービス間隔 OK イベントが生成されるのは、検査の結果、**QServiceInterval** 属性によって示されている期間内に、このキューからメッセージが取り出されていることが分かった場合です。

注：この属性の値は動的に変化します。

この属性には、以下のいずれかの値を使用できます。

QSIEHI

キュー・サービス間隔上限イベントは有効です。

- キュー・サービス間隔上位イベントが**使用可能**であり、
- キュー・サービス間隔 OK イベントは**使用不可**である。

QSIEOK

キュー・サービス間隔 OK イベントは有効です。

- キュー・サービス間隔上位イベントが**使用不可**であり、
- キュー・サービス間隔 OK イベントは**使用可能**である。

QSIENO

どのキュー・サービス間隔イベントも無効です。

- キュー・サービス間隔上位イベントが**使用不可**であり、
- キュー・サービス間隔 OK イベントも**使用不可**である。

共有キューでは、この属性の値は無視されます。値 QSIENO が想定されます。

イベントの詳細については、[イベント・モニター](#)を参照してください。

この属性の値を判別するには、MQINQ 呼び出しで IAQSIE セレクターを使用します。

IBM i IBM i での QSGDisp (10 桁の符号付き整数)

キュー共有グループ後処理。

表 835. この属性が適用されるキュー・タイプ

ローカル	モデル	別名	リモート	クラスター
X		X	X	

キューの属性指定を設定します。値は、次のいずれか 1 つです。

QSGDQM

キュー・マネージャーの後処理。

このオブジェクトには、キュー・マネージャーの後処理が含まれます。これは、このオブジェクトの定義を、ローカル・キュー・マネージャーだけが認識し、キュー共有グループ内の他のキュー・マネージャーは認識しないことを意味します。

キュー共有グループ内の各キュー・マネージャーが現行オブジェクトと同じ名前およびタイプのオブジェクトを持つことは可能ですが、それらは別個のオブジェクトであり、相関関係はありません。それらの属性が互いに同じになるように制約されることはありません。

QSGDCP

コピーされたオブジェクトの後処理。

このオブジェクトは、共用リポジトリ内に存在するマスター・オブジェクト定義のローカル・コピーです。キュー共有グループ内の各キュー・マネージャーが、このオブジェクトの独自のコピーを持つことができます。最初はどのコピーの属性も同じですが、MQSC コマンドを使用して各コピーを変更し、属性を他のコピーと違うものにすることができます。共有リポジトリのマスター定義が更新されると、コピーの属性は再同期化されます。

QSGDSH

共有後処理。

このオブジェクトは共有の属性指定を持ちます。これは、共有リポジトリ内にこのオブジェクトの単一インスタンスが存在していて、それがキュー共有グループ内の全キュー・マネージャーから認識されることを意味します。グループ内のキュー・マネージャーは、このオブジェクトにアクセスするとき、このオブジェクトの単一の共有インスタンスにアクセスしています。

この属性の値を判別するには、MQINQ 呼び出しで IAQSGD セレクターを使用します。

z/OS この属性は、z/OS でのみサポートされます。

IBM i IBM i での QType (10 桁の符号付き整数)

キュー・タイプ。

表 836. この属性が適用されるキュー・タイプ

ローカル	モデル	別名	リモート	クラスター
X		X	X	X

この属性の値は、次のいずれかです。

QTALS

別名キュー定義。

QTCLUS

クラスター・キュー。

QTLOC

ローカル・キュー。

QTREM

リモート・キューのローカル定義。

この属性の値を判別するには、MQINQ 呼び出しで IAQTYP セレクターを使用します。

IBM i IBM i での RemoteQMgrName (48 バイトの文字ストリング)

リモート・キュー・マネージャーの名前。

表 837. この属性が適用されるキュー・タイプ

ローカル	モデル	別名	リモート	クラスター
			X	

RemoteQName が定義されているリモート・キュー・マネージャーの名前です。 *RemoteQName* キューに QSGDCP または QSGDSH の *QSGDisp* 値がある場合は、 *RemoteQMgrName* は *RemoteQName* を所有する キュー共有グループの名前にすることができます。

アプリケーションがリモート・キューのローカル定義をオープンする場合、 *RemoteQMgrName* をブランク にしたりローカル・キュー・マネージャーの名前を指定することはできません。 *XmitQName* がブランクで ある場合、 *RemoteQMgrName* と同じ名前のローカル・キューが伝送キューとして使用されます。

RemoteQMgrName という名前のキューが存在しない場合は、 **DefXmitQName** キュー・マネージャー属性 に指定されているキューが使用されます。

この定義がキュー・マネージャー別名に使用される場合、 *RemoteQMgrName* は別名が割り当てられる キュー・マネージャーの名前です。 これは、ローカル・キュー・マネージャーの名前であっても構いません。 また、オープンが行われるときに *XmitQName* がブランクであれば、 *RemoteQMgrName* と同じ名前のロー カル・キューが存在しなければなりません。このキューは、伝送キューとして使用されます。

この定義が応答先別名として使用される場合、この名前はキュー・マネージャーの名前であり、それは *MDRM* になります。

注：キュー定義の作成時または変更時には、この属性に関して指定されている値の妥当性検査は行われま せん。

この属性の値を判別するには、MQINQ 呼び出しで CARQMN セレクターを使用します。 この属性の長さは、 LNQMN で指定されます。

IBM i IBM i での RemoteQName (48 バイトの文字ストリング)

リモート・キューの名前。

表 838. この属性が適用されるキュー・タイプ

ローカル	モデル	別名	リモート	クラスター
			X	

これは、リモート・キュー・マネージャー *RemoteQMgrName* で認識されているキューの名前です。

アプリケーションがリモート・キューのローカル定義をオープンする場合、オープン時に *RemoteQName* をブランクにしてはなりません。

この定義がキュー・マネージャー別名定義に使用される場合、オープン時に *RemoteQName* はブランクで なければなりません。

定義が応答先 (reply-to) 別名に使用されている場合、この名前は、MDRQ になるキューの名前です。

注: キュー定義の作成時または変更時には、この属性に関して指定されている値の妥当性検査は行われません。

この属性の値を判別するには、MQINQ 呼び出しで CARQN セレクターを使用します。この属性の長さは LNQN で指定します。

IBM i IBM i での RetentionInterval (10 桁の符号付き整数)

保存インターバル。

表 839. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

キューを保存する期間です。この期間が過ぎると、キューは削除の対象となります。

時間は、そのキューが作成された日付と時刻を基点として時間数で測定されます。キューの作成日は、CreationDate 属性に記録され、キューの作成時刻は、CreationTime 属性に記録されます。

この情報を用いて、ハウスキーピング・アプリケーションやオペレーターは、不要になったキューを識別して削除することができます。

注: キュー・マネージャーが、この属性に基づいてキューの削除を試みたり、あるいは保存期間が満了していないキューが削除されるのを防止しようとしたりすることはありません。必要な処置は、ユーザーの責任で行ってください。

永続動的キューが累積されないようにするには、現実的な保存間隔を使用する必要があります (DefinitionType を参照)。ただし、この属性は、事前定義されたキューでも使用することができます。

この属性の値を判別するには、MQINQ 呼び出しで IARINT セレクターを使用します。

IBM i IBM i での Scope (10 桁の符号付き整数)

このキューに対するエントリーがセル・ディレクトリーでも存在するかどうかを制御します。

表 840. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X		X	X	

セル・ディレクトリーは、インストール可能な名前サービスにより提供されます。これは、以下の値のいずれかになります。

SCOQM

キュー・マネージャー有効範囲。

キュー定義の有効範囲は、キュー・マネージャー内です。すなわち、キューの定義は、そのキューを所有しているキュー・マネージャーの有効範囲を超えません。他のキュー・マネージャーからそのキューを出力用にオープンするときには、キューを所有しているキュー・マネージャーの名前を指定する必要があります。あるいは、呼び出す側のキュー・マネージャーがそのキューのローカル定義を持っている必要があります。

SCOCEL

セルの有効範囲。

キュー定義の有効範囲は、セルになります。すなわち、キュー定義は、セル内のすべてのキュー・マネージャーが使用できるセル・ディレクトリー内に入っています。キューの名前を指定するのみで、セル内のどのキュー・マネージャーからも出力できるように、キューをオープンすることができます。

キューを所有するキュー・マネージャーの名前を指定する必要はありません。しかし、その名前を持つキューのローカル定義も持っているセル内のキュー・マネージャーは、そのキュー定義を使用できません。ローカル定義の方が優先されるためです。

セル・ディレクトリーは、LDAP (Lightweight Directory Access Protocol) などのインストール可能な名前サービスにより提供されます。IBM MQ は DCE (分散コンピューティング環境) の名前サービスをもサポートしていないことに注意してください。この名前サービスは、キュー定義を DCE ディレクトリー (これももうサポートされていません) に挿入するために以前使用されていました。

モデル・キューと動的キューは、セル有効範囲を持つことはできません。

値は、セル・ディレクトリーをサポートする名前サービスが構成されている場合にのみ有効です。

この属性の値を判別するには、MQINQ 呼び出しで IASCOPI セレクターを使用します。

この属性のサポートには、次のような制約事項があります。

- IBM i ではこの属性がサポートされますが、有効なのは SCOQM のみです。

IBM i IBM i での Shareability (10 桁の符号付き整数)

キューを入力用に共有できるかどうか。

表 841. この属性が適用されるキュー・タイプ

ローカル	モデル	別名	リモート	クラスター
X	X			

これは、キューを同時に複数回、入力用にオープンできるかどうかを示します。これは、以下の値のいずれかになります。

QASHR

キューは共有可能。

OOINPS オプションを使用して、複数のオープンが可能です。

QANSHR

キューは共有不可。

OOINPS オプションを指定した MQOPEN 呼び出しは、OOINPX と見なされます。

この属性の値を判別するには、MQINQ 呼び出しで IASHAR セレクターを使用します。

IBM i IBM i での TriggerControl (10 桁の符号付き整数)

トリガー制御。

表 842. この属性が適用されるキュー・タイプ

ローカル	モデル	別名	リモート	クラスター
X	X			

アプリケーションにキューの処理を開始させるために、開始キューにトリガー・メッセージを書き込むかどうかを制御します。これは、以下のいずれかになります。

TCOFF

トリガー・メッセージは不要。

トリガー・メッセージはこのキューに書き込まれません。TriggerType の値は、この場合には無効です。

TCON

トリガー・メッセージは必要。

該当するトリガー・イベントが起こったときに、トリガー・メッセージがこのキューに書き込まれます。

この属性の値を判別するには、MQINQ 呼び出しで IATRGC セレクターを使用します。この属性の値を変更するには、MQSET 呼び出しを使用します。

IBM i IBM i での TriggerData (64 バイトの文字ストリング)

トリガー・データです。

表 843. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

メッセージがこのキューに到着した結果、トリガー・メッセージが開始キューに書き込まれることになった場合に、キュー・マネージャーがトリガー・メッセージに挿入する自由形式のデータです。

このデータの内容は、キュー・マネージャーにとっては意味のないものです。データは、開始キューを処理するトリガー・モニター・アプリケーション、あるいはトリガー・モニターによって開始されるアプリケーションにとって意味があります。

文字ストリング内にヌルを入れることはできません。このストリングは、必要に応じて、右側にブランクが埋め込まれます。

この属性の値を判別するには、MQINQ 呼び出しで CATRGD セレクターを使用します。この属性の値を変更するには、MQSET 呼び出しを使用します。この属性の長さは、LNTRGD で指定されます。

IBM i IBM i での TriggerDepth (10 桁の符号付き整数)

トリガー項目数。

表 844. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

トリガー・メッセージを書き込む前にキューに入れなければならない、優先順位が *TriggerMsgPriority* 以上のメッセージの数です。この属性は、*TriggerType* が TTDPTH に設定されている場合に適用されます。*TriggerDepth* の値は、1 以上です。この属性は他の場合には使われません。

この属性の値を判別するには、MQINQ 呼び出しで IATRGC セレクターを使用します。この属性の値を変更するには、MQSET 呼び出しを使用します。

IBM i IBM i での TriggerMsgPriority (10 桁の符号付き整数)

IBM MQ for IBM i でのトリガーに対するしきい値メッセージ優先度。

表 845. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

この値より低いメッセージ優先順位を持つメッセージがトリガー・メッセージの生成の対象にならないことを表します(つまり、キュー・マネージャーは、トリガー・メッセージを生成するかどうかを決定する際に、それらのメッセージを無視します)。*TriggerMsgPriority* は、ゼロ (最低) から *MaxPriority* (最

高。1412 ページの『[IBM iでのキュー・マネージャーの属性](#)』を参照)の範囲にすることができます。値をゼロにすると、すべてのメッセージがトリガー・メッセージの生成に寄与します。

この属性の値を判別するには、MQINQ 呼び出しで IATRGP セレクターを使用します。この属性の値を変更するには、MQSET 呼び出しを使用します。

IBM i IBM i での *TriggerType* (10 桁の符号付き整数)

トリガー・タイプ。

表 846. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

メッセージがこのキューに到着した結果トリガー・メッセージが書き込まれる条件を制御します。値は、次のいずれか 1 つです。

TTNONE

トリガー・メッセージは書き込まれません。

メッセージがキューに到着した結果、トリガー・メッセージは書き込まれません。これは、*TriggerControl* を TCOFF に設定するのと同じことです。

TTFRST

トリガー・メッセージは、キューのサイズが 0 から 1 になったときに書き込まれます。

トリガー・メッセージは、キューの中で優先順位が *TriggerMsgPriority* 以上であるメッセージの数が 0 から 1 に変化すると、必ず書き込まれます。

TTEVRY

トリガー・メッセージは、すべてのメッセージについて書き込まれます。

トリガー・メッセージは、優先順位が *TriggerMsgPriority* 以上であるメッセージがキューに到着するたびに書き込まれます。

TTDPTH

トリガー・メッセージは、サイズのしきい値を超えた場合に書き込まれます。

トリガー・メッセージは、キュー上の優先順位が *TriggerMsgPriority* 以上のメッセージの数が *TriggerDepth* 以上になるたびに書き込まれます。トリガー・メッセージが書き込まれると、*TriggerControl* が TCOFF に設定されて、トリガーが明示的にオンになるまで、トリガーが起動されないようにします。

この属性の値を判別するには、MQINQ 呼び出しで IATRGT セレクターを使用します。この属性の値を変更するには、MQSET 呼び出しを使用します。

IBM i IBM i での *Usage* (10 桁の符号付き整数)

キューの用途。

表 847. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
X	X			

キューの用途を示します。値は、次のいずれか 1 つです。

USNORM

通常使用。

アプリケーションがメッセージを書き込んだり読み取ったりする際に通常使用するキューです。伝送キューではありません。

USTRAN

伝送キュー。

リモート・キュー・マネージャー宛てのメッセージを保存するために使用されるキューです。通常のアプリケーションがリモート・キューにメッセージを送信すると、ローカル・キュー・マネージャーは、そのメッセージを特別な形式で該当する伝送キューに一時保管します。次に、メッセージ・チャネル・エージェントが、伝送キューからメッセージを読み取り、リモート・キュー・マネージャーに伝送します。伝送キューの詳細については、[伝送キュー](#)を参照してください。

伝送キューを OOOOUT でオープンしてそれに直接メッセージを書き込めるのは、特権を与えられたアプリケーションのみです。通常この処理を行うのは、ユーティリティ・アプリケーションのみです。メッセージ・データの形式を間違えないようにしてください ([1255 ページの『IBM i での MQXQH \(伝送キュー・ヘッダー\)』](#)を参照)。形式が正しくないと、伝送処理時にエラーが発生する可能性があります。PM* コンテキスト・オプションが指定されていない場合は、コンテキストの引き渡しも設定も行われません。

この属性の値を判別するには、MQINQ 呼び出しで IAUSAG セレクターを使用します。

IBM i IBM i での XmitQName (48 バイトの文字ストリング)

伝送キュー名。

表 848. この属性が適用されるキュー・タイプ				
ローカル	モデル	別名	リモート	クラスター
			X	

リモート・キューまたはキュー・マネージャー別名定義のいずれかについてオープンが行われるときに、この属性がブランクでなければ、それは、メッセージの転送に使用されるローカル伝送キューの名前を指定します。

XmitQName がブランクの場合は、RemoteQMgrName と同じ名前のローカル・キューが伝送キューとして使用されます。RemoteQMgrName という名前のキューが存在しない場合は、DefXmitQName キュー・マネージャー属性に指定されているキューが使用されます。

定義がキュー・マネージャーの別名として使用され、RemoteQMgrName がローカル・キュー・マネージャーの名前である場合には、この属性は無視されます。また、この定義が応答先キュー別名定義として使用されている場合にも、これは無視されます。

この属性の値を判別するには、MQINQ 呼び出しで CAXQN セレクターを使用します。この属性の長さは LNQN で指定します。

名前リストの属性

このトピックでは、名前リストに特有の属性について 要約しています。属性の説明は、アルファベット順に掲載しています。

注: 示されている属性の名前は、MQINQ 呼び出しおよび MQSET 呼び出しで使用する名前です。

属性の説明

名前リスト・オブジェクトには、以下の属性があります。

AlterationDate (12 バイトの文字ストリング)

定義が最後に変更された日付。

これは、定義を最後に変更した日付です。日付の形式は YYYY-MM-DD で、その後に 2 つの末尾ブランクを付けて長さ 12 バイトになります。

この属性の値を判別するには、MQINQ 呼び出しで CAALTD セレクターを使用します。この属性の長さは LNDATE で指定します。

AlterationTime (8 バイトの文字ストリング)

定義が最後に変更された時刻。

これは、定義を最後に変更した時刻です。時刻の形式は HH.MM.SS です。

この属性の値を判別するには、MQINQ 呼び出しで CAALTT セレクターを使用します。この属性の長さは LNTIME で指定します。

NameCount (10 桁の符号付き整数)

名前リスト内の名前の数。

これは、ゼロ以上です。以下の値が定義されます。

NCMXNL

名前リスト内の名前の最大数。

この属性の値を判別するには、MQINQ 呼び出しで IANAMC セレクターを使用します。

NamelistDesc (64 バイトの文字ストリング)

名前リストの記述。

これは、説明コメントの記入に使用できるフィールドです。この値は、定義プロセスで設定されます。このフィールドの内容は、キュー・マネージャーにとっては特に意味を持ちませんが、キュー・マネージャーが、このフィールドに表示できる文字のみが含まれていることを要件としている場合もあります。フィールドにヌル文字を入れることはできません。また、必要に応じて、右側がブランクで埋められます。DBCS をインストール済みの環境では、このフィールドに DBCS 文字を入れることができます (最大フィールド長として 64 バイトが適用されます)。

注: このフィールドに、(**CodedCharSetId** キュー・マネージャー属性で定義されている) キュー・マネージャーの文字セットに含まれていない文字が含まれている場合、このフィールドが別のキュー・マネージャーに送信されると、それらの文字が正しく変換されない可能性があります。

この属性の値を判別するには、MQINQ 呼び出しで CALSTD セレクターを使用します。

この属性の長さは LNNLD によって指定されます。

NamelistName (48 バイトの文字ストリング)

名前リストの名前。

ローカル・キュー・マネージャーに定義されている名前リストの名前です。

名前リストは、それぞれ、同じキュー・マネージャーに属する他の名前リストとは異なる名前を持ちますが、別のタイプのキュー・マネージャー・オブジェクト (キューなど) の名前とは重複していてもかまいません。

この属性の値を判別するには、MQINQ 呼び出しで CALSTN セレクターを使用します。

この属性の長さは LNNLN によって指定されます。

Names (48 バイトの文字ストリング x NameCount)

NameCount 個の名前のリスト。

各名前は、ローカル・キュー・マネージャーに定義されている オブジェクトの名前です。オブジェクト名について詳しくは、[IBM MQ オブジェクトの命名](#)を参照してください。

この属性の値を判別するには、MQINQ 呼び出しで CANAMS セレクターを使用します。

リスト内の各名前の長さは LNOBJN で指定します。

このトピックでは、プロセス定義に固有の属性をまとめて紹介します。属性の説明は、アルファベット順に掲載しています。

注: 示されている属性の名前は、MQINQ 呼び出しおよび MQSET 呼び出しで使用する名前です。MQSC コマンドを使用して属性を定義、変更、または表示するときには、代替の短縮名が使用されます。詳細については、[MQSC コマンド](#)を参照してください。

属性の説明

プロセス定義オブジェクトには、以下の属性があります。

AlterationDate (12 バイトの文字ストリング)

定義が最後に変更された日付。

これは、定義を最後に変更した日付です。日付の形式は YYYY-MM-DD で、その後に 2 つの末尾空白を付けて長さ 12 バイトになります。

この属性の値を判別するには、MQINQ 呼び出しで CAALTD セレクターを使用します。この属性の長さは LNDATE で指定します。

AlterationTime (8 バイトの文字ストリング)

定義が最後に変更された時刻。

これは、定義を最後に変更した時刻です。時刻の形式は HH.MM.SS です。

この属性の値を判別するには、MQINQ 呼び出しで CAALTT セレクターを使用します。この属性の長さは LNTIME で指定します。

ApplId (256 バイトの文字ストリング)

アプリケーション ID。

これは、開始されるアプリケーションを識別する文字ストリングです。この情報は、開始キュー上のメッセージを処理するトリガー・モニター・アプリケーションが使用するものです。この情報は、トリガー・メッセージの一部として開始キューに送信されます。

ApplId の意味は、トリガー・モニター・アプリケーションによって決まります。IBM MQ によって提供されるトリガー・モニターでは、*ApplId* を実行可能プログラムの名前にする必要があります。

文字ストリング内にヌルを入れることはできません。このストリングは、必要に応じて、右側に空白が埋め込まれます。

この属性の値を判別するには、MQINQ 呼び出しで CAAPPI セレクターを使用します。この属性の長さは LNPROA によって指定されます。

ApplType (10 桁の符号付き整数)

アプリケーション・タイプ。

これは、トリガー・メッセージの受信にตอบสนองして開始されるプログラムの性質を識別します。この情報は、開始キュー上のメッセージを処理するトリガー・モニター・アプリケーションが使用するものです。この情報は、トリガー・メッセージの一部として開始キューに送信されます。

ApplType には任意の値を設定できます。標準のタイプとして、以下の値を使用できます。ユーザー定義のアプリケーション・タイプは、ATUFST から ATULST の範囲の値に制限されています。

時刻 CICS

CICS トランザクション。

AT400

IBM i アプリケーション。

ATUFST

ユーザー定義のアプリケーション・タイプの最低値。

ATULST

ユーザー定義のアプリケーション・タイプの最高値。

この属性の値を判別するには、MQINQ 呼び出しで IAAPPT セレクターを使用します。

EnvData (128 バイトの文字ストリング)

環境データ。

これは、始動するアプリケーションに関する環境関連の情報を含む文字ストリングです。この情報は、開始キュー上のメッセージを処理するトリガー・モニター・アプリケーションが使用するものです。この情報は、トリガー・メッセージの一部として開始キューに送信されます。

EnvData の意味は、トリガー・モニター・アプリケーションによって決まります。IBM MQ によって提供されるトリガー・モニターは、開始されたアプリケーションに渡されるパラメーター・リストに *EnvData* を追加します。パラメーター・リストは、MQTMC2 構造体、1 つのブランク、および末尾ブランクを除去した *EnvData* で構成されます。

文字ストリング内にヌルを入れることはできません。このストリングは、必要に応じて、右側にブランクが埋め込まれます。

この属性の値を判別するには、MQINQ 呼び出しで CAENVD セレクターを使用します。この属性の長さは LNPROE によって指定されます。

ProcessDesc (64 バイトの文字ストリング)

プロセスの説明。

これは、コメントを記述するために使用できるフィールドです。フィールドの内容はキュー・マネージャーにとって重要なものではありませんが、表示できる文字以外は使用しないでください。フィールドにヌル文字を入れることはできません。また、必要に応じて、右側がブランクで埋められます。DBCS をインストール済みの環境では、このフィールドに DBCS 文字を入れることができます (最大フィールド長として 64 バイトが適用されます)。

注: このフィールドに、(**CodedCharSetId** キュー・マネージャー属性で定義されている) キュー・マネージャーの文字セットに含まれていない文字が含まれている場合、このフィールドが別のキュー・マネージャーに送信されると、それらの文字が正しく変換されない可能性があります。

この属性の値を判別するには、MQINQ 呼び出しで CAPROD セレクターを使用します。

この属性の長さは LNPROD によって指定されます。

ProcessName (48 バイトの文字ストリング)

プロセス名。

これは、ローカル・キュー・マネージャーで定義されるプロセス定義の名前です。

それぞれのプロセス定義には、キュー・マネージャーに所属する他のプロセス定義の名前とは異なる名前があります。しかし、プロセス定義の名前は、タイプの異なる他のキュー・マネージャー・オブジェクト (例えば、キュー) の名前と同じでも構いません。

この属性の値を判別するには、MQINQ 呼び出しで CAPRON セレクターを使用します。

この属性の長さは LNPRON で指定します。

UserData (128 バイトの文字ストリング)

ユーザー・データ。

これは、始動するアプリケーションに関するユーザー情報を含む文字ストリングです。この情報は、開始キュー上のメッセージを処理するトリガー・モニター・アプリケーションか、あるいは、トリガー・モニターによって開始されるアプリケーションが使用するものです。この情報は、トリガー・メッセージの一部として開始キューに送信されます。

UserData の意味は、トリガー・モニター・アプリケーションによって決まります。IBM MQ によって提供されるトリガー・モニターは、パラメーター・リストの一部として、始動するアプリケーションに

UserData を渡します。パラメーター・リストは、MQTMC2 構造体 (*UserData* を含む) と、それに続く 1 つのブランク、および末尾ブランクを除去した *EnvData* で構成されます。

文字ストリング内にヌルを入れることはできません。このストリングは、必要に応じて、右側にブランクが埋め込まれます。

この属性の値を判別するには、MQINQ 呼び出しで CAUSRD セレクターを使用します。この属性の長さは LNPROU で指定します。

IBM i IBM i でのキュー・マネージャーの属性

キュー・マネージャー属性の要約。

キュー・マネージャー属性には、特定の実装に固定されているものもあれば、MQSC コマンド ALTER QMGR を使用して変更できるものもあります。属性は、DISPLAY QMGR コマンドを使用して表示することもできます。ほとんどのキュー・マネージャー属性は、特別な OTQM オブジェクトをオープンし、戻されたハンドルを指定して MQINQ 呼び出しを発行することによって調べることができます。

次の表では、キュー・マネージャーに固有の属性が要約されています。属性の説明は、アルファベット順に掲載しています。

注：このセクションで示されている属性の名前は、MQINQ 呼び出しおよび MQSET 呼び出しで使用する名前です。MQSC コマンドを使用して属性を定義、変更、または表示するときには、代替の短縮名が使用されます。詳細については、[MQSC コマンド](#) を参照してください。

表 849. キュー・マネージャーの属性	
属性	説明
AlterationDate	定義が最後に変更された日付
AlterationTime	定義が最後に変更された時刻
AuthorityEvent	許可 (不許可) イベントを生成するかどうかを制御します。
BridgeEvent	IMS ブリッジ・イベントを生成するかどうかを制御します。
ChannelAutoDef	自動チャンネル定義が許可されているかどうかを制御します。
ChannelAutoDefEvent	チャンネル自動定義イベントを生成するかどうかを制御します。
ChannelAutoDefExit	自動チャンネル定義のためのユーザー出口の名前。
ChannelEvent	チャンネル・イベントを生成するかどうかを制御します
ClusterCacheType	クラスター・キャッシュのサイズを固定するか、または動的にサイズ変更するかを制御します
ClusterWorkloadData	クラスター・ワークロード出口のユーザー・データ。
ClusterWorkloadExit	クラスター・ワークロード管理のためのユーザー出口の名前。
ClusterWorkloadLength	クラスター・ワークロード出口に受け渡されるメッセージ・データの最大長。
CodedCharSetId	コード化文字セット ID
CommandEvent	コマンド・イベント・メッセージをキューに入れるかどうかを制御します。
CommandInputQName	コマンド入力キュー名。
CommandLevel	コマンド・レベル
ConfigurationEvent	構成イベント
DeadLetterQName	送達不能キューの名前。

表 849. キュー・マネージャーの属性 (続き)	
属性	説明
DefClusterXmitQueueType	デフォルトのクラスタ伝送キュータイプ。
DefXmitQName	デフォルトの伝送キューの名前。
DistLists	配布リスト・サポート
InhibitEvent	禁止 (読み取り禁止および書き込み禁止) イベントを生成するかどうかを制御します。
LocalEvent	ローカル・エラー・イベントを生成するかどうかを制御します。
LoggerEvent	リカバリー・ログ・イベントを生成するかどうかを制御します
MaxHandles	ハンドルの最大数
MaxMsgLength	メッセージの最大長 (バイト数)。
MaxPriority	最高の優先順位
MaxUncommittedMsgs	1 つの作業単位内のコミットされていないメッセージの最大数
PerformanceEvent	パフォーマンスに関連したイベントを生成するかどうかを制御します。
Platform	キュー・マネージャーが実行されているプラットフォーム。
PubSubMode	パブリッシュ/サブスクライブ・エンジンとキュー・パブリッシュ/サブスクライブ・インターフェースが実行されているかどうか
QMgrDesc	キュー・マネージャーの記述。
QMgrIdentifier	キュー・マネージャーの固有の内部生成 ID
QMgrName	キュー・マネージャー名
RemoteEvent	リモート・エラー・イベントを生成するかどうかを制御します。
RepositoryName	このキュー・マネージャーがリポジトリ・サービスを提供しているクラスターの名前。
RepositoryNamelist	このキュー・マネージャーがリポジトリ・サービスを提供しているクラスターの名前を含む名前リスト・オブジェクトの名前。
SSLCRLNamelist	認証情報オブジェクトの名前が入った 名前リスト・オブジェクトの名前 (注の 1 を参照)
SSLEvent	TLS イベントを生成するかどうかを制御します
SSLKeyRepository	TLS キー・リポジトリの場所 (注の 1 を参照)
SSLKeyResetCount	暗号鍵を再折衝する前に TLS 通信内で送受信される非暗号化バイト数を指定します
StartStopEvent	開始および停止イベントを生成するかどうかを制御します。
SyncPoint	同期点の可用性
TraceRouteRecording	メッセージに関するトレース・ルート情報の記録を制御します
TreeLifeTime	非管理トピックの存続時間 (ミリ秒単位)
TriggerInterval	トリガー・メッセージの間隔。

表 849. キュー・マネージャーの属性 (続き)

属性	説明
注: 1. この属性は MQINQ 呼び出しを使用して照会することができず、このセクションでは説明されていません。この属性について詳しくは、Change Queue Manager を参照してください。	

IBM i IBM i での AlterationDate (12 バイトの文字ストリング)

定義が最後に変更された日付。

これは、定義を最後に変更した日付です。日付の形式は YYYY-MM-DD で、その後に 2 つの末尾ブランクを付けて長さ 12 バイトになります。

この属性の値を判別するには、MQINQ 呼び出しで CAALTD セレクターを使用します。この属性の長さは LNDATE で指定します。

IBM i IBM i での AlterationTime (8 バイトの文字ストリング)

定義が最後に変更された時刻。

これは、定義を最後に変更した時刻です。時刻の形式は HH.MM.SS です。

この属性の値を判別するには、MQINQ 呼び出しで CAALTT セレクターを使用します。この属性の長さは LNTIME で指定します。

IBM i IBM i での AuthorityEvent (10 桁の符号付き整数)

許可 (非許可) イベントが生成されるかどうかを制御する。

AuthorityEvent 属性は、次のいずれかの値に設定する必要があります。

EVRDIS

イベント報告は無効です。

EVRENA

イベント報告は有効です。

イベントの詳細については、[イベント・モニター](#)を参照してください。

この属性の値を判別するには、MQINQ 呼び出しで IAAUTE セレクターを使用します。

IBM i IBM i での BridgeEvent (文字ストリング)

この属性は、IMS ブリッジ・イベント・メッセージが SYSTEM.ADMIN.CHANNEL.EVENT キューに書き込まれるかどうかを決定します。これは z/OS でのみサポートされます。

IBM i IBM i での ChannelAutoDef (10 桁の符号付き整数)

自動チャネル定義が許可されているかどうかを制御する。

この属性はタイプ CTRCVR および CTSVCN のチャネルの自動定義を制御します。CTCLSD チャネルの自動定義は常に使用可能です。これは、以下の値のいずれかになります。

CHADDI

チャネルの自動定義は無効です。

CHADEN

チャネルの自動定義は有効です。

この属性の値を判別するには、MQINQ 呼び出しで IACAD セレクターを使用します。

IBM i IBM i での ChannelAutoDefEvent (10 桁の符号付き整数)

チャネル自動定義のイベントが生成されるかどうかを制御する。

これは、タイプ CTRCVR、CTSVCN、および CTCLSD のチャネルに適用されます。これは、以下の値のいずれかになります。

EVRDIS

イベント報告は無効です。

EVRENA

イベント報告は有効です。

イベントの詳細については、[モニターとパフォーマンス](#)を参照してください。

この属性の値を判別するには、MQINQ 呼び出しで IACADE セレクターを使用します。

IBM i IBM i での ChannelAutoDefExit (20 バイトの文字ストリング)

チャンネル自動定義用のユーザー出口の名前。

この名前がブランクではなく、`ChannelAutoDef` に値 `CHADEN` がある場合、出口はキュー・マネージャーがチャンネル定義を作成しようとするたびに呼び出されます。これは、タイプ `CTRCVR`、`CTSVCN`、および `CTCLSD` のチャンネルに適用されます。このとき、出口は次のいずれかの処理を行うことができます。

- 何も変更することなくチャンネル定義の作成ができます。
- 作成されたチャンネル定義の属性を変更する。
- チャンネルの作成をまったく行わない。

この属性の値を判別するには、MQINQ 呼び出しで `CACADX` セレクターを使用します。属性の長さは、`LNEXN` で指定します。

IBM i IBM i での ChannelEvent (文字ストリング)

チャンネル・イベント・メッセージが生成されるかどうかを決定します。

この属性は、チャンネル・イベント・メッセージが `SYSTEM.ADMIN.CHANNEL.EVENT` キューに書き込まれるかどうかを決定します。書き込む場合には、キューに入れられるメッセージのタイプを決定します (例えば、`'channel started'`、`'channel stopped'`、`'channel not activated'`)。この属性を実装する前は、チャンネル・イベント・メッセージがキューに入れられないようにする唯一の方法はターゲット・キューを削除することでした。

この属性を使用することにより、IMS ブリッジ・イベントのみを収集することもできます (チャンネル・イベントをオフに切り替えることができるため、チャンネル・イベントが同じキューに書き込まれることはありません)。同じことは、チャンネル・イベントを収集せずに収集することができる `TLS` イベントにも当てはまります。

この属性を使用して、ユーザーは重要なイベントのみを収集することもできます (例えば、通常どおり開始および停止する場合ではなく、チャンネルにエラーが含まれている場合など)。

`ChannelEvent` 属性の値は以下のいずれかです。

- `EVREXP` (次のチャンネル・イベントだけが生成されます: `RC2279`、`RC2283`、`RC2284`、`RC2295`、`RC2296`)。
- `EVRENA` (すべてのチャンネル・イベントが生成されます。つまり、`EVREXP` によって生成されるイベントに加えて、`RC2282`、および `RC2283` イベントも生成されます)。
- `EVRDIS` (チャンネル・イベントは生成されません。これはキュー・マネージャーの初期デフォルト値です)。

この属性値を調べるには、MQINQ 呼び出しで `IACHNE` セレクターを使用します。

IBM i IBM i での ClusterCacheType (32 バイトの文字ストリング)

クラスター・キャッシュのサイズを固定するか、あるいは動的にサイズ変更するかを制御します。

これは、クラスター・ワークロード出口が呼び出されたとき、この出口に引き渡される 32 バイトのユーザー定義文字ストリングです。出口に引き渡すデータがない場合、そのストリングはブランクになります。

この属性の値を判別するには、MQINQ 呼び出しで `CACLWD` セレクターを使用します。

IBM i IBM i での ClusterWorkloadData (32 バイトの文字ストリング)

クラスター・ワークロード出口についてのユーザー・データ。

これは、クラスター・ワークロード出口が呼び出されたとき、この出口に引き渡される 32 バイトのユーザー定義文字ストリングです。出口に引き渡すデータがない場合、そのストリングはブランクになります。

この属性の値を判別するには、MQINQ 呼び出しで CACLWD セレクターを使用します。

IBM i IBM i での *ClusterWorkloadExit* (20 バイトの文字ストリング)

クラスター・ワークロード管理のためのユーザー出口の名前。

この名前がブランクでない場合、メッセージがクラスター・キューに書き込まれるか、あるクラスター送信側キューから別のクラスター送信側キューに移動されるたびに、この出口が呼び出されます。その後、この出口は、メッセージ宛先としてキュー・マネージャーによって選択されたキュー・インスタンスを受け入れるか、または別のキュー・インスタンスを選択することができます。

この属性の値を判別するには、MQINQ 呼び出しで CACLWX セレクターを使用します。属性の長さは、LNEXN で指定します。

IBM i IBM i での *ClusterWorkloadLength* (10 桁の符号付き整数)

クラスター・ワークロード出口に引き渡されるメッセージ・データの最大長。

これは、クラスター・ワークロード出口に引き渡されるメッセージ・データの最大長です。出口に引き渡されるデータの実際の長さは、以下の値のうちで最小の値です。

- メッセージの長さ。
- キュー・マネージャーの **MaxMsgLength** 属性。
- **ClusterWorkloadLength** 属性。

この属性の値を判別するには、MQINQ 呼び出しで IACLWL セレクターを使用します。

IBM i IBM i での *CodedCharSetId* (10 桁の符号付き整数)

コード化文字セット ID。

これは、オブジェクトの名前やキュー作成の日付および時刻など、MQI 内に定義されているすべての文字ストリング・フィールドで、キュー・マネージャーが使用する文字セットを定義します。文字セットは、オブジェクト名で有効な文字を指定するために 1 バイト文字を備えている必要があります。メッセージ内に取り込まれるアプリケーション・データには適用されません。値は環境によって異なります。

- IBM i では、この値は、キュー・マネージャーの最初の作成時に環境に設定される値です。

この属性の値を判別するには、MQINQ 呼び出しで IACCSI セレクターを使用します。

IBM i IBM i での *CommandEvent* (整数)

コマンドが発行されたときにメッセージがローカル・キューに書き込まれるかどうかを制御します。

これは、コマンドが発行されるたびに、メッセージが新しいイベント・キュー **SYSTEM.ADMIN.COMMAND.EVENT** に書き込まれるかどうかを制御します。この機能は、コマンド・トラッキング通知および問題診断に役立ちます。CommandEvent キュー・マネージャー属性を照会するには、以下のいずれかの値を持つ新規の属性セレクター **iacev** を使用します。

- **EVRENA** - すべての成功コマンドについて、コマンド・イベント・メッセージが生成され、このキューに書き込まれます。
- **EVND** - **DISPLAY (MQSC)** コマンド以外のすべての成功コマンドと **Inquire (PCF)** コマンドについて、コマンド・イベント・メッセージが生成され、このキューに書き込まれます。
- **EVRDIS** - コマンド・イベント・メッセージは生成されたり、またはキューに書き込まれたりしません (これはキュー・マネージャーの初期デフォルト値です)。

この属性の値を判別するには、MQINQ 呼び出しで **CMDEV** セレクターを使用します。

IBM i IBM i での *CommandInputQName* (48 バイトの文字ストリング)

コマンド入力キュー名。

CommandInputQName は、ローカル・キュー・マネージャーで定義されているコマンド入力キューの名前です。ユーザーがコマンドを送ることができるキューです (ただし、アプリケーションがその許可を持っている場合)。キューの名前は以下のように環境によって変わります。

- IBM i では、キューの名前は SYSTEM.ADMIN.COMMAND.QUEUE で、キューに送信できるのは PCF コマンドのみです。ただし、MQSC コマンドをタイプ CMESC の PCF コマンド内に取り込んだ場合は、MQSC コマンドをこのキューに送信できます。Escape コマンドの詳細については、[Escape](#) を参照してください。

この属性の値を判別するには、MQINQ 呼び出しで CACMDQ セレクターを使用します。この属性の長さは LNQN で指定します。

IBM i IBM i での CommandLevel (10 桁の符号付き整数)

コマンド・レベル。これは、キュー・マネージャーによってサポートされるシステム制御コマンドのレベルを示します。

レベルは次のいずれかの値です。

CML800

レベル 800 のシステム制御コマンド。

値は次のアプリケーションから戻されます。

- IBM MQ for IBM i
 - バージョン 8.0

CML900

レベル 900 のシステム制御コマンド。

値は次のアプリケーションから戻されます。

- IBM MQ for IBM i
 - バージョン 9.0

CML910

レベル 910 のシステム制御コマンド。

値は次のアプリケーションから戻されます。

- IBM MQ for IBM i
 - バージョン 9.1

CML920

システム制御コマンドのレベル 920。

値は次のアプリケーションから戻されます。

- IBM MQ for IBM i
 - バージョン 9.2

CML930

システム制御コマンドのレベル 930。

値は次のアプリケーションから戻されます。

- IBM MQ for IBM i
 - バージョン 9.3

特定の **CommandLevel** 属性の値に対応するシステム制御コマンドのセットは、**Platform** 属性の値によって異なります。このため、サポートされるシステム制御コマンドを調べるには、両方の属性を使用する必要があります。

この属性の値を判別するには、MQINQ 呼び出しで IACMDL セレクターを使用します。

IBM i での ConfigurationEvent

構成イベントが生成されて、SYSTEM.ADMIN.CONFIG.EVENT キューのデフォルト・オブジェクトに送信されるかどうかを制御します。

ConfigurationEvent 属性の値は以下のいずれかです。

- EVRENA
- EVRDIS

ConfigurationEvent 属性が EVRENA に設定されており、特定のコマンドが runmqsc または PCF によって正常に発行された場合、構成イベントは生成され、SYSTEM.ADMIN.CONFIG.EVENT キューに送信されます。alter コマンドによって関係するオブジェクトが変更されない場合でも、以下のコマンドのイベントは発行されます。構成イベントが生成および送信されるコマンドは、以下のとおりです。

- DEFINE/ALTER AUTHINFO
- DEFINE/ALTER CHANNEL
- DEFINE/ALTER NAMELIST
- DEFINE/ALTER PROCESS
- DEFINE/ALTER QLOCAL (一時動的キューである場合を除く)
- DEFINE/ALTER QMODEL/QALIAS/QREMOTE
- DELETE AUTHINFO
- DELETE CHANNEL
- DELETE NAMELIST
- DELETE PROCESS
- DELETE QLOCAL (一時動的キューである場合を除く)
- DELETE QMODEL/QALIAS/QREMOTE
- ALTER QMGR (CONFIGEV 属性が使用不可になっており、使用可能に変更されない場合を除く)
- REFRESH QMGR
- MQSET 呼び出し (一時動的キュー用を除く)。

以下の環境では、イベントは生成されません (使用可能な場合でも)。

- コマンドまたは MQSET 呼び出しが失敗する。
- キュー・マネージャーがイベント・メッセージをイベント・キューに書き込むことができない。コマンドは依然として正常に完了されます。
- 一時動的キュー。
- 内部属性が直接または暗黙的に変更された (MQSET またはコマンドによってではなく)。これは、TRIGGER、CURDEPTH、IPPROCS、OPPROCS、QDPHIEV、QDPLOEV、QDPMAXEV、QSVCIEV に影響します。
- 構成イベント・キューが変更された場合。ただし、リフレッシュが要求された場合には、その変更のイベント・メッセージが生成されます。
- コマンド REFRESH/RESET CLUSTER および RESUME/SUSPEND QMGR による変更のクラスター化。
- キュー・マネージャーの作成または削除。

IBM i での DeadLetterQName (48 バイトの文字ストリング)

送達不能 (未配布メッセージ) キューの名前。

これは、ローカル・キュー・マネージャー上に定義されたキューの名前です。メッセージは、正しい宛先に経路指定されない場合に、このキューに送られます。

例えば、次の場合に、このキューにメッセージが書き込まれます。

- メッセージがキュー・マネージャーに着信したが、宛先のキューが、そのキュー・マネージャーではまだ定義されていない。

- メッセージがキュー・マネージャーに着信したが、宛先のキューがそのメッセージを受信できない。次のような理由が考えられます。
 - キューが満杯である。
 - 書き込み要求が使用禁止になっている。
 - 送信側のノードが、キューにメッセージを書き込む許可を、持っていない。

アプリケーションは、送達不能キューにもメッセージを書き込むことができます。

レポート・メッセージは、通常のメッセージと同様に扱われます。レポート・メッセージをその宛先キュー (通常、元のメッセージのメッセージ記述子の *MDRQ* フィールドによって指定されるキュー) に送達できない場合、レポート・メッセージは送達不能 (未配布メッセージ) キューに入れられます。

注: 満了時間を経過したメッセージ (1114 ページの『*IBM i*での MQMD (メッセージ記述子)』の *MDEXP* フィールドを参照) は、メッセージが廃棄されたとき、このキューに転送されません。ただし、送信側アプリケーションにより要求される場合は、満了レポート・メッセージ (ROEXP) が生成され、*MDRQ* キューに送信されます。

書き込み要求を発行したアプリケーションが、MQPUT または MQPUT1 呼び出しの結果、戻された理由コード (例えば、書き込み要求が使用禁止になっているローカル・キューにメッセージが入れられた) によって定期的に問題を通知された場合、メッセージは送達不能 (未配布メッセージ) キューに書き込まれません。

送達不能 (未配布メッセージ) キューのメッセージには、そのメッセージのアプリケーションのメッセージ・データに MQDLH 構造体の接頭部が付けられた情報があるものもあります。この構造体には、そのメッセージが送達不能 (未配布メッセージ) キューに書き込まれた理由を示す追加情報が入っています。この構造体の詳細については、1068 ページの『MQDLH (デッドレターヘッダー) on IBM i』を参照してください。

このキューは、**Usage** 属性が USNORM のローカル・キューであることが必要です。

送達不能 (未配布メッセージ) キューがキュー・マネージャーによってサポートされていない、あるいは送達不能キューが定義されていない場合は、名前はすべてブランクです。すべての IBM MQ キュー・マネージャーは送達不能 (未配布メッセージ) キューをサポートしますが、デフォルトでは、送達不能キューは定義されません。

送達不能 (未配布メッセージ) キューが定義されていないか、満杯になっているか、あるいは他のなんらかの理由で使用不可になっている場合には、メッセージ・チャネル・エージェントによって送達不能キューに転送されるはずのメッセージが代わりに伝送キューに保持されます。

この属性の値を判別するには、MQINQ 呼び出しで CADLQ セレクターを使用します。この属性の長さは LNQN で指定します。

DefClusterXmitQueueType (10 桁の符号付き整数)

DefClusterXmitQueueType 属性は、cluster-sender チャネルがメッセージを受信し、cluster-receiver チャネルに送信する際に、デフォルトでどの送信キューが選択されるかを制御します。

DefClusterXmitQueueType の値は MQCLXQ_SCTQ または MQCLXQ_CHANNEL です。

MQCLXQ_SCTQ

すべてのクラスター送信側チャネルは、メッセージを SYSTEM.CLUSTER.TRANSMIT.QUEUE から送信します。伝送キューに入れられたメッセージの *correlID* は、メッセージの宛先のクラスター送信側チャネルを示します。

SCTQ は、キューマネージャが定義されたときに設定されます。

MQCLXQ_CHANNEL

各クラスター送信側チャネルは、別の伝送キューからメッセージを送信します。各伝送キューは、永続的な動的キューとしてモデル・キュー SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE から作成されます。

キュー・マネージャー属性 DefClusterXmitQueueType を CHANNEL に設定すると、デフォルト構成は変更され、クラスター送信側チャネルが個々のクラスター伝送キューと関連付けられるようになります。伝送キューは、モデル・キュー SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE から作成される永続的に動的なキューです。各伝送キューは 1 つのクラスター送信側チャネルに関連付けられます。1 つのクラスター

一送信側チャンネルが1つのクラスター伝送キューにサービスを提供するため、伝送キューにも1つのクラスター内の1つのキュー・マネージャーへのメッセージだけが入ります。クラスター内の各キュー・マネージャーが使用するクラスター・キューが1つだけになるように構成することもできます。この場合、キュー・マネージャーから各クラスター・キューへのメッセージ・トラフィックは、それぞれ他のキューへのメッセージとは別に転送されます。

値を照会するには、MQINQ を呼び出すか、または MQIA_DEF_CLUSTER_XMIT_Q_TYPE セレクターを設定して Inquire Queue Manager (MQCMD_INQUIRE_Q_MGR) の PCF コマンドを送信します。値を変更するには、MQIA_DEF_CLUSTER_XMIT_Q_TYPE セレクターを設定して Change Queue Manager (MQCMD_CHANGE_Q_MGR) の PCF コマンドを送信します。

関連資料

[Change Queue Manager](#)

[Inquire Queue Manager](#)

[1316 ページの『IBM i での MQINQ \(オブジェクト属性の照会\)』](#)

MQINQ 呼び出しは、オブジェクトの属性が入っている整数の配列と一連の文字ストリングを戻します。

▶ IBM i IBM i での DefXmitQName (48 バイトの文字ストリング)

デフォルト伝送キュー名。

使用する伝送キューが特に示されていない場合、リモート・キュー・マネージャーに対するメッセージの伝送に使用される伝送キューの名前です。

デフォルトの伝送キューがない場合には、名前はすべてブランクです。この属性の初期値はブランクです。

この属性の値を判別するには、MQINQ 呼び出しで CADXQN セレクターを使用します。この属性の長さは LNQN で指定します。

▶ IBM i IBM i での DistLists (10 桁の符号付き整数)

配布リスト・サポート。

これは、ローカル・キュー・マネージャーが MQPUT および MQPUT1 呼び出しにある配布リストをサポートしているかどうかを示します。これは、以下の値のいずれかになります。

DLSUPP

配布リストがサポートされています。

DLNSUP

配布リストはサポートされていません。

この属性の値を判別するには、MQINQ 呼び出しで IADIST セレクターを使用します。

▶ IBM i IBM i での InhibitEvent (10 桁の符号付き整数)

禁止 (読み取りおよび書き込み禁止) イベントが生成されるかどうかを制御します。

これは、以下の値のいずれかになります。

EVRDIS

イベント報告は無効です。

EVRENA

イベント報告は有効です。

イベントの詳細については、[モニターとパフォーマンス](#)を参照してください。

この属性の値を判別するには、MQINQ 呼び出しで IAINHE セレクターを使用します。

▶ IBM i IBM i での LocalEvent (10 桁の符号付き整数)

ローカル・エラー・イベントが生成されるかどうかを制御する。

値は、次のいずれか1つです。

EVRDIS

イベント報告は無効です。

EVRENA

イベント報告は有効です。

イベントの詳細については、[イベント・モニター](#)を参照してください。

この属性の値を判別するには、MQINQ 呼び出しで IALCLE セレクターを使用します。

IBM i IBM i での LoggerEvent (10 桁の符号付き整数)

リカバリー・ロガー・イベントが生成されるかどうかを制御します。

これは、以下の値のいずれかになります。

ENABLED

ロガー・イベントを生成します。

DISABLED

ロガー・イベントは生成されません。これがキュー・マネージャーの初期デフォルト値です。

イベントの詳細については、[モニターとパフォーマンス](#)を参照してください。

IBM i IBM i での MaxHandles (10 桁の符号付き整数)

ハンドルの最大数です。

任意の 1 つのタスクが並行して使用できるオープン・ハンドルの最大数です。ある 1 つのキュー (またはキュー以外のオブジェクト) に対する MQOPEN 呼び出しが正常に実行されるたびに、ハンドルが 1 つ使用されます。オープンしたオブジェクトをクローズすると、そのハンドルを再使用できるようになります。ただし、配布リストをオープンしたときには、配布リスト内のそれぞれのキューに個別のハンドルが割り振られるため、MQOPEN 呼び出しによって、配布リスト内にあるキューと同じ数のハンドルが使用されます。*MaxHandles* の適切な値を判断するときには、このことを考慮する必要があります。

MQPUT1 呼び出しでは、処理の一部として MQOPEN 呼び出しが実行されます。このため、MQOPEN と同じ数のハンドルが使用されますが、MQPUT1 でハンドルが使用されるのは、その MQPUT1 呼び出し自身が実行されている間のみです。

値の範囲は、1 から 999 999 999 です。IBM i の場合、デフォルト値は 256 です。

この属性の値を判別するには、MQINQ 呼び出しで IAMHND セレクターを使用します。

IBM i IBM i での MaxMsgLength (10 桁の符号付き整数)

メッセージの最大長 (バイト数)。

キュー・マネージャーで処理できる物理メッセージの最大長です。ただし、**MaxMsgLength** キュー・マネージャー属性は、**MaxMsgLength** キュー属性とは別に設定できるため、キューに入れることのできる物理メッセージの最大長は、これら 2 つの値の小さい方の値になります。

キュー・マネージャーがセグメント化をサポートしている場合は、アプリケーションで MFSEGA フラグを MQMD に指定しているときに限って、これら 2 つの **MaxMsgLength** 属性の小さい方の値より長い論理メッセージをアプリケーションから書き込むことができます。そのフラグを指定した場合、論理メッセージの長さの上限は 999 999 999 バイトですが、オペレーティング・システムによって、またはアプリケーションが実行されている環境によってリソースが制約される結果、通常、これより小さい値になります。

MaxMsgLength 属性の下限は 32 KB (32 768 バイト) です。IBM i の場合、最大メッセージ長は 100 MB (104 857 600 バイト) です。

この属性の値を判別するには、MQINQ 呼び出しで IAMLEN セレクターを使用します。

IBM i IBM i での MaxPriority (10 桁の符号付き整数)

最高の優先順位。

これは、キュー・マネージャーによってサポートされる最高のメッセージ優先順位です。優先順位の範囲は、ゼロ (最低) から *MaxPriority* (最高) までです。

この属性の値を判別するには、MQINQ 呼び出しで IAMPRI セレクターを使用します。

IBM i での MaxUncommittedMsgs (10 桁の符号付き整数)

1 つの作業単位のコミットされていないメッセージの最大数。

これは、1 つの作業単位に存在できるコミットされていないメッセージの最大数です。コミットされていないメッセージの数は、現行の作業単位が開始されてからの以下の合計です。

- PMSYP オプションでアプリケーションが書き込んだメッセージ。
- GMSYP オプションでアプリケーションが取り出したメッセージ。
- トリガー・メッセージおよび COA レポート・メッセージのうち、PMSYP オプションで書き込んだメッセージにキュー・マネージャーが生成したメッセージ。
- COD レポート・メッセージのうち、GMSYP オプションで検索したメッセージにキュー・マネージャーが生成したメッセージ。

以下のメッセージはコミットされていないメッセージとしては数えません。

- 作業単位外でアプリケーションが書き込みまたは検索したメッセージ
- トリガー・メッセージまたは COA/COD レポート・メッセージのうち、作業単位外で書き込んだ、または検索したメッセージの結果としてキュー・マネージャーが生成したメッセージ
- キュー・マネージャーが生成した満了レポート・メッセージ (満了レポート・メッセージを発生させた呼び出しに GMSYP が指定された場合も含まれます)。
- キュー・マネージャーが生成したイベント・メッセージ (イベント・メッセージを発生させた呼び出しに PMSYP または GMSYP が指定された場合も含まれます)。

注:

1. 例外レポート・メッセージは、メッセージ・チャネル・エージェント (MCA) またはアプリケーションが生成します。したがって、これもアプリケーションが書き込んだ、または検索した通常のメッセージと同様に扱われます。
2. メッセージまたはセグメントが PMSYP オプションを指定して書き込まれると、その書き込みの結果として実際に発行される物理メッセージの数にかかわらず、コミットされていないメッセージの数が 1 つずつ増分されます。(キュー・マネージャーがメッセージまたはセグメントを分割する必要がある場合、複数の物理メッセージが発行されることがあります)。
3. 配布リストが PMSYP オプションを指定して書き込まれると、コミットされていないメッセージの数は、生成される物理メッセージごとに 1 つずつ増分されます。これは、配布リスト中の宛先の数と同じである必要はありません。

この属性の下限は 1 で、上限は 999 999 999 です。

この属性の値を判別するには、MQINQ 呼び出しで IAMUNC セレクターを使用します。

IBM i での PerformanceEvent (10 桁の符号付き整数)

パフォーマンス関連イベントが生成されるかどうかを制御します。

PerformanceEvent には、以下のいずれかの値を指定できます。

EVRDIS

イベント報告は無効です。

EVRENA

イベント報告は有効です。

イベントの詳細については、[イベント・モニター](#)を参照してください。

この属性の値を判別するには、MQINQ 呼び出しで IAPFME セレクターを使用します。

IBM i での Platform (10 桁の符号付き整数)

キュー・マネージャーが実行されているプラットフォーム。

これは、キュー・マネージャーが実行されているオペレーティング・システムを示します。その値は、以下のものです。

PL400

IBM i.

IBM i IBM i での PubSubMode (10 桁の符号付き整数)

パブリッシュ/サブスクライブ・エンジンおよびキューに入れられたパブリッシュ/サブスクライブ・インターフェースが実行中なので、アプリケーション・プログラミング・インターフェースと、キューに入れられたパブリッシュ/サブスクライブ・インターフェースによってモニターされているキューを使用して、アプリケーションがパブリッシュ/サブスクライブできるかどうか。

これは、以下の値のいずれかになります。

PSMCP

パブリッシュ/サブスクライブ・エンジンが実行中。したがって、アプリケーション・プログラミング・インターフェースを使用してパブリッシュ/サブスクライブできます。キュー・パブリッシュ/サブスクライブ・インターフェースは実行されていないため、キュー・パブリッシュ/サブスクライブ・インターフェースがモニターするキューに書き込まれるメッセージは処理されません。この設定は、キューに入れられたパブリッシュ/サブスクライブ・インターフェースの通常の読み取り元と同じキューを読み取る必要があるため、このキュー・マネージャーを使用する WebSphere Message Broker V6 以前のバージョンとの互換用に使用されます。

PSMDS

パブリッシュ/サブスクライブ・エンジンとキュー・パブリッシュ/サブスクライブ・インターフェースはどちらも実行されていません。したがって、アプリケーション・プログラミング・インターフェースを使用してパブリッシュ/サブスクライブできません。キュー・パブリッシュ/サブスクライブ・インターフェースがモニターするキューに書き込まれるパブリッシュ/サブスクライブ・メッセージは処理されません。

PSMEN

パブリッシュ/サブスクライブ・エンジンとキュー・パブリッシュ/サブスクライブ・インターフェースはどちらも実行されています。したがって、アプリケーション・プログラミング・インターフェースと、キューに入れられたパブリッシュ/サブスクライブ・インターフェースによってモニターされているキューを使用してパブリッシュ/サブスクライブできます。これがキュー・マネージャーの初期デフォルト値です。

この属性の値を判別するには、MQINQ 呼び出しで PSMODE セレクターを使用します。

IBM i IBM i での QMgrDesc (64 バイトの文字ストリング)

キュー・マネージャーの記述。

これは、コメントを記述するために使用できるフィールドです。フィールドの内容はキュー・マネージャーにとって重要なものではありませんが、表示できる文字以外は使用しないでください。フィールドにヌル文字を入れることはできません。また、必要に応じて、右側がブランクで埋められます。DBCS をインストール済みの環境では、このフィールドに DBCS 文字を入れることができます (最大フィールド長として 64 バイトが適用されます)。

注: このフィールドに、(CodedCharSetId キュー・マネージャー属性で定義されている) キュー・マネージャーの文字セットに含まれていない文字が含まれている場合、このフィールドが別のキュー・マネージャーに送信されると、それらの文字が正しく変換されない可能性があります。

IBM i の場合、デフォルト値はブランクです。

この属性の値を判別するには、MQINQ 呼び出しで CAQMD セレクターを使用します。この属性の長さは、LNQMD で指定されます。

IBM i IBM i での QMgrIdentifier (48 バイトの文字ストリング)

キュー・マネージャーの内部的に生成される固有の ID。

これは、キュー・マネージャーの内部的に生成される固有の名前です。

この属性の値を判別するには、MQINQ 呼び出しで CAQMID セレクターを使用します。属性の長さは、LNQMID で指定します。

▶ IBM i IBM i での QMgrName (48 バイトの文字ストリング)

キュー・マネージャー名。

これは、ローカル・キュー・マネージャーの名前、つまりアプリケーションが接続されているキュー・マネージャーの名前です。

この名前の先頭の 12 文字は、固有のメッセージ ID を生成するために使用されます (1114 ページの『IBM i での MQMD (メッセージ記述子)』で説明している MDMID フィールドを参照してください)。したがって、相互通信するキュー・マネージャーには、キュー・マネージャー・ネットワーク内でのメッセージ ID がそれぞれ固有なものになるように、先頭の 12 文字をそれぞれ区別できるような名前を付ける必要があります。

この属性の値を判別するには、MQINQ 呼び出しで CAQMN セレクターを使用します。この属性の長さは、LNQMN で指定されます。

▶ IBM i IBM i での RemoteEvent (10 桁の符号付き整数)

リモート・エラー・イベントが生成されるかどうかを制御します。

値は、次のいずれか 1 つです。

EVRDIS

イベント報告は無効です。

EVRENA

イベント報告は有効です。

イベントの詳細については、[イベント・モニター](#)を参照してください。

この属性の値を判別するには、MQINQ 呼び出しで IARMTE セレクターを使用します。

▶ IBM i IBM i での RepositoryName (48 バイトの文字ストリング)

このキュー・マネージャーがリポジトリ・サービスを提供するクラスターの名前。

これは、このキュー・マネージャーがリポジトリ・マネージャーのサービスを提供するクラスターの名前です。キュー・マネージャーが複数のクラスターにこのサービスを提供する場合は、それらのクラスターを識別する名前リスト・オブジェクトの名前を *RepositoryNameList* において指定し、*RepositoryName* はブランクにします。*RepositoryName* および *RepositoryNameList* の少なくとも 1 つがブランクでなければなりません。

この属性の値を判別するには、MQINQ 呼び出しで CARPN セレクターを使用します。この属性の長さは、LNQMN で指定されます。

▶ IBM i IBM i での RepositoryNameList (48 バイトの文字ストリング)

このキュー・マネージャーがリポジトリ・サービスを提供するクラスターの名前が入った名前リスト・オブジェクトの名前。

これは、このキュー・マネージャーがリポジトリ・マネージャーのサービスを提供するクラスターの名前を含む名前リスト・オブジェクトの名前です。キュー・マネージャーが 1 つのクラスターだけにこのサービスを提供する場合、名前リスト・オブジェクトには 1 つの名前だけが含まれます。代わりに *RepositoryName* を使用して、そのクラスターの名前を指定できます。この場合、*RepositoryNameList* はブランクにします。*RepositoryName* および *RepositoryNameList* の少なくとも 1 つがブランクでなければなりません。

この属性の値を判別するには、MQINQ 呼び出しで CARPNL セレクターを使用します。この属性の長さは LNNLN によって指定されます。

▶ IBM i IBM i での SSLEvent (文字ストリング)

TLS イベントが生成されるかどうかを決定します。

値は、次のいずれか 1 つです。

- EVRENA (MQINQ/PCF/config event) ENABLED (MQSC): TLS イベントが生成されます (つまり、RC2371 イベントが生成されます)。

- EVRDIS (MQINQ/PCF/config event) DISABLED (MQSC): TLS イベントは生成されません。これがキュー・マネージャーの初期デフォルト値です。

この属性値を調べるには、MQINQ 呼び出しで IASSLE セレクターを使用します。

IBM i IBM i での SSLKeyResetCount (整数)

共通鍵が再折衝される前に、TLS 会話内で送受信される暗号化されていないバイトの総数を決定します。バイト数には、メッセージ・チャンネル・エージェント (MCA) によって送信される制御情報が含まれます。

この値は、このキュー・マネージャーから通信を開始する TLS チャンネル MCA (つまり、送信側受信側チャンネル・ペアの送信側チャンネル MCA) によってのみ使用されます。

この属性の値が 0 より大きく、チャンネル・ハートビートがチャンネルで有効になっている場合、チャンネル・ハートビートの後にデータが送受信される前に共通鍵も再折衝されます。次の共通鍵の再折衝までのバイト・カウントは、それぞれの再折衝が正常に行われた後でリセットされます。

値の範囲は 0 から 999 999 999 です。この属性の値を 0 にすると、共通鍵が再折衝されないことを示します。TLS 秘密鍵のリセット・カウントを 1 バイトから 32 KB の範囲で指定すると、TLS チャンネルは 32 KB の秘密鍵リセット・カウントを使用します。これは、TLS 秘密鍵リセット値が小さい場合に生じる、過剰な鍵リセットによる処理コストを避けるためです。

SSL サーバーが IBM MQ キュー・マネージャーで、共通鍵のリセットとチャンネル・ハートビートの両方が有効になっている場合、各チャンネルのハートビートの直後に再折衝が行われます。

この属性値を調べるには、MQINQ 呼び出しで IASSRC セレクターを使用します。

IBM i IBM i での StartStopEvent (10 桁の符号付き整数)

開始イベントおよび停止イベントが生成されるかどうかを制御します。

この属性には、以下のいずれかの値を使用できます。

EVRDIS

イベント報告は無効です。

EVRENA

イベント報告は有効です。

イベントの詳細については、[イベント・モニター](#)を参照してください。

この属性の値を判別するには、MQINQ 呼び出しで IASSE セレクターを使用します。

IBM i IBM i での SyncPoint (10 桁の符号付き整数)

同期点の可用性。

これは、ローカル・キュー・マネージャーが複数の作業単位をサポートし、MQGET、MQPUT、および MQPUT1 呼び出しによる同期化をサポートするかどうかを示します。

SPAVL

作業単位および同期化が可能。

SPNAVL

作業単位および同期化が不可。

この属性の値を判別するには、MQINQ 呼び出しで IASYNL セレクターを使用します。

IBM i IBM i での TraceRouteRecording (10 桁の符号付き整数)

これは、キュー・マネージャーを介してフローするときに、メッセージに関する情報を記録するかどうかを制御します。

値は、次のいずれか 1 つです。

- RECDD: トレース経路メッセージに追加することはできない。
- RECDQ: メッセージは固定された名前付きキューに書き込まれる。
- RECDM: メッセージの使用を決定する (これは初期のデフォルト設定です)。

トレース経路メッセージがシステム内に残らないようにするには、0 より大きい満了時間値を設定し、RODISC レポート・オプションを指定します。レポート・メッセージまたは応答メッセージがシステム内に残らないようにするには、レポート・オプション ROPDAE を設定します。詳しくは、[1433 ページの『IBM iでのレポート・オプションおよびメッセージ・フラグ』](#)を参照してください。

この属性の値を判別するには、MQINQ 呼び出しで IATRGI セレクターを使用します。

IBM i **IBM i での TreeLifeTime (10 桁の符号付き整数)**

非管理トピックの存続期間 (秒数)。

非管理トピックとは、アプリケーションが管理ノードとして存在しないトピック・ストリングにパブリッシュしたり、それをサブスクライブしたりするときに作成されるものです。このパラメーターでは、この非管理ノードにアクティブなサブスクリプションがなくなった時点から、キュー・マネージャーがそのノードを除去するまでの待機時間を指定します。キュー・マネージャーがリサイクルされた後は、永続サブスクリプションによって使用中の非管理トピックのみが残ります。

0 から 604 000 の範囲の値を指定します。値 0 は、非管理トピックがキュー・マネージャーによって削除されないことを意味します。キュー・マネージャーの初期デフォルト値は 1800 です。

この属性の値を判別するには、MQINQ 呼び出しで IATRLFT セレクターを使用します。

IBM i **IBM i での TriggerInterval (10 桁の符号付き整数)**

トリガー・メッセージの間隔。

これは、トリガー・メッセージの数を制限するために使用される時間間隔 (単位はミリ秒) です。*TriggerType* が TTFIRST の場合にのみ関連があります。この場合、通常、適切なメッセージがキューに届き、キューがすでに空であるときにのみ、トリガー・メッセージは生成されます。ただし、特定の状況の下では、キューが空でなくとも TTFIRST のトリガー操作で、追加のトリガー・メッセージが生成されることがあります。これらの追加のトリガー・メッセージは、*TriggerInterval* ミリ秒ごとよりも頻繁に生成されることはありません。

トリガー操作の詳細については、[チャンネルのトリガー操作](#)を参照してください。

値の範囲は 0 から 999 999 999 です。デフォルト値は 999 999 999 です。

この属性の値を判別するには、MQINQ 呼び出しで IATRGI セレクターを使用します。

IBM i の戻りコード (ILE RPG)

この情報は、MQI および MQAI に関連した戻りコードについて説明します。

戻りコードは以下と関連しています。

- プログラマブル・コマンド・フォーマット (PCF) コマンドについては、[プログラム可能なコマンド形式のリファレンス](#)に記載されています。
- C++ 呼び出しについては、[C++の使用](#)にリストされています。

呼び出しが行われるたびに、その呼び出しが成功したか失敗したかを示すための完了コードと理由コードが、キュー・マネージャーまたは出口ルーチンによって戻されます。

特に指定がある場合を除いて、エラーが特定の順序で検査されると想定してアプリケーションを作成しないでください。呼び出しによって複数の完了コードまたは理由コードが生じた場合、どのエラーが報告されるかは、実現方法 (キュー・マネージャーか、あるいは出口ルーチンか) によって異なります。

IBM i の完了コード (ILE RPG)

完了コード・パラメーター (CMPCOD) は、その呼び出しが成功したか、部分的に完了したか、または失敗したかを、呼び出し元で素早く知ることができるようにするためのものです。

CCOK

(他のプラットフォームでの MQCC_OK)

正常終了。

呼び出しはすべて完了しました。すべての出力パラメーターが設定されました。この場合、**REASON** パラメーターの値は常に RCNONE です。

CCWARN

(他のプラットフォームでの MQCC_WARN)

警告 (部分完了)。

呼び出しは部分的に完了しました。CMPCOD および REASON 出力パラメーターの他にも、いくつかの出力パラメーターが設定されている場合があります。**REASON** パラメーターで部分的な完了についての情報がさらに分かります。

CCFAIL

(他のプラットフォームでの MQCC_FAIL)

呼び出し失敗。

呼び出しの処理は完了しませんでした。通常、キュー・マネージャーの状態は変わりません。例外は個々に示されます。CMPCOD および REASON 出力パラメーターが設定されました。その他のパラメーターは (特に断りがない限り) 変更されませんでした。

理由は、アプリケーション・プログラム内の障害である場合や、そのプログラムの外部の状態の結果である場合があります (ユーザーの権限が取り消された場合など)。**REASON** パラメーターでエラーについての情報がさらに分かります。

IBM i の理由コード (ILE RPG)

理由コード・パラメーター (REASON) は、完了コード・パラメーター (CMPCOD) をさらに詳しく説明するものです。

特に報告する理由がない場合には、RCNONE という値が戻されます。成功した呼び出しは、CCOK および RCNONE を戻します。

完了コードが CCWARN または CCFAIL の場合、キュー・マネージャーは常に識別した理由を報告してきます。その詳細は、各呼び出しの説明部分に表示されます。

ユーザー作成出口ルーチンが完了コードと理由を設定する場合には、これらの規則に必ず従ってください。さらに、ユーザー作成出口ルーチンによって定義された特殊な理由値は、キュー・マネージャーによって定義されている値と重複しないようにするために、ゼロ未満の値にしてください。出口ルーチンでは、該当する場合、キュー・マネージャーによってすでに定義されている理由を設定することができます。

理由コードは、以下のフィールドにも設定されます。

- MQDLH 構造体の DLREA フィールド
- MQMD 構造体の MDFB フィールド

理由コードの全リストについては、[API の完了コードと理由コード](#)を参照してください。

このリストの中で IBM i 理由コードを見つけるには、先頭の「RC」を除去します (例えば、RC2002 は 2002 になります)。また、次のように、完了コードは他のプラットフォームと同じように示されます。

表 850. IBM i および他のプラットフォームでの理由コード名	
IBM i	他のプラットフォーム
CCOK	MQCC_OK
CCWARN	MQCC_WARN
CCFAIL	MQCC_FAIL

IBM i の MQI オプションの妥当性検査に関する規則 (ILE RPG)

このトピックでは、MQOPEN、MQPUT、MQPUT1、MQGET、または MQCLOSE の各呼び出しから RC2046 理由コードが生成される状況について説明します。

IBM i での MQOPEN 呼び出し

MQOPEN 呼び出しのオプションについては、次の規則が適用されます。

- 以下のいずれかを 1 つ以上 指定する必要があります。
 - OOBROW
 - OOINPQ
 - OOINPX
 - OOINPS
 - OOINQ
 - OOOOUT
 - OOSSET
- 次のオプションのうち指定できるのは、1 つ のみです。
 - OOINPQ
 - OOINPX
 - OOINPS
- 次のオプションのうち指定できるのは、1 つ のみです。
 - OOBNDQ
 - OOBNDN
 - OOBNDQ

注: 前述のオプションは相互に排他的です。ただし、OOBNDQ の値がゼロのときは、その他の 2 つのバインド・オプションのいずれかと一緒にこのオプションを指定しても、理由コード RC2046 は戻されません。OOBNDQ は、プログラムの文書化を助けるために指定します。

- OOSAVA を指定する場合には、OOINP* オプションのうちのいずれかを指定することも必要です。
- OOSSET* または OOPAS* オプションのいずれかを指定した場合には、OOOOUT も指定しなければなりません。

IBM i での MQPUT 呼び出し

書き込みメッセージ・オプションについては、次の規則が適用されます。

- PMSYP と PMNSYP の組み合わせはできません。
- 次のオプションのうち指定できるのは、1 つ のみです。
 - PMDEFC
 - PMNOC
 - PMPASA
 - PMPASI
 - PMSETA
 - PMSETI
- PMALTU は使用できません (MQPUT1 呼び出しでのみ有効です)。

IBM i での MQPUT1 呼び出し

書き込みメッセージ・オプションについては、次のオプションを除いて、MQPUT 呼び出しの場合と同じ規則が適用されます。

- PMALTU を許可する。
- PMLOGO を許可しない。

IBM i での MQGET 呼び出し

読み取りメッセージ・オプションについては、次の規則が適用されます。

- 次のオプションのうち指定できるのは、1 つのみです。
 - GMNSYP
 - GMSYP
 - GMPSYP
- 次のオプションのうち指定できるのは、1 つのみです。
 - GMBRWF
 - GMBRWC
 - GMBRWN
 - GMMUC
- GMSYP を次のいずれかのオプションと一緒に指定することはできません。
 - GMBRWF
 - GMBRWC
 - GMBRWN
 - GMLK
 - GMUNLK
- GMPSYP を次のいずれかのオプションと一緒に指定することはできません。
 - GMBRWF
 - GMBRWC
 - GMBRWN
 - GMCMPM
 - GMUNLK
- GMLK を指定する場合は、次のオプションのいずれか 1 つも指定する必要があります。
 - GMBRWF
 - GMBRWC
 - GMBRWN
- GMUNLK を指定する場合は、次のオプションのみを指定できます。
 - GMNSYP
 - GMNWT

IBM i での MQCLOSE 呼び出し

- MQCLOSE 呼び出しのオプションについては、次の規則が適用されます。CODEL と COPURG の組み合わせはできません。
- 次のオプションのうち指定できるのは、1 つのみです。
 - COKPSB
 - CORMSB

IBM i での MQSUB 呼び出し

MQSUB 呼び出しのオプションについては、次の規則が適用されます。

- 以下のいずれかを 1 つ以上指定する必要があります。
- 以下のいずれかを 1 つ以上指定する必要があります。
 - SOALT
 - SORES
 - SOCRT
- 次のオプションのうち指定できるのは、1 つのみです。
 - SODUR
 - SONDUR

注記: 前述のオプションは相互に排他的です。ただし、SONDUR の値はゼロであるため、それを SODUR と一緒に指定しても、理由コード RC2046 は返されません。SONDUR は、プログラムの文書化を助けるために指定します。

- SOGRP と SOMAN の組み合わせはできません。
- SOGRP では SOCID を指定する必要があります。
- SOAUDID と SOFUID のうち、指定できるのは、1 つのみです。
- SONEWP と SOPUBR の組み合わせはできません。
- SONEWP は、SOCRT との組み合わせのみが可能です。
- 次のオプションのうち指定できるのは、1 つのみです。
 - SOWCHR
 - SOWTOP

IBM i でのマシン・エンコーディング

この情報は、メッセージ記述子の *MDENC* フィールドの構造について学習するために使用します。

メッセージ記述子の詳細については、[1114 ページの『IBM i での MQMD \(メッセージ記述子\)』](#)を参照してください。

MDENC フィールドは 32 ビットの整数で、4 つの別個のサブフィールドに分割されています。それらのサブフィールドは、それぞれ次に示すものを表しています。

- 2 進整数用のエンコード
- パック 10 進整数用のエンコード
- 浮動小数点用のエンコード
- 予約ビット

各サブフィールドは、サブフィールドに対応する位置に 1 のビット、それ以外の位置に 0 のビットを持つビット・マスクによって識別されます。これらのビットには、ビット 0 が最上位ビットになり、ビット 31 が最下位ビットになるように番号が付けられています。定義されているマスクは次のとおりです。

JAIMSK

2 進整数エンコードをマスクします。

このサブフィールドは、*MDENC* フィールド内のビット位置 28 から 31 を占めています。

ENDMSK

パック 10 進整数エンコードをマスクします。

このサブフィールドは、*MDENC* フィールド内のビット位置 24 から 27 を占めています。

ENFMSK

浮動小数点エンコードをマスクします。

このサブフィールドは、MDENC フィールド内のビット位置 20 から 23 を占めています。

ENRMSK

予約済みビットをマスクします。

このサブフィールドは、MDENC フィールド内のビット位置 0 から 19 を占めています。

IBM i IBM i での 2 進整数のエンコード

2 進整数のエンコードに有効な値。

2 進整数のエンコードとして有効な値は次のとおりです。

ENIUND

未定義整数のエンコード。

2 進整数は、定義されていないエンコードを用いて表されます。

ENINOR

標準の整数エンコード方式。

2 進整数は標準的な方法で表されます。

- 数値内の最下位バイトは、その数値内のバイトの中で最上位のアドレスを持っています。逆に、最上位バイトは最下位のアドレスを持っています。
- 各バイト内の最下位ビットは、その次に上位のアドレスを持つバイトに隣接しています。各バイト内の最上位ビットは、その次に下位のアドレスを持つバイトに隣接しています。

ENIREV

逆方向の整数エンコード方式。

2 進整数は ENINOR と同様に表現されますが、バイトは逆順に並べられます。各バイト内部のビットは、ENINOR と同様に並べられます。

IBM i IBM i でのパック 10 進整数のエンコード

パック 10 進整数のエンコードの有効値

パック 10 進整数のエンコードとして有効な値は次のとおりです。

ENDUND

未定義パック 10 進数のエンコード。

パック 10 進数は、定義されていないエンコードを用いて表されます。

ENDNOR

通常のパック 10 進数のエンコード。

パック 10 進整数は標準的な方法で表されます。

- 10 進数の印刷可能形式の各桁は、パック 10 進数では X'0' から X'9' までの 1 個の 16 進数で表現されます。16 進数の各桁はそれぞれ 4 ビットを占有します。したがって、パック 10 進数の各バイトは、印刷可能な数値形式では 2 桁の 10 進数字を表します。
- パック 10 進数内の最下位バイトは、最下位の 10 進数字が入っているバイトです。そのバイトの中で最上位の 4 ビットには最下位の 10 進数字が入っており、最下位の 4 ビットには符号が入っています。符号は、X'C' (正)、X'D' (負)、あるいは X'F' (無符号) です。
- 数値内の最下位バイトは、その数値内のバイトの中で最上位のアドレスを持っています。逆に、最上位バイトは最下位のアドレスを持っています。
- 各バイト内の最下位ビットは、その次に上位のアドレスを持つバイトに隣接しています。各バイト内の最上位ビットは、その次に下位のアドレスを持つバイトに隣接しています。

ENDREV

逆方向パック 10 進数のエンコード。

パック 10 進整数は ENDNOR と同様に表現されますが、バイトは逆順に並べられます。各バイト内部のビットは、ENDNOR と同様に並べられます。

▶ IBM i IBM i での浮動小数点エンコード

浮動小数点のエンコードの有効値

浮動小数点のエンコードとして有効な値は次のとおりです。

ENFUND

未定義浮動小数点のエンコード。

浮動小数点数は、定義されていないエンコードを用いて表されます。

ENFNOR

通常の IEEE (米国電気電子学会) 浮動小数点のエンコード。

浮動小数点数は、標準の IEEE 浮動小数点形式で表されます。バイトの配列は次のとおりです。

- 小数部の中の最下位バイトは、その数値内のすべてのバイトのうちで最上位アドレスを持っています。指数が入っているバイトは最下位アドレスを持っています。
- 各バイト内の最下位ビットは、その次に上位のアドレスを持つバイトに隣接しています。各バイト内の最上位ビットは、その次に下位のアドレスを持つバイトに隣接しています。

IEEE 浮動小数点のエンコードについての詳細は、IEEE 標準 754 で参照できます。

ENFREV

逆方向 IEEE 浮動小数点エンコード。

浮動小数点数は ENFNOR と同様に表現されますが、バイトは逆順に並べられます。各バイト内部のビットは、ENFNOR と同様に並べられます。

ENF390

System/390 アーキテクチャーに基づく浮動小数点のエンコード。

浮動小数点数は、標準の System/390 浮動小数点形式を用いて表されます。これは、System/370 でも使用されます。

▶ IBM i IBM i でのエンコードの組み立て

MQMD に MDENC フィールドの値を構成するには、必要とされるエンコードを記述する関連定数を加算します。

ENI* エンコードの値の 1 つを、END* エンコードの値の 1 つおよび ENF* エンコードの値の 1 つのみと組み合わせてください。

▶ IBM i IBM i でのエンコードの分析

MDENC フィールドには、サブフィールドが含まれます。したがって、整数のエンコード、パック 10 進数のエンコード、または浮動小数点数のエンコードの検査を必要とするアプリケーションは、このトピックで説明する技法を使用する必要があります。

算術演算の使用

整数の算術演算を使用して、次の手順を実行してください。

1. 必須のエンコードのタイプに応じて、以下の値のいずれか 1 つを選択します。

- バイナリー整数エンコード方式では、1
- パック 10 進数エンコード方式では、16
- 浮動小数点エンコード方式では、256

値を A とします。

2. *MDENC* フィールドの値を A で割り、その結果を B と呼ぶことにします。
3. B を 16 で割り、その結果を C とします。
4. C に 16 を掛けて、B から引きます。その結果を D とします。
5. D に A を掛け、その結果を E とします。
6. E は必須のエンコードであり、そのエンコードのタイプに有効な値のそれぞれについて値が等しいかどうかをテストできます。

IBM i IBM i でのマシン・アーキテクチャー・エンコードの要約

以下の表に、マシン・アーキテクチャーのエンコードを要約します。

マシン・アーキテクチャーのエンコードについては、[1433 ページの表 851](#) に示されています。

表 851. マシン・アーキテクチャーのエンコードの要約			
マシン・アーキテクチャー	2 進整数のエンコード	パック 10 進整数のエンコード	浮動小数点エンコード
IBM i	normal	normal	通常 IEEE
Intel x86	逆方向	逆方向	逆方向 IEEE
PowerPC	normal	normal	通常 IEEE
System/390	normal	normal	System/390

IBM i IBM i でのレポート・オプションおよびメッセージ・フラグ

このトピックでは *MDREP* フィールドおよび *MDMFL* フィールドについて説明します。これらは、*MQGET*、*MQPUT*、および *MQPUT1* 呼び出しで指定されるメッセージ記述子 *MQMD* の一部です。

メッセージ記述子の詳細については、[1114 ページの『IBM i での MQMD \(メッセージ記述子\)』](#)を参照してください。この情報では、以下について説明しています。

- レポート・フィールドの構造と、キュー・マネージャーがレポート・フィールドを処理する方法
- アプリケーションによるレポート・フィールドの分析方法。
- メッセージ・フラグ・フィールドの構造

レポート・フィールドの構造

MDREP フィールドは 32 ビット長の整数で、別々の 3 つのサブフィールドに分かれています。

これらのサブフィールドは、次のオプションを識別します。

- ローカル・キュー・マネージャーによって認識されない場合に拒否されるレポート・オプション
- ローカル・キュー・マネージャーによって認識されない場合でも、常に受け入れられるレポート・オプション
- 他の特定の条件を満たした場合にのみ受け入れられるレポート・オプション

各サブフィールドは、サブフィールドに対応する位置に 1 のビット、それ以外の位置に 0 のビットを持つビット・マスクによって識別されます。サブフィールド内のビットは必ずしも隣接している必要はないことに注意してください。これらのビットには、ビット 0 が最上位ビットになり、ビット 31 が最下位ビットになるように番号が付けられています。サブフィールドを識別するために定義されているマスクは次のとおりです。

RORUM

拒否される、サポートされないレポート・オプションのためのマスク。

このマスクが示している *MDREP* フィールド内のビット位置は、ローカル・キュー・マネージャーがサポートしていないレポート・オプションが原因で *MQPUT* 呼び出しまたは *MQPUT1* 呼び出しが完了コード *CCFAIL* および理由コード *RC2061* で失敗する場所を示します。

このサブフィールドは、ビット位置 3、およびビット位置 11 から 13 までを占めます。

ROAUM

受け入れられているサポートされないレポート・オプションをマスクします。

このマスクが示している *MDREP* フィールド内のビット位置は、ローカル・キュー・マネージャーがサポートしていないレポート・オプションであっても *MQPUT* 呼び出しまたは *MQPUT1* 呼び出しで受け入れられる場所を示します。この場合は、完了コードの *CCWARN* と理由コードの *RC2104* が戻されます。

このサブフィールドは、ビット位置 0 から 2 まで、4 から 10 まで、および 24 から 31 までを占めます。

このサブフィールドには、以下のレポート・オプションがあります。

- ROCMTC
- RODLQ
- RODISC
- ROEXC
- ROEXCD
- ROEXCF
- ROEXP
- ROEXPD
- ROEXPF
- RONAN
- RONMI
- RONONE
- ROPAN
- ROPCI
- ROPMI

ROAUXM

ある特定の環境でのみ受け入れられる、サポートされないレポート・オプションのためのマスク。

このマスクが示している *MDREP* フィールド内のビット位置は、ローカル・キュー・マネージャーがサポートしていないレポート・オプションであっても、次の条件を両方とも満たしている場合に *MQPUT* 呼び出しまたは *MQPUT1* 呼び出しで受け入れられる場所を示します。

- メッセージの宛先がリモート・キュー・マネージャーである。
- アプリケーションが、ローカル伝送キューに直接メッセージを書き込んでいない (つまり、*MQOPEN* または *MQPUT1* 呼び出しに指定されたオブジェクト記述子の *ODMN* および *ODON* フィールドにより識別されるキューが、ローカル伝送キューではない)。

これらの条件が満たされていれば、完了コードの *CCWARN* と理由コードの *RC2104* が戻され、満たされていなければ、完了コードの *CCFAIL* と理由コードの *RC2061* が戻されます。

このサブフィールドは、ビット位置 14 から 23 を占めます。

このサブフィールドには、以下のレポート・オプションがあります。

- ROCOA
- ROCOAD
- ROCOAF
- ROCOD
- ROCODD
- ROCODF

MDREP フィールドにキュー・マネージャーが認識しないオプションが指定されている場合、キュー・マネージャーはビット単位の AND 演算を使用して各サブフィールドを順に検査し、MDREP フィールドとそのサブフィールドのマスクを結合します。この演算の結果が 0 でない場合は、上記の完了コードと理由コードが戻ります。

CCWARN が戻された場合、他の警告条件が存在していれば、どのような理由コードが戻されるかは定義されていません。

レポート・オプションは、ローカル・キュー・マネージャーで認識されない場合でも指定でき、受け入れることができます。これは、ローカル・キュー・マネージャーでは認識されないが、リモート・キュー・マネージャーでは認識され、処理できるようなレポート・オプションでメッセージを送信する必要がある場合に役立ちます。

IBM i でのレポート・フィールドの分析

MDREP フィールドには、いくつかのサブフィールドが含まれます。そのため、アプリケーションによっては、メッセージの送信元が特定のレポートを要求したかどうかをチェックすることが必要です。そのようなアプリケーションでは、このトピックで説明されている技法を使用しなければなりません。

算術演算の使用

整数の算術演算を使用して、次の手順を実行してください。

1. 検査するレポートのタイプに応じて、以下の値のいずれかを選択します。

- COA レポートの場合は ROCOA
- COD レポートの場合は ROCOD
- 例外レポートの場合は ROEXC
- 満了レポートの場合は ROEXP

値を A とします。

2. MDREP フィールドの値を A で割ります。その結果を B と呼ぶことにします。
3. B を 8 で割り、その結果を C とします。
4. C に 8 を掛けて、B から引きます。その結果を D とします。
5. D に A を掛け、その結果を E とします。
6. E がレポートのタイプとして可能性のあるそれぞれの値と等しいかどうかをテストします。

例えば、A が ROEXC の場合、メッセージの送信側が何を指定したか判別するために、E が以下のそれぞれの値と等しいかどうかを調べてください。

- RONONE
- ROEXC
- ROEXCD
- ROEXCF

テストは、アプリケーション論理として最適であれば、どのような順序で行っても構いません。

以下の疑似コードは、この技法を例外レポート・メッセージに対して使用する場合を示しています。

```
A = ROEXC
B = Report/A
C = B/8
D = B - C*8
E = D*A
```

同様の方法を使用して、ROPMI オプションまたは ROPCI オプションについてテストすることができます。この 2 つの定数のどちらか適切な方を値 A として選択してから、上記の処理を進めてください。ただし、前の手順の値 8 は値 2 に置き換えてください。

MDMFL フィールドは 32 ビット長の整数で、別々の 3 つのサブフィールドに分かれています。

これらのサブフィールドは、次のオプションを識別します。

- ローカル・キュー・マネージャーによって認識されない場合に拒否されるメッセージ・フラグ
- ローカル・キュー・マネージャーによって認識されない場合でも、常に受け入れられるメッセージ・フラグ
- 他の特定の条件が満たされた場合にのみ受け入れられるメッセージ・フラグ

注: MDMFL のサブフィールドはすべてキュー・マネージャーが使用するために予約済みです。

各サブフィールドは、サブフィールドに対応する位置に 1 のビット、それ以外の位置に 0 のビットを持つビット・マスクによって識別されます。これらのビットには、ビット 0 が最上位ビットになり、ビット 31 が最下位ビットになるように番号が付けられています。サブフィールドを識別するために定義されているマスクは次のとおりです。

MFRUM

拒否された、サポートされないメッセージ・フラグのためのマスク。

このマスクが示している MDMFL フィールド内のビット位置は、ローカル・キュー・マネージャーがサポートしていないメッセージ・フラグが原因で MQPUT 呼び出しまたは MQPUT1 呼び出しが完了コード CCFAIL および理由コード RC2249 で失敗する場所を示します。

このサブフィールドは、ビット位置 20 から 31 までを占めます。

このサブフィールドには、以下のメッセージ・フラグがあります。

- MFLMIG
- MFLSEG
- MFMIG
- MFSEG
- MFSEGA
- MFSEGI

MFAUM

受け入れられた、サポートされないメッセージ・フラグのためのマスク。

このマスクが示している MDMFL フィールド内のビット位置は、ローカル・キュー・マネージャーがサポートしていないメッセージ・フラグであっても MQPUT 呼び出しまたは MQPUT1 呼び出しで受け入れられる場所を示します。完了コードは、CCOK です。

このサブフィールドは、ビット位置 0 から 11 を占めます。

MFAUXM

ある特定の環境でのみ受け入れられる、サポートされないメッセージ・フラグのためのマスク。

このマスクが示している MDMFL フィールド内のビット位置は、ローカル・キュー・マネージャーがサポートしていないメッセージ・フラグであっても、次の条件を両方とも満たしている場合に MQPUT 呼び出しまたは MQPUT1 呼び出しで受け入れられる場所を示します。

- メッセージの宛先がリモート・キュー・マネージャーである。
- アプリケーションが、ローカル伝送キューに直接メッセージを書き込んでいない (つまり、MQOPEN または MQPUT1 呼び出しに指定されたオブジェクト記述子の ODMN および ODOM フィールドにより識別されるキューが、ローカル伝送キューではない)。

これらの条件が満たされていれば、完了コードの CCOK が戻され、満たされていなければ、完了コードの CCFAIL と理由コードの RC2249 が戻されます。

このサブフィールドは、ビット位置 12 から 19 を占めます。

MDMFL フィールドにキュー・マネージャーが認識しないフラグが指定されている場合、キュー・マネージャーはビット単位の AND 演算を使用して各サブフィールドを順に検査し、MDMFL フィールドとそのサブフィールドのマスクを結合します。この演算の結果が 0 でない場合は、上記の完了コードと理由コードが戻ります。

IBM i IBM i 上でのデータ変換

このトピックでは、データ変換出口へのインターフェース、およびデータ変換が必要な場合にキュー・マネージャーが行う処理について説明します。

データ変換出口は、MQGET 呼び出し処理の一部として呼び出されます。これは、受信側アプリケーションで必須の表記にアプリケーション・メッセージ・データを変換するために使用されます。アプリケーション・メッセージ・データの変換はオプションです。変換を行う場合は、MQGET 呼び出しに GMCONV オプションを指定する必要があります。

データ変換の以下の側面について説明します。

- GMCONV オプションに応答して、キュー・マネージャーが行った処理。1437 ページの『IBM i での変換処理』を参照してください。
- 組み込み形式を処理する際にキュー・マネージャーが使用する処理規則。これらの規則はユーザー作成出口に対しても推奨されます。1438 ページの『IBM i での処理規則』を参照してください。
- レポート・メッセージの変換に対する特別な考慮事項。1442 ページの『IBM i でのレポート・メッセージの変換』を参照してください。
- データ変換出口に渡すパラメーター。1453 ページの『IBM i での MQCONVX (データ変換出口)』を参照してください。
- 出口で利用できる呼び出し。これは、異なる表示間で文字データを変換するために使用されます。1448 ページの『IBM i での MQXCNCV (文字の変換)』を参照してください。
- 出口に特有なデータ構造体のパラメーター。1443 ページの『IBM i での MQDXP (データ変換出口パラメーター)』を参照してください。

IBM i IBM i での変換処理

ここでは、GMCONV オプションへの応答として、キュー・マネージャーが実行する処理について説明します。

GMCONV オプションが MQGET 呼び出しで指定されている場合、およびアプリケーションに戻るメッセージがある場合、キュー・マネージャーは以下の処理を行います。

1. 次の条件を 1 つでも満たせば、変換は不要です。

- メッセージ・データはすでに、MQGET 呼び出しを発行したアプリケーションに必要な文字セットとエンコードで記述されています。アプリケーションは、呼び出しを発行する前に、MQGET 呼び出しの **MSGDSC** パラメーター内の **MDCSI** および **MDENC** フィールドを、必要な値に設定する必要があります。
- メッセージ・データの長さがゼロ。
- MQGET 呼び出しの **BUFFER** パラメーターの値がゼロ。

これらの場合、メッセージは MQGET 呼び出しを発行するアプリケーションに変換されずに戻されます。**MSGDSC** パラメーターの **MDCSI** 値と **MDENC** 値は、メッセージ内の制御情報の値に設定され、呼び出しは以下のいずれかの完了コードと理由コードの組み合わせで完了します。

完了コード
理由コード

CCOK
RCNONE

CCWARN
RC2079

CCWARN RC2080

以下のステップが実行されるのは、メッセージ・データの文字セットとエンコードが、**MSGDSC** パラメーター内の対応する値と異なり、かつ変換されるデータがある場合のみです。

1. メッセージ内の制御情報にある **MDFMT** フィールドに値 **FMNONE** がある場合、メッセージは変換されずに戻り、完了コード **CCWARN** と理由コード **RC2110** が戻ります。

これ以外の場合には、処理は継続されます。

2. メッセージはキューから削除され、**BUFFER** パラメーターと同じサイズの一時バッファに置かれます。ブラウズ操作の場合、メッセージはキューからは削除されずに、一時バッファにコピーされます。
3. メッセージをバッファのサイズに合わせるために切り捨てる場合は、以下の処理が行われます。
 - **GMATM** オプションを指定していない場合、メッセージは変換されずに戻り、完了コード **CCWARN** と理由コード **RC2080** が戻る。
 - **GMATM** オプションを指定した場合は、完了コードは **CCWARN** に、理由コードは **RC2079** に設定され、変換処理は継続される。
4. メッセージを切り捨てずにバッファに収容できる場合、または **GMATM** オプションを指定した場合は、以下の処理が行われます。
 - 形式が組み込み形式の場合、バッファはキュー・マネージャーのデータ変換サービスに渡される。
 - 形式が組み込み形式ではない場合、バッファは形式と同じ名前を持つユーザー作成出口に渡される。出口が検出できない場合、メッセージは変換されずに戻り、完了コード **CCWARN** と理由コード **RC2110** が戻る。

エラーが発生しない場合は、データ変換サービスからの出力またはユーザー作成出口からの出力はメッセージに変換され、さらに完了コードおよび理由コードが **MQGET** 呼び出しを発行するアプリケーションに戻ります。

5. 変換が成功した場合は、キュー・マネージャーは変換したメッセージをアプリケーションに戻します。この場合、**MQGET** 呼び出しが戻す完了コードおよび理由コードは、大抵、以下の組み合わせのいずれかです。

完了コード
理由コード

CCOK
RCNONE

CCWARN
RC2079

ただし、変換がユーザー作成出口によって行われた場合、変換が正常に終了しても、これ以外の理由コードが戻される場合があります。

何らかの理由で変換が失敗した場合、キュー・マネージャーは未変換のメッセージをアプリケーションに返します。このとき、**MSGDSC** パラメーターの **MDCSI** フィールドと **MDENC** フィールドは、メッセージ内の制御情報の値に設定され、完了コードは **CCWARN** に設定されます。

IBM i IBM i での処理規則

組み込み形式を変換する場合、キュー・マネージャーはこのトピックに記載されている処理規則に従います。

これらの規則をユーザー作成出口に適用することがキュー・マネージャーによって強制されているわけではありませんが、そのようにすることを検討してください。キュー・マネージャーが変換する組み込み形式は、以下のとおりです。

- **FMADMN**
- **FMMDE**
- **FMCICS**

- FMPCF
- FMCMD1
- FMRMH
- FMCMD2
- FMRFH
- FMDLH
- FMRFH2
- FMDH
- FMSTR
- FMEVNT
- FMTM
- FMIMS
- FMXQH
- FMIMVS

1. メッセージが変換中に拡張し、**BUFFER** パラメーターのサイズを超える場合は、次の処理が行われます。
 - GMATM オプションを指定していない場合、メッセージは変換されずに戻り、完了コード CCWARN と理由コード RC2120 が戻ります。
 - GMATM オプションを指定した場合は、メッセージは切り捨てられ、完了コードは CCWARN に、理由コードは RC2079 に設定されて、変換処理は継続されます。
2. 切り捨てが発生した場合は、それが変換の前でも変換中でも、**BUFFER** パラメーターに戻る有効なバイト数がバッファの長さより短い可能性があります。

例えば、これは 4 バイトの整数または DBCS 文字がバッファの端にまたがってしまう場合などに発生します。情報の要素が不完全な場合は変換されず、したがって戻ったメッセージ中の未変換部分のバイトには有効な情報が入っていません。これは、変換前に切り捨てられたメッセージが変換中に縮小した場合にも発生します。

戻った有効なバイト数がバッファの長さより短い場合、バッファの末尾の未使用バイトはヌルに設定されます。

3. 配列またはストリングがバッファの端にまたがっている場合、データは最大限変換されます。つまり、特殊な配列エレメントまたは不完全な DBCS 文字だけが変換されず、先行する配列エレメントまたは文字は変換されます。
4. 切り捨てが発生した場合は、それが変換の前でも変換中でも、**DATLEN** パラメーターに戻る長さは、切り捨て前の未変換のメッセージの長さになります。
5. ストリングが 1 バイト文字セット (SBCS)、2 バイト文字セット (DBCS)、マルチバイト文字セット (MBCS) の間で変換された場合、ストリングは拡張されるか縮小されることがあります。
 - FMADMN、FMEVNT、および FMPCF の PCF 形式では、MQCFST 構造体と MQCFSL 構造体の中のストリングは、変換後のストリングを格納するため、必要に応じて拡張されるか縮小されます。
ストリング・リスト構造体 MQCFSL の場合、リスト内のストリングは拡張されるか縮小される場合があります、その量はさまざまです。そのような処理が行われた場合、キュー・マネージャーは変換後に、短いストリングにブランクを埋め込み、最も長いストリングと同じ長さにします。
 - FMRMH 形式では、RMSEO、RMSNO、RMDEO、および RMDNO フィールドによってアドレス指定されるストリングは、変換後のストリングを格納するため、必要に応じて拡張または縮小されます。
 - FMRFH 形式では、RFNVS フィールドは、変換後の名前と値の組を格納するため、必要に応じて拡張または縮小されます。
 - 固定フィールド・サイズの構造体では、キュー・マネージャーは重要な情報が失われなければ、固定フィールド内でのストリングの拡張または縮小を許可します。この関係から、フィールド内の末尾ブランクと最初のヌル文字以降の文字は、重要でないものとして扱われます。

- スtringが拡張された場合、ただし、重要でない文字のみを廃棄して変換後のStringをフィールドに格納する必要がある場合には、変換は成功し、CCOK および理由コード RCNONE (他にエラーがないと見なされる) で呼び出しが完了します。
 - 変換後のStringをフィールドに収納するために重要な文字を廃棄する必要がある場合は、メッセージが戻されて変換は行われず、CCWARN および理由コード RC2190 で呼び出しが完了します。
- 注: この場合、GMATM オプションが指定されたかどうかにかかわらず、理由コード RC2190 が返されます。
- Stringが縮小された場合、キュー・マネージャーはそのフィールドの長さまでStringにブランクを埋め込みます。
6. 1つ以上の IBM MQ ヘッダー構造体とそれにユーザー・データが続くメッセージについては、1つ以上のヘッダー構造体は変換されるが、残りのメッセージは変換されない可能性があります。ただし、2つの例外を除いて、それぞれのヘッダー構造体の中にある MDCSI および MDENC フィールドは、ヘッダー構造体に続く文字セットとデータのエンコードを常に正しく示しています。
- 2つの例外とは、MQCIH および MQIIH 構造体です。これらの構造体の MDCSI および MDENC フィールドの中にある値は有効ではありません。これらの構造体については、構造体に続くデータは、MQCIH または MQIIH 構造体と、文字セットとエンコードは同じです。
7. 取り出されるメッセージの制御情報の MDCSI または MDENC フィールド、あるいは **MSGDSC** パラメーターに、未定義またはサポートされていない値を指定した場合、未定義またはサポートされていない値をメッセージの変換に使用する必要がなければ、キュー・マネージャーはエラーを無視する可能性があります。
- 例えば、メッセージ内の MDENC フィールドが、サポートされていない浮動小数点のエンコードを指定したがメッセージには整数データしか入っていない場合、または、浮動小数点データが入っているが変換不要の場合 (ソース浮動小数点のエンコードとターゲット浮動小数点のエンコードが同一の場合)、エラーの診断は行われることも行われないこともあります。
- エラーが診断されると、完了コード CCWARN と、RC2111、RC2112、RC2113、RC2114 または RC2115、RC2116、RC2117、RC2118 理由コード (該当する場合) のいずれかとともに、メッセージは変換されずに戻されます。**MSGDSC** パラメーターの MDCSI および MDENC フィールドは、メッセージ内の制御情報の値に設定されます。
- エラーが診断されず、変換が正常に完了した場合、**MSGDSC** パラメーターの MDCSI および MDENC フィールドに返される値は、MQGET 呼び出しを発行するアプリケーションによって指定された値です。
8. いずれの場合も、未変換のメッセージがアプリケーションに戻されると、完了コードは CCWARN に設定され、**MSGDSC** パラメーターの MDCSI および MDENC フィールドは未変換のデータに該当する値に設定されます。これは FMNONE の場合にも適用されます。
- REASON** パラメーターは、変換が行われなかった理由を表示するコードに設定されます。ただし、メッセージも切り捨てる必要がある場合は除きます。切り捨てに関する理由コードは、変換に関する理由コードより先に表示されます。(切り捨てられたメッセージが変換されたかどうかを判別するには、**MSGDSC** パラメーターの MDCSI および MDENC フィールドに返された値を確認します。)
- エラーの診断が行われると、特定の理由コードが戻るか、または一般的な理由コード RC2119 が戻ります。戻る理由コードは、データ変換サービスに基づいた診断機能によって決まります。
9. 完了コード CCWARN が戻った場合、また関係のある理由コードが複数ある場合は、コードの優先順位は次のようになります。
- a. 以下の理由コードはすべての理由コードに優先します。
 - RC2079
 - b. 次に優先するのは以下の理由コードです。
 - RC2110
 - c. 残りの理由コードの優先順位は定義されていません。
10. MQGET 呼び出し完了時の理由コードの説明は、次のとおりです。

- 次の理由コードは、メッセージが正常に変換されたことを示しています。
 - RCNONE
- 以下の理由コードは、メッセージが正常に変換された可能性があることを示しています (**MSGDSC** パラメーターの **MDCSI** および **MDENC** フィールドを調べて確認してください)。
 - RC2079
- その他の理由コードはすべて、メッセージが変換されなかったことを表します。

以下の処理は組み込み形式に特有の処理で、ユーザー定義の形式には適用されません。

1. 以下の形式を除きます。

- FMADMN
- FMEVNT
- FMIMVS
- FMPCF
- FMSTR

いずれの組み込み形式も、キュー名の中で有効な文字について SBCS 文字を備えていない文字セットとの間では、変換はできません。そのような変換を試みた場合、メッセージは変換されずに戻され、完了コード CCWARN と、RC2111 または RC2115 の理由コードが戻されます。

UNICODE 文字セット UTF-16 は、キュー名の中で有効な文字について SBCS 文字を備えていない文字セットの 1 つの例です。

2. 組み込み形式用のメッセージ・データが所定の長さまで切り捨てられた場合、ストリングの長さや、要素あるいは構造体の数が入ったメッセージ内のフィールドは、調整されません。したがって、これらフィールドは、アプリケーションに戻るデータの長さを反映しません。メッセージ・データ内のこのようなフィールドに戻される値は、切り捨て前のメッセージでの値です。

切り捨て後の FMADMN メッセージなどの処理中は、アプリケーションが戻ったデータの端より先のデータにアクセスしようとしないように注意する必要があります。

3. 形式の名前が FMDLH の場合、メッセージ・データは MQDLH 構造体で始まり、この後に 0 バイト以上のアプリケーション・メッセージ・データが続きます。アプリケーション・メッセージ・データの形式、文字セット、およびエンコードは、メッセージの先頭にある MQDLH 構造体内の DLFMT、DLCSI、および DLENC のフィールドでそれぞれ定義されます。MQDLH 構造体およびアプリケーション・メッセージ・データは、異なる文字セットおよび異なるエンコードを持つことができるため、MQDLH 構造体とアプリケーション・メッセージ・データのうちの 1 つまたは他方、あるいは両方に変換を要求できます。

キュー・マネージャーは必要に応じて MQDLH 構造体を最初に変換します。変換が成功した場合、または MQDLH 構造体には変換が不要の場合、キュー・マネージャーは MQDLH 構造体内の DLCSI および DLENC フィールドをチェックして、アプリケーション・メッセージ・データの変換が必要かどうか調べます。変換が必要な場合、キュー・マネージャーは MQDLH 構造体の DLFMT フィールドで付けた名前ユーザー作成出口を呼び出します。または DLFMT が組み込み形式の名前である場合、変換そのものを行います。

MQGET 呼び出しが完了コード CCWARN を戻し、理由コードが変換の不成功を示すコードのうちの 1 つである場合、次のいずれかが適用されます。

- MQDLH 構造体は変換できなかった。この場合にはアプリケーション・メッセージ・データも変換されません。
- MQDLH 構造体は変換されたが、アプリケーション・メッセージ・データは変換されなかった。

アプリケーションは、**MSGDSC** パラメーターの **MDCSI** フィールドと **MDENC** フィールド、および MQDLH 構造体のフィールドに返された値を調べて、上記のいずれに該当するかを判別することができます。

4. 形式の名前が FMXQH の場合、メッセージ・データは MQXQH 構造体で始まり、この後に 0 バイト以上の追加データが続く場合があります。この追加データは、長さがゼロの場合もありますが、通常はアプ

リケーション・メッセージ・データです。ただし、追加データの先頭に 1 つ以上の IBM MQ ヘッダー構造体がさらに付いている場合もあります。

MQXQH 構造体はキュー・マネージャーの文字セット内およびエンコード内になければなりません。MQXQH 構造体の後のデータの形式、文字セット、およびエンコードは、MQXQH 内に含まれている MQMD 構造体 MDFMT、MDCSI、および MDENC フィールドで与えられます。後に続く各 IBM MQ ヘッダー構造体については、構造体の MDFMT、MDCSI、および MDENC フィールドで、その構造体の後に続くデータを説明しています。つまり、そのデータが別の IBM MQ ヘッダー構造体であるか、それともアプリケーション・メッセージ・データであるかを示しています。

GMCONV オプションを FMXQH メッセージに指定した場合、アプリケーション・メッセージ・データおよび特定の MQ ヘッダー構造体は変換されます。ただし、MQXQH 構造体のデータは変換されません。したがって、MQGET 呼び出しからの戻り値は、次のようになります。

- **MSGDSC** パラメーター内の MDFMT、MDCSI、および MDENC フィールドの値は、MQXQH 構造体のデータを説明していますが、アプリケーション・メッセージ・データについては説明していません。したがって、値は MQGET 呼び出しを発行したアプリケーションが指定した値とは異なります。

これにより、GMCONV オプションを指定して伝送キューからメッセージを繰り返し取得するアプリケーションは、各 MQGET 呼び出しの前に、**MSGDSC** パラメーターの MDCSI および MDENC フィールドを、アプリケーション・メッセージ・データに必要な値にリセットする必要があります。

- 最後の MQ ヘッダー構造体内にある MDFMT、MDCSI、および MDENC フィールドの値は、アプリケーション・メッセージ・データを説明します。IBM MQ ヘッダー構造体がない場合、アプリケーション・メッセージ・データは MQXQH 構造体内の MQMD 構造体にある上記と同じフィールドで説明されます。変換が成功した場合、値は MQGET 呼び出しを発行したアプリケーションによって **MSGDSC** パラメーター内に指定された値と同じ値になります。

メッセージが配布リスト・メッセージの場合、MQXQH 構造体のあとには MQDH 構造体 (それに加えてその MQOR レコードと MQPMR レコードの配列) が続きます。このあとに、さらに 1 つ以上の IBM MQ ヘッダー構造体と、1 バイト以上のアプリケーション・メッセージ・データが続くこともあります。MQXQH 構造体と同様、MQDH 構造体はキュー・マネージャーの文字セット内およびエンコード内になければなりません。また、GMCONV オプションが指定されている場合も、MQDH 構造体は MQGET 呼び出しでは変換されません。

上記の MQXQH 構造体および MQDH 構造体の処理は、主に、メッセージ・チャネル・エージェントが伝送キューからメッセージを読み取る際に使用するものです。

IBM i IBM i でのレポート・メッセージの変換

レポート・メッセージに入るアプリケーション・メッセージ・データの量は、元のメッセージの送信側が指定したレポート・オプションに従って変えることができます。

特に、レポート・メッセージに入れるデータを以下のいずれかにすることができます。

1. アプリケーション・メッセージ・データなし
2. 元のメッセージからの複数のアプリケーション・メッセージ・データ

これが発生するのは、元のメッセージの送信側が RO*D を指定し、メッセージが 100 バイトよりも長い場合です。

3. 元のメッセージからのすべてのアプリケーション・メッセージ・データ。

これが発生するのは、元のメッセージの送信側が RO*F または RO*D を指定し、メッセージの長さが 100 バイト以下の場合です。

キュー・マネージャーまたはメッセージ・チャネル・エージェントがレポート・メッセージを生成すると、元のメッセージからレポート・メッセージ内の制御情報の *MDFMT* フィールドに形式名をコピーします。したがって、レポート・メッセージ内の形式名からわかるデータ長は、レポート・メッセージに存在する長さとは異なることがあります (前述の 1 および 2 の場合)。

レポート・メッセージを取り出すときに GMCONV オプションが指定された場合、以下のことが発生します。

- 前述の 1 の場合、データ変換出口は呼び出されません (レポート・メッセージにデータがまったく含まれないため)。
- 前述の 3 の場合、形式名はメッセージ・データの長さを正確に暗黙指定します。
- しかし前述の 2 の場合は、データ変換出口が呼び出されて、形式名で暗黙指定された長さより短いメッセージが変換されます。

なお、出口に渡される理由コードは通常、RCNONE です (つまり、この理由コードはメッセージが切り捨てられたことを示しません)。メッセージ・データがレポート・メッセージの送信側に切り捨てられ、MQGET 呼び出しに応答した受信側のキュー・マネージャーに切り捨てられるわけではないからです。

上記の問題があるため、データ変換出口に渡すデータの長さを形式名を使用して縮めることは避けてください。その代わりに、出口は与えられたデータの長さをチェックし、さらに形式名で暗黙指定された長さより短いデータを変換する準備を整える必要があります。データが正常に変換された場合は、出口が完了コード CCOK および理由コード RCNONE を戻す必要があります。変換されるメッセージ・データの長さは、**INLEN** パラメーターとして出口に渡されます。

製品センシティブ・プログラミング・インターフェース

行われたアクティビティーに関する情報がレポート・メッセージに含まれる場合、それはアクティビティー・レポートと呼ばれます。アクティビティーには以下の例があります。

- MCA がメッセージをキューからチャンネルを経由して送信する
- MCA がメッセージをチャンネルから受け取り、それをキューに書き込む
- MCA のデッドレターによる配信不能メッセージのキューイング
- MCA がメッセージをキューから取得し、それを廃棄する
- 送達不能ハンドラーがメッセージをキューに戻す
- コマンド・サーバーが PCF 要求を処理する - ブローカーがパブリッシュ要求を処理する
- ユーザー・アプリケーションがメッセージをキューから取得する - ユーザー・アプリケーションがキュー上のメッセージを参照する

キュー・マネージャーを含むすべてのアプリケーションにおいて、メッセージ・データの一部をアクティビティー・レポートのレポート・ヘッダーの後に追加できます。データの一部が送信される場合に提供する必要がありますデータの量は固定されておらず、アプリケーションにより決定されます。返される情報は、アクティビティー・レポートを処理するアプリケーションにとって有用なものとなります。キュー・マネージャーのアクティビティー・レポートは、元のメッセージに含まれるすべての標準 IBM MQ ヘッダー構造体 (「MQH」で始まる) を含めて戻します。これには、例えば、元のメッセージに含まれていたすべての MQRFH2 ヘッダーが含まれます。また、キュー・マネージャーは検出された MQCFH ヘッダーも返しますが、それに関連付けられた PCF パラメーターは返しません。これによりモニター・アプリケーションは、メッセージが何に関するものであったかを知ることができます。

IBM i IBM i での MQDXP (データ変換出口パラメーター)

データ変換出口パラメーター・ブロック。

概要

目的: MQDXP 構造体は、MQGET 呼び出し処理の一環として、メッセージ・データを変換するためにデータ変換出口を呼び出す時に、キュー・マネージャーがこの出口に渡すパラメーターです。データ変換出口の詳細については、MQCONVX 呼び出しに関する説明を参照してください。

文字セットとエンコード: MQDXP 内の文字データは、ローカル・キュー・マネージャーの文字セットで記述されています。これは、**CodedCharSetId** キュー・マネージャー属性で指定します。MQDXP の数値データはネイティブ・マシンのエンコードであり、ENNAT によって指定されます。

使用法: MQDXP の DXLEN、DXCC、DXREA および DXRES フィールドのみが出口によって変更できます。その他のフィールドへの変更は無視されます。ただし、変換対象のメッセージが論理メッセージの一部だけを含むセグメントである場合、DXLEN フィールドは変更できません。

出口からキュー・マネージャーに制御が戻されると、キュー・マネージャーは、MQDXP 内に返されたこれらの値をチェックします。戻された値が有効でない場合は、出口が *DXRES* に *XRFAIL* を戻したかのようにキュー・マネージャーは処理を続けます。ただしこの場合、出口によって戻された *DXCC* および *DXREA* フィールドの値をキュー・マネージャーは無視し、出口への入力 でこれらのフィールドにある値を代わりに使用します。この処理が行われるのは、MQDXP 内の値が次のような場合です。

- *DXRES* フィールドは *XROK* でなく *XRFAIL* でない。
- *DXCC* フィールドは *CCOK* でなく *CCWARN* でない。
- *DXLEN* フィールドがゼロより小さい、または変換対象のメッセージが論理メッセージの一部だけを含むセグメントである場合に *DXLEN* フィールドが変更された。
- [1444 ページの『フィールド』](#)
- [1448 ページの『RPG 宣言 \(コピー・ファイル CMQDXPH\)』](#)

フィールド

MQDXP 構造体には、以下のフィールドが含まれます。フィールドはアルファベット順に説明されています。

DXAOP (10 桁の符号付き整数)

アプリケーション・オプション。

これは、MQGET 呼び出しを発行したアプリケーションが指定していた MQGMO 構造体の *GMOPT* フィールドのコピーです。GMATM オプションが指定されたかどうかを確認するために、出口はこのフィールドを調べる必要があります。

これは、出口に対する入力フィールドです。

DXCC (10 桁の符号付き整数)

完了コード。

出口が呼び出された時点でこのフィールドに入る完了コードは、出口が何もしないことを選択した場合に、MQGET 呼び出しを発行したアプリケーションに戻されます。これは常に *CCWARN* です。それはメッセージが切り捨てられたか、またはメッセージが変換を要求したがまだ変換されていないかのいずれかです。

出口からの出力時にこのフィールドに入る完了コードは、MQGET 呼び出しの **CMPCOD** パラメーターに入れて、アプリケーションに戻されます。有効な値は、*CCOK* および *CCWARN* のみです。出口が出力でこのフィールドをどのように設定する必要があるかの推奨事項については、*DXREA* フィールドの説明を参照してください。

これは、出口に対する入出力フィールドです。

DXCSI (10 桁の符号付き整数)

アプリケーションに必須の文字セット。

これは、MQGET 呼び出しを発行しているアプリケーションに必須の文字セットのコード化文字セット ID です。詳細については、MQMD 構造体の *MDCSI* フィールドを参照してください。アプリケーションが MQGET 呼び出しに特殊値 *CSQM* を指定する場合、キュー・マネージャーは、この値をキュー・マネージャーが使用する文字セットの実際の文字セット ID に変更してから、出口を呼び出します。

変換が正常に行われた場合、出口はこれをメッセージ記述子の *MDCSI* フィールドにコピーする必要があります。

これは、出口に対する入力フィールドです。

DXENC (10 桁の符号付き整数)

アプリケーションが必要とする数値エンコード方式。

これは、MQGET 呼び出しを発行したアプリケーションが必要とする数値エンコード方式です。詳細については、MQMD 構造体の *MDENC* フィールドを参照してください。

変換が正常に行われた場合、出口はこれをメッセージ記述子の **MDENC** フィールドにコピーする必要があります。

これは、出口に対する入力フィールドです。

DXHCN (10 桁の符号付き整数)

接続ハンドル。

これは、MQXCNCV 呼び出しで利用できる接続ハンドルです。このハンドルは、MQGET 呼び出しを発行したアプリケーションによって指定されたハンドルと、必ずしも同じでなくても構いません。

DXLEN (10 桁の符号付き整数)

メッセージ・データ長 (バイト単位)。

出口の呼び出し時、このフィールドには、アプリケーションのメッセージ・データの元のデータ長が入っています。アプリケーションによって提供されるバッファに収まるようにメッセージが切り捨てられた場合、出口に提供されるメッセージのサイズは、**DXLEN** の値より小さくなります。出口に提供されるメッセージのサイズは、発生した可能性のある切り捨てに関係なく、出口の **INLEN** パラメータによって常に指定されます。

切り捨ては、出口への入力時に **RC2079** 値がある **DXREA** フィールドにより示されます。

ほとんどの変換ではこの長さを変更する必要はありませんが、必要な場合は出口が長さを変更することができます。出口によって設定される値は、MQGET 呼び出しの **DATLEN** パラメータでアプリケーションに戻されます。ただし、変換対象のメッセージが論理メッセージの一部だけを含むセグメントである場合、長さは変更できません。これは、長さの変更によって、論理メッセージ内の以降のセグメントのオフセットが不正確になるからです。

出口がデータの長さを変更する場合は、メッセージ・データがアプリケーションのバッファに収まるかどうかの判断 (未変換データの長さに基づく) をキュー・マネージャーがすでに行っているという点に注意する必要があります。この判断によって、メッセージをキューから削除するかどうか (あるいはブラウズ要求の場合は、ブラウズ・カーソルを移動させるかどうか) が決定されます。この判断は、変換で生じたデータ長の変更による影響を受けません。このため、変換出口ではアプリケーションのメッセージ・データ長を変更しないことをお勧めします。

文字変換で長さの変化が発生する場合は、末尾ブランクを切り捨てたり、あるいは必要に応じてブランクを埋め込んだりして、ストリングを同じ長さ (バイト数) の別のストリングに変換できます。

メッセージにアプリケーション・メッセージ・データが含まれていない場合は出口は呼び出されません。したがって **DXLEN** は必ずゼロより大きい値になります。

これは、出口に対する入出力フィールドです。

DXREA (10 桁の符号付き整数)

DXCC を限定する理由コード。

出口が呼び出された時点でこのフィールドに入る理由コードは、出口が何もしないことを選択した場合に、MQGET 呼び出しを発行したアプリケーションに戻されます。可能な値のうち **RC2079** は、アプリケーションによって提供されるバッファのサイズに適合するためにメッセージが切り捨てられたことを示し、**RC2119** は、メッセージが変換を要求したがまだ変換されていないことを示します。

出口からの出力時、このフィールドには、アプリケーションの MQGET 呼び出しの **REASON** パラメータに返される理由が入っています。以下のことをお勧めします。

- **DXREA** に出口への入力における値 **RC2079** がある場合、変換が成功するか失敗するかに関係なく、**DXREA** および **DXCC** フィールドを変更してはなりません。

(**DXCC** フィールドが **CCOK** でない場合、メッセージを検索するアプリケーションは、要求された値とメッセージの記述子内の **MDENC** および **MDCSI** の戻り値を比較して変換の失敗を識別できます。一方、アプリケーションは切り捨てられたメッセージとバッファのサイズに適合しているメッセージを区別できません。このため、**RC2079** は変換の失敗を示す理由より優先して戻されます。)

- **DXREA** に出口への入力でその他の値がある場合

- 変換が成功する場合は、*DXCC* は *CCOK* に設定され、*DXREA* は *RCNONE* に設定されます。
- 変換が失敗した場合、またはメッセージが拡張されたためバッファに収まるように切り捨てる必要がある場合は、*DXCC* を *CCWARN* に設定し (または変更せずそのままにして)、*DXREA* を次のリストのいずれかの値に設定して、失敗の性質を示す必要があります。

変換後のメッセージが大きくてバッファに収まらない場合は、*MQGET* 呼び出しを発行したアプリケーションが *GMATM* を指定している場合に限り、切り捨てが行われます。

- そのオプションを指定した場合、理由コード *RC2079* が戻されます。
- そのオプションを指定していない場合、理由コードの *RC2120* と共に未変換のメッセージが戻されます。

次のリストの理由コードは、変換に失敗した理由を示すために出口が使用する推奨コードです。ただし、適切と見なされる場合、出口は *RC** コードのセットに含まれる他の値を返すことができます。また、*MQGET* 呼び出しを発行するアプリケーションと通信するために出口が必要とする条件を示すために、出口が使用する、*RC0900* から *RC0999* までの値の範囲が割り振られます。

注: メッセージが正常に変換できない場合、出口は *DXRES* フィールドに *XRFAIL* を戻す必要があります。これは、キュー・マネージャーが未変換のメッセージを戻すことができるようにするためです。これは、*DXREA* フィールドに返される理由コードに関係なく当てはまります。

RC0900

(900, X'384') アプリケーション定義の理由コードの最小値。

RC0999

(999, X'3E7') アプリケーション定義の理由コードの最大値。

RC2120

(2120, X'848') 変換されたデータが、バッファには大きすぎる。

RC2119

(2119, X'847') メッセージ・データが変換されなかった。

RC2111

(2111, X'83F') ソース・エンコード文字セット ID が無効である。

RC2113

(2113, X'841') メッセージ内のパック 10 進数のエンコードが認識できない。

RC2114

(2114, X'842') メッセージ内の浮動小数点のエンコードが認識できない。

RC2112

(2112, X'840') ソース整数エンコードが認識できない。

RC2115

(2115, X'843') ターゲット・エンコード文字セット ID が無効である。

RC2117

(2117, X'845') 受信側で指定されたパック 10 進数のエンコードが認識できない。

RC2118

(2118, X'846') 受信側で指定された浮動小数点のエンコードが認識できない。

RC2116

(2116, X'844') ターゲット整数エンコードが認識できない。

RC2079

(2079, X'81F') 切り捨てられたメッセージが戻された (処理は完了している)。

これは、出口に対する入出力フィールドです。

DXRES (10 桁の符号付き整数)

出口からの応答。

このフィールドは出口によって設定され、変換が成功したかどうかを示します。これは以下のいずれかです。

XROK

変換は正常に行われました。

出口がこの値を指定した場合、キュー・マネージャーは、MQGET 呼び出しを発行したアプリケーションに、以下を返します。

- 出口からの出力で **DXCC** フィールドの値。
- 出口からの出力で **DXREA** フィールドの値。
- 出口からの出力で **DXLEN** フィールドの値。
- 出口の出力バッファー **OUTBUF** の内容。返されるバイト数は、出口の **OUTLEN** パラメーターと、出口からの出力時の **DXLEN** フィールドの値のうち、小さい方です。

出口のメッセージ記述子パラメーター内の **MDENC** および **MDCSI** フィールドが、両方とも変更されていない場合は、キュー・マネージャーは以下を返します。

- 出口に対する入力 の MQDXP 構造体の **MDENC** および **MDCSI** フィールドの値。

出口のメッセージ記述子パラメーター内の **MDENC** および **MDCSI** フィールドのうち、一方または両方が変更された場合、キュー・マネージャーは以下を返します。

- 出口からの出力時の 出口のメッセージ記述子パラメーター内の **MDENC** および **MDCSI** フィールドの値。
-

XRFAIL

変換は失敗しました。

出口がこの値を指定した場合、キュー・マネージャーは、MQGET 呼び出しを発行したアプリケーションに、以下を返します。

- 出口からの出力で **DXCC** フィールドの値。
- 出口からの出力で **DXREA** フィールドの値。
- 出口への入力 で **DXLEN** フィールドの値。
- 出口の入力バッファー **INBUF** の内容。戻されるバイト数は、**INLEN** パラメーターにより指定される。

出口で **INBUF** を変更した場合の結果は定義されていません。

DXRES は、出口からの出力フィールドです。

DXSID (4 バイトの文字ストリング)

構造体 ID

値は次のものでなければなりません。

DXSIDV

データ変換出口パラメーター構造体の ID。

これは、出口に対する入力フィールドです。

DXVER (10 桁の符号付き整数)

構造体のバージョン番号。

値は次のものでなければなりません。

DXVER1

データ変換出口パラメーター構造体のバージョン番号。

以下の定数は、現行バージョンのバージョン番号を指定しています。

DXVERC

データ変換出口パラメーター構造体の現行バージョンです。

注: この構造体の新しいバージョンが導入されても、既存の部分のレイアウトは変わりません。したがって、DXVER フィールドが、出口で使用するしなければならないフィールドが含まれている最小バージョンと等しいか、またはそれより大きいことを、出口で検査する必要があります。

これは、出口に対する入力フィールドです。

DXXOP (10 桁の符号付き整数)

予約済み。

これは予約フィールドで、値は 0 です。

RPG 宣言 (コピー・ファイル CMQDXPH)

```
D*..1.....2.....3.....4.....5.....6.....7..
D* MQDXP Structure
D*
D* Structure identifier
D DXSID 1 4
D* Structure version number
D DXVER 5 8I 0
D* Reserved
D DXXOP 9 12I 0
D* Application options
D DXAOP 13 16I 0
D* Numeric encoding required by application
D DXENC 17 20I 0
D* Character set required by application
D DXCSI 21 24I 0
D* Length in bytes of message data
D DXLEN 25 28I 0
D* Completion code
D DXCC 29 32I 0
D* Reason code qualifying DXCC
D DXREA 33 36I 0
D* Response from exit
D DXRES 37 40I 0
D* Connection handle
D DXHCN 41 44I 0
```

IBM i IBM i での MQXCNCV (文字の変換)

MQXCNCV 呼び出しは、文字セットを別の文字セットに変換します。

この呼び出しは、IBM MQ フレームワーク・インターフェースの 1 つである IBM MQ データ変換インターフェース (DCI) の一部です。注: この呼び出しは、データ変換出口からの場合にのみ使用できます。

- [1448 ページの『構文』](#)
- [1448 ページの『パラメーター』](#)
- [1453 ページの『RPG での呼び出し \(ILE\)』](#)

構文

MQXCNCV HCONN, OPTS, SRCCSI, SRCLEN, SRCBUF, TGTCSE, TGTLEN, TGTBUF, DATLEN, CMPCOD, REASON)

パラメーター

MQXCNCV 呼び出しには、以下のパラメーターがあります。

HCONN (10 桁の符号付き整数) - 入力

接続ハンドル。

このハンドルは、キュー・マネージャーに対する接続を表します。通常、MQDXP 構造体の DXHCN フィールドに入れてデータ変換出口に渡されるハンドルです。このハンドルは、必ずしも、MQGET 呼び出しを発行したアプリケーションが指定するハンドルと同じものではありません。

IBM i では、HCONN に次の特殊値を指定できます。

HCDEFH

デフォルトの接続ハンドル。

OPTS (10 桁の符号付き整数) - 入力

MQXCNCV のアクションを制御するオプション。

このセクションで後述するオプションをゼロ個以上指定できます。複数のオプションを必要とする場合は、値を追加できます (同じ定数を複数回追加しないでください)。

デフォルト変換オプション: 以下のオプションはデフォルトの文字変換の使用を制御します。

DCCDEF

デフォルト変換。

このオプションは、呼び出しで指定された文字セットが片方または両方ともサポートされていない場合、デフォルト文字変換が使用できることを指定します。これにより、キュー・マネージャーはインストール時に指定されたデフォルト文字セットを使用できます。このオプションを指定しておく、と、ストリング変換時には、デフォルトの文字セットのうち、呼び出しで指定された文字セットに最も近いものが使用されます。

注: 呼び出し時に指定された文字セット以外を使用してストリングの変換を行うと、一部の文字が正しく変換されない場合があります。指定された文字セットとデフォルト文字セットの両方に共通する文字だけをストリング内に使用するようになれば、これを回避することができます。

デフォルト文字セットは、キュー・マネージャーのインストール時または再始動時に、構成オプションによって定義されます。

DCCDEF が指定されない場合は、キュー・マネージャーは指定された文字セットのみを使用してストリングを変換します。この場合、呼び出しは文字セットの片方または両方がサポートされていないと失敗します。

埋め込みオプション: 以下のオプションを指定すると、変換後のストリングがターゲット・バッファーに収まるよう、キュー・マネージャーは、変換後のストリングにブランクを埋め込んだり、有意でない末尾文字を廃棄します。

DCCFIL

ターゲット・バッファーを埋める。

このオプションは、ターゲット・バッファーが完全に埋められるように変換が行われることを要求します。

- ストリングを変換すると短縮する場合、ターゲット・バッファーを埋めるために末尾ブランクが追加されます。
- ストリングを変換すると拡張する場合、変換されたストリングがターゲット・バッファーに収まるよう、有意でない末尾文字が廃棄されます。この廃棄が正常に行われると、CCOK と理由コード RCNONE が戻されて呼び出しが完了します。

有意でない末尾文字の数が足りないときは、ストリングのうちのターゲット・バッファーに収まるだけの数の文字がターゲット・バッファーに入れられ、CCWARN と理由コード RC2120 が戻されて呼び出しが完了します。

有意でない文字とは、次の文字です。

- 末尾ブランク
- ストリング内の最初のヌル文字に続く文字 (ただし、最初のヌル文字自体は除く)
- ストリング TGTCSE および TGTLEN が、有効な文字でターゲット・バッファーを完全に設定できない場合、呼び出しは CCFAIL および理由コード RC2144 で失敗します。これは、TGTCSE が

純粋な DBCS 文字セット (UTF-16 など) であるが、TGTLEN が奇数バイト数の長さを指定している場合に発生することがあります。

- TGTLEN の値は、SRCLen より小さくても大きくても構いません。MQXCNCV から戻ったとき、DATLEN の値は、TGTLEN と同じになっています。

このオプションを指定しない場合は、以下のようになります。

- スtring は、ターゲット・バッファ内で必要に応じて短縮したり、拡張することができます。有意でない末尾の文字が追加されることも廃棄されることもありません。

変換された String がターゲット・バッファに収まる場合は、CCOK と理由コード RCNONE が戻されて呼び出しが完了します。

変換された String がターゲット・バッファにとって大き過ぎる場合は、String のうちのそのバッファに収まるだけの数の文字がターゲット・バッファに入れられ、CCWARN と理由コード RC2120 が戻されて呼び出しが完了します。この場合、戻されるバイト数は TGTLEN のバイト数より少ないことがあるので注意してください。

- TGTLEN の値は、SRCLen より小さくても大きくても構いません。MQXCNCV から戻ったとき、DATLEN の値は、TGTLEN 以下になっています。

エンコード・オプション: 以下のオプションを使用して、ソース・String およびターゲット・String の整数エンコードを指定できます。該当するエンコードが使用される条件は、対応する文字セットの ID から決まります。すなわち、この文字セットの主記憶内での表現が 2 進整数のエンコードに依存することを ID が意味している場合にのみ、このエンコードが使用されます。これによって影響を受けるのは、特定のマルチバイト文字セット (例えば、UTF-16 文字セット) だけです。

文字セットが 1 バイト文字セット (SBCS)、または主記憶内の表現が整数エンコードによって変わらないマルチバイト文字セットの場合、エンコードは無視されます。

次に示す、DCCS* の値の 1 つと DCCT* の値の 1 つを組み合わせ指定してください。

DCCSNA

ソース・エンコードはこの環境およびプログラミング言語ではデフォルト。

DCCSNO

ソース・エンコードは順方向。

DCCSRE

ソース・エンコードは逆方向。

DCCSUN

ソース・エンコードは未定義。

DCCTNA

ターゲット・エンコードはこの環境およびプログラミング言語ではデフォルト。

DCCTNO

ターゲット・エンコードは順方向。

DCCTRE

ターゲット・エンコードは逆方向。

DCCTUN

ターゲット・エンコードは未定義。

定義済みのエンコードの値は、OPTS フィールドに直接加算できます。ただし、ソース・エンコードおよびターゲット・エンコードを MQMD 構造体または他の構造体の MDENC フィールドから得た場合は、以下の処理を行う必要があります。

1. MDENC フィールドから浮動小数点のエンコードおよびパック 10 進のエンコードを除去して、整数のエンコードを取り出す。この方法の詳細については、[1432 ページの『IBM i でのエンコードの分析』](#)を参照してください。
2. 上記 1 で得られた整数のエンコードに適切な係数を掛けてから、OPTS フィールドに加えます。係数には、次の 2 つがあります。

DCCSFA

ソースのエンコード用の係数。

DCCTFA

ターゲットのエンコード用の係数。

エンコード・オプションを指定しない場合は、未定義を表す (DCC*UN) をデフォルトにとります。ほとんどの場合、MQXCNVNVC 呼び出しの正常な完了に対してこれが影響を与えることはありません。ただし、対応する文字セットがマルチバイト文字セットであり、その表示がエンコードによって変わる場合 (例えば UTF-16 文字セット)、呼び出しは理由コード RC2112 または RC2116 (該当する方) で失敗します。

デフォルト・オプション: これまで説明したオプションを何も指定しない場合、以下のオプションを使用できます。

DCCNON

指定されるオプションはありません。

DCCNON は、プログラムの文書化を支援するために定義します。このオプションを他のオプションと組み合わせて使用することは意図されていませんが、値がゼロであるため、そのような使い方をしても検出できません。

SRCCSI (10 桁の符号付き整数) - 入力

変換前のストリングのコード化文字セット ID。

SRCBUF 内の入力ストリングのコード化文字セット ID です。

SRCLLEN (10 桁の符号付き整数) - 入力

変換前のストリングの長さ。

SRCBUF 内の入力ストリングの長さ (バイト数) です。これはゼロ以上でなければなりません。

SRCBUF (1 バイトの文字ストリング x SRCLLEN) - 入力

変換するストリング。

文字セットを変換するストリングが入るバッファです。

TGTCSI (10 桁の符号付き整数) - 入力

変換後のストリングのコード化文字セット ID。

SRCBUF の変換先の文字セットのコード化文字セット ID。

TGTLEN (10 桁の符号付き整数) - 入力

出力バッファの長さ。

出力バッファ TGTBUF の長さ (バイト数) です。これはゼロ以上でなければなりません。この値は、SRCLLEN より小さくても大きくても構いません。

TGTBUF (1 バイトの文字ストリング x TGTLEN) - 出力

変換後のストリング。

TGTCSI が定義した文字セットに変換された後のストリングです。変換後のストリングは、未変換のストリングより短くなることも長くなることもあります。DATLEN パラメーターは、戻される有効バイト数を示します。

DATLEN (10 桁の符号付き整数) - 出力

出力ストリングの長さ。

出力バッファ TGTBUF に戻されるストリングの長さです。変換後のストリングは、未変換のストリングより短くなることも長くなることもあります。

CMPCOD (10 桁の符号付き整数) - 出力

完了コード。

これは、以下のいずれかになります。

CCOK

正常終了。

CCWARN

警告 (部分完了)。

CCFAIL

呼び出し失敗。

REASON (10 桁の符号付き整数) - 出力

CMPCOD を限定する理由コード。

CMPCOD が CCOK の場合

RCNONE

(0, X'000') レポートする理由コードはありません。

CMPCOD が CCWARN の場合

RC2120

(2120, X'848') 変換されたデータが、バッファーには大きすぎる。

CMPCOD が CCFAIL の場合

RC2010

(2010, X'7DA') データ長パラメーターが無効である。

RC2150

(2150, X'866') DBCS スtringが無効である。

RC2018

(2018, X'7E2') 接続ハンドルが無効です。

RC2046

(2046, X'7FE') オプションが無効であるか、矛盾しています。

RC2102

(2102, X'836') 使用できるシステム・リソースが不足しています。

RC2145

(2145, X'861') ソース・バッファー・パラメーターが無効。

RC2111

(2111, X'83F') ソース・エンコード文字セット ID が無効である。

RC2112

(2112, X'840') ソース整数エンコードが認識できない。

RC2143

(2143, X'85F') ソース長パラメーターが無効である。

RC2071

(2071, X'817') ストレージが不足しています。

RC2146

(2146, X'862') ターゲット・バッファー・パラメーターが無効である。

RC2115

(2115, X'843') ターゲット・エンコード文字セット ID が無効である。

RC2116

(2116, X'844') ターゲット整数エンコードが認識できない。

RC2144

(2144, X'860') ターゲット長パラメーターが無効。

RC2195

(2195, X'893') 予期しないエラーが発生しました。

これらの理由コードの詳細については、[1426 ページの『IBM i の戻りコード \(ILE RPG\)』](#)を参照してください。

RPG での呼び出し (ILE)

```
C*..1.....2.....3.....4.....5.....6.....7..  
C                                CALLP      MQXCNV (HCONN : OPTS : SRCCSI :  
C                                  SRCLEN : SRCBUF : TGTCSI :  
C                                  TGTLEN : TGTBUF : DATLEN :  
C                                  CMPCOD : REASON)
```

呼び出しのプロトタイプ定義は次のようになります。

```
D*..1.....2.....3.....4.....5.....6.....7..  
DMQXCNV PR                                EXTPROC('MQXCNV')  
D* Connection handle  
D HCONN                                10I 0 VALUE  
D* Options that control the action of MQXCNV  
D OPTS                                10I 0 VALUE  
D* Coded character set identifier of string before conversion  
D SRCCSI                              10I 0 VALUE  
D* Length of string before conversion  
D SRCLEN                              10I 0 VALUE  
D* String to be converted  
D SRCBUF                              *    VALUE  
D* Coded character set identifier of string after conversion  
D TGTCSI                              10I 0 VALUE  
D* Length of output buffer  
D TGTLEN                              10I 0 VALUE  
D* String after conversion  
D TGTBUF                              *    VALUE  
D* Length of output string  
D DATLEN                              10I 0  
D* Completion code  
D CMPCOD                              10I 0  
D* Reason code qualifying CMPCOD  
D REASON                              10I 0
```

IBM i IBM i での MQCONVX (データ変換出口)

この呼び出し定義は、データ変換出口に渡すパラメーターを記述します。

MQCONVX というエントリー・ポイントがキュー・マネージャーから提供されるわけではありません (『使用上の注意』の [1455 ページの『11』](#)を参照してください)。

この定義は、IBM MQ フレームワーク・インターフェースの 1 つである IBM MQ データ変換インターフェース (DCI) の一部です。

- [1453 ページの『構文』](#)
- [1453 ページの『使用上の注意』](#)
- [1455 ページの『パラメーター』](#)
- [1456 ページの『RPG での呼び出し \(ILE\)』](#)

構文

MQCONVX (MQDXP, MQMD, INLEN, INBUF, OUTLEN, OUTBUF)

使用上の注意

1. データ変換出口とは、MQGET 呼び出しの処理中に制御を受け取るユーザー作成出口です。データ変換出口が実行する関数は、出口の作成者が定義します。ただし、この出口は、ここで述べる規則および関連のパラメーター構造体 MQDXP に示されている規則に従っていなければなりません。

データ変換出口用に使用できるプログラミング言語は、環境によって決まります。

2. この出口は、次のすべての記述が該当する場合に限り、呼び出されます。

- MQGET 呼び出しで GMCONV オプションが指定されている。
 - メッセージ記述子の *MDFMT* フィールドが FMNONE ではない。
 - メッセージがまだ必須表現形式になっていない。つまり、メッセージの *MDCSI* および *MDENC* の一方または両方が、アプリケーションが MQGET 呼び出しのメッセージ記述子に指定した値と異なっている。
 - キュー・マネージャーが、まだ変換に成功していない。
 - アプリケーションのバッファの長さがゼロより大きい。
 - メッセージ・データの長さがゼロより大きい。
 - MQGET 操作中の理由コードが RCNONE または RC2079 である。
3. 出口を作成するときには、切り捨てにより短くなったメッセージを変換できる方法でコーディングする配慮が必要です。メッセージの切り捨てが起こるのは次のような場合です。
- 受信側のアプリケーションがメッセージより小さいバッファを提供したにもかかわらず、MQGET 呼び出しで GMATM オプションを指定した。
- この場合、出口への入力の **MQDXP** パラメーターの *DXREA* フィールドの値は RC2079 になります。
- メッセージの送信元が、送信前にメッセージを切り捨てた。これはレポート・メッセージで発生する可能性があります (詳細については、[1442 ページの『IBM iでのレポート・メッセージの変換』](#)を参照してください)。
- この場合、出口への入力の **MQDXP** パラメーターの *DXREA* フィールドの値は RCNONE になります (受信側アプリケーションがメッセージに十分な大きさのバッファを提供した場合)。
- したがって、出口への入力で *DXREA* フィールドの値は必ずしもメッセージが切り捨てられたかどうかを判断するのに使用できません。
- 切り捨てられたメッセージの特徴は、**INLEN** パラメーターで出口に提供される長さが、メッセージ記述子の *MDFMT* フィールドに含まれる形式名によって暗黙指定される長さより 短くなる ことです。したがってデータを変換しようとする前に、出口は *INLEN* の値を調べなければなりません。出口では、形式名が示すデータの総量が提供されていることを想定していません。
- 切り捨てられたメッセージを変換するように出口が作成されていない場合、**INLEN** が予期した値より小さいと、出口は **MQDXP** パラメーターの *DXRES* フィールドで XRFAIL を返し、*DXCC* フィールドは CCWARN に設定され、*DXREA* フィールドは RC2110 に設定されます。
- 出口が切り捨てられたメッセージを変換するために作成された場合、出口はできるだけ多くのデータを変換します (次の『使用上の注意』を参照)。*INBUF* の最後を超えたデータを調べたり変換したりしないよう注意してください。変換が正常に完了した場合、出口は **MQDXP** パラメーターの *DXREA* フィールドを未変更のままにしておく必要があります。これにより、受信側のキュー・マネージャーによってメッセージが切り捨てられた場合は RC2079 が返され、メッセージの送信側によってメッセージが切り捨てられた場合は RCNONE が返されます。
- メッセージが、変換中に *OUTBUF* より大きい位置まで拡大することも可能です。この場合、出口はメッセージを切り捨てるかどうかを決定する必要があります。**MQDXP** パラメーターの *DXAOP* フィールドは、受信側アプリケーションが GMATM オプションを指定したかどうかを示します。
4. 一般に、*INBUF* で出口に提供されるメッセージのデータすべてが変換されるか、またはデータが何も変換されないことをお勧めします。ただし、変換前または変換中にメッセージが切り捨てられた場合は例外です。その場合は、バッファの末尾に不完全な項目が入っていることがあります (例えば、2 バイト文字の 1 バイト分、または 4 バイト整数の 3 バイト分など)。この状況では不完全項目を省略し、*OUTBUF* の未使用のバイトをヌルに設定することをお勧めします。しかし、配列またはストリング内の完全な要素または文字は、変換するようにしてください。
5. 出口が初めて必要になった時点で、キュー・マネージャーは、形式と同じ名前 (拡張子は除く) を持つオブジェクトをロードしようとします。ロードされるオブジェクトには、その形式名をもつメッセージを処理する出口が含まれていなければなりません。出口名と、その出口を含むオブジェクトの名前を同じにすることをお勧めします。ただし、すべての環境でこの条件が必要なわけではありません。

6. アプリケーションがキュー・マネージャーに接続されてから、*MDFMT* を使用する最初のメッセージを検索しようとするとき、出口の新しいコピーがロードされます。キュー・マネージャーが前回ロードしたコピーを破棄した場合は、別の機会に新しいコピーをロードすることもできます。したがって、出口では、静的ストレージを使用して、出口の呼び出しからその次の呼び出しへと情報を伝えるようにすることはしないでください。この 2 回の呼び出しの間に出口がアンロードされることがあります。
7. ユーザー作成の出口の名前が、キュー・マネージャーがサポートする組み込み形式の 1 つと同じ名前でも、組み込み変換ルーチンは置換されません。このような出口が呼び出される状況としては、次の場合があります。
 - ・ 組み込み変換ルーチンが *MDCSI* または *MDENC* への変換、あるいはこの 2 つのいずれかのフィールドからの変換を処理できない場合
 - ・ 組み込み変換ルーチンがデータの変換に失敗した場合 (変換不能のフィールドまたは文字がある場合など)
8. 出口の有効範囲は環境によって異なります。*MDFMT* 名は、他の形式とのクラッシュのリスクを最小限に抑えるよう選択しなければなりません。名前には、形式名を定義するアプリケーションを識別する文字で始めることをお勧めします。
9. データ変換出口は、*MQGET* 呼び出しを発行したプログラムの環境に似た環境で実行されます。環境には、アドレス・スペースおよびユーザー・プロファイル (適用される場合) が含まれます。このプログラムは、メッセージ変換をサポートしない宛先キュー・マネージャーにメッセージを送るメッセージ・チャネル・エージェントであることもあります。出口は、キュー・マネージャーの環境で実行されるわけではないので、キュー・マネージャーの整合性を損なうことはありません。
10. 出口により使用できる *MQI* 呼び出しは *MQXCNVC* のみです。他の *MQI* 呼び出しを使用しようとすると、その呼び出しは失敗し、理由コード *RC2219* が戻るか、その他の予測不能のエラーが起こります。
11. キュー・マネージャーは、*MQCONVX* という名前の入り口点を提供しません。出口の名前は、すべての環境で必須ではありませんが、形式名 (*MQMD* の *MDFMT* フィールドに含まれる名前) と同じにする必要があります。

パラメーター

MQCONVX 呼び出しには、以下のパラメーターがあります。

MQDXP (MQDXP) - 入出力

データ変換出口パラメーター・ブロック。

この構造体には出口の呼び出しに関する情報があります。出口はこの構造体に情報を設定して、変換の結果を表示します。この構造体のフィールドの詳細については、[1443 ページの『IBM i での MQDXP \(データ変換出口パラメーター\)』](#)を参照してください。

MQMD (MQMD) - 入出力

メッセージ記述子。

出口への入力時は、変換が行われなかった場合にアプリケーションに戻されるメッセージ記述子です。したがって、*INBUF* に含まれる未変換メッセージの *MDFMT*、*MDENC*、および *MDCSI* が含まれます。

注： 出口に渡された **MQMD** パラメーターは、常に *MQMD* の最新のバージョンです。これは、出口を呼び出すキュー・マネージャーにサポートされています。出口を異なる環境間で移植可能にする予定の場合、*MQMD* 内の *MDVER* フィールドをチェックして、出口がアクセスする必要のあるフィールドがその構造体にあることを確認する必要があります。

IBM i の場合、出口はバージョン 2 の *MQMD* に渡されます。

出力では、変換が正常に行われた場合、出口は *MDENC* および *MDCSI* フィールドを、アプリケーションによって要求された値に変更する必要があります。これらの変更は、アプリケーションに反映されます。出口が構造体に加えたこれ以外の変更は無視され、アプリケーションには反映されません。

出口が MQDXP 構造体の DXRES フィールドに XROK を戻すが、メッセージ記述子の MDENC および MDCSI フィールドを変更しない場合は、キュー・マネージャーは、これらのフィールドについて、出口への入力時に MQDXP 構造体の対応するフィールドが保持していた値を戻します。

INLEN (10 桁の符号付き整数) - 入力

INBUF の長さ (バイト)。

これは入力バッファー INBUF の長さで、出口により処理されるバイト数を指定します。INLEN は、変換前のメッセージ・データの長さと、MQGET 呼び出しでアプリケーションが指定するバッファーの長さの、どちらか小さい方です。

値は常にゼロより大きくなります。

INBUF (1 バイトのビット・ストリング x INLEN) - 入力

未変換メッセージが入るバッファー。

変換前のメッセージ・データが入ります。出口がデータを変換できない場合は、キュー・マネージャーは、出口の完了後に、このバッファーの内容をアプリケーションに戻します。

注：出口は INBUF を変更しません。このパラメーターが変更される場合、結果は未定義になります。

OUTLEN (10 桁の符号付き整数) - 出力

OUTBUF の長さ (バイト)。

出力バッファー OUTBUF の長さで、アプリケーションが MQGET 呼び出しで指定するバッファーの長さと同じです。

値は常にゼロより大きくなります。

OUTBUF (1 バイトのビット・ストリング x OUTLEN) - 出力

変換後のメッセージが入るバッファー。

出口からの出力時に、変換が成功した場合 (MQDXP パラメーターの DXRES フィールドの値 X 大韓民国によって示される)、OUTBUF には、アプリケーションに送信されるメッセージ・データが、要求された表現で含まれます。変換が失敗した場合、出口がこのバッファーに加えた変更はすべて無視されます。

RPG での呼び出し (ILE)

```
C*.1.....2.....3.....4.....5.....6.....7..
C                                CALLP      exitname(MQDXP : MQMD : INLEN :
C                                INBUF : OUTLEN : OUTBUF)
```

呼び出しのプロトタイプ定義は次のようになります。

```
D*.1.....2.....3.....4.....5.....6.....7..
Dexitname          PR          EXTPROC('exitname')
D* Data-conversion exit parameter block
D MQDXP              44A
D* Message descriptor
D MQMD              364A
D* Length in bytes of INBUF
D INLEN              10I 0 VALUE
D* Buffer containing the unconverted message
D INBUF              *    VALUE
D* Length in bytes of OUTBUF
D OUTLEN             10I 0 VALUE
D* Buffer containing the converted message
D OUTBUF             *    VALUE
```

製品センシティブ・プログラミング・インターフェースの終わり

ユーザー出口、API 出口、およびインストール可能サービス参照

ユーザー出口、API 出口、およびインストール可能サービス・アプリケーションを開発する際には、このセクションに記載するリンク情報を使用してください。

- [1457 ページの『MQIEP 構造体』](#)
- [1460 ページの『データ変換出口リファレンス』](#)
- [1464 ページの『MQ_PUBLISH_EXIT - パブリッシュ出口』](#)
- [1473 ページの『チャネル出口呼び出しおよびデータ構造体』](#)
- [1565 ページの『API 出口参照』](#)
- [1623 ページの『インストール可能サービス・インターフェースの参照情報』](#)

関連概念

[ユーザー出口、API 出口、および IBM MQ インストール可能サービス](#)

関連タスク

[キュー・マネージャーの機能の拡張](#)

MQIEP 構造体

MQIEP 構造体には、出口が作成することが許可されている関数呼び出しごとに 1 つのエントリー・ポイントが含まれています。

フィールド

StrucId

タイプ: MQCHAR4 - 入力

構造体 ID 値は次のとおりです。

MQIEP_STRUC_ID

バージョン

タイプ: MQLONG - 入力

構造体のバージョン番号。値は次のとおりです。

MQIEP_VERSION_1

バージョン 1 構造体バージョン番号。

MQIEP_CURRENT_VERSION

構造の現在のバージョン。

StrucLength

タイプ: MQLONG

MQIEP 構造体のサイズ (バイト)。値は次のとおりです。

MQIEP_LENGTH_1

フラグ

タイプ: MQLONG

関数のアドレスに関する情報を提供します。ライブラリーがスレッド化されているかどうかを示すフラグを、ライブラリーがクライアント・ライブラリーまたはサーバー・ライブラリーであるかどうかを示すフラグとともに使用できます。

ライブラリー情報がないことを指定するために、以下の値が使用されます。

MQIEPF_NONE

共有ライブラリーがスレッド化されているか非スレッド化されているかを指定するために、以下の値のいずれかが使用されます。

MQIEPF_NON_THREADED_LIBRARY

非スレッド化共有ライブラリー

MQIEPF_THREADED_LIBRARY

スレッド化共有ライブラリー

共有ライブラリーがクライアント共有ライブラリーであるかサーバー共有ライブラリーであるかを指定するために、以下の値のいずれかが使用されます。

MQIEPF_CLIENT_LIBRARY

クライアント共有ライブラリー

MQIEPF_LOCAL_LIBRARY

サーバー共有ライブラリー

Reserved

タイプ: MQPTR

MQBACK_Call

タイプ: PMQ_BACK_CALL

MQBACK 呼び出しのアドレス。

MQBEGIN_Call

タイプ: PMQ_BEGIN_CALL

MQBEGIN 呼び出しのアドレス。

MQBUFMH_Call

タイプ: PMQ_BUFMH_CALL

MQBUFMH 呼び出しのアドレス。

MQCB_Call

タイプ: PMQ_CB_CALL

MQCB 呼び出しのアドレス。

MQCLOSE_Call

タイプ: PMQ_CLOSE_CALL

MQCLOSE 呼び出しのアドレス。

MQCMIT_Call

タイプ: PMQ_CMIT_CALL

MQCMIT 呼び出しのアドレス。

MQCONN_Call

タイプ: PMQ_CONN_CALL

MQCONN 呼び出しのアドレス。

MQCONNX_Call

タイプ: PMQ_CONNX_CALL

MQCONNX 呼び出しのアドレス。

MQCRTMH_Call

タイプ: PMQ_CRTMH_CALL

MQCRTMH 呼び出しのアドレス。

MQCTL_Call

タイプ: PMQ_CTL_CALL

MQCTL 呼び出しのアドレス。

MQDISC_Call

タイプ: PMQ_DISC_CALL

MQDISC 呼び出しのアドレス。

MQDLTMH_Call

タイプ: PMQ_DLTMH_CALL

MQDLTMH 呼び出しのアドレス。

MQDLTMP_Call

タイプ: PMQ_DLTMP_CALL

MQDLTMP 呼び出しのアドレス。

MQGET_Call

タイプ: PMQ_GET_CALL

MQGET 呼び出しのアドレス。

MQINQ_Call

タイプ: PMQ_INQ_CALL

MQINQ 呼び出しのアドレス。

MQINQMP_Call

タイプ: PMQ_INQMP_CALL

MQINQMP 呼び出しのアドレス。

MQMHBUF_Call

タイプ: PMQ_MHBUF_CALL

MQMHBUF 呼び出しのアドレス。

MQOPEN_Call

タイプ: PMQ_OPEN_CALL

MQOPEN 呼び出しのアドレス。

MQPUT_Call

タイプ: PMQ_PUT_CALL

MQPUT 呼び出しのアドレス。

MQPUT1_Call

タイプ: PMQ_PUT1_CALL

MQPUT1 呼び出しのアドレス。

MQSET_Call

タイプ: PMQ_SET_CALL

MQSET 呼び出しのアドレス。

MQSETMP_Call

タイプ: PMQ_SETMP_CALL

MQSETMP 呼び出しのアドレス。

MQSTAT_Call

タイプ: PMQ_STAT_CALL

MQSTAT 呼び出しのアドレス。

MQSUB_Call

タイプ: PMQ_SUB_CALL

MQSUB 呼び出しのアドレス。

MQSUBRQ_Call

タイプ: PMQ_SUBRQ_CALL

MQSUBRQ 呼び出しのアドレス。

MQXCNVC_Call

タイプ: PMQ_XCNVC_CALL

MQXCNVNC 呼び出しのアドレス。

MQXCLWLN_Call

タイプ: PMQ_XCLWLN_CALL

MQXCLWLN 呼び出しのアドレス。

MQXDX_Call

タイプ: PMQ_XDX_CALL

MQXDX 呼び出しのアドレス。

MQXEP_Call

タイプ: PMQ_XEP_CALL

MQXEP 呼び出しのアドレス。

MQZEP_Call

タイプ: PMQ_ZEP_CALL

MQZEP 呼び出しのアドレス。

C 宣言

```
struct tagMQIEP {
    MQCHAR4      StrucId;           /* Structure identifier */
    MQLONG        Version;          /* Structure version number */
    MQLONG        StrucLength;      /* Structure length */
    MQLONG        Flags;            /* Flags */
    MQPTR         Reserved;         /* Reserved */
    PMQ_BACK_CALL MQBACK_Call;      /* Address of MQBACK */
    PMQ_BEGIN_CALL MQBEGIN_Call;    /* Address of MQBEGIN */
    PMQ_BUFMH_CALL MQBUFMH_Call;    /* Address of MQBUFMH */
    PMQ_CB_CALL   MQCB_Call;        /* Address of MQCB */
    PMQ_CLOSE_CALL MQCLOSE_Call;    /* Address of MQCLOSE */
    PMQ_CMtT_CALL MQCMIT_Call;      /* Address of MQCMIT */
    PMQ_CONN_CALL MQCONN_Call;      /* Address of MQCONN */
    PMQ_CONNX_CALL MQCONNX_Call;    /* Address of MQCONNX */
    PMQ_CRTMH_CALL MQCRTMH_Call;    /* Address of MQCRTMH */
    PMQ_CTL_CALL   MQCTL_Call;      /* Address of MQCTL */
    PMQ_DISC_CALL  MQDISC_Call;     /* Address of MQDISC */
    PMQ_DLTMH_CALL MQDLTMH_Call;    /* Address of MQDLTMH */
    PMQ_DLTMP_CALL MQDLTMP_Call;    /* Address of MQDLTMP */
    PMQ_GET_CALL   MQGET_Call;      /* Address of MQGET */
    PMQ_INQ_CALL   MQINQ_Call;      /* Address of MQINQ */
    PMQ_INQMP_CALL MQINQMP_Call;    /* Address of MQINQMP */
    PMQ_MHBUF_CALL MQMHBUF_Call;    /* Address of MQMHBUF */
    PMQ_OPEN_CALL  MQOPEN_Call;     /* Address of MQOPEN */
    PMQ_PUT_CALL   MQPUT_Call;      /* Address of MQPUT */
    PMQ_PUT1_CALL  MQPUT1_Call;     /* Address of MQPUT1 */
    PMQ_SET_CALL   MQSET_Call;      /* Address of MQSET */
    PMQ_SETMP_CALL MQSETMP_Call;    /* Address of MQSETMP */
    PMQ_STAT_CALL  MQSTAT_Call;     /* Address of MQSTAT */
    PMQ_SUB_CALL   MQSUB_Call;      /* Address of MQSUB */
    PMQ_SUBRQ_CALL MQSUBRQ_Call;    /* Address of MQSUBRQ */
    PMQ_XCLWLN_CALL MQXCLWLN_Call;  /* Address of MQXCLWLN */
    PMQ_XCNVC_CALL MQXCNVNC_Call;   /* Address of MQXCNVNC */
    PMQ_XDX_CALL   MQXDX_Call;      /* Address of MQXDX */
    PMQ_XEP_CALL   MQXEP_Call;      /* Address of MQXEP */
    PMQ_ZEP_CALL   MQZEP_Call;      /* Address of MQZEP */
};
```

データ変換出口リファレンス

z/OS では、必ずアセンブラー言語を使用してデータ変換出口を作成してください。それ以外のプラットフォームでは、C プログラミング言語を使用することをお勧めします。

データ変換出口プログラムの作成を支援するため、次のリソースが提供されています。

- スケルトン・ソース・ファイル
- 文字変換呼び出し

- データ・タイプ構造体でデータ変換を実行するコードのフラグメントを作成するユーティリティ。このユーティリティへの入力に使用できるのはCのみです。z/OSでは、このユーティリティによってアセンブラ・コードが生成されます。

このようなプログラムを作成する手順については、次を参照してください。

-  IBM MQ for IBM i 用のデータ変換出口プログラムの作成
-  IBM MQ for z/OS 用のデータ変換出口プログラムの作成
- IBM MQ for AIX or Linux システム用のデータ変換出口の作成
- IBM MQ for Windows 用のデータ変換出口の作成

スケルトン・ソース・ファイル

これらは、データ変換出口プログラムを作成するときの開始点として使用することができます。

用意されているファイルを、[1461 ページの表 852](#) にリストしています。

表 852. スケルトン・ソース・ファイル	
プラットフォーム	ファイル
 AIX	amqsvfc0.c
 IBM i	QMMSAMP/QCSRC(AMQSVFC4)
 Linux	amqsvfc0.c
 Windows システム	amqsvfc0.c
 z/OS	CSQ4BAX8 (1461 ページの『1』) CSQ4BAX9 (1461 ページの『2』) CSQ4CAX9 (1461 ページの『3』)
注： - MQXCVNC 呼び出しの例を示します。 - CICS 以外のすべての環境で使用するためにユーティリティによって生成されるコード断片用のラッパー。 - CICS 環境で使用するためにユーティリティによって生成されるコード断片用のラッパー。	

文字変換呼び出し

データ変換出口プログラム内から MQXCNVC (文字変換) 呼び出しを使用して、文字メッセージ・データを文字セット間で変換できます。特定のマルチバイト文字セット (例えば、UTF-16 文字セット) の場合は、適切なオプションを使用する必要があります。

この出口から他の MQI 呼び出しを実行することはできません。そのような呼び出しを実行しようとすると、理由コード MQRC_CALL_IN_PROGRESS で失敗します。

MQXCNVC 呼び出しと該当するオプションに関する詳細は、[915 ページの『MQXCNVC - 文字の変換』](#)を参照してください。

変換出口コードを作成するためのユーティリティ

この情報は、変換出口コードの作成についてさらに詳しく学習するために使用します。

変換出口コードを作成するためのコマンドは、以下のとおりです。

IBM i IBM i

CVTMQMMDTA (IBM MQ データ・タイプの変換)

ALW AIX, Linux, and Windows システム

crtmqcvx (IBM MQ 変換出口の作成)

z/OS z/OS

CSQUCVX

使用中のプラットフォームのコマンドにより、データ・タイプ構造体でデータ変換を実行するコードのフラグメントを作成し、使用中のデータ変換出口プログラムで使えるようにします。このコマンドは、1つ以上の C 言語の構造体定義を含むファイルを取ります。z/OS では、アセンブラー・コード・フラグメントと変換関数を含むデータ・セットが生成されます。その他のプラットフォームでは、それぞれの構造体定義を変換するための C 関数を含むファイルが生成されます。z/OS では、このユーティリティーに LE/370 ランタイム・ライブラリー SCEERUN へのアクセス権限が必要です。

z/OS での CSQUCVX ユーティリティーの呼び出し

z/OS

1462 ページの図 9 は、CSQUCVX ユーティリティーを呼び出すための JCL の例です。

```
//CVX EXEC PGM=CSQUCVX
//STEPLIB DD DISP=SHR,DSN=thlqual.SCSQANLE
// DD DISP=SHR,DSN=thlqual.SCSQLOAD
// DD DISP=SHR,DSN=le370qual.SCEERUN
//SYSPRINT DD SYSOUT=*
//CSQUINP DD DISP=SHR,DSN=MY.MQSERIES.FORMATS(MSG1)
//CSQUOUT DD DISP=OLD,DSN=MY.MQSERIES.EXIT(S(MSG1)
```

図 9. CSQUCVX ユーティリティーを呼び出すために使用されるサンプル JCL

z/OS データ定義ステートメント

z/OS

CSQUCVX ユーティリティーは、1462 ページの表 853 に示す次の DD 名を持った DD ステートメントを必要とします。

表 853. データ定義ステートメント名と説明	
DD ステートメント	説明
SYSPRINT	レポートおよびエラー・メッセージのためのデータ・セットまたはプリント・スプール・クラスを指定します。
CSQUINP	変換するデータ構造体の定義を含む区分データ・セットを指定します。
CSQUOUT	変換コードのフラグメントを書き込む区分データ・セットを指定します。論理レコード長 (LRECL) は必ず 80 であり、レコード形式 (RECFM) は必ず FB である必要があります。

AIX, Linux, and Windows システムにおけるエラー・メッセージ

crtmqcvx コマンドは、AMQ7953 から AMQ7970 までの範囲のメッセージを戻します。

これらのメッセージは、メッセージとエラーコード「IBM MQ メッセージ」にリストされています。

主に、次の 2 つのタイプのエラーがあります。

- 構文エラーなどの、処理が続行できない場合の主要なエラー。

画面上に、入力ファイル内のエラーのある行番号を示すメッセージが表示されます。出力ファイルが部分的に作成されている可能性があります。

- 問題が検出されたことを示すメッセージが表示されるが、構造体の構文解析は続行可能な場合の、それ以外のエラー。

発生した問題に関するエラー情報を含んだ出力ファイルが作成されます。作成したコードが、それらの問題を修正するための介入なしでコンパイラに受け入れられることがないように、このエラー情報には接頭部 `#error` が付けられます。

有効な構文

ユーティリティの入力ファイルは、C 言語の構文に準拠している必要があります。

C についての知識がない場合は、このトピックの [C の例](#) を参照してください。

加えて、以下の規則に注意してください。

- `struct` キーワードの前では、`typedef` しか認識されません。
- 構造体を宣言するときには、構造体タグが必要です。
- 空の大括弧 `[]` を使用して、メッセージの最後に可変長の配列またはストリングを指示できます。
- 多次元配列やストリングの配列はサポートされません。
- 次の追加データ・タイプが認識されます。

- `MQBOOL`
- `MQBYTE`
- `MQCHAR`
- `MQFLOAT32`
- `MQFLOAT64`
- `MQSHORT`
- `MLONG`
- `MQINT8`
- `MQUINT8`
- `MQINT16`
- `MQUINT16`
- `MQINT32`
- `MQUINT32`
- `MQINT64`
- `MQUINT64`

`MQCHAR` フィールドは変換されたコード・ページですが、`MQBYTE`、`MQINT8`、および `MQUINT8` は未処理のままです。エンコード方式が異なる場合、`MQSHORT`、`MLONG`、`MQINT16`、`MQUINT16`、`MQINT32`、`MQUINT32`、`MQINT64`、`MQUINT64`、`MQFLOAT32`、`MQFLOAT64`、および `MQBOOL` はそれに沿って変換されます。

- 以下のデータ・タイプは使用しないでください。

- `double`
- `pointers`
- `bit-fields`

これは、変換出口コードを作成するためのユーティリティには、これらのデータ・タイプを変換する機能がないためです。この問題を解決するには、独自のルーチンを作成して、その出口からそれらのルーチンを呼び出すようにできます。

上記以外の注意点を、以下に示します。

- 入力データ・セット内で順序番号は使用しないでください。
- 独自の変換ルーチンを指定したいフィールドがあれば、それらを MQBYTE として宣言し、生成された CMQXCFBA マクロを独自の変換コードに置き換えます。

C の例

```
struct TEST { MQLONG    SERIAL_NUMBER;
              MQCHAR    ID[5];
              MQINT16   VERSION;
              MQBYTE    CODE[4];
              MQLONG    DIMENSIONS[3];
              MQCHAR    NAME[24];
            } ;
```

この構文に対応する、その他のプログラム言語での宣言を以下に示します。

COBOL

```
10 TEST.
  15 SERIAL-NUMBER PIC S9(9) BINARY.
  15 ID             PIC X(5).
  15 VERSION        PIC S9(4) BINARY.
  * CODE IS NOT TO BE CONVERTED
  15 CODE           PIC X(4).
  15 DIMENSIONS     PIC S9(9) BINARY OCCURS 3 TIMES.
  15 NAME           PIC X(24).
```

System/390

```
TEST          EQU *
SERIAL_NUMBER DS F
ID            DS CL5
VERSION       DS H
CODE          DS XL4
DIMENSIONS    DS 3F
NAME         DS CL24
```

PL/I

z/OS でのみサポートされています。

```
DCL 1 TEST,
  2 SERIAL_NUMBER FIXED BIN(31),
  2 ID             CHAR(5),
  2 VERSION        FIXED BIN(15),
  2 CODE           CHAR(4),          /* not to be converted */
  2 DIMENSIONS(3)  FIXED BIN(31),
  2 NAME           CHAR(24);
```

MQ_PUBLISH_EXIT - パブリッシュ出口

MQ_PUBLISH_EXIT 呼び出しは、サブスクライバーに送信されるメッセージを検査したり、変更したりできます。

目的

パブリッシュ出口を使用すると、以下のようにサブスクライバーに送信されるメッセージを検査したり、変更したりできます。

- 各サブスクライバーにパブリッシュされるメッセージの内容を検査する
- 各サブスクライバーにパブリッシュされるメッセージの内容を変更する
- メッセージが入れられるキューを変更する
- サブスクライバーへのメッセージの送達を停止する

この出口は IBM MQ for z/OS では使用できません。

構文

MQ_PUBLISH_EXIT (*ExitParms*, *PubContext*, *SubContext*)

パラメーター

ExitParms (MQPSXP) - Input/Output

ExitParms には、出口の呼び出しに関する情報が記載されています。

PubContext (MQPBC) - Input

PubContext には、パブリケーションのパブリッシャーに関するコンテキスト情報が含まれます。

SubContext (MQSBC) - Input/Output

SubContext には、パブリケーションを受け取るサブスクライバーに関するコンテキスト情報が含まれます。

MQPSXP - パブリッシュ出口データ構造体

MQPSXP 構造体は、パブリッシュ出口に渡される情報およびパブリッシュ出口から戻される情報を記述します。

1465 ページの表 854 で、この構造体のフィールドの概要を示します。

表 854. MQPSXP 内のフィールド	
フィールド	説明
<u>StrucID</u>	構造体 ID
<u>Version</u>	構造体のバージョン番号
<u>ExitId</u>	呼び出される出口のタイプ
<u>ExitReason</u>	出口を呼び出す理由
<u>ExitResponse</u>	出口からの応答
<u>ExitResponse2</u>	出口からの 2 次応答
<u>Feedback</u>	フィードバック・コード
<u>ExitUserArea</u>	出口ユーザー域
<u>ExitData</u>	出口データ
<u>QMgrName</u>	ローカル・キュー・マネージャーの名前
<u>Hconn</u>	接続ハンドル
<u>MsgDescPtr</u>	メッセージ記述子 (MQMD) のアドレス
<u>MsgHandle</u>	メッセージ・プロパティ (MQHMSG) のハンドル
<u>MsgInPtr</u>	入力メッセージのアドレス
<u>MsgInLength</u>	入力メッセージの長さ
<u>MsgOutPtr</u>	出力メッセージのアドレス
<u>MsgOutLength</u>	出力メッセージの長さ
<u>pEntryPoints</u>	MQIEP 構造体のアドレス

フィールド

StrucID (MQCHAR4)

StrucID は、構造体の ID です。値は次のとおりです。

MQPSXP_STRUCID

MQPSXP_STRUCID は、パブリッシュ出口パラメーター構造体の ID です。C プログラミング言語の場合、定数 MQPSXP_STRUC_ID_ARRAY も定義されます。この値は MQPSXP_STRUC_ID と同じ値ですが、ストリングではなく文字の配列です。

StrucID は、出口への入力フィールドです。

Version (MQLONG)

Version は、構造体のバージョン番号です。値は次のとおりです。

MQPSXP_VERSION_1

MQPSXP_VERSION_1 は、バージョン 1 パブリッシュ出口パラメーター構造体です。定数 MQPSXP_CURRENT_VERSION も同じ値で定義されます。

Version は、出口への入力フィールドです。

ExitId (MQLONG)

ExitId は、呼び出される出口のタイプです。値は次のとおりです。

MQXT_PUBLISH_EXIT

パブリッシュ出口。

ExitId は、出口への入力フィールドです。

ExitReason (MQLONG)

ExitReason は、出口を呼び出す理由です。指定できる値は以下のとおりです。

MQXR_INIT

この接続の出口は、初期化のために呼び出されます。出口は必要なリソース (例えば、主ストレージ) を獲得して初期化することがあります。

MQXR_TERM

この接続の出口は、停止しようとしているために、呼び出されます。出口は、初期化された後で獲得したすべてのリソース (例えば、主ストレージ) を解放する必要があります。

MQXR_PUBLICATION

出口は、パブリケーションをサブスクライバーのメッセージ・キューに書き込む前に、キュー・マネージャーに呼び出されます。この出口は、メッセージを変更できますが、キューにメッセージを書き込んだり、パブリケーションを一時停止したりすることはできません。

ExitReason は、出口への入力フィールドです。

ExitResponse (MQLONG)

ExitResponse は、処理を継続する方法を指定するため、出口において設定します。*ExitResponse* は以下のいずれかの値です。

MQXCC_OK

MQXCC_OK を設定すると、処理は正常に継続します。ExitReason の任意の値への応答として、MQXCC_OK を設定します。

ExitReason の値が MQXR_PUBLICATION である場合、MQSBC 構造体の *DestinationQName* フィールドおよび *DestinationQMGrName* フィールドは、メッセージを送信する宛先を指定します。

MQXCC_FAILED

MQXCC_FAILED を設定すると、パブリッシュ操作が停止します。完了コード MQCC_FAILED および理由コード 2557 (09FD) (RC2557): MQRC_PUBLISH_EXIT_ERROR は、出口からの戻り時に設定されます。

MQXCC_SUPPRESS_FUNCTION

MQXCC_SUPPRESS_FUNCTION を設定すると、メッセージの通常処理が停止します。

MQXCC_SUPPRESS_FUNCTION は、*ExitReason* の値が MQXR_PUBLICATION である場合にのみ設定します。

メッセージのメッセージ記述子の *Report* フィールドにある MQRO_DISCARD_MSG オプションに応じて、キュー・マネージャーはメッセージの処理を続けます。

- MQRO_DISCARD_MSG オプションが指定されている場合には、メッセージはサブスクライバーに送信されません。
- MQRO_DISCARD_MSG オプションが指定されていない場合は、メッセージは送達不能キューに入れます。送達不能キューがない場合、またはメッセージを正常に送達不能キューに入れることができない場合は、パブリケーションはサブスクライバーに送信されません。他のサブスクライバーへのパブリケーションの送信は、PMSGDLV トピック・オブジェクト属性および NPMMSGDLV トピック・オブジェクト属性の値に応じて決まります。これらの属性の説明については、[DEFINE TOPIC](#) コマンドのパラメーターの説明を参照してください。

ExitResponse は、出口からの出力フィールドです。

ExitResponse2 (MQLONG)

ExitResponse2 は、将来の使用のために予約されています。

Feedback (MQLONG)

Feedback は、出口が *ExitResponse* で MQXCC_SUPPRESS_FUNCTION を戻した場合に使用されるフィールドバック・コードです。

出口への入力では、*Feedback* の値は常に MQFB_NONE になっています。出口が MQXCC_SUPPRESS_FUNCTION を戻す場合は、*Feedback* を、キュー・マネージャーがメッセージを送達不能キューに入れるときにそのメッセージに使用される値に設定します。出口から戻る時点で、*Feedback* に元の値 MQFB_NONE が設定されていると、キュー・マネージャーは *Feedback* を MQFB_STOPPED_BY_PUBSUB_EXIT に設定します。

Feedback は、出口の入出力フィールドです。

ExitUserArea (MQBYTE16)

ExitUserArea は出口で利用できるフィールドです。各接続に個別の *ExitUserArea* があります。*ExitUserArea* の長さは、MQ_EXIT_USER_AREA_LENGTH により指定されます。

出口の最初の呼び出しでは、*ExitReason* フィールドには MQXR_INIT の値が設定されています。接続のために出口が最初に呼び出されたときに、*ExitUserArea* が MQXUA_NONE に初期化されます。それ以降に *ExitUserArea* に対して行われる変更は、出口の呼び出しを通じて保持されます。

ExitUserArea は、出口の入出力フィールドです。

ExitData (MQCHAR32)

ExitData は、キュー・マネージャーの初期設定ファイル内のスタンザの **PublishExitData** パラメーターにより定義される固定出口データです。データには、フィールドの全長になるまで空白が埋め込まれます。初期設定ファイルで固定出口データが定義されていない場合、*ExitData* は空白になります。*ExitData* の長さは、MQ_EXIT_DATA_LENGTH により指定されます。

ExitData は、出口への入力フィールドです。

QMgrName (MQCHAR48)

QMgrName は、ローカル・キュー・マネージャーの名前です。この名前には、フィールドの全長になるまで空白が埋め込まれます。このフィールドの長さは、MQ_Q_MGR_NAME_LENGTH によって指定されます。

QMgrName は、出口への入力フィールドです。

Hconn (MQHCONN)

Hconn は、キュー・マネージャーへの接続を表すハンドルです。 *Hconn* は、メッセージ・プロパティを処理する MQSETMP、MQINQMMP、または MQDLTMP の各メッセージ・プロパティ関数呼び出しに対するパラメーターとしてのみ使用します。

Hconn は、出口への入力フィールドです。

MsgDescPtr (PMQMD)

MsgDescPtr は、処理されるメッセージのメッセージ記述子 (MQMD) のアドレスであり、MQPUT 呼び出しから返される MQMD のコピーです。 出口は、メッセージ記述子の内容を変更できます。 メッセージ記述子の内容の変更は、注意して行う必要があります。 特に、MQSBC 構造の *SubType* フィールドの値が MQSUBTYPE_PROXY である場合、メッセージ記述子の *CorrelId* フィールドは変更しないでください。

ExitReason が MQXR_INIT または MQXR_TERM の場合、メッセージ記述子は出口に渡されません。 その場合、*MsgDescPtr* は NULL ポインターになります。

MsgDescPtr は、出口への入力フィールドです。

MsgHandle (MQHMSG)

MsgHandle は、メッセージ・プロパティのハンドルです。 *MsgHandle* は必ず、メッセージ・プロパティを処理する MQSETMP、MQINQMMP、または MQDLTMP の各メッセージ・プロパティ関数呼び出しと共に使用します。

MsgHandle は、出口への入力フィールドです。

MsgInPtr (PMQVOID)

MsgInPtr は、入力メッセージ・データのアドレスです。 *MsgInPtr* によりアドレス指定されたバッファーの内容は、出口により変更される可能性があります。 [MsgOutPtr](#) を参照してください。

MsgInPtr は、出口への入力フィールドです。

MsgInLength (MQLONG)

MsgInLength は、出口に渡されるメッセージ・データの、バイト単位の長さです。 データのアドレスは、*MsgInPtr* によって指定されます。

MsgInLength は、出口への入力フィールドです。

MsgOutPtr (PMQVOID)

MsgOutPtr は、出口から戻されるメッセージ・データが入るバッファーのアドレスです。 出口に入る時点で、*MsgOutPtr* は NULL です。 出口から戻る時点で、この値が NULL のままになっている場合は、キュー・マネージャーは *MsgInPtr* で指定されたメッセージを、*MsgInLength* で指定された長さで送信します。

出口でメッセージ・データを変更する場合は、次のいずれかの手順を使用してください。

- データの長さが変更されない場合、データは、*MsgInPtr* でアドレス指定されたバッファー内で変更できます。 その場合、*MsgOutPtr* と *MsgOutLength* は変更しないでください。
- 変更されたデータの長さが元のデータより短い場合、データは、*MsgInPtr* でアドレス指定されたバッファー内で変更できます。 その場合、*MsgOutPtr* に入力メッセージ・バッファーのアドレスを指定し、*MsgOutLength* にメッセージ・データの新しい長さを指定してください。
- 変更されたデータが元のデータより長い場合 (または長くなる可能性がある場合)、出口は新しいメッセージ・バッファーを取得する必要があります。 そこに変更されたデータをコピーしてください。 *MsgOutPtr* を新しいバッファーのアドレスに設定し、*MsgOutLength* を新しいメッセージ・データの長さに設定します。 出口は、次に呼び出されるときに、*MsgOutPtr* によりアドレス指定されるバッファーを解放する必要があります。

注: *MsgOutPtr* は常に出口に対する入力の NULL ポインターであり、以前に取得されたメッセージ・バッファーのアドレスではありません。 以前に取得したバッファーを解放するには、出口はそのバッファーのアドレスと長さを保存する必要があります。 *ExitUserArea*、または *ExitUserArea* にそのアドレスが保存されている制御ブロックにこの情報を保存します。

MsgOutPtr は、出口の入出力フィールドです。

MsgOutLength (MQLONG)

MsgOutLength は、出口から戻されるメッセージ・データの、バイト単位の長さです。出口への入力では、このフィールドは常にゼロになっています。出口から戻る時点で、*MsgOutPtr* が NULL になっている場合、このフィールドは無視されます。メッセージ・データの変更について詳しくは、[MsgOutPtr](#) を参照してください。

MsgOutLength は、出口の入出力フィールドです。

pEntryPoints (PMQIEP)

pEntryPoints は、MQIEP 構造体のアドレスです。ここから MQI 呼び出しおよび DCI 呼び出しを行うことができます。

C 言語宣言 - MQPSXP

```
typedef struct tagMQPSXP {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQLONG     ExitId;           /* Type of exit */
    MQLONG     ExitReason;       /* Reason for invoking exit */
    MQLONG     ExitResponse;     /* Response from exit */
    MQLONG     ExitResponse2;    /* Reserved */
    MQLONG     Feedback;        /* Feedback code */
    MQBYTE16   ExitUserArea;     /* Exit user area */
    MQCHAR32   ExitData;        /* Exit data */
    MQCHAR48   QMgrName;        /* Name of local queue manager */
    MQHCONN    Hconn;           /* Connection handle */
    MQHMSG     MsgHandle;        /* Handle to message properties */
    PMQMD      MsgDescPtr;      /* Address of message descriptor */
    PMQVOID    MsgInPtr;        /* Address of input message data */
    MQLONG     MsgInLength;     /* Length of input message data */
    PMQVOID    MsgOutPtr;       /* Address of output message data */
    MQLONG     MsgOutLength;    /* Length of output message data */
    /* Ver:1 */
    PMQIEP     pEntryPoints;    /* Address of the MQIEP structure */
    /* Ver:2 */
} MQPSXP;
```

MQPBC - パブリケーション・コンテキスト・データ構造体

MQPBC 構造体には、パブリッシュ出口に渡される、パブリケーションのパブリッシャーに関連するコンテキスト情報が含まれます。

1469 ページの表 855 で、この構造体のフィールドの概要を示します。

表 855. MQPBC 内のフィールド	
フィールド	説明
<i>StrucID</i>	構造体 ID
<i>Version</i>	構造体のバージョン番号
<i>PubTopicString</i>	パブリッシュ・トピック・ストリング
<i>MsgDescPtr</i>	メッセージ記述子 (MQMD) のアドレス

フィールド

StrucID (MQCHAR4)

StrucID は、構造体の ID です。値は次のとおりです。

MQPBC_STRUCID

MQPBC_STRUCID は、パブリケーション・コンテキスト構造体の ID です。C プログラミング言語の場合、定数 MQPBC_STRUC_ID_ARRAY も定義されます。この値は MQPBC_STRUC_ID と同じ値ですが、ストリングではなく文字の配列です。

StrucID は、出口への入力フィールドです。

Version (MQLONG)

Version は、構造体のバージョン番号です。値は次のとおりです。

MQPBC_VERSION_1

MQPBC_VERSION_1 は、バージョン 1 パブリッシュ出口パラメーター構造体です。

MQPBC_VERSION_2

MQPBC_VERSION_2 は、バージョン 2 パブリッシュ出口パラメーター構造体です。定数 *MQPBC_CURRENT_VERSION* も同じ値で定義されます。

Version は、出口への入力フィールドです。

PubTopicString (MQCHARV)

PubTopicString は、パブリッシュ対象のトピック・ストリングです。

PubTopicString は、出口への入力フィールドです。

MsgDescPtr (PMQMD)

MsgDescPtr は、処理中のメッセージのメッセージ記述子 (MQMD) のコピーのアドレスを示します。

MsgDescPtr は、出口への入力フィールドです。

C 言語宣言 - MQPBC

```
typedef struct tagMQPBC {  
    MQCHAR4      StrucID;           /* Structure identifier */  
    MQLONG       Version;          /* Structure version number */  
    MQCHARV      PubTopicString;    /* Publish topic string */  
    PMQMD        MsgDescPtr;       /* Address of message descriptor */  
} MQPBC;
```

MQSBC - サブスクリプション・コンテキスト・データ構造体

MQSBC 構造体には、パブリッシュ出口に渡される、パブリケーションを受け取るサブスクライバーに関連するコンテキスト情報が含まれます。

1470 ページの表 856 で、この構造体のフィールドの概要を示します。

表 856. MQSBC 内のフィールド	
フィールド	説明
<u><i>StrucID</i></u>	構造体 ID
<u><i>Version</i></u>	構造体のバージョン番号
<u><i>DestinationQMGrName</i></u>	宛先キュー・マネージャーの名前
<u><i>DestinationQName</i></u>	宛先キューの名前
<u><i>SubType</i></u>	サブスクリプションのタイプ
<u><i>SubOptions</i></u>	サブスクリプション・オプション
<u><i>ObjectName</i></u>	オブジェクト名
<u><i>ObjectString</i></u>	オブジェクト・ストリング
<u><i>SubTopicString</i></u>	サブスクリプション・トピック・ストリング
<u><i>SubName</i></u>	サブスクリプション名
<u><i>SubId</i></u>	サブスクリプション ID
<u><i>SelectionString</i></u>	選択ストリングのアドレス

表 856. MQSBC 内のフィールド (続き)	
フィールド	説明
<u>SubLevel</u>	サブスクリプション・レベル
<u>PSPProperties</u>	パブリッシュ/サブスクライブ・プロパティ

フィールド

StrucID (MQCHAR4)

構造体 ID 値は次のとおりです。

MQSBC_STRUCID

MQSBC_STRUCID は、パブリッシュ出口パラメーター構造体の ID です。C プログラミング言語の場合、定数 MQSBC_STRUC_ID_ARRAY も定義されます。MQSBC_STRUC_ID_ARRAY の値は MQSBC_STRUC_ID と同じ値ですが、ストリングではなく文字の配列です。

StrucID は、出口への入力フィールドです。

Version (MQLONG)

構造体のバージョン番号。値は次のとおりです。

MQSBC_VERSION_1

バージョン 1 パブリッシュ出口パラメーター構造。定数 MQSBC_CURRENT_VERSION も同じ値で定義されます。

Version は、出口への入力フィールドです。

DestinationQMgrName (MQCHAR48)

DestinationQMgrName は、メッセージの送信先のキュー・マネージャーの名前です。この名前には、フィールドの全長になるまでブランクが埋め込まれます。この名前は、出口によって変更される可能性があります。このフィールドの長さは、MQ_Q_MGR_NAME_LENGTH によって指定されます。

DestinationQMgrName は、出口の入出力フィールドです。[注](#)を参照してください。

DestinationQName (MQCHAR48)

DestinationQName は、メッセージの送信先キューの名前です。この名前には、フィールドの全長になるまでブランクが埋め込まれます。この名前は、出口によって変更される可能性があります。このフィールドの長さは、MQ_Q_NAME_LENGTH によって指定されます。

DestinationQName は、出口の入出力フィールドです。[注](#)を参照してください。

SubType (MQLONG)

SubType は、サブスクリプションが作成された方法を示します。有効な値は MQSUBTYPE_API、MQSUBTYPE_ADMIN、および MQSUBTYPE_PROXY です。[Inquire Subscription Status \(応答\)](#) を参照してください。

SubType は、出口への入力フィールドです。

SubOptions (MQLONG)

SubOptions は、サブスクリプション・オプションです。このフィールドで使用できる値については、[557 ページの『Options \(MQLONG\)』](#)を参照してください。

SubOptions は、出口への入力フィールドです。

ObjectName (MQCHAR48)

ObjectName は、ローカル・キュー・マネージャーで定義されているトピック・オブジェクトの名前です。このフィールドの長さは、MQ_TOPIC_NAME_LENGTH によって指定されます。オブジェクト名は、キュー・マネージャーがトピック・ストリングと関連付けた管理トピック・オブジェクトの名前です。サブスクライバーがサブスクリプションの一部としてトピック・オブジェクトを提供している場合であっても、ObjectName は異なるトピック・オブジェクトであることがあります。トピック・オブジェクトとサブスクリプションの関連は、SubTopicString の完全な解決に依存します。

ObjectName は、出口への入力フィールドです。

ObjectString (MQCHARV)

ObjectString は、サブスクライブされたパブリケーションのフル・トピック・ストリングです。元のサブスクリプション・ストリングに含まれるワイルドカードはすべて解決されます。これは、ワイルドカードを含む可能性があり、サブスクライバーが提供するオブジェクト名を一切含まない、[567 ページの『ObjectString \(MQCHARV\)』](#)で説明されている MQSD サブスクリプションフィールド *ObjectString* とは異なります。

ObjectString は、出口への入力フィールドです。

SubTopicString (MQCHARV)

SubTopicString は、サブスクライバーによって提供される完全なトピック・ストリングです。

SubTopicString は、トピック・オブジェクトで定義されるトピック・ストリングと、任意の1つのトピック・ストリングの組み合わせです。サブスクライバーは、トピック・オブジェクト、トピック・ストリング、またはその両方のいずれかを提供する必要があります。サブスクライバーがトピック・ストリングを提供する場合、ワイルドカードが含まれることがあります。

SubTopicString は、出口への入力フィールドです。

SubName (MQCHARV)

SubName は、サブスクライバーにより提供されるサブスクリプション名、または生成された名前です。

SubName は、出口への入力フィールドです。

SubId (MQBYTE 24)

SubId は、固有内部サブスクリプション ID です。

SubId は、出口への入力フィールドです。

SelectionString (MQCHARV)

SelectionString は、トピックからメッセージをサブスクライブする際に使用する選択基準です。[セレクトター](#)を参照してください。

SelectionString は、出口への入力フィールドです。

SubLevel (MQLONG)

SubLevel は、サブスクリプションに関連付けられたインターセプト・レベルです。詳しくは、[570 ページの『SubLevel \(MQLONG\)』](#)を参照してください。

SubLevel は、出口への入力フィールドです。

PSProperties (MQLONG)

PSProperties は、パブリッシュ/サブスクライブ・プロパティーです。これは、パブリッシュ/サブスクライブに関連したメッセージ・プロパティーが、このサブスクリプションに送信されるメッセージにどのように追加されるかを指定します。可能な値は MQPSPROP_NONE, MQPSPROP_COMPAT, MQPSPROP_RFH2, です。これらの値の説明については、[オプションパラメータ \(変更 MQPSPROP_MSGPROP、コピー、およびサブスクリプションの作成\)](#)を参照してください。

PSProperties は、出口への入力フィールドです。

注：許可検査は、*DestinationQMgrName* および *DestinationQName* の元の値に対してのみ、それらがパブリッシュ出口に渡される前に実行されます。*DestinationQMgrName* または *DestinationQName* のいずれかを変更することにより、出口が宛先キューを変更した場合に、新しい許可検査が実行されることはありません。

C 言語宣言 - MQSBC

```
typedef struct tagMQSBC {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQCHAR48   DestinationQMgrName; /* Destination queue manager */
    MQCHAR48   DestinationQName; /* Destination queue name */
    MQLONG     SubType;          /* Type of subscription */
}
```

```

MQLONG      SubOptions;           /* Subscription options */
MQCHAR48    ObjectName;          /* Object name */
MQCHARV     ObjectString;        /* Object string */
MQCHARV     SubTopicString;      /* Subscription topic string */
MQCHARV     SubName;             /* Subscription name */
MQBYTE24    SubId;               /* Subscription identifier */
MQCHARV     SelectionString;     /* Subscription selection string */
MQLONG      SubLevel;            /* Subscription level */
MQLONG      PSPProperties;        /* Publish/subscribe properties */
} MQSBC;

```

チャネル出口呼び出しおよびデータ構造体

ここでの一連のトピックでは、チャネル出口プログラムを作成する際に使用できる特定の IBM MQ 呼び出しおよびデータ構造体に関する参照情報を記載しています。

この情報は、プロダクト・センシティブ・プログラミング・インターフェース情報です。次のプログラミング言語で IBM MQ ユーザー出口を作成することができます。

表 857. IBM MQ ユーザー出口: プラットフォームとプログラミング言語	
プラットフォーム	プログラム言語
IBM MQ for z/OS	アセンブラーおよび C (システム出口の C システム・プログラミング環境に合致するものでなければなりません。これについては、「z/OS C/C++ プログラミングの手引き」に記載されています。)
IBM MQ for IBM i	ILE C、ILE COBOL、および ILE RPG
他のすべての IBM MQ プラットフォーム	C    (x86 アーキテクチャでのみ 64 ビット対応)

また、ユーザー・エグジットを記述することもできます。Java ただし、これ Java らは JMS 特定のアプリケーションでのみ使用可能です。チャネル出口を IBM MQ classes for Java で作成および使用する場合は詳細については、[IBM MQ classes for Java](#) でのチャネル出口の使用を、IBM MQ classes for JMS の場合の詳細については、[IBM MQ classes for JMS](#) でのチャネル出口の使用を参照してください。

IBM MQ ユーザー出口を、TAL または Visual Basic で作成することはできません。しかし、Visual Basic には MQCD 構造体の宣言が備えられていて、IBM MQ MQI client プログラムからの MQCONNX 呼び出しで使用できます。

以降の説明では多くの場合、パラメーターはサイズが固定されていない配列または文字ストリングです。これらのパラメーターの場合は、小文字の "n" を使用して数値定数を表します。そのパラメーターの宣言がコーディングされている場合は、"n" を必要な数値で置き換える必要があります。これらの説明で使用する規則の詳細については、[177 ページの『基本データ・タイプ』](#)を参照してください。

データ定義ファイル

データ定義ファイルは、サポートされている各プログラミング言語用の IBM MQ と共に出荷されます。これらのファイルの詳細については、[COPY、ヘッダー、インクルード、IBM MQ およびモジュールファイル](#)を参照してください。

MQ_CHANNEL_EXIT - チャネル出口

MQ_CHANNEL_EXIT 呼び出しは、メッセージ・チャネル・エージェントによって呼び出される各チャネル出口に渡されるパラメーターを記述します。

MQ_CHANNEL_EXIT という名前の入り口点が、キュー・マネージャーによって提供されるわけではありません。チャネル出口の名前はチャネル定義 MQCD によって提供されるため、MQ_CHANNEL_EXIT という名前には特別の意味はありません。

チャネル出口プログラムのタイプは次の 5 つです。

- チャネル・セキュリティ出口
- チャネル・メッセージ出口

- チャンネル送信出口
- チャンネル受信出口
- チャンネル・メッセージ再試行出口

どのタイプの出口でも、パラメーターは類似しており、ここで述べる説明は、特に明記されているものを除き、すべてのタイプに適用されます。

構文

MQ_CHANNEL_EXIT (*ChannelExitParms*, *ChannelDefinition*, *DataLength*, *AgentBufferLength*, *AgentBuffer*, *ExitBufferLength*, *ExitBufferAddr*)

パラメーター

MQ_CHANNEL_EXIT 呼び出しには次のような 2 つのパラメーターがあります。

ChannelExitParms (MQCXP) - 入出力

チャンネル出口パラメーター・ブロック。

この構造体には、出口の呼び出しに関連する追加情報が入っています。出口は、MCA が処理を続行する方法を示す情報を、この構造体の中に設定します。

ChannelDefinition (MQCD) - 入出力

チャンネル定義。

この構造体には、チャンネルの動作を制御するためのパラメーターが管理者によって組み込まれます。

DataLength (MQLONG) - 入出力

データの長さ。

データは、出口のタイプによって異なります。

- チャンネル・セキュリティー出口の場合、*ExitReason* が MQXR_SEC_MSG になっているときには、この出口が呼び出されると、このパラメーターには *AgentBuffer* フィールド内のセキュリティー・メッセージの長さが入ります。メッセージがない場合には、ゼロに設定されます。出口は、*ExitResponse* を MQXCC_SEND_SEC_MSG または MQXCC_SEND_AND_REQUEST_SEC_MSG に設定する場合は、このフィールドをパートナーに送るセキュリティー・メッセージの長さに設定しなければなりません。メッセージ・データは *AgentBuffer* または *ExitBufferAddr* のいずれかに入っています。

セキュリティー出口で設定しなければならないのは、セキュリティー・メッセージの内容だけです。

- チャンネル・メッセージ出口の場合、出口が呼び出されると、このパラメーターにメッセージの長さ (伝送キュー・ヘッダーも含む) が入ります。出口は、処理される *AgentBuffer* または *ExitBufferAddr* 内のメッセージの長さに合わせて、このフィールドを設定しなければなりません。これは伝送キュー・ヘッダー (MQXQH) の長さ以上でなければなりません。
- チャンネル送信または受信出口の場合、出口が呼び出されると、このパラメーターに伝送の長さが入ります。出口は、処理される *AgentBuffer* または *ExitBufferAddr* 内の伝送の長さに合わせて、このフィールドを設定しなければなりません。

セキュリティー出口がメッセージを送信しているときに、チャンネルの反対側にセキュリティー出口がない場合、またはチャンネルの反対側が *ExitResponse* として MQXCC_OK を設定する場合は、開始出口は MQXR_SEC_MSG で再び呼び出され、ヌルの応答 (*DataLength*=0) を戻します。

AgentBufferLength (MQLONG) - 入力

エージェント・バッファの長さ。

このパラメーターは、呼び出しの *DataLength* よりも大きくすることができます。

チャンネル・メッセージ、送信出口および受信出口では、データを拡張したいときは、呼び出しで使用されないスペースを使用できます。これを行う場合は、出口が **DataLength** パラメーターを適切に設定しなければなりません。

C プログラミング言語では、このパラメーターはアドレスによって渡されます。

AgentBuffer (MQBYTE x AgentBufferLength) - 入出力

エージェント・バッファ。

このパラメーターの内容は出口のタイプによって異なります。

- チャンネル・セキュリティー出口の場合、*ExitReason* が MQXR_SEC_MSG になっていると、出口の呼び出し時にセキュリティー・メッセージがこの中に入ります。出口は、セキュリティー・メッセージを送り返すために、このバッファを使用することも、独自のバッファ (*ExitBufferAddr*) を使用することもできます。
- チャンネル・メッセージ出口の場合、出口の呼び出し時に、このパラメーターに以下が入ります。
 - メッセージ記述子 (この中にはメッセージの内容情報が入ります) が入っている伝送キュー・ヘッダー (MQXQH)、さらにその直後に
 - メッセージ・データ

メッセージが続行する場合、出口は次のいずれかを行うことができます。

- バッファの内容をそのままにしておく
- 内容をその場所を変更する (データの新しい長さを *DataLength* に戻します。この値は *AgentBufferLength* を超えてはなりません)
- 必要な変更を行って、内容を *ExitBufferAddr* にコピーする

伝送キュー・ヘッダーに対して出口が行った変更は、検査されません。誤った変更を行うと、メッセージを宛先に入れることができなくなる可能性があります。

- チャンネル送信または受信出口の場合、出口が呼び出されると、伝送データが入ります。出口は次のいずれかを行うことができます。
 - バッファの内容をそのままにしておく
 - 内容をその場所を変更する (データの新しい長さを *DataLength* に戻します。この値は *AgentBufferLength* を超えてはなりません)
 - 必要な変更を行って、内容を *ExitBufferAddr* にコピーする

データの最初の 8 バイトを出口で変更しないでください。

ExitBufferLength (MQLONG) - 入出力

出口バッファの長さ。

出口が初めて呼び出されたときには、このパラメーターはゼロに設定されています。そのあとの各呼び出しで出口から渡された値は、次の呼び出し時に出口に提示されます。この値は MCA では使用されません。

注: このパラメーターは、ポインター・データ・タイプをサポートしないプログラミング言語で書かれた出口では使用しないでください。

ExitBufferAddr (MQPTR) - 入出力

出口バッファのアドレス。

このパラメーターは、出口によって管理されるストレージのバッファのアドレスへのポインターです。エージェントのバッファの大きさが不十分であるか、あるいはその可能性がある場合、またはこのバッファを使用したほうが出口にとって便利な場合には、(出口のタイプに応じて) メッセージ・データまたは伝送データをこのエージェントに戻すようにすることができます。

この出口を初めて呼び出したときには、出口に渡されるアドレスはヌルになっています。そのあとの各呼び出しで出口から渡されたアドレスは、次の呼び出し時に出口に提示されます。

ExitBufferAddr がヌルの場合、使用されるデータは AgentBuffer パラメーターから取得されます。

ExitBufferAddr がヌル以外の場合、使用されるデータは ExitBufferAddr パラメーターが示すバッファーから取得されます。

注: このパラメーターは、ポインター・データ・タイプをサポートしないプログラミング言語で書かれた出口では使用しないでください。

C 言語での呼び出し

```
exitname (&ChannelExitParms, &ChannelDefinition,  
&DataLength, &AgentBufferLength, AgentBuffer,  
&ExitBufferLength, &ExitBufferAddr);
```

出口に渡されるパラメーターは、次のように宣言されます。

```
MQCXP  ChannelExitParms; /* Channel exit parameter block */  
MQCD   ChannelDefinition; /* Channel definition */  
MQLONG DataLength; /* Length of data */  
MQLONG AgentBufferLength; /* Length of agent buffer */  
MQBYTE AgentBuffer[n]; /* Agent buffer */  
MQLONG ExitBufferLength; /* Length of exit buffer */  
MQPTR  ExitBufferAddr; /* Address of exit buffer */
```

COBOL での呼び出し

```
CALL 'exitname' USING CHANNELEXITPARMS, CHANNELDEFINITION,  
                      DATALENGTH, AGENTBUFFERLENGTH, AGENTBUFFER,  
                      EXITBUFFERLENGTH, EXITBUFFERADDR.
```

出口に渡されるパラメーターは、次のように宣言されます。

```
** Channel exit parameter block  
01 CHANNELEXITPARMS.  
   COPY CMQCXPV.  
** Channel definition  
01 CHANNELDEFINITION.  
   COPY CMQCDV.  
** Length of data  
01 DATALENGTH      PIC S9(9) BINARY.  
** Length of agent buffer  
01 AGENTBUFFERLENGTH PIC S9(9) BINARY.  
** Agent buffer  
01 AGENTBUFFER       PIC X(n).  
** Length of exit buffer  
01 EXITBUFFERLENGTH PIC S9(9) BINARY.  
** Address of exit buffer  
01 EXITBUFFERADDR    POINTER.
```

RPG での呼び出し (ILE)

```
C*.1.....2.....3.....4.....5.....6.....7..  
C          CALLP      exitname(MQCXP : MQCD : DATLEN :  
C                               ABUFL : ABUF : EBUFL :  
C                               EBUF)
```

呼び出しのプロトタイプ定義は次のようになります。

```
D*.1.....2.....3.....4.....5.....6.....7..  
Dexitname      PR          EXTPROC('exitname')  
D* Channel exit parameter block  
D MQCXP                160A  
D* Channel definition  
D MQCD                 1328A  
D* Length of data  
D DATLEN              10I 0  
D* Length of agent buffer  
D ABUFL               10I 0  
D* Agent buffer  
D ABUF                *   VALUE
```



```

D* Length of exit buffer
D EBUFL                      10I 0
D* Address of exit buffer
D EBUF                        *
```

System/390 アセンブラー呼び出し

```

CALL EXITNAME, (CHANNELEXITPARMS, CHANNELDEFINITION, DATALENGTH, X
                AGENTBUFFERLENGTH, AGENTBUFFER, EXITBUFFERLENGTH, X
                EXITBUFFERADDR)
```

出口に渡されるパラメーターは、次のように宣言されます。

CHANNELEXITPARMS	CMQCPA	,	Channel exit parameter block
CHANNELDEFINITION	CMQCDA	,	Channel definition
DATALENGTH	DS	F	Length of data
AGENTBUFFERLENGTH	DS	F	Length of agent buffer
AGENTBUFFER	DS	CL(n)	Agent buffer
EXITBUFFERLENGTH	DS	F	Length of exit buffer
EXITBUFFERADDR	DS	F	Address of exit buffer

使用上の注意

1. チャンネル出口によって実行される関数は、出口の提供者によって定義されます。ただしこの出口は、ここで定義された規則、および関連の制御ブロック MQCXP で定義された規則に従わなければなりません。
2. チャンネル出口に渡される **ChannelDefinition** パラメーターは、いくつかのバージョンのうちの1つになることがあります。詳しくは、MQCD 構造体の *Version* フィールドを参照してください。
3. チャンネル出口は、*Version* フィールドに MQCD_VERSION_1 より大きい値に設定された MQCD 構造体を受信した場合は、*ShortConnectionName* フィールドより、MQCD 内の *ConnectionName* フィールドを優先して使用する必要があります。
4. 一般的に、チャンネル出口は、メッセージ・データの長さを変更できます。この変更は、出口によるメッセージへのデータの追加、メッセージからのデータの削除、メッセージの圧縮または暗号化のどれかを行った場合に生じます。ただし、メッセージが論理メッセージの一部だけを取り入れたセグメントである場合、特別な制限が適用されます。特に、相補的な送信出口および受信出口のアクションの結果、メッセージの長さに最終的な変更があってはなりません。

例えば、送信出口が圧縮によってメッセージを短くすることはできます。ただし、対応する受信出口は、圧縮解除によってもとのメッセージの長さを復元する必要があります。そうすれば、メッセージの長さは最終的には変わりません。

この制約事項が必要である理由は、セグメント長の変更によってメッセージ内のその後のセグメントのオフセットが不正確になり、完全な論理メッセージを構成するセグメントを識別するというキュー・マネージャーの機能が使用禁止になる可能性があるからです。

MQ_CHANNEL_AUTO_DEF_EXIT - チャンネル自動定義出口

MQ_CHANNEL_AUTO_DEF_EXIT 呼び出しは、メッセージ・チャンネル・エージェントによって呼び出されたチャンネル自動定義出口に渡すパラメーターを記述します。

MQ_CHANNEL_AUTO_DEF_EXIT という名前の入り口点が、キュー・マネージャーによって提供されるわけではありません。自動定義出口の名前がキュー・マネージャーによって提供されるため、MQ_CHANNEL_AUTO_DEF_EXIT という名前には特別の意味はありません。

構文

```
MQ_CHANNEL_AUTO_DEF_EXIT '(ChannelExitParms, ChannelDefinition)
```

パラメーター

MQ_CHANNEL_AUTO_DEF_EXIT 呼び出しには次のような2つのパラメーターがあります。

ChannelExitParms (MQCXP) - 入出力

チャンネル出口パラメーター・ブロック。

この構造体には、出口の呼び出しに関連する追加情報が入っています。出口は、MCA が処理を続行する方法を示す情報を、この構造体の中に設定します。

ChannelDefinition (MQCD) - 入出力

チャンネル定義。

この構造体には、自動的に作成されるチャンネルの動作を制御するためのパラメーターが、管理者によって組み込まれます。出口は、管理者によって設定されたデフォルトの動作を変更するための情報を、この構造体の中に設定します。

以下にリストされている MQCD フィールドが出口によって変更されないようにする必要があります。

- *ChannelName*
- *ChannelType*
- *StrucLength*
- *Version*

他のフィールドが変更されると、出口で設定された値は必ず有効値になります。有効値にならない場合は、使用している環境に応じてエラー・メッセージがエラー・ログ・ファイルに書き込まれるか、またはコンソールに表示されます。



重要: チャンネル自動定義 (CHAD) 出口によって作成される自動定義されたチャンネルは、証明書ラベルを設定できません。これは、チャンネルが作成される時点までに TLS ハンドシェークが既に発生しているためです。インバウンド・チャンネル用に CHAD 出口で証明書ラベルを設定しても効果がありません。

C 言語での呼び出し

```
exitname (&ChannelExitParms, &ChannelDefinition);
```

出口に渡されるパラメーターは、次のように宣言されます。

```
MQCXP  ChannelExitParms; /* Channel exit parameter block */
MQCD   ChannelDefinition; /* Channel definition */
```

COBOL での呼び出し

```
CALL 'exitname' USING CHANNELEXITPARMS, CHANNELDEFINITION.
```

出口に渡されるパラメーターは、次のように宣言されます。

```
** Channel exit parameter block
01 CHANNELEXITPARMS.
   COPY CMQCXPV.
** Channel definition
01 CHANNELDEFINITION.
   COPY CMQCDV.
```

RPG での呼び出し (ILE)

```
C*.1.....2.....3.....4.....5.....6.....7..
C                                CALLP      exitname(MQCXP : MQCD)
```

呼び出しのプロトタイプ定義は次のようになります。

```
D*.1.....2.....3.....4.....5.....6.....7..
Dexitname          PR              EXTPROC('exitname')
D* Channel exit parameter block
D MQCXP                                160A
```

System/390 アセンブラー呼び出し

```
CALL EXITNAME,(CHANNELEXITPARMS,CHANNELDEFINITION)
```

出口に渡されるパラメーターは、次のように宣言されます。

```
CHANNELEXITPARMS  CMQXCPA  , Channel exit parameter block  
CHANNELDEFINITION CMQCDA   , Channel definition
```

使用上の注意

1. チャネル出口によって実行される関数は、出口の提供者によって定義されます。ただしこの出口は、ここで定義された規則、および関連の制御ブロック MQCXP で定義された規則に従わなければなりません。
2. チャネル自動定義出口に渡される **ChannelExitParms** パラメーターは、MQCXP 構造体です。渡される MQCXP のバージョンは、出口が実行される環境により異なります。詳細については、[1522 ページの『MQCXP - チャネル出口パラメーター』](#)の *Version* フィールドの説明を参照してください。
3. チャネル自動定義出口に渡される **ChannelDefinition** パラメーターは、MQCD 構造体です。渡される MQCD のバージョンは、出口が実行される環境により異なります。詳細については、[1480 ページの『MQCD - チャネル定義』](#)の *Version* フィールドの説明を参照してください。

MQXWAIT - 出口での待機

MQXWAIT 呼び出しは、イベントの発生を待機します。z/OS のチャネル出口からのみ使用されます。

チャネル出口が待機の原因となることを実行すると、パフォーマンスの問題が生じる可能性があります、この問題は MQXWAIT を使用すると回避できます。MQXWAIT が待機しているイベントは、MVS ECB (イベント制御ブロック) によりシグナル通知されます。ECB は、MQXWD 制御ブロック記述で説明されています。

 MQXWAIT の使用およびチャネル出口プログラムの作成の詳細については、[z/OS におけるチャネル出口プログラムの作成](#)を参照してください。

構文

MQXWAIT (Hconn, WaitDesc, CompCode, Reason)

パラメーター

MQXWAIT 呼び出しには、次のパラメーターがあります。

Hconn (MQHCONN) - 入力

接続ハンドル。

このハンドルは、キュー・マネージャーに対する接続を表します。Hconn の値は、出口の同じ呼び出しまたは前の呼び出しで発行された前の MQCONN 呼び出しによって戻されたものです。

WaitDesc (MQXWD) - 入出力

待機記述子。

このパラメーターは、待機対象のイベントを記述します。この構造体のフィールドの詳細については、[1537 ページの『MQXWD - 出口待機記述子』](#)を参照してください。

CompCode (MQLONG) - 出力

完了コード。

これは、以下のコードのいずれかです。

MQCC_OK

正常終了。

MQCC_FAILED

呼び出し失敗。

Reason (MQLONG) - 出力

CompCode を限定する理由コード。

CompCode が MQCC_OK の場合:

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'89C') アダプターが利用できません。

MQRC_OPTIONS_ERROR

(2046, X'7FE') オプションが無効であるか、矛盾しています。

MQRC_XWAIT_CANCELED

(2107, X'83B') MQXWAIT 呼び出しが取り消されました。

MQRC_XWAIT_ERROR

(2108, X'83C') MQXWAIT の呼び出しが無効です。

C 言語での呼び出し

```
MQXWAIT (Hconn, &WaitDesc, &CompCode, &Reason);
```

パラメーターを次のように宣言します。

```
MQHCONN  Hconn;      /* Connection handle */
MQXWD     WaitDesc;   /* Wait descriptor */
MQLONG    CompCode;   /* Completion code */
MQLONG    Reason;     /* Reason code qualifying CompCode */
```

System/390 アセンブラー呼び出し

```
CALL MQXWAIT,(HCONN,WAITDESC,COMPCODE,REASON)
```

パラメーターを次のように宣言します。

HCONN	DS	F	Connection handle
WAITDESC	CMQXWDA	,	Wait descriptor
COMPCODE	DS	F	Completion code
REASON	DS	F	Reason code qualifying COMPCODE

MQCD - チャネル定義

MQCD 構造体には、チャネルの実行を制御するパラメーターが入っています。これは、メッセージ・チャネル・エージェント (MCA) から呼び出される各チャネル出口に渡されます。

チャネル出口の詳細については、[1473 ページ](#)の『MQ CHANNEL EXIT - チャネル出口』を参照してください。このトピック内の説明は、メッセージ・チャネルと MQI チャネルの両方に関連します。

出口名フィールド

出口が呼び出されると、*SecurityExit*、*MsgExit*、*SendExit*、*ReceiveExit*、および *MsgRetryExit* の関係するフィールドには、現在呼び出されている出口の名前が示されます。フィールド内の名前の意味は、MCA が実行される環境によって異なります。注記がない限り、この名前はフィールド内に左寄せで入れられ、組み込みブランクは含みません。フィールドの長さまで、名前にブランクが埋め込まれます。以下の説明では、大括弧 ([]) はオプション情報を表します。

AIX and Linux

出口名は、動的ロード可能モジュールまたはライブラリーの名前に、そのライブラリーに入っている関数の名前が接尾部として付いたものです。関数名は括弧で囲む必要があります。ライブラリー名の前にはオプションでディレクトリー・パスを付けることができます。

```
[ path ] library ( function )
```

この名前は最大 128 文字までに制限されています。

z/OS

出口名は、LINK または LOAD マクロの EP パラメーターで指定するのに有効なロード・モジュール名です。この名前は最大 8 文字までに制限されています。

Windows

出口名は、ダイナミック・リンク・ライブラリーにそのライブラリーに入っている関数の名前が接尾部として付いたものです。関数名は括弧で囲む必要があります。ライブラリー名の前にはオプションでディレクトリー・パスとドライブを付けることができます。

```
[d:][ path ] library ( function )
```

この名前は最大 128 文字までに制限されています。

IBM i

出口名は、10 バイトのプログラム名に 10 バイトのライブラリー名が付いたものです。名前が 10 バイトに満たない場合は、どの名前にもブランクが埋め込まれて、10 バイトにされます。ライブラリー名を *LIBL にすることができます。ただし、チャンネル自動定義出口を呼び出す場合を除きます。その場合、完全修飾名が必要となります。

チャンネル出口での MQCD フィールドの変更

チャンネル出口は、MQCD のフィールドを変更できます。変更された値は MQCD に残り、出口チェーン内の残りの出口、およびチャンネル・インスタンスを共用する会話に渡されます。変更された MQCD は MCA によって、チャンネルの存続期間が続いている間の通常の処理でも使用されます。

以下の MQCD フィールドは、出口によって変更されてはなりません。

- ChannelName
- ChannelType
- StrucLength
- バージョン

関連資料

[1482 ページの『フィールド』](#)

このトピックでは、MQCD 構造体のすべてのフィールドをリストし、それぞれのフィールドについて説明しています。

[1509 ページの『C 宣言』](#)

以下の宣言は、MQCD 構造体の C 宣言です。

[1511 ページの『COBOL 宣言』](#)

以下の宣言は、MQCD 構造体の COBOL 宣言です。

[1514 ページの『RPG 宣言 \(ILE\)』](#)

以下の宣言は、MQCD 構造体の RPG 宣言です。

[1516 ページの『System/390 アセンブラ宣言』](#)

以下の宣言は、MQCD 構造体の System/390 アセンブラ宣言です。

[1518 ページの『Visual Basic の宣言』](#)

以下の宣言は、MQCD 構造体の Visual Basic 宣言です。

[1519 ページの『チャンネル出口での MQCD フィールドの変更』](#)

チャンネル出口は、MQCD のフィールドを変更できます。ただし、リストされている状況を除いて、通常はこれらの変更に応じて動作することはありません。

フィールド

このトピックでは、MQCD 構造体のすべてのフィールドをリストし、それぞれのフィールドについて説明しています。

BatchDataLimit (MQLONG)

このフィールドは、同期点を取る前にチャンネルを介して送信可能なデータ量の限度(キロバイト単位)を指定します。

限度に達した際のメッセージがチャンネルを通過して送信された後に、同期点が取られます。

バッチは、次の条件のいずれかが満たされた場合に終了します。

- **BatchSize** メッセージが送信されました。
- **BatchDataLimit** バイトが送信されました。
- 伝送キューが空で、**BatchInterval** が経過した。

値は 0 から 999999 の範囲でなければなりません。デフォルト値は 5000 です。

この属性の値がゼロの場合、それはこのチャンネルに対するバッチに適用されるデータ限度がないことを意味します。

このパラメーターは、*ChannelType* が MQCHT_SENDER、MQCHT_SERVER、MQCHT_CLUSRCVR、または MQCHT_CLUSSDR であるチャンネルにのみ適用されます。

これは、出口に対する入力フィールドです。*Version* が MQCD_VERSION_11 より小さい場合は、このフィールドは提供されません。

BatchHeartbeat (MQLONG)

このフィールドは、チャンネルのバッチ・ハートビートをトリガーする時間間隔を指定します。

バッチ・ハートビートを指定すると、送信側チャンネルは、リモート・チャンネル・インスタンスが未確定になる前にアクティブかどうかを判別できるようになります。バッチ・ハートビートは、送信側チャンネルが、指定した時間間隔内にリモート・チャンネル・インスタンスと通信しない場合に発生します。

値の範囲は 0 から 999 999 です。単位はミリ秒です。ゼロの値は、バッチ・ハートビートが使用可能ではないことを示します。

このフィールドは、*ChannelType* が MQCHT_SENDER、MQCHT_SERVER、MQCHT_CLUSSDR、または MQCHT_CLUSRCVR のチャンネルだけに適用されます。

これは、出口に対する入力フィールドです。*Version* が MQCD_VERSION_7 より小さい場合は、このフィールドは提供されません。

BatchInterval (MQLONG)

このフィールドは、現行のバッチで伝送されたメッセージ数が *BatchSize* より少ない場合に、チャンネルがバッチをオープン状態にしておく概算時間(ミリ秒単位)を指定します。

BatchInterval が 0 より大きい場合は、次のイベントのどちらかが先に発生するとバッチが終了します。

- *BatchSize* で指定した数のメッセージが送信された。
- バッチが開始されてから、*BatchInterval* で指定した時間(ミリ秒)が経過した。

BatchInterval が 0 の場合は、次のイベントのどちらかが先に発生するとバッチが終了します。

- *BatchSize* で指定した数のメッセージが送信された。
- 伝送キューが空になった

BatchInterval は 0 から 999 999 999 の範囲でなければなりません。

このフィールドは、*ChannelType* として MQCHT_SENDER、MQCHT_SERVER、MQCHT_CLUSSDR、または MQCHT_CLUSRCVR が指定されているチャンネルのみに関連します。

これは、出口に対する入力フィールドです。 *Version* が MQCD_VERSION_4 より小さい場合は、このフィールドは提供されません。

BatchSize (MQLONG)

このフィールドは、チャンネルを同期するまでにチャンネル経由で送信可能なメッセージの最大数を指定します。

このフィールドは、*ChannelType* として MQCHT_SVRCONN または MQCHT_CLNTCONN が指定されているチャンネルとは関係ありません。

CertificateLabel (MQCHAR64)

このフィールドには、使用されている証明書ラベルの詳細が表示されます。

 複数の証明書ラベルを、 スペースまたは  コンマで区切って設定できます。複数のラベルが指定された場合、IBM MQ は各ラベルを試行し、選択された証明書が TLS ハンドシェイク中に有効な CipherSuite の選択につながるかどうかを判断します。これは、TLS、1.2、RSA、または ECC (CipherSuites) のいずれかが使用可能な場面での利用を想定しています。

 複数のラベルを指定する場合は、各ラベルの間にコンマまたはスペースを **1** つ入れてください。不要な区切り文字があるとエラーが発生します。

 一般的に、証明書のラベル内にコンマやスペースを含めることは推奨されません。既存の証明書ラベルにコンマやスペースが含まれている場合は、バックスペースを付けてエスケープしてください。val1\, \ val2 たとえば、キーストア内の証明書ラベルが val, val2 (コンマとスペース) の場合、その **CERTLABL** 値となります。エスケープ文字を使用すると、キューマネージャーに対して、そのコンマやスペースがラベルの一部であることを伝えます。 IBM i では、スペース文字は区切り文字としてサポートされていないため、エスケープ処理を行わないでください。

IBM MQ は、*CertificateLabel* フィールドのデフォルト値をブランクとして初期化します。

これは実行時にデフォルト値として解釈され、後方互換です。

例えば 11 未満の MQCD バージョンを指定するか、ブランクのデフォルト値を *CertificateLabel* フィールドに使用した場合、このフィールドは無視されます。

このフィールドの長さは MQ_CERT_LABEL_LENGTH によって指定されます。

ChannelMonitoring (MQLONG)

このフィールドは、チャンネルのモニター・データ収集の現行レベルを指定します。

このフィールドは、*ChannelType* が MQCHT_CLNTCONN のチャンネルには適用されません。

これは、次の値のいずれかです。

- MQMON_OFF
- MQMON_LOW
- MQMON_MEDIUM
- MQMON_HIGH

これは、出口に対する入力フィールドです。 *Version* が MQCD_VERSION_8 より小さい場合は、提供されません。

ChannelName (MQCHAR20)

このフィールドは、チャンネル定義名を指定します。

通信を行うためには、リモート・マシンに同じ名前のチャンネル定義がなければなりません。

この名前には、次の文字だけを使用します。

- 大文字の A から Z
- 小文字の a から z
- 数値の 0 から 9

- ピリオド (.)
- スラッシュ (/)
- 下線 (_)
- パーセント記号 (%)

また、右側にブランクを埋め込む必要があります。先行ブランクや組み込みブランクは使用できません。
このフィールドの長さは MQ_CHANNEL_NAME_LENGTH によって指定されます。

ChannelStatistics (MQLONG)

このフィールドは、チャンネルの統計データ収集の現行レベルを指定します。

このフィールドは、ChannelType が MQCHT_CLNTCONN または MQCHT_SVRCONN であるチャンネルには関係ありません。

これは、次の値のいずれかです。

- MQMON_OFF
- MQMON_LOW
- MQMON_MEDIUM
- MQMON_HIGH

これは、出口に対する入力フィールドです。Version が MQCD_VERSION_8 より小さい場合は、提供されません。

ChannelType (MQLONG)

このフィールドは、チャンネルのタイプを指定します。

これは、次の値のいずれかです。

MQCHT_SENDER

送信側。

MQCHT_SERVER

サーバー。

MQCHT_RECEIVER

受信側。

MQCHT_REQUESTER

要求側。

MQCHT_CLNTCONN

クライアント接続。

MQCHT_SVRCONN

サーバー接続 (クライアントが使用)。

MQCHT_CLUSSDR

クラスター送信側。

MQCHT_CLUSRCVR

クラスター受信側。

ClientChannelWeight (MQLONG)

このフィールドは、どのクライアント接続チャンネル定義を使用するかに影響を与える加重を指定します。

ClientChannelWeight 属性を使用すると、複数の適切な定義が選択可能な場合に、クライアント・チャンネル定義を加重に基づいてランダムに選択できます。先頭がアスタリスクのキュー・マネージャー名を指定して、クライアントが MQCONN 要求接続をキュー・マネージャー・グループに対して発行し、複数の適切なチャンネル定義がクライアント・チャンネル定義テーブル (CCDT) で選択可能な場合、使用する定義は加重に基づいてランダムに選択されます。適用可能な任意の ClientChannelWeight(0) の定義が、アルファベット順に従って最初に選択されます。

0 から 99 の範囲の値を指定します。デフォルトは 0 です。

値として 0 を指定すると、ロード・バランシングが実行されず、該当する定義がアルファベット順で選択されます。ロード・バランシングを有効にするには、1 から 99 までの範囲の値を選択します (1 が最低の加重値、99 が最高の加重値です)。非ゼロの加重を持つ 2 つ以上のチャンネル間でのメッセージの分散は、それらの加重の比率に比例したものになります。例えば、ClientChannelWeight 値として 2、4、および 14 を持つ 3 つのチャンネルは、ほぼ 10%、20%、および 70% の時間の割合で選択されます。この分散は保証されているわけではありません。

この属性は、クライアント接続チャンネル・タイプでのみ有効です。

これは、出口に対する入力フィールドです。Version が MQCD_VERSION_9 より小さい場合は、このフィールドは提供されません。

ClusterPtr (MQPTR)

このフィールドは、クラスター名のリストのアドレスを指定します。

ClustersDefined がゼロより大きい場合、このアドレスはクラスター名のリストのアドレスです。チャンネルは、リストされた各クラスターに属します。

このフィールドは、ChannelType として MQCHT_CLUSSDR または MQCHT_CLUSRCVR が指定されているチャンネルのみに関連します。

これは、出口に対する入力フィールドです。Version が MQCD_VERSION_5 より小さい場合は、このフィールドは提供されません。

ClustersDefined (MQLONG)

このフィールドは、チャンネルが所属するクラスターの数指定します。

このフィールドは ClusterPtr が指すクラスター名の数です。ゼロまたはゼロより大きい数です。

このフィールドは、ChannelType として MQCHT_CLUSSDR または MQCHT_CLUSRCVR が指定されているチャンネルのみに関連します。

これは、出口に対する入力フィールドです。Version が MQCD_VERSION_5 より小さい場合は、このフィールドは提供されません。

CLWLChannelPriority (MQLONG)

このフィールドは、クラスター・ワークロード・チャンネル優先順位を指定します。

ワークロード・マネージャー選択アルゴリズムは、ランクに基づいて選択された一連の宛先から、最も高い優先順位が設定されている宛先を選択します。選択可能な宛先キュー・マネージャーが 2 つある場合、この属性を使用して一方のキュー・マネージャーを他方のキュー・マネージャーにフェイルオーバーさせることができます。すべてのメッセージは、最高位の優先順位が設定されたキュー・マネージャーに送信されます。そのキュー・マネージャーが終了すると、今度は次に高い優先順位が設定されているキュー・マネージャーにメッセージが送信されます。

値は、0 から 9 までの範囲です。デフォルトは 0 です。

これは、出口に対する入力フィールドです。Version が MQCD_VERSION_8 より小さい場合は、このフィールドは提供されません。

詳細については、[キュー・マネージャー・クラスターの構成](#)を参照してください。

CLWLChannelRank (MQLONG)

このフィールドは、クラスター・ワークロード・チャンネル・ランクを指定します。

ワークロード・マネージャー選択アルゴリズムは、最高ランクの宛先を選択します。最終宛先が別のクラスターのキュー・マネージャーである場合、選択アルゴリズムが最終宛先に最も近い宛先キュー・マネージャーを正しく選択するように、(隣接するクラスターの交点にある) 中間ゲートウェイ・キュー・マネージャーのランクを設定することができます。

値は、0 から 9 までの範囲です。デフォルトは 0 です。

これは、出口に対する入力フィールドです。Version が MQCD_VERSION_8 より小さい場合は、このフィールドは提供されません。

詳細については、[キュー・マネージャー・クラスターの構成](#)を参照してください。

CLWLChannelWeight (MQLONG)

このフィールドは、クラスター・ワークロード・チャンネル・ウェイトを指定します。

クラスター・ワークロード・チャンネル・ウェイト

ワークロード・マネージャー選択アルゴリズムにより、チャンネルの「ウェイト」属性を使用して宛先選択をスキューして、特定のマシンに多くのメッセージが送信されるようにできます。例えば、大規模 UNIX サーバー上のチャンネルに他の小規模デスクトップ PC 上のチャンネルより大きなウェイトを与えると、選択アルゴリズムにより、PC よりも UNIX サーバーが頻繁に選択されるようになります。

値の範囲は 1 から 99 です。デフォルトは 50 です。

これは、出口に対する入力フィールドです。Version が MQCD_VERSION_8 より小さい場合は、このフィールドは提供されません。

詳細については、[キュー・マネージャー・クラスターの構成](#)を参照してください。

ConnectionAffinity (MQLONG)

このフィールドは、同じキュー・マネージャー名を使用して複数回接続するクライアント・アプリケーションが、同じクライアント・チャンネルを使用するかどうかを指定します。

この属性は、該当するチャンネル定義が複数存在する場合に使用します。

値は、次のいずれか 1 つです。

MQCAFTY_PREFERRED

クライアント・チャンネル定義テーブル (CCDT) を読み取るプロセス内の最初の接続は、加重に基づいて適用可能な定義のリストを作成します。これは先頭が適用可能な CLNTWGHT(0) 定義で、アルファベット順です。プロセス内の各接続は、リスト内の最初の定義を使用して接続を試行します。接続が失敗した場合は、次の定義が使用されます。0 以外の CLNTWGHT 値を持つ失敗した定義は、リストの末尾に移動します。CLNTWGHT(0) 定義は、リストの先頭に残り、各接続の最初に選択されます。

同じホスト名を持つ各クライアント・プロセスは、常に同じリストを作成します。

C、C++、または .NET プログラミング・フレームワーク (完全に管理された .NET を含む) で作成されたクライアント・アプリケーションでは、リストの作成以降に CCDT が変更された場合、リストは更新されます。

この値がデフォルト値です。

MQCAFTY_NONE

CCDT を読み取るプロセス内の最初の接続が、適用可能な定義のリストを作成します。プロセス内のすべての接続は、加重に基づいて適用可能な定義を選択します。適用可能な CLNTWGHT(0) の定義を最初にアルファベット順に選択していきます。

C、C++、または .NET プログラミング・フレームワーク (完全に管理された .NET を含む) で作成されたクライアント・アプリケーションでは、リストの作成以降に CCDT が変更された場合、リストは更新されます。

この属性は、クライアント接続チャンネル・タイプでのみ有効です。

これは、出口に対する入力フィールドです。Version が MQCD_VERSION_9 より小さい場合は、このフィールドは提供されません。

ConnectionName (MQCHAR264)

このフィールドは、チャンネルの接続名を指定します。

クラスター受信側チャンネル (指定されているとき) の場合は、CONNAME はローカル・キュー・マネージャーに関連し、その他のチャンネルの場合は、CONNAME は宛先キュー・マネージャーに関連します。指定する値は、使用される伝送プロトコル (TransportType) によって異なります。

- MQXPT_LU62 の場合、これは、パートナー論理装置の完全修飾名です。
- MQXPT_NETBIOS の場合、これはリモート・マシン上で定義された NetBIOS 名です。
- MQXPT_TCP の場合、これはホスト名、IPv4 ドット 10 進または IPv6 16 進形式で指定されたリモート・マシン、またはクラスター受信側チャンネルのローカル・マシンのネットワーク・アドレスです。

- MQXPT_SPX の場合、これは 4 バイトのネットワーク・アドレス、6 バイトのノード・アドレス、および 2 バイトのソケット番号から構成される SPX スタイルのアドレスです。

チャンネルの定義時は、このフィールドは、*ChannelType* が MQCHT_SVRCONN または MQCHT_RECEIVER のチャンネルには適用されません。しかし、チャンネル定義が出口に渡されるときには、どのチャンネル・タイプの場合にも、このフィールドにはパートナーのアドレスが入ります。

このフィールドの長さは MQ_CONN_NAME_LENGTH によって指定されます。*Version* が MQCD_VERSION_2 より小さい場合は、このフィールドは提供されません。

DataConversion (MQLONG)

このフィールドは、受信側のメッセージ・チャンネル・エージェントがアプリケーション・メッセージ・データを変換できない場合、送信側のメッセージ・チャンネル・エージェントがその変換を行うかどうかを指定します。

このフィールドは、論理メッセージのセグメントではないメッセージにだけ適用されます。MCA は、セグメントであるメッセージを変換しようとはしません。

このフィールドは、*ChannelType* として MQCHT_SENDER、MQCHT_SERVER、MQCHT_CLUSSDR、または MQCHT_CLUSRCVR が指定されているチャンネルのみに関連します。これは、以下のいずれかになります。

MQCDC_SENDER_CONVERSION

送信側による変換。

MQCDC_NO_SENDER_CONVERSION

送信側による変換なし。

DefReconnect (MQLONG)

DefReconnect チャンネル属性は、クライアント接続チャンネルのデフォルト再接続属性値を設定します。

デフォルトの自動クライアント再接続オプション。自動的にクライアント・アプリケーションを再接続するように IBM MQ MQI client を構成できます。IBM MQ MQI client は、接続に失敗した後、キュー・マネージャーへの再接続を試みます。この再接続試行は、アプリケーション・クライアントが MQCONN または MQCONNX MQI 呼び出しを発行しなくても行われます。

再接続は MQCONNX のオプションです。DefReconnect チャンネル属性を使用することで、MQCONN を使用する既存のアプリケーションに再接続の動作を追加できます。また、MQCONNX を使用するアプリケーションの再接続の動作を変更することもできます。

mqclient.ini ファイルの DefRecon 値を設定して、再接続の動作を設定または変更することもできます。mqclient.ini ファイルの DefRecon 値は、DefReconnect チャンネル属性に優先されます。

Syntax

DefReconnect (MQRCN_NO (default) |MQRCN_YES|MQRCN_Q_MGR|
MQRCN_DISABLED)

パラメーター

MQRCN_NO

MQRCN_NO はデフォルト値です。

MQCONNX によって上書きされない限り、クライアントは自動的に再接続されません。

MQRCN_YES

MQCONNX によって上書きされない限り、クライアントは自動的に再接続します。

MQRCN_Q_MGR

MQCONNX によって上書きされない限り、クライアントは自動的に再接続しますが、同じキューマネージャーへのみ再接続します。MQCNO_RECONNECT_Q_MGRQMGR オプションは、. と同じ効果があります。

MQRCN_DISABLED

クライアントプログラムが MQI 呼び出し **MQCONN** を使用して再接続を要求した場合でも、再接続は無効になります。

IBM MQ classes for Java は自動クライアント再接続をサポートしていません。

表 858. アプリケーションおよびチャネル定義に設定された値によって異なる自動再接続

DefReconnect	アプリケーションで設定される再接続オプション			
	MQCNO_RECONNE CT	MQCNO_RECONNE CT_Q_MGR	MQCNO_RECONNE CT_AS_DEF	MQCNO_RECONNE CT_DISABLED
MQRCN_NO	YES	QMGR	NO	NO
MQRCN_YES	YES	QMGR	YES	NO
MQRCN_Q_MGR	YES	QMGR	QMGR	NO
MQRCN_DISABLED	NO	NO	NO	NO

関連概念

[クライアントの自動再接続](#)

[チャネルおよびクライアントの再接続](#)

[クライアント構成ファイルの CHANNELS スタンザ](#)

関連情報

272 ページの『Options (MQLONG)』

Desc (MQCHAR64)

このフィールドは、説明コメントのために使用できます。

このフィールドの内容は、メッセージ・チャネル・エージェントにとっては特に意味を持ちません。ただし、ここには表示可能な文字だけが含まれている必要があります。フィールドにヌル文字を入れることはできません。また、必要に応じて、右側がブランクで埋められます。DBCS をインストール済みの環境では、このフィールドに DBCS 文字を入れることができます (最大フィールド長として 64 バイトが適用されます)。

注: このフィールドにキュー・マネージャーの文字セット (**CodedCharSetId** キュー・マネージャー属性によって定義されている) にない文字が入っていると、フィールドが別のキュー・マネージャーに送信される場合に、それらの文字が間違って変換されることがあります。

このフィールドの長さは MQ_CHANNEL_DESC_LENGTH によって指定されます。

DiscInterval (MQLONG)

このフィールドは、チャネルが終了する前に、伝送キューにメッセージが着信するまでチャネルが待機する秒単位の最長時間を指定します。

言い換えると、このフィールドは切断の間隔を指定します。

値 0 を指定すると、MCA は無期限に待機します。

TCP プロトコルを使用するサーバー接続チャネルでは、この間隔はクライアントの非アクティブ切断の値 (秒単位) を表します。サーバー接続はパートナーのクライアントから通信を受けない状態がこの長さの時間に達すると、接続を終了します。サーバー接続の非アクティビティー間隔は、クライアントからの IBM MQ API 呼び出し間にも適用されるため、待機呼び出しを伴う長時間実行している MQGET 中にクライアントが切断されることはありません。

この属性は、TCP 以外のプロトコルを使用するサーバー接続チャネルには適用されません。

このフィールドは、*ChannelType* として MQCHT_SENDER、MQCHT_SERVER、MQCHT_CLUSSDR、MQCHT_CLUSRCVR、または MQCHT_SVRCONN が指定されているチャネルのみに関連します。

ExitDataLength (MQLONG)

このフィールドは、*MsgUserDataPtr*、*SendUserDataPtr*、および *ReceiveUserDataPtr* フィールドに指定された出口ユーザー・データのリスト内の各ユーザー・データ項目の長さ (バイト数) を指定します。

この長さは、MQ_EXIT_DATA_LENGTH と同じでなくてもかまいません。

これは、出口に対する入力フィールドです。Version が MQCD_VERSION_4 より小さい場合は、このフィールドは提供されません。

ExitNameLength (MQLONG)

このフィールドは、*MsgExitPtr*、*SendExitPtr*、および *ReceiveExitPtr* フィールドに指定された出口名のリスト内の各出口名の長さ (バイト数) を指定します。

この長さは、MQ_EXIT_NAME_LENGTH と同じでなくてもかまいません。

これは、出口に対する入力フィールドです。Version が MQCD_VERSION_4 より小さい場合は、このフィールドは提供されません。

HdrCompList [2] (MQLONG)

このフィールドは、チャンネルがサポートするヘッダー・データ圧縮技法のリストを指定します。

リストには、以下の 1 つ以上の値が含まれます。

MQCOMPRESS_NONE

ヘッダー・データ圧縮は実行されません。

MQCOMPRESS_SYSTEM

ヘッダー・データ圧縮が実行されます。

MQCOMPRESS_NOT_AVAILABLE

リスト内の未使用の値は、この値に設定されます。

これは、出口に対する入力フィールドです。Version が MQCD_VERSION_8 より小さい場合は、このフィールドは提供されません。

HeartbeatInterval (MQLONG)

このフィールドは、ハートビート・フローの間隔 (秒数) を指定します。

このフィールドの解釈は、次のようにチャンネル・タイプによって異なります。

- チャンネル・タイプが MQCHT_SENDER、MQCHT_SERVER、MQCHT_RECEIVER、MQCHT_REQUESTER、MQCHT_CLUSSDR、または MQCHT_CLUSRCVR である場合、このフィールドが指定するのは、伝送キューにメッセージが存在しないときに送信側 MCA から渡される ハートビート・フローの時間間隔 (秒数) です。この間隔中に、受信側 MCA ではチャンネルを静止させることができます。役に立つようにするには、*HeartbeatInterval* を *DiscInterval* より小さくする必要があります。
- チャンネル・タイプが MQCHT_CLNTCONN または MQCHT_SVRCONN で、MQCD 共有会話フィールドがゼロに設定されている場合、このフィールドが指定するのは、サーバー MCA がクライアント・アプリケーションのために MQGMO_WAIT オプションを指定した MQGET 呼び出しを発行したときに、そのサーバー MCA から渡されるハートビート・フローの時間間隔 (秒数) です。これは、MQGMO_WAIT を指定した MQGET の実行中にクライアント接続に障害が発生した場合に、その状況をサーバー MCA で処理できるようにするためです。
- チャンネル・タイプが MQCHT_CLNTCONN または MQCHT_SVRCONN で、MQCD 共有会話フィールドがゼロ以外の値に設定されている場合、このフィールドが指定するのは、送受信されるデータ・フローがないときのハートビート・フローの時間間隔 (秒数) です。これによって、チャンネルを効率良く静止させることができます。

値の範囲は 0 から 999 999 です。送信側または受信側のいずれかで値 0 が設定されているために、ハートビートの交換は行われないという場合を除き、送信側と受信側で指定した値よりも大きい値を使用します。

これは、出口に対する入力フィールドです。Version が MQCD_VERSION_4 より小さい場合は、このフィールドは提供されません。

KeepAliveInterval (MQLONG)

このフィールドは、チャンネルのキープアライブ・タイミングのために、通信スタックに渡される値です。

この値は、TCP/IP および SPX 通信プロトコルに適用できるものですが、すべてのインプリメンテーションで、このパラメーターをサポートするわけではありません。

値の範囲は 0 から 99 999 です。単位は秒です。値がゼロの場合、チャンネル・キープアライブが使用可能ではないことを示します。この場合でも、(チャンネル・キープアライブではなく) TCP/IP キープアライブが使用可能であれば、キープアライブを発生させることが可能です。次の特殊値も有効です。

MQKAI_AUTO

自動。

この値は、以下に示すように、折衝されたハートビート間隔からキープアライブ間隔が計算されることを示します。

- 折衝されたハートビート間隔がゼロより大きい場合、使用されるキープアライブ間隔は、ハートビート間隔に 60 秒を足したものです。
- 折衝されたハートビート間隔がゼロである場合、使用されるキープアライブ間隔はゼロです。
- z/OS では、キュー・マネージャー・オブジェクトで TCPKEEP(YES) が指定されると、TCP/IP キープアライブが発生します。
- 他の環境では、分散キューイング構成ファイルの TCP スタンザに **KEEPALIVE=YES** パラメーターが指定されていれば、TCP/IP キープアライブが発生します。

このフィールドは、*TransportType* が MQXPT_TCP または MQXPT_SPX のチャンネルだけに適用されます。

これは、出口に対する入力フィールドです。Version が MQCD_VERSION_7 より小さい場合は、このフィールドは提供されません。

LocalAddress (MQCHAR48)

このフィールドは、アウトバウンド通信チャンネルに定義されたローカル TCP/IP アドレスを指定します。

アウトバウンド通信用に特定のアドレスが定義されていない場合、このフィールドはブランクになります。このアドレスには、任意でポート番号またはポート番号の範囲を含めることができます。このアドレスの書式は次のとおりです。

```
[ip-addr] [(low-port[,high-port])]
```

ここで、大括弧 ([]) はオプション情報で、ip-addr は IPv4 ドット 10 進数、IPv6 16 進数、または英数字形式で指定します。また、low-port と high-port は括弧で囲まれたポート番号です。すべて任意指定です。

アウトバウンド通信用の特定の IP アドレス、ポート、またはポート範囲は、別の TCP/IP スタックでチャンネルを開始するようリカバリー・シナリオで役立ちます。

LocalAddress は *ConnectionName* の形式に類似していますが、混同しないでください。

LocalAddress はローカル通信の特性を指定する一方、*ConnectionName* はリモート・キュー・マネージャーに到達する方法を指定します。

バージョンから IBM MQ 9.3.0、メッセージキューイングインターフェイス Java (JMQUI) が更新され、チャンネルインスタンスが作成されキューマネージャに接続された後、MQCD オブジェクトのローカルアドレスフィールドが確実に設定されるようになりました。これは、Java で作成されたチャンネル出口がメソッド MQCD.getLocalAddress() を呼び出すと、メソッドによってチャンネル・インスタンスが使用しているローカル・アドレスが返されることを意味します。以前は IBM MQ 9.3.0、チャンネルセキュリティ出口がチャンネルインスタンスが使用しているローカルアドレスにアクセスできず、メソッドは null を返 MQCD.getLocalAddress() していました。

このフィールドは、*TransportType* が MQXPT_TCP のチャンネルと、*ChannelType* が MQCHT_SENDER、MQCHT_SERVER、MQCHT_REQUESTER、MQCHT_CLNTCONN、MQCHT_CLUSSDR、または MQCHT_CLUSRCVR のチャンネルだけに関係します。

このフィールドの長さは MQ_LOCAL_ADDRESS_LENGTH によって指定されます。Version が MQCD_VERSION_7 より小さい場合は、このフィールドは提供されません。

LongMCAUserIdLength (MQLONG)

このフィールドは、*LongMCAUserIdPtr* が指す完全な MCA ユーザー ID の長さ (バイト) を指定します。

このフィールドは、*ChannelType* として MQCHT_CLNTCONN が指定されているチャンネルとは関係ありません。

これは、出口に対する入出力フィールドです。 *Version* が MQCD_VERSION_6 より小さい場合は、このフィールドは提供されません。

LongMCAUserIdPtr (MQPTR)

このフィールドは、長い MCA ユーザー ID のアドレスを指定します。

LongMCAUserIdLength がゼロより大きい場合、このアドレスは完全な MCA ユーザー ID のアドレスです。完全な ID の長さは *LongMCAUserIdLength* で指定されます。フィールド *MCAUserIdIdentifier* には、MCA ユーザー ID の最初の 12 バイトも入っています。

MCA ユーザー ID の詳細については、*MCAUserIdIdentifier* フィールドの説明を参照してください。

このフィールドは、*ChannelType* が MQCHT_SDR、MQCHT_SVR、MQCHT_CLNTCONN、または MQCHT_CLUSSDR に指定されているチャンネルには適用されません。

これは、出口に対する入出力フィールドです。 *Version* が MQCD_VERSION_6 より小さい場合は、このフィールドは提供されません。

LongRemoteUserIdLength (MQLONG)

このフィールドは、*LongRemoteUserIdPtr* が指す完全なリモート・ユーザー ID の長さ (バイト) を指定します。

このフィールドは、*ChannelType* として MQCHT_CLNTCONN または MQCHT_SVRCONN が指定されているチャンネルのみに関連します。

これは、出口に対する入力フィールドです。 *Version* が MQCD_VERSION_6 より小さい場合は、このフィールドは提供されません。

LongRemoteUserIdPtr (MQPTR)

このフィールドは、長いリモート・ユーザー ID のアドレスを指定します。

LongRemoteUserIdLength がゼロより大きい場合、このフラグは完全なリモート・ユーザー ID のアドレスです。完全な ID の長さは *LongRemoteUserIdLength* で指定されます。フィールド *RemoteUserIdIdentifier* には、リモート・ユーザー ID の最初の 12 バイトも入っています。

リモート・ユーザー ID の詳細については、*RemoteUserIdIdentifier* フィールドの説明を参照してください。

このフィールドは、*ChannelType* として MQCHT_CLNTCONN または MQCHT_SVRCONN が指定されているチャンネルのみに関連します。

これは、出口に対する入力フィールドです。 *Version* が MQCD_VERSION_6 より小さい場合は、このフィールドは提供されません。

LongRetryCount (MQLONG)

このフィールドは、*ShortRetryCount* で指定された回数を試みた後に再試行する回数を指定します。

これは、*LongRetryInterval* で指定された間隔でリモート・マシンとの接続を再試行する最大回数を指定します。この最大回数に達すると、オペレーターに知らせるためにエラーが記録されます。

このフィールドは、*ChannelType* として MQCHT_SENDER、MQCHT_SERVER、MQCHT_CLUSSDR、または MQCHT_CLUSRCVR が指定されているチャンネルのみに関連します。

LongRetryInterval (MQLONG)

このフィールドは、リモート・マシンへの接続を再度試みるまで、最大何秒間待つかを指定します。

チャンネルがアクティブになるのを待つ必要がある場合、再試行間隔が延長されることがあります。

このフィールドは、*ChannelType* として MQCHT_SENDER、MQCHT_SERVER、MQCHT_CLUSSDR、または MQCHT_CLUSRCVR が指定されているチャンネルのみに関連します。

MaxInstances (MQLONG)

このフィールドは、個別のサーバー接続チャンネルの、開始可能な同時インスタンスの最大数を指定します。

このフィールドは、サーバー接続チャンネルでのみ使用されます。

このフィールドには、0 から 999 999 999 の範囲の値を指定できます。値 0 では、すべてのクライアント・アクセスができなくなります。

このフィールドのデフォルト値は 999 999 999 です。

このフィールドの値が、現在実行中のサーバー接続チャンネルのインスタンス数より少ない数まで引き下げられる場合でも、実行中のインスタンスは影響を受けません。しかし、十分な数の既存のインスタンスが実行を終了して、現在実行中のインスタンスの数がフィールドの値を下回らないと、新規インスタンスは開始できません。

MaxInstancesPerClient (MQLONG)

このフィールドは、個別のサーバー接続チャンネルの、単一クライアントから開始可能な同時インスタンスの最大数を指定します。

このコンテキストでは、同じリモート・ネットワーク・アドレスから発信された接続は、同じクライアントから着信したものと見なされます。

このフィールドは、サーバー接続チャンネルでのみ使用されます。

このフィールドには、0 から 999 999 999 の範囲の値を指定できます。値 0 では、すべてのクライアント・アクセスができなくなります。

このフィールドのデフォルト値は 999 999 999 です。

このフィールドの値が、個別のクライアントから現在実行中のサーバー接続チャンネルのインスタンス数より少ない数まで引き下げられる場合でも、実行中のインスタンスは影響を受けません。しかし、十分な数の既存インスタンスが実行を終了して、新規インスタンスの開始を試みているクライアントから起動した現在実行中のインスタンスの数がフィールドの値を下回るまで、これらのクライアントのいずれかから新規インスタンスを開始することはできません。

MaxMsgLength (MQLONG)

このフィールドは、チャンネル上で送信可能な最大メッセージ長を指定します。

この値は、リモート・チャンネルの値と比較され、実際の最大長は、2 つの値のうちの小さいほうの値になります。

MCAName (MQCHAR20)

このフィールドは予約フィールドです。

このフィールドの値はブランクです。

このフィールドの長さは MQ_MCA_NAME_LENGTH によって指定されます。

MCASecurityId (MQBYTE40)

このフィールドは、MCA のセキュリティ ID を指定します。

このフィールドは、*ChannelType* として MQCHT_CLNTCONN が指定されているチャンネルとは関係ありません。

次の特殊値は、セキュリティ ID がないことを示します。

MQSID_NONE

セキュリティ ID が指定されていない。

値は、フィールドの長さについては 2 進ゼロです。

C 言語の場合、定数 MQSID_NONE_ARRAY も定義されます。この定数は、MQSID_NONE と同じ値ですが、ストリングではなく文字の配列です。

これは、出口に対する入出力フィールドです。このフィールドの長さは MQ_SECURITY_ID_LENGTH によって指定されます。Version が MQCD_VERSION_6 より小さい場合は、このフィールドは提供されません。

MCAType (MQLONG)

このフィールドは、メッセージ・チャンネル・エージェント・プログラムのタイプを指定します。

このフィールドは、ChannelType として MQCHT_SENDER、MQCHT_SERVER、MQCHT_REQUESTER、MQCHT_CLUSSDR、または MQCHT_CLUSRCVR が指定されているチャンネルのみに関連します。

値は、次のいずれか 1 つです。

MQMCAT_PROCESS

プロセス。

メッセージ・チャンネル・エージェントは、独立のプロセスとして動作します。

MQMCAT_THREAD

スレッド (マルチプラットフォーム)。

メッセージ・チャンネル・エージェントは独立したスレッドとして実行されます。

Version が MQCD_VERSION_2 より小さい場合は、このフィールドは提供されません。

MCAUserIdentifier (MQCHAR12)

このフィールドは、メッセージ・チャンネル・エージェント (MCA) のユーザー ID を指定します。

このフィールドは、MCA ユーザー ID の最初の 12 バイトを使用し、セキュリティー・エージェントによって設定できます。

MCA ユーザー ID が入るフィールドは次の 2 つです。

- **MCAUserIdentifier**。MCA ユーザー ID の最初の 12 バイトが入り、ID が 12 バイトより短い場合はブランクが埋められます。MCAUserIdentifier はブランクにすることができます。
- **LongMCAUserIdPtr**。完全な MCA ユーザー ID を指します。ID は 12 バイトより長くてもかまいません。ID の長さは LongMCAUserIdLength で指定されます。完全な ID は末尾ブランクを含まず、ヌル文字で終了しません。ID がブランクの場合は、LongMCAUserIdLength はゼロとなり、LongMCAUserIdPtr の値は定義されません。

注: Version が MQCD_VERSION_6 より小さい場合は、LongMCAUserIdPtr は提供されません。

MCA ユーザー ID がブランクでない場合、これはメッセージ・チャンネル・エージェントが IBM MQ リソースのアクセス権限を得るために使用するユーザー ID を指定します。チャンネル・タイプ MQCHT_REQUESTER、MQCHT_RECEIVER、および MQCHT_CLUSRCVR では、PutAuthority が MQPA_DEFAULT である場合、これは、宛先キューへの書き込み操作の許可検査を行うために使用されるユーザー ID です。

MCA ユーザー ID がブランクの場合は、メッセージ・チャンネル・エージェントはデフォルトのユーザー ID を使用します。

MCA ユーザー ID は、メッセージ・チャンネル・エージェントが使用するはずのユーザー ID を示すために、セキュリティー出口によって設定できます。出口では、MCAUserIdentifier を変更するか、LongMCAUserIdPtr が指すストリングを変更することができます。両方変更した結果が一致しない場合は、MCA は LongMCAUserIdPtr を MCAUserIdentifier より優先して使用します。LongMCAUserIdPtr が示すストリングの長さを出口で変更した場合は、それに対応して LongMCAUserIdLength も設定する必要があります。出口で ID の長さを増やす場合は、必要な長さのストレージを割り振って、割り振ったストレージを必要な ID に設定し、このストレージのアドレスを LongMCAUserIdPtr に入れる必要があります。あとで MQXR_TERM の理由コードで出口が呼び出されたときは、このストレージの解放は出口の責任です。

ChannelType が MQCHT_SVRCONN のチャンネルについては、チャンネル定義内の *MCAUserIdentifier* がブランクであると、クライアントから転送される任意のユーザー ID がこれにコピーされます。クライアント・アプリケーションは、このようなユーザー ID (サーバーでセキュリティ出口によってなんらかの変更が行われてから) のいずれかのもとで実行されるものと想定されます。

MCA ユーザー ID は、*ChannelType* が MQCHT_SDR、MQCHT_SVR、MQCHT_CLNTCONN、MQCHT_CLUSSDR のチャンネルには適用されません。

これは、出口に対する入出力フィールドです。このフィールドの長さは MQ_USER_ID_LENGTH によって指定されます。*Version* が MQCD_VERSION_2 より小さい場合は、このフィールドは提供されません。

ModeName (MQCHAR8)

このフィールドは、LU 6.2 モード名を指定します。

このフィールドは、伝送プロトコル (*TransportType*) が MQXPT_LU62 であり、*ChannelType* が MQCHT_SVRCONN でも MQCHT_RECEIVER でもない場合にのみ適用されます。

このフィールドは常にブランクです。このフィールドの代わりに、通信サイド・オブジェクトに情報が格納されます。

このフィールドの長さは MQ_MODE_NAME_LENGTH によって指定されます。

MsgCompList [16] (MQLONG)

このフィールドは、チャンネルがサポートするメッセージ・データ圧縮技法のリストを指定します。

リストには、以下の 1 つ以上の値が含まれます。

MQCOMPRESS_NONE

メッセージ・データ圧縮は実行されません。

MQCOMPRESS_RLE

ラン・レングス・エンコードを使用してメッセージ・データ圧縮が実行されます。

MQCOMPRESS_ZLIBFAST

zlib 圧縮手法を使用してメッセージ・データ圧縮が実行されます。高速圧縮時間を推奨します。

MQCOMPRESS_ZLIBHIGH

zlib 圧縮手法を使用してメッセージ・データ圧縮が実行されます。ハイレベル圧縮を推奨します。

LZ4FAST

メッセージデータの圧縮は、速度を優先した LZ4 符号化を用いて行われる。

LZ4HIGH

メッセージデータの圧縮は、圧縮を優先した LZ4 エンコーディングを用いて行われる。

MQCOMPRESS_ANY

キューマネージャがサポートする任意の圧縮技術は、メッセージ圧縮に使用できます。

MQCOMPRESS_ANY は受信側、要求側、およびサーバー接続チャンネルの場合にのみ有効です。

MQCOMPRESS_NOT_AVAILABLE

リスト内の未使用の値はこの値に設定されます。

これは、出口に対する入力フィールドです。*Version* が MQCD_VERSION_8 より小さい場合は、このフィールドは提供されません。

MsgExit (MQCHARn)

このフィールドは、チャンネル・メッセージ出口名を指定します。

この名前が非ブランクの場合、出口は以下の時点で呼び出されます。

- メッセージが伝送キュー (送信側またはサーバー) から検索された直後、またはメッセージが宛先キュー (受信側または要求側) に書き込まれる直前。

出口には、アプリケーション・メッセージおよび変更用伝送キュー・ヘッダーの全体が提供されます。

- チャンネルの初期設定時および終了時

このフィールドは、*ChannelType* が MQCHT_SVRCONN または MQCHT_CLNTCONN であるチャンネルには適用されません。このようなチャンネルに対してメッセージ出口を呼び出すことはできません。

種々の環境におけるこのフィールド内容の説明については、1480 ページの『MQCD - チャンネル定義』を参照してください。

このフィールドの長さは MQ_EXIT_NAME_LENGTH によって指定されます。

注：この定数の値は環境によります。

MsgExitPtr (MQPTR)

このフィールドは、最初の *MsgExit* フィールドのアドレスを指定します。

MsgExitsDefined がゼロより大きいと、このアドレスはチェーン内の各チャンネル・メッセージ出口の名前リストのアドレスです。

各名前は、*ExitNameLength* で指定される長さのフィールドにあり、フィールド内の右側は空白で埋められます。各出口ごとに、隣接し合う *MsgExitsDefined* フィールドがあります。

これらの名前に対して出口が加えた変更は保存されますが、メッセージ・チャンネル出口は明示的なアクションをとりません。つまり、メッセージ・チャンネル出口は、呼び出す出口を変更しません。

MsgExitsDefined がゼロの場合は、このフィールドはヌル・ポインターとなります。

プログラミング言語がポインターのデータ・タイプをサポートしていないプラットフォームでは、このフィールドは適切な長さのバイト・ストリングとして宣言されます。

これは、出口に対する入力フィールドです。Version が MQCD_VERSION_4 より小さい場合は、このフィールドは提供されません。

MsgExitsDefined (MQLONG)

このフィールドは、チェーン内に定義されるチャンネル・メッセージ出口の数を指定します。

これは、ゼロ以上の値です。

これは、出口に対する入力フィールドです。Version が MQCD_VERSION_4 より小さい場合は、このフィールドは提供されません。

MsgRetryCount (MQLONG)

このフィールドは、MCA がメッセージを最初に書き込もうとして失敗した後に、書き込みを試行する回数を指定します。

このフィールドは、最初の MQOPEN または MQPUT が完了コード MQCC_FAILED で失敗した場合に、MCA が OPEN 操作または PUT 操作を試行する回数を指定します。この属性の効果は、*MsgRetryExit* が空白か非空白かによって異なります。

- *MsgRetryExit* が空白の場合は、**MsgRetryCount** 属性は MCA が再試行を行うかどうかを制御します。属性値が 0 の場合は、再試行は行われません。属性値が 0 より大きい場合は、再試行は **MsgRetryInterval** 属性で指定されている間隔で行われます。

再試行は、次の理由コードが戻された場合にだけ行われます。

- MQRC_PAGESET_FULL
- MQRC_PUT_INHIBITED
- MQRC_Q_FULL

それ以外の理由コードの場合は、MCA は、障害の起こったメッセージを再試行しないで、すぐに通常の障害処理を続行します。

- *MsgRetryExit* が非空白の場合には、**MsgRetryCount** 属性は MCA に影響しません。代わりに、メッセージ再試行出口が、再試行を行う回数と間隔を決定します。この出口は、**MsgRetryCount** 属性がゼロであっても呼び出されます。

MsgRetryCount 属性は MQCD 構造体内でこの出口に使用可能になりますが、出口がこの属性を採用する必要は必ずしもありません。出口が MQCXP の *ExitResponse* フィールドに MQXCC_SUPPRESS_FUNCTION を戻すまで、再試行は無限に続行されます。

このフィールドは、*ChannelType* として MQCHT_REQUESTER、MQCHT_RECEIVER、または MQCHT_CLUSRCVR が指定されているチャンネルのみに関連します。

Version が MQCD_VERSION_3 より小さい場合は、このフィールドは提供されません。

MsgRetryExit (MQCHARn)

このフィールドは、チャンネル・メッセージ再試行出口名を指定します。

メッセージ再試行出口は、MCA が MQOPEN 呼び出しまたは MQPUT 呼び出しから MQCC_FAILED の完了コードを受信したときに MCA によって呼び出される出口です。この出口の目的は、MCA が MQOPEN または MQPUT 操作を再試行するまでに待機する時間間隔を指定することです。あるいは、操作が再試行されないように出口を設定することもできます。

出口は、完了コードが MQCC_FAILED のあらゆる理由コードに対して呼び出されます。出口の設定により、MCA に再試行させる理由コード、再試行の回数、および再試行の時間間隔が決まります。

操作を再試行しない場合には、MCA は通常の障害処理を行います。この処理では、例外レポート・メッセージを作成し (送信側によって指定されている場合)、元のメッセージを送達不能キューに入れるか、またはメッセージを廃棄します (送信側が MQRO_DEAD_LETTER_Q または MQRO_DISCARD_MSG のどちらを指定しているかによって決まります)。送達不能キューに関する障害 (例えば、送達不能キューが満杯になっているなど) の場合には、メッセージ再試行出口は呼び出されません。

出口名が非ブランクの場合、この出口は以下の時点で呼び出されます。

- メッセージの送信を再試行するまでの待機状態に入る直前。
- チャンネルの初期設定時および終了時。

種々の環境におけるこのフィールド内容の説明については、[1480 ページの『MQCD - チャンネル定義』](#)を参照してください。

このフィールドは、*ChannelType* として MQCHT_REQUESTER、MQCHT_RECEIVER、または MQCHT_CLUSRCVR が指定されているチャンネルのみに関連します。

このフィールドの長さは MQ_EXIT_NAME_LENGTH によって指定されます。

注: この定数の値は環境によります。

Version が MQCD_VERSION_3 より小さい場合は、このフィールドは提供されません。

MsgRetryInterval (MQLONG)

このフィールドは、OPEN 操作または PUT 操作が再試行された後の最小間隔 (ミリ秒数) を指定します。

この属性の効果は、*MsgRetryExit* がブランクか非ブランクかによって異なります。

- *MsgRetryExit* がブランクの場合には、**MsgRetryInterval** 属性が、最初の MQOPEN または MQPUT が完了コード MQCC_FAILED で失敗した場合に、MCA がメッセージを再試行するまでに待機する最小期間を指定します。値 0 は、前回の試みの直後に再試行を行うことを意味します。再試行は、*MsgRetryCount* が 0 より大きい場合にだけ行われます。

この属性は、メッセージ再試行出口が MQCXP の *MsgRetryInterval* フィールドに無効な値を返した場合の待機時間としても使用されます。

- *MsgRetryExit* が非ブランクの場合には、**MsgRetryInterval** 属性は MCA に影響しません。代わりに、メッセージ再試行出口が、MCA の待機時間を決定します。**MsgRetryInterval** 属性は MQCD 構造体内でこの出口に使用可能になりますが、出口がこの属性を使用する必要は必ずしもありません。

値の範囲は 0 から 999 999 999 です。

このフィールドは、*ChannelType* として MQCHT_REQUESTER、MQCHT_RECEIVER、または MQCHT_CLUSRCVR が指定されているチャンネルのみに関連します。

Version が MQCD_VERSION_3 より小さい場合は、このフィールドは提供されません。

Version が MQCD_VERSION_4 より小さい場合は、この構造体の以下のフィールドは提供されません。

MsgRetryUserData (MQCHAR32)

このフィールドは、チャンネル・メッセージ再試行出口ユーザー・データを指定します。

このデータは、チャンネル・メッセージ再試行出口の **ChannelExitParms** パラメーターの *ExitData* フィールドに渡されます (MQ_CHANNEL_EXIT を参照してください)。

初めは、このフィールドには、チャンネル定義内に設定されたデータが入っています。しかし、この MCA インスタンスの存続期間中に、いずれかのタイプの出口によってこのフィールドの内容が変更されると、それは MCA によって保存され、この MCA インスタンスにおける後続の出口呼び出し (タイプに関係なく) で見ることができます。このような変更は、MCA インスタンスが使用するチャンネル定義には影響ありません。どのような文字でも (2 進データも含みます) 使用可能です。

このフィールドは、*ChannelType* として MQCHT_REQUESTER、MQCHT_RECEIVER、または MQCHT_CLUSRCVR が指定されているチャンネルのみに関連します。

このフィールドの長さは MQ_EXIT_DATA_LENGTH によって指定されます。 *Version* が MQCD_VERSION_3 より小さい場合は、このフィールドは提供されません。

このフィールドは、IBM MQ for IBM i では関連しません。

MsgUserData (MQCHAR32)

このフィールドは、チャンネル・メッセージ出口ユーザー・データを指定します。

このデータは、**ChannelExitParms** パラメーターの *ExitData* フィールドでチャンネル・メッセージ出口に渡されます (MQ_CHANNEL_EXIT を参照してください)。

初めは、このフィールドには、チャンネル定義内に設定されたデータが入っています。しかし、この MCA インスタンスの存続期間中に、いずれかのタイプの出口によってこのフィールドの内容が変更されると、それは MCA によって保存され、この MCA インスタンスにおける後続の出口呼び出し (タイプに関係なく) で見ることができます。このような変更は、MCA インスタンスが使用するチャンネル定義には影響ありません。どのような文字でも (2 進データも含みます) 使用可能です。

このフィールドの長さは MQ_EXIT_DATA_LENGTH によって指定されます。

このフィールドは、IBM MQ for IBM i では関連しません。

MsgUserDataPtr (MQPTR)

このフィールドは、最初の *MsgUserData* フィールドのアドレスを指定します。

MsgExitsDefined がゼロより大きいと、このアドレスはチェーン内の各チャンネル・メッセージ出口用のユーザー・データ項目リストのアドレスです。

各ユーザー・データ項目は、*ExitDataLength* で指定される長さのフィールドにあり、フィールド内の右側はブランクで埋められます。各出口ごとに、隣接し合う *MsgExitsDefined* フィールドがあります。定義済みユーザー・データ項目数が、出口名の数より少ない場合、未定義のユーザー・データ項目はブランクに設定されます。逆に、定義済みのユーザー・データ項目数が出口名の数より多い場合、超過したユーザー・データ項目は無視され、出口には渡されません。

出口がこれらの値に対して加えた変更は保存されます。そのため、ある出口が別の出口に情報を渡すことができます。変更について妥当性検査は行われません。例えば、必要であれば 2 進データをこれらのフィールドに書き込むことができます。

MsgExitsDefined がゼロの場合は、このフィールドはヌル・ポインターとなります。

プログラミング言語がポインターのデータ・タイプをサポートしていないプラットフォームでは、このフィールドは適切な長さのバイト・ストリングとして宣言されます。

これは、出口に対する入力フィールドです。 *Version* が MQCD_VERSION_4 より小さい場合は、このフィールドは提供されません。

NetworkPriority (MQLONG)

このフィールドは、チャンネルのネットワーク接続の優先順位を指定します。

特定の宛先への複数のパスが使用できる場合、最高の優先順位のパスが選択されます。値の範囲は 0 から 9 であり、0 が最低の優先順位です。

このフィールドは、*ChannelType* として MQCHT_CLUSSDR または MQCHT_CLUSRCVR が指定されているチャンネルのみに関連します。

これは、出口に対する入力フィールドです。Version が MQCD_VERSION_5 より小さい場合は、このフィールドは提供されません。

Version が MQCD_VERSION_6 より小さい場合は、この構造体の以下のフィールドは提供されません。

NonPersistentMsgSpeed (MQLONG)

このフィールドは、非永続メッセージがチャンネル内を移動する速度を指定します。

このフィールドは、*ChannelType* として MQCHT_SENDER、MQCHT_SERVER、MQCHT_RECEIVER、MQCHT_REQUESTER、MQCHT_CLUSSDR、または MQCHT_CLUSRCVR が指定されているチャンネルのみに関連します。

値は、次のいずれか 1 つです。

MQNPMS_NORMAL

通常速度。

チャンネルが MQNPMS_NORMAL と定義されている場合は、非永続メッセージはチャンネル内を標準速度で移動します。この場合、チャンネル障害が発生しても、これらのメッセージが失われないという利点があります。また、同じ伝送キュー内の持続メッセージおよび非永続メッセージは、お互いに相対的な順序を維持します。

MQNPMS_FAST

高速。

チャンネルが MQNPMS_FAST と定義されている場合は、非永続メッセージはチャンネル内を高速で移動します。これにより、チャンネルのスループットは改善されますが、チャンネル障害が発生すると、非永続メッセージが失われることになります。また、非永続メッセージが同じ伝送キュー内で待機している持続メッセージを飛び越えることも可能です。非永続メッセージの順序は持続メッセージと相対的に維持されないということです。ただし、非永続メッセージ間の相対的な順序は維持されます。同様に、永続メッセージ間の相対的な順序も維持されます。

Password (MQCHAR12)

このフィールドは、メッセージ・チャンネル・エージェントが、リモート・メッセージ・チャンネル・エージェントとの保護 SNA セッションの開始を試みるときに使用するパスワードを指定します。

このフィールドは、AIX, Linux, and Windows でのみ非ブランクにでき、*ChannelType* が MQCHT_SENDER、MQCHT_SERVER、MQCHT_REQUESTER、または MQCHT_CLNTCONN のチャンネルにのみ適用されます。z/OS では、このフィールドは適用されません。

このフィールドの長さは MQ_PASSWORD_LENGTH によって指定されます。ただし、使用されるのは最初の 10 文字のみです。

Version が MQCD_VERSION_2 より小さい場合は、このフィールドは提供されません。

PropertyControl (MQLONG)

このフィールドは、メッセージが V6 またはそれより前のキュー・マネージャー (プロパティー記述子の概念を理解しないキュー・マネージャー) に送信されるときに、メッセージのプロパティーに対して行われる処置を指定します。

値には以下のいずれかの値を指定できます。

MQPROP_COMPATIBILITY

メッセージに接頭部 **mcd.**、**jms.**、**usr.**、または **mqext.** があるプロパティーが含まれている場合、すべてのメッセージ・プロパティーは、MQRFH2 ヘッダーに入れられてアプリケーションに送達されます。それらの接頭部を持つプロパティーがない場合、メッセージ記述子 (または拡張) に含まれるプ

ロパティを除外して、メッセージのプロパティはすべて廃棄され、アプリケーションからはアクセスできなくなります。

この値はデフォルト値です。これにより、JMS 関連プロパティがメッセージ・データの MQRFH2 ヘッダーに存在することを想定して機能するアプリケーションは、未変更のまま処理を続行できます。

MQPROP_NONE

メッセージがリモート・キュー・マネージャーに送信される前に、メッセージ記述子 (または拡張子) に含まれるプロパティを除外して、メッセージのプロパティはすべてメッセージから除去されます。

MQPROP_ALL

メッセージのすべてのプロパティは、リモート・キュー・マネージャーへの送信時にメッセージに組み込まれます。メッセージ記述子 (または拡張子) に含まれるプロパティを除き、プロパティはメッセージ・データ内の 1 つ以上の MQRFH2 ヘッダーに入れられます。

この属性は、送信側、サーバー、クラスター送信側、およびクラスター受信側の各チャンネルに適用可能です。

68 ページの『MQIA * (整数属性セクター)』

108 ページの『MQPROP * (キューおよびチャンネル・プロパティ制御値および 最大プロパティ長)』

PutAuthority (MQLONG)

宛先キューにメッセージを書き込む権限を確立するために、メッセージに関連するコンテキスト情報にあるユーザー ID を使用するかどうかを指定します。

このフィールドは、*ChannelType* として MQCHT_REQUESTER、MQCHT_RECEIVER、または MQCHT_CLUSRCVR が指定されているチャンネルのみに関連します。これは、以下のいずれかになります。

MQPA_DEFAULT

デフォルト・ユーザー ID が使用されます。

MQPA_CONTEXT

コンテキスト・ユーザー ID が使用されます。

MQPA_ALTERNATE_OR_MCA

メッセージ記述子の UserIdentifier フィールドから得たユーザー ID が使用されます。ネットワークから受信したユーザー ID はどれも使用されません。この値は z/OS でのみサポートされます。

MQPA_ONLY_MCA

デフォルトのユーザー ID が使用されます。ネットワークから受信したユーザー ID はどれも使用されません。この値は z/OS でのみサポートされます。

QuantumSafeAlgorithm (MQLONG)

このフィールドは、TLS 接続のセキュリティを確保する際に使用する量子耐性アルゴリズムを指定します。

値は、次のいずれか 1 つです。

MQQS_UNPROTECTED

量子耐性アルゴリズムは使用しないでください。

MQQS_HYBMLKEM

ハイブリッド ML-KEM 量子耐性アルゴリズムを使用する。

MQQS_MLKEM

ML-KEM 量子耐性アルゴリズムを使用してください。

これは、出口に対する入力フィールドです。が MQCD_VERSION_13 未満の場合 Version、このフィールドは表示されません。

詳細については、『IBM MQ』の「Quantum Safe TLS」をご覧ください。

QuantumSafeRequired (MQLONG)

このフィールドは、量子耐性通信が必須か、それとも従来の TLS へのフォールバックが許可されるかを指定します。

値は、次のいずれか 1 つです。

MQQSR_OPTIONAL

従来の TLS または Quantum Safe 通信を許可します。

MQQSR_必須

Quantum Safe 通信が利用できない場合は、チャネルの起動を阻止する。

これは、出口に対する入力フィールドです。が MQCD_VERSION_13 未満の場合 Version、このフィールドは表示されません。

詳細については、『IBM MQ』の「Quantum Safe TLS」をご覧ください。

QMGrName (MQCHAR48)

このフィールドは、出口が接続可能なキュー・マネージャーの名前を指定します。

ChannelType が MQCHT_CLNTCONN 以外のチャネルの場合、出口が接続できるキュー・マネージャーの名前を指定するこのフィールドは、AIX, Linux, and Windows システム上では常に非ブランクです。

このフィールドの長さは MQ_Q_MGR_NAME_LENGTH で指定します。

ReceiveExit (MQCHARn)

このフィールドは、チャネル受信出口名を指定します。

この名前が非ブランクの場合、出口は以下の時点で呼び出されます。

- 受信されたネットワーク・データが処理される直前。

出口には、受信された伝送バッファ全体が与えられます。バッファの内容は、必要に応じて変更できます。

- チャネルの初期設定時および終了時

種々の環境におけるこのフィールド内容の説明については、[1480 ページの『MQCD - チャネル定義』](#)を参照してください。

このフィールドの長さは MQ_EXIT_NAME_LENGTH によって指定されます。

注：この定数の値は環境によります。

ReceiveExitPtr (MQPTR)

このフィールドは、最初の *ReceiveExit* フィールドのアドレスを指定します。

ReceiveExitsDefined がゼロより大きいと、このアドレスはチェーン内の各チャネル受信出口名のリストのアドレスです。

各名前は、*ExitNameLength* で指定される長さのフィールドにあり、フィールド内の右側はブランクで埋められます。各出口ごとに、隣接し合う *ReceiveExitsDefined* フィールドがあります。

これらの名前に対して出口が加えた変更は保存されますが、メッセージ・チャネル出口は明示的なアクションをとりません。つまり、メッセージ・チャネル出口は、呼び出す出口を変更しません。

ReceiveExitsDefined がゼロの場合は、このフィールドはヌル・ポインターとなります。

プログラミング言語がポインターのデータ・タイプをサポートしていないプラットフォームでは、このフィールドは適切な長さのバイト・ストリングとして宣言されます。

これは、出口に対する入力フィールドです。Version が MQCD_VERSION_4 より小さい場合は、このフィールドは提供されません。

ReceiveExitsDefined (MQLONG)

このフィールドは、チェーン内に定義されるチャネル受信出口の数を指定します。

これは、ゼロ以上の値です。

これは、出口に対する入力フィールドです。Version が MQCD_VERSION_4 より小さい場合は、このフィールドは提供されません。

ReceiveUserData (MQCHAR32)

このチャネルは、チャネル受信出口ユーザー・データを指定します。

このデータは、**ChannelExitParms** パラメーターの *ExitData* フィールドでチャンネル受信出口に渡されます (MQ_CHANNEL_EXIT を参照してください)。

初めは、このフィールドには、チャンネル定義内に設定されたデータが入っています。しかし、この MCA インスタンスの存続期間中に、いずれかのタイプの出口によってこのフィールドの内容が変更されると、それは MCA によって保存され、この MCA インスタンスにおける後続の出口呼び出し (タイプに関係なく) で見ることができます。これは種々の会話での出口に適用されます。このような変更は、MCA インスタンスが使用するチャンネル定義には影響ありません。どのような文字でも (2 進データも含みます) 使用可能です。

このフィールドの長さは MQ_EXIT_DATA_LENGTH によって指定されます。

このフィールドは、IBM MQ for IBM i では関連しません。

Version が MQCD_VERSION_2 より小さい場合は、この構造体の以下のフィールドは提供されません。

ReceiveUserDataPtr (MQPTR)

このフィールドは、最初の *ReceiveUserData* フィールドのアドレスを指定します。

ReceiveExitsDefined がゼロより大きいと、このアドレスはチェーン内の各チャンネル受信出口用のユーザー・データ項目リストのアドレスです。

各ユーザー・データ項目は、*ExitDataLength* で指定される長さのフィールドにあり、フィールド内の右側はブランクで埋められます。各出口ごとに、隣接し合う *ReceiveExitsDefined* フィールドがあります。定義済みユーザー・データ項目数が、出口名の数より少ない場合、未定義のユーザー・データ項目はブランクに設定されます。逆に、定義済みのユーザー・データ項目数が出口名の数より多い場合、超過したユーザー・データ項目は無視され、出口には渡されません。

出口がこれらの値に対して加えた変更は保存されます。そのため、ある出口が別の出口に情報を渡すことができます。変更について妥当性検査は行われません。例えば、必要であれば 2 進データをこれらのフィールドに書き込むことができます。

ReceiveExitsDefined がゼロの場合は、このフィールドはヌル・ポインターとなります。

プログラミング言語がポインターのデータ・タイプをサポートしていないプラットフォームでは、このフィールドは適切な長さのバイト・ストリングとして宣言されます。

これは、出口に対する入力フィールドです。*Version* が MQCD_VERSION_4 より小さい場合は、このフィールドは提供されません。

Version が MQCD_VERSION_5 より小さい場合は、この構造体の以下のフィールドは提供されません。

RemotePassword (MQCHAR12)

このフィールドは、相手側からのパスワードを指定します。

このフィールドに有効な情報が入るのは、*ChannelType* が MQCHT_CLNTCONN または MQCHT_SVRCONN の場合だけです。

- MQCHT_CLNTCONN チャンネルのセキュリティー出口の場合、このパスワードは環境から取得されたパスワードです。出口は、サーバーのセキュリティー出口にそれを送信するよう選択することができます。
- MQCHT_SVRCONN チャンネルでのセキュリティー出口の場合、クライアント・セキュリティー出口がなければ、このフィールドにはクライアントの環境から入手されるパスワードが入ることがあります。出口は、このパスワードを使用して、*RemoteUserIdentifier* のユーザー ID を検証できます。

クライアントにセキュリティー出口があれば、この情報はクライアントからのセキュリティーの一環として入手することができます。

このフィールドの長さは MQ_PASSWORD_LENGTH によって指定されます。*Version* が MQCD_VERSION_2 より小さい場合は、このフィールドは提供されません。

RemoteSecurityId (MQBYTE40)

このフィールドは、リモート・ユーザーのセキュリティー ID を指定します。

このフィールドは、*ChannelType* として MQCHT_CLNTCONN または MQCHT_SVRCONN が指定されているチャンネルのみに関連します。

次の特殊値は、セキュリティー ID がないことを示します。

MQSID_NONE

セキュリティー ID が指定されていない。

値は、フィールドの長さについては 2 進ゼロです。

C 言語の場合、定数 `MQSID_NONE_ARRAY` も定義されます。この定数は、`MQSID_NONE` と同じ値ですが、ストリングではなく文字の配列です。

これは、出口に対する入力フィールドです。このフィールドの長さは `MQ_SECURITY_ID_LENGTH` によって指定されます。*Version* が `MQCD_VERSION_6` より小さい場合は、このフィールドは提供されません。

Version が `MQCD_VERSION_7` より小さい場合は、この構造体の以下のフィールドは提供されません。

RemoteUserIdentifier (MQCHAR12)

このフィールドは、相手側からのユーザー ID の最初の 12 バイトを指定します。

リモート・ユーザー ID が入るフィールドは次の 2 つです。

- *RemoteUserIdentifier*。リモート・ユーザー ID の最初の 12 バイトが入り、ID が 12 バイトより短い場合はブランクが埋められます。*RemoteUserIdentifier* はブランクにすることができます。
- *LongRemoteUserIdPtr*。完全なリモート・ユーザー ID を指します。ID は 12 バイトより長くてもかまいません。ID の長さは *LongRemoteUserIdLength* で指定されます。完全な ID は末尾ブランクを含まず、ヌル文字で終了しません。ID がブランクの場合は、*LongRemoteUserIdLength* はゼロとなり、*LongRemoteUserIdPtr* の値は定義されません。

Version が `MQCD_VERSION_6` より小さい場合は、*LongRemoteUserIdPtr* は提供されません。

リモート・ユーザー ID は、*ChannelType* として `MQCHT_CLNTCONN` または `MQCHT_SVRCONN` が指定されているチャンネルのみに関連します。

- `MQCHT_CLNTCONN` チャンネルのセキュリティー出口の場合、この値は環境から取得されたユーザー ID です。出口は、サーバーのセキュリティー出口にそれを送信するよう選択することができます。
- `MQCHT_SVRCONN` チャンネルでのセキュリティー出口の場合、クライアント・セキュリティー出口がなければ、このフィールドにはクライアントの環境から入手されたユーザー ID が入ることがあります。その場合、出口はこのユーザー ID を (おそらく、*RemotePassword* に格納されたパスワードを使用して) 検証し、*MCAUserIdentifier* の値を更新します。

クライアントにセキュリティー出口があれば、この情報はクライアントからのセキュリティーの一環として入手することができます。

このフィールドの長さは `MQ_USER_ID_LENGTH` によって指定されます。*Version* が `MQCD_VERSION_2` より小さい場合は、このフィールドは提供されません。

SecurityExit (MQCHARn)

このフィールドは、チャンネル・セキュリティー出口名を指定します。

この名前が非ブランクの場合、出口は以下の時点で呼び出されます。

- チャンネルが確立された直後。
いかなるメッセージ転送も行われないうちに、この出口には、セキュリティー・フローを開始し、接続許可の妥当性を検査することができます。
- セキュリティー・メッセージ・フローに対する応答を受け取ったとき。
リモート・マシンのリモート・プロセッサから受け取ったセキュリティー・メッセージ・フローが出口に与えられます。
- チャンネルの初期設定時および終了時

種々の環境におけるこのフィールド内容の説明については、[1480 ページの『MQCD - チャンネル定義』](#)を参照してください。

このフィールドの長さは `MQ_EXIT_NAME_LENGTH` によって指定されます。

注: この定数の値は環境によります。

SecurityUserData (MQCHAR32)

このチャネルは、チャネル・セキュリティ・出口ユーザー・データを指定します。

このデータは、**ChannelExitParms** パラメーターの *ExitData* フィールドでチャネル・セキュリティ・出口に渡されます (MQ_CHANNEL_EXIT を参照してください)。

初めは、このフィールドには、チャネル定義内に設定されたデータが入っています。しかし、この MCA インスタンスの存続期間中に、いずれかのタイプの出口によってこのフィールドの内容が変更されると、それは MCA によって保存され、この MCA インスタンスにおける後続の出口呼び出し (タイプに関係なく) で見ることができます。これは種々の会話での出口に適用されます。このような変更は、MCA インスタンスが使用するチャネル定義には影響ありません。どのような文字でも (2 進データも含みます) 使用可能です。

このフィールドの長さは MQ_EXIT_DATA_LENGTH によって指定されます。

このフィールドは、IBM MQ for IBM i では関連しません。

SendExit (MQCHARn)

このフィールドは、チャネル送信出口名を指定します。

この名前が非ブランクの場合、出口は以下の時点で呼び出されます。

- データがネットワークに送り出される直前。

伝送バッファが伝送される前に、出口に伝送バッファ全体が提供されます。バッファの内容は、必要に応じて変更できます。

- チャネルの初期設定時および終了時

種々の環境におけるこのフィールド内容の説明については、[1480 ページの『MQCD - チャネル定義』](#)を参照してください。

このフィールドの長さは MQ_EXIT_NAME_LENGTH によって指定されます。

注: この定数の値は環境によります。

SendExitPtr (MQPTR)

このフィールドは、最初の *SendExit* フィールドのアドレスを指定します。

SendExitsDefined がゼロより大きいと、このアドレスはチェーン内の各チャネル送信出口名のリストのアドレスです。

各名前は、*ExitNameLength* で指定される長さのフィールドにあり、フィールド内の右側はブランクで埋められます。各出口ごとに、隣接し合う *SendExitsDefined* フィールドがあります。

これらの名前に対して出口が加えた変更は保存されますが、メッセージ送信出口は明示的なアクションをとりません。つまり、メッセージ送信出口は、呼び出す出口を変更しません。

SendExitsDefined がゼロの場合は、このフィールドはヌル・ポインターとなります。

プログラミング言語がポインターのデータ・タイプをサポートしていないプラットフォームでは、このフィールドは適切な長さのバイト・ストリングとして宣言されます。

これは、出口に対する入力フィールドです。Version が MQCD_VERSION_4 より小さい場合は、このフィールドは提供されません。

SendExitsDefined (MQLONG)

このフィールドは、チェーン内に定義されるチャネル送信出口の数を指定します。

これは、ゼロ以上の値です。

これは、出口に対する入力フィールドです。Version が MQCD_VERSION_4 より小さい場合は、このフィールドは提供されません。

SendUserData (MQCHAR32)

このフィールドは、チャンネル送信出口ユーザー・データを指定します。

このデータは、**ChannelExitParms** パラメーターの *ExitData* フィールドでチャンネル送信出口に渡されます (MQ_CHANNEL_EXIT を参照してください)。

初めは、このフィールドには、チャンネル定義内に設定されたデータが入っています。しかし、この MCA インスタンスの存続期間中に、いずれかのタイプの出口によってこのフィールドの内容が変更されると、それは MCA によって保存され、この MCA インスタンスにおける後続の出口呼び出し (タイプに関係なく) で見ることができます。これは種々の会話での出口に適用されます。このような変更は、MCA インスタンスが使用するチャンネル定義には影響ありません。どのような文字でも (2 進データも含みます) 使用可能です。

このフィールドの長さは MQ_EXIT_DATA_LENGTH によって指定されます。

このフィールドは、IBM MQ for IBM i では関連しません。

SendUserDataPtr (MQPTR)

このフィールドは、*SendUserData* フィールドのアドレスを指定します。

SendExitsDefined がゼロより大きいと、このアドレスはチェーン内の各チャンネル・メッセージ出口用のユーザー・データ項目リストのアドレスです。

各ユーザー・データ項目は、*ExitDataLength* で指定される長さのフィールドにあり、フィールド内の右側はブランクで埋められます。各出口ごとに、隣接し合う *MsgExitsDefined* フィールドがあります。定義済みユーザー・データ項目数が、出口名の数より少ない場合、未定義のユーザー・データ項目はブランクに設定されます。逆に、定義済みのユーザー・データ項目数が出口名の数より多い場合、超過したユーザー・データ項目は無視され、出口には渡されません。

出口がこれらの値に対して加えた変更は保存されます。そのため、ある出口が別の出口に情報を渡すことができます。変更について妥当性検査は行われません。例えば、必要であれば 2 進データをこれらのフィールドに書き込むことができます。

SendExitsDefined がゼロの場合は、このフィールドはヌル・ポインターとなります。

プログラミング言語がポインターのデータ・タイプをサポートしていないプラットフォームでは、このフィールドは適切な長さのバイト・ストリングとして宣言されます。

これは、出口に対する入力フィールドです。Version が MQCD_VERSION_4 より小さい場合は、このフィールドは提供されません。

SeqNumberWrap (MQLONG)

このフィールドは、メッセージ・シーケンス番号の最大許容値を指定します。

この値に達すると、シーケンス番号は折り返され、再び 1 から始まります。

この値は折衝不能であり、ローカルおよびリモートの両方のチャンネル定義で一致しなければなりません。

このフィールドは、*ChannelType* として MQCHT_SVRCONN または MQCHT_CLNTCONN が指定されているチャンネルとは関係ありません。

SharingConversations (MQLONG)

このフィールドは、このチャンネルに関連するチャンネル・インスタンスを共有できる会話の最大数を指定します。

このフィールドは、クライアント接続チャンネルとサーバー接続チャンネルに使用されます。

値 0 は、以下の属性に関して、チャンネルが IBM WebSphere MQ 7.0 より前のバージョンと同じように作動することを意味します。

- 会話共有
- 先読み
- STOP CHANNEL(*channelname*) MODE(QUIESCE)

- ハートビート中
- クライアント非同期コンシューム

値 1 は、IBM MQ の動作の最小値です。チャンネルインスタンスでは 1 つの会話のみが許可されますが、先読み、非同期消費、ハートビート動作、および CLNTCONN-SVRCONN 休止状態のチャンネル停止が利用可能です。

これは、出口に対する入力フィールドです。Version が MQCD_VERSION_9 より小さい場合は、提供されません。

このフィールドのデフォルト値は 10 です。

注: チャンネルに適用される *MaxInstances* および *MaxInstancesPerClient* の各制限は、チャンネル・インスタンスの数を制限しますが、それらのインスタンス間で共有できる会話の数を制限するわけではありません。

ShortConnectionName (MQCHAR20)

このフィールドは接続名の最初の 20 バイトを指定します。

Version フィールドが MQCD_VERSION_1 である場合は、*ShortConnectionName* には完全な接続名が入ります。

Version フィールドが MQCD_VERSION_2 より大きい場合は、*ShortConnectionName* には接続名の最初の 20 文字が入ります。完全な接続名は *ConnectionName* フィールドに入っています。

ShortConnectionName と、*ConnectionName* の最初の 20 文字は同じです。

このフィールドの内容の詳細については、『*ConnectionName*』を参照してください。

注: MQCD_VERSION_2 および MQCD 以降のバージョンでは、このフィールドの名前が変更されました。このフィールドは以前は *ConnectionName* という名前でした。

このフィールドの長さは MQ_SHORT_CONN_NAME_LENGTH によって指定されます。

ShortRetryCount (MQLONG)

このフィールドは、リモート・マシンとの接続試行を行う最大回数を指定します。

このフィールドは、(通常、比較的長い) *LongRetryCount* と *LongRetryInterval* が用いられる前に、*ShortRetryInterval* で指定した間隔でリモート・マシンに接続するために行われる試行の最大回数を指定します。

このフィールドは、*ChannelType* として MQCHT_SENDER、MQCHT_SERVER、MQCHT_CLUSSDR、または MQCHT_CLUSRCVR が指定されているチャンネルのみに関連します。

ShortRetryInterval (MQLONG)

このフィールドは、リモート・マシンへの接続を再度試みるまで、最大何秒間待つかを指定します。

チャンネルがアクティブになるのを待機する必要がある場合、再試行間隔が延長されることがあります。

このフィールドは、*ChannelType* として MQCHT_SENDER、MQCHT_SERVER、MQCHT_CLUSSDR、または MQCHT_CLUSRCVR が指定されているチャンネルのみに関連します。

SPLProtection (MQLONG)

このフィールドは、AMS セキュリティー・ポリシー保護の値を指定します。

値は、次のいずれか 1 つです。

MQSPL_PASSTHRU

このチャンネルで MCA が送受信するメッセージを変更なしでパススルーします。

この値は、*ChannelType* として MQCHT_SENDER、MQCHT_SERVER、MQCHT_RECEIVER、MQCHT_REQUESTER が指定されているチャンネルの場合にのみ関連します。これがデフォルト値です。

MQSPL_REMOVE

MCA が伝送キューから受け取ったメッセージの AMS 保護を解除し、そのメッセージをパートナーに送信します。

この値は、*ChannelType* として MQCHT_SENDER または MQCHT_SERVER が指定されているチャンネルのみに関連します。

MQSPL_ASPOLICY

ターゲット・キューに定義されたポリシーに基づいて、インバウンド・メッセージに AMS 保護を適用してからターゲット・キューに書き込まれるようにします。

この値は、*ChannelType* として MQCHT_RECEIVER または MQCHT_REQUESTER が指定されているチャンネルのみに関連します。

これは、出口に対する入力フィールドです。 *Version* が MQCD_VERSION_12 より小さい場合は、このフィールドは提供されません。

SSLCipherSpec (MQCHAR32)

このフィールドは、TLS を使用するとき適用する暗号仕様を指定します。

SSLCipherSpec がブランクの場合、チャンネルは TLS を使用していません。ブランクでない場合は、使用中の CipherSpec を指定するストリングがこのフィールドに入ります。

このパラメーターは、すべてのチャンネル・タイプで有効です。以下のプラットフォームでサポートされています。

- ▶ **AIX** AIX
- ▶ **IBM i** IBM i
- ▶ **Linux** Linux
- ▶ **Windows** Windows
- ▶ **z/OS** z/OS

これは、TCP のトランスポート・タイプ (TRPTYPE) のチャンネル・タイプにのみ有効です。

これは、出口に対する入力フィールドです。このフィールドの長さは MQ_SSL_CIPHER_SPEC_LENGTH によって指定されます。 *Version* が MQCD_VERSION_7 より小さい場合は、このフィールドは提供されません。

SSLClientAuth (MQLONG)

このフィールドは、TLS クライアント認証を必要とするかどうかを指定します。

このフィールドは、SVRCONN チャンネル定義とだけ関係があります。

これは、次の値のいずれかです。

MQSCA_REQUIRED

クライアント認証が必要です。

MQSCA_OPTIONAL

クライアント認証は任意指定です。

これは、出口に対する入力フィールドです。 *Version* が MQCD_VERSION_7 より小さい場合は、このフィールドは提供されません。

SSLPeerNameLength (MQLONG)

このフィールドは、*SSLPeerNamePtr* が指す TLS ピア名の長さ (バイト数) を指定します。

これは、出口に対する入力フィールドです。 *Version* が MQCD_VERSION_7 より小さい場合は、このフィールドは提供されません。

SSLPeerNamePtr (MQPTR)

このフィールドは、TLS ピア名のアドレスを指定します。

正常な TLS ハンドシェイクの間に証明書が受け取られると、証明書を受け取るチャンネルの側で、*SSLPeerNamePtr* によってアクセスされる証明書のサブジェクトの識別名が MQCD フィールドにコピーされます。ローカル・ユーザーのチャンネル定義に *SSLPeerName* 値が指定されている場合、このフィールド

の値がその値を上書きします。セキュリティー出口がチャンネルのこの終端に指定されている場合、セキュリティー出口は MQCD 内のピア証明書から識別名を受け取ります。

これは、出口に対する入力フィールドです。 *Version* が MQCD_VERSION_7 より小さい場合は、このフィールドは提供されません。

注： IBM WebSphere MQ 7.1 より前のリリースで作成されたセキュリティー出口アプリケーションは、更新が必要になる場合があります。 詳しくは、[チャンネル・セキュリティー出口プログラム](#)を参照してください。

StrucLength (MQLONG)

このフィールドは、MQCD 構造体の長さ (バイト数) を指定します。

この長さには、構造体に入っているポインター・フィールドが示すストリングは含まれません。 値は、次のいずれか 1 つです。

MQCD_LENGTH_4

バージョン 4 のチャンネル定義構造体の長さ。

MQCD_LENGTH_5

バージョン 5 のチャンネル定義構造体の長さ。

MQCD_LENGTH_6

バージョン 6 のチャンネル定義構造体の長さ。

MQCD_LENGTH_7

バージョン 7 のチャンネル定義構造体の長さ。

MQCD_LENGTH_8

バージョン 8 のチャンネル定義構造体の長さ。

MQCD_LENGTH_9

バージョン 9 のチャンネル定義構造体の長さ。

MQCD_LENGTH_10

バージョン 10 のチャンネル定義構造体の長さ。

MQCD_LENGTH_11

バージョン 11 のチャンネル定義構造体の長さ。

MQCD_LENGTH_12

バージョン 12 のチャンネル定義構造体の長さ。

以下の定数は、現行バージョンの長さを指定しています。

MQCD_CURRENT_LENGTH

チャンネル定義構造体の現行バージョンの長さ。

注：これらの定数の値は環境に特有です。

Version が MQCD_VERSION_4 より小さい場合は、このフィールドは提供されません。

TpName (MQCHAR64)

このフィールドは、LU 6.2 トランザクション・プログラム名を指定します。

このフィールドは、伝送プロトコル (*TransportType*) が MQXPT_LU62 であり、*ChannelType* が MQCHT_SVRCONN でも MQCHT_RECEIVER でもない場合にのみ適用されます。

通信サイド・オブジェクトの方に情報が格納されるプラットフォームでは、このフィールドは常にブランクとなります。

このフィールドの長さは MQ_TP_NAME_LENGTH によって指定されます。

TransportType (MQLONG)

このフィールドは、使用する伝送プロトコルを指定します。

この値は、チャンネルが相手側から開始された場合にはチェックされません。

これは、次の値のいずれかです。

MQXPT_LU62

LU 6.2 トランスポート・プロトコル。

MQXPT_TCP

TCP/IP トランスポート・プロトコル。

MQXPT_NETBIOS

NetBIOS トランスポート・プロトコル。

この値は、Windows 環境でサポートされています。

MQXPT_SPX

SPX トランスポート・プロトコル。

この値は、Windows、およびこれらのシステムに接続されている IBM MQ クライアントの各環境でサポートされています。

UseDLQ (MQLONG)

このフィールドは、チャンネルでメッセージが配信できない場合に、送達不要キュー (または未配布メッセージ・キュー) を使用するかどうかを指定します。

以下のいずれかの値を取ります。

MQUSEDLQ_NO

チャンネルによって送信できないメッセージは、失敗したものとして扱われます。NPMSPEED の設定に従って、チャンネルがメッセージを破棄するか、チャンネルが終了します。

MQUSEDLQ_YES

DEADQ キュー・マネージャーの属性が送達不能キューの名前を指定している場合は、それが使用されます。指定されていない場合、動作は NO の場合のようになります。YES はデフォルト値です。

UserIdentifier (MQCHAR12)

このフィールドは、メッセージ・チャンネル・エージェントが、リモート・メッセージ・チャンネル・エージェントとの保護 SNA セッションの開始を試みるときに使用するユーザー ID を指定します。

このフィールドは、AIX, Linux, and Windows でのみ非ブランクにでき、*ChannelType* が MQCHT_SENDER、MQCHT_SERVER、MQCHT_REQUESTER、または MQCHT_CLNTCONN のチャンネルにのみ適用されます。z/OS では、このフィールドは適用されません。

このフィールドの長さは MQ_USER_ID_LENGTH によって指定されます。ただし、使用されるのは最初の 10 文字のみです。

Version が MQCD_VERSION_2 より小さい場合は、このフィールドは提供されません。

Version (MQLONG)

Version フィールドは、構造体に対して設定できる最大のバージョン番号を指定します。

値は環境によって異なります。

MQCD _VERSION_1

バージョン 1 チャンネル定義構造体。

MQCD _VERSION_2

バージョン 2 チャンネル定義構造体。

MQCD _VERSION_3

バージョン 3 チャンネル定義構造体。

MQCD _VERSION_4

バージョン 4 チャンネル定義構造体。

MQCD _VERSION_5

バージョン 5 チャンネル定義構造体。

MQCD _VERSION_6

バージョン 6 チャンネル定義構造体。

MQCD _VERSION_7

バージョン 7 チャンネル定義構造体。

MQCD _VERSION_8

バージョン 8 チャンネル定義構造体。

MQCD _VERSION_9

バージョン 9 チャンネル定義構造体。

MQCD _VERSION_10

バージョン 10 チャンネル定義構造体。

MQCD _VERSION_11

バージョン 11 チャンネル定義構造体。

バージョン 11 は、すべてのプラットフォームにおける IBM MQ 8.0 でフィールドに設定できる最大値です。

MQCD _VERSION_12

バージョン 12 チャンネル定義構造体。

バージョン 12 は、IBM MQ 9.1.3 でフィールドに設定できる最大値です。

これより新しいバージョンの構造体にのみ存在するフィールドは、そのフィールドの説明にその旨記載されています。以下の定数は、現行バージョンのバージョン番号を指定しています。

MQCD_CURRENT_VERSION

MQCD_CURRENT_VERSION に設定される値は、使用されているチャンネル定義構造の現行バージョンです。

MQCD_CURRENT_VERSION の値は、環境によって異なります。そこには、プラットフォームによりサポートされる最大値が入ります。

MQCD_CURRENT_VERSION は、異なるプログラミング言語向けのヘッダー、コピー、およびインクルード・ファイルに提供されているデフォルト構造体を初期化するためには使用されません。Version のデフォルトの初期化は、プラットフォームとリリースに応じて異なります。

ヘッダーファイル、コピーファイル、およびインクルードファイル内の宣言 MQCD は、初期化されます MQCD_VERSION_6。追加の MQCD フィールドを使用するには、アプリケーションでバージョン番号を MQCD_CURRENT_VERSION に設定する必要があります。複数の環境で移植可能なアプリケーションを作成する場合は、すべての環境でサポートされるバージョンを選択する必要があります。

ヒント: 新しいバージョンの MQCD 構造体を導入しても、既存部分のレイアウトは変更されません。出口はバージョン番号を確認する必要があります。これは出口が使用する必要があるフィールドが含まれる最低バージョン以上のものでなければなりません。

XmitQName (MQCHAR48)

このフィールドは、メッセージの取得元とする伝送キューの名前を指定します。

このフィールドは、*ChannelType* として MQCHT_SENDER または MQCHT_SERVER が指定されているチャンネルのみに関連します。

このフィールドの長さは MQ_Q_NAME_LENGTH によって指定されます。

C 宣言

以下の宣言は、MQCD 構造体の C 宣言です。

 V1000

```
typedef struct tagMQCD MQCD;
typedef MQCD MQPOINTER PMQCD;
typedef PMQCD MQPOINTER PPMQCD;

struct tagMQCD {
    MQCHAR    ChannelName[20];          /* Channel definition name */
    MQLONG    Version;                  /* Structure version number */
    MQLONG    ChannelType;              /* Channel type */
    MQLONG    TransportType;            /* Transport type */
}
```

MQCHAR	Desc[64];	/* Channel description */
MQCHAR	QMgrName[48];	/* Queue manager name */
MQCHAR	XmitQName[48];	/* Transmission queue name */
MQCHAR	ShortConnectionName[20];	/* First 20 bytes of */
		/* connection name */
MQCHAR	MCAName[20];	/* Reserved */
MQCHAR	ModeName[8];	/* LU 6.2 Mode name */
MQCHAR	TpName[64];	/* LU 6.2 transaction program */
		/* name */
MQLONG	BatchSize;	/* Batch size */
MQLONG	DiscInterval;	/* Disconnect interval */
MQLONG	ShortRetryCount;	/* Short retry count */
MQLONG	ShortRetryInterval;	/* Short retry wait interval */
MQLONG	LongRetryCount;	/* Long retry count */
MQLONG	LongRetryInterval;	/* Long retry wait interval */
MQCHAR	SecurityExit[128];	/* Channel security exit name */
MQCHAR	MsgExit[128];	/* Channel message exit name */
MQCHAR	SendExit[128];	/* Channel send exit name */
MQCHAR	ReceiveExit[128];	/* Channel receive exit name */
MQLONG	SeqNumberWrap;	/* Highest allowable message */
		/* sequence number */
MQLONG	MaxMsgLength;	/* Maximum message length */
MQLONG	PutAuthority;	/* Put authority */
MQLONG	DataConversion;	/* Data conversion */
MQCHAR	SecurityUserData[32];	/* Channel security exit user */
		/* data */
MQCHAR	MsgUserData[32];	/* Channel message exit user */
		/* data */
MQCHAR	SendUserData[32];	/* Channel send exit user */
		/* data */
MQCHAR	ReceiveUserData[32];	/* Channel receive exit user */
		/* data */
/* Ver:1 */		
MQCHAR	UserIdentifier[12];	/* User identifier */
MQCHAR	Password[12];	/* Password */
MQCHAR	MCAUserIdentifier[12];	/* First 12 bytes of MCA user */
		/* identifier */
MQLONG	MCAType;	/* Message channel agent type */
MQCHAR	ConnectionName[264];	/* Connection name */
MQCHAR	RemoteUserIdentifier[12];	/* First 12 bytes of user */
		/* identifier from partner */
MQCHAR	RemotePassword[12];	/* Password from partner */
/* Ver:2 */		
MQCHAR	MsgRetryExit[128];	/* Channel message retry exit */
		/* name */
MQCHAR	MsgRetryUserData[32];	/* Channel message retry exit */
		/* user data */
MQLONG	MsgRetryCount;	/* Number of times MCA will */
		/* try to put the message, */
		/* after first attempt has */
		/* failed */
MQLONG	MsgRetryInterval;	/* Minimum interval in */
		/* milliseconds after which */
		/* the open or put operation */
		/* will be retried */
/* Ver:3 */		
MQLONG	HeartbeatInterval;	/* Time in seconds between */
		/* heartbeat flows */
MQLONG	BatchInterval;	/* Batch duration */
MQLONG	NonPersistentMsgSpeed;	/* Speed at which */
		/* nonpersistent messages are */
		/* sent */
MQLONG	StrucLength;	/* Length of MQCD structure */
MQLONG	ExitNameLength;	/* Length of exit name */
MQLONG	ExitDataLength;	/* Length of exit user data */
MQLONG	MsgExitsDefined;	/* Number of message exits */
		/* defined */
MQLONG	SendExitsDefined;	/* Number of send exits */
		/* defined */
MQLONG	ReceiveExitsDefined;	/* Number of receive exits */
		/* defined */
MQPTR	MsgExitPtr;	/* Address of first MsgExit */
		/* field */
MQPTR	MsgUserDataPtr;	/* Address of first */
		/* MsgUserData field */
MQPTR	SendExitPtr;	/* Address of first SendExit */
		/* field */
MQPTR	SendUserDataPtr;	/* Address of first */
		/* SendUserData field */
MQPTR	ReceiveExitPtr;	/* Address of first */
		/* ReceiveExit field */
MQPTR	ReceiveUserDataPtr;	/* Address of first */

```

/* Ver:4 */
MQPTR      ClusterPtr;
/* ReceiveUserData field */
/* Address of a list of */
/* cluster names */
/* Number of clusters to */
/* which the channel belongs */
/* Network priority */
MQLONG      ClustersDefined;
/* Ver:5 */
MQLONG      NetworkPriority;
/* Ver:5 */
MQLONG      LongMCAUserIdLength;
/* Length of long MCA user */
/* identifier */
MQLONG      LongRemoteUserIdLength;
/* Length of long remote user */
/* identifier */
MQPTR      LongMCAUserIdPtr;
/* Address of long MCA user */
/* identifier */
MQPTR      LongRemoteUserIdPtr;
/* Address of long remote */
/* user identifier */
MQBYTE40    MCASecurityId;
/* MCA security identifier */
MQBYTE40    RemoteSecurityId;
/* Remote security identifier */
/* Ver:6 */
MQCHAR      SSLCipherSpec[32];
/* TLS CipherSpec */
MQPTR      SSLPeerNamePtr;
/* Address of TLS peer name */
MQLONG      SSLPeerNameLength;
/* Length of TLS peer name */
MQLONG      SSLClientAuth;
/* Whether TLS client */
/* authentication is required */
MQLONG      KeepAliveInterval;
/* Keepalive interval */
MQCHAR      LocalAddress[48];
/* Local communications */
/* address */
MQLONG      BatchHeartbeat;
/* Batch heartbeat interval */
/* Ver:7 */
MQLONG      HdrCompList[2];
/* Header data compression */
/* list */
MQLONG      MsgCompList[16];
/* Message data compression */
/* list */
MQLONG      CLWLChannelRank;
/* Channel rank */
MQLONG      CLWLChannelPriority;
/* Channel priority */
MQLONG      CLWLChannelWeight;
/* Channel weight */
MQLONG      ChannelMonitoring;
/* Channel monitoring */
MQLONG      ChannelStatistics;
/* Channel statistics */
/* Ver:8 */
MQLONG      SharingConversations;
/* Limit on sharing */
/* conversations */
MQLONG      PropertyControl;
/* Message property control */
MQLONG      MaxInstances;
/* Limit on SVRCONN channel */
/* instances */
MQLONG      MaxInstancesPerClient;
/* Limit on SVRCONN channel */
/* instances per client */
MQLONG      ClientChannelWeight;
/* Client channel weight */
MQLONG      ConnectionAffinity;
/* Connection affinity */
/* Ver:9 */
MQLONG      BatchDataLimit;
/* Batch data limit */
MQLONG      UseDLQ;
/* Use Dead Letter Queue */
MQLONG      DefReconnect;
/* Default client reconnect */
/* option */
/* Ver:10 */
MQCHAR64    CertificateLabel;
/* Certificate label */
/* Ver:11 */
MQLONG      SPLProtection
/* AMS Security policy protection */
/* Ver:12 */
MQLONG      QuantumSafeAlgorithm;
/* Quantum Cryptography */
/* Algorithm */
MQLONG      QuantumSafeRequired;
/* Quantum Cryptography */
/* Requirement */
/* Ver:13 */
};

```

COBOL 宣言

以下の宣言は、MQCD 構造体の COBOL 宣言です。

```

** MQCD structure
  10 MQCD.
    ** Channel definition name
      15 MQCD-CHANNELNAME PIC X(20).
    ** Structure version number
      15 MQCD-VERSION PIC S9(9) BINARY.
    ** Channel type
      15 MQCD-CHANNELTYPE PIC S9(9) BINARY.
    ** Transport type
      15 MQCD-TRANSPORTTYPE PIC S9(9) BINARY.

```

```

** Channel description
  15 MQCD-DESC PIC X(64).
** Queue manager name
  15 MQCD-QMGRNAME PIC X(48).
** Transmission queue name
  15 MQCD-XMITQNAME PIC X(48).
** First 20 bytes of connection name
  15 MQCD-SHORTCONNECTIONNAME PIC X(20).
** Reserved
  15 MQCD-MCANAME PIC X(20).
** LU 6.2 Mode name
  15 MQCD-MODENAME PIC X(8).
** LU 6.2 transaction program name
  15 MQCD-TPNAME PIC X(64).
** Batch size
  15 MQCD-BATCHSIZE PIC S9(9) BINARY.
** Disconnect interval
  15 MQCD-DISCINTERVAL PIC S9(9) BINARY.
** Short retry count
  15 MQCD-SHORTRETRYCOUNT PIC S9(9) BINARY.
** Short retry wait interval
  15 MQCD-SHORTRETRYINTERVAL PIC S9(9) BINARY.
** Long retry count
  15 MQCD-LONGRETRYCOUNT PIC S9(9) BINARY.
** Long retry wait interval
  15 MQCD-LONGRETRYINTERVAL PIC S9(9) BINARY.
** Channel security exit name
  15 MQCD-SECURITYEXIT PIC X(20).
** Channel message exit name
  15 MQCD-MSGEXIT PIC X(20).
** Channel send exit name
  15 MQCD-SENDEXIT PIC X(20).
** Channel receive exit name
  15 MQCD-RECEIVEEXIT PIC X(20).
** Highest allowable message sequence number
  15 MQCD-SEQNUMBERWRAP PIC S9(9) BINARY.
** Maximum message length
  15 MQCD-MAXMSGLENGTH PIC S9(9) BINARY.
** Put authority
  15 MQCD-PUTAUTHORITY PIC S9(9) BINARY.
** Data conversion
  15 MQCD-DATACONVERSION PIC S9(9) BINARY.
** Channel security exit user data
  15 MQCD-SECURITYUSERDATA PIC X(32).
** Channel message exit user data
  15 MQCD-MSGUSERDATA PIC X(32).
** Channel send exit user data
  15 MQCD-SENDUSERDATA PIC X(32).
** Channel receive exit user data
  15 MQCD-RECEIVEUSERDATA PIC X(32).
** Ver:1 **
** User identifier
  15 MQCD-USERIDENTIFIER PIC X(12).
** Password
  15 MQCD-PASSWORD PIC X(12).
** First 12 bytes of MCA user identifier
  15 MQCD-MCAUSERIDENTIFIER PIC X(12).
** Message channel agent type
  15 MQCD-MCATYPE PIC S9(9) BINARY.
** Connection name
  15 MQCD-CONNECTIONNAME PIC X(264).
** First 12 bytes of user identifier from partner
  15 MQCD-REMOTEUSERIDENTIFIER PIC X(12).
** Password from partner
  15 MQCD-REMOTEPASSWORD PIC X(12).
** Ver:2 **
** Channel message retry exit name
  15 MQCD-MSGRETRYEXIT PIC X(20).
** Channel message retry exit user data
  15 MQCD-MSGRETRYUSERDATA PIC X(32).
** Number of times MCA will try to put the message, after first
** attempt has failed
  15 MQCD-MSGRETRYCOUNT PIC S9(9) BINARY.
** Minimum interval in milliseconds after which the open or put
** operation will be retried
  15 MQCD-MSGRETRYINTERVAL PIC S9(9) BINARY.
** Ver:3 **
** Time in seconds between heartbeat flows
  15 MQCD-HEARTBEATINTERVAL PIC S9(9) BINARY.
** Batch duration
  15 MQCD-BATCHINTERVAL PIC S9(9) BINARY.
** Speed at which nonpersistent messages are sent

```

```

15 MQCD-NONPERSISTENTMSGSPD PIC S9(9) BINARY.
** Length of MQCD structure
15 MQCD-STRUCLNGTH PIC S9(9) BINARY.
** Length of exit name
15 MQCD-EXITNAMELENGTH PIC S9(9) BINARY.
** Length of exit user data
15 MQCD-EXITDATALENGTH PIC S9(9) BINARY.
** Number of message exits defined
15 MQCD-MSGEXITSDEFINED PIC S9(9) BINARY.
** Number of send exits defined
15 MQCD-SENDEXITSDEFINED PIC S9(9) BINARY.
** Number of receive exits defined
15 MQCD-RECEIVEEXITSDEFINED PIC S9(9) BINARY.
** Address of first MsgExit field
15 MQCD-MSGEXITPTR POINTER.
** Address of first MsgUserData field
15 MQCD-MSGUSERDATAPTR POINTER.
** Address of first SendExit field
15 MQCD-SENDEXITPTR POINTER.
** Address of first SendUserData field
15 MQCD-SENDUSERDATAPTR POINTER.
** Address of first ReceiveExit field
15 MQCD-RECEIVEEXITPTR POINTER.
** Address of first ReceiveUserData field
15 MQCD-RECEIVEUSERDATAPTR POINTER.
** Ver:4 **
** Address of a list of cluster names
15 MQCD-CLUSTERPTR POINTER.
** Number of clusters to which the channel belongs
15 MQCD-CLUSTERSDEFINED PIC S9(9) BINARY.
** Network priority
15 MQCD-NETWORKPRIORITY PIC S9(9) BINARY.
** Ver:5 **
** Length of long MCA user identifier
15 MQCD-LONGMCAUSERIDLENGTH PIC S9(9) BINARY.
** Length of long remote user identifier
15 MQCD-LONGREMOTEUSERIDLENGTH PIC S9(9) BINARY.
** Address of long MCA user identifier
15 MQCD-LONGMCAUSERIDPTR POINTER.
** Address of long remote user identifier
15 MQCD-LONGREMOTEUSERIDPTR POINTER.
** MCA security identifier
15 MQCD-MCASECURITYID PIC X(40).
** Remote security identifier
15 MQCD-REMOTESECURITYID PIC X(40).
** Ver:6 **
** TLS CipherSpec
15 MQCD-SSLCIPHERSPEC PIC X(32).
** Address of TLS peer name
15 MQCD-SSLPEERNAMEPTR POINTER.
** Length of TLS peer name
15 MQCD-SSLPEERNAMELENGTH PIC S9(9) BINARY.
** Whether TLS client authentication is required
15 MQCD-SSLCLIENTAUTH PIC S9(9) BINARY.
** Keepalive interval
15 MQCD-KEEPALIVEINTERVAL PIC S9(9) BINARY.
** Local communications address
15 MQCD-LOCALADDRESS PIC X(48).
** Batch heartbeat interval
15 MQCD-BATCHHEARTBEAT PIC S9(9) BINARY.
** Ver:7 **
** Header data compression list
15 MQCD-HDRCOMPLIST PIC S9(9) BINARY.
** Message data compression list
15 MQCD-MSGCOMPLIST PIC S9(9) BINARY.
** Channel rank
15 MQCD-CLWLCHANNELRANK PIC S9(9) BINARY.
** Channel priority
15 MQCD-CLWLCHANNELPRIORITY PIC S9(9) BINARY.
** Channel weight
15 MQCD-CLWLCHANNELWEIGHT PIC S9(9) BINARY.
** Channel monitoring
15 MQCD-CHANNELMONITORING PIC S9(9) BINARY.
** Channel statistics
15 MQCD-CHANNELSTATISTICS PIC S9(9) BINARY.
** Ver:8 **
** Limit on sharing conversations
15 MQCD-SHARINGCONVERSATIONS PIC S9(9) BINARY.
** Message property control
15 MQCD-PROPERTYCONTROL PIC S9(9) BINARY.
** Limit on SVRCONN channel instances
15 MQCD-MAXINSTANCES PIC S9(9) BINARY.

```

```

** Limit on SVRCONN channel instances per client
15 MQCD-MAXINSTANCESPERCLIENT PIC S9(9) BINARY.
** Client channel weight
15 MQCD-CLIENTCHANNELWEIGHT PIC S9(9) BINARY.
** Connection affinity
15 MQCD-CONNECTIONAFFINITY PIC S9(9) BINARY.
** Ver:9 **
** Batch data limit
15 MQCD-BATCHDATA LIMIT PIC S9(9) BINARY.
** Use Dead Letter Queue
15 MQCD-USEDLQ PIC S9(9) BINARY.
** Default client reconnect option
15 MQCD-DEFRECONNECT PIC S9(9) BINARY.
** Ver:10 **
** Certificate Label
15 MQCD-CERTLABEL PIC X (64)
** Ver:11 **
** AMS Security policy protection
15 MQCD-SPLPROTECTION PIC S9(9) BINARY
** Ver:12 **

```

RPG 宣言 (ILE)

以下の宣言は、MQCD 構造体の RPG 宣言です。

```

D* MQCD Structure
D*
D* Channel definition name
D CDCHN 1 20
D* Structure version number
D CDVER 21 24I 0
D* Channel type
D CDCHT 25 28I 0
D* Transport type
D CDTRT 29 32I 0
D* Channel description
D CDDDES 33 96
D* Queue manager name
D CDQM 97 144
D* Transmission queue name
D CDXQ 145 192
D* First 20 bytes of connection name
D CDSCN 193 212
D* Reserved
D CDMCA 213 232
D* LU 6.2 Mode name
D CDMOD 233 240
D* LU 6.2 transaction program name
D CDTP 241 304
D* Batch size
D CDBS 305 308I 0
D* Disconnect interval
D CDDI 309 312I 0
D* Short retry count
D CDSRC 313 316I 0
D* Short retry wait interval
D CDSRI 317 320I 0
D* Long retry count
D CDLRC 321 324I 0
D* Long retry wait interval
D CDLRI 325 328I 0
D* Channel security exit name
D CDSCX 329 348
D* Channel message exit name
D CDMSX 349 368
D* Channel send exit name
D CDSNX 369 388
D* Channel receive exit name
D CDRCX 389 408
D* Highest allowable message sequence number
D CDSNW 409 412I 0
D* Maximum message length
D CDMML 413 416I 0
D* Put authority
D CDPA 417 420I 0
D* Data conversion
D CDDC 421 424I 0
D* Channel security exit user data
D CDSCD 425 456
D* Channel message exit user data

```

```

D CDMSD 457 488
D* Channel send exit user data
D CDSND 489 520
D* Channel receive exit user data
D CDRCD 521 552
D* Ver:1 **
D* User identifier
D CDUID 553 564
D* Password
D CDPW 565 576
D* First 12 bytes of MCA user identifier
D CDAUI 577 588
D* Message channel agent type
D CDCAT 589 592I 0
D* Connection name
D CDCON 593 848
D CDCN2 849 856
D* First 12 bytes of user identifier from partner
D CDRUI 857 868
D* Password from partner
D CDRPW 869 880
D* Ver:2 **
D* Channel message retry exit name
D CDMRX 881 900
D* Channel message retry exit user data
D CDMRD 901 932
D* Number of times MCA will try to put the message, after first
D* attempt has failed
D CDMRC 933 936I 0
D* Minimum interval in milliseconds after which the open or put
D* operation will be retried
D CDMRI 937 940I 0
D* Ver:3 **
D* Time in seconds between heartbeat flows
D CDHBI 941 944I 0
D* Batch duration
D CDBI 945 948I 0
D* Speed at which nonpersistent messages are sent
D CDNPM 949 952I 0
D* Length of MQCD structure
D CDLEN 953 956I 0
D* Length of exit name
D CDXNL 957 960I 0
D* Length of exit user data
D CDXDL 961 964I 0
D* Number of message exits defined
D CDMXD 965 968I 0
D* Number of send exits defined
D CDSXD 969 972I 0
D* Number of receive exits defined
D CDRXD 973 976I 0
D* Address of first MsgExit field
D CDMXP 977 992*
D* Address of first MsgUserData field
D CDMUP 993 1008*
D* Address of first SendExit field
D CDSXP 1009 1024*
D* Address of first SendUserData field
D CDSUP 1025 1040*
D* Address of first ReceiveExit field
D CDRXP 1041 1056*
D* Address of first ReceiveUserData field
D CDRUP 1057 1072*
D* Ver:4 **
D* Address of a list of cluster names
D CDCLP 1073 1088*
D* Number of clusters to which the channel belongs
D CDCLD 1089 1092I 0
D* Network priority
D CDNP 1093 1096I 0
D* Ver:5 **
D* Length of long MCA user identifier
D CDLML 1097 1100I 0
D* Length of long remote user identifier
D CDLRL 1101 1104I 0
D* Address of long MCA user identifier
D CDLMP 1105 1120*
D* Address of long remote user identifier
D CDLRP 1121 1136*
D* MCA security identifier
D CDMSI 1137 1176
D* Remote security identifier

```

```

D CDRSI 1177 1216
D* Ver:6 **
D* TLS CipherSpec
D CDSCS 1217 1248
D* Address of TLS peer name
D CDSPN 1249 1264*
D* Length of TLS peer name
D CDSPL 1265 1268I 0
D* Whether TLS client authentication is required
D CDSCA 1269 1272I 0
D* Keepalive interval
D CDKAI 1273 1276I 0
D* Local communications address
D CDLOA 1277 1324
D* Batch heartbeat interval
D CDBHB 1325 1328I 0
D* Ver:7 **
D* Header data compression list
D CDHCL0
D CDHCL1 1329 1332I 0
D CDHCL2 1333 1336I 0
D CDHCL 10I 0 DIM(2) OVERLAY(CDHCL0)
D* Message data compression list
D CDMCL0
D CDMCL1 1337 1340I 0
D CDMCL2 1341 1344I 0
D CDMCL3 1345 1348I 0
D CDMCL4 1349 1352I 0
D CDMCL5 1353 1356I 0
D CDMCL6 1357 1360I 0
D CDMCL7 1361 1364I 0
D CDMCL8 1365 1368I 0
D CDMCL9 1369 1372I 0
D CDMCL10 1373 1376I 0
D CDMCL11 1377 1380I 0
D CDMCL12 1381 1384I 0
D CDMCL13 1385 1388I 0
D CDMCL14 1389 1392I 0
D CDMCL15 1393 1396I 0
D CDMCL16 1397 1400I 0
D CDMCL 10I 0 DIM(16) OVERLAY(CDMCL0)
D* Channel rank
D CDCWCR 1401 1404I 0
D* Channel priority
D CDCWCP 1405 1408I 0
D* Channel weight
D CDCWCW 1409 1412I 0
D* Channel monitoring
D CDCHLMON 1413 1416I 0
D* Channel statistics
D CDCHLST 1417 1420I 0
D* Ver:8 **
D* Limit on sharing conversations
D CDSHC 1421 1424I 0
D* Message property control
D CDPRC 1425 1428I 0
D* Limit on SVRCONN channel instances
D CDMXIN 1429 1432I 0
D* Limit on SVRCONN channel instances per client
D CDMXIC 1433 1436I 0
D* Client channel weight
D CDCLNCHLW 1437 1440I 0
D* Connection affinity
D CDCONNAFF 1441 1444I 0
D* Ver:9 **
D* Batch data limit
D CDBDL 1445 1448I 0
D* Use Dead Letter Queue
D CDUDLQ 1449 1452I 0
D* Default client reconnect option
D CDDRCN 1453 1456I 0
D* Ver:10 **

```

System/390 アセンブラー宣言

以下の宣言は、MQCD 構造体の System/390 アセンブラー宣言です。

MQCD	DSECT		
MQCD_CHANNELNAME	DS	CL20	Channel definition name
MQCD_VERSION	DS	F	Structure version number

MQCD_CHANNELTYPE	DS	F	Channel type
MQCD_TRANSPORTTYPE	DS	F	Transport type
MQCD_DESC	DS	CL64	Channel description
MQCD_QMGRNAME	DS	CL48	Queue manager name
MQCD_XMITQNAME	DS	CL48	Transmission queue name
MQCD_SHORTCONNECTIONNAME	DS	CL20	First 20 bytes of connection name
*			
MQCD_MCANAME	DS	CL20	Reserved
MQCD_MODENAME	DS	CL8	LU 6.2 Mode name
MQCD_TPNAME	DS	CL64	LU 6.2 transaction program name
MQCD_BATCHSIZE	DS	F	Batch size
MQCD_DISCINTERVAL	DS	F	Disconnect interval
MQCD_SHORTRETRYCOUNT	DS	F	Short retry count
MQCD_SHORTRETRYINTERVAL	DS	F	Short retry wait interval
MQCD_LONGRETRYCOUNT	DS	F	Long retry count
MQCD_LONGRETRYINTERVAL	DS	F	Long retry wait interval
MQCD_SECURITYEXIT	DS	CLn	Channel security exit name
MQCD_MSGEXIT	DS	CLn	Channel message exit name
MQCD_SENDEXIT	DS	CLn	Channel send exit name
MQCD_RECEIVEEXIT	DS	CLn	Channel receive exit name
MQCD_SEQNUMBERWRAP	DS	F	Highest allowable message sequence number
*			
MQCD_MAXMSGLLENGTH	DS	F	Maximum message length
MQCD_PUTAUTHORITY	DS	F	Put authority
MQCD_DATACONVERSION	DS	F	Data conversion
MQCD_SECURITYUSERDATA	DS	CL32	Channel security exit user data
MQCD_MSGUSERDATA	DS	CL32	Channel message exit user data
MQCD_SENDUSERDATA	DS	CL32	Channel send exit user data
MQCD_RECEIVEUSERDATA	DS	CL32	Channel receive exit user data
MQCD_USERIDENTIFIER	DS	CL12	User identifier
MQCD_PASSWORD	DS	CL12	Password
MQCD_MCAUSERIDENTIFIER	DS	CL12	First 12 bytes of MCA user identifier
*			
MQCD_MCATYPE	DS	F	Message channel agent type
MQCD_CONNECTIONNAME	DS	CL264	Connection name
MQCD_REMOTEUSERIDENTIFIER	DS	CL12	First 12 bytes of user identifier from partner
*			
MQCD_REMOTEPASSWORD	DS	CL12	Password from partner
MQCD_MSGRETRYEXIT	DS	CLn	Channel message retry exit name
MQCD_MSGRETRYUSERDATA	DS	CL32	Channel message retry exit user data
*			
MQCD_MSGRETRYCOUNT	DS	F	Number of times MCA will try to put the message, after the first attempt has failed
*			
MQCD_MSGRETRYINTERVAL	DS	F	Minimum interval in milliseconds after which the open or put operation will be retried
*			
MQCD_HEARTBEATINTERVAL	DS	F	Time in seconds between heartbeat flows
*			
MQCD_BATCHINTERVAL	DS	F	Batch duration
MQCD_NONPERSISTENTMSGSPD	DS	F	Speed at which nonpersistent messages are sent
*			
MQCD_STRUCLLENGTH	DS	F	Length of MQCD structure
MQCD_EXITNAMELENGTH	DS	F	Length of exit name
MQCD_EXITDATALENGTH	DS	F	Length of exit user data
MQCD_MSGEXITSDEFINED	DS	F	Number of message exits defined
MQCD_SENDEXITSDEFINED	DS	F	Number of send exits defined
MQCD_RECEIVEEXITSDEFINED	DS	F	Number of receive exits defined
MQCD_MSGEXITPTR	DS	F	Address of first MSGEXIT field
MQCD_MSGUSERDATAPTR	DS	F	Address of first MSGUSERDATA field
*			
MQCD_SENDEXITPTR	DS	F	Address of first SENDEXIT field
MQCD_SENDUSERDATAPTR	DS	F	Address of first SENDUSERDATA field
*			
MQCD_RECEIVEEXITPTR	DS	F	Address of first RECEIVEEXIT field
*			
MQCD_RECEIVEUSERDATAPTR	DS	F	Address of first RECEIVEUSERDATA field
*			
MQCD_CLUSTERPTR	DS	F	Address of a list of cluster names
*			
MQCD_CLUSTERSDEFINED	DS	F	Number of clusters to which the channel belongs
*			
MQCD_NETWORKPRIORITY	DS	F	Network priority
MQCD_LONGMCAUSERIDLENGTH	DS	F	Length of long MCA user identifier
*			
MQCD_LONGREMOTEUSERIDLENGTH	DS	F	Length of long remote user identifier
*			
MQCD_LONGMCAUSERIDPTR	DS	F	Address of long MCA user identifier
*			
MQCD_LONGREMOTEUSERIDPTR	DS	F	Address of long remote user identifier
*			

MQCD_MCASESECURITYID	DS	XL40	MCA security identifier
MQCD_REMOTESECURITYID	DS	XL40	Remote security identifier
MQCD_SSLCIPHERSPEC	DS	CL32	TLS CipherSpec
MQCD_SSLPEERNAMEPTR	DS	F	Address of TLS peer name
MQCD_SSLPEERNAMELENGTH	DS	F	Length of TLS peer name
MQCD_SSLCLIENTAUTH	DS	F	Whether TLS client authentication is required
* MQCD_KEEPAIVEINTERVAL	DS	F	Keepalive interval
MQCD_LOCALADDRESS	DS	CL48	Local communications address
MQCD_BATCHHEARTBEAT	DS	F	Batch heartbeat interval
MQCD_HDRCOMPLIST	DS	CL2	Header data compression list
MQCD_MSGCOMPLIST	DS	CL16	Message data compression list
MQCD_CLWLCHANNELRANK	DS	F	Channel rank
MQCD_CLWLCHANNELPRIORITY	DS	F	Channel priority
MQCD_CLWLCHANNELWEIGHT	DS	F	Channel weight
MQCD_CHANNELMONITORING	DS	F	Channel monitoring
MQCD_CHANNELSTATISTICS	DS	F	Channel statistics
MQCD_SHARINGCONVERSATIONS	DS	F	Limit on sharing conversations
* MQCD_PROPERTYCONTROL	DS	F	Message property control
* MQCD_SHARINGCONVERSATIONS	DS	F	Limit on sharing conversations
MQCD_PROPERTYCONTROL	DS	F	Message property control
MQCD_MAXINSTANCES	DS	F	Limit on SVRCONN chl instances
MQCD_MAXINSTANCESPERCLIENT	DS	F	Limit on SVRCONN chl instances per client
MQCD_CLIENTCHANNELWEIGHT	DS	F	Channel weight
MQCD_CONNECTIONAFFINITY	DS	F	Connection Affinty
MQCD_BATCHDATALIMIT	DS	F	Batch data limit
MQCD_USEDLQ	DS	F	Use dead-letter queue
MQCD_DEFRECONNECT	DS	F	Default client reconnect option
MQCD_CERTLABL	DS	F	Certificate label
MQCD_SPLPROTECTION	DS	F	AMS Security policy protection
MQCD_LENGTH	EQU	*-MQCD	
	ORG	MQCD	
MQCD_AREA	DS	CL(MQCD_LENGTH)	

Visual Basic の宣言

以下の宣言は、MQCD 構造体の Visual Basic 宣言です。

Visual Basic では、MQCONN 呼び出しで、MQCD 構造体を MQCNO 構造体と共に使用することができます。

Type MQCD		
ChannelName	As String*20	'Channel definition name'
Version	As Long	'Structure version number'
ChannelType	As Long	'Channel type'
TransportType	As Long	'Transport type'
Desc	As String*64	'Channel description'
QMgrName	As String*48	'Queue manager name'
XmitQName	As String*48	'Transmission queue name'
ShortConnectionName	As String*20	'First 20 bytes of connection name'
MCAName	As String*20	'Reserved'
ModeName	As String*8	'LU 6.2 Mode name'
TpName	As String*64	'LU 6.2 transaction program name'
BatchSize	As Long	'Batch size'
DiscInterval	As Long	'Disconnect interval'
ShortRetryCount	As Long	'Short retry count'
ShortRetryInterval	As Long	'Short retry wait interval'
LongRetryCount	As Long	'Long retry count'
LongRetryInterval	As Long	'Long retry wait interval'
SecurityExit	As String*128	'Channel security exit name'
MsgExit	As String*128	'Channel message exit name'
SendExit	As String*128	'Channel send exit name'
ReceiveExit	As String*128	'Channel receive exit name'
SeqNumberWrap	As Long	'Highest allowable message sequence number'
MaxMsgLength	As Long	'Maximum message length'
PutAuthority	As Long	'Put authority'
DataConversion	As Long	'Data conversion'
SecurityUserData	As String*32	'Channel security exit user data'
MsgUserData	As String*32	'Channel message exit user data'
SendUserData	As String*32	'Channel send exit user data'
ReceiveUserData	As String*32	'Channel receive exit user data'
UserIdentifier	As String*12	'User identifier'
Password	As String*12	'Password'
MCAUserIdentifier	As String*12	'First 12 bytes of MCA user identifier'

MCAType	As Long	'Message channel agent type'
ConnectionName	As String*264	'Connection name'
RemoteUserIdentifier	As String*12	'First 12 bytes of user identifier from partner'
RemotePassword	As String*12	'Password from partner'
MsgRetryExit	As String*128	'Channel message retry exit name'
MsgRetryUserData	As String*32	'Channel message retry exit user data'
MsgRetryCount	As Long	'Number of times MCA will try to put the message, after the first attempt has failed'
MsgRetryInterval	As Long	'Minimum interval in milliseconds after which the open or put operation will be retried'
HeartbeatInterval	As Long	'Time in seconds between heartbeat flows'
BatchInterval	As Long	'Batch duration'
NonPersistentMsgSpeed	As Long	'Speed at which nonpersistent messages are sent'
StrucLength	As Long	'Length of MQCD structure'
ExitNameLength	As Long	'Length of exit name'
ExitDataLength	As Long	'Length of exit user data'
MsgExitsDefined	As Long	'Number of message exits defined'
SendExitsDefined	As Long	'Number of send exits defined'
ReceiveExitsDefined	As Long	'Number of receive exits defined'
MsgExitPtr	As MQPTR	'Address of first MsgExit field'
MsgUserDataPtr	As MQPTR	'Address of first MsgUserData field'
SendExitPtr	As MQPTR	'Address of first SendExit field'
SendUserDataPtr	As MQPTR	'Address of first SendUserData field'
ReceiveExitPtr	As MQPTR	'Address of first ReceiveExit field'
ReceiveUserDataPtr	As MQPTR	'Address of first ReceiveUserData field'
ClusterPtr	As MQPTR	'Address of a list of cluster names'
ClustersDefined	As Long	'Number of clusters to which the channel belongs'
NetworkPriority	As Long	'Network priority'
LongMCAUserIdLength	As Long	'Length of long MCA user identifier'
LongRemoteUserIdLength	As Long	'Length of long remote user identifier'
LongMCAUserIdPtr	As MQPTR	'Address of long MCA user identifier'
LongRemoteUserIdPtr	As MQPTR	'Address of long remote user identifier'
MCASecurityId	As MQBYTE40	'MCA security identifier'
RemoteSecurityId	As MQBYTE40	'Remote security identifier'
SSLCipherSpec	As String*32	'TLS CipherSpec'
SSLPeerNamePtr	As MQPTR	'Address of TLS peer name'
SSLPeerNameLength	As Long	'Length of TLS peer name'
SSLClientAuth	As Long	'Whether TLS client authentication is required'
KeepAliveInterval	As Long	'Keepalive interval'
LocalAddress	As String*48	'Local communications address'
BatchHeartbeat	As Long	'Batch heartbeat interval'
HdrCompList(0 to 1)	As Long2	'Header data compression list'
MsgCompList(0 To 15)	As Long16	'Message data compression list'
CLWLChannelRank	As Long	'Channel Rank'
CLWLChannelPriority	As Long	'Channel priority'
CLWLChannelWeight	As Long	'Channel Weight'
ChannelMonitoring	As Long	'Channel Monitoring control'
ChannelStatistics	As Long	'Channel Statistics'
End Type		

チャネル出口での MQCD フィールドの変更

チャネル出口は、MQCD のフィールドを変更できます。ただし、リストされている状況を除いて、通常はこれらの変更に応じて動作することはありません。

チャネル出口プログラムが MQCD データ構造体のフィールドを変更する場合、新規の値は通常 IBM MQ チャネル・プロセスにより無視されます。しかし新規の値は MQCD に残り、出口チェーンの残りの出口、およびチャネル・インスタンスを共用する会話に渡されます。

SharingConversations が MQCXP 構造体で FALSE に設定される場合、特定のフィールドへの変更は、出口プログラムのタイプ、チャネルのタイプ、および出口理由コード次第で、それに応じた動作をすることが

あります。次の表には、変更可能でチャンネルの動作に影響を与える可能性のあるフィールド、およびどのような状況でそれが可能かが示されています。 出口プログラムが他の状況でこれらのフィールドのいずれかを変更する場合、またはリストされていないフィールドを変更する場合、新規の値はチャンネル・プロセスにより無視されます。 新規の値は MQCD に残り、出口チェーンの残りの出口、およびチャンネル・インスタンスを共有する会話に渡されます。

初期化 (MQXR_INIT) 用に呼び出されるすべてのタイプの出口プログラムは、MQCXP SharingConverstions が FALSE に設定されている限り、どのタイプのチャンネルの ChannelName フィールドをも変更できます。 セキュリティー出口だけが、MQCXP SharingConverstions の値にかかわらず MCAUserIdentifier フィールドを変更できます。

表 859. 変更可能でチャンネルの動作に影響を与える可能性のあるフィールド			
フィールド	出口理由コード	出口タイプ	チャンネル・タイプ
ChannelName	MQXR_INIT	すべて	すべて
TransportType	MQXR_INIT	すべて	すべて
XmitQName	MQXR_INIT	すべて	SDR, RCVR
ModeName	MQXR_INIT	すべて	すべて
TpName	MQXR_INIT	すべて	すべて
BatchSize	MQXR_INIT	すべて	SDR、SVR、RCVR、RQSTR、CLUSSDR、CLUSRCVR
DiscInterval	MQXR_INIT	すべて	SDR、SVR、RCVR、RQSTR、CLUSSDR、CLUSRCVR
ShortRetryCount	MQXR_INIT	すべて	SDR、SVR、RCVR、RQSTR、CLUSSDR、CLUSRCVR
ShortRetryInterval	MQXR_INIT	すべて	SDR、SVR、RCVR、RQSTR、CLUSSDR、CLUSRCVR
LongRetryCount	MQXR_INIT	すべて	SDR、SVR、RCVR、RQSTR、CLUSSDR、CLUSRCVR
LongRetryInterval	MQXR_INIT	すべて	SDR、SVR、RCVR、RQSTR、CLUSSDR、CLUSRCVR

表 859. 変更可能でチャンネルの動作に影響を与える可能性のあるフィールド (続き)

フィールド	出口理由コード	出口タイプ	チャンネル・タイプ
SeqNumberWrap	MQXR_INIT	すべて	SDR、SVR、RCVR、RQSTR、CLUSSDR、CLUSRCVR
MaxMsgLength	MQXR_INIT	すべて	すべて
PutAuthority	MQXR_INIT	すべて	SDR、SVR、RCVR、RQSTR、CLUSSDR、CLUSRCVR
DataConversion	MQXR_INIT	すべて	すべて
MCAUserIdentifier	MQXR_INIT、MQXR_INIT_SEC、MQXR_SEC_MSG、MQXR_SEC_PARMS	セキュリティー	RCVR、RQSTR、SVRCONN、CLUSRCVR
ConnectionName	MQXR_INIT	すべて	SDR、SVR、RQSTR、CLNTCONN、CLUSSDR、CLUSRCVR
MsgRetryUserData	MQXR_INIT	すべて	RCVR、RQSTR、CLUSRCVR
MsgRetryCount	MQXR_INIT	すべて	RCVR、RQSTR、CLUSRCVR
MsgRetryInterval	MQXR_INIT	すべて	RCVR、RQSTR、CLUSRCVR
HeartbeatInterval	MQXR_INIT	すべて	すべて
BatchInterval	MQXR_INIT	すべて	SDR、SVR、CLUSSDR、CLUSRCVR
NonPersistentMsgSpeed	MQXR_INIT	すべて	SDR、SVR、RCVR、RQSTR、CLUSSDR、CLUSRCVR
MCASecurityId	MQXR_INIT、MQXR_INIT_SEC、MQXR_SEC_MSG、MQXR_SEC_PARMS	セキュリティー	SDR、SVR、RCVR、RQSTR、SVRCONN、CLUSSDR、CLUSRCVR

表 859. 変更可能でチャンネルの動作に影響を与える可能性のあるフィールド (続き)			
フィールド	出口理由コード	出口タイプ	チャンネル・タイプ
SSLCipherSpec	MQXR_INIT	すべて	すべて
SSLPeerNamePtr	MQXR_INIT	すべて	すべて
SSLPeerNameLength	MQXR_INIT	すべて	すべて
SSLClientAuth	MQXR_INIT	すべて	SVR、RCVR、RQSTR、SVRCONN、CLUSRCVR
KeepAliveInterval	MQXR_INIT	すべて	すべて
LocalAddress	MQXR_INIT	すべて	SDR、SVR、RQSTR、CLNTCONN、CLUSSDR、CLUSRCVR
BatchHeartbeat	MQXR_INIT	すべて	SDR、SVR、CLUSSDR、CLUSRCVR
HdrCompList	MQXR_INIT	すべて	すべて
MsgCompList	MQXR_INIT	すべて	すべて
ChannelMonitoring	MQXR_INIT	すべて	SDR、SVR、RCVR、RQSTR、SVRCONN、CLUSSDR、CLUSRCVR
ChannelStatistics	MQXR_INIT	すべて	SDR、SVR、RCVR、RQSTR、CLUSSDR、CLUSRCVR
SharingConversations	MQXR_INIT	すべて	SVRCONN、CLNTCONN
PropertyControl	MQXR_INIT	すべて	SDR、SVR、CLUSSDR、CLUSRCVR

MQCXP - チャンネル出口パラメーター

MQCXP 構造体は、メッセージ・チャンネル・エージェント (MCA)、クライアント接続チャンネル、またはサーバー接続チャンネルによって呼び出された出口のタイプごとに渡されます。

MQ_CHANNEL_EXIT を参照してください。

以下の説明で「出口への入力」として記述されるフィールドは、出口がチャンネルに制御を戻すとチャンネルによって無視されます。チャンネル出口パラメーター・ブロック内で出口が変更になるなどの入力フィールドも、次の呼び出し用に保存されることはありません。入出力フィールド (例えば、*ExitUserArea* フィールド) への変更は、出口の当該インスタンスの呼び出し用에만保存されます。このような変更内容を、

同じチャンネルで定義された異なる出口間でデータを渡したり、異なるチャンネルで定義されている同一出口に渡したりするために使用することはできません。

関連資料

[1523 ページの『フィールド』](#)

このトピックでは、MQCXP 構造体のすべてのフィールドをリストし、それぞれのフィールドについて説明しています。

[1534 ページの『C 宣言』](#)

以下の宣言は、MQCXP 構造体の C 宣言です。

[1535 ページの『COBOL 宣言』](#)

以下の宣言は、MQCXP 構造体の COBOL 宣言です。

[1536 ページの『RPG 宣言 \(ILE\)』](#)

以下の宣言は、MQCXP 構造体の RPG 宣言です。

[1537 ページの『System/390 アセンブラ宣言』](#)

以下の宣言は、MQCXP 構造体の System/390 アセンブラ宣言です。

フィールド

このトピックでは、MQCXP 構造体のすべてのフィールドをリストし、それぞれのフィールドについて説明しています。

StrucId (MQCHAR4)

このフィールドは、構造体 ID を指定します。

値は次のものでなければなりません。

MQCXP_STRUC_ID

チャンネル出口パラメーター構造体の ID。

C 言語の場合、定数 MQCXP_STRUC_ID_ARRAY も定義されます。この定数は MQCXP_STRUC_ID と同じ値ですが、ストリングではなく文字の配列です。

これは、出口に対する入力フィールドです。

Version (MQLONG)

このフィールドは、構造体バージョン番号を指定します。

値は環境によって異なります。

MQCXP_VERSION_1

バージョン 1 チャンネル出口パラメーター構造体。

MQCXP_VERSION_2

バージョン 2 チャンネル出口パラメーター構造体。

MQCXP_VERSION_3

バージョン 3 チャンネル出口パラメーター構造体。



他の箇所で明示されていない AIX and Linux システムでは、このフィールドにはこの値が入ります。

MQCXP_VERSION_4

バージョン 4 チャンネル出口パラメーター構造体。

MQCXP_VERSION_5

バージョン 5 チャンネル出口パラメーター構造体。

MQCXP_VERSION_6

バージョン 6 チャンネル出口パラメーター構造体。

MQCXP_VERSION_8

バージョン 8 チャンネル出口パラメーター構造体。



z/OS では、このフィールドにはこの値が入ります。

MQCXP_VERSION_9

バージョン 9 チャネル出口パラメーター構造体。

以下の環境では、このフィールドにはこの値が入ります。

-  AIX
-  IBM i
-  Linux
-  Windows
-  z/OS

これより新しいバージョンの構造体にのみ存在するフィールドは、そのフィールドの説明にその旨記載されています。以下の定数は、現行バージョンのバージョン番号を指定しています。

MQCXP_CURRENT_VERSION

チャネル出口パラメーター構造体の現行バージョン。

値は環境によって異なります。

注: 新しいバージョンの MQCXP 構造体を導入しても、既存部分のレイアウトは変更されません。したがって、バージョン番号が出口で使用しなければならないフィールドが含まれている最小バージョンと等しいかまたはそれより大きいことを、出口でチェックする必要があります。

これは、出口に対する入力フィールドです。

ExitId (MQLONG)

このフィールドは、呼び出される出口のタイプを指定します。このフィールドは、出口ルーチンへの入り口で設定されます。

属性の値は以下のとおりです。

MQXT_CHANNEL_SEC_EXIT

チャネル・セキュリティー出口。

MQXT_CHANNEL_MSG_EXIT

チャネル・メッセージ出口。

MQXT_CHANNEL_SEND_EXIT

チャネル送信出口。

MQXT_CHANNEL_RCV_EXIT

チャネル受信出口。

MQXT_CHANNEL_MSG_RETRY_EXIT

チャネル・メッセージ再試行出口。

MQXT_CHANNEL_AUTO_DEF_EXIT

チャネル自動定義出口。

z/OS では、この出口は、MQCHT_CLUSSDR タイプと MQCHT_CLUSRCVR タイプのチャネルに対してのみサポートされています。

これは、出口に対する入力フィールドです。

ExitReason (MQLONG)

このフィールドは、出口の呼び出し理由を指定します。このフィールドは、出口ルーチンへの入り口で設定されます。

自動定義出口は、これを使用しません。属性の値は以下のとおりです。

MQXR_INIT

出口の初期化。

この値は、出口が初めて呼び出されていることを意味します。この場合、出口は必要なすべてのリソース (例えば、メモリー) を獲得して初期化できます。

MQXR_TERM

出口の終了。

この値は、出口が終了されることを意味します。出口は、初期化された後で獲得したすべてのリソース (例えば、メモリー) を解放する必要があります。

MQXR_MSG

メッセージの処理。

この値は、出口がメッセージを処理するために呼び出されていることを意味します。この値は、チャネル・メッセージ出口にのみ使用されます。

MQXR_XMIT

伝送の処理。

この値は、チャネル送信出口および受信出口にのみ使用されます。

MQXR_SEC_MSG

セキュリティー・メッセージの受信。

この値は、チャネル・セキュリティー出口にのみ使用されます。

MQXR_INIT_SEC

セキュリティー交換の開始。

この値は、チャネル・セキュリティー出口にのみ使用されます。

受信側のセキュリティー出口は、MQXR_INIT で呼び出された直後に常にこの理由で呼び出され、セキュリティー交換を開始する機会が与えられます。出口がこの機会を (MQXCC_SEND_SEC_MSG または MQXCC_SEND_AND_REQUEST_SEC_MSG の代わりに MQXCC_OK を返すことによって) 拒否した場合は、送信側のセキュリティー出口が MQXR_INIT_SEC で呼び出されます。

受信側のセキュリティー出口が (MQXCC_SEND_SEC_MSG または MQXCC_SEND_AND_REQUEST_SEC_MSG を返すことによって) セキュリティー交換を開始した場合、送信側のセキュリティー出口が MQXR_INIT_SEC で呼び出されることはありません。代わりに、受信側のメッセージを処理するために MQXR_SEC_MSG で呼び出されます。(いずれの場合も、最初に MQXR_INIT を指定して呼び出されます。)

いずれかのセキュリティー出口がチャネルの終了要求を出した場合 (*ExitResponse* を MQXCC_SUPPRESS_FUNCTION または MQXCC_CLOSE_CHANNEL に設定した場合) を除き、セキュリティー交換は、その交換を開始した側で完了させなければなりません。したがって、MQXR_INIT_SEC でセキュリティー出口を呼び出し、出口が交換を開始した場合、次にその出口を呼び出すと、MQXR_SEC_MSG が指定されます。これは、出口が処理するセキュリティー・メッセージの有無に関わらずに行われることです。相手側が MQXCC_SEND_SEC_MSG または MQXCC_SEND_AND_REQUEST_SEC_MSG を返す場合には、セキュリティー・メッセージがありますが、相手側が MQXCC_OK を返す場合、または相手側にセキュリティー出口がない場合には、セキュリティー・メッセージはありません。処理対象のセキュリティー・メッセージがなければ、*DataLength* がゼロに設定された状態で開始側のセキュリティー出口が再度呼び出されます。

MQXR_RETRY

メッセージの再試行。

この値は、メッセージ再試行出口にのみ使用されます。

MQXR_AUTO_CLUSSDR

クラスター送信側チャネルの自動定義。

この値は、チャネル自動定義出口にのみ使用されます。

MQXR_AUTO_RECEIVER

受信側チャネルの自動定義。

この値は、チャネル自動定義出口にのみ使用されます。

MQXR_AUTO_SVRCONN

サーバー接続チャネルの自動定義。

この値は、チャンネル自動定義出口にのみ使用されます。

MQXR_AUTO_CLUSRCVR

クラスター受信側チャンネルの自動定義。

この値は、チャンネル自動定義出口にのみ使用されます。

MQXR_SEC_PARMS

セキュリティー・パラメーター。

この値は、セキュリティー出口にのみ適用されます。この値は、MQCSP 構造体が出口に渡されることを意味します。詳しくは、[286 ページの『MQCSP - セキュリティー・パラメーター』](#)を参照してください。

注：

1. チャンネルに複数の出口が定義されている場合、MCA の初期化時には、それぞれの出口が MQXR_INIT で呼び出されます。また、MCA の終了時にも、それぞれが MQXR_TERM で呼び出されます。
2. *Version* が MQCXP_VERSION_4 より小さい場合、チャンネル自動定義出口には *ExitReason* は設定されません。この場合、MQXR_AUTO_SVRCONN 値が暗黙指定されます。

これは、出口に対する入力フィールドです。

ExitResponse (MQLONG)

このフィールドは、出口からの応答を指定します。

このフィールドは、MCA と通信するために、出口により設定されます。値は、次のいずれかでなければなりません。

MQXCC_OK

出口が正常に終了した。

- チャンネル・セキュリティー出口の場合、この値は、メッセージ転送を正常に続行できることを意味します。
- チャンネル・メッセージ再試行出口の場合、この値は、MCA は出口によって MQCXP の *MsgRetryInterval* フィールドに返された時間間隔だけ待機してから、メッセージを再試行しなければならないことを意味します。

ExitResponse2 フィールドに、追加情報が格納されている場合もあります。

MQXCC_SUPPRESS_FUNCTION

機能を抑止。

- チャンネル・セキュリティー出口の場合、この値は、チャンネルを終了する必要があることを意味します。
- チャンネル・メッセージ出口の場合、この値は、宛先へのメッセージ転送がこれ以上は続行されないことを意味します。代わりに、MCA は例外レポート・メッセージを作成し (元のメッセージの送信側によって要求されている場合)、元のバッファーに入れているメッセージを送達不能キューに入れるか (送信側が MQRO_DEAD_LETTER_Q を指定している場合)、または廃棄します (送信側が MQRO_DISCARD_MSG を指定している場合)。

持続メッセージの場合、送信側が MQRO_DEAD_LETTER_Q を指定していても、送達不能キューへの書き込みが失敗した場合、または送達不能キューがない場合は、元のメッセージは伝送キューに残され、レポート・メッセージは作成されません。レポート・メッセージが正常に作成できなかった場合も、元のメッセージは伝送キューに残されます。

送達不能キュー上のメッセージの始めにある MQDLH 構造体の *Feedback* フィールドは、メッセージが送達不能キューに入れられた理由を示します。このフィードバック・コードは、例外レポート・メッセージのメッセージ記述子でも使用されます (送信側が例外レポート・メッセージを要求している場合)。

- チャンネル・メッセージ再試行出口の場合、この値は、MCA が待機してメッセージの再試行を行わないことを意味します。代わりに、MCA は即時に通常の障害処理を続行します (メッセージの送信側の指定に従って、メッセージを送達不能キューに入れるか、またはメッセージを破棄します)。

- チャンネル自動定義出口では、MQXCC_OK または MQXCC_SUPPRESS_FUNCTION を指定する必要があります。どちらの値も指定されていない場合は、デフォルトで MQXCC_SUPPRESS_FUNCTION が想定され、自動定義が中止されます。

この応答は、チャンネル送信および受信出口ではサポートされません。

MQXCC_SEND_SEC_MSG

セキュリティー・メッセージを送信します。

この値は、チャンネル・セキュリティー出口によってのみ設定できます。これは、出口によって、相手側に送信するセキュリティー・メッセージが指定されていることを意味します。

MQXCC_SEND_AND_REQUEST_SEC_MSG

応答が必要なセキュリティー・メッセージを送信します。

この値は、チャンネル・セキュリティー出口によってのみ設定できます。これは以下のことを意味します。

- 相手側に送信するセキュリティー・メッセージは、出口が指定しています。
- 出口には、パートナーからの応答が必要です。応答が受信されない場合、出口は通信を継続できるかどうかをまだ判別していないため、チャンネルを終了する必要があります。

MQXCC_SUPPRESS_EXIT

出口を抑止します。

- この値は、セキュリティー出口または自動定義出口以外のすべてのタイプのチャンネル出口で設定することができます。ExitReason に MQXR_TERM を指定して出口が再び呼び出されると、このチャンネルが終了するまでその出口の後続呼び出しが抑止されます (チャンネル定義で出口の名前がブランクになっている場合と同様です)。
- メッセージ再試行出口がこの値を戻した場合、後続メッセージのメッセージ再試行は、通常どおり *MsgRetryCount* および *MsgRetryInterval* チャンネル属性によって制御されます。現行メッセージについては、理由コードが、通常 MCA が再試行を行うことになる理由コードである場合 (1480 ページの『MQCD - チャンネル定義』の *MsgRetryCount* フィールドを参照)、MCA は *MsgRetryInterval* チャンネル属性で指定された間隔で、再試行を残っている回数だけ実行します。残りの再試行回数は、**MsgRetryCount** 属性の値です。これは、出口が現行メッセージについて MQXCC_OK に返した回数よりも小さい値です。この値が負の値の場合は、MCA は現行メッセージについてそれ以上の再試行を行いません。

MQXCC_CLOSE_CHANNEL

チャンネルをクローズします。

この値は、自動定義出口以外のタイプのチャンネル出口で設定することができます。

共有会話が使用可能になっていない場合、この値によってチャンネルがクローズされます。

共有会話が使用可能になっている場合、この値によって会話が終了されます。この会話がチャンネルでの唯一の会話である場合には、チャンネルもクローズされます。

このフィールドは、出口からの入出力フィールドです。

ExitResponse2 (MQLONG)

このフィールドは、出口からの 2 次応答を指定します。

このフィールドは、出口ルーチンへの入り口でゼロに設定されます。このフィールドは、IBM MQ チャンネル関数に追加情報を提供するために出口によって設定されることがあります。自動定義出口は、これを使用しません。

出口は、以下の 1 つ以上の値を設定できます。複数の値が必要な場合は、値が追加されます。無効な組み合わせは表示されます。それ以外の組み合わせは有効です。

MQXR2_PUT_WITH_DEF_ACTION

デフォルト・アクションで書き込み。

この値は、受信側のチャンネル・メッセージ出口が設定します。これは、MCA のデフォルト・アクションでメッセージが書き込まれることを意味します。つまり、MCA のデフォルト・ユーザー ID、または

メッセージの MQMD (メッセージ記述子) に含まれるコンテキスト *UserIdentifier* のいずれかが書き込まれます。

値はゼロです。これは、出口を呼び出すときに設定される初期値に対応します。この定数は、文書化の目的で提供されています。

MQXR2_PUT_WITH_DEF_USERID

デフォルト・ユーザー ID で書き込み。

この値を設定できるのは、受信側のチャネル・メッセージ出口のみです。これは、メッセージに MCA のデフォルト・ユーザー ID が書き込まれることを意味します。

MQXR2_PUT_WITH_MSG_USERID

メッセージのユーザー ID で書き込み。

この値を設定できるのは、受信側のチャネル・メッセージ出口のみです。これは、メッセージに、メッセージの MQMD (メッセージ記述子) に含まれる (出口によって変更されている可能性のある) コンテキスト *UserIdentifier* が書き込まれることを意味します。

MQXR2_PUT_WITH_DEF_ACTION、MQXR2_PUT_WITH_DEF_USERID、および MQXR2_PUT_WITH_MSG_USERID のうち、1 つだけを設定してください。

MQXR2_USE_AGENT_BUFFER

エージェント・バッファの使用。

この値は、渡されるデータはすべて、*ExitBufferAddr* ではなく *AgentBuffer* 内にあることを意味します。

値はゼロです。これは、出口を呼び出すときに設定される初期値に対応します。この定数は、文書化の目的で提供されています。

MQXR2_USE_EXIT_BUFFER

出口バッファの使用。

この値は、渡されるデータはすべて、*AgentBuffer* ではなく *ExitBufferAddr* 内にあることを意味します。

MQXR2_USE_AGENT_BUFFER および MQXR2_USE_EXIT_BUFFER のうち、どちらか 1 つだけを設定してください。

MQXR2_DEFAULT_CONTINUATION

デフォルト継続。

チェーン内での次の出口への継続は、以下のように最後に呼び出された出口からの応答によって異なります。

- MQXCC_SUPPRESS_FUNCTION または MQXCC_CLOSE_CHANNEL が戻ると、これ以上チェーン内の出口は呼び出されません。
- 別の戻り値の場合は、チェーン内の次の出口が呼び出されます。

MQXR2_CONTINUE_CHAIN

次の出口へと続きます。

MQXR2_SUPPRESS_CHAIN

チェーン内の残りの出口をスキップします。

これは、出口に対する入出力フィールドです。

Feedback (MQLONG)

このフィールドは、フィードバック・コードを指定します。

このフィールドは、出口ルーチンへの入り口で MQFB_NONE に設定されます。

チャネル・メッセージ出口で *ExitResponse* フィールドが MQXCC_SUPPRESS_FUNCTION に設定されている場合、*Feedback* フィールドは、メッセージが送達不能 (未配布メッセージ) キューに入れられた理由を示すフィードバック・コードを示します。例外レポートが要求されている場合、このフィールドは、例

外レポートを送信するためにも使用されます。この場合、*Feedback* フィールドが *MQFB_NONE* に設定されていると、以下のフィードバック・コードが使用されます。

MQFB_STOPPED_BY_MSG_EXIT

メッセージはチャンネル・メッセージ出口によって停止されました。

チャンネル・セキュリティ出口、送信出口、受信出口、およびメッセージ再試行出口がこのフィールドに返す値は、MCA では使用しません。

自動定義出口がこのフィールドに返す値は、*ExitResponse* が *MQXCC_OK* に設定されている場合には使用されません。それ以外の場合は、この戻り値がイベント・メッセージ内の *AuxErrorDataInt1* パラメーターとして使用されます。

これは、出口からの入出力フィールドです。

MaxSegmentLength (MQLONG)

このフィールドには、単一の伝送で送信することのできる最大長をバイト単位で指定します。

自動定義出口は、これを使用しません。これは、チャンネル送信出口にとっては、意味があります。この出口では、伝送セグメントのサイズを *MaxSegmentLength* よりも大きな値にならないようにする必要がありますからです。この長さの初期値は 8 バイトであり、出口で変更しないでください。この値は、チャンネルの開始時に IBM MQ チャンネル機能間で折衝されます。セグメント長についての詳細は、[チャンネル出口プログラムの作成](#)を参照してください。

ExitReason が *MQXR_INIT* である場合は、このフィールドの値は意味をもちません。

これは、出口に対する入力フィールドです。

ExitUserArea (MQBYTE16)

このフィールドは、出口ユーザー域を指定します。これは、出口によって使用可能なフィールドです。

出口が最初に呼び出される前 (*ExitReason* が *MQXR_INIT* に設定されています) には、2 進数のゼロに初期設定されていて、そのあとで出口がこのフィールドを変更すると、変更内容はその出口の呼び出し間で保存されます。

以下の値が定義されます。

MQXUA_NONE

ユーザー情報なし。

値は、フィールドの長さについては 2 進ゼロです。

C 言語の場合、定数 *MQXUA_NONE_ARRAY* も定義されます。この定数は、*MQXUA_NONE* と同じ値ですが、ストリングではなく文字の配列です。

このフィールドの長さは *MQ_EXIT_USER_AREA_LENGTH* によって指定されます。これは、出口に対する入出力フィールドです。

ExitData (MQCHAR32)

このフィールドは、出口データを指定します。

このフィールドは、出口ルーチンへの入り口で、IBM MQ チャンネル機能がチャンネル定義から得た情報に設定されます。この情報が得られない場合には、このフィールドは完全にブランクになります。

このフィールドの長さは *MQ_EXIT_DATA_LENGTH* によって指定されます。

これは、出口に対する入力フィールドです。

Version が *MQCXP_VERSION_2* より小さい場合は、この構造体の以下のフィールドは提供されません。

MsgRetryCount (MQLONG)

このフィールドは、メッセージが再試行された回数を指定します。

特定のメッセージについて初めて出口が呼び出されるときには、このフィールドの値はゼロになっています (再試行はまだ行われていません)。以後、そのメッセージについて出口が呼び出されるたびに、MCA は値を 1 ずつ増分します。

これは、出口に対する入力フィールドです。 *ExitReason* が MQXR_INIT である場合は、このフィールドの値は意味をもちません。 *Version* が MQCXP_VERSION_2 より小さい場合は、このフィールドは提供されません。

MsgRetryInterval (MQLONG)

PUT 操作の再試行が行われるまでの最小間隔 (ミリ秒単位) を指定します。

特定のメッセージについて初めて出口が呼び出されたときは、このフィールドには *MsgRetryInterval* チャンネル属性の値が入っています。 出口は、この値をそのままにしておくことも、別の時間間隔 (ミリ秒単位) を指定するように変更することもできます。 出口が *ExitResponse* に MQXCC_OK を返した場合、MCA は MQOPEN 操作または MQPUT 操作を再試行する前に、少なくともこの時間間隔だけは待機します。 指定する時間間隔は、ゼロ以上でなければなりません。

該当のメッセージについて出口が 2 回目以降に呼び出されたときは、このフィールドには出口の前の呼び出しで戻された値が入っています。

MsgRetryInterval フィールドで返された値が 0 より小さいか、または 999 999 999 より大きく、しかも *ExitResponse* が MQXCC_OK である場合、MCA は MQCXP の *MsgRetryInterval* フィールドを無視し、*MsgRetryInterval* チャンネル属性で指定された間隔だけ待機します。

これは、出口に対する入出力フィールドです。 *ExitReason* が MQXR_INIT である場合は、このフィールドの値は意味をもちません。 *Version* が MQCXP_VERSION_2 より小さい場合は、このフィールドは提供されません。

MsgRetryReason (MQLONG)

このフィールドは、前回のメッセージ書き込み試行の理由コードを指定します。

このフィールドは、前回のメッセージ書き込み試行の理由コードで、MQRC_* 値の 1 つです。

これは、出口に対する入力フィールドです。 *ExitReason* が MQXR_INIT である場合は、このフィールドの値は意味をもちません。 *Version* が MQCXP_VERSION_2 より小さい場合は、このフィールドは提供されません。

Version が MQCXP_VERSION_3 より小さい場合は、この構造体の以下のフィールドは提供されません。

HeaderLength (MQLONG)

このフィールドは、ヘッダー情報の長さを指定します。

このフィールドは、メッセージ出口およびメッセージ再試行出口にのみ適用されます。 値はメッセージ・データの最初にある経路指定ヘッダー構造の長さです。これらは、MQXQH 構造、MQMDE (メッセージ記述拡張ヘッダー)、および MQDH 構造と MQXQH 構造に続く MQOR および MQPMR レコードの配列 (配布リスト・メッセージの場合) です。

メッセージ出口は、このヘッダー情報を調べて、必要であれば変更できます。ただし、出口が戻したデータは、正しい形式のままにしておく必要があります。送信側の出口でヘッダー・データを暗号化したり圧縮したりすることは禁止されます。たとえ受信側のメッセージ出口が復元する機能を持っている場合でも同じです。

メッセージ出口が、ヘッダー情報に対して長さを変える (例えば、別の宛先を配布リスト・メッセージに追加する) などの変更を行う場合、それに対応して *HeaderLength* の値も戻る前に変更する必要があります。

これは、出口に対する入出力フィールドです。 *ExitReason* が MQXR_INIT である場合は、このフィールドの値は意味をもちません。 *Version* が MQCXP_VERSION_3 より小さい場合は、このフィールドは提供されません。

PartnerName (MQCHAR48)

このフィールドは、パートナーの名前を指定します。

パートナー名は、次のとおりです。

- SVRCONN チャンネルの場合は、クライアントのログオン・ユーザー ID です。
- 他のすべてのタイプのチャンネルの場合は、パートナーのキュー・マネージャー名です。

出口が初期化されると、このフィールドはブランクになります。これは、キュー・マネージャーが、初期折衝が行われるまでパートナーの名前を知らないからです。

これは、出口に対する入力フィールドです。 *Version* が MQCXP_VERSION_3 より小さい場合は、このフィールドは提供されません。

FAPLevel (MQLONG)

折衝形式および折衝プロトコルのレベル。

これは、出口に対する入力フィールドです。このフィールドは、IBM サービスの指示があった場合にのみ変更します。 *Version* が MQCXP_VERSION_3 より小さい場合は、このフィールドは提供されません。

CapabilityFlags (MQLONG)

機能フラグは MQCF_NONE または MQCF_DIST_LISTS に設定できます。

以下のいずれかの機能フラグを設定できます。

MQCF_NONE

フラグなし。

MQCF_DIST_LISTS

配布リストがサポートされています。

これは、出口に対する入力フィールドです。 *Version* が MQCXP_VERSION_3 より小さい場合は、このフィールドは提供されません。

ExitNumber (MQLONG)

このフィールドは、出口の順序番号を指定します。

ExitId で定義されたタイプ内での出口の順序数。例えば、呼び出される出口が 3 番目のメッセージ出口で定義されている場合、このフィールドには値 3 が入ります。出口タイプが、出口のリストを定義できない場合 (例えば、セキュリティ出口)、このフィールドの値は 1 になります。

これは、出口に対する入力フィールドです。 *Version* が MQCXP_VERSION_3 より小さい場合は、このフィールドは提供されません。

Version が MQCXP_VERSION_5 より小さい場合は、この構造体の以下のフィールドは提供されません。

ExitSpace (MQLONG)

このフィールドは、使用する出口に予約されている伝送バッファ内のバイト数を指定します。

このフィールドは、送信出口にのみ適用されます。使用する出口のために IBM MQ チャネル機能が予約する伝送バッファ内のスペース合計をバイト単位で指定します。このフィールドは、出口は、補足し合う他方の受信出口で使用するために、伝送バッファに少量のデータ (一般には、数百バイトを超えない範囲) を追加できるようになります。送信出口によって追加されたデータは、受信出口で除去する必要があります。

z/OS では、この値は常に 0 です。

注：この機能は、大量のデータを送信するときに使用しないでください。そのように使用すると、パフォーマンスが低下したり、チャネルの操作が禁止される可能性があります。

ExitSpace を設定すると、出口で使用するための最低バイト数が伝送バッファで必ず使用可能になることが保証されます。ただし、出口は、予約された量より少ない量を使用することも、伝送バッファに使用可能なスペースがあれば、予約された量より多くの量を使用することもできます。バッファ内の出口スペースは、既存のデータの大きさに応じて提供されます。

ExitSpace は、*ExitReason* の値が MQXR_INIT である場合に限り、出口によって設定できます。その他の場合はすべて、出口によって戻される値は無視されます。出口に対する入力では、*ExitSpace* は、MQXR_INIT 呼び出しの場合はゼロになり、その他の場合は MQXR_INIT 呼び出しによって戻される値になります。

MQXR_INIT 呼び出しで戻される値が負であるか、チェーン内のすべての送信出口の要求出口スペースを予約後に、メッセージ・データ用に伝送バッファで使用するスペースが 1024 バイトを切るような場合、MCA は、エラー・メッセージを出力してチャネルをクローズします。同様に、データ転送時に、メッセー

ジ・データ用のスペースが伝送バッファに 1024 バイトも残っていないのに、送信出口チェーンの出口が予約した以上のユーザー・スペースを割り振る場合、MCA は、エラー・メッセージを出力してチャンネルをクローズします。1024 という制限があることにより、送信出口のチェーンは、フローをセグメント化せずに、チャンネルの制御フローと管理フローを処理できるようになります。

ExitReason が MQXR_INIT の場合には、これは入出力フィールドになり、それ以外の場合にはすべて入力フィールドになります。*Version* が MQCXP_VERSION_5 より小さい場合は、このフィールドは提供されません。

SSLCertUserId (MQCHAR12)

このフィールドには、リモート証明書に関連したユーザー ID が指定されます。

z/OS 以外のすべてのプラットフォームでは、このフィールドはブランクです。

これは、出口に対する入力フィールドです。*Version* が MQCXP_VERSION_6 より小さい場合は、このフィールドは提供されません。

SSLRemCertIssNameLength (MQLONG)

このフィールドは、*SSLCertRemoteIssuerNamePtr* によって示されたリモート証明書発行者の完全識別名の長さ (バイト数) を指定します。

これは、出口に対する入力フィールドです。*Version* が MQCXP_VERSION_6 より小さい場合は、このフィールドは提供されません。これが TLS チャンネルでない場合、値はゼロです。

SSLRemCertIssNamePtr (PMQVOID)

このフィールドは、リモート証明書発行者の完全識別名のアドレスを指定します。

TLS チャンネルでない場合、値はヌル・ポインターです。

これは、出口に対する入力フィールドです。*Version* が MQCXP_VERSION_6 より小さい場合は、このフィールドは提供されません。

注：サブジェクト識別名および発行者識別名を決定するときのチャンネル・セキュリティ出口の動作は、IBM WebSphere MQ 7.1 から変更されています。詳しくは、[チャンネル・セキュリティ出口プログラム](#)を参照してください。

SecurityParms (PMQCSP)

このフィールドは、認証資格情報を指定するために使用される MQCSP 構造体のアドレスを指定します。

このフィールドの初期値は、ヌル・ポインターです。

これは、出口に対する入出力フィールドです。*Version* が MQCXP_VERSION_6 より小さい場合は、このフィールドは提供されません。

このフィールドで終了処理が返す値は、MQXR_TERM まで使用可能 IBM MQ でなければならない。

CurHdrCompression (MQLONG)

このフィールドには、現在、ヘッダー・データの圧縮にどの手法が使用されているかを指定します。

値は次のいずれかに設定されます。

MQCOMPRESS_NONE

ヘッダー・データ圧縮は実行されません。

MQCOMPRESS_SYSTEM

ヘッダー・データ圧縮が実行されます。

値は、送信側チャンネルのメッセージ出口によって変更可能で、MQCD の *HdrCompList* フィールドからアクセスされる折衝済みサポート値の 1 つに変更することができます。これにより、ヘッダー・データの圧縮に使用する方法を、メッセージの内容に基づいてメッセージごとに決めることができます。変更された値は現在のメッセージにのみ使用されます。この属性がサポートされない値に変更されると、チャンネルは終了します。送信側チャンネルのメッセージ出口の外で変更された場合、値は無視されます。

これは、出口に対する入出力フィールドです。*Version* が MQCXP_VERSION_6 より小さい場合は、このフィールドは提供されません。

CurMsgCompression (MQLONG)

このフィールドには、現在、メッセージ・データの圧縮にどの手法が使用されているかを指定します。

値は次のいずれかに設定されます。

MQCOMPRESS_NONE

ヘッダー・データ圧縮は実行されません。

MQCOMPRESS_RLE

ラン・レンジス・エンコードを使用してメッセージ・データ圧縮が実行されます。

MQCOMPRESS_ZLIBFAST

zlib 圧縮手法を使用してメッセージ・データ圧縮が実行されます。高速圧縮時間を推奨します。

MQCOMPRESS_ZLIBHIGH

zlib 圧縮手法を使用してメッセージ・データ圧縮が実行されます。ハイレベル圧縮を推奨します。

値は、送信側チャンネルのメッセージ出口によって変更可能で、MQCD の `MsgCompList` フィールドからアクセスされる折衝済みサポート値の 1 つに変更することができます。これにより、メッセージ・データを圧縮するために使用する技法を、メッセージの内容に応じてメッセージごとに決定することが可能になります。変更された値は現在のメッセージにのみ使用されます。この属性がサポートされない値に変更されると、チャンネルは終了します。送信側チャンネルのメッセージ出口の外で変更された場合、値は無視されます。

これは、出口に対する入出力フィールドです。Version が `MQCXP_VERSION_6` より小さい場合は、このフィールドは提供されません。

Hconn (MQHCONN)

このフィールドは、出口がその内部で MQI 呼び出しを行う必要がある場合に使用する接続ハンドルを指定します。

このフィールドは、クライアント接続チャンネルで実行する出口 (この場合の値は `MQHC_UNUSABLE_HCONN (-1)`) とは関係がありません。

これは、出口に対する入力フィールドです。Version が `MQCXP_VERSION_7` より小さい場合は、このフィールドは提供されません。

SharingConversations (MQBOOL)

このフィールドは、この会話がこのチャンネル・インスタンスで現在実行可能な唯一の会話であるかどうか、またはこのチャンネル・インスタンスで現在複数の会話が実行可能であるかどうかを指定します。

さらにこれは、この出口プログラムが、同時に実行中の別の出口プログラムにより MQCD が変更される、というリスクを負うかどうかを示します。

このフィールドは、クライアント接続チャンネルまたはサーバー接続チャンネルで実行する出口プログラムだけに関係します。

値は次のいずれかに設定されます。

FALSE

この出口インスタンスは、このチャンネル・インスタンスで現在実行可能な唯一の出口インスタンスです。これにより、他のチャンネル・インスタンスで実行する他の出口と競合することなく、この出口は安全に MQCD フィールドを更新することができます。MQCD フィールドへの変更に応じてチャンネルが動作するかどうかは、[1519 ページの『チャンネル出口での MQCD フィールドの変更』](#)にある MQCD フィールドの表で定義されています。

TRUE

この出口インスタンスは、このチャンネル・インスタンスで現在実行可能な唯一の出口インスタンスではありません。MQCD に変更が加えられても、チャンネルがそれに応じて動作することはありません。ただし、`MQXR_INIT` 以外の出口理由について、[1519 ページの『チャンネル出口での MQCD フィールドの変更』](#)の MQCD フィールドの表にリストされている変更を除きます。この出口で MQCD フィールドを更新する場合は、他の会話で同時に実行する他の出口との競合が生じないように、このチャンネル・インスタンスで実行する出口を直列化してください。

これは、出口に対する入力フィールドです。Version が `MQCXP_VERSION_7` より小さい場合は、このフィールドは提供されません。

MCAUserSource (MQLONG)

このフィールドは、指定された MCA ユーザー ID のソースを指定します。

以下のいずれかの値を取ります。

MQUSRC_MAP

MCAUSER 属性に指定されたユーザー ID。

MQUSRC_CHANNEL

ユーザー ID は、インバウンド・パートナーから流れてくるか、チャンネル・オブジェクト内に定義された MCAUSER フィールドに指定されます。

これは、出口に対する入力フィールドです。Version が MQCXP_VERSION_8 より小さい場合は、このフィールドは提供されません。

pEntryPoints (PMQIEP)

このフィールドは、MQI または DCI 呼び出しのインターフェース・エンタリー・ポイントのアドレスを指定します。

Version が MQCXP_VERSION_8 より小さい場合は、このフィールドは提供されません。

RemoteProduct (MQCHAR4)

このフィールドはリモート製品名を示します。

このフィールドは、DISPLAY CHSATUS の **RPRODUCT** フィールドに表示される、クライアントのリモート製品 (例えば、C または Java) を示します。

Version が MQCXP_VERSION_9 より小さい場合は、このフィールドは提供されません。

RemoteVersion (MQCHAR8)

このフィールドは、リモート・バージョンの名前を示します。

このフィールドは、DISPLAY CHSTATUS の **RVERSION** フィールドに表示されるクライアント・ライブラリーのバージョンを示します。

Version が MQCXP_VERSION_9 より小さい場合は、このフィールドは提供されません。

C 宣言

以下の宣言は、MQCXP 構造体の C 宣言です。

```
typedef struct tagMQCXP MQCXP;
struct tagMQCXP {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQLONG     ExitId;           /* Type of exit */
    MQLONG     ExitReason;       /* Reason for invoking exit */
    MQLONG     ExitResponse;     /* Response from exit */
    MQLONG     ExitResponse2;    /* Secondary response from exit */
    MQLONG     Feedback;         /* Feedback code */
    MQLONG     MaxSegmentLength; /* Maximum segment length */
    MQBYTE16   ExitUserArea;     /* Exit user area */
    MQCHAR32   ExitData;         /* Exit data */
    MQLONG     MsgRetryCount;    /* Number of times the message has been
                                retried */
    MQLONG     MsgRetryInterval; /* Minimum interval in milliseconds after
                                which the put operation should be
                                retried */
    MQLONG     MsgRetryReason;   /* Reason code from previous attempt to
                                put the message */
    MQLONG     HeaderLength;     /* Length of header information */
    MQCHAR48   PartnerName;      /* Partner Name */
    MQLONG     FAPLevel;         /* Negotiated Formats and Protocols
                                level */
    MQLONG     CapabilityFlags;  /* Capability flags */
    MQLONG     ExitNumber;       /* Exit number */
    /* Ver:3 */
    /* Ver:4 */
    MQLONG     ExitSpace;        /* Number of bytes in transmission buffer
                                reserved for exit to use */
    /* Ver:5 */
    MQCHAR12   SSLCertUserid;    /* User identifier associated
                                with remote TLS certificate */
}
```

```

MQLONG    SSLRemCertIssNameLength; /* Length of
                                     distinguished name of issuer
                                     of remote TLS certificate */
MQPTR     SSLRemCertIssNamePtr; /* Address of
                                     distinguished name of issuer
                                     of remote TLS certificate */
PMQVOID    SecurityParms; /* Security parameters */
MQLONG     CurHdrCompression; /* Header data compression
                                used for current message */
MQLONG     CurMsgCompression; /* Message data compression
                                used for current message */

/* Ver:6 */
MQHCONN    Hconn; /* Connection handle */
MQBOOL     SharingConversations; /* Multiple conversations
                                   possible on channel inst? */

/* Ver:7 */
MQLONG     MCAUserSource; /* Source of the provided MCA user ID */
PMQIEP     pEntryPoints; /* Address of the MQIEP structure */

/* Ver:8 */
MQCHAR4     RemoteProduct; /* The identifier for the remote product */
MQCHAR8     RemoteVersion; /* The version of the remote product */

/* Ver:9 */
};

```

COBOL 宣言

以下の宣言は、MQCXP 構造体の COBOL 宣言です。

```

** MQCXP structure
10 MQCXP.
** Structure identifier
15 MQCXP-STRUCID PIC X(4).
** Structure version number
15 MQCXP-VERSION PIC S9(9) BINARY.
** Type of exit
15 MQCXP-EXITID PIC S9(9) BINARY.
** Reason for invoking exit
15 MQCXP-EXITREASON PIC S9(9) BINARY.
** Response from exit
15 MQCXP-EXITRESPONSE PIC S9(9) BINARY.
** Secondary response from exit
15 MQCXP-EXITRESPONSE2 PIC S9(9) BINARY.
** Feedback code
15 MQCXP-FEEDBACK PIC S9(9) BINARY.
** Maximum segment length
15 MQCXP-MAXSEGMENTLENGTH PIC S9(9) BINARY.
** Exit user area
15 MQCXP-EXITUSERAREA PIC X(16).
** Exit data
15 MQCXP-EXITDATA PIC X(32).
** Number of times the message has been retrieved
15 MQCXP-MSGRETRYCOUNT PIC S9(9) BINARY.
** Minimum interval in milliseconds after which the put operation
** should be retried
15 MQCXP-MSGRETRYINTERVAL PIC S9(9) BINARY.
** Reason code from previous attempt to put the message
15 MQCXP-MSGRETRYREASON PIC S9(9) BINARY.
** Length of header information
15 MQCXP-HEADERLENGTH PIC S9(9) BINARY.
** Partner Name
15 MQCXP-PARTNERNAME PIC X(48).
** Negotiated Formats and Protocols level
15 MQCXP-FAPLEVEL PIC S9(9) BINARY.
** Capability flags
15 MQCXP-CAPABILITYFLAGS PIC S9(9) BINARY.
** Exit number
15 MQCXP-EXITNUMBER PIC S9(9) BINARY.
** Number of bytes in transmission buffer reserved for exit to use
15 MQCXP-EXITSPACE PIC S9(9) BINARY.
** User Id associated with remote certificate
15 MQCXP-SSLCERTUSERID PIC X(12).
** Length of distinguished name of issuer of remote TLS
** certificate
15 MQCXP-SSLREMCERTISSNAMELENGTH PIC S9(9) BINARY.
** Address of distinguished name of issuer of remote TLS
** certificate
15 MQCXP-SSLREMCERTISSNAMEPTR POINTER.
** Security parameters
15 MQCXP-SECURITYPARMS PIC S9(18) BINARY.
** Header data compression used for current message

```

```

15 MQCXP-CURHDRCOMPRESSION      PIC S9(9) BINARY.
** Message data compression used for current message
15 MQCXP-CURMSGCOMPRESSION      PIC S9(9) BINARY.
** Connection handle
15 MQCXP-HCONN                  PIC S9(9) BINARY.
** Multiple conversations possible on channel instance?
15 MQCXP-SHARINGCONVERSATIONS   PIC S9(9) BINARY.
** Source of the provided MCA user ID
15 MQCXP-MCAUSERSOURCE          PIC S9(9) BINARY.
** Identifier of the remote product
15 MQCXP-RPRODUCT               PIC X(4).
** Identifier of the remote version
15 MQCXP-RVERSION               PIC X(8).

```

RPG 宣言 (ILE)

以下の宣言は、MQCXP 構造体の RPG 宣言です。

```

D*.1.....2.....3.....4.....5.....6.....7..
D* MQCXP Structure
D*
D* Structure identifier
D  CXSID              1          4
D* Structure version number
D  CXVER              5          8I 0
D* Type of exit
D  CXXID              9          12I 0
D* Reason for invoking exit
D  CXREA             13          16I 0
D* Response from exit
D  CXRES             17          20I 0
D* Secondary response from exit
D  CXRE2             21          24I 0
D* Feedback code
D  CXFB              25          28I 0
D* Maximum segment length
D  CXMSL             29          32I 0
D* Exit user area
D  CXUA              33          48
D* Exit data
D  CXDAT             49          80
D* Number of times the message has been retried
D  CXMRC             81          84I 0
D* Minimum interval in milliseconds after which the put operation
D* should be retried
D  CXMRI            85          88I 0
D* Reason code from previous attempt to put the message
D  CXMRR             89          92I 0
D* Length of header information
D  CXHDL             93          96I 0
D* Partner Name
D  CXPNM            97          144
D* Negotiated Formats and Protocols level
D  CXFAP            145          148I 0
D* Capability flags
D  CXCAP            149          152I 0
D* Exit number
D  CXEXN            153          156I 0
D* Number of bytes in transmission buffer reserved for exit to use
D  CXHDL            157          160I 0
D* User identifier associated with remote TLS certificate
D  CXSSLCU           161          172
D* Length of distinguished name of issuer of remote TLS certificate
D  CXSRCINL          173          176I 0
D* Address of distinguished name of issuer of remote TLS certificate
D  CXSRCINP          177          192*
D* Security parameters
D  CXSECP            193          208*
D* Header data compression used for current message
D  CXCHC            209          212I 0
D* Message data compression used for current message
D  CXCMC            213          216I 0
D* Connection handle
D  CXHCONN           217          220I 0
D* Multiple conversations possible on channel instance?
D  CXSHARECONV       221          224I 0
D* Source of the provided MCA user ID
D  MCAUSERSOURCE     225          228I 0
D* Identifier of the remote product
D  CXRPRO            229          232I 0

```

D*	Identifier of the remote version			
D	CXRVER	233	240I	0

System/390 アセンブラー宣言

以下の宣言は、MQCXP 構造体の System/390 アセンブラー宣言です。

MQCXP	DSECT		
MQCXP_STRUCID	DS	CL4	Structure identifier
MQCXP_VERSION	DS	F	Structure version number
MQCXP_EXITID	DS	F	Type of exit
MQCXP_EXITREASON	DS	F	Reason for invoking exit
MQCXP_EXITRESPONSE	DS	F	Response from exit
MQCXP_EXITRESPONSE2	DS	F	Secondary response from exit
MQCXP_FEEDBACK	DS	F	Feedback code
MQCXP_MAXSEGMENTLENGTH	DS	F	Maximum segment length
MQCXP_EXITUSERAREA	DS	XL16	Exit user area
MQCXP_EXITDATA	DS	CL32	Exit data
MQCXP_MSGRETRYCOUNT	DS	F	Number of times the message has been retried
* MQCXP_MSGRETRYINTERVAL	DS	F	Minimum interval in milliseconds after which the put operation should be retried
* MQCXP_MSGRETRYREASON	DS	F	Reason code from previous attempt to put the message
* MQCXP_HEADERLENGTH	DS	F	Length of header information
MQCXP_PARTNERNAME	DS	CL48	Partner Name
MQCXP_FAPLEVEL	DS	F	Negotiated Formats and Protocols level
* MQCXP_CAPABILITYFLAGS	DS	F	Capability flags
MQCXP_EXITNUMBER	DS	F	Exit number
MQCXP_EXITSPACE	DS	F	Number of bytes in transmission buffer reserved for exit to use
* MQCXP_SSLCERTUSERID	DS	CL12	User identifier associated with remote TLS certificate
* MQCXP_SSLREMCERTISSNAMELENGTH	DS	F	Length of distinguished name of issuer of remote TLS certificate
* MQCXP_SSLREMCERTISSNAMEPTR	DS	F	Address of distinguished name of issuer of remote TLS certificate
* MQCXP_SECURITYPARMS	DS	F	Address of security parameters
MQCXP_CURHDRCOMPRESSION	DS	F	Header data compression used for current message
* MQCXP_CURMSGCOMPRESSION	DS	F	Message data compression used for current message
* MQCXP_HCONN	DS	F	Connection handle
MQCXP_SHARINGCONVERSATIONS	DS	F	Multiple conversations possible on channel inst?
* MQCXP_MCAUSERSOURCE	DS	F	Source of the provided MCA user ID
MQCXP_RPRODUCT	DS	CL4	Identifier of the remote product
MQCXP_RVERSION	DS	CL8	Identifier of the remote version
MQCXP_LENGTH	EQU	*-MQCXP	
	ORG	MQCXP	
MQCXP_AREA	DS	CL(MQCXP_LENGTH)	

MQXWD - 出口待機記述子

MQXWD 構造体は、MQXWAIT 呼び出しに指定する入出力パラメーターです。

この構造は、z/OS でのみサポートされます。

関連資料

[1538 ページの『フィールド』](#)

このトピックでは、MQXWD 構造体のすべてのフィールドをリストし、それぞれのフィールドについて説明しています。

[1538 ページの『C 宣言』](#)

以下の宣言は、MQXWD 構造体の C 宣言です。

[1538 ページの『System/390 アセンブラー宣言』](#)

以下の宣言は、MQXWD 構造体の System/390 アセンブラー宣言です。

フィールド

このトピックでは、MQXWD 構造体のすべてのフィールドをリストし、それぞれのフィールドについて説明しています。

StrucId (MQCHAR4)

このフィールドは、構造体 ID を指定します。

値は次のものでなければなりません。

MQXWD_STRUC_ID

出口待機記述子構造体の ID。

C 言語の場合、定数 MQXWD_STRUC_ID_ARRAY も定義されます。この定数は MQXWD_STRUC_ID と同じ値ですが、ストリングではなく文字の配列です。

このフィールドの初期値は、MQXWD_STRUC_ID です。

Version (MQLONG)

このフィールドは、構造体バージョン番号を指定します。

値は次のものでなければなりません。

MQXWD_VERSION_1

出口待機記述子構造体のバージョン番号。

このフィールドの初期値は MQXWD_VERSION_1 です。

Reserved1 (MQLONG)

このフィールドは予約済みです。値はゼロでなければなりません。

これは入力フィールドです。

Reserved2 (MQLONG)

このフィールドは予約済みです。値はゼロでなければなりません。

これは入力フィールドです。

Reserved3 (MQLONG)

このフィールドは予約済みです。値はゼロでなければなりません。

これは入力フィールドです。

ECB (MQLONG)

このフィールドは、待機に使用するイベント制御ブロックを指定します。

このフィールドは、待機に使用するイベント制御ブロックです。MQXWAIT 呼び出しが実行される前は、ゼロに設定されていなければなりません。呼び出しが正常に完了すると、ここに通知コードが格納されます。

このフィールドは入出力フィールドです。

C 宣言

以下の宣言は、MQXWD 構造体の C 宣言です。

```
typedef struct tagMQXWD MQXWD;
struct tagMQXWD {
    MQCHAR4   StrucId;      /* Structure identifier */
    MQLONG    Version;      /* Structure version number */
    MQLONG    Reserved1;    /* Reserved */
    MQLONG    Reserved2;    /* Reserved */
    MQLONG    Reserved3;    /* Reserved */
    MQLONG    ECB;          /* Event control block to wait on */
};
```

System/390 アセンブラ宣言

以下の宣言は、MQXWD 構造体の System/390 アセンブラ宣言です。

MQXWD	DSECT		
MQXWD_STRUCID	DS	CL4	Structure identifier
MQXWD_VERSION	DS	F	Structure version number
MQXWD_RESERVED1	DS	F	Reserved
MQXWD_RESERVED2	DS	F	Reserved
MQXWD_RESERVED3	DS	F	Reserved
MQXWD_ECB	DS	F	Event control block to wait on
*			
MQXWD_LENGTH	EQU	*-MQXWD	
	ORG	MQXWD	
MQXWD_AREA	DS	CL(MQXWD_LENGTH)	

クラスター・ワークロード出口呼び出しとデータ構造体

ここでは、クラスター・ワークロード出口と、出口で使用するデータ構造体について説明します。ここで説明するのは、汎用プログラミング・インターフェース情報です。

クラスター・ワークロード出口は、以下のプログラミング言語で作成することができます。

- C
- System/390 アセンブラ (IBM MQ for z/OS)

呼び出しについては、次の節で説明します。

- [1540 ページの『MQ_CLUSTER_WORKLOAD_EXIT - 呼び出しの説明』](#)

出口で使用する構造体のデータ・タイプについては、次の各節で説明します。

- [1541 ページの『MQXCLWLN - クラスター・ワークロードのレコードのナビゲート』](#)
- [1545 ページの『MQWXP - クラスター・ワークロード出口のパラメーター構造体』](#)
- [1553 ページの『MQWDR - クラスター・ワークロード宛先レコード構造体』](#)
- [1558 ページの『MQWQR - クラスター・ワークロードのキュー・レコード構造体』](#)
- [1563 ページの『MQWCR - クラスター・ワークロードのクラスター・レコード構造体』](#)
-  [z/OS における CLUSTER コマンドの非同期の動作](#)

このセクション全体で、キュー・マネージャー属性とキュー属性は完全に表示されています。MQSC コマンドで使用する同等の名前をその下に表示します。MQSC コマンドの詳細については、[MQSC コマンド](#)を参照してください。

表 860. キュー・マネージャーの属性	
完全な名前	MQSC で使用する名前
<i>ClusterWorkloadData</i>	CLWLDATA
<i>ClusterWorkloadExit</i>	CLWLEXIT
<i>ClusterWorkloadLength</i>	CLWLEN

表 861. キューの属性	
完全な名前	MQSC で使用する名前
<i>DefBind</i>	DEFBIND
<i>DefPersistence</i>	DEFPSIST
<i>DefPriority</i>	DEFPRTY
<i>InhibitPut</i>	PUT
<i>QDesc</i>	DESCR

関連タスク

[クラスター・ワークロード出口の作成とコンパイル](#)

MQ_CLUSTER_WORKLOAD_EXIT - 呼び出しの説明

このクラスター・ワークロード出口は、使用可能なキュー・マネージャーにメッセージをルーティングするためにキュー・マネージャーによって呼び出されます。

注：MQ_CLUSTER_WORKLOAD_EXIT という名前のエントリー・ポイントは、キュー・マネージャーによって提供されません。代わりに、クラスター・ワークロード出口の名前は ClusterWorkloadExit キュー・マネージャー属性によって定義されます。

MQ_CLUSTER_WORKLOAD_EXIT 出口は、すべてのプラットフォームでサポートされています。

構文

MQ_CLUSTER_WORKLOAD_EXIT (*ExitParms*)

関連資料

[MQXCLWLN - クラスター・ワークロードのレコードのナビゲート](#)

MQXCLWLN 呼び出しは、クラスター・キャッシュ内に保管された MQWDR、MQWQR、および MQWCR の各レコードのチェーン全体をナビゲートするために使用します。

[MQWXP - クラスター・ワークロード出口のパラメーター構造体](#)

以下の表には、MQWXP - クラスター・ワークロード出口のパラメーター構造体内のフィールドがまとめられています。

[MQWDR - クラスター・ワークロード宛先レコード構造体](#)

以下の表には、MQWDR - クラスター・ワークロードの宛先レコード構造体内のフィールドがまとめられています。

[MQWQR - クラスター・ワークロードのキュー・レコード構造体](#)

以下の表には、MQWQR - クラスター・ワークロードのキュー・レコード構造体内のフィールドがまとめられています。

[MQWCR - クラスター・ワークロードのクラスター・レコード構造体](#)

以下の表には、MQWCR クラスター・ワークロードのレコード構造体内のフィールドがまとめられています。

MQ_CLUSTER_WORKLOAD_EXIT のパラメーター

MQ_CLUSTER_WORKLOAD_EXIT 呼び出しのパラメーターの説明。

ExitParms (MQWXP) - 入出力

出口パラメーター・ブロック。

- 出口では、ワークロードの管理方法を示すために MQWXP 内に情報を設定します。

関連資料

使用上の注意

クラスター・ワークロード出口によって実行される関数は、出口の提供者によって定義されます。ただし、出口は、関連付けられた制御ブロックの MQWXP で定義された規則に従っていなければなりません。

[MQ_CLUSTER_WORKLOAD_EXIT の言語呼び出し](#)

MQ_CLUSTER_WORKLOAD_EXIT では、C および高水準アセンブラの 2 つの言語がサポートされています。

使用上の注意

クラスター・ワークロード出口によって実行される関数は、出口の提供者によって定義されます。ただし、出口は、関連付けられた制御ブロックの MQWXP で定義された規則に従っていなければなりません。

MQ_CLUSTER_WORKLOAD_EXIT という名前のエントリー・ポイントは、キュー・マネージャーによって提供されません。ただし、C プログラミング言語では、MQ_CLUSTER_WORKLOAD_EXIT という名前の typedef が提供されています。この typedef を使用してユーザー作成の出口を宣言して、パラメーターが正しくなるようにします。

関連資料

[MQ_CLUSTER_WORKLOAD_EXIT のパラメーター](#)

[MQ_CLUSTER_WORKLOAD_EXIT 呼び出しのパラメーターの説明。](#)

[MQ_CLUSTER_WORKLOAD_EXIT の言語呼び出し](#)

[MQ_CLUSTER_WORKLOAD_EXIT](#) では、C および高水準アセンブラーの 2 つの言語がサポートされています。

MQ_CLUSTER_WORKLOAD_EXIT の言語呼び出し

[MQ_CLUSTER_WORKLOAD_EXIT](#) では、C および高水準アセンブラーの 2 つの言語がサポートされています。

C 言語での呼び出し

```
MQ_CLUSTER_WORKLOAD_EXIT (&ExitParms);
```

[MQ_CLUSTER_WORKLOAD_EXIT](#) をクラスター・ワークロード出口機能の名前に置き換えます。

MQ_CLUSTER_WORKLOAD_EXIT パラメーターを以下のように宣言します。

```
MQWXP ExitParms; /* Exit parameter block */
```

高水準アセンブラー呼び出し

```
CALL EXITNAME, (EXITPARMS)
```

パラメーターを次のように宣言します。

EXITPARMS	CMQWXP	Exit parameter block

関連資料

[MQ_CLUSTER_WORKLOAD_EXIT のパラメーター](#)

[MQ_CLUSTER_WORKLOAD_EXIT 呼び出しのパラメーターの説明。](#)

使用上の注意

クラスター・ワークロード出口によって実行される関数は、出口の提供者によって定義されます。ただし、出口は、関連付けられた制御ブロックの MQWXP で定義された規則に従っていなければなりません。

MQXCLWLN - クラスター・ワークロードのレコードのナビゲート

MQXCLWLN 呼び出しは、クラスター・キャッシュ内に保管された MQWDR、MQWQR、および MQWCR の各レコードのチェーン全体をナビゲートするために使用します。

クラスター・キャッシュは、クラスターに関連した情報を保管するために使用される主記憶の領域です。

クラスター・キャッシュが静的である場合は、固定サイズになっています。クラスター・キャッシュを「動的」に設定すると、このキャッシュを必要に応じて拡張することができます。

クラスター・キャッシュのタイプは、システム・パラメーターまたはシステム・マクロのいずれかを使用して、STATIC または DYNAMIC に設定します。

- ▶ **Multi** システムパラメータを ClusterCacheType オンにしてください [マルチプラットフォーム](#)。
- ▶ **z/OS** マクロ CSQ6SYSP 内の CLCACHE パラメータを使用してください。z/OS

構文

```
MQXCLWLN (ExitParms, CurrentRecord, NextOffset, NextRecord, Compcode, Reason)
```

関連資料

MQ_CLUSTER_WORKLOAD_EXIT - 呼び出しの説明

このクラスター・ワークロード出口は、使用可能なキュー・マネージャーにメッセージをルーティングするためにキュー・マネージャーによって呼び出されます。

MQWXP - クラスター・ワークロード出口のパラメーター構造体

以下の表には、MQWXP - クラスター・ワークロード出口のパラメーター構造体内のフィールドがまとめられています。

MQWDR - クラスター・ワークロード宛先レコード構造体

以下の表には、MQWDR - クラスター・ワークロードの宛先レコード構造体内のフィールドがまとめられています。

MQWQR - クラスター・ワークロードのキュー・レコード構造体

以下の表には、MQWQR - クラスター・ワークロードのキュー・レコード構造体内のフィールドがまとめられています。

MQWCR - クラスター・ワークロードのクラスター・レコード構造体

以下の表には、MQWCR クラスター・ワークロードのレコード構造体内のフィールドがまとめられています。

MQXCLWLN - クラスター・ワークロード・レコードのナビゲート用のパラメーター

MQXCLWLN 呼び出しのパラメーターの説明。

ExitParms (MQWXP) - 入出力

出口パラメーター・ブロック。

この構造体には出口の呼び出しに関する情報があります。出口では、ワークロードの管理方法を示すために、この構造体内に情報を設定します。

CurrentRecord (MQPTR) - 入力

現行レコードのアドレス。

この構造体は、出口によって現在調べられているレコードのアドレスに関連した情報を格納します。レコードは、以下のタイプのいずれかでなければなりません。

- クラスター・ワークロードの宛先レコード (MQWDR)
- クラスター・ワークロードのキュー・レコード (MQWQR)
- クラスター・ワークロードのクラスター・レコード (MQWCR)

NextOffset (MQLONG) - 入力

次のレコードのオフセット。

この構造体は、次のレコードまたは構造体のオフセットに関連した情報を格納します。 *NextOffset* は、現行レコード内の該当するオフセット・フィールドの値であり、以下のいずれかのフィールドでなければなりません。

- MQWDR 内の ChannelDefOffset フィールド
- MQWDR 内の ClusterRecOffset フィールド
- MQWQR 内の ClusterRecOffset フィールド
- MQWCR 内の ClusterRecOffset フィールド

NextRecord (MQPTR) - 出力

次のレコードまたは構造体のアドレス。

この構造体は、次のレコードまたは構造体のアドレスに関連した情報を格納します。 *CurrentRecord* が MQWDR のアドレスであり、*NextOffset* が ChannelDefOffset フィールドの値である場合、*NextRecord* はチャンネル定義構造体のアドレス (MQCD) です。

次のレコードまたは構造体がない場合、キュー・マネージャーは *NextRecord* を NULL ポインターに設定し、呼び出しは完了コード MQCC_WARNING と理由コード MQRC_NO_RECORD_AVAILABLE を戻します。

CompCode (MQLONG) - 出力

完了コード。

完了コードには、以下のいずれかの値が設定されます。

MQCC_OK

正常終了。

MQCC_WARNING

警告 (部分完了)。

MQCC_FAILED

呼び出し失敗。

Reason (MQLONG) - 出力

CompCode を限定する理由コード

CompCode が MQCC_OK の場合:

MQRC_NONE

(0, X'0000')

レポートする理由コードはありません。

CompCode が MQCC_WARNING の場合:

MQRC_NO_RECORD_AVAILABLE

(2359, X'0937')

使用可能なレコードはありません。MQXCLWLN 呼び出しが、チェーン内の次のレコードのアドレスを取得するためにクラスター・ワークロード出口から発行されました。現行レコードがチェーン内の最後のレコードです。修正アクション: なし。

CompCode が MQCC_FAILED の場合:

MQRC_CURRENT_RECORD_ERROR

(2357, X'0935')

CurrentRecord パラメーターが正しくありません。MQXCLWLN 呼び出しが、チェーン内の次のレコードのアドレスを取得するためにクラスター・ワークロード出口から発行されました。

CurrentRecord パラメーターで指定されたアドレスは、有効なレコードのアドレスではありません。

CurrentRecord は、宛先レコード、MQWDR、キュー・レコード (MQWQR)、またはクラスター・レコード (MQWCR) のアドレスでなければなりません。クラスター・キャッシュ内に常駐しています。修正アクション: クラスター・ワークロード出口が、クラスター・キャッシュに存在する有効なレコードのアドレスを渡すようにしてください。

MQRC_ENVIRONMENT_ERROR

(2012, X'07DC')

呼び出しが環境内で無効です。MQXCLWLN 呼び出しが発行されましたが、クラスター・ワークロード出口からではありませんでした。

MQRC_NEXT_OFFSET_ERROR

(2358, X'0936')

NextOffset パラメーターが正しくありません。MQXCLWLN 呼び出しが、チェーン内の次のレコードのアドレスを取得するためにクラスター・ワークロード出口から発行されました。

NextOffset パラメーターで指定されたオフセットが無効です。**NextOffset** は、以下のいずれかのフィールドの値でなければなりません。

- MQWDR 内の ChannelDefOffset フィールド
- MQWDR 内の ClusterRecOffset フィールド
- MQWQR 内の ClusterRecOffset フィールド
- MQWCR 内の ClusterRecOffset フィールド

修正アクション: **NextOffset** パラメーターに指定された値が、前述のいずれかのフィールドの値であることを確認してください。

MQRC_NEXT_RECORD_ERROR (2361, X'0939')

NextRecord パラメーターが正しくありません。

MQRC_WXP_ERROR (2356, X'0934')

ワークロード出口パラメーター構造体が無効です。MQXCLWLN 呼び出しが、チェーン内の次のレコードのアドレスを取得するためにクラスター・ワークロード出口から発行されました。以下のいずれかの理由により、ワークロード出口パラメーター構造体 **ExitParms** が無効です。

- パラメーター・ポインターが無効である。有効ではないパラメーター・ポインターを検出することは必ずしも可能ではありません。検出されない場合は、予測不能な結果が生じます。
- 構造 ID フィールドが MQWXP_STRUC_ID ではない。
- Version フィールドが MQWXP_VERSION_2 ではありません。
- キュー・マネージャーによって出口に渡された値が Context フィールドにない。

修正処置: **ExitParms** に指定されたパラメーターが、出口が呼び出されたときに出口に渡された MQWXP 構造体であることを確認してください。

関連資料

[MQXCLWLN - クラスター・ワークロード・レコードのナビゲートの使用上の注意](#)

キャッシュが静的な場合でも、MQXCLWLN を使用してクラスター・レコード全体をナビゲートします。

[MQXCLWLN の言語呼び出し](#)

MQXCLWLN では、C および高水準アセンブラーの 2 つの言語がサポートされています。

MQXCLWLN - クラスター・ワークロード・レコードのナビゲートの使用上の注意

キャッシュが静的な場合でも、MQXCLWLN を使用してクラスター・レコード全体をナビゲートします。

クラスター・キャッシュが動的な場合、MQXCLWLN 呼び出しを使用してレコード全体をナビゲートする必要があります。レコード全体をナビゲートするために単純なポインターとオフセットの演算を使用すると、出口が異常終了します。

クラスター・キャッシュが静的な場合、レコード全体をナビゲートするために MQXCLWLN 呼び出しを使用する必要はありません。キャッシュが静的な場合も、通常 MQXCLWLN を使用してください。この後、ワークロード出口の変更を必要とせずに、クラスター・キャッシュを「動的」に変更することができます。

関連資料

[MQXCLWLN - クラスター・ワークロード・レコードのナビゲート用のパラメーター](#)

MQXCLWLN 呼び出しのパラメーターの説明。

[MQXCLWLN の言語呼び出し](#)

MQXCLWLN では、C および高水準アセンブラーの 2 つの言語がサポートされています。

MQXCLWLN の言語呼び出し

MQXCLWLN では、C および高水準アセンブラーの 2 つの言語がサポートされています。

C 言語での呼び出し

```
MQXCLWLN (&ExitParms, CurrentRecord, NextOffset, &NextRecord, &CompCode, &Reason);
```

パラメーターを次のように宣言します。

```
Typedef struct tagMQXCLWLN {  
    MQWXP    ExitParms;        /* Exit parameter block */  
    MQPTR    CurrentRecord;    /* Address of current record*/  
    MQLONG    NextOffset;      /* Offset of next record */  
    MQPTR    NextRecord;       /* Address of next record or structure */  
    MQLONG    CompCode;        /* Completion code */  
    MQLONG    Reason;          /* Reason code qualifying CompCode */  
};
```

高水準アセンブラー呼び出し

```
CALL MQXCLWLN, (CLWLEXITPARMS, CURRENTRECORD, NEXTOFFSET, NEXTRECORD, COMPCODE, REASON)
```

パラメーターを次のように宣言します。

```
CLWLEXITPARMS CMQWXP, Cluster workload exit parameter block
CURRENTRECORD CMQWDRA, Current record
NEXTOFFSET     DS F    Next offset
NEXTRECORD     DS F    Next record
COMPCODE       DS F    Completion code
REASON         DS F    Reason code qualifying COMPCODE
```

関連資料

[MQXCLWLN - クラスター・ワークロード・レコードのナビゲート用のパラメーター](#)

[MQXCLWLN 呼び出しのパラメーターの説明。](#)

[MQXCLWLN - クラスター・ワークロード・レコードのナビゲートの使用上の注意](#)

キャッシュが静的な場合でも、MQXCLWLN を使用してクラスター・レコード全体をナビゲートします。

MQWXP - クラスター・ワークロード出口のパラメーター構造体

以下の表には、MQWXP - クラスター・ワークロード出口のパラメーター構造体内のフィールドがまとめられています。

表 862. MQWXP のフィールド		
フィールド	説明	参照ページ
<i>StrucId</i>	構造体 ID	StrucId
<i>Version</i>	構造体のバージョン番号	バージョン
<i>ExitId</i>	出口のタイプ	ExitId
<i>ExitReason</i>	出口を呼び出す理由	ExitReason
<i>ExitResponse</i>	出口からの応答	ExitResponse
<i>ExitResponse2</i>	出口からの 2 次応答	ExitResponse2
<i>Feedback</i>	フィードバック・コード	Feedback
<i>Flags</i>	フラグ値。これらのビット・フラグは、書き込まれるメッセージの情報を示すために使用されます	Flags
<i>ExitUserArea</i>	出口ユーザー域	ExitUserArea
<i>ExitData</i>	出口データ	ExitData
<i>MsgDescPtr</i>	メッセージ記述子 (MQMD) のアドレス	MsgDescPtr
<i>MsgBufferPtr</i>	メッセージ・データの一部または全体が入っているバッファのアドレス	MsgBufferPtr
<i>MsgBufferLength</i>	メッセージ・データが入っているバッファの長さ	MsgBufferLength
<i>MsgLength</i>	完全なメッセージの長さ	MsgLength
<i>QName</i>	キューの名前	QName
<i>QMgrName</i>	ローカル・キュー・マネージャーの名前	QMgrName
<i>DestinationCount</i>	有効な宛先の数	DestinationCount
<i>DestinationChosen</i>	選択された宛先	DestinationChosen

表 862. MQWXP のフィールド (続き)		
フィールド	説明	参照ページ
<i>DestinationArrayPtr</i>	宛先レコード (MQWDR) を指すポインタの配列のアドレス	DestinationArrayPtr
<i>QArrayPtr</i>	キュー・レコード (MQWQR) を指すポインタの配列のアドレス	QArrayPtr
注: Version が MQWXP_VERSION_2 より小さい場合、残りのフィールドは無視されます。		
<i>CacheContext</i>	コンテキスト情報	CacheContext
<i>CacheType</i>	クラスター・キャッシュのタイプ	CacheType
注: Version が MQWXP_VERSION_3 より小さい場合、残りのフィールドは無視されます。		
<i>CLWLMRUChannels</i>	許可されたアクティブ・アウトバウンド・クラスター・チャンネルの最大数	CLWLMRUChannels
注: Version が MQWXP_VERSION_4 より小さい場合、残りのフィールドは無視されます。		
<i>pEntryPoints</i>	MQI と DCI の呼び出しを可能にする MQIEP 構造体のアドレス	pEntryPoints

クラスター・ワークロード出口のパラメーター構造体によって、クラスター・ワークロード出口に渡される情報が記述されます。

クラスター・ワークロード出口のパラメーター構造体は、すべてのプラットフォーム上でサポートされます。

また、後方互換性のため、MQWXP1、MQWXP2、および MQWXP3 構造体も用意されています。

関連資料

[MQ_CLUSTER_WORKLOAD_EXIT - 呼び出しの説明](#)

このクラスター・ワークロード出口は、使用可能なキュー・マネージャーにメッセージをルーティングするためにキュー・マネージャーによって呼び出されます。

[MQXCLWLN - クラスター・ワークロードのレコードのナビゲート](#)

MQXCLWLN 呼び出しは、クラスター・キャッシュ内に保管された MQWDR、MQWQR、および MQWCR の各レコードのチェーン全体をナビゲートするために使用します。

[MQWDR - クラスター・ワークロード宛先レコード構造体](#)

以下の表には、MQWDR - クラスター・ワークロードの宛先レコード構造体内のフィールドがまとめられています。

[MQWQR - クラスター・ワークロードのキュー・レコード構造体](#)

以下の表には、MQWQR - クラスター・ワークロードのキュー・レコード構造体内のフィールドがまとめられています。

[MQWCR - クラスター・ワークロードのクラスター・レコード構造体](#)

以下の表には、MQWCR クラスター・ワークロードのレコード構造体内のフィールドがまとめられています。

MQWXP - クラスター・ワークロード出口のパラメーター構造体内のフィールド

MQWXP - クラスター・ワークロード出口のパラメーター構造体内のフィールドの説明

StrucId (MQCHAR4) - 入力

クラスター・ワークロード出口のパラメーター構造体の構造体 ID。

- StrucId 値は MQWXP_STRUC_ID です。
- C プログラミング言語の場合、定数 MQWXP_STRUC_ID_ARRAY も定義されます。これは、MQWXP_STRUC_ID と同じ値です。これは、ストリングではなく文字の配列です。

Version (MQLONG) - 入力

構造体のバージョン番号を示します。Version では、次のいずれかの値を取ります。

MQWXP_VERSION_1

クラスター・ワークロード出口のパラメーター構造体のバージョン 1。

MQWXP_VERSION_1 は、すべての環境でサポートされます。

MQWXP_VERSION_2

クラスター・ワークロード出口のパラメーター構造体のバージョン 2。

MQWXP_VERSION_2 は、以下の環境でサポートされます。

-  AIX
-  IBM i
-  Linux
-  Windows

MQWXP_VERSION_3

クラスター・ワークロード出口のパラメーター構造体のバージョン 3。

MQWXP_VERSION_3 は、以下の環境でサポートされます。

-  AIX
-  IBM i
-  Linux
-  Windows

MQWXP_VERSION_4

クラスター・ワークロード出口のパラメーター構造体のバージョン 4。

MQWXP_VERSION_4 は、以下の環境でサポートされます。

-  AIX
-  IBM i
-  Linux
-  Windows

MQWXP_CURRENT_VERSION

クラスター・ワークロード出口パラメーター構造体の現行バージョン。

ExitId (MQLONG) - 入力

呼び出す出口のタイプを示します。クラスター・ワークロード出口が唯一サポートされる出口です。

- ExitId 値は MQXT_CLUSTER_WORKLOAD_EXIT でなければなりません。

ExitReason (MQLONG) - 入力

クラスター・ワークロード出口を呼び出す理由を示します。ExitReason では、次のいずれかの値を取ります。

MQXR_INIT

出口が初めて呼び出されることを示します。

出口で必要になる可能性がある主記憶などのリソースをすべて獲得して初期化します。

MQXR_TERM

出口が終了されようとしていることを示します。

出口が初期化されてから獲得している可能性がある主記憶などのリソースを、すべて解放します。

MQXR_CLWL_OPEN

MQOPEN によって呼び出されます。

MQXR_CLWL_PUT

MQPUT または MQPUT1 によって呼び出されます。

MQXR_CLWL_MOVE

チャンネル状態が変化した場合に MCA によって呼び出されます。

MQXR_CLWL_REPOS

リポジトリ・マネージャーの PCF メッセージ用の MQPUT または MQPUT1 によって呼び出されます。

MQXR_CLWL_REPOS_MOVE

チャンネル状態が変化した場合に、リポジトリ・マネージャーの PCF メッセージ用の MCA によって呼び出されます。

ExitResponse (MQLONG) - 出力

メッセージの処理を続行するかどうかを示すために ExitResponse を設定します。値は、次のいずれかでなければなりません。

MQXCC_OK

メッセージの処理を正常に続行させます。

- DestinationChosen では、メッセージを送信する宛先を特定します。

MQXCC_SUPPRESS_FUNCTION

メッセージの処理を中止します。

- キュー・マネージャーによって実行される処置は、以下に示すように出口が呼び出された理由によって決まります。

表 863. キュー・マネージャーによって実行される処置	
ExitReason	実行される処置
– MQXR_CLWL_OPEN – MQXR_CLWL_REPOS – MQXR_CLWL_PUT	MQOPEN 呼び出し、MQPUT 呼び出し、または MQPUT1 呼び出しは、完了コード MQCC_FAILED および理由コード MQRC_STOPPED_BY_CLUSTER_EXIT で失敗します。
– MQXR_CLWL_MOVE – MQXR_CLWL_REPOS_MOVE	メッセージが送達不能キューに配置されます。

MQXCC_SUPPRESS_EXIT

現在のメッセージの処理を正常に続行させます。キュー・マネージャーがシャットダウンするまで、出口を再度呼び出さないでください。

キュー・マネージャーでは、ClusterWorkloadExit キュー・マネージャー属性がブランクであるかのように後続のメッセージを処理します。DestinationChosen では、現在のメッセージを送信する宛先を特定します。

他のすべての値

MQXCC_SUPPRESS_FUNCTION が指定されているかのようにメッセージを処理します。

ExitResponse2 (MQLONG) - 入出力

キュー・マネージャーに詳しい情報を提供するには、ExitResponse2 を設定します。

- MQXR2_STATIC_CACHE はデフォルト値で、出口への入り口で設定されます。
- ExitReason の値が MQXR_INIT の場合、出口は ExitResponse2 に以下のいずれかの値を設定できます。

MQXR2_STATIC_CACHE

出口には、静的クラスター・キャッシュが必要です。

- クラスター・キャッシュが静的な場合、出口では MQXCLWLN 呼び出しを使用してクラスター・キャッシュ内にあるレコードのチェーンをナビゲートする必要はありません。
- クラスター・キャッシュが動的な場合、出口ではキャッシュ内のレコードを正しくナビゲートできません。

注: キュー・マネージャーは、出口が ExitResponse フィールドに MQXCC_SUPPRESS_EXIT を戻したかのように、MQXR_INIT 呼び出しからの戻りを処理します。

MQXR2_DYNAMIC_CACHE

出口は、静的キャッシュまたは動的キャッシュのいずれかと作動可能です。

- 出口がこの値を返した場合、出口では MQXCLWLN 呼び出しを使用してクラスター・キャッシュ内にあるレコードのチェーンをナビゲートしなければなりません。

Feedback (MQLONG) - 入力

予約フィールド。値はゼロです。

Flags (MQLONG) - 入力

出力されるメッセージについての情報を示します。

- Flags の値は MQWXP_PUT_BY_CLUSTER_CHL です。メッセージは、クラスター・チャンネル以外からやローカルではなく、クラスター・チャンネルから発信されています。つまり、メッセージは別のクラスター・キュー・マネージャーから届いています。

Reserved (MQLONG) - 入力

予約フィールド。値はゼロです。

ExitUserArea (MQBYTE16) - 入出力

出口の呼び出し間で通信するには、ExitUserArea を設定します。

- ExitUserArea は、出口の最初の呼び出しの前に 2 進ゼロに初期化されます。このフィールドに対して出口によって加えられる変更は、MQCONN 呼び出しと対応する MQDISC 呼び出しとの間で発生する出口の呼び出しごとにすべて保存されます。このフィールドは、MQDISC 呼び出しが発生すると 2 進ゼロにリセットされます。
- 出口の最初の呼び出しは、値 MQXR_INIT を持つ ExitReason フィールドによって示されます。
- 以下の定数が定義されています。

MQXUA_NONE - スtring

MQXUA_NONE_ARRAY - 文字配列

ユーザー情報なし。これらの定数は両方とも、フィールドの長さ分の 2 進ゼロです。

MQ_EXIT_USER_AREA_LENGTH

ExitUserArea の長さ。

ExitData (MQCHAR32) - 入力

ClusterWorkloadData キュー・マネージャー属性の値。この属性に値が定義されていない場合、このフィールドはすべてブランクになります。

- ExitData の長さは MQ_EXIT_DATA_LENGTH によって指定されます。

MsgDescPtr (PMQMD) - 入力

処理されるメッセージのメッセージ記述子 (MQMD) のコピーのアドレス。

- 出口によってメッセージ記述子に加えられた変更は、キュー・マネージャーですべて無視されます。
- ExitReason に以下の値のいずれかが設定されている場合、MsgDescPtr は NULL ポインターに設定され、メッセージ記述子は出口に渡されません。

- MQXR_INIT

- MQXR_TERM

- MQXR_CLWL_OPEN

MsgBufferPtr (PMQVOID) - 入力

メッセージ・データの最初の MsgBufferLength バイトのコピーが含まれたバッファのアドレス。

- 出口によってメッセージ・データに加えられた変更は、キュー・マネージャーですべて無視されます。
 - 以下の場合、メッセージ・データは出口に渡されません。
 - MsgDescPtr が NULL ポインターである。
 - メッセージにデータがない。
 - ClusterWorkloadLength キュー・マネージャー属性がゼロである。
- これらの場合、MsgBufferPtr は NULL ポインターです。

MsgBufferLength (MQLONG) - 入力

出口に渡されるメッセージ・データが含まれたバッファの長さ。

- この長さは、ClusterWorkloadLength キュー・マネージャー属性によって制御されます。
- この長さは、完全なメッセージの長さよりも短い場合があります (MsgLength を参照)。

MsgLength (MQLONG) - 入力

出口に渡される完全なメッセージの長さ。

- MsgBufferLength は、完全なメッセージの長さよりも短い場合があります。
- ExitReason が MQXR_INIT、MQXR_TERM、または MQXR_CLWL_OPEN の場合、MsgLength はゼロです。

QName (MQCHAR48) - 入力

宛先キューの名前。このキューはクラスター・キューです。

- QName の長さは MQ_Q_NAME_LENGTH です。

QMgrName (MQCHAR48) - 入力

クラスター・ワークロード出口を呼び出しているローカル・キュー・マネージャーの名前。

- QMgrName の長さは MQ_Q_MGR_NAME_LENGTH です。

DestinationCount (MQLONG) - 入力

可能な宛先の数。宛先は、宛先キューのインスタンスで、宛先レコードによって記述されます。

- 宛先レコードは MQWDR 構造体です。キューのインスタンスへの有効な経路ごとに、1つの構造体があります。
- MQWDR 構造体は、ポインターの配列によってアドレス指定されます (DestinationArrayPtr を参照)。

DestinationChosen (MQLONG) - 入出力

選択された宛先。

- メッセージを送信する経路とキュー・インスタンスを特定する MQWDR 構造体の番号。
- この値の範囲は、1 から DestinationCount までです。
- DestinationChosen は、出口への入力時にキュー・マネージャーで選択している経路とキュー・インスタンスを示します。出口では、この選択を受け入れることができ、また別の経路とキューのインスタンスを選択することもできます。
- 出口によって設定される値は、1 から DestinationCount までの範囲になければなりません。他の値が返された場合、キュー・マネージャーでは出口への入力時に DestinationChosen の値を使用します。

DestinationArrayPtr (PPMQWDR) - 入力

宛先レコード (MQWDR) を指すポインターの配列のアドレス。

- DestinationCount 個の宛先レコードが存在します。

QArrayPtr (PPMQWR) - 入力

キュー・レコード (MQWR) を指すポインターの配列のアドレス。

- キュー・レコードが使用できる場合、それらは DestinationCount 個存在します。
- キュー・レコードが使用できない場合、QArrayPtr は NULL ポインターです。

注: DestinationCount がゼロより大きい場合でも、QArrayPtr が NULL ポインターの場合があります。

CacheContext (MQPTR) : バージョン 2 - 入力

CacheContext フィールドは、キュー・マネージャーによる使用のために予約されています。 出口では、このフィールドの値を変更してはなりません。

CacheType (MQLONG) : バージョン 2 - 入力

クラスター・キャッシュは、次のいずれかのタイプになります。

MQCLCT_STATIC

キャッシュは静的です。

- キャッシュのサイズは固定されており、キュー・マネージャーの作動に伴って増やすことはできません。
- このタイプのキャッシュ内のレコードをナビゲートするために、MQXCLWLN 呼び出しを使用する必要はありません。

MQCLCT_DYNAMIC

キャッシュは動的です。

- キャッシュのサイズは、変化するクラスター情報を収容するために増やすことができます。
- このタイプのキャッシュ内のレコードをナビゲートするには、MQXCLWLN 呼び出しを使用する必要があります。

CLWLMRUChannels (MQLONG) : バージョン 3 - 入力

クラスター・ワークロード選択アルゴリズムによって使用されると予想される、アクティブ・アウトバウンド・クラスター・チャンネルの最大数を示します。

- CLWLMRUChannels の値は 1 から 999 999 999 までです。

pEntryPoints (PMQIEP) : バージョン 4

MQIEP 構造体のアドレス。これによって、MQI および DCI 呼び出しを実行できます。

関連資料

[MQWXP の初期値および言語ごとの宣言](#)

MQWXP - クラスター・ワークロード出口のパラメーター構造体用の初期値および C と高水準アセンブラー言語の宣言。

MQWXP の初期値および言語ごとの宣言

MQWXP - クラスター・ワークロード出口のパラメーター構造体用の初期値および C と高水準アセンブラー言語の宣言。

表 864. MQWXP のフィールド		
フィールド名	定数の名前	定数の値
StrucId	MQWXP_STRUC_ID	'WXP'
Version	MQWXP_VERSION_2	2
ExitId	なし	0
ExitReason	MQXCC_OK	0
ExitResponse	なし	0
ExitResponse2	なし	0

表 864. MQWXP のフィールド (続き)

フィールド名	定数の名前	定数の値
<i>Flags</i>	なし	0
<i>ExitUserArea</i>	{MQXUA_NONE_ARRAY}	0
<i>ExitData</i>	なし	" "
<i>MsgDescPtr</i>	なし	NULL
<i>MsgBufferPtr</i>	なし	NULL
<i>MsgBufferLength</i>	なし	0
<i>MsgBufferPtr</i>	なし	0
<i>QName</i>	なし	" "
<i>QMgrName</i>	なし	" "
<i>DestinationCount</i>	なし	0
<i>DestinationChosen</i>	なし	0
<i>DestinationArrayPtr</i>	なし	NULL
<i>QArrayPtr</i>	なし	NULL
<i>CacheContext</i>	なし	NULL
<i>CacheType</i>	MQCLCT_DYNAMIC	1
<i>CLWLMRUChannels</i>	なし	0
<i>pEntryPoints</i>	なし	NULL

注:

1. 記号-は、単一のブランク文字を表します。
2. C プログラミング言語では、マクロ変数 MQWXP_DEFAULT にデフォルト値が入っています。このマクロ変数を以下の方法で使用して、構造体のフィールドに初期値を設定します。

```
MQWDR MyWXP = {MQWXP_DEFAULT};
```

C 宣言

```
typedef struct tagMQWXP {
    MQCHAR4   StrucId;           /* Structure identifier */
    MQLONG    Version;           /* Structure version number */
    MQLONG    ExitId;            /* Type of exit */
    MQLONG    ExitReason;        /* Reason for invoking exit */
    MQLONG    ExitResponse;      /* Response from exit */
    MQLONG    ExitResponse2;     /* Reserved */
    MQLONG    Feedback;          /* Reserved */
    MQLONG    Flags;             /* Flags */
    MQBYTE16  ExitUserArea;      /* Exit user area */
    MQCHAR32  ExitData;          /* Exit data */
    PMQMD     MsgDescPtr;        /* Address of message descriptor */
    PMQVOID   MsgBufferPtr;      /* Address of buffer containing some
                                or all of the message data */
    MQLONG    MsgBufferLength;   /* Length of buffer containing message
                                data */
    MQLONG    MsgLength;         /* Length of complete message */
    MQCHAR48  QName;             /* Queue name */
    MQCHAR48  QMgrName;          /* Name of local queue manager */
    MQLONG    DestinationCount;  /* Number of possible destinations */
    MQLONG    DestinationChosen; /* Destination chosen */
}
```

```

PPMQWDR  DestinationArrayPtr; /* Address of an array of pointers to
                                destination records */
PPMQWQR  QArrayPtr;           /* Address of an array of pointers to
                                queue records */

/* version 1 */
MQPTR    CacheContext;        /* Context information */
MQLONG   CacheType;           /* Type of cluster cache */
/* version 2 */
MQLONG   CLWLMRUChannels;     /* Maximum number of most recently
                                used cluster channels */

/* version 3 */
PMQIEP   pEntryPoints;        /* Address of the MQIEP structure */
/* version 4 */
};

```

High Level Assembler

```

MQWXP                                DSECT
MQWXP_STRUCID                        DS    CL4      Structure identifier
MQWXP_VERSION                        DS    F         Structure version number
MQWXP_EXITID                         DS    F         Type of exit
MQWXP_EXITREASON                     DS    F         Reason for invoking exit
MQWXP_EXITRESPONSE                   DS    F         Response from exit
MQWXP_EXITRESPONSE2                 DS    F         Reserved
MQWXP_FEEDBACK                       DS    F         Reserved
MQWXP_RESERVED                       DS    F         Reserved
MQWXP_EXITUSERAREA                   DS    XL16      Exit user area
MQWXP_EXITDATA                       DS    CL32      Exit data
MQWXP_MSGDESCPTR                     DS    F         Address of message
*                                   descriptor
MQWXP_MSGBUFFERPTR                   DS    F         Address of buffer containing
*                                   some or all of the message
*                                   data
MQWXP_MSGBUFFERLENGTH                DS    F         Length of buffer containing
*                                   message data
MQWXP_MSGLENGTH                      DS    F         Length of complete message
MQWXP_QNAME                          DS    CL48      Queue name
MQWXP_QMGRNAME                       DS    CL48      Name of local queue manager
MQWXP_DESTINATIONCOUNT              DS    F         Number of possible
*                                   destinations
MQWXP_DESTINATIONCHOSEN              DS    F         Destination chosen
MQWXP_DESTINATIONARRAYPTR            DS    F         Address of an array of
*                                   pointers to destination
*                                   records
MQWXP_QARRAYPTR                      DS    F         Address of an array of
*                                   pointers to queue records
MQWXP_CACHECONTEXT                   DS    F         Context information
MQWXP_CACHETYPE                      DS    F         Type of cluster cache
MQWXP_CLWLMRUCHANNELS                DS    F         Number of most recently used
*                                   channels for workload balancing

MQWXP_LENGTH                         EQU    *-MQWXP  Length of structure
MQWXP_AREA                           ORG    MQWXP
MQWXP_AREA                           DS      CL(MQWXP_LENGTH)

```

関連資料

[MQWXP - クラスター・ワークロード出口のパラメーター構造体内のフィールド](#)

[MQWXP - クラスター・ワークロード出口のパラメーター構造体内のフィールドの説明](#)

MQWDR - クラスター・ワークロード宛先レコード構造体

以下の表には、MQWDR - クラスター・ワークロードの宛先レコード構造体内のフィールドがまとめられています。

表 865. MQWDR のフィールド		
フィールド	説明	参照ページ
<i>StrucId</i>	構造体 ID	StrucId
<i>Version</i>	構造体のバージョン番号	バージョン
<i>StrucLength</i>	MQWDR 構造体の長さ	StrucLength

表 865. MQWDR のフィールド (続き)		
フィールド	説明	参照ページ
<i>QMgrFlags</i>	キュー・マネージャー・フラグ	QMgrFlags
<i>QMgrIdentifier</i>	キュー・マネージャー ID	QMgrIdentifier
<i>QMgrName</i>	キュー・マネージャー名	QMgrName
<i>ClusterRecOffset</i>	最初のクラスター・レコード (MQWCR) の論理オフセット	ClusterRecOffset
<i>ChannelState</i>	チャネルの状態	ChannelState
<i>ChannelDefOffset</i>	チャネル定義構造体 (MQCD) の論理オフセット	ChannelDefOffset
注: Version が MQWDR_VERSION_2 より小さい場合、残りのフィールドは無視されます。		
<i>DestSeqNumber</i>	チャネル宛先順序番号	DestSeqNumber
<i>DestSeqFactor</i>	加重用のチャネル宛先順序係数	DestSeqFactor

クラスター・ワークロードの宛先レコード構造体には、メッセージの有効な宛先の 1 つに関連する情報が含まれています。宛先キューのインスタンスごとに、1 つのクラスター・ワークロードの宛先レコード構造体があります。

クラスター・ワークロードの宛先レコード構造体は、すべての環境でサポートされます。

また、後方互換性のために、MQWDR1 構造体および MQWDR2 構造体も用意されています。

関連資料

[MQ CLUSTER WORKLOAD EXIT - 呼び出しの説明](#)

このクラスター・ワークロード出口は、使用可能なキュー・マネージャーにメッセージをルーティングするためにキュー・マネージャーによって呼び出されます。

[MQXCLWLN - クラスター・ワークロードのレコードのナビゲート](#)

MQXCLWLN 呼び出しは、クラスター・キャッシュ内に保管された MQWDR、MQWQR、および MQWCR の各レコードのチェーン全体をナビゲートするために使用します。

[MQWXP - クラスター・ワークロード出口のパラメーター構造体](#)

以下の表には、MQWXP - クラスター・ワークロード出口のパラメーター構造体内のフィールドがまとめられています。

[MQWQR - クラスター・ワークロードのキュー・レコード構造体](#)

以下の表には、MQWQR - クラスター・ワークロードのキュー・レコード構造体内のフィールドがまとめられています。

[MQWCR - クラスター・ワークロードのクラスター・レコード構造体](#)

以下の表には、MQWCR クラスター・ワークロードのレコード構造体内のフィールドがまとめられています。

MQWDR - クラスター・ワークロードの宛先レコード構造体内のフィールド

MQWDR - クラスター・ワークロードの宛先レコード構造体内のパラメーターの説明。

StrucId (MQCHAR4) - 入力

クラスター・ワークロードの宛先レコード構造体の構造体 ID。

- StrucId 値は MQWDR_STRUC_ID です。
- C プログラミング言語の場合、定数 MQWDR_STRUC_ID_ARRAY も定義されます。これは、MQWDR_STRUC_ID と同じ値です。これは、ストリングではなく文字の配列です。

Version (MQLONG) - 入力

構造体のバージョン番号。Version では、次のいずれかの値を取ります。

MQWDR_VERSION_1

クラスター・ワークロードの宛先レコードのバージョン 1。

MQWDR_VERSION_2

クラスター・ワークロードの宛先レコードのバージョン 2。

MQWDR_CURRENT_VERSION

クラスター・ワークロードの宛先レコードの現行バージョン。

StrucLength (MQLONG) - 入力

MQWDR 構造体の長さ。StrucLength では、次のいずれかの値を取ります。

MQWDR_LENGTH_1

クラスター・ワークロードの宛先レコードのバージョン 1 の長さ。

MQWDR_LENGTH_2

クラスター・ワークロードの宛先レコードのバージョン 2 の長さ。

MQWDR_CURRENT_LENGTH

クラスター・ワークロードの宛先レコードの現行バージョンの長さ。

QMgrFlags (MQLONG) - 入力

MQWDR 構造体によって記述された宛先キューのインスタンスをホストするキュー・マネージャーのプロパティを示すキュー・マネージャー・フラグ。以下のフラグが定義されます。

MQQMF_REPOSITORY_Q_MGR

宛先は、フル・リポジトリのキュー・マネージャーです。

MQQMF_CLUSSDR_USER_DEFINED

クラスター送信側チャンネルが手動で定義されました。

MQQMF_CLUSSDR_AUTO_DEFINED

クラスター送信側チャンネルが自動で定義されました。

MQQMF_AVAILABLE

宛先キュー・マネージャーが、メッセージを受信できます。

その他の値

このフィールドには、内部目的のためにキュー・マネージャーによって他のフラグが設定される場合があります。

QMgrIdentifier (MQCHAR48) - 入力

キュー・マネージャー ID とは、MQWDR 構造体によって記述された宛先キューのインスタンスをホストするキュー・マネージャーの固有 ID です。

- この ID は、キュー・マネージャーによって生成されます。
- QMgrIdentifier の長さは MQ_Q_MGR_IDENTIFIER_LENGTH です。

QMgrName (MQCHAR48) - 入力

MQWDR 構造体によって記述された宛先キューのインスタンスをホストするキュー・マネージャーの名前。

- QMgrName は、ローカル・キュー・マネージャーおよびクラスター内の別のキュー・マネージャーの名前にすることができます。
- QMgrName の長さは MQ_Q_MGR_NAME_LENGTH です。

ClusterRecOffset (MQLONG) - 入力

MQWDR 構造体に属している最初の MQWCR 構造体の論理オフセット。

- 静的キャッシュの場合、ClusterRecOffset は、MQWDR 構造体に属している最初の MQWCR 構造体のオフセットです。
- オフセット (相対位置) は、MQWDR 構造体の先頭からバイト単位で測定されます。
- 論理オフセットは、動的キャッシュのポインター演算に使用しないでください。次のレコードのアドレスを取得するには、MQXCLWLN 呼び出しを使用する必要があります。

ChannelState (MQLONG) - 入力

MQWDR 構造体で特定されたキュー・マネージャーにローカル・キュー・マネージャーをリンクするチャンネルの状態。属性の値は以下のとおりです。

MQCHS_BINDING

チャンネルはパートナーと折衝中です。

MQCHS_INACTIVE

チャンネルはアクティブではありません。

MQCHS_INITIALIZING

チャンネルは初期化中です。

MQCHS_PAUSED

チャンネルは一時停止しています。

MQCHS_REQUESTING

要求側チャンネルが接続を要求しています。

MQCHS_RETRYING

チャンネルは接続の確立を再試行しています。

MQCHS_RUNNING

チャンネルはメッセージの転送中またはメッセージ待ちの状態です。

MQCHS_STARTING

チャンネルはアクティブになるのを待っています。

MQCHS_STOPPING

チャンネルは停止中です。

MQCHS_STOPPED

チャンネルは停止しました。

ChannelDefOffset (MQLONG) - 入力

MQWDR 構造体で特定されたキュー・マネージャーにローカル・キュー・マネージャーをリンクするチャンネルのチャンネル定義 (MQCD) の論理オフセット。

- ChannelDefOffset は ClusterRecOffset に類似しています
- 論理オフセットはポインター演算に使用することはできません。次のレコードのアドレスを取得するには、MQXCLWLN 呼び出しを使用する必要があります。

DestSeqFactor (MQLONG) - 入力

重みづけに基づいたチャンネルの選択を可能にする宛先順序係数。

- DestSeqFactor が使用されると、キュー・マネージャーはそれを変更します。
- ワークロード・マネージャーでは、メッセージがチャンネルの重みづけに従って各チャンネルに分散されるようにする方法で DestSeqFactor を増やします。

DestSeqNumber (MQLONG) - 入力

キュー・マネージャーが変更する前のクラスター・チャンネル宛先の値。

- メッセージがこのチャンネルに書き込まれるたびに、ワークロード・マネージャーでは DestSeqNumber を増やします。
- ワークロード出口では、メッセージを書き込むチャンネルを決定するために DestSeqNumber を使用できます。

関連資料

MQWDR の初期値および言語ごとの宣言

MQWDR - クラスター・ワークロードの宛先レコード用の初期値および C と高水準アセンブラー言語の宣言。

MQWDR の初期値および言語ごとの宣言

MQWDR - クラスター・ワークロードの宛先レコード用の初期値および C と高水準アセンブラー言語の宣言。

表 866. MQWDR のフィールド

フィールド名	定数の名前	定数の値
<i>StrucId</i>	MQWDR_STRUC_ID	'WDR↵'
<i>Version</i>	MQWDR_VERSION_1	1
<i>StrucLength</i>	MQWDR_CURRENT_LENGTH ³	136
<i>QMgrFlags</i>	MQWDR_NONE	0
<i>QMgrIdentifier</i>	なし	" "
<i>QMgrName</i>	なし	" "
<i>ClusterRecOffset</i>	なし	0
<i>ChannelState</i>	なし	0
<i>ChannelDefOffset</i>	なし	0
<i>DestSeqNumber</i>	なし	0
<i>DestSeqFactor</i>	なし	0

注:

1. 記号↵は、単一のブランク文字を表します。
2. C プログラミング言語では、マクロ変数 MQWDR_DEFAULT にデフォルト値が入っています。このマクロ変数を以下の方法で使用して、構造体のフィールドに初期値を設定します。

```
MQWDR MyWDR = {MQWDR_DEFAULT};
```

3. 初期値では、構造体の長さを構造体のバージョン 1 ではなく、現行バージョンの長さに意図的に設定します。

High Level Assembler

MQWDR	DSECT	
MQWDR_STRUCID	DS	CL4 Structure identifier
MQWDR_VERSION	DS	F Structure version number
MQWDR_STRUCLNGTH	DS	F Length of MQWDR structure
MQWDR_QMGRFLAGS	DS	F Queue manager flags
MQWDR_QMGRIDENTIFIER	DS	CL48 Queue manager identifier
MQWDR_QMGRNAME	DS	CL48 Queue manager name
MQWDR_CLUSTERRECOFFSET	DS	F Offset of first cluster
*		record
MQWDR_CHANNELSTATE	DS	F Channel state
MQWDR_CHANNELDEFOFFSET	DS	F Offset of channel definition
*		structure
MQWDR_LENGTH	EQU	*-MQWDR Length of structure
	ORG	MQWDR
MQWDR_AREA	DS	CL(MQWDR_LENGTH)

C 宣言

```
typedef struct tagMQWDR {
    MQCHAR4   StrucId;           /* Structure identifier */
    MQLONG    Version;           /* Structure version number */
    MQLONG    StrucLength;       /* Length of MQWDR structure */
    MQLONG    QMgrFlags;         /* Queue manager flags */
    MQCHAR48   QMgrIdentifier;   /* Queue manager identifier */
    MQCHAR48   QMgrName;         /* Queue manager name */
    MQLONG    ClusterRecOffset;  /* Offset of first cluster record */
    MQLONG    ChannelState;      /* Channel state */
    MQLONG    ChannelDefOffset;  /* Offset of channel definition structure */
    /* Ver:1 */
    MQLONG    DestSeqNumber;     /* Cluster channel destination sequence number */
}
```

```

MQINT64   DestSeqFactor;      /* Cluster channel factor sequence number */
/* Ver:2 */
};

```

関連資料

[MQWDR - クラスター・ワークロードの宛先レコード構造体内のフィールド](#)

[MQWDR - クラスター・ワークロードの宛先レコード構造体内のパラメーターの説明。](#)

MQWQR - クラスター・ワークロードのキュー・レコード構造体

以下の表には、MQWQR - クラスター・ワークロードのキュー・レコード構造体内のフィールドがまとめられています。

表 867. MQWQR のフィールド		
フィールド	説明	参照ページ
<i>StrucId</i>	構造体 ID	StrucId
<i>Version</i>	構造体のバージョン番号	バージョン
<i>StrucLength</i>	MQWQR 構造体の長さ	StrucLength
<i>QFlags</i>	キュー・フラグ	QFlags
<i>QName</i>	キュー名	QName
<i>QMgrIdentifier</i>	キュー・マネージャー ID	QMgrIdentifier
<i>ClusterRecOffset</i>	最初のクラスター・レコード (MQWCR) のオフセット	ClusterRecOffset
<i>QType</i>	キュー・タイプ	QType
<i>QDesc</i>	キューの記述	QDesc
<i>DefBind</i>	デフォルトのバインド	DefBind
<i>DefPersistence</i>	デフォルトのメッセージ持続性	DefPersistence
<i>DefPriority</i>	デフォルトのメッセージ優先順位	DefPriority
<i>InhibitPut</i>	キューへの PUT 操作が許可されるかどうか	InhibitPut
注: Version が MQWQR_VERSION_2 より小さい場合、残りのフィールドは無視されます。		
<i>CWLQueuePriority</i>	キューの優先順位を表す 0 から 9 までの値	CLWLQueuePriority
<i>CLWLQueueRank</i>	キューのランクを表す 0 から 9 までの値	CLWLQueueRank
注: Version が MQWQR_VERSION_3 より小さい場合、残りのフィールドは無視されます。		
<i>DefPutResponse</i>	デフォルトの書き込み応答	DefPutResponse

クラスター・ワークロードのキュー・レコード構造体には、メッセージの有効な宛先の 1 つに関連する情報が含まれています。宛先キューのインスタンスごとに、1 つのクラスター・ワークロードのキュー・レコード構造体があります。

クラスター・ワークロードのキュー・レコード構造体は、すべての環境でサポートされます。

また、後方互換性のために、MQWQR1 構造体および MQWQR2 構造体も用意されています。

関連資料

[MQ_CLUSTER_WORKLOAD_EXIT - 呼び出しの説明](#)

このクラスター・ワークロード出口は、使用可能なキュー・マネージャーにメッセージをルーティングするためにキュー・マネージャーによって呼び出されます。

[MQXCLWLN - クラスター・ワークロードのレコードのナビゲート](#)

MQXCLWLN 呼び出しは、クラスター・キャッシュ内に保管された MQWDR、MQWQR、および MQWCR の各レコードのチェーン全体をナビゲートするために使用します。

MQWXP - クラスター・ワークロード出口のパラメーター構造体

以下の表には、MQWXP - クラスター・ワークロード出口のパラメーター構造体内のフィールドがまとめられています。

MQWDR - クラスター・ワークロード宛先レコード構造体

以下の表には、MQWDR - クラスター・ワークロードの宛先レコード構造体内のフィールドがまとめられています。

MQWCR - クラスター・ワークロードのクラスター・レコード構造体

以下の表には、MQWCR クラスター・ワークロードのレコード構造体内のフィールドがまとめられています。

MQWQR - クラスター・ワークロードのキュー・レコード構造体内のフィールド

MQWQR - クラスター・ワークロードのキュー・レコード構造体内のフィールドの説明。

StrucId (MQCHAR4) - 入力

クラスター・ワークロードのキュー・レコード構造体の構造体 ID。

- StrucId 値は MQWQR_STRUC_ID です。
- C プログラミング言語の場合、定数 MQWQR_STRUC_ID_ARRAY も定義されます。これは、MQWQR_STRUC_ID と同じ値です。これは、ストリングではなく文字の配列です。

Version (MQLONG) - 入力

構造体のバージョン番号。Version では、次のいずれかの値を取ります。

MQWQR_VERSION_1

クラスター・ワークロードのキュー・レコードのバージョン 1。

MQWQR_VERSION_2

クラスター・ワークロードのキュー・レコードのバージョン 2。

MQWQR_VERSION_3

クラスター・ワークロードのキュー・レコードのバージョン 3。

MQWQR_CURRENT_VERSION

クラスター・ワークロードのキュー・レコードの現行バージョン。

StrucLength (MQLONG) - 入力

MQWQR 構造体の長さ。StrucLength では、次のいずれかの値を取ります。

MQWQR_LENGTH_1

クラスター・ワークロードのキュー・レコードのバージョン 1 の長さ。

MQWQR_LENGTH_2

クラスター・ワークロードのキュー・レコードのバージョン 2 の長さ。

MQWQR_LENGTH_3

クラスター・ワークロードのキュー・レコードのバージョン 3 の長さ。

MQWQR_CURRENT_LENGTH

クラスター・ワークロードのキュー・レコードの現行バージョンの長さ。

QFlags (MQLONG) - 入力

キュー・フラグはキューのプロパティを示します。以下のフラグが定義されます。

MQQF_LOCAL_Q

宛先はローカル・キューです。

MQQF_CLWL_USEQ_ANY

ローカルおよびリモート・キューを書き込みに使用することを許可します。

MQQF_CLWL_USEQ_LOCAL

ローカル・キューへの書き込みのみを許可します。

その他の値

このフィールドには、内部目的のためにキュー・マネージャーによって他のフラグが設定される場合があります。

QName (MQCHAR48) - 入力

メッセージの有効な宛先の 1 つであるキューの名前。

- QName の長さは MQ_Q_NAME_LENGTH です。

QMgrIdentifier (MQCHAR48) - 入力

キュー・マネージャー ID とは、MQWQR 構造体によって記述されたキューのインスタンスをホストするキュー・マネージャーの固有 ID です。

- この ID は、キュー・マネージャーによって生成されます。
- QMgrIdentifier の長さは MQ_Q_MGR_IDENTIFIER_LENGTH です。

ClusterRecOffset (MQLONG) - 入力

MQWQR 構造体に属している最初の MQWCR 構造体の論理オフセット。

- 静的キャッシュの場合、ClusterRecOffset は、MQWQR 構造体に属している最初の MQWCR 構造体のオフセットです。
- オフセット (相対位置) は、MQWQR 構造体の先頭からバイト単位で測定されます。
- 論理オフセットは、動的キャッシュのポインター演算に使用しないでください。次のレコードのアドレスを取得するには、MQXCLWLN 呼び出しを使用する必要があります。

QType (MQLONG) - 入力

宛先キューのキュー・タイプ。属性の値は以下のとおりです。

MQCQT_LOCAL_Q

ローカル・キュー。

MQCQT_ALIAS_Q

別名キュー。

MQCQT_REMOTE_Q

リモート・キュー。

MQCQT_Q_MGR_ALIAS

キュー・マネージャー別名。

QDesc (MQCHAR64) - 入力

MQWQR 構造体によって記述された宛先キューのインスタンスをホストするキュー・マネージャーに定義されたキューの記述のキュー属性。

- QDesc の長さは MQ_Q_DESC_LENGTH です。

DefBind (MQLONG) - 入力

MQWQR 構造体によって記述された宛先キューのインスタンスをホストするキュー・マネージャーに定義されたデフォルトのバインディングのキュー属性。クラスタとグループを併用する場合は、`either` または MQBND_BIND_ON_OPEN `or` MQBND_BIND_ON_GROUP ` のいずれかを指定する必要があります。属性の値は以下のとおりです。

MQBND_BIND_ON_OPEN

MQOPEN 呼び出しで固定されたバインディング。

MQBND_BIND_NOT_FIXED

固定されていないバインディング。

MQBND_BIND_ON_GROUP

グループ内のメッセージすべてを同じ宛先のインスタンスに割り振る要求をアプリケーションが行えるようになります。

DefPersistence (MQLONG) - 入力

MQWQR 構造体によって記述された宛先キューのインスタンスをホストするキュー・マネージャーに定義されたデフォルトのメッセージ持続性のキュー属性。属性の値は以下のとおりです。

MQPER_PERSISTENT

メッセージは持続します。

MQPER_NOT_PERSISTENT

メッセージは持続しません。

DefPriority (MQLONG) - 入力

MQWQR 構造体によって記述された宛先キューのインスタンスをホストするキュー・マネージャーに定義されたデフォルトのメッセージ優先順位のキュー属性。優先順位の範囲は、0 から MaxPriority までです。

- 0 が最も低い優先順位です。
- MaxPriority は、宛先キューのこのインスタンスをホストするキュー・マネージャーのキュー・マネージャー属性です。

InhibitPut (MQLONG) - 入力

MQWQR 構造体によって記述された宛先キューのインスタンスをホストするキュー・マネージャーに定義された書き込み禁止のキュー属性。属性の値は以下のとおりです。

MQQA_PUT_INHIBITED

書き込み操作は使用禁止です。

MQQA_PUT_ALLOWED

書き込み操作が許可されています。

CLWLQueuePriority (MQLONG) - 入力

MQWQR 構造体によって記述された宛先キューのインスタンスをホストするキュー・マネージャーに定義されたクラスター・ワークロード・キュー優先順位属性。

CLWLQueueRank (MQLONG) - 入力

MQWQR 構造体によって記述された宛先キューのインスタンスをホストするキュー・マネージャーに定義されたクラスター・ワークロード・キュー・ランク。

DefPutResponse (MQLONG) - 入力

MQWQR 構造体によって記述された宛先キューのインスタンスをホストするキュー・マネージャーに定義されたデフォルトの書き込み応答キュー属性。属性の値は以下のとおりです。

MQPRT_SYNC_RESPONSE

MQPUT 呼び出しまたは MQPUT1 呼び出しへの同期応答。

MQPRT_ASYNC_RESPONSE

MQPUT 呼び出しまたは MQPUT1 呼び出しへの非同期応答。

関連資料

[MQWQR の初期値および言語ごとの宣言](#)

MQWQR - クラスター・ワークロードのキュー・レコード用の初期値および C と高水準アセンブラー言語の宣言。

MQWQR の初期値および言語ごとの宣言

MQWQR - クラスター・ワークロードのキュー・レコード用の初期値および C と高水準アセンブラー言語の宣言。

表 868. MQWQR のフィールド		
フィールド名	定数の名前	定数の値
StrucId	MQWQR_STRUC_ID_ARRAY	'WQR'
Version	MQWQR_VERSION_1	1
StrucLength	MQWQR_CURRENT_LENGTH ³	212
QFlags	なし	0
QName	なし	" "

表 868. MQWQR のフィールド (続き)

フィールド名	定数の名前	定数の値
<i>QMgrIdentifier</i>	なし	" "
<i>ClusterRecOffset</i>	なし	0
<i>QType</i>	なし	0
<i>QDesc</i>	なし	" "
<i>DefBind</i>	なし	0
<i>DefPersistence</i>	なし	0
<i>DefPriority</i>	なし	0
<i>InhibitPut</i>	なし	0
<i>CLWLQueuePriority</i>	なし	0
<i>CLWLQueueRank</i>	なし	0
<i>DefPutResponse</i>	なし	1

注:

1. 記号-は、単一のブランク文字を表します。
2. C プログラミング言語では、マクロ変数 `MQWQR_DEFAULT` にデフォルト値が入っています。このマクロ変数を以下の方法で使用して、構造体のフィールドに初期値を設定します。

```
MQWQR MyWQR = {MQWQR_DEFAULT};
```

3. 初期値では、構造体の長さを構造体のバージョン 1 ではなく、現行バージョンの長さに意図的に設定します。

C 宣言

```
typedef struct tagMQWQR {
    MQCHAR4   StrucId;           /* Structure identifier */
    MQLONG    Version;           /* Structure version number */
    MQLONG    StrucLength;       /* Length of MQWQR structure */
    MQLONG    QFlags;            /* Queue flags */
    MQCHAR48   QName;            /* Queue name */
    MQCHAR48   QMgrIdentifier;    /* Queue manager identifier */
    MQLONG    ClusterRecOffset;   /* Offset of first cluster record */
    MQLONG    QType;             /* Queue type */
    MQCHAR64   QDesc;            /* Queue description */
    MQLONG    DefBind;           /* Default binding */
    MQLONG    DefPersistence;     /* Default message persistence */
    MQLONG    DefPriority;        /* Default message priority */
    MQLONG    InhibitPut;        /* Whether put operations on the queue
                                are allowed */

    /* version 2 */
    MQLONG    CLWLQueuePriority; /* Queue priority */
    MQLONG    CLWLQueueRank;    /* Queue rank */
    /* version 3 */
    MQLONG    DefPutResponse;    /* Default put response */
};
```

High Level Assembler

MQWQR	DSECT	
MQWQR_STRUCID	DS	CL4 Structure identifier
MQWQR_VERSION	DS	F Structure version number
MQWQR_STRUCLNGTH	DS	F Length of MQWQR structure
MQWQR_QFLAGS	DS	F Queue flags
MQWQR_QNAME	DS	CL48 Queue name

MQWQR_QMGRIDENTIFIER	DS	CL48	Queue manager identifier
MQWQR_CLUSTERRECOFFSET	DS	F	Offset of first cluster record
* MQWQR_QTYPE	DS	F	Queue type
MQWQR_QDESC	DS	CL64	Queue description
MQWQR_DEFBIND	DS	F	Default binding
MQWQR_DEFPERSISTENCE	DS	F	Default message persistence
MQWQR_DEFPRIORITY	DS	F	Default message priority
MQWQR_INHIBITPUT	DS	F	Whether put operations on the queue are allowed
* MQWQR_DEFPUTRESPONSE	DS	F	Default put response
MQWQR_LENGTH	EQU	*-MQWQR	Length of structure
	ORG	MQWQR	
MQWQR_AREA	DS	CL(MQWQR_LENGTH)	

関連資料

[MQWQR - クラスター・ワークロードのキュー・レコード構造体内のフィールド](#)

MQWQR - クラスター・ワークロードのキュー・レコード構造体内のフィールドの説明。

MQWCR - クラスター・ワークロードのクラスター・レコード構造体

以下の表には、MQWCR クラスター・ワークロードのレコード構造体内のフィールドがまとめられています。

表 869. MQWCR のフィールド		
フィールド	説明	参照ページ
<i>ClusterName</i>	クラスターの名前	ClusterName
<i>ClusterRecOffset</i>	次のクラスター・レコード (MQWCR) のオフセット	ClusterRecOffset
<i>ClusterFlags</i>	クラスター・フラグ	ClusterFlags

クラスター・ワークロードのクラスター・レコード構造体には、クラスターに関する情報が含まれています。宛先キューが属しているクラスターごとに、1つのクラスター・ワークロードのクラスター・レコード構造体があります。

クラスター・ワークロードのクラスター・レコード構造体は、すべての環境でサポートされます。

関連資料

[MQ_CLUSTER_WORKLOAD_EXIT - 呼び出しの説明](#)

このクラスター・ワークロード出口は、使用可能なキュー・マネージャーにメッセージをルーティングするためにキュー・マネージャーによって呼び出されます。

[MQXCLWLN - クラスター・ワークロードのレコードのナビゲート](#)

MQXCLWLN 呼び出しは、クラスター・キャッシュ内に保管された MQWDR、MQWQR、および MQWCR の各レコードのチェーン全体をナビゲートするために使用します。

[MQWXP - クラスター・ワークロード出口のパラメーター構造体](#)

以下の表には、MQWXP - クラスター・ワークロード出口のパラメーター構造体内のフィールドがまとめられています。

[MQWDR - クラスター・ワークロード宛先レコード構造体](#)

以下の表には、MQWDR - クラスター・ワークロードの宛先レコード構造体内のフィールドがまとめられています。

[MQWQR - クラスター・ワークロードのキュー・レコード構造体](#)

以下の表には、MQWQR - クラスター・ワークロードのキュー・レコード構造体内のフィールドがまとめられています。

MQWCR - クラスター・ワークロードのクラスター・レコード構造体内のフィールド。

MQWCR - クラスター・ワークロードのクラスター・レコード構造体内のフィールドの説明。

ClusterName (MQCHAR48) - 入力

MQWCR 構造体を所有する宛先キューのインスタンスが属しているクラスターの名前。宛先キューのインスタンスは、MQWDR 構造体によって記述されます。

- `ClusterName` の長さは `MQ_CLUSTER_NAME_LENGTH` です。

ClusterRecOffset (MQLONG) - 入力

次の MQWCR 構造体の論理オフセット。

- それ以上 MQWCR 構造体がない場合、`ClusterRecOffset` はゼロです。
- オフセット (相対位置) は、MQWCR 構造体の先頭からバイト単位で測定されます。

ClusterFlags (MQLONG) - 入力

クラスター・フラグは、MQWCR 構造体によって特定されたキュー・マネージャーのプロパティーを示します。以下のフラグが定義されます。

MQQMF_REPOSITORY_Q_MGR

宛先は、フル・リポジトリのキュー・マネージャーです。

MQQMF_CLUSSDR_USER_DEFINED

クラスター送信側チャンネルが手動で定義されました。

MQQMF_CLUSSDR_AUTO_DEFINED

クラスター送信側チャンネルが自動で定義されました。

MQQMF_AVAILABLE

宛先キュー・マネージャーが、メッセージを受信できます。

その他の値

このフィールドには、内部目的のためにキュー・マネージャーによって他のフラグが設定される場合があります。

関連資料

[MQWCR の初期値および言語ごとの宣言](#)

MQWCR - クラスター・ワークロードのクラスター・レコード構造体用の初期値および C と高水準アセンブラー言語の宣言。

MQWCR の初期値および言語ごとの宣言

MQWCR - クラスター・ワークロードのクラスター・レコード構造体用の初期値および C と高水準アセンブラー言語の宣言。

表 870. MQWCR のフィールド		
フィールド名	定数の名前	定数の値
<code>ClusterName</code>	なし	" "
<code>ClusterRecOffset</code>	なし	0
<code>ClusterFlags</code>	なし	0

C 宣言

```
typedef struct tagMQWCR {
    MQCHAR48  ClusterName;          /* Cluster name */
    MQLONG    ClusterRecOffset;     /* Offset of next cluster record */
    MQLONG    ClusterFlags;         /* Cluster flags */
};
```

High Level Assembler

```
MQWCR
MQWCR_CLUSTERNAME      DSECT
MQWCR_CLUSTERRECOFFSET DS    CL48      Cluster name
*                        DS    F        Offset of next cluster
MQWCR_CLUSTERFLAGS     DS    F        record
MQWCR_LENGTH           EQU    *-MQWCR  Cluster flags
                        ORG    MQWCR    Length of structure
MQWCR_AREA             DS    CL(MQWCR_LENGTH)
```


関連資料

MQWCR - クラスター・ワークロードのクラスター・レコード構造体内のフィールド。

MQWCR - クラスター・ワークロードのクラスター・レコード構造体内のフィールドの説明。

API 出口参照

このセクションでは、API 出口を作成するプログラマーを対象とした参照情報を提供します。

一般的な使用上の注意

注：

1. MQXEP 呼び出しは、すべての出口関数で発行できます。この呼び出しは、特に API 出口関数で使用するために設計された呼び出しです。
2. MQ_INIT_EXIT 関数では、MQXEP 以外の MQ 呼び出しは発行できません。
3. 現在の接続に対する MQDISC 呼び出しは発行できません。
4. 出口関数が MQCONN 呼び出しまたは MQCNO_HANDLE_SHARE_NONE オプションを指定して MQCONN 呼び出しを発行すると、理由コード MQRC_ALREADY_CONNECTED で呼び出しが完了し、出口にパラメーターとして渡されたハンドルと同じハンドルが戻されます。
5. 通常、API 出口関数で MQI 呼び出しを発行した場合、API 出口は再帰的に呼び出されません。ただし、出口関数で、MQCNO_HANDLE_SHARE_BLOCK オプションまたは MQCNO_HANDLE_SHARE_NO_BLOCK オプションを指定して MQCONN 呼び出しを発行すると、この呼び出しによって新規の共有ハンドルが戻されます。これによって、その出口スイートには独自の接続ハンドルができ、さらにはアプリケーションの作業単位とは別の作業単位が提供されます。出口スイートはこのハンドルを用いて、自身の作業単位内でのメッセージの書き込みと読み取り、およびその作業単位のコミットとバックアウトを行うことができます。これらすべてを、アプリケーションの作業単位にはまったく影響を与えることなく行うことが可能です。

出口関数では、アプリケーションで使用されているハンドルとは別の接続ハンドルが使用されているため、その出口関数によって MQ 呼び出しが発行される場合は、関係する API 出口関数が呼び出されることになります。つまり、出口関数は再帰的に呼び出すことができます。ただし、MQAXP の *ExitUserArea* フィールドと出口チェーン領域にはいずれも接続ハンドルの有効範囲があることに注意してください。このため、出口関数では、再帰的に呼び出され、既に活動状態になっている自身の別のインスタンスに、これらの領域を使用して信号を送ることはできません。

6. 出口関数では、アプリケーションの作業単位内でメッセージの書き込みや読み取りを行うこともできます。その場合、アプリケーションが作業単位をコミットまたはバックアウトすると、アプリケーションで使用されているメッセージであっても出口関数で使用されているメッセージであっても、作業単位内のメッセージがすべて同時にコミットまたはバックアウトされます。ただし、このように出口が作業単位を使用することによって、通常よりも早くアプリケーションがシステムしきい値を超えてしまう場合 (例えば、作業単位内に含まれるコミットされていないメッセージが最大数を超えてしまうなど) もあります。

このようにアプリケーションの作業単位を使用する際、出口関数は、MQCMIT 呼び出しを出すとアプリケーションの作業単位をコミットしてしまい、アプリケーションの正常な機能を損なうおそれがあるため、通常はこの呼び出しの発行を控えます。ただし、作業単位のコミットを妨げる深刻なエラー (例えば、アプリケーションの作業単位の一部としてメッセージを書き込む際のエラーなど) を出口関数が検出した場合は、出口関数が MQBACK 呼び出しを発行しなければならないこともあります。MQBACK が呼び出された場合は、アプリケーションの作業単位境界が変更されないように注意してください。この場合、出口関数では、アプリケーションに完了コード MQCC_WARNING と理由コード MQRC_BACKED_OUT が戻されるような適切な値を設定して、作業単位がバックアウトされたことをアプリケーションが検出できるようにする必要があります。

出口関数がアプリケーションの接続ハンドルを使用して MQ 呼び出しを発行する場合は、これらの呼び出しそのものによってさらに別の API 出口関数が呼び出されることはありません。

7. MQXR_BEFORE 出口関数が異常終了しても、キュー・マネージャーが障害からリカバリーできる場合があります。リカバリーが可能な場合、キュー・マネージャーは、出口関数が MQXCC_FAILED を戻し

たかのように処理を続行します。キュー・マネージャーがリカバリーできない場合、アプリケーションは終了します。

8. MQXR_AFTER 出口関数が異常終了しても、キュー・マネージャーが障害からリカバリーできる場合があります。リカバリーが可能な場合、キュー・マネージャーは、出口関数が MQXCC_FAILED を戻したかのように処理を続行します。キュー・マネージャーがリカバリーできない場合、アプリケーションは終了します。後者の場合、作業単位の外に取り出されたメッセージは失われます(これは、キューからメッセージを除去した直後にアプリケーションで障害が発生した状況と同じです)。

9. MCA プロセスでは、2 フェーズ・コミットが実行されます。

API 出口が準備済み MCA プロセスから MQCMIT をインターセプトし、その作業単位内でアクションを実行しようとする、そのアクションは理由コード MQRC_UOW_NOT_AVAILABLE で失敗します。

10. IBM MQ の複数インストール済み環境が使用可能な場合、後のバージョンで追加された新機能は前のバージョンでは動作できないので、以前のバージョンの IBM MQ 用に記述された出口を使用します。リリース間の変更点の詳細については、[IBM MQ 8.0 での変更点](#)を参照してください。

IBM MQ API 出口パラメーター構造体 (MQAXP)

MQAXP 構造体、外部制御ブロックは、API 出口に対する入出力パラメーターとして使用されます。このトピックでは、さらに、キュー・マネージャーが出口関数を処理する方法についても情報を提供します。

MQAXP には、次の C 宣言があります。

```
typedef struct tagMQAXP {
    MQCHAR4    StructId;           /* Structure identifier */
    MQLONG     Version;            /* Structure version number */
    MQLONG     ExitId;             /* Exit Identifier */
    MQLONG     ExitReason;         /* Exit invocation reason */
    MQLONG     ExitResponse;       /* Response code from exit */
    MQLONG     ExitResponse2;      /* Secondary response code from exit */
    MQLONG     Feedback;           /* Feedback code from exit */
    MQLONG     APICallerType;      /* MQSeries API caller type */
    MQBYTE16   ExitUserArea;       /* User area for use by exit */
    MQCHAR32   ExitData;           /* Exit data area */
    MQCHAR48   ExitInfoName;       /* Exit information name */
    MQBYTE48   ExitPDArea;         /* Problem determination area */
    MQCHAR48   QMgrName;           /* Name of local queue manager */
    PMQACH     ExitChainAreaPtr;   /* Inter exit communication area */
    MQHCONFIG   Hconfig;           /* Configuration handle */
    MQLONG     Function;           /* Function Identifier */
    /* Ver:1 */
    MQHMSG     ExitMsgHandle       /* Exit message handle
    /* Ver:2 */
};
```

API 出口の関数が呼び出されるとき、次のパラメーター・リストが引き渡されます。

StrucId (MQCHAR4) - 入力

出口パラメーター構造 ID。値は次のとおりです。

```
MQAXP_STRUC_ID.
```

出口ハンドラーは、このフィールドを、各出口関数への入り口で設定します。

Version (MQLONG) - 入力

構造体のバージョン番号。値は次のとおりです。

MQAXP_VERSION_1

バージョン 1 API 出口パラメーター構造体。

MQAXP_VERSION_2

バージョン 2 API 出口パラメーター構造体。

MQAXP_CURRENT_VERSION

API 出口パラメーター構造体の現行バージョン番号。

出口ハンドラーは、このフィールドを、各出口関数への入り口で設定します。

ExitId (MQLONG) - 入力

出口ルーチンへの入り口で設定される出口 ID で、出口のタイプを示します。

MQXT_API_EXIT

API 出口。

ExitReason (MQLONG) - 入力

各出口関数への入り口で設定される、出口を呼び出す理由。

MQXR_CONNECTION

出口は、MQCONN または MQCONNX 呼び出しの前にそれ自体を初期化するため、または MQDISC 呼び出しの後にそれ自体を終了するために呼び出されています。

MQXR_BEFORE

出口は、API 呼び出しの実行前、または MQGET 上のデータを変換する前に呼び出されています。

MQXR_AFTER

出口は、API 呼び出し実行後に呼び出されています。

ExitResponse (MQLONG) - 出力

各出口関数への入り口で初期化される、出口からの応答。

MQXCC_OK

通常どおりに続行します。

このフィールドは、キュー・マネージャーに対して出口関数の実行結果を通信するため、出口関数によって設定する必要があります。値は次のいずれかでなければなりません。

MQXCC_OK

出口関数は正常に完了しました。通常どおりに続行します。

この値は、すべての MQXR_* 出口関数によって設定できます。ExitResponse2 は、チェーン中で後から出口関数を呼び出すかどうかを決定します。

MQXCC_FAILED

出口関数はエラーのために失敗しました。

この値は、すべての MQXR_* 出口関数によって設定できます。キュー・マネージャーは CompCode を MQCC_FAILED に、Reason を以下のように設定します。

- 関数が MQ_INIT_EXIT である場合 MQRC_API_EXIT_INIT_ERROR
- 関数が MQ_TERM_EXIT である場合 MQRC_API_EXIT_TERM_ERROR
- 他のすべての出口関数の場合 MQRC_API_EXIT_ERROR

設定される値は、後でチェーン中の出口関数を使って変更できます。

ExitResponse2 は無視されます。キュー・マネージャーは、MQXR2_SUPPRESS_CHAIN が戻された場合と同じように処理を続行します。

MQXCC_SUPPRESS_FUNCTION

IBM MQ API 関数を抑制します。

この値は MQXR_BEFORE 出口関数によってのみ設定可能です。この値は API 呼び出しを迂回します。この値が MQ_DATA_CONV_ON_GET_EXIT によって戻されると、データ変換も迂回されます。

キュー・マネージャーは CompCode を MQCC_FAILED に、Reason を MQRC_SUPPRESSED_BY_EXIT に設定しますが、設定される値は、後でチェーン中の出口関数を使って変更できます。呼び出し用の他のパラメーターは、出口が残したままとなります。ExitResponse2 は、チェーン中で後から出口関数を呼び出すかどうかを決定します。

この値が MQXR_AFTER または MQXR_CONNECTION 出口関数によって設定される場合、キュー・マネージャーは MQXCC_FAILED が戻された場合と同じように処理を続行します。

MQXCC_SKIP_FUNCTION

IBM MQ API 関数をスキップします。

この値は MQXR_BEFORE 出口関数によってのみ設定可能です。この値は API 呼び出しを迂回します。この値が MQ_DATA_CONV_ON_GET_EXIT によって戻されると、データ変換も迂回されます。キュー・マネージャーは CompCode および Reason をアプリケーションに戻される値に設定しますが、設定される値は、後でチェーン中の出口関数を使って変更できます。呼び出し用の他のパラメ

ーターは、出口が残したままとなります。ExitResponse2 は、チェーン中で後から出口関数を呼び出すかどうかを決定します。

この値が MQXR_AFTER または MQXR_CONNECTION 出口関数によって設定される場合、キュー・マネージャーは MQXCC_FAILED が戻された場合と同じように処理を続行します。

MQXCC_SUPPRESS_EXIT

出口のセットに所属するすべての出口関数を抑制します。

この値は MQXR_BEFORE および MQXR_AFTER 出口関数によってのみ設定可能です。この値は、この論理接続用の出口のセットに所属する出口関数の、すべての 後続呼び出しを迂回します。この迂回は、論理切断要求が出されるまで、つまり MQ_TERM_EXIT 関数が、MQXR_CONNECTION の ExitReason で呼び出されるまで続行します。

キュー・マネージャーは CompCode および Reason をアプリケーションに戻される値に設定しますが、設定される値は、後でチェーン中の出口関数を使って変更できます。呼び出し用の他のパラメーターは、出口が残したままとなります。ExitResponse2 は無視されます。

この値が MQXR_CONNECTION 出口関数によって設定される場合、キュー・マネージャーは MQXCC_FAILED が戻された場合と同じように処理を続行します。

ExitResponse および ExitResponse2 間の対話、および出口処理での影響に関する詳細は、[1570 ページ](#)の『キュー・マネージャーが出口関数を処理する方法』を参照してください。

ExitResponse2 (MQLONG) - 出力

これは、MQXR_BEFORE 出口関数の 1 次出口応答コードを修飾する 2 次出口応答コードです。次のように初期化されます。

```
MQXR2_DEFAULT_CONTINUATION
```

この初期化は、IBM MQ API 呼び出し出口関数への入り口で行われます。その後、次のいずれかの値に設定できます。

MQXR2_DEFAULT_CONTINUATION

ExitResponse の値にしたがって、チェーン中の次の出口に続行するかどうか。

ExitResponse が MQXCC_SUPPRESS_FUNCTION または MQXCC_SKIP_FUNCTION である場合、後で MQXR_BEFORE チェーン中の出口関数、および MQXR_AFTER チェーン中の一致する出口関数を迂回します。また、MQXR_BEFORE チェーン中の、前の出口関数と一致する MQXR_AFTER チェーン中の出口関数を呼び出します。

これ以外の場合は、チェーン中の次の出口を呼び出します。

MQXR2_SUPPRESS_CHAIN

チェーンを抑制します。

後で MQXR_BEFORE チェーン中の出口関数、およびこの API 呼び出しの MQXR_AFTER チェーン中で一致する出口関数を迂回します。また、MQXR_BEFORE チェーン中の、前の出口関数と一致する MQXR_AFTER チェーン中の出口関数を呼び出します。

MQXR2_CONTINUE_CHAIN

チェーン中の次の出口を続行します。

ExitResponse および ExitResponse2 間の対話、および出口処理での影響に関する詳細は、[1570 ページ](#)の『キュー・マネージャーが出口関数を処理する方法』を参照してください。

Feedback (MQLONG) - 入出力

出口関数の呼び出し間でフィードバックを通信します。これは次のように初期化されます。

```
MQFB_NONE (0)
```

初期化は、チェーン中の最初の出口の最初の関数の呼び出しの前に行われます。

出口は、有効な MQFB_* または MQRC 値を含め、このフィールドをどんな値にでも設定できます。また、このフィールドを、MQFB_APPL_FIRST から MQFB_APPL_LAST までの範囲の、ユーザー定義のフィードバック値に設定することもできます。

APICallerType (MQLONG) - 入力

API 呼び出し元タイプ。IBM MQ API 呼び出し元が、キュー・マネージャーに対して外部か内部か (MQXACT_EXTERNAL か MQXACT_INTERNAL) を示します。

ExitUserArea (MQBYTE16) - 入出力

特定の ExitInfoObject に関連するすべての出口で使用可能なユーザー域。hconn の最初の出口関数 (MQ_INIT_EXIT) を呼び出す前に、MQXUA_NONE (ExitUserArea の長さの場合、バイナリー・ゼロ) に初期化されます。その後は、出口関数によってこのフィールドに加えられる変更は、同じ出口の関数が呼び出されると保存されます。

このフィールドは、4 MQLONG の倍数に調整されます。

出口は、この領域から割り振るストレージを固定することもできます。

hconn ごとに、出口のチェーン中の出口に、それぞれ異なる ExitUserArea があります。ExitUserArea を、チェーン中の出口で共有することはできません。また、1 つの出口の ExitUserArea の内容を、チェーン中の別の出口が使用することもできません。

C プログラムの場合、定数 MQXUA_NONE_ARRAY も MQXUA_NONE と同じ値で定義されますが、ストリングではなく文字の配列として定義されます。

このフィールドの長さは MQ_EXIT_USER_AREA_LENGTH によって指定されます。

ExitData (MQCHAR32) - 入力

各出口関数への入力上で、出口で提供される 32 文字以内の出口固有のデータに設定される、出口データ。出口に値を定義しないと、このフィールドはブランクになります。

このフィールドの長さは MQ_EXIT_DATA_LENGTH によって指定されます。

ExitInfoName (MQCHAR48) - 入力

各出口関数への入力上で、スタンザの出口定義で指定される ApiExit_name に設定される、出口情報名。

ExitPDArea (MQBYTE48) - 入出力

出口関数の各呼び出しごとに、MQXPDA_NONE (フィールドの長さの場合バイナリー・ゼロ) に初期化される、問題判別エリア。

C プログラムの場合、定数 MQXPDA_NONE_ARRAY も MQXPDA_NONE と同じ値で定義されますが、ストリングではなく文字の配列として定義されます。

出口ハンドラーは、関数が正常に実行された場合も含め、常に出口の終了時に IBM MQ トレースにこのエリアを書き込みます。

このフィールドの長さは MQ_EXIT_PD_AREA_LENGTH により指定されます。

QMGrName (MQCHAR48) - 入力

IBM MQ API 呼び出しの処理の結果として出口を呼び出した、アプリケーションの接続先のキュー・マネージャーの名前。

MQCONN 呼び出しまたは MQCONNX 呼び出しで指定されたキュー・マネージャーの名前がブランクであっても、このフィールドは、アプリケーションの接続先のキュー・マネージャーの名前に設定されます。アプリケーションは、サーバーであってもクライアントであっても構いません。

出口ハンドラーは、このフィールドを、各出口関数への入り口で設定します。

このフィールドの長さは MQ_Q_MGR_NAME_LENGTH で指定します。

ExitChainAreaPtr (PMQACH) - 入出力

チェーン中のさまざまな出口の呼び出しを介してデータを通信するのに使用されます。これは、出口のチェーン中の最初の出口の最初の関数 (ExitReason MQXR_CONNECTION の MQ_INIT_EXIT) を呼び出す前に、NULL ポインターに設定されます。1 回の呼び出しで出口によって戻される値は、次の呼び出しに渡されます。

出口チェーン・エリアの詳細については、[1574 ページの『出口チェーン領域および出口チェーン領域ヘッダー \(MQACH\)』](#)を参照してください。

Hconfig (MQHCONFIG) - 入力

初期化中の関数のセットを表す、構成ハンドル。この値は、キュー・マネージャーによって MQ_INIT_EXIT 関数に生成され、後で API 出口関数に引き渡されます。この値は、各出口関数に入るたびに設定されます。

Hconfig は、MQI および DCI 呼び出しを行うための MQIEP 構造体へのポインターとして使用できます。HConfig パラメーターを MQIEP 構造体へのポインターとして使用する前に、HConfig の最初の 4 バイトが MQIEP 構造体の StrucId と一致していることを確認する必要があります。

Function (MQLONG) - 入力

1575 ページの『外部定数』で説明されている、有効な値の MQXF_* 定数である関数識別 ID。

出口ハンドラーは、呼び出されている出口の結果である IBM MQ API 呼び出しによって、各出口関数への入り口で、このフィールドを正しい値に設定します。

ExitMsgHandle (MQHMSG) - 入出力

Function が MQXF_GET で ExitReason が MQXR_AFTER の場合、有効なメッセージ処理がこのフィールドに戻され、API 出口はメッセージ記述子フィールド、および API 出口を登録したときに MQXEPO 構造で指定された ExitProperties スtring に一致する他のすべてのプロパティにアクセスできるようになります。

ExitMsgHandle に戻された非メッセージ記述子プロパティは、MQGMO 構造 (指定されている場合) またはメッセージ・データの MsgHandle から使用することはできません。

Function が MQXF_GET で、ExitReason が MQXR_BEFORE である場合、出口プログラムによってこのフィールドが MQHM_NONE に設定されると、ExitMsgHandle プロパティへのデータ挿入は抑止されます。

Version が MQAXP_VERSION_2 より小さい場合、このフィールドはセットされません。

キュー・マネージャーが出口関数を処理する方法

出口関数から戻るときにキュー・マネージャーによって実行される処理は、ExitResponse および ExitResponse2 の両方に依存しています。

1571 ページの表 871 に、可能な組み合わせと、MQXR_BEFORE 出口関数に対するその効果を要約し、次の点を説明します。

- 誰が API 呼び出しの CompCode および Reason パラメーターを設定するか
- MQXR_BEFORE チェーン内の残りの出口関数と、MQXR_AFTER チェーン内の一致する出口関数とが起動されるかどうか
- API 呼び出しが起動されるかどうか

MQXR_AFTER 出口関数では

- CompCode および Reason は、MQXR_BEFORE と同じ方法で設定されます
- ExitResponse2 は無視されます (MQXR_AFTER チェーン内の残りの出口関数は常に起動されます)
- MQXCC_SUPPRESS_FUNCTION および MQXCC_SKIP_FUNCTION は無効です

MQXR_CONNECTION 出口関数では

- CompCode および Reason は、MQXR_BEFORE と同じ方法で設定されます
- ExitResponse2 は無視されます
- MQXCC_SUPPRESS_FUNCTION、MQXCC_SKIP_FUNCTION、MQXCC_SUPPRESS_EXIT は無効です

出口またはキュー・マネージャーが CompCode および Reason を設定するすべての場合で、設定された値は後に起動される出口または API 呼び出し (API 呼び出しが後に起動される場合) によって 変更可能です。

表 871. MQXR_BEFORE 出口の処理

ExitResponse の値	CompCode と Reason の設定者	ExitResponse2 (デフォルト継続) チェーンの値	ExitResponse2 (デフォルト継続) API の値
MQXCC_OK	出口	Y	Y
MQXCC_SUPPRESS_EXIT	出口	Y	Y
MQXCC_SUPPRESS_FUNCTION	キュー・マネージャー	N	N
MQXCC_SKIP_FUNCTION	出口	N	N
MQXCC_FAILED	キュー・マネージャー	N	N

クライアントによる出口関数の処理方法

一般的に、クライアントではサーバー・アプリケーションと同じ方法で出口関数を処理し、この構造体の *QMgrName* 属性が、機能がサーバー上またはクライアント上のいずれにあるかにかかわらず適用されます。

ただし、クライアントには *mqs.ini* ファイルの概念がないため、*ApiExitCommon* スタンザおよび *APIExitTemplate* スタンザは適用されません。*ApiExitLocal* スタンザのみが適用されます。このスタンザは *mqclient.ini* ファイルで構成されます。

IBM MQ API 出口のコンテキスト構造体 (MQAXC)

外部制御ブロックである MQAXC 構造体は、API 出口への入力パラメーターとして使用されます。

MQAXC には、以下の C 宣言があります。

```
typedef struct tagMQAXC {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQLONG     Environment;      /* Environment */
    MQCHAR12   UserId;           /* UserId associated with appl */
    MQBYTE40   SecurityId;       /* Extension to UserId running appl */
    MQCHAR264  ConnectionName;   /* Connection name */
    MQLONG     LongMCAUserIdLength; /* long MCA user identifier length */
    MQLONG     LongRemoteUserIdLength; /* long remote user identifier length */
    MQPTR      LongMCAUserIdPtr;  /* long MCA user identifier address */
    MQPTR      LongRemoteUserIdPtr; /* long remote user identifier address */
    MQCHAR28   ApplName;         /* Application name */
    MQLONG     ApplType;         /* Application type */
    MQPID      ProcessId;        /* Process identifier */
    MQTID      ThreadId;         /* Thread identifier */

    /* Ver:1 */
    MQCHAR     ChannelName[20];  /* Channel Name */
    MQBYTE4    Reserved1;       /* Reserved */
    PMQCD      pChannelDefinition; /* Channel Definition pointer */
};
```

MQAXC へのパラメーターは、以下のとおりです。

StrucId (MQCHAR4) - 入力

値が MQAXC_STRUC_ID の出口構造体 ID。C プログラムの場合、定数 MQAXC_STRUC_ID_ARRAY も定義されます。この値は MQAXC_STRUC_ID と同じですが、これはストリングではなく文字の配列です。

出口ハンドラーは、このフィールドを、各出口関数への入り口で設定します。

Version (MQLONG) - 入力

構造体のバージョン番号。値は次のとおりです。

MQAXC_VERSION_2

出口コンテキスト構造体のバージョン番号。

MQAXC_CURRENT_VERSION

出口コンテキスト構造体の現行バージョン番号。

出口ハンドラーは、このフィールドを、各出口関数への入り口で設定します。

Environment (MQLONG) - 入力

出口関数を起動した IBM MQ API 呼び出しの発行元の環境。このフィールドに有効な値は、次のとおりです。

MQXE_OTHER

API 出口がサーバー・アプリケーションから呼び出される場合、この値は API 出口が参照する呼び出しと一貫性のある値になります。これは、API 出口が変更されずにクライアント上で実行され、それ以外のものは参照しないことを意味します。

出口がクライアント上で実行されているかどうかを判別する必要がある場合、出口は *ChannelName* フィールドおよび *ChannelDefinition* フィールドを参照して、これを判別できます。

MQXE_MCA

MSG チャンネル・エージェント

MQXE_MCA_SVRCONN

クライアントの代わりに活動しているメッセージ・チャンネル・エージェント。

MQXE_COMMAND_SERVER

コマンド・サーバー

MQXE_MQSC

runmqsc コマンド・インタープリター。

出口ハンドラーは、このフィールドを、各出口関数への入り口で設定します。

UserId (MQCHAR12) - 入力

アプリケーションに関連したユーザー ID。特にクライアント接続の場合、このフィールドにはチャンネル・コードが実行しているユーザー ID ではなく、受諾されたユーザーのユーザー ID が含まれます。クライアントからブランク・ユーザー ID が渡される場合、既に使用中のユーザー ID は変更されません。つまり、新規ユーザー ID は受諾されません。

出口ハンドラーは、このフィールドを、各出口関数への入り口で設定します。このフィールドの長さは MQ_USER_ID_LENGTH によって指定されます。

クライアントの場合、これはクライアントからサーバーに送信されるユーザー ID です。MCAUser 構成または CHLAUTH 構成によりユーザー ID が変更される可能性があるため、これはキュー・マネージャーでクライアントを実行する際に使用される有効なユーザー ID ではない場合があることに注意してください。

SecurityId (MQBYTE40) - 入力

アプリケーションを実行しているユーザー ID の拡張版。この長さは、MQ_SECURITY_ID_LENGTH によって設定されます。

クライアントの場合、これはクライアントからサーバーに送信されるユーザー ID です。MCAUser 構成または CHLAUTH 構成によりユーザー ID が変更される可能性があるため、これはキュー・マネージャーでクライアントを実行する際に使用される有効なユーザー ID ではない場合があることに注意してください。

ConnectionName (MQCHAR264) - 入力

クライアントのアドレスに設定された接続名フィールド。例えば TCP/IP では、これはクライアントの IP アドレスになります。

このフィールドの長さは MQ_CONN_NAME_LENGTH によって指定されます。

クライアントの場合、これはキュー・マネージャーのパートナー・アドレスです。

LongMCAUserIdLength (MQLONG) - 入力

長い MCA ユーザー ID の長さ。

MCA がキュー・マネージャーに接続するとき、このフィールドは長い MCA ユーザー ID の長さに (そのような ID がない場合はゼロに) 設定されます。

クライアントの場合、これはクライアントの長いユーザー ID です。

LongRemoteUserIdLength (MQLONG) - 入力

長いリモート・ユーザー ID の長さ。

MCA がキュー・マネージャーに接続するとき、このフィールドは長いリモート・ユーザー ID の長さに設定されます。その他の場合には、このフィールドはゼロに設定されます。

クライアントの場合は、このフィールドをゼロに設定します。

LongMCAUserIdPtr (MQPTR) - 入力

長い MCA ユーザー ID のアドレス。

MCA がキュー・マネージャーに接続するとき、このフィールドは長い MCA ユーザー ID のアドレスに (そのような ID がない場合はヌル・ポインターに) 設定されます。

クライアントの場合、これはクライアントの長いユーザー ID です。

LongRemoteUserIdPtr (MQPTR) - 入力

長いリモート・ユーザー ID のアドレス。

MCA がキュー・マネージャーに接続するとき、このフィールドは長いリモート・ユーザー ID のアドレスに (そのような ID がない場合はヌル・ポインターに) 設定されます。

クライアントの場合は、このフィールドをゼロに設定します。

ApplName (MQCHAR28) - 入力

IBM MQ API 呼び出しを発行したアプリケーションまたはコンポーネントの名前。

ApplName を生成するための規則は、MQPUT のデフォルト名を生成するための規則と同じです。

このフィールドの値は、プログラム名のオペレーティング・システムを照会して見つけることができます。この長さは、MQ_APPL_NAME_LENGTH によって設定されます。

ApplType (MQLONG) - 入力

IBM MQ API 呼び出しを発行したアプリケーションまたはコンポーネントのタイプ。

この値はアプリケーションがコンパイルされたプラットフォームの MQAT_DEFAULT、または定義済み MQAT_* 値の 1 つと同じになります。

出口ハンドラーは、このフィールドを、各出口関数への入り口で設定します。

ProcessId (MQPID) - 入力

オペレーティング・システムのプロセス ID。

該当する場合、出口ハンドラーは、このフィールドを、各出口関数への入り口で設定します。

ThreadId (MQTID) - 入力

MQ スレッド ID。これは、MQ トレースおよび FFST ダンプで使用する ID と同じものですが、オペレーティング・システムのスレッド ID とは異なる場合があります。

該当する場合、出口ハンドラーは、このフィールドを、各出口関数への入り口で設定します。

ChannelName (MQCHAR) - 入力

ブランクが埋め込まれたチャネルの名前 (該当し、既知の場合)。

該当しない場合、このフィールドはヌル文字に設定されます。

Reserved1 (MQBYTE4) - 入力

このフィールドは予約済みです。

ChanneDefinition (PMQCD) - 入力

使用中のチャネル定義へのポインター (該当し、既知の場合)。

該当しない場合、このフィールドはヌル文字に設定されます。

接続の処理が IBM MQ チャネルの代わりに行われており、そのチャネル定義が読み取り済みである場合にのみ、ポインターが入力されることに注意してください。

特に、チャネルで最初の MQCONN 呼び出しが行われるときには、サーバーにチャネル定義は指定されません。さらに、ポインターが入力されると、そのポインターによりポイントされる構造体 (およびす

すべてのサブ構造体)を読み取り専用として処理する必要があります。構造体の更新は予測不能な結果を発生し、サポートされていません。

クライアントの場合、クライアントに指定された値を含まないフィールドには、クライアント・アプリケーションに適した値が含まれます。

出口チェーン領域および出口チェーン領域ヘッダー (MQACH)

必要な場合、出口関数は出口チェーン領域用のストレージを獲得し、MQAXP 内の ExitChainAreaPtr をこのストレージを指すように設定することができます。

出口 (同一または別個の出口関数) は、複数の出口チェーン領域を取得して、それらを相互にリンクすることができます。このリストに対する出口チェーン領域の追加や削除は、それらが出口ハンドラーから呼び出されている間に行う必要があります。こうすることで、複数の異なるスレッドがリストに対して同時に領域を追加または削除することによって生じる、直列化の問題を回避することができます。

出口チェーン領域は、MQACH ヘッダー構造体により開始しなければなりません。その C 宣言は以下のとおりです。

```
typedef struct tagMQACH {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQLONG     StrucLength;       /* Length of the MQACH structure */
    MQLONG     ChainAreaLength;   /* Exit chain area length */
    MQCHAR48   ExitInfoName;     /* Exit information name */
    PMQACH     NextChainAreaPtr; /* Pointer to next exit chain area */
};
```

出口チェーン領域ヘッダー内のフィールドは、以下のとおりです。

StrucId (MQCHAR4) - 入力

出口チェーン領域の構造体 ID。初期値は MQACH_DEFAULT によって MQACH_STRUC_ID に定義されます。

C プログラムの場合、定数 MQACH_STRUC_ID_ARRAY も定義されます。この値は MQACH_STRUC_ID と同じですが、これはストリングではなく文字の配列です。

Version (MQLONG) - 入力

構造体のバージョン番号。値は次のとおりです。

MQACH_VERSION_1

出口パラメーター構造体のバージョン番号。

MQACH_CURRENT_VERSION

出口コンテキスト構造体の現行バージョン番号。

MQACH_DEFAULT によって定義されるこのフィールドの初期値は MQACH_CURRENT_VERSION です。

注：この構造体の新しいバージョンを導入する場合、既存のパーツのレイアウトは変更されません。出口関数は、このバージョン番号が、出口関数が必要のあるフィールドを含む最小のバージョン以上であることを検査しなければなりません。

StrucLength (MQLONG) - 入力

MQACH 構造体の長さ。出口はこのフィールドを使用して、出口データの開始を判別し、そのデータを出口によって作成された構造体の長さに設定することができます。

MQACH_DEFAULT によって定義されるこのフィールドの初期値は MQACH_CURRENT_LENGTH です。

ChainAreaLength (MQLONG) - 入力

現行の出口チェーン領域の長さに設定された、MQACH ヘッダーを含む出口チェーン領域の長さ。

MQACH_DEFAULT によって定義されるこのフィールドの初期値はゼロです。

ExitInfoName (MQCHAR48) - 入力

出口情報名。

出口が MQACH 構造体を作成するときに、独自の ExitInfoName を使用してこのフィールドを初期化することにより、後にこの出口の別のインスタンスまたは協働している出口が、この MQACH 構造体を見つめられるようにしなければなりません。

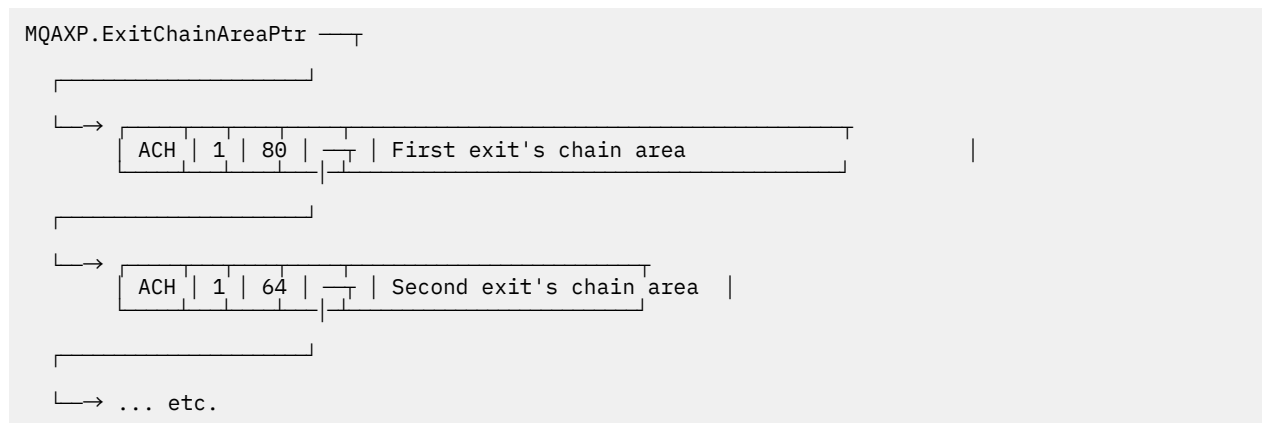
MQACH_DEFAULT によって定義されるこのフィールドの初期値は、長さがゼロの文字列 ({""}) です。

NextChainAreaPtr (PMQACH) - 入力

MQACH_DEFAULT によって定義され、初期値がヌル・ポインタ (NULL) の、次の出口チェーン領域へのポインタ。

出口関数は、取得したすべての出口チェーン領域のストレージを解放し、チェーン・ポインタを操作して出口チェーン領域をリストから除去しなければなりません。

出口チェーン領域は、次のように構成することができます。



外部定数

このトピックは、API 出口で利用可能な外部定数に関する参照情報として使用してください。

以下に示す外部定数を API 出口に使用できます。

MQXF_* (出口関数 ID)

MQXF_INIT	1	X'00000001'
MQXF_TERM	2	X'00000002'
MQXF_CONN	3	X'00000003'
MQXF_CONNX	4	X'00000004'
MQXF_DISC	5	X'00000005'
MQXF_OPEN	6	X'00000006'
MQXF_CLOSE	7	X'00000007'
MQXF_PUT1	8	X'00000008'
MQXF_PUT	9	X'00000009'
MQXF_GET	10	X'0000000A'
MQXF_DATA_CONV_ON_GET	11	X'0000000B'
MQXF_INQ	12	X'0000000C'
MQXF_SET	13	X'0000000D'
MQXF_BEGIN	14	X'0000000E'
MQXF_CMIT	15	X'0000000F'
MQXF_BACK	16	X'00000010'
MQXF_STAT	18	X'00000012'
MQXF_CB	19	X'00000013'
MQXF_CTL	20	X'00000014'
MQXF_CALLBACK	21	X'00000015'
MQXF_SUB	22	X'00000016'
MQXF_SUBRQ	23	X'00000017'
MQXF_XACLOSE	24	X'00000018'
MQXF_XACOMMIT	25	X'00000019'
MQXF_XACOMplete	26	X'0000001A'
MQXF_XAEND	27	X'0000001B'
MQXF_XAFORGET	28	X'0000001C'
MQXF_XAOPEN	29	X'0000001D'
MQXF_XAPREPARE	30	X'0000001E'
MQXF_XARECOVER	31	X'0000001F'
MQXF_XAROLLBACK	32	X'00000020'
MQXF_XASTART	33	X'00000021'

MQXF_AXREG	34	X'00000022'
MQXF_AXUNREG	35	X'00000023'

MQXR_* (出口の理由)

MQXR_BEFORE	1	X'00000001'
MQXR_AFTER	2	X'00000002'
MQXR_CONNECTION	3	X'00000003'

MQXE_* (環境)

MQXE_OTHER	0	X'00000000'
MQXE_MCA	1	X'00000001'
MQXE_MCA_SVRCONN	2	X'00000002'
MQXE_COMMAND_SERVER	3	X'00000003'
MQXE_MQSC	4	X'00000004'

MQ*_* (追加の定数)

MQAXP_VERSION_1	1	
MQAXP_VERSION_2	2	
MQAXC_VERSION_1	1	
MQACH_VERSION_1	1	
MQAXP_CURRENT_VERSION	1	
MQAXC_CURRENT_VERSION	1	
MQACH_CURRENT_VERSION	1	
MQXACT_EXTERNAL	1	
MQXACT_INTERNAL	2	
MQXT_API_EXIT	2	
MQACH_LENGTH_1	68 (32-bit platforms) 72 (64-bit platforms) 80 (128-bit platforms)	
MQACH_CURRENT_LENGTH	68 (32-bit platforms) 72 (64-bit platforms) 80 (128-bit platforms)	

MQ*_* (スル定数)

MQXPDA_NONE	X'00...00' (48 nulls)
MQXPDA_NONE_ARRAY	'\0','\0',...,'\0','\0'

MQXCC_* (完了コード)

MQXCC_FAILED	-8

MQRC_* (理由コード)

MQRC_API_EXIT_ERROR 2374 X'00000946'

出口関数の呼び出しが無効な応答コードを戻した、または何かの障害が生じたために、キュー・マネージャーは次に行うアクションを判別できません。

MQAXP の ExitResponse および ExitResponse2 フィールドの両方を調べて、無効な応答コードを判別し、有効な応答コードが戻されるように出口を変更してください。

MQRC_API_EXIT_INIT_ERROR 2375 X'00000947'

API 出口関数の実行環境を初期化する際に、キュー・マネージャーがエラーを検出しました。

MQRC_API_EXIT_TERM_ERROR 2376 X'00000948'

API 出口関数の実行環境をクローズする際に、キュー・マネージャーがエラーを検出しました。

MQRC_EXIT_REASON_ERROR 2377 X'00000949'

出口入り口点登録 (MQXEP) 呼び出しに指定された ExitReason フィールドの値にエラーが生じました。

ExitReason フィールドの値を検討して、無効な出口理由値を判別し、訂正してください。

MQRC_RESERVED_VALUE_ERROR 2378 X'0000094A'

Reserved フィールドの値にエラーが生じました。

Reserved フィールドの値を検討して、Reserved 値を判別し、訂正してください。

C 言語の typedef

このトピックでは、C 言語で使用できる API 出口関連の typedef について説明します。

API 出口に関連した C 言語の typedef を以下に示します。

```
typedef PMQLONG      MQPOINTER PPMQLONG;
typedef PMQBYTE      MQPOINTER PPMQBYTE;
typedef PMQHOBJS     MQPOINTER PPMQHOBJS;
typedef PMQOD        MQPOINTER PPMQOD;
typedef PMQMD        MQPOINTER PPMQMD;
typedef PMQPMO       MQPOINTER PPMQPMO;
typedef PMQGM        MQPOINTER PPMQGM;
typedef PMQCNO       MQPOINTER PPMQCNO;
typedef PMQBO        MQPOINTER PPMQBO;

typedef MQAXP        MQPOINTER PMQAXP;
typedef MQACH        MQPOINTER PMQACH;
typedef MQAXC        MQPOINTER PMQAXC;

typedef MQCHAR       MQCHAR16[16];
typedef MQCHAR16     MQPOINTER PMQCHAR16;

typedef MQLONG       MQPID;
typedef MQLONG       MQTID;
```

出口エントリー・ポイント登録呼び出し (MQXEP)

この情報は、MQXEP について、MQXEP C 言語呼び出しと、MQXEP C 関数プロトタイプについて学習するために使用してください。

MQXEP 呼び出しを使用して以下を行います。

1. 出口関数を呼び出す IBM MQ API 出口呼び出し点の前と後を登録します。
2. 出口関数の入り口点を指定します。
3. 出口関数の入り口点を登録解除します。

MQXEP 呼び出しは、通常、MQ_INIT_EXIT 出口関数の中でコーディングしますが、それ以降のどの出口関数の中でも指定できます。

MQXEP 呼び出しを使用して既に登録済みの出口関数を登録する場合、その 2 回目の MQXEP 呼び出しは正常に終了して、登録済みの出口関数は置換されます。

MQXEP 呼び出しを使用して NULL 出口関数を登録する場合、その MQXEP 呼び出しは正常に終了して、出口関数の登録が解除されます。

接続要求のライフ期間中に MQXEP 呼び出しを使用して特定の出口関数を登録、登録解除、および再登録する場合、以前に登録した出口関数が再活動化されます。出口の関数は、この出口関数のインスタンスにまだ割り振られて関連付けられている任意のストレージを使用できます。(このストレージは、通常、終了出口関数の呼び出しの間に解放されます。)

MQXEP へのインターフェースは、以下のとおりです。

```
MQXEP (Hconfig, ExitReason, Function, EntryPoint, &ExitOpts, &CompCode, &Reason)
```

ここで、

Hconfig (MQHCONFIG) - 入力

初期化中の機能のセットを含む API 出口を表す、構成ハンドル。この値は、MQ_INIT_EXIT 関数を起動する直前にキュー・マネージャーによって生成され、MQAXP に入れて各 API 出口関数に渡されます。

ExitReason (MQLONG) - 入力

入り口点が登録される理由。次の理由のいずれかです。

- 接続レベルの初期化または終了 (MQXR_CONNECTION)
- IBM MQ API 呼び出しの前 (MQXR_BEFORE)
- IBM MQ API 呼び出しの後 (MQXR_AFTER)

Function (MQLONG) - 入力

関数 ID。有効な値は、MQXF_* 定数です ([1575 ページの『外部定数』](#)を参照してください)。

EntryPoint (PMQFUNC) - 入力

登録する出口機能の入り口点のアドレス。値 NULL は、出口関数が用意されていないこと、または出口関数の以前の登録が登録解除されていることを示します。

ExitOpts(MQXEPO)

API 出口では、API 出口の登録方法を制御するオプションを指定できます。このフィールドに NULL ポインターを指定すると、MQXEPO 構造体のデフォルト値が設定されていると想定されます。

CompCode (MQLONG) - 出力

完了コード、有効な値は以下のとおりです。

MQCC_OK

正常終了。

MQCC_FAILED

呼び出し失敗。

Reason (MQLONG) - 出力

完了コードを修飾する理由コード。

完了コードが MQCC_OK の場合、以下のようになります。

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

完了コードが MQCC_FAILED の場合、以下のようになります。

MQRC_HCONFIG_ERROR

(2280, X'8E8') 指定された構成ハンドルが無効です。MQAXP にある構成ハンドルを使用してください。

MQRC_EXIT_REASON_ERROR

(2377, X'949') 指定された出口機能の起動理由が無効であるか、または指定された出口機能 ID に関して無効です。

有効な出口機能の起動理由 (MQXR_* 値) の 1 つを使用するか、または有効な機能 ID と出口理由との組み合わせを使用します。([1578 ページの表 872](#)を参照してください。)

MQRC_FUNCTION_ERROR

(2281, X'8E9') 指定された機能 ID が API 出口理由に関して無効です。以下の表は、機能 ID と ExitReasons との有効な組み合わせを示しています。

表 872. 機能 ID と ExitReasons との有効な組み合わせ	
Function	ExitReason
MQXF_INIT MQXF_TERM	MQXR_CONNECTION

表 872. 機能 ID と ExitReasons との有効な組み合わせ (続き)

Function	ExitReason
MQXF_CONN MQXF_CONNX MQXF_DISC MQXF_OPEN MQXF_CLOSE MQXF_PUT1 MQXF_PUT MQXF_GET MQXF_INQ MQXF_SET MQXF_BEGIN MQXF_CMIT MQXF_BACK MQXF_STAT MQXF_CB MQXF_CTL MQXF_CALLBACK MQXF_SUB MQXF_SUBRQ	MQXR_BEFORE MQXR_AFTER
MQXF_DATA_CONV_ON_GET	MQXR_BEFORE

MQRC_RESOURCE_PROBLEM

(2102, X'836') 出口関数を登録または登録解除しようとして、リソースの問題のために失敗しました。

MQRC_UNEXPECTED_ERROR

(2195, X'893') 出口関数を登録または登録解除しようとして、予期しない理由で失敗しました。

MQRC_PROPERTY_NAME_ERROR

(2442, X'098A') 無効な ExitProperties 名です。

MQRC_XEPO_ERROR

(2507, X'09CB') 終了オプションの構造体が無効です。

MQXEP C 言語呼び出し

```
MQXEP (Hconfig, ExitReason, Function, EntryPoint, &ExitOpts, &CompCode, &Reason);
```

パラメーター・リストの宣言:

```

MQHCONFIG    Hconfig;        /* Configuration handle */
MQLONG       ExitReason;     /* Exit reason */
MQLONG       Function;       /* Function identifier */
PMQFUNC      EntryPoint;     /* Function entry point */
MQXEPO       ExitOpts;      /* Options that control the action of MQXEP */
MQLONG       CompCode;       /* Completion code */
MQLONG       Reason;         /* Reason code qualifying completion
                             code */

```

MQXEP C 関数プロトタイプ

```

void MQXEP (
MQHCONFIG    Hconfig,        /* Configuration handle */
MQLONG       ExitReason,     /* Exit reason */
MQLONG       Function,       /* Function identifier */
PMQFUNC      EntryPoint,     /* Function entry point */
PMQXEPO      pExitOpts;      /* Options that control the action of MQXEP */
PMQLONG      pCompCode,      /* Address of completion code */

```

```
PMQLONG      pReason);          /* Address of reason code qualifying completion
                                code */
```

出口関数

このセクションでは、機能の呼び出しを使用するときに役立つ一般情報、および個別の出口関数を呼び出す方法について説明します。

この情報は、API 出口ルーチンに関する一般規則を理解し、出口の実行環境をセットアップおよびクリーンアップするために使用します。

API 出口ルーチンについての汎用規則

以下の一般規則が API 出口ルーチンを起動する際に適用されます。

- いずれの場合も、API 出口関数は、API 呼び出しパラメーターの妥当性検査の前、およびすべてのセキュリティ検査の前 (MQCONN、MQCONNEX、または MQOPEN の場合) に実行されます。
- フィールドに入力される値と、出口ルーチンから出力される値は、次のとおりです。
 - 前 IBM MQ API 出口機能への入力では、フィールドの値は、アプリケーション・プログラムまたは前の出口機能呼び出しによって設定できます。
 - 前 IBM MQ API 出口機能からの出力では、フィールドの値を未変更のままにすることも、出口機能によって他の値に設定することもできます。
 - 後 IBM MQ API 出口機能への入力では、フィールドの値は、IBM MQ API 呼び出しの処理後にキュー・マネージャーによって設定された値にすることも、出口機能のチェーン内の前の出口機能呼び出しによって設定された値にすることもできます。
 - 後 IBM MQ API 呼び出し出口機能からの出力では、フィールドの値は未変更のままにすることも、出口機能によって他の値に設定することもできます。
- 出口関数は ExitResponse および ExitResponse2 フィールドを使用してキュー・マネージャーと通信する必要があります。
- CompCode および Reason コード・フィールドは、アプリケーションに返されます。キュー・マネージャーおよび出口関数は、CompCode および Reason コード・フィールドを設定できます。
- MQXEP 呼び出しは、MQXEP を呼び出す出口関数に新しい理由コードを戻します。ただし、出口関数は、これらの新しい理由コードを、既存のアプリケーションおよび新規アプリケーションが理解できる既存の理由コードに変換することができます。
- 各出口関数プロトタイプには、追加のレベルの間接参照を備えた、API 機能に対する同様のパラメーター (CompCode と Reason を除く) があります。
- API 出口は MQI 呼び出し (MQDISC は除く) を発行できますが、これらの MQI 呼び出しは、それ自体では API 出口を呼び出しません。

アプリケーションがサーバー上またはクライアント上のいずれにあるかに関係なく、API 出口呼び出しの順序を予測することはできないことに注意してください。API 出口 BEFORE 呼び出しを AFTER 呼び出しの直前に置くことはできません。

BEFORE 呼び出しを別の BEFORE 呼び出しの前に置くことはできます。以下に例を示します。

```
BEFORE MQCTL
BEFORE Callback
BEFORE MQPUT
AFTER MQPUT
AFTER Callback
AFTER MQCTL
```

または

```
BEFORE XAOPEN
BEFORE MQCONNEX
AFTER MQCONNEX
```


AFTER XAOPEN

クライアントでは、PreConnect 出口という、MQCONN 呼び出しまたは MQCONNX 呼び出しの動作を変更できる出口があります。PreConnect 出口は、MQCONN 呼び出しまたは MQCONNX 呼び出しのすべてのパラメーター (キュー・マネージャー名を含む) を変更できます。クライアントはこの出口を最初に呼び出し、続けて MQCONN 呼び出しまたは MQCONNX 呼び出しを起動します。最初の MQCONN 呼び出しまたは MQCONNX 呼び出しのみが API 出口を呼び出すことに注意してください。それ以降の再接続呼び出しはいずれも効果がありません。

実行環境

一般に、出口関数からのすべてのエラーは、MQAXP 内の ExitResponse および ExitResponse2 フィールドを使用して出口ハンドラーに返信されます。

その後、これらのエラーは MQCC_* および MQRC_* 値に変換されて、CompCode フィールドと Reason フィールドに示される形でアプリケーションに返信されます。ただし、出口ハンドラー・ロジック内で検出されたすべてのエラーは、CompCode フィールドと Reason フィールドに MQCC_* 値および MQRC_* 値として示されてアプリケーションに返信されます。

MQ_TERM_EXIT 機能がエラーを戻す場合は、次のようになります。

- MQDISC 呼び出しは既に実行されています
- 後の MQ_TERM_EXIT 出口関数を実行する (それにより出口実行環境のクリーンアップを実行する) 機会
は、この他にありません。
- 出口実行環境のクリーンアップは行われていません。

出口は、まだ使用中である可能性があるため、アンロードできません。また、前 出口が正常に実行された出口チェーンをさかのぼって、他の登録済み出口を逆の順序で実行できます。

出口実行環境のセットアップ

明示的な MQCONN または MQCONNX 呼び出しを処理するとき、出口ハンドリング論理は出口初期化機能 (MQ_INIT_EXIT) を起動する前に出口実行環境をセットアップします。出口実行環境のセットアップには、出口のロード、ストレージの取得、および出口パラメーター構造体の初期化が含まれます。出口構成ハンドルも割り振られます。

この段階でエラーが生じる場合、MQCONN または MQCONNX 呼び出しは CompCode MQCC_FAILED および以下の理由コードの 1 つを戻して失敗します。

MQRC_API_EXIT_LOAD_ERROR

API 出口モジュールをロードしようとして失敗しました。

MQRC_API_EXIT_NOT_FOUND

API 出口関数が API 出口モジュール内で見つかりませんでした。

MQRC_STORAGE_NOT_AVAILABLE

API 出口関数の実行環境を初期化しようとして、使用可能なストレージが不足しているために失敗しました。

MQRC_API_EXIT_INIT_ERROR

API 出口関数の実行環境を初期化する際にエラーが検出されました。

出口実行環境のクリーンアップ

明示的な MQDISC 呼び出しか、アプリケーション終了の結果としての暗黙の切断要求を処理している間に、出口ハンドリング論理は、出口終了機能 (MQ_TERM_EXIT) が登録されていればそれを起動した後で、出口実行環境をクリーンアップしなければならない場合があります。

出口実行環境のクリーンアップには、それまでにメモリーにロードされているモジュールを削除するなどして、出口パラメーター構造体のストレージを解放することが含まれます。

この段階でエラーが発生する場合、明示的な MQDISC 呼び出しは CompCode MQCC_FAILED および以下の理由コードの 1 つを戻して失敗します (暗黙の切断要求ではエラーは指摘されません)。

MQRC_API_EXIT_TERM_ERROR

API 出口関数の実行環境をクローズする際にエラーが検出されました。出口は、MQ_TERM* API 出口関数呼び出しの前後に、MQDISC から失敗を戻さないようにします。

クライアント上の API 出口

クライアントは PreConnect 出口を使用して MQCONN 呼び出しおよび MQCONNX 呼び出しの動作を変更し、API 出口プロパティをサポートしません。

PreConnect 出口

クライアントでは、PreConnect 出口を使用して、LDAP サーバーなどの中央リポジトリからチャネル定義を検索できます。

PreConnect 出口は、MQCONN 呼び出しまたは MQCONNX 呼び出し自体で、キュー・マネージャー名などの任意のパラメーター、またはすべてのパラメーターを変更することもできます。

クライアント・アプリケーションの場合、PreConnect 出口は API 出口より前に呼び出す必要があります。これは、MQCONN API 出口または MQCONNX API 出口はキュー・マネージャーの名前が既知である場合にのみ呼び出され、この名前は PreConnect 出口により変更される可能性があるためです。

最初の MQCONN 呼び出しまたは MQCONNX 呼び出しのみが出口を呼び出すことに注意してください。

API 出口プロパティ

サーバーでは、API 出口は初期設定時に MQXEPO 構造体を登録できます。MQXEPO 構造体には、出口に係のあるプロパティのグループについての詳細が記された ExitProperties フィールドが含まれます。このフィールドにより、出口がどのアプリケーション・メッセージ・プロパティ・ハンドルとも区別して操作できる、個別のメッセージ・プロパティ・ハンドルが生成されます。

クライアントでは、API 出口プロパティはサポートされません。クライアント上でプロパティ・グループ名の登録が試行されると、関数は理由コード MQRC_EXIT_PROPS_NOT_SUPPORTED により失敗します。

バックアウト - MQ_BACK_EXIT

MQ_BACK_EXIT は、バックアウト処理の前 および後に実行するバックアウト出口関数を提供します。関数 ID MQXF_BACK に出口理由 MQXR_BEFORE および MQXR_AFTER を指定して、バックアウト呼び出し前およびバックアウト呼び出し後 出口関数を登録します。

この関数へのインターフェースは、以下のとおりです。

```
MQ_BACK_EXIT (&ExitParms, &ExitContext, &Hconn, &CompCode, &Reason)
```

パラメーターは、以下のとおりです。

ExitParms (MQAXP) - 入出力

出口パラメーター構造体。

ExitContext (MQAXC) - 入出力

出口コンテキスト構造体。

Hconn (MQHCONN) - 入力

接続ハンドル。

CompCode (MQLONG) - 入出力

完了コード、有効な値は以下のとおりです。

MQCC_OK

正常終了。

MQCC_WARNING

一部完了。

MQCC_FAILED

呼び出し失敗

Reason (MQLONG) - 入出力

完了コードを修飾する理由コード。

完了コードが MQCC_OK の場合、以下の値だけが有効です。

MQRC_NONE

(0, x'000') レポートする理由コードはありません。

完了コードが MQCC_FAILED または MQCC_WARNING の場合、出口関数は理由コード・フィールドを任意の有効な MQRC_* 値に設定できます。

C 言語呼び出し

キュー・マネージャーは、以下の変数を論理的に定義します。

```
MQAXP    ExitParms;      /* Exit parameter structure */
MQAXC    ExitContext;    /* Exit context structure */
MQHCONN   Hconn;         /* Connection handle */
MQLONG    CompCode;      /* Completion code */
MQLONG    Reason;        /* Reason code qualifying completion code */
```

その後、キュー・マネージャーは以下のように出口を論理的に呼び出します。

```
MQ_BACK_EXIT (&ExitParms, &ExitContext, &Hconn, &CompCode, &Reason);
```

出口は以下の C 関数プロトタイプと一致していなければなりません。

```
void MQENTRY MQ_BACK_EXIT (
PMQAXP    pExitParms,      /* Address of exit parameter structure */
PMQAXC    pExitContext,    /* Address of exit context structure */
PMQHCONN   pHconn,        /* Address of connection handle */
PMQLONG    pCompCode,      /* Address of completion code */
PMQLONG    pReason);      /* Address of reason code qualifying completion
                           code */
```

開始 - MQ_BEGIN_EXIT

MQ_BEGIN_EXIT は、MQBEGIN 呼び出し処理の前 および後 に実行する開始出口関数を提供します。関数 ID MQXF_BEGIN と終了理由 MQXR_BEFORE および MQXR_AFTER を使用して、MQBEGIN 呼び出し出口関数の前と後に登録します。

この関数へのインターフェースは、以下のとおりです。

```
MQ_BEGIN_EXIT (&ExitParms, &ExitContext, &Hconn, &pBeginOptions, &CompCode,
               &Reason)
```

パラメーターは、以下のとおりです。

ExitParms (MQAXP) - 入出力

出口パラメーター構造体。

ExitContext (MQAXC) - 入出力

出口コンテキスト構造体。

Hconn (MQHCONN) - 入力

接続ハンドル。

pBeginOptions (PMQBO) - 入出力

開始オプションのポインター。

CompCode (MQLONG) - 入出力

完了コード、有効な値は以下のとおりです。

MQCC_OK

正常終了。

MQCC_WARNING

一部完了。

MQCC_FAILED

呼び出し失敗

Reason (MQLONG) - 入出力

完了コードを修飾する理由コード。

完了コードが MQCC_OK の場合、以下の値だけが有効です。

MQRC_NONE

(0, x'000') レポートする理由コードはありません。

完了コードが MQCC_FAILED または MQCC_WARNING の場合、出口関数は理由コード・フィールドを任意の有効な MQRC_* 値に設定できます。

C 言語呼び出し

キュー・マネージャーは、以下の変数を論理的に定義します。

```
MQAXP    ExitParms;        /* Exit parameter structure */
MQAXC    ExitContext;      /* Exit context structure */
MQHCONN  Hconn;            /* Connection handle */
PMQBO    pBeginOptions;    /* Ptr to begin options */
MQLONG   CompCode;         /* Completion code */
MQLONG   Reason;           /* Reason code qualifying completion code */
```

その後、キュー・マネージャーは以下のように出口を論理的に呼び出します。

```
MQ_BEGIN_EXIT (&ExitParms, &ExitContext, &Hconn, &pBeginOptions, &CompCode,
               &Reason);
```

出口は以下の C 関数プロトタイプと一致していなければなりません。

```
void MQENTRY MQ_BEGIN_EXIT (
PMQAXP    pExitParms,      /* Address of exit parameter structure */
PMQAXC    pExitContext,    /* Address of exit context structure */
PMQHCONN  pHconn,          /* Address of connection handle */
PPMQBO    ppBeginOptions,  /* Address of ptr to begin options */
PMQLONG   pCompCode,       /* Address of completion code */
PMQLONG   pReason);        /* Address of reason code qualifying completion
                             code */
```

コールバック - MQ_CALLBACK_EXIT

MQ_CALLBACK_EXIT は、コールバック処理の前 および後 に実行する出口関数を提供します。関数 ID MQXF_CALLBACK に出口理由 MQXR_BEFORE および MQXR_AFTER を指定して、コールバック呼び出し前 およびコールバック呼び出し後 出口関数を登録します。

この関数へのインターフェースは、以下のとおりです。

```
MQ_CALLBACK_EXIT (&ExitParms, &ExitContext, &Hconn, &pMsgDesc, &pGetMsgOpts,
                  &pBuffer, &pMQCBCContext)
```

パラメーターは、以下のとおりです。

ExitParms (MQAXP) - 入出力

出口パラメーター構造体

ExitContext (MQAXC) - 入出力

出口コンテキスト構造体

Hconn (MQHCONN) - 入出力

接続ハンドル

pMsgDesc

メッセージ記述子

pGetMsgOpts

MQGET のアクションを制御するオプション

pBuffer

メッセージ・データが入れられる区域

pMQCBCContext

コールバックのコンテキスト・データ

C 言語呼び出し

キュー・マネージャーは、以下の変数を論理的に定義します。

```
MQAXP      ExitParms;      /* Exit parameter structure */
MQAXC      ExitContext;    /* Exit context structure */
MQHCONN    Hconn;         /* Connection handle */
PMQMD      pMsgDesc;       /* Message descriptor */
PMQGMO     pGetMsgOpts;    /* Options that define the operation of the consumer */
PMQVOID    pBuffer;        /* Area to contain the message data */
PMQCBC     pContext;       /* Context data for the callback */
```

その後、キュー・マネージャーは以下のように出口を論理的に呼び出します。

```
MQ_SUBRQ_EXIT (&ExitParms, &ExitContext, &Hconn, &pMsgDesc, &pGetMsgOpts, &pBuffer,
               &pContext);
```

出口は以下の C 関数プロトタイプと一致していなければなりません。

```
void MQENTRY MQ_CALLBACK_EXIT (
    PMQAXP      pExitParms;    /* Exit parameter structure */
    PMQAXC      pExitContext;  /* Exit context structure */
    PMQHCONN    pHconn;        /* Connection handle */
    PPMQMD      ppMsgDesc;     /* Message descriptor */
    PPMQGMO     ppGetMsgOpts;  /* Options that define the operation of the consumer */
    PPMQVOID    ppBuffer;      /* Area to contain the message data */
    PPMQCBC     ppContext;)    /* Context data for the callback */
```

使用上の注意

1. コールバック出口は、コンシューマーが呼び出される前、およびコンシューマーのコンシューマー関数が完了した後に呼び出されます。MQMD 構造体と MQGMO 構造体は変更可能ですが、前出口で値を変更しても、キューからのメッセージの取り出しは再駆動されません。これは、コンシューマー関数に送信されるはずのキューからメッセージが既に除去されているためです。

コールバック管理関数 - MQ_CB_EXIT

MQ_CB_EXIT は、MQCB 呼び出しの前 および後 に実行する出口関数を提供します。関数 ID MQXF_CB に出口理由 MQXR_BEFORE および MQXR_AFTER を指定して、MQCB 呼び出し出口関数の前 と後 を登録します。

この関数へのインターフェースは、以下のとおりです。

```
MQ_CB_EXIT (&ExitParms, &ExitContext, &Hconn, &Operation, &pCallbackDesc,
            &Hobj, &pMsgDesc, &pGetMsgOpts, &CompCode, &Reason)
```

パラメーターは、以下のとおりです。

ExitParms (MQAXP) - 入出力

出口パラメーター構造体

ExitContext (MQAXC) - 入出力

出口コンテキスト構造体

Hconn (MQHCONN) - 入出力

接続ハンドル

Operation (MQLONG) - 入出力

操作の値

pCallbackDesc (PMQCBD) - 入出力

コールバック記述子

Hobj (MQHOBJ) - 入出力

オブジェクト・ハンドル

pMsgDesc (PMQMD) - 入出力

メッセージ記述子

pGetMsgOpts (PMQGMO) - 入出力

MQCB のアクションを制御するオプション

CompCode (MQLONG) - 入出力

完了コード

Reason (MQLONG) - 入出力

CompCode を限定する理由コード

C 言語呼び出し

キュー・マネージャーは、以下の変数を論理的に定義します。

```
MQAXP    ExitParms;      /* Exit parameter structure */
MQAXC    ExitContext;    /* Exit context structure */
MQHCONN   Hconn;         /* Connection handle */
MQLONG    Operation;     /* Operation value. */
MQCBD     pMsgDesc;      /* Callback descriptor. */
MQHOBJ    Hobj;          /* Object handle. */
PMQMD     pMsgDesc;      /* Message descriptor */
PMQGMO    pGetMsgOpts;   /* Options that define the operation of the consumer */
PMQLONG   CompCode;      /* Completion code. */
PMQLONG   Reason;        /* Reason code qualifying CompCode. */
```

その後、キュー・マネージャーは以下のように出口を論理的に呼び出します。

```
MQ_CB_EXIT (&ExitParms, &ExitContext, &Hconn, &Operation, &Hobj, &pMsgDesc,
            &pGetMsgOpts, &CompCode, &Reason);
```

出口は以下の C 関数プロトタイプと一致していなければなりません。

```
void MQENTRY MQ_CB_EXIT (
    PMQAXP    pExitParms;      /* Exit parameter structure */
    PMQAXC    pExitContext;    /* Exit context structure */
    PMQHCONN   pHconn;         /* Connection handle */
    PMQLONG    pOperation;     /* Callback operation */
    PMQHOBJS   pHobj;          /* Object handle */
    PPMQMD     ppMsgDesc;      /* Message descriptor */
    PPMQGMO    ppGetMsgOpts;   /* Options that control the action of MQCB */
    PMQLONG    pCompCode;      /* Completion code */
    PMQLONG    pReason;        /* Reason code qualifying CompCode */
```

クローズ - MQ_CLOSE_EXIT

MQ_CLOSE_EXIT は、MQCLOSE 呼び出し処理の前 および後 に実行するクローズ出口関数を提供します。MQCLOSE 呼び出し終了関数の前と後に登録するには、終了理由 MQXR_BEFORE、MQXR_AFTER とともに関数識別子 MQXF_CLOSE を使用します。

この関数へのインターフェースは、以下のとおりです。

```
MQ_CLOSE_EXIT (&ExitParms, &ExitContext, &Hconn, &pHobj,
               &Options, &CompCode, &Reason)
```

パラメーターは、以下のとおりです。

ExitParms (MQAXP) - 入出力

出口パラメーター構造体。

ExitContext (MQAXC) - 入出力

出口コンテキスト構造体。

Hconn (MQHCONN) - 入力

接続ハンドル。

pHobj (PMQHOBJS) - 入力

オブジェクト・ハンドルへのポインター。

Options (MQLONG)- 入出力

クローズ・オプション。

CompCode (MQLONG) - 入出力

完了コード、有効な値は以下のとおりです。

MQCC_OK

正常終了。

MQCC_FAILED

呼び出し失敗

Reason (MQLONG) - 入出力

完了コードを修飾する理由コード。

完了コードが MQCC_OK の場合、以下の値だけが有効です。

MQRC_NONE

(0, x'000') レポートする理由コードはありません。

完了コードが MQCC_FAILED の場合、出口関数は理由コード・フィールドを任意の有効な MQRC_* 値に設定できます。

C 言語呼び出し

キュー・マネージャーは、以下の変数を論理的に定義します。

```
MQAXP      ExitParms;      /* Exit parameter structure */
MQAXC      ExitContext;    /* Exit context structure */
MQHCONN     Hconn;         /* Connection handle */
PMQHOBj     pHobj;         /* Ptr to object handle */
MQLONG      Options;       /* Close options */
MQLONG      CompCode;      /* Completion code */
MQLONG      Reason;        /* Reason code */
```

その後、キュー・マネージャーは以下のように出口を論理的に呼び出します。

```
MQ_CLOSE_EXIT (&ExitParms, &ExitContext, &Hconn, &pHobj, &Options,
               &CompCode, &Reason);
```

出口は以下の C 関数プロトタイプと一致していなければなりません。

```
void MQENTRY MQ_CLOSE_EXIT (
PMQAXP      pExitParms,    /* Address of exit parameter structure */
PMQAXC      pExitContext,  /* Address of exit context structure */
PMQHCONN     pHconn,       /* Address of connection handle */
PPMHOBj     ppHobj,        /* Address of ptr to object handle */
PMQLONG      pOptions,     /* Address of close options */
PMQLONG      pCompCode,    /* Address of completion code */
PMQLONG      pReason);     /* Address of reason code qualifying
                             completion code */
```

コミット - MQ_CMtT_EXIT

MQ_CMtT_EXIT は、コミット処理の前と後に実行するコミット出口関数を提供します。MQXR_BEFORE および MQXR_AFTER を使用して関数 ID MQXF_CMtT を使用して、前および後のコミット呼び出し出口関数を登録します。

コミット操作が失敗してトランザクションがバックアウトした場合、MQCMIT 呼び出しは MQCC_WARNING および MQRC_BACKED_OUT を出して失敗します。これらの戻りコードおよび理由コードは後 MQCMIT 出口関数に渡されて、作業単位がバックアウトされたことを出口関数に示します。

この関数へのインターフェースは、以下のとおりです。

```
MQ_CMtT_EXIT (&ExitParms, &ExitContext, &Hconn, &CompCode, &Reason)
```

パラメーターは、以下のとおりです。

ExitParms (MQAXP) - 入出力

出口パラメーター構造体。

ExitContext (MQAXC) - 入出力

出口コンテキスト構造体。

Hconn (MQHCONN) - 入力

接続ハンドル。

CompCode (MQLONG) - 入出力

完了コード、有効な値は以下のとおりです。

MQCC_OK

正常終了。

MQCC_WARNING

一部完了。

MQCC_FAILED

呼び出し失敗

Reason (MQLONG) - 入出力

完了コードを修飾する理由コード。

完了コードが MQCC_OK の場合、以下の値だけが有効です。

MQRC_NONE

(0, x'000') レポートする理由コードはありません。

完了コードが MQCC_FAILED または MQCC_WARNING の場合、出口関数は理由コード・フィールドを任意の有効な MQRC_* 値に設定できます。

C 言語呼び出し

キュー・マネージャーは、以下の変数を論理的に定義します。

```
MQAXP    ExitParms;      /* Exit parameter structure */
MQAXC    ExitContext;    /* Exit context structure */
MQHCONN  Hconn;          /* Connection handle */
MQLONG   CompCode;       /* Completion code */
MQLONG   Reason;         /* Reason code qualifying completion code */
```

その後、キュー・マネージャーは以下のように出口を論理的に呼び出します。

```
MQ_CMIT_EXIT (&ExitParms, &ExitContext, &Hconn, &CompCode, &Reason);
```

出口は以下の C 関数プロトタイプと一致していなければなりません。

```
void MQENTRY MQ_CMIT_EXIT (
PMQAXP    pExitParms,      /* Address of exit parameter structure */
PMQAXC    pExitContext,    /* Address of exit context structure */
PMQHCONN  pHconn,          /* Address of connection handle */
PMQLONG   pCompCode,       /* Address of completion code */
PMQLONG   pReason);        /* Address of reason code qualifying completion
                             code */
```

使用上の注意

1. ここで説明する MQ_GET_EXIT 関数インターフェースは、MQXF_GET 出口関数と [1594 ページの『MQXF_DATA_CONV_ON_GET』](#) 出口関数の両方に使用されます。

この 2 つの出口関数には別々のエン트리・ポイントが定義されているため、両方ともインターセプトするためには、MQXEP 呼び出しを 2 回使用しなければなりません。この呼び出しには、関数 ID MQXF_GET を使用します。

MQ_GET_EXIT インターフェースは、MQXF_GET でも MQXF_DATA_CONV_ON_GET でも同じであるため、1 つの出口関数を両方に使用することができます。どちらの出口関数が呼び出されているかは、MQAXP 構造体の *Function* フィールドに示されます。あるいは、MQXEP 呼び出しを使用して、2 つのケースに別々の出口関数を登録することもできます。

接続および接続拡張 - MQ_CONNX_EXIT

MQ_CONNX_EXIT は、MQCONN 処理の前 および後 に実行する出口関数、および MQCONNX 処理の前 および後 に実行する接続拡張出口関数を提供します。

ここで説明するものと同じインターフェースが、MQCONN および MQCONNX 出口呼び出し機能に対して起動されます。

メッセージ・チャンネル・エージェント (MCA) がインバウンド・クライアント接続に応答するとき、MCA はクライアントの状態が完全に判明する前に接続して何回か IBM MQ API 呼び出しを行うことができます。これらの API 呼び出しでは、MCA プログラムそのものに基づく MQAXC を使用して API 出口関数が呼び出されます (MQAXC の UserId および ConnectionName フィールドなど)。

MCA が後続のインバウンド・クライアント API 呼び出しに応答するときは、MQAXC 構造体にはインバウンド・クライアントに基づいて UserId および ConnectionName フィールドが適切に設定されます。

アプリケーションによって MQCONN または MQCONNX 呼び出しに設定されたキュー・マネージャー名は、基礎となる接続呼び出しに渡されます。前 MQ_CONNX_EXIT がキュー・マネージャーの名前を変更しようとしても、それは無効です。

機能 ID MQXF_CONN および MQXF_CONNX に出口理由 MQXR_BEFORE および MQXR_AFTER を指定して、前と後の MQCONN および MQCONNX 呼び出し出口関数を登録します。

理由 MQXR_BEFORE のために呼び出される MQ_CONNX_EXIT 出口は、この時点では正しい環境がセットアップされていないため、IBM MQ API 呼び出しを発行してはなりません。

MQ_CONNX_EXIT では、呼び出しの対象となっている接続に関して API 出口呼び出しから MQDISC を呼び出すことはできません。この制約事項は、クライアントおよびサーバーの両方の API 出口に当てはまります。

MQCONN および MQCONNX へのインターフェースは同一です。

```
MQ_CONNX_EXIT (&ExitParms, &ExitContext, &pQMgrName, &pConnectOpts,  
&pHconn, &CompCode, &Reason);
```

パラメーターは、以下のとおりです。

ExitParms (MQAXP) - 入出力

出口パラメーター構造体。

ExitContext (MQAXC) - 入出力

出口コンテキスト構造体。

pQMgrName (PMQCHAR) - 入力

MQCONNX 呼び出しで指定されるキュー・マネージャー名へのポインター。出口で MQCONN または MQCONNX 呼び出しに指定されたこの名前を変更してはなりません。

pConnectOpts (PMQCN0) - 入出力

MQCONNX 呼び出しのアクションを制御するオプションへのポインター。

詳細は [265 ページの『MQCNO - 接続オプション』](#)を参照してください。

出口関数 MQXF_CONN では、pConnectOpts はデフォルトの接続オプション構造体 (MQCNO_DEFAULT) を指します。

pHconn (PMQHCONN) - 入力

接続ハンドルへのポインター。

CompCode (MQLONG) - 入出力

完了コード、有効な値は以下のとおりです。

MQCC_OK

正常終了。

MQCC_WARNING

警告 (部分完了)

MQCC_FAILED

呼び出し失敗

Reason (MQLONG) - 入出力

完了コードを修飾する理由コード。

完了コードが MQCC_OK の場合、以下の値だけが有効です。

MQRC_NONE

(0, x'000') レポートする理由コードはありません。

完了コードが MQCC_FAILED または MQCC_WARNING の場合、出口関数は理由コード・フィールドを任意の有効な MQRC_* 値に設定できます。

C 言語呼び出し

キュー・マネージャーは、以下の変数を論理的に定義します。

```
MQAXP      ExitParms;      /* Exit parameter structure */
MQAXC      ExitContext;    /* Exit context structure */
PMQCHAR    pQMgrName;      /* Ptr to Queue manager name */
PMQCNO     pConnectOpts;   /* Ptr to Connection options */
PMQHCONN   pHconn;        /* Ptr to Connection handle */
MQLONG     CompCode;       /* Completion code */
MQLONG     Reason;         /* Reason code */
```

その後、キュー・マネージャーは以下のように出口を論理的に呼び出します。

```
MQ_CONNX_EXIT (&ExitParms, &ExitContext, &pQMgrName, &pConnectOpts,
               &pHconn, &CompCode, &Reason);
```

出口は以下の C 関数プロトタイプと一致していなければなりません。

```
void MQENTRY MQ_CONNX_EXIT (
PMQAXP      pExitParms,    /* Address of exit parameter structure */
PMQAXC      pExitContext,  /* Address of exit context structure */
PPMQCHAR    ppQMgrName,    /* Address of ptr to queue manager name */
PPMCNO      ppConnectOpts, /* Address of ptr to connection options */
PPMHCONN    ppHconn,       /* Address of ptr to connection handle */
PMQLONG     pCompCode,     /* Address of completion code */
PMQLONG     pReason);      /* Address of reason code qualifying
                             completion code */
```

使用上の注意

1. ここで説明する MQ_CONNX_EXIT 関数インターフェースは、MQCONN 呼び出しと MQCONNX 呼び出しの両方で使用されます。しかし、この 2 つの呼び出しには、別々のエントリ・ポイントが定義されています。両方の呼び出しをインターセプトするためには、少なくとも 2 回 (関数 ID MQXF_CONN で 1 回、MQXF_CONNX でもう 1 回)、MQXEP 呼び出しを使用しなければなりません。

MQ_CONNX_EXIT インターフェースは、MQCONN でも MQCONNX でも同じであるため、1 つの出口関数を両方の呼び出しに使用できます。どちらの呼び出しが進行中であるかは、MQAXP 構造体の *Function* フィールドに示されます。あるいは、MQXEP 呼び出しを使用して、2 つの呼び出しに別々の出口関数を登録することもできます。

2. メッセージ・チャネル・エージェント (MCA) が着信クライアント接続に応答するとき、MCA は、クライアントの状態が十分にわからなくてもいくつかの MQ 呼び出しを発行することができます。これらの MQ 呼び出しが実行されると、API 出口関数は、クライアント (例えば、ユーザー ID や接続名など) ではなく、MCA に関係したデータが含まれる MQAXC 構造体で呼び出されます。ただし、一度クライアントの状態が完全に認識されたら、それより後に発行される MQ 呼び出しでは、MQAXC 構造体内の該当するクライアント・データで API 出口関数が呼び出されるようになります。
3. MQXR_BEFORE 出口関数はすべて、キュー・マネージャーによってパラメーター検証が実行される前に呼び出されます。したがって、パラメーターが有効でない場合もあります (パラメーターのアドレスを示すポインターが無効な場合など)。

MQ_CONNX_EXIT 関数は、キュー・マネージャーによって許可検査が実行される前に呼び出されます。

4. 出口関数では、MQCONN 呼び出しや MQCONNX 呼び出しで指定されたキュー・マネージャーの名前を変更することはできません。この名前を出口関数で変更した場合の結果については保証できません。

5. MQ_CONNX_EXIT の MQXR_BEFORE 出口関数では、MQXEP 以外の MQ 呼び出しは発行できません。

制御コールバック - MQ_CTL_EXIT

MQ_CTL_EXIT は、制御コールバック処理の前 および後に実行するサブスクリプション要求出口関数を提供します。関数 ID MQXF_CTL に出口理由 MQXR_BEFORE および MQXR_AFTER を指定して、制御コールバック呼び出し前 および制御コールバック呼び出し後 出口関数を登録します。

この関数へのインターフェースは、以下のとおりです。

```
MQ_CTL_EXIT (&Hconn, &Operation, &ControlOpts, &CompCode, &Reason)
```

パラメーターは、以下のとおりです。

Hconn (MQHCONN) - 入出力

接続ハンドル。

Operation (MQLONG) - 入出力

指定されたオブジェクト・ハンドルに定義されたコールバックで処理されている操作。

ControlOpts (MQCTLO) - 入出力

MQCTL のアクションを制御するオプション

CompCode (MQLONG) - 入出力

完了コード、有効な値は以下のとおりです。

MQCC_OK

正常終了。

MQCC_WARNING

一部完了。

MQCC_FAILED

呼び出し失敗

Reason (MQLONG) - 入出力

完了コードを修飾する理由コード。

完了コードが MQCC_OK の場合、以下の値だけが有効です。

MQRC_NONE

(0, x'000') レポートする理由コードはありません。

完了コードが MQCC_FAILED または MQCC_WARNING の場合、出口関数は理由コード・フィールドを任意の有効な MQRC_* 値に設定できます。

C 言語呼び出し

キュー・マネージャーは、以下の変数を論理的に定義します。

```
MQHCONN  Hconn;           /* Connection handle */
MQLONG   Operation;       /* Operation being processed */
MQCTLO   ControlOpts;     /* Options that control the action of MQCTL */
MQLONG   CompCode;        /* Completion code */
MQLONG   Reason;          /* Reason code qualifying completion code */
```

その後、キュー・マネージャーは以下のように出口を論理的に呼び出します。

```
MQ_CTL_EXIT (&Hconn, &Operation, &ControlOpts, &CompCode, &Reason);
```

出口は以下の C 関数プロトタイプと一致していなければなりません。

```
void MQENTRY MQ_CTL_EXIT (
PMQHCONN pHconn;           /* Address of connection handle */
PMQLONG  pOperation;       /* Address of operation being processed */
PMQCTLO  pControlOpts;     /* Address of options that control the action of MQCTL */
PMQLONG  pCompCode;        /* Address of completion code */
PMQLONG  pReason;          /* Address of reason code qualifying completion code */
```

切断 - MQ_DISC_EXIT

MQ_DISC_EXIT は、MQDISC 出口処理の前 と後 に実行する切断出口関数を提供します。関数識別子 MQXF_DISC と終了理由 MQXR_BEFORE および MQXR_AFTER を使用して、MQDISC 呼び出し出口関数の前と後に登録します。

この関数へのインターフェースは、以下のとおりです。

```
MQ_DISC_EXIT (&ExitParms, &ExitContext, &pHconn,  
&CompCode, &Reason);
```

パラメーターは、以下のとおりです。

ExitParms (MQAXP) - 入出力

出口パラメーター構造体。

ExitContext (MQAXC) - 入出力

出口コンテキスト構造体。

pHconn (PMQHCONN) - 入力

接続ハンドルへのポインター。

前 MQDISC 呼び出し では、このフィールドの値は以下の 1 つです。

- MQCONN または MQCONNX 呼び出しで戻された接続ハンドル
- 環境に固有のアダプターがキュー・マネージャーに接続されている環境では、ゼロ
- 出口関数の前回の起動によって設定された値

後 MQDISC 呼び出し では、このフィールドの値はゼロ、または出口関数の前回の起動によって設定された値です。

CompCode (MQLONG) - 入出力

完了コード、有効な値は以下のとおりです。

MQCC_OK

正常終了。

MQCC_WARNING

一部完了

MQCC_FAILED

呼び出し失敗

Reason (MQLONG) - 入出力

完了コードを修飾する理由コード。

完了コードが MQCC_OK の場合、以下の値だけが有効です。

MQRC_NONE

(0, x'000') レポートする理由コードはありません。

完了コードが MQCC_FAILED または MQCC_WARNING の場合、出口関数は理由コード・フィールドを任意の有効な MQRC_* 値に設定できます。

C 言語呼び出し

キュー・マネージャーは、以下の変数を論理的に定義します。

MQAXP	ExitParms;	/* Exit parameter structure */
MQAXC	ExitContext;	/* Exit context structure */
PMQHCONN	pHconn;	/* Ptr to Connection handle */
MQLONG	CompCode;	/* Completion code */
MQLONG	Reason;	/* Reason code */

その後、キュー・マネージャーは以下のように出口を論理的に呼び出します。

```
MQ_DISC_EXIT (&ExitParms, &ExitContext, &pHconn,  
&CompCode, &Reason);
```

出口は以下の C 関数プロトタイプと一致していなければなりません。

```
void MQENTRY MQ_DISC_EXIT (
    PMQAXP      pExitParms,      /* Address of exit parameter structure */
    PMQAXC      pExitContext,    /* Address of exit context structure */
    PMQHCONN    ppHconn,         /* Address of ptr to connection handle */
    PMQLONG     pCompCode,        /* Address of completion code */
    PMQLONG     pReason);        /* Address of reason code qualifying
                                completion code */
```

取得 - MQ_GET_EXIT

MQ_GET_EXIT は、前 および 後 MQGET 呼び出し処理を実行するための GET 出口機能を提供します。

機能 ID には、次の 2 つがあります。

1. MQXF_GET に出口理由 MQXR_BEFORE および MQXR_AFTER を指定して、MQGET 呼び出し出口関数の前と後を登録します。
2. MQXF_DATA_CONV_ON_GET 関数 ID の使用法については、[1594 ページの『MQXF_DATA_CONV_ON_GET』](#)を参照してください。

この関数へのインターフェースは、以下のとおりです。

```
MQ_GET_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &pMsgDesc,
             &pGetMsgOpts, &BufferLength, &pBuffer, &pDataLength,
             &CompCode, &Reason)
```

パラメーターは、以下のとおりです。

ExitParms (MQAXP) - 入出力

出口パラメーター構造体。

ExitContext (MQAXC) - 入出力

出口コンテキスト構造体。

Hconn (MQHCONN) - 入力

接続ハンドル。

Hobj (MQHOBJ) - 入出力

オブジェクト・ハンドル

pMsgDesc (PMQMD) - 入出力

メッセージ記述子へのポインター。

pGetMsgOpts (PMQGMO) - 入出力

メッセージ取得オプションへのポインター。

BufferLength (MQLONG) - 入出力

メッセージ・バッファー長。

pBuffer (PMQBYTE) - 入出力

メッセージ・バッファーへのポインター。

pDataLength (PMQLONG) - 入出力

データ長フィールドへのポインター。

CompCode (MQLONG) - 入出力

完了コード、有効な値は以下のとおりです。

MQCC_OK

正常終了。

MQCC_WARNING

一部完了。

MQCC_FAILED

呼び出し失敗

Reason (MQLONG) - 入出力

完了コードを修飾する理由コード。

完了コードが MQCC_OK の場合、以下の値だけが有効です。

MQRC_NONE

(0, x'000') レポートする理由コードはありません。

完了コードが MQCC_FAILED または MQCC_WARNING の場合、出口関数は理由コード・フィールドを任意の有効な MQRC_* 値に設定できます。

C 言語呼び出し

キュー・マネージャーは、以下の変数を論理的に定義します。

```
MQAXP      ExitParms;      /* Exit parameter structure */
MQAXC      ExitContext;    /* Exit context structure */
MQHCONN     Hconn;         /* Connection handle */
MQHOBJ      Hobj;          /* Object handle */
PMQMD       pMsgDesc;      /* Ptr to message descriptor */
PMQPMO      pGetMsgOpts;   /* Ptr to get message options */
MQLONG      BufferLength;   /* Message buffer length */
PMQBYTE     pBuffer;       /* Ptr to message buffer */
PMQLONG     pDataLength;   /* Ptr to data length field */
MQLONG      CompCode;      /* Completion code */
MQLONG      Reason;        /* Reason code */
```

その後、キュー・マネージャーは以下のように出口を論理的に呼び出します。

```
MQ_GET_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &pMsgDesc,
             &pGetMsgOpts, &BufferLength, &pBuffer, &pDataLength,
             &CompCode, &Reason)
```

出口は以下の C 関数プロトタイプと一致していなければなりません。

```
void MQENTRY MQ_GET_EXIT (
    PMQAXP      pExitParms,      /* Address of exit parameter structure */
    PMQAXC      pExitContext,    /* Address of exit context structure */
    PMQHCONN     pHconn,         /* Address of connection handle */
    PMQHOBJ      pHobj,          /* Address of object handle */
    PPMQMD       ppMsgDesc,      /* Address of ptr to message descriptor */
    PPMQPMO      ppGetMsgOpts,   /* Address of ptr to get message options */
    PMQLONG      pBufferLength,  /* Address of message buffer length */
    PMQBYTE     ppBuffer,        /* Address of ptr to message buffer */
    PPMQLONG     ppDataLength,   /* Address of ptr to data length field */
    PMQLONG      pCompCode,      /* Address of completion code */
    PMQLONG      pReason);       /* Address of reason code qualifying
                                completion code */
```

使用上の注意

1. ここで説明する MQ_GET_EXIT 関数インターフェースは、MQXF_GET 出口関数と [1594 ページの『MQXF DATA CONV ON GET』](#) 出口関数の両方に使用されます。

この 2 つの出口関数には別々のエントリー・ポイントが定義されているため、両方ともインターセプトするためには、MQXEP 呼び出しを 2 回使用しなければなりません。この呼び出しには、関数 ID MQXF_GET を使用します。

MQ_GET_EXIT インターフェースは、MQXF_GET でも MQXF_DATA_CONV_ON_GET でも同じであるため、1 つの出口関数を両方に使用することができます。どちらの出口関数が呼び出されているかは、MQAXP 構造体の *Function* フィールドに示されます。あるいは、MQXEP 呼び出しを使用して、2 つのケースに別々の出口関数を登録することもできます。

MQXF_DATA_CONV_ON_GET

MQXF_DATA_CONV_ON_GET 関数 ID は MQ_GET_EXIT と一緒に使用されます。

この呼び出しのインターフェースについて、および C 言語宣言のサンプルについては、[MQ_GET_EXIT](#) を参照してください。

使用上の注意

登録されている場合、このエントリー・ポイントは、メッセージがアプリケーションに到着し、データ変換が行われる前に呼び出されます。これは、API 出口が復号や解凍といった処理を、メッセージがデータ

変換に渡される前に実行する必要がある場合に便利です。必要に応じて、出口は `MQXCC_SUPPRESS_FUNCTION` を返すことによりデータ変換をバイパスさせることができます。詳しくは、`MQAXP` 構造体を参照してください。

クライアント上でこのエントリー・ポイントを登録すると、データ変換がクライアント・マシン上でローカルに実行されるようになります。そのため、正しい操作が行われるようにするには、アプリケーション変換出口をクライアントにインストールすることが必要になる場合があります。`MQXF_DATA_CONV_ON_GET` は非同期コンシュームにも使用されます。

`MQ_GET_EXIT` 呼び出しを使用する場合は、`MQXF_DATA_CONV_ON_GET` に出口理由 `MQXR_BEFORE` を指定して、`MQGET` データ変換前 出口関数を登録します。

`MQXF_DATA_CONV_ON_GET` には `MQXR_AFTER` 出口関数はありません。データ変換後の出口処理に必要な機能は、`MQXF_GET` の `MQXR_AFTER` 出口関数によって提供されます。

`MQ_GET_EXIT` 呼び出しには個別のエントリー・ポイントが定義されているので、両方の出口関数をインターセプトするためには、`MQXEP` 呼び出しを 2 回使用しなければなりません。この呼び出しには、関数 ID `MQXF_DATA_CONV_ON_GET` を使用します。

`MQ_GET_EXIT` インターフェースは、`MQXF_GET` でも `MQXF_DATA_CONV_ON_GET` でも同じであるため、1 つの出口関数を両方に使用することができます。どちらの出口関数が呼び出されているかは、`MQAXP` 構造体の `Function` フィールドに示されます。あるいは、`MQXEP` 呼び出しを使用して、2 つのケースに別々の出口関数を登録することもできます。

初期化 - `MQ_INIT_EXIT`

`MQ_INIT_EXIT` は、`MQAXP` 内の `ExitReason` を `MQXR_CONNECTION` に設定することによって示される接続レベルの初期化を提供します。

初期化の際に、以下の事柄に注意してください。

- `MQ_INIT_EXIT` 機能は `MQXEP` を呼び出して、インタレストのある IBM MQ API verb および入り口点と出口点を登録します。
- 出口はすべての IBM MQ API verb をインターセプトする必要はありません。出口関数が起動されるのは、インタレストが登録されている場合だけです。
- 出口が使用するストレージは、初期化の際に取得することができます。
- この機能への呼び出しが失敗した場合、これを起動した `MQCONN` または `MQCONN`X 呼び出しも、`MQAXP` 内の `ExitResponse` フィールドの値に依存する `CompCode` と `Reason` を出して失敗します。
- この時点では適切な環境がセットアップされていないので、`MQ_INIT_EXIT` 出口が IBM MQ API 呼び出しを発行できません。
- `MQ_INIT_EXIT` が `MQXCC_FAILED` を出して失敗した場合、キュー・マネージャーはそれを呼び出した `MQCONN` または `MQCONN`X 呼び出しから `MQCC_FAILED` と `MQRC_API_EXIT_ERROR` を出して戻ります。
- キュー・マネージャーが最初の `MQ_INIT_EXIT` を起動する前に、API 出口関数実行環境を初期化していてエラーを検出した場合、キュー・マネージャーは `MQ_INIT_EXIT` を起動した `MQCONN` または `MQCONN`X 呼び出しから `MQCC_FAILED` と `MQRC_API_EXIT_INIT_ERROR` を出して戻ります。

`MQ_INIT_EXIT` へのインターフェースは、以下のとおりです。

```
MQ_INIT_EXIT (&ExitParms, &ExitContext, &CompCode, &Reason)
```

パラメーターは、以下のとおりです。

ExitParms (MQAXP) - 入出力

出口パラメーター構造体。

ExitContext (MQAXC) - 入出力

出口コンテキスト構造体。

CompCode (MQLONG) - 入出力

完了コードへのポインター、有効な値は以下のとおりです。

MQCC_OK

正常終了。

MQCC_WARNING

一部完了。

MQCC_FAILED

呼び出し失敗

Reason (MQLONG) - 入出力

完了コードを修飾する理由コードへのポインター。

完了コードが MQCC_OK の場合、以下の値だけが有効です。

MQRC_NONE

(0, x'000') レポートする理由コードはありません。

完了コードが MQCC_FAILED または MQCC_WARNING の場合、出口関数は理由コード・フィールドを任意の有効な MQRC_* 値に設定できます。

アプリケーションに戻される CompCode と Reason は、MQAXP 内の ExitResponse フィールドの値に依存しています。

C 言語呼び出し

キュー・マネージャーは、以下の変数を論理的に定義します。

MQAXP	ExitParms;	/* Exit parameter structure */
MQAXC	ExitContext;	/* Exit context structure */
MQLONG	CompCode;	/* Completion code */
MQLONG	Reason;	/* Reason code */

その後、キュー・マネージャーは以下のように出口を論理的に呼び出します。

```
MQ_INIT_EXIT (&ExitParms, &ExitContext, &CompCode, &Reason)
```

出口は以下の C 関数プロトタイプと一致していなければなりません。

```
void MQENTRY MQ_INIT_EXIT (
    PMQAXP      pExitParms,      /* Address of exit parameter structure */
    PMQAXC      pExitContext,    /* Address of exit context structure */
    PMQLONG     pCompCode,       /* Address of completion code */
    PMQLONG     pReason);        /* Address of reason code qualifying
                                completion code */
```

使用上の注意

1. MQ_INIT_EXIT 関数では、MQXEP 呼び出しを発行して、インターセプトされる特定の MQ 呼び出しに出口関数のアドレスを登録できます。すべての MQ 呼び出しをインターセプトする必要も、MQXR_BEFORE 呼び出しと MQXR_AFTER 呼び出しの両方をインターセプトする必要もありません。例えば、出口スイートでは、選択的に MQPUT の MQXR_BEFORE 呼び出しだけをインターセプトすることもできます。
2. 出口スイートの出口関数で使用するストレージは、MQ_INIT_EXIT 関数で獲得できます。あるいは、出口関数が、呼び出されたときに必要に応じてストレージを獲得するようにもできます。ただし、ストレージは、出口スイートを終了する前にすべて解放する必要があります。ストレージの解放は、MQ_TERM_EXIT 関数を使用して行うこともできますが、先に呼び出された出口関数を使用して行うことも可能です。
3. MQ_INIT_EXIT が MQAXP の ExitResponse フィールドに MQXCC_FAILED を返すか、他の何らかの方法で失敗した場合、MQ_INIT_EXIT を呼び出す原因となった MQCONN 呼び出しまたは MQCONNx 呼び出しも失敗し、**CompCode** パラメーターと **Reason** パラメーターが適切な値に設定されます。
4. MQ_INIT_EXIT 関数では、MQXEP 以外の MQ 呼び出しは発行できません。

照会 - MQ_INQ_EXIT

MQ_INQ_EXIT は、MQINQ 呼び出し処理の前と後に実行する照会出口関数を提供します。関数 ID MQXF_INQ と終了理由 MQXR_BEFORE および MQXR_AFTER を使用して、MQINQ 呼び出し出口関数の前と後に登録します。

この関数へのインターフェースは、以下のとおりです。

```
MQ_INQ_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &SelectorCount,  
             &pSelectors, &IntAttrCount, &pIntAttrs, &CharAttrLength,  
             &pCharAttrs, &CompCode, &Reason)
```

パラメーターは、以下のとおりです。

ExitParms (MQAXP) - 入出力

出口パラメーター構造体。

ExitContext (MQAXC) - 入出力

出口コンテキスト構造体。

Hconn (MQHCONN) - 入力

接続ハンドル。

Hobj (MQHOBJ) - 入力

オブジェクト・ハンドル

SelectorCount (MQLONG) - 入力

セレクターのカウント

pSelectors (PMQLONG) - 入出力

セレクター値の配列へのポインター。

IntAttrCount (MQLONG) - 入力

整数属性のカウント。

pIntAttrs (PMQLONG) - 入出力

整数属性値の配列へのポインター。

CharAttrLength (MQLONG) - 入出力

文字属性配列の長さ。

pCharAttrs (PMQCHAR) - 入出力

文字属性配列へのポインター。

CompCode (MQLONG) - 入出力

完了コード、有効な値は以下のとおりです。

MQCC_OK

正常終了。

MQCC_WARNING

一部完了。

MQCC_FAILED

呼び出し失敗

Reason (MQLONG) - 入出力

完了コードを修飾する理由コード。

完了コードが MQCC_OK の場合、以下の値だけが有効です。

MQRC_NONE

(0, x'000') レポートする理由コードはありません。

完了コードが MQCC_FAILED または MQCC_WARNING の場合、出口関数は理由コード・フィールドを任意の有効な MQRC_* 値に設定できます。

C 言語呼び出し

キュー・マネージャーは、以下の変数を論理的に定義します。

```

MQAXP    ExitParms;          /* Exit parameter structure */
MQAXC    ExitContext;        /* Exit context structure */
MQHCONN  Hconn;              /* Connection handle */
MQHOBJS  Hobj;               /* Object handle */
MQLONG   SelectorCount;      /* Count of selectors */
PMQLONG  pSelectors;          /* Ptr to array of attribute selectors */
MQLONG   IntAttrCount;        /* Count of integer attributes */
PMQLONG  pIntAttrs;           /* Ptr to array of integer attributes */
MQLONG   CharAttrLength;      /* Length of char attributes array */
PMQCHAR  pCharAttrs;          /* Ptr to character attributes */
MQLONG   CompCode;            /* Completion code */
MQLONG   Reason;              /* Reason code qualifying completion code */

```

その後、キュー・マネージャーは以下のように出口を論理的に呼び出します。

```

MQ_INQ_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &SelectorCount,
             &pSelectors, &IntAttrCount, &pIntAttrs, &CharAttrLength,
             &pCharAttrs, &CompCode, &Reason)

```

出口は以下の C 関数プロトタイプと一致していなければなりません。

```

void MQENTRY MQ_INQ_EXIT (
PMQAXP    pExitParms,          /* Address of exit parameter structure */
PMQAXC    pExitContext,        /* Address of exit context structure */
PMQHCONN  pHconn,              /* Address of connection handle */
PMQHOBJS  pHobj,               /* Address of object handle */
PMQLONG   pSelectorCount,      /* Address of selector count */
PPMQLONG  ppSelectors,          /* Address of ptr to array of selectors */
PMQLONG   pIntAttrCount;        /* Address of count of integer attributes */
PPMQLONG  ppIntAttrs,           /* Address of ptr to array of integer attributes */
PMQLONG   pCharAttrLength,      /* Address of character attribute length */
PPMQCHAR  ppCharAttrs,          /* Address of ptr to character attributes array */
PMQLONG   pCompCode,            /* Address of completion code */
PMQLONG   pReason);            /* Address of reason code qualifying completion
                                code */

```

オープン - **MQ_OPEN_EXIT**

MQ_OPEN_EXIT は、MQOPEN 呼び出し処理の前 および後 に実行するオープン出口関数を提供します。関数 ID MQXF_OPEN と終了理由 MQXR_BEFORE および MQXR_AFTER を使用して、MQOPEN 呼び出し出口関数の前と後に登録します。

この関数へのインターフェースは、以下のとおりです。

```

MQ_OPEN_EXIT (&ExitParms, &ExitContext, &Hconn, &pObjDesc, &Options,
             &pHobj, &CompCode, &Reason)

```

パラメーターは、以下のとおりです。

ExitParms (MQAXP) - 入出力

出口パラメーター構造体。

ExitContext (MQAXC) - 入出力

出口コンテキスト構造体。

Hconn (MQHCONN) - 入力

接続ハンドル。

pObjDesc (PMQOD) - 入出力

オブジェクト記述子へのポインター。

Options (MQLONG)- 入出力

オープン・オプション。

pHobj (PMQHOBJS) - 入力

オブジェクト・ハンドルへのポインター。

CompCode (MQLONG) - 入出力

完了コード、有効な値は以下のとおりです。

MQCC_OK

正常終了。

MQCC_WARNING

一部完了

MQCC_FAILED

呼び出し失敗

Reason (MQLONG) - 入出力

完了コードを修飾する理由コード。

完了コードが MQCC_OK の場合、以下の値だけが有効です。

MQRC_NONE

(0, x'000') レポートする理由コードはありません。

完了コードが MQCC_FAILED または MQCC_WARNING の場合、出口関数は理由コード・フィールドを任意の有効な MQRC_* 値に設定できます。

C 言語呼び出し

キュー・マネージャーは、以下の変数を論理的に定義します。

MQAXP	ExitParms;	/* Exit parameter structure */
MQAXC	ExitContext;	/* Exit context structure */
MQHCONN	Hconn;	/* Connection handle */
PMQOD	pObjDesc;	/* Ptr to object descriptor */
MQLONG	Options;	/* Open options */
PMQHOBj	pHobj;	/* Ptr to object handle */
MQLONG	CompCode;	/* Completion code */
MQLONG	Reason;	/* Reason code */

その後、キュー・マネージャーは以下のように出口を論理的に呼び出します。

```
MQ_OPEN_EXIT (&ExitParms, &ExitContext, &Hconn, &pObjDesc, &Options,
               &pHobj, &CompCode, &Reason);
```

出口は以下の C 関数プロトタイプと一致していなければなりません。

```
void MQENTRY MQ_OPEN_EXIT (
    PMQAXP      pExitParms,      /* Address of exit parameter structure */
    PMQAXC      pExitContext,    /* Address of exit context structure */
    PMQHCONN    pHconn,          /* Address of connection handle */
    PPMQOD      ppObjDesc,       /* Address of ptr to object descriptor */
    PMQLONG     pOptions,        /* Address of open options */
    PPMQHOBj    ppHobj,          /* Address of ptr to object handle */
    PMQLONG     pCompCode,       /* Address of completion code */
    PMQLONG     pReason);        /* Address of reason code qualifying
                                completion code */
```

書き込み - MQ_PUT_EXIT

MQ_PUT_EXIT は、MQPUT 呼び出し処理の前 および後 に実行する書き込み出口関数を提供します。関数 ID MQXF_PUT と終了理由 MQXR_BEFORE および MQXR_AFTER を使用して、MQPUT 呼び出し出口関数の前と後に登録します。

この関数へのインターフェースは、以下のとおりです。

```
MQ_PUT_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &pMsgDesc,
             &pPutMsgOpts, &BufferLength, &pBuffer, &CompCode, &Reason)
```

パラメーターは、以下のとおりです。

ExitParms (MQAXP) - 入出力

出口パラメーター構造体。

ExitContext (MQAXC) - 入出力

出口コンテキスト構造体。

Hconn (MQHCONN) - 入力

接続ハンドル。

Hobj (MQHOBJ) - 入出力

オブジェクト・ハンドル

pMsgDesc (PMQMD) - 入出力

メッセージ記述子へのポインター。

pPutMsgOpts (PMQPMO) - 入出力

メッセージ書き込みオプションへのポインター。

BufferLength (MQLONG) - 入出力

メッセージ・バッファー長。

pBuffer (PMQBYTE) - 入出力

メッセージ・バッファーへのポインター。

CompCode (MQLONG) - 入出力

完了コード、有効な値は以下のとおりです。

MQCC_OK

正常終了。

MQCC_WARNING

一部完了。

MQCC_FAILED

呼び出し失敗

Reason (MQLONG) - 入出力

完了コードを修飾する理由コード。

完了コードが MQCC_OK の場合、以下の値だけが有効です。

MQRC_NONE

(0, x'000') レポートする理由コードはありません。

完了コードが MQCC_FAILED または MQCC_WARNING の場合、出口関数は理由コード・フィールドを任意の有効な MQRC_* 値に設定できます。

C 言語呼び出し

キュー・マネージャーは、以下の変数を論理的に定義します。

MQAXP	ExitParms;	/* Exit parameter structure */
MQAXC	ExitContext;	/* Exit context structure */
MQHCONN	Hconn;	/* Connection handle */
MQHOBJ	Hobj;	/* Object handle */
PMQMD	pMsgDesc;	/* Ptr to message descriptor */
PMQPMO	pPutMsgOpts;	/* Ptr to put message options */
MQLONG	BufferLength;	/* Message buffer length */
PMQBYTE	pBuffer;	/* Ptr to message data */
MQLONG	CompCode;	/* Completion code */
MQLONG	Reason;	/* Reason code */

その後、キュー・マネージャーは以下のように出口を論理的に呼び出します。

```
MQ_PUT_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &pMsgDesc,  
             &pPutMsgOpts, &BufferLength, &pBuffer, &CompCode, &Reason)
```

出口は以下の C 関数プロトタイプと一致していなければなりません。

```
void MQENTRY MQ_PUT_EXIT (  
    PMQAXP      pExitParms,      /* Address of exit parameter structure */  
    PMQAXC      pExitContext,    /* Address of exit context structure */  
    PMQHCONN    pHconn,          /* Address of connection handle */  
    PMQHOBJ     pHobj,           /* Address of object handle */  
    PPMQMD      ppMsgDesc,       /* Address of ptr to message descriptor */  
    PPMQPMO     ppPutMsgOpts,    /* Address of ptr to put message options */  
    PMQLONG     pBufferLength,   /* Address of message buffer length */  
    PPMQBYTE     ppBuffer,       /* Address of ptr to message buffer */  
    PMQLONG     pCompCode,       /* Address of completion code */  
    PMQLONG     pReason);        /* Address of reason code qualifying  
                                completion code */
```

使用上の注意

- キュー・マネージャーが生成するレポート・メッセージは、通常の呼び出し処理をスキップします。したがって、このようなメッセージを MQ_PUT_EXIT 関数や MQPUT1 関数でインターセプトすることはできません。しかし、メッセージ・チャンネル・エージェントが生成するレポート・メッセージは通常どおり処理されるので、MQ_PUT_EXIT 関数や MQ_PUT1_EXIT 関数でインターセプトすることが可能です。MCA で生成されたレポート・メッセージすべてを確実にインターセプトするためには、MQ_PUT_EXIT と MQ_PUT1_EXIT の両方を使用する必要があります。

Put1 - MQ_PUT1_EXIT

MQ_PUT1_EXIT は、MQPUT1 呼び出し処理の前 および後 に実行する単一メッセージ書き込みの 出口関数を提供します。関数 ID MQXF_PUT1 に終了理由 MQXR_BEFORE、MQXR_AFTER を付けて、前後 MQPUT1 呼び出し終了関数を登録します。

この関数へのインターフェースは、以下のとおりです。

```
MQ_PUT1_EXIT (&ExitParms, &ExitContext, &Hconn, &pObjDesc, &pMsgDesc,  
&pPutMsgOpts, &BufferLength, &pBuffer, &CompCode, &Reason)
```

パラメーターは、以下のとおりです。

ExitParms (MQAXP) - 入出力

出口パラメーター構造体。

ExitContext (MQAXC) - 入出力

出口コンテキスト構造体。

Hconn (MQHCONN) - 入力

接続ハンドル。

pObjDesc (PMQOD) - 入出力

オブジェクト記述子へのポインター。

pMsgDesc (PMQMD) - 入出力

メッセージ記述子へのポインター。

pPutMsgOpts (PMQPMO) - 入出力

メッセージ書き込みオプションへのポインター。

BufferLength (MQLONG) - 入出力

メッセージ・バッファー長。

pBuffer (PMQBYTE) - 入出力

メッセージ・バッファーへのポインター。

CompCode (MQLONG) - 入出力

完了コード、有効な値は以下のとおりです。

MQCC_OK

正常終了。

MQCC_WARNING

一部完了。

MQCC_FAILED

呼び出し失敗

Reason (MQLONG) - 入出力

完了コードを修飾する理由コード。

完了コードが MQCC_OK の場合、以下の値だけが有効です。

MQRC_NONE

(0, x'000') レポートする理由コードはありません。

完了コードが MQCC_FAILED または MQCC_WARNING の場合、出口関数は理由コード・フィールドを任意の有効な MQRC_* 値に設定できます。

C 言語呼び出し

キュー・マネージャーは、以下の変数を論理的に定義します。

```
MQAXP      ExitParms;      /* Exit parameter structure */
MQAXC      ExitContext;    /* Exit context structure */
MQHCONN    Hconn;         /* Connection handle */
PMQOD      pObjDesc;      /* Ptr to object descriptor */
PMQMD      pMsgDesc;      /* Ptr to message descriptor */
PMQPMO     pPutMsgOpts;    /* Ptr to put message options */
MQLONG     BufferLength;   /* Message buffer length */
PMQBYTE    pBuffer;       /* Ptr to message data */
MQLONG     CompCode;      /* Completion code */
MQLONG     Reason;        /* Reason code */
```

その後、キュー・マネージャーは以下のように出口を論理的に呼び出します。

```
MQ_PUT1_EXIT (&ExitParms, &ExitContext, &Hconn, &pObjDesc, &pMsgDesc,
              &pPutMsgOpts, &BufferLength, &pBuffer, &CompCode, &Reason)
```

出口は以下の C 関数プロトタイプと一致していなければなりません。

```
void MQENTRY MQ_PUT1_EXIT (
PMQAXP      pExitParms,    /* Address of exit parameter structure */
PMQAXC      pExitContext,  /* Address of exit context structure */
PMQHCONN    pHconn,       /* Address of connection handle */
PPMQOD      ppObjDesc,    /* Address of ptr to object descriptor */
PPMQMD      ppMsgDesc,    /* Address of ptr to message descriptor */
PPMQPMO     ppPutMsgOpts,  /* Address of ptr to put message options */
PMQLONG     pBufferLength, /* Address of message buffer length */
PMQBYTE     ppBuffer,      /* Address of ptr to message buffer */
PMQLONG     pCompCode,     /* Address of completion code */
PMQLONG     pReason);     /* Address of reason code qualifying
                           completion code */
```

設定 - MQ_SET_EXIT

MQ_SET_EXIT は、MQSET 呼び出し処理の前 と後 に実行する設定出口関数を提供します。関数 ID MQXF_SET は、出口理由 MQXR_BEFORE および MQXR_AFTER を指定して使用し、前 および 後 MQSET 呼び出し出口関数を登録します。

この関数へのインターフェースは、以下のとおりです。

```
MQ_SET_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &SelectorCount,
             &pSelectors, &IntAttrCount, &pIntAttrs, &CharAttrLength,
             &pCharAttr, &CompCode, &Reason)
```

パラメーターは、以下のとおりです。

ExitParms (MQAXP) - 入出力

出口パラメーター構造体。

ExitContext (MQAXC) - 入出力

出口コンテキスト構造体。

Hconn (MQHCONN) - 入力

接続ハンドル。

Hobj (MQHOBJ) - 入力

オブジェクト・ハンドル

SelectorCount (MQLONG) - 入力

セレクターのカウント

pSelectors (PMQLONG) - 入出力

セレクター値の配列へのポインター。

IntAttrCount (MQLONG) - 入力

整数属性のカウント。

pIntAttrs (PMQLONG) - 入出力

整数属性値の配列へのポインター。

CharAttrLength (MQLONG) - 入出力

文字属性配列の長さ。

pCharAttrs (PMQCHAR) - 入出力

文字属性値へのポインター。

CompCode (MQLONG) - 入出力

完了コード、有効な値は以下のとおりです。

MQCC_OK

正常終了。

MQCC_WARNING

一部完了。

MQCC_FAILED

呼び出し失敗

Reason (MQLONG) - 入出力

完了コードを修飾する理由コード。

完了コードが MQCC_OK の場合、以下の値だけが有効です。

MQRC_NONE

(0, x'000') レポートする理由コードはありません。

完了コードが MQCC_FAILED または MQCC_WARNING の場合、出口関数は理由コード・フィールドを任意の有効な MQRC_* 値に設定できます。

C 言語呼び出し

キュー・マネージャーは、以下の変数を論理的に定義します。

```
MQAXP    ExitParms;          /* Exit parameter structure */
MQAXC    ExitContext;        /* Exit context structure */
MQHCONN  Hconn;              /* Connection handle */
MQHOBJ   Hobj;                /* Object handle */
MQLONG   SelectorCount;      /* Count of selectors */
PMQLONG  pSelectors;          /* Ptr to array of attribute selectors */
MQLONG   IntAttrCount;        /* Count of integer attributes */
PMQLONG  pIntAttrs;           /* Ptr to array of integer attributes */
MQLONG   CharAttrLength;      /* Length of char attributes array */
PMQCHAR  pCharAttrs;          /* Ptr to character attributes */
MQLONG   CompCode;            /* Completion code */
MQLONG   Reason;              /* Reason code qualifying completion code */
```

その後、キュー・マネージャーは以下のように出口を論理的に呼び出します。

```
MQ_SET_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &SelectorCount,
             &pSelectors, &IntAttrCount, &pIntAttrs, &CharAttrLength,
             &pCharAttrs, &CompCode, &Reason)
```

出口は以下の C 関数プロトタイプと一致していなければなりません。

```
void MQENTRY MQ_SET_EXIT (
PMQAXP    pExitParms,          /* Address of exit parameter structure */
PMQAXC    pExitContext,        /* Address of exit context structure */
PMQHCONN  pHconn,              /* Address of connection handle */
PMQHOBJ   pHobj,                /* Address of object handle */
PMQLONG   pSelectorCount,      /* Address of selector count */
PPMQLONG  ppSelectors,          /* Address of ptr to array of selectors */
PMQLONG   pIntAttrCount;        /* Address of count of integer attributes */
PPMQLONG  ppIntAttrs,           /* Address of ptr to array of integer attributes */
PMQLONG   pCharAttrLength,      /* Address of character attribute length */
PPMQCHAR  ppCharAttrs,          /* Address of ptr to character attributes array */
PMQLONG   pCompCode,            /* Address of completion code */
PMQLONG   pReason);             /* Address of reason code qualifying completion
                                code */
```

状況 - MQ_STAT_EXIT

MQ_STAT_EXIT は、MQSTAT 呼び出し処理の前 および後 に実行する状況出口関数を提供します。関数 ID MQXF_STAT に出口理由 MQXR_BEFORE および MQXR_AFTER を指定して、MQSTAT 呼び出し前 および MQSTAT 呼び出し後 出口関数を登録します。

この関数へのインターフェースは、以下のとおりです。

```
MQ_STAT_EXIT (&ExitParms, &ExitContext, &Hconn, &Type, &pStatus  
              &CompCode, &Reason)
```

パラメーターは、以下のとおりです。

ExitParms (MQAXP) - 入出力

出口パラメーター構造体。

ExitContext (MQAXC) - 入出力

出口コンテキスト構造体。

Hconn (MQHCONN) - 入力

接続ハンドル。

Type (MQLONG) - 入力

取得する状況情報のタイプ。

pStatus (PMQSTS) - 出力

状況バッファーへのポインター。

CompCode (MQLONG) - 入出力

完了コード、有効な値は以下のとおりです。

MQCC_OK

正常終了。

MQCC_WARNING

一部完了。

MQCC_FAILED

呼び出し失敗

Reason (MQLONG) - 入出力

完了コードを修飾する理由コード。

完了コードが MQCC_OK の場合、以下の値だけが有効です。

MQRC_NONE

(0, x'000') レポートする理由コードはありません。

完了コードが MQCC_FAILED または MQCC_WARNING の場合、出口関数は理由コード・フィールドを任意の有効な MQRC_* 値に設定できます。

C 言語呼び出し

出口は以下の C 関数プロトタイプと一致していなければなりません。

```
void MQENTRY MQ_STAT_EXIT (  
    PMQAXP    pExitParms,      /* Address of exit parameter structure */  
    PMQAXC    pExitContext,    /* Address of exit context structure */  
    PMQHCONN  pHconn,          /* Address of connection handle */  
    PMQLONG   pType,           /* Address of status type */  
    PPMQSTS   ppStatus,        /* Address of status buffer */  
    PMQLONG   pCompCode,       /* Address of completion code */  
    PMQLONG   pReason);        /* Address of reason code qualifying completion  
                                code */
```

終了 - MQ_TERM_EXIT

MQ_TERM_EXIT は、関数 ID として MQXF_TERM、ExitReason として MQXR_CONNECTION を指定して登録される、接続レベルの終了を提供します。登録されている場合、MQ_TERM_EXIT は、切断要求ごとに 1 回呼び出されます。

終了処理の一部として、出口に必要なくなったストレージを解放し、必要なクリーンアップを実行することができます。

MQ_TERM_EXIT が MQXCC_FAILED を出して失敗した場合、キュー・マネージャーはそれを呼び出した MQDISC 呼び出しから MQCC_FAILED と MQRC_API_EXIT_ERROR を出して戻ります。

キュー・マネージャーが最後の MQ_TERM_EXIT を起動した後に、API 出口関数実行環境を終了していてエラーを検出した場合、キュー・マネージャーは MQ_TERM_EXIT を起動した MQDISC 呼び出しから MQCC_FAILED と MQRC_API_EXIT_TERM_ERROR を出して戻ります。

この関数へのインターフェースは、以下のとおりです。

```
MQ_TERM_EXIT (&ExitParms, &ExitContext, &CompCode, &Reason)
```

パラメーターは、以下のとおりです。

ExitParms (MQAXP) - 入出力

出口パラメーター構造体。

ExitContext (MQAXC) - 入出力

出口コンテキスト構造体。

CompCode (MQLONG) - 入出力

完了コード、有効な値は以下のとおりです。

MQCC_OK

正常終了。

MQCC_FAILED

呼び出し失敗

Reason (MQLONG) - 入出力

完了コードを修飾する理由コード。

完了コードが MQCC_OK の場合、以下の値だけが有効です。

MQRC_NONE

(0, x'000') レポートする理由コードはありません。

完了コードが MQCC_FAILED の場合、出口関数は理由コード・フィールドを任意の有効な MQRC_* 値に設定できます。

アプリケーションに戻される CompCode と Reason は、MQAXP 内の ExitResponse フィールドの値に依存しています。

C 言語呼び出し

キュー・マネージャーは、以下の変数を論理的に定義します。

```
MQAXP      ExitParms;      /* Exit parameter structure */
MQAXC      ExitContext;    /* Exit context structure */
MQLONG     CompCode;       /* Completion code */
MQLONG     Reason;         /* Reason code */
```

その後、キュー・マネージャーは以下のように出口を論理的に呼び出します。

```
MQ_TERM_EXIT (&ExitParms, &ExitContext, &CompCode, &Reason)
```

出口は以下の C 関数プロトタイプと一致していなければなりません。

```
void MQENTRY MQ_TERM_EXIT (
PMQAXP      pExitParms,    /* Address of exit parameter structure */
PMQAXC      pExitContext,  /* Address of exit context structure */
PMQLONG     pCompCode,     /* Address of completion code */
PMQLONG     pReason);      /* Address of reason code qualifying
                             completion code */
```

使用上の注意

1. MQ_TERM_EXIT 関数はオプションです。終了プロセスが行われない出口スイートには、終了出口を登録する必要はありません。

出口スイートに属する関数が接続時にリソースを獲得する場合、MQ_TERM_EXIT 関数が便利なのは、リソースの解放、例えば、動的に獲得したストレージの解放を行えるという特徴があるからです。
2. MQDISC 呼び出しが発行されたときに MQ_TERM_EXIT 関数が登録された場合、この出口関数は、すべての MQDISC 出口関数が呼び出された後に呼び出されます。
3. MQ_TERM_EXIT が、MQAXP の ExitResponse フィールドに MQXCC_FAILED を戻した場合、あるいは、他の何らかの事情で失敗した場合は、MQ_TERM_EXIT を呼び出す MQDISC 呼び出しも失敗し、その **CompCode** パラメーターや **Reason** パラメーターには、該当する値が戻されます。

サブスクリプションの登録 - MQ_SUB_EXIT

MQ_SUB_EXIT は、サブスクリプション再登録処理の前 および後に実行する出口関数を提供します。関数 ID MQXF_SUB と出口理由 MQXR_BEFORE および MQXR_AFTER を指定して、サブスクリプション登録呼び出し前 およびサブスクリプション登録呼び出し後 出口関数を登録します。

この関数へのインターフェースは、以下のとおりです。

```
MQ_SUB_EXIT (&ExitParms, &ExitContext, &Hconn, &pSubDesc, &pHobj, &pHsub, &CompCode, &Reason)
```

パラメーターは、以下のとおりです。

ExitParms (MQAXP) - 入出力

出口パラメーター構造体。

ExitContext (MQAXC) - 入出力

出口コンテキスト構造体。

Hconn (MQHCONN) - 入出力

接続ハンドル。

pSubDesc - 入出力

属性セクターの配列。

pHobj - 入出力

オブジェクト・ハンドル

pHsub (MQHOBJ) 入出力

サブスクリプション・ハンドル

CompCode (MQLONG) - 入出力

完了コード、有効な値は以下のとおりです。

MQCC_OK

正常終了。

MQCC_WARNING

一部完了。

MQCC_FAILED

呼び出し失敗

Reason (MQLONG) - 入出力

完了コードを修飾する理由コード。

完了コードが MQCC_OK の場合、以下の値だけが有効です。

MQRC_NONE

(0, x'000') レポートする理由コードはありません。

完了コードが MQCC_FAILED または MQCC_WARNING の場合、出口関数は理由コード・フィールドを任意の有効な MQRC_* 値に設定できます。

C 言語呼び出し

キュー・マネージャーは、以下の変数を論理的に定義します。

```
MQAXP      ExitParams;      /* Exit parameter structure */
MQAXC      ExitContext;     /* Exit context structure */
MQHCONN    Hconn;           /* Connection handle */
PMQSD      pSubDesc;        /* Subscription descriptor */
PMQHOBJS   pHobj;           /* Object Handle */
PMQHOBJS   pHsub;           /* Subscription handle */
MQLONG     CompCode;        /* Completion code */
MQLONG     Reason;          /* Reason code qualifying completion code */
```

その後、キュー・マネージャーは以下のように出口を論理的に呼び出します。

```
MQ_SUB_EXIT (&ExitParams, &ExitContext, &Hconn, &pSubDesc, &pHobj, &pHsub,
             &CompCode, &Reason);
```

出口は以下の C 関数プロトタイプと一致していなければなりません。

```
PMQAXP      pExitParams;     /* Exit parameter structure */
PMQAXC      pExitContext;    /* Exit context structure */
PMQHCONN    pHconn;          /* Connection handle */
PPMQSD      ppSubDesc;       /* Subscription descriptor */
PPMQHOBJS   ppHobj;          /* Object Handle */
PPMQHOBJS   ppHsub;          /* Subscription handle */
PMQLONG     pCompCode;       /* Completion code */
PMQLONG     pReason;         /* Reason code qualifying completion code */
```

サブスクリプション要求 - *MQ_SUBRQ_EXIT*

MQ_SUBRQ_EXIT は、サブスクリプション要求処理の前 および後に実行するサブスクリプション要求出口関数を提供します。関数 ID MQXF_SUBRQ に出口理由 MQXR_BEFORE および MQXR_AFTER を指定して、サブスクリプション要求呼び出し前 およびサブスクリプション要求呼び出し後 出口関数を登録します。

この関数へのインターフェースは、以下のとおりです。

```
MQ_SUBRQ_EXIT (&ExitParams, &ExitContext, &Hconn, &pHsub, &Action, &pSubRqOpts,
              &CompCode, &Reason)
```

パラメーターは、以下のとおりです。

ExitParams (MQAXP) - 入出力

出口パラメーター構造体。

ExitContext (MQAXC) - 入出力

出口コンテキスト構造体。

Hconn (MQHCONN) - 入出力

接続ハンドル。

pHsub (MQHOBJS) 入出力

サブスクリプション・ハンドル

Action (MQLONG) - 入出力

アクション

pSubRqOpts (MQSRO) - 入出力

CompCode (MQLONG) - 入出力

完了コード、有効な値は以下のとおりです。

MQCC_OK

正常終了。

MQCC_WARNING

一部完了。

MQCC_FAILED

呼び出し失敗

Reason (MQLONG) - 入出力

完了コードを修飾する理由コード。

完了コードが MQCC_OK の場合、以下の値だけが有効です。

MQRC_NONE

(0, x'000') レポートする理由コードはありません。

完了コードが MQCC_FAILED または MQCC_WARNING の場合、出口関数は理由コード・フィールドを任意の有効な MQRC_* 値に設定できます。

C 言語呼び出し

キュー・マネージャーは、以下の変数を論理的に定義します。

```
MQAXP    ExitParms;      /* Exit parameter structure */
MQAXC    ExitContext;    /* Exit context structure */
MQHCONN   Hconn;         /* Connection handle */
PMLONG    pSub;          /* Subscription handle */
MQLONG    Action;        /* Action */
PMSRO     pSubRqOpts;    /* Subscription Request Options */
MQLONG    CompCode;      /* Completion code */
MQLONG    Reason;        /* Reason code qualifying completion code */
```

その後、キュー・マネージャーは以下のように出口を論理的に呼び出します。

```
MQ_SUBRQ_EXIT (&ExitParms, &ExitContext, &Hconn, &pSub, &Action, &pSubRqOpts,
               &CompCode, &Reason);
```

出口は以下の C 関数プロトタイプと一致していなければなりません。

```
void MQENTRY MQ_SUBRQ_EXIT (
PMQAXP    pExitParms,      /* Address of exit parameter structure */
PMQAXC    pExitContext,    /* Address of exit context structure */
PMQHCONN   pHconn,        /* Address of connection handle */
PPMHOB    ppSub;          /* Address of Subscription handle */
PMLONG     pAction;        /* Address of Action */
PMSRO      ppSubRqOpts;    /* Address of Subscription Request Options */
PMLONG     pCompCode,      /* Address of completion code */
PMLONG     pReason);       /* Address of reason code qualifying completion
                             code */
```

xa_close - XA_CLOSE_EXIT

XA_CLOSE_EXIT は、xa_close 処理の前と後に実行する xa_close 出口関数を提供します。関数 ID MQXF_XACLOSE に出口理由 MQXR_BEFORE および MQXR_AFTER を指定して、xa_close の呼び出し前および呼び出し後の出口関数を登録します。

この関数へのインターフェースは、以下のとおりです。

```
XA_CLOSE_EXIT (&ExitParms, &ExitContext, &Hconn, &pXa_info, &Rmid, &Flags, &XARetCode)
```

パラメーターは、以下のとおりです。

ExitParms (MQAXP) - 入出力

出口パラメーター構造体。

ExitContext (MQAXC) - 入出力

出口コンテキスト構造体。

Hconn (MQHCONN) - 入力

接続ハンドル。

pXa_info (PMQCHAR) - 入出力

インスタンスに固有のリソース・マネージャー情報。

Rmid (MQLONG) - 入出力

リソース・マネージャー ID。

Flags (MQLONG) - 入出力

リソース・マネージャー・オプション。

XARetCode (MQLONG) - 入出力

XA 呼び出しからの応答。

C 言語呼び出し

キュー・マネージャーは、以下の変数を論理的に定義します。

```
MQAXP    ExitParms;    /* Exit parameter structure */
MQAXC    ExitContext;  /* Exit context structure */
MQHCONN   Hconn;       /* Connection handle */
PMQCHAR   pXa_info;    /* Instance-specific RM info */
MQLONG    Rmid;        /* Resource manager identifier */
MQLONG    Flags;       /* Resource manager options*/
MQLONG    XARetCode;   /* Response from XA call */
```

その後、キュー・マネージャーは以下のように出口を論理的に呼び出します。

```
XA_CLOSE_EXIT (&ExitParms, &ExitContext, &Hconn, &pXa_info, &Rmid, &Flags, &XARetCode);
```

出口は以下の C 関数プロトタイプと一致していなければなりません。

```
typedef void MQENTRY XA_CLOSE_EXIT (
    PMQAXP    pExitParms,    /* Address of exit parameter structure */
    PMQAXC    pExitContext,  /* Address of exit context structure */
    PMQHCONN   pHconn,       /* Address of connection handle */
    PPMQCHAR   ppXa_info,    /* Address of instance-specific RM info */
    PMQLONG    pRmid,        /* Address of resource manager identifier */
    PMQLONG    pFlags,       /* Address of resource manager options*/
    PMQLONG    pXARetCode);  /* Address of response from XA call */
```

xa_commit - XA_COMMIT_EXIT

XA_COMMIT_EXIT は、xa_commit 処理の前と後に実行する xa_commit 出口関数を提供します。関数 ID MQXF_XACOMMIT に出口理由 MQXR_BEFORE および MQXR_AFTER を指定して、xa_commit の呼び出し前および呼び出し後の出口関数を登録します。

この関数へのインターフェースは、以下のとおりです。

```
XA_COMMIT_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode)
```

パラメーターは、以下のとおりです。

ExitParms (MQAXP) - 入出力

出口パラメーター構造体。

ExitContext (MQAXC) - 入出力

出口コンテキスト構造体。

Hconn (MQHCONN) - 入力

接続ハンドル。

pXID (MQPTR) - 入出力

トランザクション・ブランチ ID。

Rmid (MQLONG) - 入出力

リソース・マネージャー ID。

Flags (MQLONG) - 入出力

リソース・マネージャー・オプション。

XARetCode (MQLONG) - 入出力

XA 呼び出しからの応答。

C 言語呼び出し

キュー・マネージャーは、以下の変数を論理的に定義します。

```
MQAXP    ExitParms;    /* Exit parameter structure */
MQAXC    ExitContext;  /* Exit context structure */
MQHCONN   Hconn;       /* Connection handle */
```

```

MQPTR    pXID;          /* Transaction branch ID */
MQLONG   Rmid;          /* Resource manager identifier */
MQLONG   Flags;         /* Resource manager options*/
MQLONG   XARetCode;     /* Response from XA call */

```

その後、キュー・マネージャーは以下のように出口を論理的に呼び出します。

```
XA_COMMIT_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode);
```

出口は以下の C 関数プロトタイプと一致していなければなりません。

```

typedef void MQENTRY XA_COMMIT_EXIT (
    PMQAXP    pExitParms, /* Address of exit parameter structure */
    PMQAXC    pExitContext, /* Address of exit context structure */
    PMQHCONN  pHconn,      /* Address of connection handle */
    MQPTR     ppXID,        /* Address of transaction branch ID */
    PMQLONG   pRmid,        /* Address of resource manager identifier */
    PMQLONG   pFlags,       /* Address of resource manager options*/
    PMQLONG   pXARetCode); /* Address of response from XA call */

```

xa_complete - XA_COMPLETE_EXIT

XA_COMPLETE_EXIT は、xa_complete 処理の前と後に実行する xa_complete 出口関数を提供します。関数 ID MQXF_XACOMplete に出口理由 MQXR_BEFORE および MQXR_AFTER を指定して、xa_complete の呼び出し前および呼び出し後の出口関数を登録します。

この関数へのインターフェースは、以下のとおりです。

```
XA_COMPLETE_EXIT (&ExitParms, &ExitContext, &Hconn, &pHandle, &pRetVal, &Rmid, &Flags, &XARetCode)
```

パラメーターは、以下のとおりです。

ExitParms (MQAXP) - 入出力

出口パラメーター構造体。

ExitContext (MQAXC) - 入出力

出口コンテキスト構造体。

Hconn (MQHCONN) - 入力

接続ハンドル。

pHandle (PMQLONG) - 入出力

非同期操作へのポインター。

pRetVal (PMQLONG) - 入出力

非同期操作の戻り値。

Rmid (MQLONG) - 入出力

リソース・マネージャー ID。

Flags (MQLONG) - 入出力

リソース・マネージャー・オプション。

XARetCode (MQLONG) - 入出力

XA 呼び出しからの応答。

C 言語呼び出し

キュー・マネージャーは、以下の変数を論理的に定義します。

```

MQAXP    ExitParms;     /* Exit parameter structure */
MQAXC    ExitContext;   /* Exit context structure */
MQHCONN  Hconn;         /* Connection handle */
PMQLONG  pHandle;       /* Ptr to asynchronous op */
PMQLONG  pRetVal;       /* Return value of async op */
MQLONG   Rmid;          /* Resource manager identifier */
MQLONG   Flags;         /* Resource manager options*/
MQLONG   XARetCode;     /* Response from XA call */

```

その後、キュー・マネージャーは以下のように出口を論理的に呼び出します。

```
XA_COMPLETE_EXIT (&ExitParms, &ExitContext, &Hconn, &pHandle, &pRetVal, &Rmid, &Flags,
&XARetCode);
```

出口は以下の C 関数プロトタイプと一致していなければなりません。

```
typedef void MQENTRY XA_COMPLETE_EXIT (
    PMQAXP  pExitParms, /* Address of exit parameter structure */
    PMQAXC  pExitContext, /* Address of exit context structure */
    PMQHCONN pHconn, /* Address of connection handle */
    PPMQLONG ppHandle, /* Address of ptr to asynchronous op */
    PPMQLONG ppRetVal, /* Address of return value of async op */
    PMQLONG pRmid, /* Address of resource manager identifier */
    PMQLONG pFlags, /* Address of resource manager options */
    PMQLONG pXARetCode); /* Address of response from XA call */
```

xa_end - XA_END_EXIT

XA_END_EXIT は、xa_end 処理の前と後に実行する xa_end 出口関数を提供します。関数 ID MQXF_XAEND に出口理由 MQXR_BEFORE および MQXR_AFTER を指定して、xa_end の呼び出し前および呼び出し後の出口関数を登録します。

この関数へのインターフェースは、以下のとおりです。

```
XA_END_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode)
```

パラメーターは、以下のとおりです。

ExitParms (MQAXP) - 入出力

出口パラメーター構造体。

ExitContext (MQAXC) - 入出力

出口コンテキスト構造体。

Hconn (MQHCONN) - 入力

接続ハンドル。

pXID (MQPTR) - 入出力

トランザクション・ブランチ ID。

Rmid (MQLONG) - 入出力

リソース・マネージャー ID。

Flags (MQLONG) - 入出力

リソース・マネージャー・オプション。

XARetCode (MQLONG) - 入出力

XA 呼び出しからの応答。

C 言語呼び出し

キュー・マネージャーは、以下の変数を論理的に定義します。

```
MQAXP  ExitParms; /* Exit parameter structure */
MQAXC  ExitContext; /* Exit context structure */
MQHCONN Hconn; /* Connection handle */
MQPTR  pXID; /* Transaction branch ID */
MQLONG Rmid; /* Resource manager identifier */
MQLONG Flags; /* Resource manager options */
MQLONG XARetCode; /* Response from XA call */
```

その後、キュー・マネージャーは以下のように出口を論理的に呼び出します。

```
XA_END_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode);
```

出口は以下の C 関数プロトタイプと一致していなければなりません。

```
typedef void MQENTRY XA_END_EXIT (
    PMQAXP  pExitParms, /* Address of exit parameter structure */
    PMQAXC  pExitContext, /* Address of exit context structure */
    PMQHCONN pHconn, /* Address of connection handle */
    PMQPTR  ppXID, /* Address of transaction branch ID */
```

```
PMQLONG pRmid, /* Address of resource manager identifier */
PMQLONG pFlags, /* Address of resource manager options*/
PMQLONG pXARetCode); /* Address of response from XA call */
```

xa_forget - XA_FORGET_EXIT

XA_FORGET_EXIT は、xa_forget 処理の前と後に実行する xa_forget 出口関数を提供します。関数 ID MQXF_XAFORGET に出口理由 MQXR_BEFORE および MQXR_AFTER を指定して、xa_forget の呼び出し前および呼び出し後の出口関数を登録します。

この関数へのインターフェースは、以下のとおりです。

```
XA_FORGET_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode)
```

パラメーターは、以下のとおりです。

ExitParms (MQAXP) - 入出力

出口パラメーター構造体。

ExitContext (MQAXC) - 入出力

出口コンテキスト構造体。

Hconn (MQHCONN) - 入力

接続ハンドル。

pXID (MQPTR) - 入出力

トランザクション・ブランチ ID。

Rmid (MQLONG) - 入出力

リソース・マネージャー ID。

Flags (MQLONG) - 入出力

リソース・マネージャー・オプション。

XARetCode (MQLONG) - 入出力

XA 呼び出しからの応答。

C 言語呼び出し

キュー・マネージャーは、以下の変数を論理的に定義します。

```
MQAXP ExitParms; /* Exit parameter structure */
MQAXC ExitContext; /* Exit context structure */
MQHCONN Hconn; /* Connection handle */
MQPTR pXID; /* Transaction branch ID */
MQLONG Rmid; /* Resource manager identifier */
MQLONG Flags; /* Resource manager options*/
MQLONG XARetCode; /* Response from XA call */
```

その後、キュー・マネージャーは以下のように出口を論理的に呼び出します。

```
XA_FORGET_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode);
```

出口は以下の C 関数プロトタイプと一致していなければなりません。

```
typedef void MQENTRY XA_FORGET_EXIT (
    PMQAXP pExitParms, /* Address of exit parameter structure */
    PMQAXC pExitContext, /* Address of exit context structure */
    PMQHCONN pHconn, /* Address of connection handle */
    PMQPTR ppXID, /* Address of transaction branch ID */
    PMQLONG pRmid, /* Address of resource manager identifier */
    PMQLONG pFlags, /* Address of resource manager options*/
    PMQLONG pXARetCode); /* Address of response from XA call */
```

xa_open - XA_OPEN_EXIT

XA_OPEN_EXIT は、xa_open 処理の前と後に実行する xa_open 出口関数を提供します。関数 ID MQXF_XAOPEN に出口理由 MQXR_BEFORE および MQXR_AFTER を指定して、xa_open の呼び出し前および呼び出し後の出口関数を登録します。

この関数へのインターフェースは、以下のとおりです。


```
XA_OPEN_EXIT (&ExitParms, &ExitContext, &Hconn, &pXa_info, &Rmid, &Flags, &XARetCode)
```

パラメーターは、以下のとおりです。

ExitParms (MQAXP) - 入出力

出口パラメーター構造体。

ExitContext (MQAXC) - 入出力

出口コンテキスト構造体。

Hconn (MQHCONN) - 入力

接続ハンドル。

pXa_info (PMQCHAR) - 入出力

インスタンスに固有のリソース・マネージャー情報。

Rmid (MQLONG) - 入出力

リソース・マネージャー ID。

Flags (MQLONG) - 入出力

リソース・マネージャー・オプション。

XARetCode (MQLONG) - 入出力

XA 呼び出しからの応答。

C 言語呼び出し

キュー・マネージャーは、以下の変数を論理的に定義します。

```
MQAXP   ExitParms;    /* Exit parameter structure */
MQAXC   ExitContext;  /* Exit context structure */
MQHCONN Hconn;        /* Connection handle */
PMQCHAR pXa_info;     /* Instance-specific RM info */
MQLONG  Rmid;         /* Resource manager identifier */
MQLONG  Flags;        /* Resource manager options*/
MQLONG  XARetCode;    /* Response from XA call */
```

その後、キュー・マネージャーは以下のように出口を論理的に呼び出します。

```
XA_OPEN_EXIT (&ExitParms, &ExitContext, &Hconn, &pXa_info, &Rmid, &Flags, &XARetCode);
```

出口は以下の C 関数プロトタイプと一致していなければなりません。

```
typedef void MQENTRY XA_OPEN_EXIT (
    PMQAXP   pExitParms, /* Address of exit parameter structure */
    PMQAXC   pExitContext, /* Address of exit context structure */
    PMQHCONN pHconn,      /* Address of connection handle */
    PPMQCHAR ppXa_info,   /* Address of instance-specific RM info */
    PMQLONG  pRmid,       /* Address of resource manager identifier */
    PMQLONG  pFlags,      /* Address of resource manager options*/
    PMQLONG  pXARetCode); /* Address of response from XA call */
```

xa_prepare - XA_PREPARE_EXIT

XA_PREPARE_EXIT は、xa_prepare 処理の前と後に実行する xa_prepare 出口関数を提供します。関数 ID MQXF_XAPREPARE に出口理由 MQXR_BEFORE および MQXR_AFTER を指定して、xa_prepare の呼び出し前および呼び出し後の出口関数を登録します。

この関数へのインターフェースは、以下のとおりです。

```
XA_PREPARE_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode)
```

パラメーターは、以下のとおりです。

ExitParms (MQAXP) - 入出力

出口パラメーター構造体。

ExitContext (MQAXC) - 入出力

出口コンテキスト構造体。

Hconn (MQHCONN) - 入力

接続ハンドル。

pXID (MQPTR) - 入出力

トランザクション・ブランチ ID。

Rmid (MQLONG) - 入出力

リソース・マネージャー ID。

Flags (MQLONG) - 入出力

リソース・マネージャー・オプション。

XARetCode (MQLONG) - 入出力

XA 呼び出しからの応答。

C 言語呼び出し

キュー・マネージャーは、以下の変数を論理的に定義します。

```
MQAXP  ExitParms; /* Exit parameter structure */
MQAXC  ExitContext; /* Exit context structure */
MQHCONN Hconn; /* Connection handle */
MQPTR  pXID; /* Transaction branch ID */
MQLONG Rmid; /* Resource manager identifier */
MQLONG Flags; /* Resource manager options*/
MQLONG XARetCode; /* Response from XA call */
```

その後、キュー・マネージャーは以下のように出口を論理的に呼び出します。

```
XA_PREPARE_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode);
```

出口は以下の C 関数プロトタイプと一致していなければなりません。

```
typedef void MQENTRY XA_PREPARE_EXIT (
    PMQAXP  pExitParms, /* Address of exit parameter structure */
    PMQAXC  pExitContext, /* Address of exit context structure */
    PMQHCONN pHconn, /* Address of connection handle */
    PMQPTR  ppXID, /* Address of transaction branch ID */
    PMQLONG pRmid, /* Address of resource manager identifier */
    PMQLONG pFlags, /* Address of resource manager options*/
    PMQLONG pXARetCode); /* Address of response from XA call */
```

***xa_recover* - XA_RECOVER_EXIT**

XA_RECOVER_EXIT は、xa_recover 処理の前と後に実行する xa_recover 出口関数を提供します。関数 ID MQXF_XARECOVER に出口理由 MQXR_BEFORE および MQXR_AFTER を指定して、xa_recover の呼び出し前および呼び出し後の出口関数を登録します。

この関数へのインターフェースは、以下のとおりです。

```
XA_RECOVER_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Count, &Rmid, &Flags, &XARetCode)
```

パラメーターは、以下のとおりです。

ExitParms (MQAXP) - 入出力

出口パラメーター構造体。

ExitContext (MQAXC) - 入出力

出口コンテキスト構造体。

Hconn (MQHCONN) - 入力

接続ハンドル。

pXID (MQPTR) - 入出力

トランザクション・ブランチ ID。

Count (MQLONG) - 入出力

XID 配列に含まれる最大 XID 数。

Rmid (MQLONG) - 入出力

リソース・マネージャー ID。

Flags (MQLONG) - 入出力

リソース・マネージャー・オプション。

XARetCode (MQLONG) - 入出力

XA 呼び出しからの応答。

C 言語呼び出し

キュー・マネージャーは、以下の変数を論理的に定義します。

```
MQAXP   ExitParms;    /* Exit parameter structure */
MQAXC   ExitContext;  /* Exit context structure */
MQHCONN Hconn;        /* Connection handle */
MQPTR   pXID;         /* Transaction branch ID */
MQLONG  Count;        /* Max XIDs in XID array */
MQLONG  Rmid;         /* Resource manager identifier */
MQLONG  Flags;        /* Resource manager options */
MQLONG  XARetCode;    /* Response from XA call */
```

その後、キュー・マネージャーは以下のように出口を論理的に呼び出します。

```
XA_RECOVER_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Count, &Rmid, &Flags, &XARetCode);
```

出口は以下の C 関数プロトタイプと一致していなければなりません。

```
typedef void MQENTRY XA_RECOVER_EXIT (
    PMQAXP pExitParms, /* Address of exit parameter structure */
    PMQAXC pExitContext, /* Address of exit context structure */
    PMQHCONN pHconn, /* Address of connection handle */
    PMQPTR ppXID, /* Address of transaction branch ID */
    PMQLONG pCount, /* Address of max XIDs in XID array */
    PMQLONG pRmid, /* Address of resource manager identifier */
    PMQLONG pFlags, /* Address of resource manager options */
    PMQLONG pXARetCode); /* Address of response from XA call */
```

xa_rollback - XA_ROLLBACK_EXIT

XA_ROLLBACK_EXIT は、xa_rollback 処理の前と後に実行する xa_rollback 出口関数を提供します。関数 ID MQXF_XAROLLBACK に出口理由 MQXR_BEFORE および MQXR_AFTER を指定して、xa_rollback の呼び出し前および呼び出し後の出口関数を登録します。

この関数へのインターフェースは、以下のとおりです。

```
XA_ROLLBACK_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode)
```

パラメーターは、以下のとおりです。

ExitParms (MQAXP) - 入出力

出口パラメーター構造体。

ExitContext (MQAXC) - 入出力

出口コンテキスト構造体。

Hconn (MQHCONN) - 入力

接続ハンドル。

pXID (MQPTR) - 入出力

トランザクション・ブランチ ID。

Rmid (MQLONG) - 入出力

リソース・マネージャー ID。

Flags (MQLONG) - 入出力

リソース・マネージャー・オプション。

XARetCode (MQLONG) - 入出力

XA 呼び出しからの応答。

C 言語呼び出し

キュー・マネージャーは、以下の変数を論理的に定義します。

```
MQAXP   ExitParms;    /* Exit parameter structure */
MQAXC   ExitContext;  /* Exit context structure */
MQHCONN Hconn;        /* Connection handle */
MQPTR   pXID;         /* Transaction branch ID */
MQLONG  Rmid;         /* Resource manager identifier */
MQLONG  Flags;        /* Resource manager options*/
MQLONG  XARetCode;    /* Response from XA call */
```

その後、キュー・マネージャーは以下のように出口を論理的に呼び出します。

```
XA_ROLLBACK_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode);
```

出口は以下の C 関数プロトタイプと一致していなければなりません。

```
typedef void MQENTRY XA_ROLLBACK_EXIT (
    PMQAXP pExitParms, /* Address of exit parameter structure */
    PMQAXC pExitContext, /* Address of exit context structure */
    PMQHCONN pHconn, /* Address of connection handle */
    PMQPTR ppXID, /* Address of transaction branch ID */
    PMQLONG pRmid, /* Address of resource manager identifier */
    PMQLONG pFlags, /* Address of resource manager options*/
    PMQLONG pXARetCode); /* Address of response from XA call */
```

xa_start - XA_START_EXIT

XA_START_EXIT は、xa_start 処理の前と後に実行する xa_start 出口関数を提供します。関数 ID MQXF_XASTART に出口理由 MQXR_BEFORE および MQXR_AFTER を指定して、xa_start の呼び出し前および呼び出し後の出口関数を登録します。

この関数へのインターフェースは、以下のとおりです。

```
XA_START_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode)
```

パラメーターは、以下のとおりです。

ExitParms (MQAXP) - 入出力

出口パラメーター構造体。

ExitContext (MQAXC) - 入出力

出口コンテキスト構造体。

Hconn (MQHCONN) - 入力

接続ハンドル。

pXID (MQPTR) - 入出力

トランザクション・ブランチ ID。

Rmid (MQLONG) - 入出力

リソース・マネージャー ID。

Flags (MQLONG) - 入出力

リソース・マネージャー・オプション。

XARetCode (MQLONG) - 入出力

XA 呼び出しからの応答。

C 言語呼び出し

キュー・マネージャーは、以下の変数を論理的に定義します。

```
MQAXP   ExitParms;    /* Exit parameter structure */
MQAXC   ExitContext;  /* Exit context structure */
MQHCONN Hconn;        /* Connection handle */
MQPTR   pXID;         /* Transaction branch ID */
MQLONG  Rmid;         /* Resource manager identifier */
MQLONG  Flags;        /* Resource manager options*/
MQLONG  XARetCode;    /* Response from XA call */
```

その後、キュー・マネージャーは以下のように出口を論理的に呼び出します。

```
XA_START_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode);
```

出口は以下の C 関数プロトタイプと一致していなければなりません。

```
typedef void MQENTRY XA_START_EXIT (  
    PMQAXP  pExitParms, /* Address of exit parameter structure */  
    PMQAXC  pExitContext, /* Address of exit context structure */  
    PMQHCONN pHconn, /* Address of connection handle */  
    PMQPTR  ppXID, /* Address of transaction branch ID */  
    PMQLONG pRmid, /* Address of resource manager identifier */  
    PMQLONG pFlags, /* Address of resource manager options */  
    PMQLONG pXARetCode); /* Address of response from XA call */
```

ax_reg - AX_REG_EXIT

AX_REG_EXIT は、ax_reg 処理の前と後に実行する ax_reg 出口関数を提供します。関数 ID MQXF_AXREG に出口理由 MQXR_BEFORE および MQXR_AFTER を指定して、ax_reg の呼び出し前および呼び出し後の出口関数を登録します。

この関数へのインターフェースは、以下のとおりです。

```
AX_REG_EXIT (&ExitParms, &ExitContext, &pXID, &Rmid, &Flags, &XARetCode)
```

パラメーターは、以下のとおりです。

ExitParms (MQAXP) - 入出力

出口パラメーター構造体。

ExitContext (MQAXC) - 入出力

出口コンテキスト構造体。

Hconn (MQHCONN) - 入力

接続ハンドル。

pXID (MQPTR) - 入出力

トランザクション・ブランチ ID。

Rmid (MQLONG) - 入出力

リソース・マネージャー ID。

Flags (MQLONG) - 入出力

リソース・マネージャー・オプション。

XARetCode (MQLONG) - 入出力

XA 呼び出しからの応答。

C 言語呼び出し

キュー・マネージャーは、以下の変数を論理的に定義します。

```
MQAXP  ExitParms; /* Exit parameter structure */  
MQAXC  ExitContext; /* Exit context structure */  
MQPTR  pXID; /* Transaction branch ID */  
MQLONG Rmid; /* Resource manager identifier */  
MQLONG Flags; /* Resource manager options */  
MQLONG XARetCode; /* Response from XA call */
```

その後、キュー・マネージャーは以下のように出口を論理的に呼び出します。

```
AX_REG_EXIT (&ExitParms, &ExitContext, &pXID, &Rmid, &Flags, &XARetCode);
```

出口は以下の C 関数プロトタイプと一致していなければなりません。

```
typedef void MQENTRY AX_REG_EXIT (  
    PMQAXP  pExitParms, /* Address of exit parameter structure */  
    PMQAXC  pExitContext, /* Address of exit context structure */  
    PMQPTR  ppXID, /* Address of transaction branch ID */  
    PMQLONG pRmid, /* Address of resource manager identifier */
```

```
PMQLONG pFlags,      /* Address of resource manager options*/
PMQLONG pXARetCode); /* Address of response from XA call */
```

ax_unreg - AX_UNREG_EXIT

AX_UNREG_EXIT は、ax_unreg 処理の前と後に実行する ax_unreg 出口関数を提供します。関数 ID MQXF_AXUNREG に出口理由 MQXR_BEFORE および MQXR_AFTER を指定して、ax_unreg の呼び出し前および呼び出し後の出口関数を登録します。

この関数へのインターフェースは、以下のとおりです。

```
AX_UNREG_EXIT (&ExitParms, &ExitContext, &Rmid, &Flags, &XARetCode);
```

パラメーターは、以下のとおりです。

ExitParms (MQAXP) - 入出力

出口パラメーター構造体。

ExitContext (MQAXC) - 入出力

出口コンテキスト構造体。

Rmid (MQLONG) - 入出力

リソース・マネージャー ID。

Flags (MQLONG) - 入出力

リソース・マネージャー・オプション。

XARetCode (MQLONG) - 入出力

XA 呼び出しからの応答。

C 言語呼び出し

キュー・マネージャーは、以下の変数を論理的に定義します。

```
MQAXP ExitParms; /* Exit parameter structure */
MQAXC ExitContext; /* Exit context structure */
MQLONG Rmid; /* Resource manager identifier */
MQLONG Flags; /* Resource manager options*/
MQLONG XARetCode; /* Response from XA call */
```

その後、キュー・マネージャーは以下のように出口を論理的に呼び出します。

```
AX_UNREG_EXIT (&ExitParms, &ExitContext, &Rmid, &Flags, &XARetCode);
```

出口は以下の C 関数プロトタイプと一致していなければなりません。

```
typedef void MQENTRY AX_UNREG_EXIT (
    PMQAXP pExitParms, /* Address of exit parameter structure */
    PMQAXC pExitContext, /* Address of exit context structure */
    PMQLONG pRmid, /* Address of resource manager identifier */
    PMQLONG pFlags, /* Address of resource manager options*/
    PMQLONG pXARetCode); /* Address of response from XA call */
```

出口関数の起動に関する一般情報

このトピックでは、出口の作成、特にエラーおよび予期しないイベントの処理に関する出口を作成するときに役立つ一般的な指針を記載します。

出口の障害

出口関数が、同期点の範囲外の破壊的な MQGET 呼び出しの後、かつアプリケーションにメッセージを渡す前に異常終了した場合、出口ハンドラーは障害から復旧してアプリケーションに制御を戻すことができます。

この場合、メッセージは失われる可能性があります。これは、キューからメッセージを受け取った直後にアプリケーションに障害が起こった場合と似ています。

MQGET 呼び出しは、MQCC_FAILED および MQRC_API_EXIT_ERROR を出して完了することがあります。

前 API 呼び出し出口関数が異常終了した場合、出口ハンドラーは API 呼び出しを処理せずに障害から復旧して、制御をアプリケーションに渡すことができます。この場合、出口関数は所有するすべてのリソースを復旧する必要があります。

チェーニングされた出口が使用されている場合、正常に実行された前 API 呼び出し出口に対する後 API 呼び出し出口そのものを実行することができます。API 呼び出しは、MQCC_FAILED および MQRC_API_EXIT_ERROR を出して失敗することがあります。

出口関数のエラー処理の例

以下の図は、エラーが生じる可能性のあるポイント (eN) を示しています。これは出口の動作を示すだけの例であり、後続の表と共に参照してください。この例では、チェーニングされた出口での動作を例示するために、2 つの出口関数が各 API 呼び出しの前と後の両方に起動されています。

Application	ErrPt	Exit function	API call
-----	-----	-----	-----
Start			
MQCONN	-->		
	e1	MQ_INIT_EXIT	
	e2	before MQ_CONNX_EXIT 1	
	e3	before MQ_CONNX_EXIT 2	
	e4		--> MQCONN
	e5	after MQ_CONNX_EXIT 2	
	e6	after MQ_CONNX_EXIT 1	
	e7		
MQOPEN	<-- -->		
	e8	before MQ_OPEN_EXIT 1	
	e9	before MQ_OPEN_EXIT 2	
	e10		--> MQOPEN
	e11	after MQ_OPEN_EXIT 2	
	e12	after MQ_OPEN_EXIT 1	
	e12		
MQPUT	<-- -->		
	e13	before MQ_PUT_EXIT 1	
	e14	before MQ_PUT_EXIT 2	
	e15		--> MQPUT
	e16	after MQ_PUT_EXIT 2	
	e17	after MQ_PUT_EXIT 1	
	e17		
MQCLOSE	<-- -->		
	e18	before MQ_CLOSE_EXIT 1	
	e19	before MQ_CLOSE_EXIT 2	
	e20		--> MQCLOSE
	e21	after MQ_CLOSE_EXIT 2	
	e22	after MQ_CLOSE_EXIT 1	
	e22		
MQDISC	<-- -->		
	e23	before MQ_DISC_EXIT 1	
	e24	before MQ_DISC_EXIT 2	
	e25		--> MQDISC
	e25	after MQ_DISC_EXIT 2	

```

e26      after  MQ_DISC_EXIT  1
e27
      <--
end

```

以下の表は、各エラー・ポイントで行う必要のある処置をリストしています。他のすべてのエラー・ポイントにもここで示されている規則を適用できるので、そのサブセットだけが示されています。これらの処置が、それぞれの場合の意図された動作を指定しています。

表 873. API 出口エラーおよび適切な処置		
Err Pt	説明	Actions
e1	環境のセットアップ中にエラーが発生しました。	1. 必要に応じて環境のセットアップを元に戻します 2. 出口関数を実行しないでください 3. MQCONN は MQCC_FAILED、MQRC_API_EXIT_LOAD_ERROR を出して失敗します
e2	MQ_INIT_EXIT 関数は以下を出して完了します。 • MQXCC_FAILED • MQXCC_*	• MQXCC_FAILED の場合: 1. 環境をクリーンアップします 2. MQCONN は MQCC_FAILED、MQRC_API_EXIT_INIT_ERROR を出して失敗します • MQXCC_* の場合 1. MQXCC_* および MQXR2_* の値に応じて処置を行います¹ 2. 環境をクリーンアップします
e3	前 MQ_CONNX_EXIT 1 関数は以下を出して完了します。 • MQXCC_FAILED • MQXCC_*	• MQXCC_FAILED の場合: 1. MQ_TERM_EXIT 関数を実行します 2. 環境をクリーンアップします 3. MQCONN 呼び出しは MQCC_FAILED、MQRC_API_EXIT_ERROR を出して失敗します • MQXCC_* の場合 1. MQXCC_* および MQXR2_* の値に応じて処置を行います¹ 2. 必要であれば MQ_TERM_EXIT 関数を実行します 3. 必要であれば環境をクリーンアップします

表 873. API 出口エラーおよび適切な処置 (続き)

Err Pt	説明	Actions
e4	前 MQ_CONNX_EXIT 2 関数は以下を出して完了します。 • MQXCC_FAILED • MQXCC_*	• MQXCC_FAILED の場合: 1. 後 MQ_CONNX_EXIT 1 関数を実行します 2. MQ_TERM_EXIT 関数を実行します 3. 環境をクリーンアップします 4. MQCONN 呼び出しは MQCC_FAILED、MQRC_API_EXIT_ERROR を出して失敗します • MQXCC_* の場合 1. MQXCC_* および MQXR2_* の値に応じて処置を行います¹ 2. 出口が抑制されていない場合、後 MQ_CONNX_EXIT 1 関数を実行します 3. 必要であれば MQ_TERM_EXIT 関数を実行します 4. 必要であれば環境をクリーンアップします
e5	MQCONN 呼び出しが失敗します。	1. MQCONN CompCode および Reason を渡します 2. 前 MQ_CONNX_EXIT 2 が成功して出口が抑制されていない場合、後 MQ_CONNX_EXIT 2 関数を実行します 3. 前 MQ_CONNX_EXIT 1 が成功して出口が抑制されていない場合、後 MQ_CONNX_EXIT 1 関数を実行します 4. MQ_TERM_EXIT 関数を実行します 5. 環境をクリーンアップします
e6	後 MQ_CONNX_EXIT 2 関数は以下を出して完了します。 • MQXCC_FAILED • MQXCC_*	• MQXCC_FAILED の場合: 1. 後 MQ_CONNX_EXIT 1 関数を実行します 2. MQCONN 呼び出しが MQCC_FAILED、MQRC_API_EXIT_ERROR を出して完了します • MQXCC_* の場合 1. MQXCC_* および MQXR2_* の値に応じて処置を行います¹ 2. 必要であれば、後 MQ_CONNX_EXIT 1 関数を実行します
e7	後 MQ_CONNX_EXIT 1 関数は以下を出して完了します。 • MQXCC_FAILED • MQXCC_*	• MQXCC_FAILED の場合、MQCONN 呼び出しが MQCC_FAILED、MQRC_API_EXIT_ERROR を出して完了します • MQXCC_* の場合、MQXCC_* および MQXR2_* の値に応じて処置を行います¹
e8	前 MQ_OPEN_EXIT 1 関数は以下を出して完了します。 • MQXCC_FAILED • MQXCC_*	• MQXCC_FAILED の場合、MQOPEN 呼び出しが MQCC_FAILED、MQRC_API_EXIT_ERROR を出して完了します • MQXCC_* の場合、MQXCC_* および MQXR2_* の値に応じて処置を行います¹

表 873. API 出口エラーおよび適切な処置 (続き)		
Err Pt	説明	Actions
e9	前 MQ_OPEN_EXIT 2 関数は以下を出して完了します。 • MQXCC_FAILED • MQXCC_*	• MQXCC_FAILED の場合: 1. 後 MQ_OPEN_EXIT 1 関数を実行します 2. MQOPEN 呼び出しが MQCC_FAILED、MQRC_API_EXIT_ERROR を出して完了します • MQXCC_* の場合、MQXCC_* および MQXR2_* の値に応じて処置を行います¹
e10	MQOPEN 呼び出しが失敗します	1. MQOPEN CompCode および Reason を渡します 2. 出口が抑制されていない場合、後 MQ_OPEN_EXIT 2 関数を実行します 3. 出口が抑制されておらず、チェーニングされた出口も抑制されていない場合には、後 MQ_OPEN_EXIT 1 関数を実行します
e11	後 MQ_OPEN_EXIT 2 関数は以下を出して完了します。 • MQXCC_FAILED • MQXCC_*	• MQXCC_FAILED の場合: 1. 後 MQ_OPEN_EXIT 1 関数を実行します 2. MQOPEN 呼び出しが MQCC_FAILED、MQRC_API_EXIT_ERROR を出して完了します • MQXCC_* の場合 1. MQXCC_* および MQXR2_* の値に応じて処置を行います¹ 2. 出口が抑制されていない場合、後 MQ_OPEN_EXIT 1 関数を実行します
e25	後 MQ_DISC_EXIT 2 関数は以下を出して完了します。 • MQXCC_FAILED • MQXCC_*	• MQXCC_FAILED の場合: 1. 後 MQ_DISC_EXIT 1 関数を実行します 2. MQ_TERM_EXIT 関数を実行します 3. 出口の実行環境をクリーンアップします 4. MQDISC 呼び出しが MQCC_FAILED、MQRC_API_EXIT_ERROR を出して完了します • MQXCC_* の場合 1. MQXCC_* および MQXR2_* の値に応じて処置を行います¹ 2. MQ_TERM_EXIT 関数を実行します 3. 出口の実行環境をクリーンアップします

注:

1. MQXCC_* および MQXR2_* の値と、それに対応する処置については、『[キュー・マネージャーが出口関数を処理する方法処理](#)』で定義されています。

ExitResponse フィールドの誤った設定

このトピックでは、ExitResponse フィールドにサポートされていない値が設定されている場合にどうなるかについて情報を提供します。

ExitResponse フィールドがサポートされている以外の値に設定されている場合、以下の処置が適用されます。

- 前 MQCONN または MQDISC API 出口関数の場合

- ExitResponse2 値は無視されます。
- 出口チェーンに他の前 出口関数があっても、起動されません。API 呼び出しそのものが発行されません。
- 正常に呼び出された前 出口に対して、後 出口が反対の順序で呼び出されます。
- 登録されている場合、正常に起動されたチェーン内の前 MQCONN または MQDISC 出口関数に対して終了出口関数が実行されて、それらの出口関数の後がクリーンアップされます。
- MQCONN または MQDISC 呼び出しは、MQRC_API_EXIT_ERROR を出して失敗します。
- MQCONN または MQDISC 以外の前 IBM MQ API 出口関数の場合
 - ExitResponse2 値は無視されます。
 - 出口チェーンに他の前 または後 データ変換関数があっても、起動されません。
 - 正常に呼び出された前 出口に対して、後 出口が反対の順序で呼び出されます。
 - IBM MQ API 呼び出しそのものが発行されません。
 - IBM MQ API 呼び出しは、MQRC_API_EXIT_ERROR を出して失敗します。
- 後 MQCONN または MQDISC API 出口関数の場合
 - ExitResponse2 値は無視されます。
 - API 呼び出しの前に正常に呼び出された残りの出口関数は、反対の順序で呼び出されます。
 - 正常に起動されたチェーン内の前 または後 MQCONN または MQDISC 出口関数に対して、終了出口関数が登録されていれば実行され、出口の後がクリーンアップされます。
 - より重大度の高い MQCC_WARNING の CompCode と、出口から戻された CompCode は、アプリケーションに戻されます。
 - MQRC_API_EXIT_ERROR の Reason は、アプリケーションに戻されます。
 - IBM MQ API 呼び出しは、正常に発行されます。
- MQCONN または MQDISC 以外の後 IBM MQ API 呼び出し出口関数の場合
 - ExitResponse2 値は無視されます。
 - API 呼び出しの前に正常に呼び出された残りの出口関数は、反対の順序で呼び出されます。
 - より重大度の高い MQCC_WARNING の CompCode と、出口から戻された CompCode は、アプリケーションに戻されます。
 - MQRC_API_EXIT_ERROR の Reason は、アプリケーションに戻されます。
 - IBM MQ API 呼び出しは、正常に発行されます。
- 取得出口関数での前 データ変換の場合
 - ExitResponse2 値は無視されます。
 - API 呼び出しの前に正常に呼び出された残りの出口関数は、反対の順序で呼び出されます。
 - メッセージは変換されず、変換されていないメッセージがアプリケーションに戻されます。
 - より重大度の高い MQCC_WARNING の CompCode と、出口から戻された CompCode は、アプリケーションに戻されます。
 - MQRC_API_EXIT_ERROR の Reason は、アプリケーションに戻されます。
 - IBM MQ API 呼び出しは、正常に発行されます。

注：出口でエラーが発生しているので、MQRC_NOT_CONVERTED を戻すよりも MQRC_API_EXIT_ERROR を戻すほうが適しています。

出口関数が ExitResponse2 フィールドをサポートされていない値に設定した場合、代わりに MQXR2_DEFAULT_CONTINUATION の値が推定されます。

インストール可能サービス・インターフェースの参照情報

このトピック集では、インストール可能サービスの参照情報を記載します。

関数とデータ・タイプが、各サービス・タイプのグループ内でアルファベット順にリストされています。

関連概念

- ALW** [UNIX、Linux、および Windows 用のインストール可能サービスとコンポーネント](#)
- IBM i** [IBM i 用のインストール可能サービスとコンポーネント](#)
- IBM i** [IBM i 用インストール可能サービス・インターフェースの参照情報](#)

関連タスク

[キュー・マネージャーの機能の拡張](#)

- ALW** [インストール可能サービスの構成](#)

関数の表示方法

インストール可能なサービス関数の表記方法。

各関数には、関数 ID などの記述があります (MQZEP の場合)。

表示されているパラメーターは、指定順になっていて、すべて指定する必要があります。

各パラメーター名の後に、そのデータ・タイプを示しています。これらは、[177 ページの『基本データ・タイプ』](#)で説明している基本データ・タイプです。

パラメーターの説明の後に、C 言語による呼び出しも示します。

MQZ_AUTHENTICATE_USER - ユーザーの認証

この関数は、MQZAS_VERSION_5 許可サービス・コンポーネントによって提供されています。この関数は、ユーザーを認証したり、アイデンティティ・コンテキスト・フィールドを設定したりできるように、キュー・マネージャーによって呼び出されます。この関数は、IBM MQ のユーザー・アプリケーション・コンテキストが設定されると呼び出されます。

アプリケーションのコンテキストが確立されるのは、接続呼び出し時のうち、アプリケーションのユーザー・コンテキストが初期化されたときと、アプリケーションのユーザー・コンテキストが変更されたときです。アプリケーションのユーザー・コンテキスト情報は、接続呼び出しが実行されるたびに *IdentityContext* フィールドに再取得されます。

この関数の関数 ID (MQZEP 用) は、MQZID_AUTHENTICATE_USER です。

構文

`MQZ_AUTHENTICATE_USER ( QMgrName , SecurityParms , ApplicationContext , IdentityContext , CorrelationPtr , ComponentData , Continuation , CompCode , Reason )`

パラメーター

QMgrName

タイプ: MQCHAR48 - 入力

キュー・マネージャー名。コンポーネントを呼び出すキュー・マネージャーの名前。この名前は、パラメーターがすべて埋まるまでブランクが埋め込まれます。名前の最後をヌル文字にすることはできません。

キュー・マネージャー名は、情報としてコンポーネントに渡されます。許可サービス・インターフェースでは、コンポーネントは定義されている方法でこの情報を使用する必要はありません。

SecurityParms

タイプ: MQCSP - 入力

セキュリティ・パラメーター。ユーザー ID、パスワード、および認証タイプに関するデータ。MQCSP 構造体の *AuthenticationType* 属性が、MQCSP_AUTH_USER_ID_AND_PWD と指定されている場合は、ユーザー ID とパスワードの両方が *IdentityContext* (MQZIC) パラメーター内の対応するフィールドと比較され、一致するかどうか判別されます。詳細については、[286 ページの『MQCSP - セキュリティ・パラメーター』](#)を参照してください。

MQCONN MQI 呼び出しの間は、このパラメーターは、ヌルまたはデフォルト値を含んでいます。

ApplicationContext

タイプ: MQZAC - 入力

アプリケーション・コンテキスト。呼び出しアプリケーションに関連するデータ。詳細は、[MQZAC - アプリケーション・コンテキスト](#)を参照してください。

MQCONN または MQCONNX MQI 呼び出しの際は常に、MQZAC 構造体内のユーザー・コンテキスト情報は再度取得されます。

IdentityContext

タイプ: MQZIC - 入出力

ID コンテキスト。ユーザー認証関数への入力時に、これは現在の ID コンテキストを示します。ユーザー認証関数はこれを変更でき、この場合はキュー・マネージャーは新しい ID コンテキストを採用します。MQZIC 構造体の詳細については、[MQZIC - ID コンテキスト](#)を参照してください。

CorrelationPtr

タイプ: MQPTR - 出力

相関ポインター。相関データのアドレスを指定します。このポインターは、後に他の OAM 呼び出しに渡されます。

ComponentData

タイプ: MQBYTE x ComponentDataLength - 入出力

コンポーネント・データ。このデータは、この特定のコンポーネントのためにキュー・マネージャーによって保持されます。このコンポーネントによって提供される関数のいずれかによってそのデータに変更が加えられると、その変更は保存され、次回このコンポーネントの関数の 1 つが呼び出されたときに提供されます。

このデータ域の長さは、キュー・マネージャーにより MQZ_INIT_AUTHORITY 呼び出しの ComponentDataLength パラメーターに入れられて渡されます。

Continuation

タイプ: MQLONG - 出力

継続フラグ。以下の値を指定できます。

MQZCI_DEFAULT

他のコンポーネントに依存する継続。

MQZCI_STOP

次のコンポーネントに継続しない。

CompCode

タイプ: MQLONG - 出力

完了コード。値は、次のいずれかでなければなりません。

MQCC_OK

正常終了。

MQCC_FAILED

呼び出し失敗。

理由

タイプ: MQLONG - 出力

CompCode を限定する理由コード。

CompCode が MQCC_OK の場合、次のようになります。

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_FAILED の場合、次のようになります。

MQRC_SERVICE_ERROR

(2289, X'8F1') サービスのアクセスで、予期しないエラーが発生しました。

これらの理由コードについて詳しくは、[メッセージおよび理由コード](#)を参照してください。

C 言語での呼び出し

```
MQZ_AUTHENTICATE_USER (QMgrName, SecurityParms, ApplicationContext,  
                        IdentityContext, &CorrelationPtr, ComponentData,  
                        &Continuation, &CompCode, &Reason);
```

サービスに渡されるパラメーターを次のように宣言します。

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQCSP     SecurityParms;      /* Security parameters */  
MQZAC     ApplicationContext; /* Application context */  
MQZIC     IdentityContext;    /* Identity context */  
MQPTR     CorrelationPtr;     /* Correlation pointer */  
MQBYTE    ComponentData[n];  /* Component data */  
MQLONG    Continuation;      /* Continuation indicator set by  
                             component */  
MQLONG    CompCode;           /* Completion code */  
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

MQZ_CHECK_AUTHORITY - 権限の検査

この関数は、MQZAS_VERSION_1 許可サービス・コンポーネントにより提供され、キュー・マネージャーにより開始されて、指定されたオブジェクトに特定のアクション (1 つまたは複数) を実行する権限をエンティティーが持っているどうかを検査します。

この関数の関数 ID (MQZEP 用) は、MQZID_CHECK_AUTHORITY です。

構文

```
MQZ_CHECK_AUTHORITY( QMgrName , EntityName , EntityType , ObjectName ,  
                     ObjectType , Authority , ComponentData , Continuation , CompCode , Reason )
```

パラメーター

QMgrName

タイプ: MQCHAR48 - 入力

キュー・マネージャー名。コンポーネントを呼び出すキュー・マネージャーの名前。この名前は、パラメーターがすべて埋まるまでブランクが埋め込まれます。名前の最後をヌル文字にすることはできません。

キュー・マネージャー名は、情報としてコンポーネントに渡されます。許可サービス・インターフェースでは、コンポーネントは定義されている方法でこの情報を使用する必要はありません。

EntityName

タイプ: MQCHAR12 - 入力

エンティティー名。オブジェクトに対する許可を検査するエンティティーの名前。ストリングの長さは最大で 12 文字です。12 文字未満の場合、その名前の右側がブランクで埋められます。名前の最後をヌル文字にすることはできません。

このエンティティーは、その下にあるセキュリティー・サービスに既知である必要はありません。既知でない場合、特殊な **nobody (なし)** グループ (すべてのエンティティーが所属していると想定される) の許可が検査に使用されます。すべてブランクの名前は有効で、このように使用できます。

EntityType

タイプ: MQLONG - 入力

エンティティー・タイプ。EntityName によって指定されるエンティティーのタイプ。値は、次のいずれかでなければなりません。

MQZAET_PRINCIPAL

プリンシパル。

MQZAET_GROUP

グループ。

ObjectName

タイプ: MQCHAR48 - 入力

オブジェクト名 アクセスが必要なオブジェクトの名前。 スtringの長さは最大で 48 文字です。 48 文字未満の場合、その名前の右側がブランクで埋められます。 名前の最後をヌル文字にすることはできません。

ObjectType が MQOT_Q_MGR の場合、この名前は *QMgrName* と同じになります。

ObjectType

タイプ: MQLONG - 入力

オブジェクト・タイプ *ObjectName* によって指定されるエンティティのタイプ。 値は、次のいずれかでなければなりません。

MQOT_AUTH_INFO

認証情報

MQOT_CHANNEL

チャンネル。

MQOT_CLNTCONN_CHANNEL

クライアント接続チャンネル。

MQOT_LISTENER

リスナー

MQOT_NAMELIST

名前リスト。

MQOT_PROCESS

プロセス定義。

MQOT_Q

キュー。

MQOT_Q_MGR

キュー・マネージャー。

MQOT_SERVICE

サービス

権限

タイプ: MQLONG - 入力

検査される権限。 1つの許可を検査する場合、このフィールドは該当する許可操作と同じです (MQZAO_* 定数)。 複数の許可を検査する場合、対応する MQZAO_* 定数とビットごとに OR 演算します。

MQI 呼び出しの使用に次の許可が適用されます。

MQZAO_CONNECT

MQCONN 呼び出しを使用する機能。

MQZAO_BROWSE

ブラウズ・オプションを使用して、MQGET 呼び出しを使用する機能。

これにより、MQGMO_BROWSE_FIRST、MQGMO_BROWSE_MSG_UNDER_CURSOR、または MQGMO_BROWSE_NEXT オプションを MQGET 呼び出しで指定できます。

MQZAO_INPUT

プリンシパル。 入力オプションを使用して、MQGET 呼び出しを使用する機能。

これにより、MQOO_INPUT_SHARED、MQOO_INPUT_EXCLUSIVE、または MQOO_INPUT_AS_Q_DEF オプションを MQOPEN 呼び出しで指定できます。

MQZAO_OUTPUT

MQPUT 呼び出しを使用する機能。

これにより、MQOO_OUTPUT オプションを MQOPEN 呼び出しで指定できます。

MQZAO_INQUIRE

MQINQ 呼び出しを使用する機能。

これにより、MQOO_INQUIRE オプションを MQOPEN 呼び出しで指定できます。

MQZAO_SET

MQSET 呼び出しを使用する機能。

これにより、MQOO_SET オプションを MQOPEN 呼び出しで指定できます。

MQZAO_PASS_IDENTITY_CONTEXT

識別コンテキストを渡す機能。

これにより、MQOO_PASS_IDENTITY_CONTEXT オプションを MQOPEN 呼び出しで指定でき、MQPMO_PASS_IDENTITY_CONTEXT オプションを MQPUT および MQPUT1 呼び出しで指定できます。

MQZAO_PASS_ALL_CONTEXT

すべてのコンテキストを渡す機能。

これにより、MQOO_PASS_ALL_CONTEXT オプションを MQOPEN 呼び出しで指定でき、MQPMO_PASS_ALL_CONTEXT オプションを MQPUT および MQPUT1 呼び出しで指定できます。

MQZAO_SET_IDENTITY_CONTEXT

識別コンテキストを設定する機能。

これにより、MQOO_SET_IDENTITY_CONTEXT オプションを MQOPEN 呼び出しで指定でき、MQPMO_SET_IDENTITY_CONTEXT オプションを MQPUT および MQPUT1 呼び出しで指定できます。

MQZAO_SET_ALL_CONTEXT

すべてのコンテキストを設定する機能。

これにより、MQOO_SET_ALL_CONTEXT オプションを MQOPEN 呼び出しで指定でき、MQPMO_SET_ALL_CONTEXT オプションを MQPUT および MQPUT1 呼び出しで指定できます。

MQZAO_ALTERNATE_USER_AUTHORITY

代替ユーザー権限を使用する機能。

これにより、MQOO_ALTERNATE_USER_AUTHORITY オプションを MQOPEN 呼び出しで指定でき、MQPMO_ALTERNATE_USER_AUTHORITY オプションを MQPUT1 呼び出しで指定できます。

MQZAO_ALL_MQI

すべての MQI 許可。

これにより、すべての権限が有効になります。

次の許可がキュー・マネージャーの管理に適用されます。

MQZAO_CREATE

指定されたタイプのオブジェクトを作成する機能。

MQZAO_DELETE

指定されたオブジェクトを削除する機能。

MQZAO_DISPLAY

指定されたオブジェクトの属性を表示する機能。

MQZAO_CHANGE

指定されたオブジェクトの属性を変更する機能。

MQZAO_CLEAR

指定されたキューからすべてのメッセージを削除する機能。

MQZAO_AUTHORIZE

その他のユーザーに指定されたオブジェクトを許可する機能。

MQZAO_CONTROL

リスナー、サービス、または非クライアント・チャネル・オブジェクトを開始または停止する機能と、非クライアント・チャネル・オブジェクトに ping する機能。

MQZAO_CONTROL_EXTENDED

シーケンス番号をリセット、または非クライアント・チャネル・オブジェクト上の未確定メッセージを解決する機能。

MQZAO_ALL_ADMIN

識別コンテキストを設定する機能。

MQZAO_CREATE 以外のすべての管理許可。

次の許可が、MQI の両方の使用とキュー・マネージャーの管理に適用されます。

MQZAO_ALL

MQZAO_CREATE 以外のすべての許可。

MQZAO_NONE

許可なし。

ComponentData

タイプ: MQBYTE x ComponentDataLength - 入出力

コンポーネント・データ。このデータは、この特定のコンポーネントのためにキュー・マネージャーで保持されます。このコンポーネントに用意されているいずれかの関数で変更された内容は保持され、これらのコンポーネントのいずれかの関数が次回に呼び出されると提示されます。

このデータ域の長さは、キュー・マネージャーにより MQZ_INIT_AUTHORITY 呼び出しの **ComponentDataLength** パラメーターに入れられて渡されます。

Continuation

タイプ: MQLONG - 出力

コンポーネントによって設定される継続標識。指定可能な値は、次のとおりです。

MQZCI_DEFAULT

キュー・マネージャーに依存する継続。

MQZ_CHECK_AUTHORITY の場合、MQZCI_STOP と同じ効果があります。

MQZCI_CONTINUE

次のコンポーネントに継続。

MQZCI_STOP

次のコンポーネントに継続しない。

コンポーネントの呼び出しに失敗し (つまり、*CompCode* が MQCC_FAILED を返す)、*Continuation* パラメーターが MQZCI_DEFAULT または MQZCI_CONTINUE の場合、キュー・マネージャーは他のコンポーネントの呼び出しを続けます (存在する場合)。

呼び出しに成功すると (つまり、*CompCode* が MQCC_OK を返す)、*Continuation* の設定に関係なく、コンポーネントの呼び出しはそれ以上行われなくなります。

呼び出しに失敗し、*Continuation* パラメーターが MQZCI_STOP の場合、その他のコンポーネントの呼び出しは行われず、エラーがキュー・マネージャーに返されます。コンポーネントは以前の呼び出しを認識していないため、呼び出しの前に *Continuation* パラメーターが常に MQZCI_DEFAULT に設定されます。

CompCode

タイプ: MQLONG - 出力

完了コード。値は、次のいずれかでなければなりません。

MQCC_OK

正常終了。

MQCC_FAILED

呼び出し失敗。

理由

タイプ: MQLONG - 出力

CompCode を限定する理由コード。

CompCode が MQCC_OK の場合、次のようになります。

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_FAILED の場合、次のようになります。

MQRC_NOT_AUTHORIZED

(2035, X'7F3') アクセスは許可されません。

MQRC_SERVICE_ERROR

(2289, X'8F1') サービスのアクセスで、予期しないエラーが発生しました。

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') 基本サービスを使用できません。

これらの理由コードについて詳しくは、[API 完了コードと理由コード](#)を参照してください。

C 言語での呼び出し

```
MQZ_CHECK_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName,  
                     ObjectType, Authority, ComponentData,  
                     &Continuation, &CompCode, &Reason);
```

サービスに渡されるパラメーターは次のように宣言されます。

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQCHAR12  EntityName;         /* Entity name */  
MQLONG    EntityType;         /* Entity type */  
MQCHAR48  ObjectName;         /* Object name */  
MQLONG    ObjectType;         /* Object type */  
MQLONG    Authority;          /* Authority to be checked */  
MQBYTE    ComponentData[n];   /* Component data */  
MQLONG    Continuation;       /* Continuation indicator set by  
                               component */  
MQLONG    CompCode;           /* Completion code */  
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

MQZ_CHECK_AUTHORITY_2 - 権限の検査 (拡張版)

この関数は、MQZAS_VERSION_2 許可サービス・コンポーネントにより提供され、キュー・マネージャーにより開始されて、指定されたオブジェクトに特定のアクション (1 つまたは複数) を実行する権限をエンティティーが持っているどうかを検査します。

この関数の関数 ID (MQZEP 用) は、MQZID_CHECK_AUTHORITY です。

MQZ_CHECK_AUTHORITY_2 は MQZ_CHECK_AUTHORITY と似ていますが、**EntityName** パラメーターが **EntityData** パラメーターで置き換えられています。

構文

`MQZ_CHECK_AUTHORITY_2(QMgrName , EntityData , EntityType , ObjectName ,
ObjectType , Authority , ComponentData , Continuation , CompCode , Reason)`

パラメーター

QMgrName

タイプ: MQCHAR48 - 入力

キュー・マネージャー名。コンポーネントを呼び出すキュー・マネージャーの名前。この名前は、パラメーターがすべて埋まるまでブランクが埋め込まれます。名前の最後をヌル文字にすることはできません。

キュー・マネージャー名は、情報としてコンポーネントに渡されます。許可サービス・インターフェースでは、コンポーネントは定義されている方法でこの情報を使用する必要はありません。

EntityData

タイプ: MQZED - 入力

エンティティ・データ。オブジェクトに対する許可が検査されるエンティティに関連するデータ。詳細は [1682 ページの『MQZED - エンティティ記述子』](#) を参照してください。

このエンティティは、その下にあるセキュリティー・サービスに既知である必要はありません。既知でない場合、特殊な **nobody (なし)** グループ (すべてのエンティティが所属していると想定される) の許可が検査に使用されます。すべてブランクの名前は有効で、このように使用できます。

EntityType

タイプ: MQLONG - 入力

エンティティ・タイプ。 *EntityData* によって指定されるエンティティのタイプ。値は、次のいずれかでなければなりません。

MQZAET_PRINCIPAL

プリンシパル。

MQZAET_GROUP

グループ。

ObjectName

タイプ: MQCHAR48 - 入力

オブジェクト名 アクセスが必要なオブジェクトの名前。ストリングの長さは最大で 48 文字です。48 文字未満の場合、その名前の右側がブランクで埋められます。名前の最後をヌル文字にすることはできません。

ObjectType が MQOT_Q_MGR の場合、この名前は *QMgrName* と同じになります。

ObjectType

タイプ: MQLONG - 入力

オブジェクト・タイプ *ObjectName* によって指定されるエンティティのタイプ。値は、次のいずれかでなければなりません。

MQOT_AUTH_INFO

認証情報

MQOT_CHANNEL

チャネル。

MQOT_CLNTCONN_CHANNEL

クライアント接続チャネル。

MQOT_LISTENER

リスナー

MQOT_NAMELIST

名前リスト。

MQOT_PROCESS

プロセス定義。

MQOT_Q

キュー。

MQOT_Q_MGR

キュー・マネージャー。

MQOT_SERVICE

サービス

MQOT_TOPIC

トピック。

権限

タイプ: MQLONG - 入力

検査される権限。1つの許可を検査する場合、このフィールドは該当する許可操作と同じです (MQZAO_* 定数)。複数の許可を検査する場合、対応する MQZAO_* 定数とビットごとに OR 演算します。

MQI 呼び出しの使用に次の許可が適用されます。

MQZAO_CONNECT

MQCONN 呼び出しを使用する機能。

MQZAO_BROWSE

ブラウズ・オプションを使用して、MQGET 呼び出しを使用する機能。

これにより、MQGMO_BROWSE_FIRST、MQGMO_BROWSE_MSG_UNDER_CURSOR、または MQGMO_BROWSE_NEXT オプションを MQGET 呼び出しで指定できます。

MQZAO_INPUT

プリンシパル。入力オプションを使用して、MQGET 呼び出しを使用する機能。

これにより、MQOO_INPUT_SHARED、MQOO_INPUT_EXCLUSIVE、または MQOO_INPUT_AS_Q_DEF オプションを MQOPEN 呼び出しで指定できます。

MQZAO_OUTPUT

MQPUT 呼び出しを使用する機能。

これにより、MQOO_OUTPUT オプションを MQOPEN 呼び出しで指定できます。

MQZAO_INQUIRE

MQINQ 呼び出しを使用する機能。

これにより、MQOO_INQUIRE オプションを MQOPEN 呼び出しで指定できます。

MQZAO_SET

MQSET 呼び出しを使用する機能。

これにより、MQOO_SET オプションを MQOPEN 呼び出しで指定できます。

MQZAO_PASS_IDENTITY_CONTEXT

識別コンテキストを渡す機能。

これにより、MQOO_PASS_IDENTITY_CONTEXT オプションを MQOPEN 呼び出しで指定でき、MQPMO_PASS_IDENTITY_CONTEXT オプションを MQPUT および MQPUT1 呼び出しで指定できます。

MQZAO_PASS_ALL_CONTEXT

すべてのコンテキストを渡す機能。

これにより、MQOO_PASS_ALL_CONTEXT オプションを MQOPEN 呼び出しで指定でき、MQPMO_PASS_ALL_CONTEXT オプションを MQPUT および MQPUT1 呼び出しで指定できます。

MQZAO_SET_IDENTITY_CONTEXT

識別コンテキストを設定する機能。

これにより、MQOO_SET_IDENTITY_CONTEXT オプションを MQOPEN 呼び出しで指定でき、MQPMO_SET_IDENTITY_CONTEXT オプションを MQPUT および MQPUT1 呼び出しで指定できます。

MQZAO_SET_ALL_CONTEXT

すべてのコンテキストを設定する機能。

これにより、MQOO_SET_ALL_CONTEXT オプションを MQOPEN 呼び出しで指定でき、MQPMO_SET_ALL_CONTEXT オプションを MQPUT および MQPUT1 呼び出しで指定できます。

MQZAO_ALTERNATE_USER_AUTHORITY

代替ユーザー権限を使用する機能。

これにより、MQOO_ALTERNATE_USER_AUTHORITY オプションを MQOPEN 呼び出しで指定でき、MQPMO_ALTERNATE_USER_AUTHORITY オプションを MQPUT1 呼び出しで指定できます。

MQZAO_ALL_MQI

すべての MQI 許可。

これにより、すべての権限が有効になります。

次の許可がキュー・マネージャーの管理に適用されます。

MQZAO_CREATE

指定されたタイプのオブジェクトを作成する機能。

MQZAO_DELETE

指定されたオブジェクトを削除する機能。

MQZAO_DISPLAY

指定されたオブジェクトの属性を表示する機能。

MQZAO_CHANGE

指定されたオブジェクトの属性を変更する機能。

MQZAO_CLEAR

指定されたキューからすべてのメッセージを削除する機能。

MQZAO_AUTHORIZE

その他のユーザーに指定されたオブジェクトを許可する機能。

MQZAO_CONTROL

リスナー、サービス、または非クライアント・チャネル・オブジェクトを開始または停止する機能と、非クライアント・チャネル・オブジェクトに ping する機能。

MQZAO_CONTROL_EXTENDED

シーケンス番号をリセット、または非クライアント・チャネル・オブジェクト上の未確定メッセージを解決する機能。

MQZAO_ALL_ADMIN

識別コンテキストを設定する機能。

MQZAO_CREATE 以外のすべての管理許可。

次の許可が、MQI の両方の使用とキュー・マネージャーの管理に適用されます。

MQZAO_ALL

MQZAO_CREATE 以外のすべての許可。

MQZAO_NONE

許可なし。

ComponentData

タイプ: MQBYTE x ComponentDataLength - 入出力

コンポーネント・データ。このデータは、この特定のコンポーネントのためにキュー・マネージャーで保持されます。このコンポーネントに用意されているいずれかの関数で変更された内容は保持され、これらのコンポーネントのいずれかの関数が次回に呼び出されると提示されます。

このデータ域の長さは、キュー・マネージャーにより MQZ_INIT_AUTHORITY 呼び出しの **ComponentDataLength** パラメーターに入れられて渡されます。

Continuation

タイプ: MQLONG - 出力

コンポーネントによって設定される継続標識。指定可能な値は、次のとおりです。

MQZCI_DEFAULT

キュー・マネージャーに依存する継続。

MQZ_CHECK_AUTHORITY の場合、MQZCI_STOP と同じ効果があります。

MQZCI_CONTINUE

次のコンポーネントに継続。

MQZCI_STOP

次のコンポーネントに継続しない。

CompCode

タイプ: MQLONG - 出力

完了コード。値は、次のいずれかでなければなりません。

MQCC_OK

正常終了。

MQCC_FAILED

呼び出し失敗。

理由

タイプ: MQLONG - 出力

CompCode を限定する理由コード。

CompCode が MQCC_OK の場合、次のようになります。

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_FAILED の場合、次のようになります。

MQRC_NOT_AUTHORIZED

(2035, X'7F3') アクセスは許可されません。

MQRC_SERVICE_ERROR

(2289, X'8F1') サービスのアクセスで、予期しないエラーが発生しました。

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') 基本サービスを使用できません。

これらの理由コードについて詳しくは、[API 完了コードと理由コード](#)を参照してください。

C 言語での呼び出し

```
MQZ_CHECK_AUTHORITY_2 (QMgrName, &EntityData, EntityType,  
                        ObjectName, ObjectType, Authority, ComponentData,  
                        &Continuation, &CompCode, &Reason);
```

サービスに渡されるパラメーターは次のように宣言されます。

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQZED     EntityData;         /* Entity data */  
MQLONG    EntityType;         /* Entity type */  
MQCHAR48  ObjectName;        /* Object name */  
MQLONG    ObjectType;        /* Object type */  
MQLONG    Authority;         /* Authority to be checked */  
MQBYTE    ComponentData[n];  /* Component data */  
MQLONG    Continuation;      /* Continuation indicator set by  
                             component */  
MQLONG    CompCode;          /* Completion code */  
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

MQZ_CHECK_PRIVILEGED - ユーザーが特権ユーザーかどうかの検査

この関数は MQZAS_VERSION_6 許可サービス・コンポーネントによって提供されています。この関数は、指定されたユーザーが特権ユーザーかどうかを判別するためにキュー・マネージャーによって呼び出されます。

この関数の関数 ID (MQZEP 用) は、MQZID_CHECK_PRIVILEGED です。

構文

```
MQZ_CHECK_PRIVILEGED( QMgrName , EntityData , EntityType , ComponentData ,  
                     Continuation , CompCode , Reason )
```

パラメーター

QMgrName

タイプ: MQCHAR48 - 入力

キュー・マネージャー名。コンポーネントを呼び出すキュー・マネージャーの名前。この名前は、パラメーターがすべて埋まるまでブランクが埋め込まれます。名前の最後をヌル文字にすることはできません。

キュー・マネージャー名は、情報としてコンポーネントに渡されます。許可サービス・インターフェースでは、コンポーネントは定義されている方法でこの情報を使用する必要はありません。

EntityData

タイプ: MQZED - 入力

エンティティ・データ。検査されるエンティティに関連しているデータ。詳しくは、[1682 ページ](#)の『MQZED - エンティティ記述子』を参照してください。

EntityType

タイプ: MQLONG - 入力

エンティティ・タイプ。EntityData によって指定されるエンティティのタイプ。値は、次のいずれかでなければなりません。

MQZAET_PRINCIPAL

プリンシパル。

MQZAET_GROUP

グループ。

ComponentData

タイプ: MQBYTEExComponentDataLength - 入出力

コンポーネント・データ。このデータは、この特定のコンポーネントのためにキュー・マネージャーで保持されます。このコンポーネントに用意されているいずれかの関数で変更された内容は保持され、これらのコンポーネントのいずれかの関数が次回に呼び出されると提示されます。

このデータ域の長さは、キュー・マネージャーにより MQZ_INIT_AUTHORITY 呼び出しの **ComponentDataLength** パラメーターに入れられて渡されます。

Continuation

タイプ: MQLONG - 出力

コンポーネントによって設定される継続標識。指定可能な値は、次のとおりです。

MQZCI_DEFAULT

キュー・マネージャーに依存する継続。

MQZ_CHECK_AUTHORITY の場合、MQZCI_STOP と同じ効果があります。

MQZCI_CONTINUE

次のコンポーネントに継続。

MQZCI_STOP

次のコンポーネントに継続しない。

コンポーネントの呼び出しに失敗し (つまり、CompCode が MQCC_FAILED を返す)、Continuation パラメーターが MQZCI_DEFAULT または MQZCI_CONTINUE の場合、キュー・マネージャーは他のコンポーネントの呼び出しを続けます (存在する場合)。

呼び出しに成功すると (つまり、CompCode が MQCC_OK を返す)、Continuation の設定に関係なく、コンポーネントの呼び出しはそれ以上行われなくなります。

呼び出しに失敗し、Continuation パラメーターが MQZCI_STOP の場合、その他のコンポーネントの呼び出しは行われず、エラーがキュー・マネージャーに返されます。コンポーネントは以前の呼び出しを認識していないため、呼び出しの前に Continuation パラメーターが常に MQZCI_DEFAULT に設定されます。

CompCode

タイプ: MQLONG - 出力

完了コード。値は、次のいずれかでなければなりません。

MQCC_OK

正常終了。

MQCC_FAILED

呼び出し失敗。

理由

タイプ: MQLONG - 出力

CompCode を限定する理由コード。

CompCode が MQCC_OK の場合、次のようになります。

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_FAILED の場合、次のようになります。

MQRC_NOT_PRIVILEGED

(2584, X'A18') このユーザーは特権ユーザー ID ではありません。

MQRC_UNKNOWN_ENTITY

(2292, X'8F4') サービスに対して不明なエンティティです。

MQRC_SERVICE_ERROR

(2289, X'8F1') サービスのアクセスで、予期しないエラーが発生しました。

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') 基本サービスを使用できません。

これらの理由コードについて詳しくは、[API 完了コードと理由コード](#)を参照してください。

C 言語での呼び出し

```
MQZ_CHECK_PRIVILEGED (QMgrName, &EntityData, EntityType,  
                      ComponentData, &Continuation,  
                      &CompCode, &Reason);
```

サービスに渡されるパラメーターは次のように宣言されます。

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQZED     EntityData;         /* Entity name */  
MQLONG    EntityType;         /* Entity type */  
MQBYTE    ComponentData[n];   /* Component data */  
MQLONG    Continuation;       /* Continuation indicator set by  
                               component */  
MQLONG    CompCode;           /* Completion code */  
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

MQZ_COPY_ALL_AUTHORITY - 全権限のコピー

この関数は、許可サービス・コンポーネントに用意されています。キュー・マネージャーはこの関数を開始し、参照オブジェクトに現在適用されているすべての許可を別のオブジェクトにコピーします。

この関数の関数 ID (MQZEP 用) は、MQZID_COPY_ALL_AUTHORITY です。

構文

```
MQZ_COPY_ALL_AUTHORITY( QMgrName , RefObjectName , ObjectName , ObjectType ,  
                        ComponentData , Continuation , CompCode , Reason )
```


パラメーター

QMgrName

タイプ: MQCHAR48 - 入力

キュー・マネージャー名。コンポーネントを呼び出すキュー・マネージャーの名前。この名前は、パラメーターがすべて埋まるまで空白が埋め込まれます。名前の最後をヌル文字にすることはできません。

キュー・マネージャー名は、情報としてコンポーネントに渡されます。許可サービス・インターフェースでは、コンポーネントは定義されている方法でこの情報を使用する必要はありません。

RefObjectName

タイプ: MQCHAR48 - 入力

参照オブジェクト名。参照オブジェクトの名前で、その許可がコピーされます。ストリングの長さは最大で 48 文字です。48 文字未満の場合、その名前の右側が空白で埋められます。名前の最後をヌル文字にすることはできません。

ObjectName

タイプ: MQCHAR48 - 入力

オブジェクト名 オブジェクトの名前で、それに対するアクセスが設定されます。ストリングの長さは最大で 48 文字です。48 文字未満の場合、その名前の右側が空白で埋められます。名前の最後をヌル文字にすることはできません。

ObjectType

タイプ: MQLONG - 入力

オブジェクト・タイプ *RefObjectName* および *ObjectName* に指定されたエンティティのタイプ。値は、次のいずれかでなければなりません。

MQOT_AUTH_INFO

認証情報

MQOT_CHANNEL

チャネル。

MQOT_CLNTCONN_CHANNEL

クライアント接続チャネル。

MQOT_LISTENER

リスナー

MQOT_NAMELIST

名前リスト。

MQOT_PROCESS

プロセス定義。

MQOT_Q

キュー。

MQOT_Q_MGR

キュー・マネージャー。

MQOT_SERVICE

サービス

MQOT_TOPIC

トピック。

ComponentData

タイプ: MQBYTEExComponentDataLength - 入出力

コンポーネント・データ。このデータは、この特定のコンポーネントのためにキュー・マネージャーで保持されます。このコンポーネントに用意されているいずれかの関数で変更された内容は保持され、これらのコンポーネントのいずれかの関数が次回に呼び出されると提示されます。

このデータ域の長さは、キュー・マネージャーにより MQZ_INIT_AUTHORITY 呼び出しの ComponentDataLength パラメーターに入れられて渡されます。

Continuation

タイプ: MQLONG - 出力

コンポーネントによって設定される継続標識。指定可能な値は、次のとおりです。

MQZCI_DEFAULT

キュー・マネージャーに依存する継続。

MQZ_CHECK_AUTHORITY の場合、MQZCI_STOP と同じ効果があります。

MQZCI_CONTINUE

次のコンポーネントに継続。

MQZCI_STOP

次のコンポーネントに継続しない。

CompCode

タイプ: MQLONG - 出力

完了コード。値は、次のいずれかでなければなりません。

MQCC_OK

正常終了。

MQCC_FAILED

呼び出し失敗。

理由

タイプ: MQLONG - 出力

CompCode を限定する理由コード。

CompCode が MQCC_OK の場合、次のようになります。

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_FAILED の場合、次のようになります。

MQRC_SERVICE_ERROR

(2289, X'8F1') サービスのアクセスで、予期しないエラーが発生しました。

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') 基本サービスを使用できません。

MQRC_UNKNOWN_REF_OBJECT

(2294, X'8F6') 参照オブジェクトが不明です。

これらの理由コードについて詳しくは、[API 完了コードと理由コード](#)を参照してください。

C 言語での呼び出し

```
MQZ_COPY_ALL_AUTHORITY (QMgrName, RefObjectName, ObjectName, ObjectType,
                        ComponentData, &Continuation, &CompCode,
                        &Reason);
```

サービスに渡されるパラメーターは次のように宣言されます。

```
MQCHAR48  QMgrName;           /* Queue manager name */
MQCHAR48  RefObjectName;       /* Reference object name */
MQCHAR48  ObjectName;          /* Object name */
MQLONG    ObjectType;          /* Object type */
MQBYTE    ComponentData[n];    /* Component data */
MQLONG    Continuation;        /* Continuation indicator set by
                                component */
MQLONG    CompCode;            /* Completion code */
MQLONG    Reason;              /* Reason code qualifying CompCode */
```

MQZ_DELETE_AUTHORITY - 権限の削除

この関数は、許可サービス・コンポーネントに用意されており、キュー・マネージャーにより開始され、指定されたオブジェクトに関連するすべての許可を削除します。

この関数の関数 ID (MQZEP 用) は、MQZID_DELETE_AUTHORITY です。

構文

MQZ_DELETE_AUTHORITY(QMgrName , ObjectName , ObjectType , ComponentData , Continuation , CompCode , Reason)

パラメーター

QMgrName

タイプ: MQCHAR48 - 入力

キュー・マネージャー名。コンポーネントを呼び出すキュー・マネージャーの名前。この名前は、パラメーターがすべて埋まるまでブランクが埋め込まれます。名前の最後をヌル文字にすることはできません。

キュー・マネージャー名は、情報としてコンポーネントに渡されます。許可サービス・インターフェースでは、コンポーネントは定義されている方法でこの情報を使用する必要はありません。

ObjectName

タイプ: MQCHAR48 - 入力

オブジェクト名 オブジェクトの名前で、それに対するアクセスが削除されます。ストリングの長さは最大で 48 文字です。48 文字未満の場合、その名前の右側がブランクで埋められます。名前の最後をヌル文字にすることはできません。

ObjectType が MQOT_Q_MGR の場合、この名前は *QMgrName* と同じになります。

ObjectType

タイプ: MQLONG - 入力

オブジェクト・タイプ *ObjectName* によって指定されるエンティティのタイプ。値は、次のいずれかでなければなりません。

MQOT_AUTH_INFO

認証情報

MQOT_CHANNEL

チャンネル。

MQOT_CLNTCONN_CHANNEL

クライアント接続チャンネル。

MQOT_LISTENER

リスナー

MQOT_NAMELIST

名前リスト。

MQOT_PROCESS

プロセス定義。

MQOT_Q

キュー。

MQOT_Q_MGR

キュー・マネージャー。

MQOT_SERVICE

サービス

MQOT_TOPIC

トピック。

ComponentData

タイプ: MQBYTE x ComponentDataLength - 入出力

コンポーネント・データ。このデータは、この特定のコンポーネントのためにキュー・マネージャーで保持されます。このコンポーネントに用意されているいずれかの関数で変更された内容は保持され、これらのコンポーネントのいずれかの関数が次回に呼び出されると提示されます。

このデータ域の長さは、キュー・マネージャーにより MQZ_INIT_AUTHORITY 呼び出しの ComponentDataLength パラメーターに入れられて渡されます。

Continuation

タイプ: MQLONG - 出力

コンポーネントによって設定される継続標識。指定可能な値は、次のとおりです。

MQZCI_DEFAULT

キュー・マネージャーに依存する継続。

MQZ_CHECK_AUTHORITY の場合、MQZCI_STOP と同じ効果があります。

MQZCI_CONTINUE

次のコンポーネントに継続。

MQZCI_STOP

次のコンポーネントに継続しない。

CompCode

タイプ: MQLONG - 出力

完了コード。値は、次のいずれかでなければなりません。

MQCC_OK

正常終了。

MQCC_FAILED

呼び出し失敗。

理由

タイプ: MQLONG - 出力

CompCode を限定する理由コード。

CompCode が MQCC_OK の場合、次のようになります。

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_FAILED の場合、次のようになります。

MQRC_SERVICE_ERROR

(2289, X'8F1') サービスのアクセスで、予期しないエラーが発生しました。

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') 基本サービスを使用できません。

これらの理由コードについて詳しくは、[API 完了コードと理由コード](#)を参照してください。

C 言語での呼び出し

```
MQZ_DELETE_AUTHORITY (QMgrName, ObjectName, ObjectType, ComponentData,  
                      &Continuation, &CompCode, &Reason);
```

サービスに渡されるパラメーターは次のように宣言されます。

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQCHAR48  ObjectName;         /* Object name */  
MQLONG     ObjectType;        /* Object type */  
MQBYTE     ComponentData[n]; /* Component data */  
MQLONG     Continuation;      /* Continuation indicator set by  
                               component */
```

```
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

MQZ_ENUMERATE_AUTHORITY_DATA - 権限データの列挙

この関数は、MQZAS_VERSION_4 許可サービス・コンポーネントにより提供され、キュー・マネージャーによって繰り返し開始されて、最初の呼び出しで指定された選択基準に合致するすべての権限データを取得します。

この関数の関数 ID (MQZEP 用) は、MQZID_ENUMERATE_AUTHORITY_DATA です。

構文

```
MQZ_ENUMERATE_AUTHORITY_DATA( QMgrName , StartEnumeration , Filter ,  
AuthorityBufferLength , AuthorityBuffer , AuthorityDataLength , ComponentData ,  
Continuation , CompCode , Reason )
```

パラメーター

QMgrName

タイプ: MQCHAR48 - 入力

キュー・マネージャー名。コンポーネントを呼び出すキュー・マネージャーの名前。この名前は、パラメーターがすべて埋まるまでブランクが埋め込まれます。名前の最後をヌル文字にすることはできません。

キュー・マネージャー名は、情報としてコンポーネントに渡されます。許可サービス・インターフェースでは、コンポーネントは定義されている方法でこの情報を使用する必要はありません。

StartEnumeration

タイプ: MQLONG - 入力

呼び出しで列挙を開始できるかどうかを示すフラグ。これは、この呼び出しによって権限データの列挙を開始できるか、それとも MQZ_ENUMERATE_AUTHORITY_DATA に対する前回の呼び出しによって開始された権限データの列挙を継続するかを示します。値は、以下のいずれかの値です。

MQZSE_START

列挙を開始します。この値を使用して呼び出しが開始されると、権限データの列挙を開始します。

Filter パラメーターに、この呼び出しとその後の呼び出しで返される権限データを選択するのに使用する選択基準を指定します。

MQZSE_CONTINUE

列挙を継続します。この値を使用して呼び出しが開始されると、権限データの列挙を継続します。この場合、**Filter** パラメーターは無視され、ヌル・ポインターとして指定できます (選択基準は、*StartEnumeration* が MQZSE_START に設定された呼び出しによって指定された **Filter** パラメーターによって決定されます)。

Filter

タイプ: MQZAD - 入力

フィルター。*StartEnumeration* が MQZSE_START の場合、*Filter* は、返される権限データを選択するために使用する選択基準を指定します。*Filter* がヌル・ポインターの場合、選択基準は使用されず、すべての権限データが返されます。使用できる選択基準の詳細については、[1679 ページの『MQZAD - 権限データ』](#)を参照してください。

StartEnumeration が MQZSE_CONTINUE の場合、*Filter* は無視され、ヌル・ポインターとして指定できます。

AuthorityBufferLength

タイプ: MQLONG - 入力

AuthorityBuffer の長さ。これは、**AuthorityBuffer** パラメーターのバイト単位の長さです。権限バッファは、戻されるデータを収容するのに十分な大きさでなければなりません。

AuthorityBuffer

タイプ: MQZAD - 出力

権限データ。権限データが返されるバッファーです。このバッファーは、MQZAD 構造体、MQZED 構造体、および定義された最長のエンティティ名と最長のドメイン名を格納できる大きさでなければなりません。

注: 注: このパラメーターは、MQZAD として定義されます。MQZAD は常にバッファーの先頭に現れるからです。ただし、バッファーを MQZAD として宣言すると、バッファーが小さくなりすぎます。MQZAD より大きくして、MQZAD、MQZED、およびエンティティ名とドメイン名が収まるようにする必要があります。

AuthorityDataLength

タイプ: MQLONG - 出力

AuthorityBuffer に返されるデータ長。権限バッファーが小さすぎると、*AuthorityDataLength* は必要なバッファー長に設定され、呼び出しで完了コード MQCC_FAILED と理由コード MQRC_BUFFER_LENGTH_ERROR が返されます。

ComponentData

タイプ: MQBYTE x ComponentDataLength - 入出力

コンポーネント・データ。このデータは、この特定のコンポーネントのためにキュー・マネージャーで保持されます。このコンポーネントに用意されているいずれかの関数で変更された内容は保持され、これらのコンポーネントのいずれかの関数が次回に呼び出されると提示されます。

このデータ域の長さは、キュー・マネージャーにより MQZ_INIT_AUTHORITY 呼び出しの ComponentDataLength パラメーターに入れられて渡されます。

Continuation

タイプ: MQLONG - 出力

コンポーネントによって設定される継続標識。指定可能な値は、次のとおりです。

MQZCI_DEFAULT

キュー・マネージャーに依存する継続。

MQZ_ENUMERATE_AUTHORITY_DATA の場合、これは MQZCI_CONTINUE と同じ効果があります。

MQZCI_CONTINUE

次のコンポーネントに継続。

MQZCI_STOP

次のコンポーネントに継続しない。

CompCode

タイプ: MQLONG - 出力

完了コード。値は、次のいずれかでなければなりません。

MQCC_OK

正常終了。

MQCC_FAILED

呼び出し失敗。

理由

タイプ: MQLONG - 出力

CompCode を限定する理由コード。

CompCode が MQCC_OK の場合、次のようになります。

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_FAILED の場合、次のようになります。

MQRC_BUFFER_LENGTH_ERROR

(2005, X'7D5') バッファー長パラメーターは無効です。

MQRC_NO_DATA_AVAILABLE

(2379, X'94B') 使用可能なデータはありません。

MQRC_SERVICE_ERROR

(2289, X'8F1') サービスのアクセスで、予期しないエラーが発生しました。

これらの理由コードについて詳しくは、[API 完了コードと理由コード](#)を参照してください。

C 言語での呼び出し

```
MQZ_ENUMERATE_AUTHORITY_DATA (QMgrName, StartEnumeration, &Filter,  
                               AuthorityBufferLength,  
                               &AuthorityBuffer,  
                               &AuthorityDataLength, ComponentData,  
                               &Continuation, &CompCode,  
                               &Reason);
```

サービスに渡されるパラメーターは次のように宣言されます。

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQLONG    StartEnumeration;   /* Flag indicating whether call should  
                               start enumeration */  
  
MQZAD     Filter;             /* Filter */  
MQLONG    AuthorityBufferLength; /* Length of AuthorityBuffer */  
MQZAD     AuthorityBuffer;    /* Authority data */  
MQLONG    AuthorityDataLength; /* Length of data returned in  
                               AuthorityBuffer */  
  
MQBYTE    ComponentData[n];   /* Component data */  
MQLONG    Continuation;       /* Continuation indicator set by  
                               component */  
  
MQLONG    CompCode;           /* Completion code */  
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

MQZ_FREE_USER - ユーザーの解放

この関数は、MQZAS_VERSION_5 許可サービス・コンポーネントに用意されており、キュー・マネージャーにより開始され、関連している割り振り済みのリソースを解放します。

これは、アプリケーションがすべてのユーザー・コンテキストの下で (例えば、MQDISC MQI 呼び出し時に) 実行を終了すると開始されます。

この関数の関数 ID (MQZEP 用) は、MQZID_FREE_USER です。

構文

`MQZ_FREE_USER( QMgrName , FreeParms , ComponentData , Continuation , CompCode , Reason )`

パラメーター

QMgrName

タイプ: MQCHAR48 - 入力

キュー・マネージャー名。コンポーネントを呼び出すキュー・マネージャーの名前。この名前は、パラメーターがすべて埋まるまでブランクが埋め込まれます。名前の最後をヌル文字にすることはできません。

キュー・マネージャー名は、情報としてコンポーネントに渡されます。許可サービス・インターフェースでは、コンポーネントは定義されている方法でこの情報を使用する必要はありません。

FreeParms

タイプ: MQZFP - 入力

解放パラメーター。解放の対象となるリソースに関連するデータが格納されている構造体。詳細は [1684 ページの『MQZFP - 解放パラメーター』](#)を参照してください。

ComponentData

タイプ: MQBYTE x ComponentDataLength - 入出力

コンポーネント・データ。このデータは、この特定のコンポーネントのためにキュー・マネージャーで保持されます。このコンポーネントに用意されているいずれかの関数で変更された内容は保持され、これらのコンポーネントのいずれかの関数が次回に呼び出されると提示されます。

このデータ域の長さは、キュー・マネージャーにより MQZ_INIT_AUTHORITY 呼び出しの ComponentDataLength パラメーターに入れられて渡されます。

Continuation

タイプ: MQLONG - 出力

継続フラグ。指定可能な値は、次のとおりです。

MQZCI_DEFAULT

他のコンポーネントに依存する継続。

MQZCI_STOP

次のコンポーネントに継続しない。

CompCode

タイプ: MQLONG - 出力

完了コード。値は、次のいずれかでなければなりません。

MQCC_OK

正常終了。

MQCC_FAILED

呼び出し失敗。

理由

タイプ: MQLONG - 出力

CompCode を限定する理由コード。

CompCode が MQCC_OK の場合、次のようになります。

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_FAILED の場合、次のようになります。

MQRC_SERVICE_ERROR

(2289, X'8F1') サービスのアクセスで、予期しないエラーが発生しました。

これらの理由コードについて詳しくは、[API 完了コードと理由コード](#)を参照してください。

C 言語での呼び出し

```
MQZ_AUTHENTICATE_USER (QMgrName, SecurityParms, ApplicationContext,
                        IdentityContext, CorrelationPtr, ComponentData,
                        &Continuation, &CompCode, &Reason);
```

サービスに渡されるパラメーターは次のように宣言されます。

```
MQCHAR48  QMgrName;           /* Queue manager name */
MQZFP      FreeParms;          /* Resource to be freed */
MQBYTE     ComponentData[n];   /* Component data */
MQLONG     Continuation;       /* Continuation indicator set by
                                component */
MQLONG     CompCode;           /* Completion code */
MQLONG     Reason;             /* Reason code qualifying CompCode */
```

MQZ_GET_AUTHORITY - 権限の取得

MQZAS_VERSION_1 許可サービス・コンポーネントによって提供されるこの関数は、キュー・マネージャーによって開始されて、指定されたオブジェクトに対するエンティティのアクセス権限を取得します。また、プリンシパルがメンバーになっているグループによって処理される権限も取得されます (エンティティがプリンシパルである場合)。総称プロファイルにおける権限が、返される権限セットに含められます。

この関数の関数 ID (MQZEP 用) は、MQZID_GET_AUTHORITY です。

構文

MQZ_GET_AUTHORITY(QMgrName , EntityName , EntityType , ObjectName ,
ObjectType , Authority , ComponentData , Continuation , CompCode , Reason)

パラメーター

QMgrName

タイプ: MQCHAR48 - 入力

キュー・マネージャー名。コンポーネントを呼び出すキュー・マネージャーの名前。この名前は、パラメーターがすべて埋まるまでブランクが埋め込まれます。名前の最後をヌル文字にすることはできません。

キュー・マネージャー名は、情報としてコンポーネントに渡されます。許可サービス・インターフェースでは、コンポーネントは定義されている方法でこの情報を使用する必要はありません。

EntityName

タイプ: MQCHAR12 - 入力

エンティティ名。オブジェクトに対するアクセス権限を検索するエンティティの名前。ストリングの長さは最大で 12 文字です。12 文字未満の場合、その名前の右側がブランクで埋められます。名前の最後をヌル文字にすることはできません。

EntityType

タイプ: MQLONG - 入力

エンティティ・タイプ。EntityName によって指定されるエンティティのタイプ。値は、次のいずれかでなければなりません。

MQZAET_PRINCIPAL

プリンシパル。

MQZAET_GROUP

グループ。

ObjectName

タイプ: MQCHAR48 - 入力

オブジェクト名 アクセス権限を取得するオブジェクトの名前。ストリングの長さは最大で 48 文字です。48 文字未満の場合、その名前の右側がブランクで埋められます。名前の最後をヌル文字にすることはできません。

ObjectType が MQOT_Q_MGR の場合、この名前は QMgrName と同じになります。

ObjectType

タイプ: MQLONG - 入力

オブジェクト・タイプ ObjectName によって指定されるエンティティのタイプ。値は、次のいずれかでなければなりません。

MQOT_AUTH_INFO

認証情報

MQOT_CHANNEL

チャンネル。

MQOT_CLNTCONN_CHANNEL

クライアント接続チャンネル。

MQOT_LISTENER

リスナー

MQOT_NAMELIST

名前リスト。

MQOT_PROCESS

プロセス定義。

MQOT_Q

キュー。

MQOT_Q_MGR

キュー・マネージャー。

MQOT_SERVICE

サービス

MQOT_TOPIC

トピック。

権限

タイプ: MQLONG - 入力

エンティティの権限。エンティティの権限が1つの場合、このフィールドは該当する許可演算 (MQZAO_* 定数) に等しくなります。複数の権限を持つ場合、このフィールドは対応する MQZAO_* 定数のビット単位 OR になります。

ComponentData

タイプ: MQBYTE×ComponentDataLength - 入出力

コンポーネント・データ。このデータは、この特定のコンポーネントのためにキュー・マネージャーで保持されます。このコンポーネントに用意されているいずれかの関数で変更された内容は保持され、これらのコンポーネントのいずれかの関数が次回に呼び出されると提示されます。

このデータ域の長さは、キュー・マネージャーにより MQZ_INIT_AUTHORITY 呼び出しの **ComponentDataLength** パラメーターに入れられて渡されます。

Continuation

タイプ: MQLONG - 出力

コンポーネントによって設定される継続標識。指定可能な値は、次のとおりです。

MQZCI_DEFAULT

キュー・マネージャーに依存する継続。

MQZ_GET_AUTHORITY の場合、MQZCI_CONTINUE と同じ効果があります。

MQZCI_CONTINUE

次のコンポーネントに継続。

MQZCI_STOP

次のコンポーネントに継続しない。

CompCode

タイプ: MQLONG - 出力

完了コード。値は、次のいずれかでなければなりません。

MQCC_OK

正常終了。

MQCC_FAILED

呼び出し失敗。

理由

タイプ: MQLONG - 出力

CompCode を限定する理由コード。

CompCode が MQCC_OK の場合、次のようになります。

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_FAILED の場合、次のようになります。

MQRC_NOT_AUTHORIZED

(2035, X'7F3') アクセスは許可されません。

MQRC_SERVICE_ERROR

(2289, X'8F1') サービスのアクセスで、予期しないエラーが発生しました。

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') 基本サービスを使用できません。

MQRC_UNKNOWN_ENTITY

(2292, X'8F4') サービスに対して不明なエンティティです。

これらの理由コードについて詳しくは、[API 完了コードと理由コード](#)を参照してください。

C 言語での呼び出し

```
MQZ_GET_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName,  
                   ObjectType, &Authority, ComponentData,  
                   &Continuation, &CompCode, &Reason);
```

サービスに渡されるパラメーターは次のように宣言されます。

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQCHAR12  EntityName;         /* Entity name */  
MQLONG    EntityType;         /* Entity type */  
MQCHAR48  ObjectName;         /* Object name */  
MQLONG    ObjectType;         /* Object type */  
MQLONG    Authority;          /* Authority of entity */  
MQBYTE    ComponentData[n];   /* Component data */  
MQLONG    Continuation;       /* Continuation indicator set by  
                               component */  
MQLONG    CompCode;           /* Completion code */  
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

MQZ_GET_AUTHORITY_2 - 権限の取得 (拡張版)

この関数は、MQZAS_VERSION_2 許可サービス・コンポーネントにより提供され、キュー・マネージャーにより開始されて、エンティティが持っている指定されたオブジェクトへのアクセス権限を取得します。

この関数の関数 ID (MQZEP 用) は、MQZID_GET_AUTHORITY です。

MQZ_GET_AUTHORITY_2 は MQZ_GET_AUTHORITY と似ていますが、**EntityName** パラメーターが **EntityData** パラメーターで置き換えられています。

構文

MQZ_GET_AUTHORITY_2(QMgrName , EntityData , EntityType , ObjectName ,
ObjectType , Authority , ComponentData , Continuation , CompCode , Reason)

パラメーター

QMgrName

タイプ: MQCHAR48 - 入力

キュー・マネージャー名。コンポーネントを呼び出すキュー・マネージャーの名前。この名前は、パラメーターがすべて埋まるまでブランクが埋め込まれます。名前の最後をヌル文字にすることはできません。

キュー・マネージャー名は、情報としてコンポーネントに渡されます。許可サービス・インターフェースでは、コンポーネントは定義されている方法でこの情報を使用する必要はありません。

EntityData

タイプ: MQZED - 入力

エンティティ・データ。オブジェクトに対する許可が取得されるエンティティに関連するデータです。詳細は [1682 ページ](#)の『MQZED - エンティティ記述子』を参照してください。

EntityType

タイプ: MQLONG - 入力

エンティティ・タイプ。 *EntityData* によって指定されるエンティティのタイプ。 値は、次のいずれかでなければなりません。

MQZAET_PRINCIPAL

プリンシパル。

MQZAET_GROUP

グループ。

ObjectName

タイプ: MQCHAR48 - 入力

オブジェクト名 エンティティの権限を取得するオブジェクトの名前です。 スtringの長さは最大で 48 文字です。 48 文字未満の場合、その名前の右側がブランクで埋められます。 名前の最後をヌル文字にすることはできません。

ObjectType が MQOT_Q_MGR の場合、この名前は *QMgrName* と同じになります。

ObjectType

タイプ: MQLONG - 入力

オブジェクト・タイプ *ObjectName* によって指定されるエンティティのタイプ。 値は、次のいずれかでなければなりません。

MQOT_AUTH_INFO

認証情報

MQOT_CHANNEL

チャネル。

MQOT_CLNTCONN_CHANNEL

クライアント接続チャネル。

MQOT_LISTENER

リスナー

MQOT_NAMELIST

名前リスト。

MQOT_PROCESS

プロセス定義。

MQOT_Q

キュー。

MQOT_Q_MGR

キュー・マネージャー。

MQOT_SERVICE

サービス

MQOT_TOPIC

トピック。

権限

タイプ: MQLONG - 入力

エンティティの権限。 エンティティの権限が 1 つの場合、このフィールドは該当する許可演算 (MQZAO_* 定数) に等しくなります。 複数の権限を持つ場合、このフィールドは対応する MQZAO_* 定数のビット単位 OR になります。

ComponentData

タイプ: MQBYTE×ComponentDataLength - 入出力

コンポーネント・データ。 このデータは、この特定のコンポーネントのためにキュー・マネージャーで保持されます。 このコンポーネントに用意されているいずれかの関数で変更された内容は保持され、これらのコンポーネントのいずれかの関数が次回に呼び出されると提示されます。

このデータ域の長さは、キュー・マネージャーにより MQZ_INIT_AUTHORITY 呼び出しの **ComponentDataLength** パラメーターに入れられて渡されます。

Continuation

タイプ: MQLONG - 出力

コンポーネントによって設定される継続標識。指定可能な値は、次のとおりです。

MQZCI_DEFAULT

キュー・マネージャーに依存する継続。

MQZ_CHECK_AUTHORITY の場合、MQZCI_STOP と同じ効果があります。

MQZCI_CONTINUE

次のコンポーネントに継続。

MQZCI_STOP

次のコンポーネントに継続しない。

CompCode

タイプ: MQLONG - 出力

完了コード。値は、次のいずれかでなければなりません。

MQCC_OK

正常終了。

MQCC_FAILED

呼び出し失敗。

理由

タイプ: MQLONG - 出力

CompCode を限定する理由コード。

CompCode が MQCC_OK の場合、次のようになります。

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_FAILED の場合、次のようになります。

MQRC_NOT_AUTHORIZED

(2035, X'7F3') アクセスは許可されません。

MQRC_SERVICE_ERROR

(2289, X'8F1') サービスのアクセスで、予期しないエラーが発生しました。

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') 基本サービスを使用できません。

MQRC_UNKNOWN_ENTITY

(2292, X'8F4') サービスに対して不明なエンティティです。

これらの理由コードについて詳しくは、[API 完了コードと理由コード](#)を参照してください。

C 言語での呼び出し

```
MQZ_GET_AUTHORITY_2 (QMgrName, &EntityData, EntityType, ObjectName,  
                    ObjectType, &Authority, ComponentData,  
                    &Continuation, &CompCode, &Reason);
```

サービスに渡されるパラメーターは次のように宣言されます。

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQZED     EntityData;         /* Entity data */  
MQLONG    EntityType;         /* Entity type */  
MQCHAR48  ObjectName;        /* Object name */  
MQLONG    ObjectType;         /* Object type */  
MQLONG    Authority;          /* Authority of entity */  
MQBYTE    ComponentData[n];   /* Component data */  
MQLONG    Continuation;       /* Continuation indicator set by
```

		component */
MQLONG	CompCode;	/* Completion code */
MQLONG	Reason;	/* Reason code qualifying CompCode */

MQZ_GET_EXPLICIT_AUTHORITY - 明示的権限の取得

MQZAS_VERSION_1 許可サービス・コンポーネントによって提供されるこの関数は、キュー・マネージャーによって開始されて、指定されたオブジェクトに対するエンティティのアクセス権限を取得します。また、プリンシパルがメンバーになっているグループによって処理される権限も取得されます (エンティティがプリンシパルである場合)。総称プロファイルにおける権限が、返される権限セットに含められます。

AIX and Linux 上の組み込み IBM MQ オブジェクト権限マネージャー (OAM) の場合は、プリンシパルの 1 次グループのみが所有する権限が返されます。

この関数の関数 ID (MQZEP 用) は、MQZID_GET_EXPLICIT_AUTHORITY です。

構文

MQZ_GET_EXPLICIT_AUTHORITY(QMgrName , EntityName , EntityType , ObjectName , ObjectType , Authority , ComponentData , Continuation , CompCode , Reason)

パラメーター

QMgrName

タイプ: MQCHAR48 - 入力

キュー・マネージャー名。コンポーネントを呼び出すキュー・マネージャーの名前。この名前は、パラメーターがすべて埋まるまでブランクが埋め込まれます。名前の最後をヌル文字にすることはできません。

キュー・マネージャー名は、情報としてコンポーネントに渡されます。許可サービス・インターフェースでは、コンポーネントは定義されている方法でこの情報を使用する必要はありません。

EntityName

タイプ: MQCHAR12 - 入力

エンティティ名。オブジェクトに対するアクセス権限を取得するエンティティの名前。ストリングの長さは最大で 12 文字です。12 文字未満の場合、その名前の右側がブランクで埋められます。名前の最後をヌル文字にすることはできません。

EntityType

タイプ: MQLONG - 入力

エンティティ・タイプ。EntityName によって指定されるエンティティのタイプ。値は、次のいずれかでなければなりません。

MQZAET_PRINCIPAL

プリンシパル。

MQZAET_GROUP

グループ。

ObjectName

タイプ: MQCHAR48 - 入力

オブジェクト名 エンティティの権限を取得するオブジェクトの名前です。ストリングの長さは最大で 48 文字です。48 文字未満の場合、その名前の右側がブランクで埋められます。名前の最後をヌル文字にすることはできません。

ObjectType が MQOT_Q_MGR の場合、この名前は QMgrName と同じになります。

ObjectType

タイプ: MQLONG - 入力

オブジェクト・タイプ ObjectName によって指定されるエンティティのタイプ。値は、次のいずれかでなければなりません。

MQOT_AUTH_INFO

認証情報

MQOT_CHANNEL

チャンネル。

MQOT_CLNTCONN_CHANNEL

クライアント接続チャンネル。

MQOT_LISTENER

リスナー

MQOT_NAMELIST

名前リスト。

MQOT_PROCESS

プロセス定義。

MQOT_Q

キュー。

MQOT_Q_MGR

キュー・マネージャー。

MQOT_SERVICE

サービス

MQOT_TOPIC

トピック。

権限

タイプ: MQLONG - 入力

エンティティの権限。エンティティの権限が 1 つの場合、このフィールドは該当する許可演算 (MQZAO_* 定数) に等しくなります。複数の権限を持つ場合、このフィールドは対応する MQZAO_* 定数のビット単位 OR になります。

ComponentData

タイプ: MQBYTE x ComponentDataLength - 入出力

コンポーネント・データ。このデータは、この特定のコンポーネントのためにキュー・マネージャーで保持されます。このコンポーネントに用意されているいずれかの関数で変更された内容は保持され、これらのコンポーネントのいずれかの関数が次回に呼び出されると提示されます。

このデータ域の長さは、キュー・マネージャーにより MQZ_INIT_AUTHORITY 呼び出しの **ComponentDataLength** パラメーターに入れられて渡されます。

Continuation

タイプ: MQLONG - 出力

コンポーネントによって設定される継続標識。指定可能な値は、次のとおりです。

MQZCI_DEFAULT

キュー・マネージャーに依存する継続。

MQZ_GET_AUTHORITY の場合、MQZCI_CONTINUE と同じ効果があります。

MQZCI_CONTINUE

次のコンポーネントに継続。

MQZCI_STOP

次のコンポーネントに継続しない。

CompCode

タイプ: MQLONG - 出力

完了コード。値は、次のいずれかでなければなりません。

MQCC_OK

正常終了。

MQCC_FAILED

呼び出し失敗。

理由

タイプ: MQLONG - 出力

CompCode を限定する理由コード。

CompCode が MQCC_OK の場合、次のようになります。

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_FAILED の場合、次のようになります。

MQRC_NOT_AUTHORIZED

(2035, X'7F3') アクセスは許可されません。

MQRC_SERVICE_ERROR

(2289, X'8F1') サービスのアクセスで、予期しないエラーが発生しました。

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') 基本サービスを使用できません。

MQRC_UNKNOWN_ENTITY

(2292, X'8F4') サービスに対して不明なエンティティです。

これらの理由コードについて詳しくは、[API 完了コードと理由コード](#)を参照してください。

C 言語での呼び出し

```
MQZ_GET_EXPLICIT_AUTHORITY (QMgrName, EntityName, EntityType,
                             ObjectName, ObjectType, &Authority,
                             ComponentData, &Continuation,
                             &CompCode, &Reason);
```

サービスに渡されるパラメーターは次のように宣言されます。

```
MQCHAR48  QMgrName;           /* Queue manager name */
MQCHAR12  EntityName;         /* Entity name */
MQLONG    EntityType;         /* Entity type */
MQCHAR48  ObjectName;         /* Object name */
MQLONG    ObjectType;         /* Object type */
MQLONG    Authority;          /* Authority of entity */
MQBYTE    ComponentData[n];   /* Component data */
MQLONG    Continuation;       /* Continuation indicator set by
                               component */
MQLONG    CompCode;           /* Completion code */
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

MQZ_GET_EXPLICIT_AUTHORITY_2 - 明示的権限の取得 (拡張版)

この関数は、MQZAS_VERSION_2 許可サービス・コンポーネントにより提供され、キュー・マネージャーにより開始されて、名前付きグループが持っている指定されたオブジェクトへのアクセス権限 (ただし、**nobody (なし)** グループによる追加権限は除く) を取得するか、名前付きプリンシパルの主グループが持っている指定されたオブジェクトへのアクセス権限を取得します。

この関数の関数 ID (MQZEP 用) は、MQZID_GET_EXPLICIT_AUTHORITY です。

MQZ_GET_EXPLICIT_AUTHORITY_2 は MQZ_GET_EXPLICIT_AUTHORITY と似ていますが、**EntityName** パラメーターが **EntityData** パラメーターで置き換えられています。

構文

```
MQZ_GET_EXPLICIT_AUTHORITY_2( QMgrName , EntityData , EntityType , ObjectName ,
                               ObjectType , Authority , ComponentData , Continuation , CompCode , Reason )
```


パラメーター

QMgrName

タイプ: MQCHAR48 - 入力

キュー・マネージャー名。コンポーネントを呼び出すキュー・マネージャーの名前。この名前は、パラメーターがすべて埋まるまでブランクが埋め込まれます。名前の最後をヌル文字にすることはできません。

キュー・マネージャー名は、情報としてコンポーネントに渡されます。許可サービス・インターフェースでは、コンポーネントは定義されている方法でこの情報を使用する必要はありません。

EntityData

タイプ: MQZED - 入力

エンティティ・データ。オブジェクトに対する許可が取得されるエンティティに関連するデータです。詳細は [1682 ページの『MQZED - エンティティ記述子』](#) を参照してください。

EntityType

タイプ: MQLONG - 入力

エンティティ・タイプ。 *EntityData* によって指定されるエンティティのタイプ。値は、次のいずれかでなければなりません。

MQZAET_PRINCIPAL

プリンシパル。

MQZAET_GROUP

グループ。

ObjectName

タイプ: MQCHAR48 - 入力

オブジェクト名 エンティティの権限を取得するオブジェクトの名前です。stringの長さは最大で 48 文字です。48 文字未満の場合、その名前の右側がブランクで埋められます。名前の最後をヌル文字にすることはできません。

ObjectType が MQOT_Q_MGR の場合、この名前は *QMgrName* と同じになります。

ObjectType

タイプ: MQLONG - 入力

オブジェクト・タイプ *ObjectName* によって指定されるエンティティのタイプ。値は、次のいずれかでなければなりません。

MQOT_AUTH_INFO

認証情報

MQOT_CHANNEL

チャンネル。

MQOT_CLNTCONN_CHANNEL

クライアント接続チャンネル。

MQOT_LISTENER

リスナー

MQOT_NAMELIST

名前リスト。

MQOT_PROCESS

プロセス定義。

MQOT_Q

キュー。

MQOT_Q_MGR

キュー・マネージャー。

MQOT_SERVICE

サービス

MQOT_TOPIC

トピック。

権限

タイプ: MQLONG - 入力

エンティティの権限。エンティティの権限が1つの場合、このフィールドは該当する許可演算 (MQZAO_* 定数) に等しくなります。複数の権限を持つ場合、このフィールドは対応する MQZAO_* 定数のビット単位 OR になります。

ComponentData

タイプ: MQBYTE×ComponentDataLength - 入出力

コンポーネント・データ。このデータは、この特定のコンポーネントのためにキュー・マネージャーで保持されます。このコンポーネントに用意されているいずれかの関数で変更された内容は保持され、これらのコンポーネントのいずれかの関数が次回に呼び出されると提示されます。

このデータ域の長さは、キュー・マネージャーにより MQZ_INIT_AUTHORITY 呼び出しの **ComponentDataLength** パラメーターに入れられて渡されます。

Continuation

タイプ: MQLONG - 出力

コンポーネントによって設定される継続標識。指定可能な値は、次のとおりです。

MQZCI_DEFAULT

キュー・マネージャーに依存する継続。

MQZ_CHECK_AUTHORITY の場合、MQZCI_STOP と同じ効果があります。

MQZCI_CONTINUE

次のコンポーネントに継続。

MQZCI_STOP

次のコンポーネントに継続しない。

CompCode

タイプ: MQLONG - 出力

完了コード。値は、次のいずれかでなければなりません。

MQCC_OK

正常終了。

MQCC_FAILED

呼び出し失敗。

理由

タイプ: MQLONG - 出力

CompCode を限定する理由コード。

CompCode が MQCC_OK の場合、次のようになります。

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_FAILED の場合、次のようになります。

MQRC_NOT_AUTHORIZED

(2035, X'7F3') アクセスは許可されません。

MQRC_SERVICE_ERROR

(2289, X'8F1') サービスのアクセスで、予期しないエラーが発生しました。

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') 基本サービスを使用できません。

MQRC_UNKNOWN_ENTITY

(2292, X'8F4') サービスに対して不明なエンティティです。

これらの理由コードについて詳しくは、[API 完了コードと理由コード](#)を参照してください。

C 言語での呼び出し

```
MQZ_GET_EXPLICIT_AUTHORITY_2 (QMgrName, &EntityData, EntityType,  
                               ObjectName, ObjectType, &Authority,  
                               ComponentData, &Continuation,  
                               &CompCode, &Reason);
```

サービスに渡されるパラメーターは次のように宣言されます。

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQZED     EntityData;         /* Entity data */  
MQLONG    EntityType;         /* Entity type */  
MQCHAR48  ObjectName;         /* Object name */  
MQLONG    ObjectType;         /* Object type */  
MQLONG    Authority;          /* Authority of entity */  
MQBYTE    ComponentData[n];   /* Component data */  
MQLONG    Continuation;       /* Continuation indicator set by  
                               component */  
MQLONG    CompCode;           /* Completion code */  
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

MQZ_INIT_AUTHORITY - 許可サービスの初期化

この関数は、許可サービス・コンポーネントにより提供され、コンポーネントの構成中にキュー・マネージャーにより開始されます。キュー・マネージャーに情報を提供するために、MQZEP を呼び出すことが想定されます。

この関数の関数 ID (MQZEP 用) は、MQZID_INIT_AUTHORITY です。

構文

MQZ_INIT_AUTHORITY(Hconfig , Options , QMgrName , ComponentDataLength , ComponentData , Version , CompCode , Reason)

パラメーター

Hconfig

タイプ: MQHCONFIG - 入力

構成ハンドル。このハンドルは、初期化される特定のコンポーネントを表します。これは、MQZEP 関数を使用してキュー・マネージャーを呼び出す際に、コンポーネントにより使用されます。

オプション

タイプ: MQLONG - 入力

初期化オプション。値は、次のいずれかでなければなりません。

MQZIO_PRIMARY

1 次初期化。

MQZIO_SECONDARY

2 次初期化。

QMgrName

タイプ: MQCHAR48 - 入力

キュー・マネージャー名。コンポーネントを呼び出すキュー・マネージャーの名前。この名前は、パラメーターがすべて埋まるまでブランクが埋め込まれます。名前の最後をヌル文字にすることはできません。

キュー・マネージャー名は、情報としてコンポーネントに渡されます。許可サービス・インターフェースでは、コンポーネントは定義されている方法でこの情報を使用する必要はありません。

ComponentDataLength

タイプ: MQLONG - 入力

コンポーネント・データの長さ。 *ComponentData* 域の長さ (バイト単位)。この長さは、コンポーネント構成データに定義されます。

ComponentData

タイプ: MQBYTE x ComponentDataLength - 入出力

コンポーネント・データ。コンポーネントの 1 次初期化関数が呼び出される前に、すべてゼロに初期化されます。このデータは、この特定のコンポーネントの代わりにキュー・マネージャーが保持します。このコンポーネントの提供するいずれかの関数 (初期化関数を含む) によって変更された内容がすべて保持され、それらのコンポーネント関数の 1 つが次回に呼び出されるときに提示されます。

このデータ域の長さは、キュー・マネージャーにより MQZ_INIT_AUTHORITY 呼び出しの **ComponentDataLength** パラメーターに入れられて渡されます。

バージョン

タイプ: MQLONG - 入出力

バージョン番号。初期化関数への入力時に、これによりそのキュー・マネージャーがサポートする最新のバージョン番号を識別します。初期化関数は、必要により、キュー・マネージャーがサポートするインターフェースのバージョンに変更する必要があります。キュー・マネージャーがコンポーネントから返されるバージョンをサポートしないことが返されると、キュー・マネージャーはコンポーネントの MQZ_TERM_AUTHORITY 関数を呼び出し、その後はこのコンポーネントを使用しません。

次の値がサポートされます。

MQZAS_VERSION_1

バージョン 1。

MQZAS_VERSION_2

バージョン 2。

MQZAS_VERSION_3

バージョン 3。

MQZAS_VERSION_4

バージョン 4。

MQZAS_VERSION_5

バージョン 5。

MQZAS_VERSION_6

IBM WebSphere MQ 6。

CompCode

タイプ: MQLONG - 出力

完了コード。値は、次のいずれかでなければなりません。

MQCC_OK

正常終了。

MQCC_FAILED

呼び出し失敗。

理由

タイプ: MQLONG - 出力

CompCode を限定する理由コード。

CompCode が MQCC_OK の場合、次のようになります。

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_FAILED の場合、次のようになります。

MQRC_INITIALIZATION_FAILED

(2286, X'8EE') 未定義の理由で初期化に失敗しました。

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') 基本サービスを使用できません。

これらの理由コードについて詳しくは、[API 完了コードと理由コード](#)を参照してください。

C 言語での呼び出し

```
MQZ_INIT_AUTHORITY (Hconfig, Options, QMgrName, ComponentDataLength,  
                    ComponentData, &Version, &CompCode,  
                    &Reason);
```

サービスに渡されるパラメーターは次のように宣言されます。

```
MQHCONFIG  Hconfig;           /* Configuration handle */  
MQLONG     Options;           /* Initialization options */  
MQCHAR48   QMgrName;         /* Queue manager name */  
MQLONG     ComponentDataLength; /* Length of component data */  
MQBYTE     ComponentData[n]; /* Component data */  
MQLONG     Version;          /* Version number */  
MQLONG     CompCode;         /* Completion code */  
MQLONG     Reason;           /* Reason code qualifying CompCode */
```

MQZ_INQUIRE - 許可サービスの照会

この関数は、MQZAS_VERSION_5 許可サービス・コンポーネントにより提供され、キュー・マネージャーにより開始されて、サポートされている機能を照会します。

複数のサービス・コンポーネントが使用されている場合、サービス・コンポーネントは、インストールされたときの順序とは逆の順序で呼び出されます。

この関数の関数 ID (MQZEP 用) は、MQZID_INQUIRE です。

構文

```
MQZ_INQUIRE( QMgrName , SelectorCount , Selectors , IntAttrCount , IntAttrs ,  
CharAttrLength , CharAttrs , SelectorReturned , ComponentData , Continuation ,  
CompCode , Reason )
```

パラメーター

QMgrName

タイプ: MQCHAR48 - 入力

キュー・マネージャー名。コンポーネントを呼び出すキュー・マネージャーの名前。この名前は、パラメーターがすべて埋まるまでブランクが埋め込まれます。名前の最後をヌル文字にすることはできません。

キュー・マネージャー名は、情報としてコンポーネントに渡されます。許可サービス・インターフェースでは、コンポーネントは定義されている方法でこの情報を使用する必要はありません。

SelectorCount

タイプ: MQLONG - 入力

セレクターの数。 **Selectors** パラメーターで提供されているセレクターの数。

値は 0 から 256 の範囲でなければなりません。

Selectors

タイプ: MQLONGxSelectorCount - 入力

セレクターの配列。各セレクターは必須の属性を示し、以下のいずれかである必要があります。

- MQIACF_INTERFACE_VERSION (整数)
- MQIACF_USER_ID_SUPPORT (整数)

- MQCACF_SERVICE_COMPONENT (文字)

選択子は任意の順序で指定できます。配列内のセレクター数は、**SelectorCount** パラメーターで指定します。

セレクターによって識別される整数属性は、*Selectors* に表示されるのと同じ順序で **IntAttrs** パラメーターに返されます。

セレクターによって識別される文字属性は、表示される順序と同じ順序で **CharAttrs** パラメーターに返されます *Selectors*。

IntAttrCount

タイプ: MQLONG - 入力

IntAttrs パラメーターで提供される整数属性の数。

値は 0 から 256 の範囲でなければなりません。

IntAttrs

タイプ: MQLONG x IntAttrCount - 出力

整数属性。整数属性の配列。整数属性は、*Selectors* 配列内の対応する整数セレクターと同じ順序で戻されます。

CharAttrCount

タイプ: MQLONG - 入力

文字属性バッファの長さ。 **CharAttrs** パラメーターの長さ (バイト単位)。

この値は、要求された文字属性の長さの合計以上でなければなりません。文字属性の要求がない場合は、ゼロが有効な値です。

CharAttrs

タイプ: MQLONG x CharAttrCount - 出力

文字属性バッファ。連結された文字属性が入るバッファ。文字属性は、*Selectors* 配列内の対応する文字セレクターと同じ順序で戻されます。

バッファ長は、CharAttrCount パラメーターで与えられます。

SelectorReturned

タイプ: MQLONG x SelectorCount - 入力

戻されたセレクター。 *Selectors* パラメーター内のセレクターにより要求されたセットの中から、どの属性が戻されたかを示す値の配列。この配列内の値の数は、**SelectorCount** パラメーターにより示されます。この配列内のそれぞれの値は、*Selectors* 配列の対応する位置にあるセレクターに関係付けられています。それぞれの値は以下のいずれかです。

MQZSL_RETURNED

Selectors パラメーター内の対応するセレクターにより要求された属性が戻されました。

MQZSL_NOT_RETURNED

Selectors パラメーター内の対応するセレクターにより要求された属性が戻されません。

この配列では、すべての値は **MQZSL_NOT_RETURNED** として初期化されます。許可サービス・コンポーネントは、属性を戻すと、配列内の該当する値を **MQZSL_NOT_RETURNED** に設定します。このようにすることで、照会呼び出しがなされた他の許可サービス・コンポーネントは、どの属性が既に返されたかを識別できます。

ComponentData

タイプ: MQBYTE x ComponentDataLength - 入出力

コンポーネント・データ。このデータは、この特定のコンポーネントのためにキュー・マネージャーで保持されます。このコンポーネントに用意されているいずれかの関数で変更された内容は保持され、これらのコンポーネントのいずれかの関数が次回に呼び出されると提示されます。

このデータ域の長さは、キュー・マネージャーにより **MQZ_INIT_AUTHORITY** 呼び出しの **ComponentDataLength** パラメーターに入れられて渡されます。

Continuation

タイプ: MQLONG - 出力

コンポーネントによって設定される継続標識。指定可能な値は、次のとおりです。

MQZCI_DEFAULT

キュー・マネージャーに依存する継続。

MQZ_CHECK_AUTHORITY の場合、MQZCI_STOP と同じ効果があります。

MQZCI_STOP

次のコンポーネントに継続しない。

CompCode

タイプ: MQLONG - 出力

完了コード。値は、次のいずれかでなければなりません。

MQCC_OK

正常終了。

MQCC_WARNING

一部完了。

MQCC_FAILED

呼び出し失敗。

理由

タイプ: MQLONG - 出力

CompCode を限定する理由コード。

CompCode が MQCC_OK の場合、次のようになります。

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_WARNING の場合、次のようになります。

MQRC_CHAR_ATTRS_TOO_SHORT

文字属性用のスペースが十分ではありません。

MQRC_INT_COUNT_TOO_SMALL

整数属性用のスペースが十分ではありません。

CompCode が MQCC_FAILED の場合、次のようになります。

MQRC_SELECTOR_COUNT_ERROR

セレクターの数が無効です。

MQRC_SELECTOR_ERROR

属性セレクターが無効です。

MQRC_SELECTOR_LIMIT_EXCEEDED

指定されたセレクターの数が多すぎます。

MQRC_INT_ATTR_COUNT_ERROR

整数属性の数が無効です。

MQRC_INT_ATTRS_ARRAY_ERROR

整数属性配列が無効です。

MQRC_CHAR_ATTR_LENGTH_ERROR

文字属性の数が無効です。

MQRC_CHAR_ATTRS_ERROR

文字属性ストリングが無効です。

MQRC_SERVICE_ERROR

(2289, X'8F1') サービスのアクセスで、予期しないエラーが発生しました。

これらの理由コードについて詳しくは、[API 完了コードと理由コード](#)を参照してください。

C 言語での呼び出し

```
MQZ_INQUIRE (QMgrName, SelectorCount, Selectors, IntAttrCount,
              &IntAttrs, CharAttrLength, &CharAttrs,
              SelectorReturned, ComponentData, &Continuation,
              &CompCode, &Reason);
```

サービスに渡されるパラメーターは次のように宣言されます。

```
MQCHAR48  QMgrName;           /* Queue manager name */
MQLONG     SelectorCount;      /* Selector count */
MQLONG     Selectors[n];       /* Selectors */
MQLONG     IntAttrCount;       /* IntAttrs count */
MQLONG     IntAttrs[n];       /* Integer attributes */
MQLONG     CharAttrCount;      /* CharAttrs count */
MQLONG     CharAttrs[n];      /* Character attributes */
MQLONG     SelectorReturned[n]; /* Selector returned */
MQBYTE     ComponentData[n];   /* Component data */
MQLONG     Continuation;       /* Continuation indicator set by
                                component */
MQLONG     CompCode;           /* Completion code */
MQLONG     Reason;            /* Reason code qualifying CompCode */
```

MQZ_REFRESH_CACHE - すべての許可の最新表示

この関数は、MQZAS_VERSION_3 許可サービス・コンポーネントに用意されており、キュー・マネージャーから呼び出されて、コンポーネントの内部に保持されている許可のリストをリフレッシュします。

この関数の関数 ID (MQZEP 用) は MQZID_REFRESH_CACHE (8L) です。

構文

```
MQZ_REFRESH_CACHE( QMgrName , ComponentData , Continuation , CompCode ,
Reason )
```

パラメーター

QMgrName

タイプ: MQCHAR48 - 入力

キュー・マネージャー名。コンポーネントを呼び出すキュー・マネージャーの名前。この名前は、パラメーターがすべて埋まるまで空白が埋め込まれます。名前の最後をヌル文字にすることはできません。

キュー・マネージャー名は、情報としてコンポーネントに渡されます。許可サービス・インターフェースでは、コンポーネントは定義されている方法でこの情報を使用する必要はありません。

ComponentData

タイプ: MQBYTE×ComponentDataLength - 入出力

コンポーネント・データ。このデータは、この特定のコンポーネントのためにキュー・マネージャーによって保持されます。このコンポーネントによって提供される関数のいずれかによってそのデータに変更が加えられると、その変更は保存され、次回このコンポーネントの関数の 1 つが呼び出されたときに提供されます。

このデータ域の長さは、キュー・マネージャーにより MQZ_INIT_AUTHORITY 呼び出しの **ComponentDataLength** パラメーターに入れられて渡されます。

Continuation

タイプ: MQLONG - 出力

コンポーネントによって設定される継続標識。指定可能な値は、次のとおりです。

MQZCI_DEFAULT

キュー・マネージャーに依存する継続。

MQZ_CHECK_AUTHORITY の場合、MQZCI_STOP と同じ効果があります。

MQZCI_CONTINUE

次のコンポーネントに継続。

MQZCI_STOP

次のコンポーネントに継続しない。

CompCode

タイプ: MQLONG - 出力

完了コード。値は、次のいずれかでなければなりません。

MQCC_OK

正常終了。

MQCC_FAILED

呼び出し失敗。

理由

タイプ: MQLONG - 出力

CompCode を限定する理由コード。

CompCode が MQCC_OK の場合、次のようになります。

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_WARNING の場合、次のようになります。

MQRC_SERVICE_ERROR

(2289, X'8F1') サービスのアクセスで、予期しないエラーが発生しました。

C 言語での呼び出し

```
MQZ_REFRESH_CACHE (QMgrName, ComponentData,  
                  &Continuation, &CompCode, &Reason);
```

パラメーターを次のように宣言します。

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQBYTE    ComponentData[n];  /* Component data */  
MQLONG    Continuation;      /* Continuation indicator set by  
                             component */  
MQLONG    CompCode;          /* Completion code */  
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

MQZ_SET_AUTHORITY - 権限の設定

この関数は、MQZAS_VERSION_1 許可サービス・コンポーネントに用意されており、キュー・マネージャーにより開始され、エンティティーが持つ指定されたオブジェクトへのアクセス権限を設定します。

この関数の関数 ID (MQZEP 用) は、MQZID_SET_AUTHORITY です。

注: この関数は、既存のすべての権限を指定変更します。既存の権限を維持するには、この関数を使用して再度設定する必要があります。

構文

```
MQZ_SET_AUTHORITY( QMgrName , EntityName , EntityType , ObjectName ,  
                  ObjectType , Authority , ComponentData , Continuation , CompCode , Reason )
```

パラメーター

QMgrName

タイプ: MQCHAR48 - 入力

キュー・マネージャー名。コンポーネントを呼び出すキュー・マネージャーの名前。この名前は、パラメーターがすべて埋まるまで空白が埋め込まれます。名前の最後をヌル文字にすることはできません。

キュー・マネージャー名は、情報としてコンポーネントに渡されます。許可サービス・インターフェースでは、コンポーネントは定義されている方法でこの情報を使用する必要はありません。

EntityName

タイプ: MQCHAR12 - 入力

エンティティ名。オブジェクトに対するアクセス権限を取得するエンティティの名前。ストリングの長さは最大で 12 文字です。12 文字未満の場合、その名前の右側が空白で埋められます。名前の最後をヌル文字にすることはできません。

EntityType

タイプ: MQLONG - 入力

エンティティ・タイプ。 *EntityName* によって指定されるエンティティのタイプ。値は、次のいずれかでなければなりません。

MQZAET_PRINCIPAL

プリンシパル。

MQZAET_GROUP

グループ。

ObjectName

タイプ: MQCHAR48 - 入力

オブジェクト名 アクセスが必要なオブジェクトの名前。ストリングの長さは最大で 48 文字です。48 文字未満の場合、その名前の右側が空白で埋められます。名前の最後をヌル文字にすることはできません。

ObjectType が MQOT_Q_MGR の場合、この名前は *QMgrName* と同じになります。

ObjectType

タイプ: MQLONG - 入力

オブジェクト・タイプ *ObjectName* によって指定されるエンティティのタイプ。値は、次のいずれかでなければなりません。

MQOT_AUTH_INFO

認証情報

MQOT_CHANNEL

チャネル。

MQOT_CLNTCONN_CHANNEL

クライアント接続チャネル。

MQOT_LISTENER

リスナー

MQOT_NAMELIST

名前リスト。

MQOT_PROCESS

プロセス定義。

MQOT_Q

キュー。

MQOT_Q_MGR

キュー・マネージャー。

MQOT_SERVICE

サービス

MQOT_TOPIC

トピック。

権限

タイプ: MQLONG - 入力

エンティティの権限。設定される権限が1つの場合、このフィールドは、その該当する権限操作 (MQZAO_* 定数) に相当します。複数の権限を設定する場合、このフィールドは対応する MQZAO_* 定数をビット単位で OR 演算したものです。

ComponentDatarname>

タイプ: MQBYTEComponentDataLength - 入出力

コンポーネント・データ。このデータは、この特定のコンポーネントのためにキュー・マネージャーで保持されます。このコンポーネントに用意されているいずれかの関数で変更された内容は保持され、これらのコンポーネントのいずれかの関数が次回に呼び出されると提示されます。

このデータ域の長さは、キュー・マネージャーにより MQZ_INIT_AUTHORITY 呼び出しの **ComponentDataLength** パラメーターに入れられて渡されます。

Continuation

タイプ: MQLONG - 出力

コンポーネントによって設定される継続標識。指定可能な値は、次のとおりです。

MQZCI_DEFAULT

キュー・マネージャーに依存する継続。

MQZ_GET_AUTHORITY の場合、MQZCI_CONTINUE と同じ効果があります。

MQZCI_CONTINUE

次のコンポーネントに継続。

MQZCI_STOP

次のコンポーネントに継続しない。

CompCode

タイプ: MQLONG - 出力

完了コード。値は、次のいずれかでなければなりません。

MQCC_OK

正常終了。

MQCC_FAILED

呼び出し失敗。

理由

タイプ: MQLONG - 出力

CompCode を限定する理由コード。

CompCode が MQCC_OK の場合、次のようになります。

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_FAILED の場合、次のようになります。

MQRC_NOT_AUTHORIZED

(2035, X'7F3') アクセスは許可されません。

MQRC_SERVICE_ERROR

(2289, X'8F1') サービスのアクセスで、予期しないエラーが発生しました。

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') 基本サービスを使用できません。

MQRC_UNKNOWN_ENTITY

(2292, X'8F4') サービスに対して不明なエンティティです。

これらの理由コードについて詳しくは、[API 完了コードと理由コード](#)を参照してください。

C 言語での呼び出し

```
MQZ_SET_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName,  
                   ObjectType, Authority, ComponentData,  
                   &Continuation, &CompCode, &Reason);
```

サービスに渡されるパラメーターは次のように宣言されます。

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQCHAR12  EntityName;         /* Entity name */  
MQLONG    EntityType;         /* Entity type */  
MQCHAR48  ObjectName;         /* Object name */  
MQLONG    ObjectType;         /* Object type */  
MQLONG    Authority;          /* Authority to be checked */  
MQBYTE    ComponentData[n];   /* Component data */  
MQLONG    Continuation;       /* Continuation indicator set by  
                               component */  
MQLONG    CompCode;           /* Completion code */  
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

MQZ_SET_AUTHORITY_2 - 権限の設定 (拡張版)

この関数は、MQZAS_VERSION_2 許可サービス・コンポーネントにより提供され、キュー・マネージャーにより開始されて、エンティティーの指定されたオブジェクトに対するアクセス権限を設定します。

この関数の関数 ID (MQZEP 用) は、MQZID_SET_AUTHORITY です。

注: この関数は、既存のすべての権限を指定変更します。既存の権限を維持するには、この関数を使用して再度設定する必要があります。

MQZ_SET_AUTHORITY_2 は MQZ_SET_AUTHORITY と似ていますが、**EntityName** パラメーターが **EntityData** パラメーターで置き換えられています。

構文

```
MQZ_SET_AUTHORITY_2( QMgrName , EntityData , EntityType , ObjectName ,  
                    ObjectType , Authority , ComponentData , Continuation , CompCode , Reason )
```

パラメーター

QMgrName

タイプ: MQCHAR48 - 入力

キュー・マネージャー名。コンポーネントを呼び出すキュー・マネージャーの名前。この名前は、パラメーターがすべて埋まるまでブランクが埋め込まれます。名前の最後をヌル文字にすることはできません。

キュー・マネージャー名は、情報としてコンポーネントに渡されます。許可サービス・インターフェースでは、コンポーネントは定義されている方法でこの情報を使用する必要はありません。

EntityData

タイプ: MQZED - 入力

エンティティー・データ。オブジェクトに対する許可が設定されるエンティティーに関連するデータです。詳細は [1682 ページの『MQZED - エンティティー記述子』](#)を参照してください。

EntityType

タイプ: MQLONG - 入力

エンティティー・タイプ。EntityData によって指定されるエンティティーのタイプ。値は、次のいずれかでなければなりません。

MQZAET_PRINCIPAL

プリンシパル。

MQZAET_GROUP

グループ。

ObjectName

タイプ: MQCHAR48 - 入力

オブジェクト名 エンティティの権限を設定するオブジェクトの名前です。ストリングの長さは最大で 48 文字です。48 文字未満の場合、その名前の右側が空白で埋められます。名前の最後をヌル文字にすることはできません。

ObjectType が MQOT_Q_MGR の場合、この名前は *QMgrName* と同じになります。

ObjectType

タイプ: MQLONG - 入力

オブジェクト・タイプ *ObjectName* によって指定されるエンティティのタイプ。値は、次のいずれかでなければなりません。

MQOT_AUTH_INFO

認証情報

MQOT_CHANNEL

チャンネル。

MQOT_CLNTCONN_CHANNEL

クライアント接続チャンネル。

MQOT_LISTENER

リスナー

MQOT_NAMELIST

名前リスト。

MQOT_PROCESS

プロセス定義。

MQOT_Q

キュー。

MQOT_Q_MGR

キュー・マネージャー。

MQOT_SERVICE

サービス

MQOT_TOPIC

トピック。

権限

タイプ: MQLONG - 入力

エンティティの権限。設定される権限が 1 つの場合、このフィールドは、その該当する権限操作 (MQZAO_* 定数) に相当します。複数の権限を設定する場合、このフィールドは対応する MQZAO_* 定数をビット単位で OR 演算したものです。

ComponentData

タイプ: MQBYTE×ComponentDataLength - 入出力

コンポーネント・データ。このデータは、この特定のコンポーネントのためにキュー・マネージャーで保持されます。このコンポーネントに用意されているいずれかの関数で変更された内容は保持され、これらのコンポーネントのいずれかの関数が次回に呼び出されると提示されます。

このデータ域の長さは、キュー・マネージャーにより MQZ_INIT_AUTHORITY 呼び出しの **ComponentDataLength** パラメーターに入れられて渡されます。

Continuation

タイプ: MQLONG - 出力

コンポーネントによって設定される継続標識。指定可能な値は、次のとおりです。

MQZCI_DEFAULT

キュー・マネージャーに依存する継続。

MQZ_CHECK_AUTHORITY の場合、MQZCI_STOP と同じ効果があります。

MQZCI_CONTINUE

次のコンポーネントに継続。

MQZCI_STOP

次のコンポーネントに継続しない。

CompCode

タイプ: MQLONG - 出力

完了コード。値は、次のいずれかでなければなりません。

MQCC_OK

正常終了。

MQCC_FAILED

呼び出し失敗。

理由

タイプ: MQLONG - 出力

CompCode を限定する理由コード。

CompCode が MQCC_OK の場合、次のようになります。

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_FAILED の場合、次のようになります。

MQRC_NOT_AUTHORIZED

(2035, X'7F3') アクセスは許可されません。

MQRC_SERVICE_ERROR

(2289, X'8F1') サービスのアクセスで、予期しないエラーが発生しました。

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') 基本サービスを使用できません。

MQRC_UNKNOWN_ENTITY

(2292, X'8F4') サービスに対して不明なエンティティです。

これらの理由コードについて詳しくは、[API 完了コードと理由コード](#)を参照してください。

C 言語での呼び出し

```
MQZ_SET_AUTHORITY_2 (QMgrName, &EntityData, EntityType, ObjectName,  
                    ObjectType, Authority, ComponentData,  
                    &Continuation, &CompCode, &Reason);
```

サービスに渡されるパラメーターは次のように宣言されます。

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQZED     EntityData;         /* Entity data */  
MQLONG    EntityType;         /* Entity type */  
MQCHAR48  ObjectName;        /* Object name */  
MQLONG    ObjectType;        /* Object type */  
MQLONG    Authority;         /* Authority to be checked */  
MQBYTE    ComponentData[n];  /* Component data */  
MQLONG    Continuation;      /* Continuation indicator set by  
                             component */  
MQLONG    CompCode;          /* Completion code */  
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

MQZ_TERM_AUTHORITY - 許可サービスの終了

この関数は許可サービス・コンポーネントに用意されています。キュー・マネージャーはこのコンポーネントのサービスが必要なくなった時に、この関数を起動します。この関数はコンポーネントで必要なすべてのクリーンアップを実行する必要があります。

この関数の関数 ID (MQZEP 用) は、MQZID_TERM_AUTHORITY です。

構文

MQZ_TERM_AUTHORITY(Hconfig , Options , QMgrName , ComponentData , CompCode , Reason)

パラメーター

Hconfig

タイプ: MQHCONFIG - 入力

構成ハンドル。このハンドルは、終了する特定のコンポーネントを表します。これは、MQZEP 関数を使用してキュー・マネージャーを呼び出す際に、コンポーネントにより使用されます。

オプション

タイプ: MQLONG - 入力

終了オプション。値は、次のいずれかでなければなりません。

MQZTO_PRIMARY

1 次終了。

MQZTO_SECONDARY

2 次終了。

QMgrName

タイプ: MQCHAR48 - 入力

キュー・マネージャー名。コンポーネントを呼び出すキュー・マネージャーの名前。この名前は、パラメーターがすべて埋まるまでブランクが埋め込まれます。名前の最後をヌル文字にすることはできません。

キュー・マネージャー名は、情報としてコンポーネントに渡されます。許可サービス・インターフェースでは、コンポーネントは定義されている方法でこの情報を使用する必要はありません。

ComponentData

タイプ: MQBYTE x ComponentDataLength - 入出力

コンポーネント・データ。このデータは、この特定のコンポーネントのためにキュー・マネージャーで保持されます。このコンポーネントに用意されているいずれかの関数で変更された内容は保持され、これらのコンポーネントのいずれかの関数が次回に呼び出されると提示されます。

このデータ域の長さは、キュー・マネージャーにより MQZ_INIT_AUTHORITY 呼び出しの ComponentDataLength パラメーター内に渡されます。

MQZ_TERM_AUTHORITY 呼び出しが完了すると、キュー・マネージャーはこのデータを廃棄します。

CompCode

タイプ: MQLONG - 出力

完了コード。値は、次のいずれかでなければなりません。

MQCC_OK

正常終了。

MQCC_FAILED

呼び出し失敗。

理由

タイプ: MQLONG - 出力

CompCode を限定する理由コード。

CompCode が MQCC_OK の場合、次のようになります。

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_FAILED の場合、次のようになります。

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') 基本サービスを使用できません。

MQRC_TERMINATION_FAILED

(2287, X'8FF') 未定義の理由で終了に失敗しました。

これらの理由コードについて詳しくは、[API 完了コードと理由コード](#)を参照してください。

C 言語での呼び出し

```
MQZ_TERM_AUTHORITY (Hconfig, Options, QMgrName, ComponentData,  
                    &CompCode, &Reason);
```

サービスに渡されるパラメーターは次のように宣言されます。

```
MQHCONFIG  Hconfig;          /* Configuration handle */  
MQLONG     Options;          /* Termination options */  
MQCHAR48   QMgrName;         /* Queue manager name */  
MQBYTE     ComponentData[n]; /* Component data */  
MQLONG     CompCode;         /* Completion code */  
MQLONG     Reason;           /* Reason code qualifying CompCode */
```

MQZ_DELETE_NAME - 名前の削除

この関数は、ネーム・サービス・コンポーネントにより提供され、キュー・マネージャーにより開始されて、指定されたキューのエントリーを削除します。

この関数の関数 ID (MQZEP 用) は、MQZID_DELETE_NAME です。

構文

`MQZ_DELETE_NAME( QMgrName , QName , ComponentData , Continuation , CompCode , Reason )`

パラメーター

QMgrName

タイプ: MQCHAR48 - 入力

キュー・マネージャー名。コンポーネントを呼び出すキュー・マネージャーの名前。この名前は、パラメーターがすべて埋まるまでブランクが埋め込まれます。名前の最後をヌル文字にすることはできません。

キュー・マネージャー名は、情報としてコンポーネントに渡されます。許可サービス・インターフェースでは、コンポーネントは定義されている方法でこの情報を使用する必要はありません。

QName

タイプ: MQCHAR48 - 入力

キュー名。項目が削除されるキューの名前。この名前は、パラメーターがすべて埋まるまでブランクが埋め込まれます。名前の最後をヌル文字にすることはできません。

ComponentData

タイプ: MQBYTE x ComponentDataLength - 入出力

コンポーネント・データ。このデータは、この特定のコンポーネントのためにキュー・マネージャーで保持されます。このコンポーネントに用意されているいずれかの関数で変更された内容は保持され、これらのコンポーネントのいずれかの関数が次回に呼び出されると提示されます。

このデータ域の長さは、キュー・マネージャーにより MQZ_INIT_NAME 呼び出しの ComponentDataLength パラメーターに入れて渡されます。

Continuation

タイプ: MQLONG - 出力

コンポーネントによって設定される継続標識。値は、次のいずれかでなければなりません。

MQZCI_DEFAULT

キュー・マネージャーに依存する継続。

MQZCI_STOP

次のコンポーネントに継続しない。

MQZ_DELETE_NAME コマンドでは、**Continuation** パラメーターに何が返されるかに関係なく、キュー・マネージャーは別のコンポーネントの起動を試行しません。

CompCode

タイプ: MQLONG - 出力

完了コード。値は、次のいずれかでなければなりません。

MQCC_OK

正常終了。

MQCC_WARNING

警告 (部分完了)。

MQCC_FAILED

呼び出し失敗。

理由

タイプ: MQLONG - 出力

CompCode を限定する理由コード。

CompCode が MQCC_OK の場合、次のようになります。

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_WARNING の場合、次のようになります。

MQRC_UNKNOWN_NAME

(2288, X'8F0') キュー名が見つかりません。

注: 基礎にあるサービスがこのケースに成功の応答を戻した場合、このコードを戻せないことがあります。

CompCode が MQCC_FAILED の場合、次のようになります。

MQRC_SERVICE_ERROR

(2289, X'8F1') サービスのアクセスで、予期しないエラーが発生しました。

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') 基本サービスを使用できません。

これらの理由コードについて詳しくは、[API 完了コードと理由コード](#)を参照してください。

C 言語での呼び出し

```
MQZ_DELETE_NAME (QMgrName, QName, ComponentData, &Continuation,  
                 &CompCode, &Reason);
```

サービスに渡されるパラメーターは次のように宣言されます。

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQCHAR48  QName;              /* Queue name */  
MQBYTE    ComponentData[n];   /* Component data */  
MQLONG     Continuation;       /* Continuation indicator set by  
                               component */  
MQLONG     CompCode;           /* Completion code */  
MQLONG     Reason;            /* Reason code qualifying CompCode */
```

MQZ_INIT_NAME - ネーム・サービスの初期化

この関数は、ネーム・サービス・コンポーネントにより提供され、コンポーネントの構成中にキュー・マネージャーにより開始されます。キュー・マネージャーに情報を提供するために、MQZEP を呼び出すことが想定されます。

この関数の関数 ID (MQZEP 用) は、MQZID_INIT_NAME です。

構文

MQZ_INIT_NAME(Hconfig , Options , QMgrName , ComponentDataLength ,
ComponentData , Version , CompCode , Reason)

パラメーター

Hconfig

タイプ: MQHCONFIG - 入力

構成ハンドル。このハンドルは、初期化される特定のコンポーネントを表します。これは、MQZEP 関数を使用してキュー・マネージャーを呼び出す際に、コンポーネントにより使用されます。

オプション

タイプ: MQLONG - 入力

初期化オプション。値は、次のいずれかでなければなりません。

MQZIO_PRIMARY

1 次初期化。

MQZIO_SECONDARY

2 次初期化。

QMgrName

タイプ: MQCHAR48 - 入力

キュー・マネージャー名。コンポーネントを呼び出すキュー・マネージャーの名前。この名前は、パラメーターがすべて埋まるまでブランクが埋め込まれます。名前の最後をヌル文字にすることはできません。

キュー・マネージャー名は、情報としてコンポーネントに渡されます。許可サービス・インターフェースでは、コンポーネントは定義されている方法でこの情報を使用する必要はありません。

ComponentDataLength

タイプ: MQLONG - 入力

コンポーネント・データの長さ。ComponentData 域の長さ (バイト単位)。この長さは、コンポーネント構成データに定義されます。

ComponentData

タイプ: MQBYTE x ComponentDataLength - 入出力

コンポーネント・データ。コンポーネントの 1 次初期化関数が呼び出される前に、すべてゼロに初期化されます。このデータは、この特定のコンポーネントの代わりにキュー・マネージャーが保持します。このコンポーネントの提供するいずれかの関数 (初期化関数を含む) によって変更された内容がすべて保持され、それらのコンポーネント関数の 1 つが次回に呼び出されるときに提示されます。

このデータ域の長さは、キュー・マネージャーにより MQZ_INIT_AUTHORITY 呼び出しの ComponentDataLength パラメーターに入れられて渡されます。

バージョン

タイプ: MQLONG - 入出力

バージョン番号。初期化関数への入力時に、これによりそのキュー・マネージャーがサポートする最新のバージョン番号を識別します。初期化関数は、必要により、キュー・マネージャーがサポートするインターフェースのバージョンに変更する必要があります。キュー・マネージャーがコンポーネントから返されるバージョンをサポートしないことが返されると、コンポーネントの MQZ_TERM_NAME 関数を呼び出し、その後この関数を使用しません。

次の値がサポートされます。

MQZAS_VERSION_1

バージョン 1。

CompCode

タイプ: MQLONG - 出力

完了コード。値は、次のいずれかでなければなりません。

MQCC_OK

正常終了。

MQCC_FAILED

呼び出し失敗。

理由

タイプ: MQLONG - 出力

CompCode を限定する理由コード。

CompCode が MQCC_OK の場合、次のようになります。

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_FAILED の場合、次のようになります。

MQRC_INITIALIZATION_FAILED

(2286, X'8EE') 未定義の理由で初期化に失敗しました。

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') 基本サービスを使用できません。

これらの理由コードについて詳しくは、[API 完了コードと理由コード](#)を参照してください。

C 言語での呼び出し

```
MQZ_INIT_NAME (Hconfig, Options, QMgrName, ComponentDataLength,  
                ComponentData, &Version, &CompCode, &Reason);
```

サービスに渡されるパラメーターは次のように宣言されます。

```
MQHCONFIG    Hconfig;           /* Configuration handle */  
MQLONG       Options;           /* Initialization options */  
MQCHAR48     QMgrName;          /* Queue manager name */  
MQLONG       ComponentDataLength; /* Length of component data */  
MQBYTE       ComponentData[n];  /* Component data */  
MQLONG       Version;           /* Version number */  
MQLONG       CompCode;          /* Completion code */  
MQLONG       Reason;            /* Reason code qualifying CompCode */
```

MQZ_INSERT_NAME - 名前の挿入

この関数は、ネーム・サービス・コンポーネントに用意されています。キュー・マネージャーはこの関数を開始して、指定したキューの項目(キューを所有しているキュー・マネージャーの名前を含む)を挿入します。キューがすでにサービス内で定義されている場合、呼び出しは失敗します。

この関数の関数 ID (MQZEP 用) は、MQZID_INSERT_NAME です。

構文

```
MQZ_INSERT_NAME( QMgrName , QName , ResolvedQMgrName , ComponentData ,  
                 Continuation , CompCode , Reason )
```

パラメーター

QMgrName

タイプ: MQCHAR48 - 入力

キュー・マネージャー名。コンポーネントを呼び出すキュー・マネージャーの名前。この名前は、パラメーターがすべて埋まるまで空白が埋め込まれます。名前の最後をヌル文字にすることはできません。

キュー・マネージャー名は、情報としてコンポーネントに渡されます。許可サービス・インターフェースでは、コンポーネントは定義されている方法でこの情報を使用する必要はありません。

QName

タイプ: MQCHAR48 - 入力

キュー名。項目が挿入されるキューの名前。この名前は、パラメーターがすべて埋まるまで空白が埋め込まれます。名前の最後をヌル文字にすることはできません。

ResolvedQMgrName

タイプ: MQCHAR48 - 入力

解決済みのキュー・マネージャーの名前。キューが解決するキュー・マネージャーの名前。この名前は、パラメーターがすべて埋まるまで空白が埋め込まれます。名前の最後をヌル文字にすることはできません。

ComponentData

タイプ: MQBYTE×ComponentDataLength - 入出力

コンポーネント・データ。このデータは、この特定のコンポーネントの代わりにキュー・マネージャーが保持します。このコンポーネントの提供するいずれかの関数 (初期化関数を含む) によって変更された内容がすべて保持され、それらのコンポーネント関数の 1 つが次回に呼び出されるときに提示されます。

このデータ域の長さは、キュー・マネージャーにより MQZ_INIT_NAME 呼び出しの **ComponentDataLength** パラメーターに入れられて渡されます。

Continuation

タイプ: MQLONG - 入出力

コンポーネントによって設定される継続標識。MQZ_INSERT_NAME では、**Continuation** パラメーターに何が返されるかにかかわらず、キュー・マネージャーは別のコンポーネントの起動を試行しません。

次の値がサポートされます。

MQZCI_DEFAULT

キュー・マネージャーに依存する継続。

MQZCI_STOP

次のコンポーネントに継続しない。

CompCode

タイプ: MQLONG - 出力

完了コード。値は、次のいずれかでなければなりません。

MQCC_OK

正常終了。

MQCC_FAILED

呼び出し失敗。

理由

タイプ: MQLONG - 出力

CompCode を限定する理由コード。

CompCode が MQCC_OK の場合、次のようになります。

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_FAILED の場合、次のようになります。

MQRC_Q_ALREADY_EXISTS

(2290, X'8F2') キュー・オブジェクトがすでに存在します。

MQRC_SERVICE_ERROR

(2289, X'8F1') サービスのアクセスで、予期しないエラーが発生しました。

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') 基本サービスを使用できません。

これらの理由コードについて詳しくは、[API 完了コードと理由コード](#)を参照してください。

C 言語での呼び出し

```
MQZ_INSERT_NAME (QMgrName, QName, ResolvedQMgrName, ComponentData,  
                 &Continuation, &CompCode, &Reason);
```

サービスに渡されるパラメーターは次のように宣言されます。

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQCHAR48  QName;              /* Queue name */  
MQCHAR48  ResolvedQMgrName;   /* Resolved queue manager name */  
MQBYTE    ComponentData[n];   /* Component data */  
MQLONG    Continuation;       /* Continuation indicator set by  
                               component */  
MQLONG    CompCode;           /* Completion code */  
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

MQZ_LOOKUP_NAME - 名前の検索

この関数は、ネーム・サービス・コンポーネントに用意されています。キュー・マネージャーはこの関数を開始し、指定されたキューに関して、所有しているキュー・マネージャーの名前を検索します。

この関数の関数 ID (MQZEP 用) は、MQZID_LOOKUP_NAME です。

構文

```
MQZ_LOOKUP_NAME( QMgrName , QName , ResolvedQMgrName , ComponentData ,  
Continuation , CompCode , Reason )
```

パラメーター

QMgrName

タイプ: MQCHAR48 - 入力

キュー・マネージャー名。コンポーネントを呼び出すキュー・マネージャーの名前。この名前は、パラメーターがすべて埋まるまでブランクが埋め込まれます。名前の最後をヌル文字にすることはできません。

キュー・マネージャー名は、情報としてコンポーネントに渡されます。許可サービス・インターフェースでは、コンポーネントは定義されている方法でこの情報を使用する必要はありません。

QName

タイプ: MQCHAR48 - 入力

キュー名。エントリーを解決するキューの名前。この名前は、パラメーターがすべて埋まるまでブランクが埋め込まれます。名前の最後をヌル文字にすることはできません。

ResolvedQMgrName

タイプ: MQCHAR48 - 出力

解決済みのキュー・マネージャーの名前。関数が正常に完了した場合、これはキューを所有するキュー・マネージャーの名前です。

サービス・コンポーネントによって戻される名前は、パラメーターの右側がすべて埋まるまで空白を埋め込む必要があります。名前の最後をヌル文字にしたり、先頭や途中に空白を入れることはできません。

ComponentData

タイプ: MQBYTEExComponentDataLength - 入出力

コンポーネント・データ。このデータは、この特定のコンポーネントの代わりにキュー・マネージャーが保持します。このコンポーネントの提供するいずれかの関数 (初期化関数を含む) によって変更された内容がすべて保持され、それらのコンポーネント関数の 1 つが次回に呼び出されるときに提示されます。

このデータ域の長さは、キュー・マネージャーにより MQZ_INIT_NAME 呼び出しの **ComponentDataLength** パラメーターに入れられて渡されます。

Continuation

タイプ: MQLONG - 出力

コンポーネントによって設定される継続標識。MQZ_LOOKUP_NAME の場合、キュー・マネージャーは以下のようにして、別のネーム・サービス・コンポーネントを開始するかどうかを指定します。

- *CompCode* が MQCC_OK の場合、*Continuation* で戻される値に関係なく、さらにコンポーネントは開始されません。
- *CompCode* が MQCC_OK ではない場合、*Continuation* が MQZCI_STOP でない限り、さらにコンポーネントが開始されます。

次の値がサポートされます。

MQZCI_DEFAULT

キュー・マネージャーに依存する継続。

MQZCI_CONTINUE

次のコンポーネントに継続。

MQZCI_STOP

次のコンポーネントに継続しない。

CompCode

タイプ: MQLONG - 出力

完了コード。値は、次のいずれかでなければなりません。

MQCC_OK

正常終了。

MQCC_FAILED

呼び出し失敗。

理由

タイプ: MQLONG - 出力

CompCode を限定する理由コード。

CompCode が MQCC_OK の場合、次のようになります。

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_FAILED の場合、次のようになります。

MQRC_SERVICE_ERROR

(2289, X'8F1') サービスのアクセスで、予期しないエラーが発生しました。

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') 基本サービスを使用できません。

MQRC_UNKNOWN_Q_NAME

(2288, X'8F0') キュー名が見つかりません。

これらの理由コードについて詳しくは、[API 完了コードと理由コード](#)を参照してください。

C 言語での呼び出し

```
MQZ_LOOKUP_NAME (QMgrName, QName, ResolvedQMgrName, ComponentData,  
                  &Continuation, &CompCode, &Reason);
```

サービスに渡されるパラメーターは次のように宣言されます。

```
MQCHAR48  QMgrName;          /* Queue manager name */  
MQCHAR48  QName;             /* Queue name */  
MQCHAR48  ResolvedQMgrName;   /* Resolved queue manager name */  
MQBYTE    ComponentData[n];   /* Component data */  
MQLONG     Continuation;      /* Continuation indicator set by  
                               component */  
MQLONG     CompCode;          /* Completion code */  
MQLONG     Reason;           /* Reason code qualifying CompCode */
```

MQZ_TERM_NAME - ネーム・サービスの終了

この関数はネーム・サービス・コンポーネントに用意されています。キュー・マネージャーはこのコンポーネントのサービスが必要なくなった時に、この関数を開始します。この関数はコンポーネントに必要なすべてのクリーンアップを実行する必要があります。

この関数の関数 ID (MQZEP 用) は、MQZID_TERM_NAME です。

構文

```
MQZ_TERM_NAME( Hconfig , Options , QMgrName , ComponentData , CompCode ,  
               Reason )
```

パラメーター

Hconfig

タイプ: MQHCONFIG - 入力

構成ハンドル。このハンドルは、終了する特定のコンポーネントを表します。これは、MQZEP 関数を使用してキュー・マネージャーを呼び出す際に、コンポーネントで使用されます。

オプション

タイプ: MQLONG - 入力

終了オプション。値は、次のいずれかでなければなりません。

MQZTO_PRIMARY

1 次終了。

MQZTO_SECONDARY

2 次終了。

QMgrName

タイプ: MQCHAR48 - 入力

キュー・マネージャー名。コンポーネントを呼び出すキュー・マネージャーの名前。この名前は、パラメーターがすべて埋まるまでブランクが埋め込まれます。名前の最後をヌル文字にすることはできません。

キュー・マネージャー名は、情報としてコンポーネントに渡されます。許可サービス・インターフェースでは、コンポーネントは定義されている方法でこの情報を使用する必要はありません。

ComponentData

タイプ: MQBYTE x ComponentDataLength - 入出力

コンポーネント・データ。このデータは、この特定のコンポーネントの代わりにキュー・マネージャーが保持します。このコンポーネントの提供するいずれかの関数 (初期化関数を含む) によって変更された内容がすべて保持され、それらのコンポーネント関数の 1 つが次回に呼び出されるときに提示されます。

コンポーネント・データは、すべてのプロセスがアクセス可能な共有メモリー内に入れられます。

このデータ域の長さは、キュー・マネージャーにより MQZ_INIT_NAME 呼び出しの **ComponentDataLength** パラメーターに入れられて渡されます。

MQZ_TERM_NAME 呼び出しが完了すると、キュー・マネージャーはこのデータを廃棄します。

CompCode

タイプ: MQLONG - 出力

完了コード。値は、次のいずれかでなければなりません。

MQCC_OK

正常終了。

MQCC_FAILED

呼び出し失敗。

理由

タイプ: MQLONG - 出力

CompCode を限定する理由コード。

CompCode が MQCC_OK の場合、次のようになります。

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_FAILED の場合、次のようになります。

MQRC_TERMINATION_FAILED

(2287, X'8FF') 未定義の理由で終了に失敗しました。

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') 基本サービスを使用できません。

これらの理由コードについて詳しくは、[API 完了コードと理由コード](#)を参照してください。

C 言語での呼び出し

```
MQZ_TERM_NAME (Hconfig, Options, QMgrName, ComponentData, &CompCode,  
               &Reason);
```

サービスに渡されるパラメーターは次のように宣言されます。

```
MQHCONFIG  Hconfig;           /* Configuration handle */  
MQLONG     Options;           /* Termination options */  
MQCHAR48   QMgrName;         /* Queue manager name */  
MQBYTE     ComponentData[n]; /* Component data */  
MQLONG     CompCode;          /* Completion code */  
MQLONG     Reason;            /* Reason code qualifying CompCode */
```

MQZAC - アプリケーション・コンテキスト

MQZAC 構造体は、MQZ_AUTHENTICATE_USER 呼び出しの *ApplicationContext* パラメーターに使用します。このパラメーターは、呼び出し側アプリケーションに関連するデータを指定します。

表 1 に、この構造体の各フィールドを要約します。

表 874. MQZAC のフィールド	
フィールド	説明
StrucId	構造体 ID
バージョン	構造体のバージョン番号
ProcessId	プロセス ID
ThreadId	スレッド ID

表 874. MQZAC のフィールド (続き)	
フィールド	説明
<u>ApplName</u>	アプリケーション名
<u>UserID</u>	ユーザー ID
<u>EffectiveUserID</u>	有効ユーザー ID
<u>環境</u>	Environment
<u>CallerType</u>	呼び出し元のタイプ
<u>AuthenticationType</u>	認証のタイプ
<u>BindType</u>	バインドのタイプ

フィールド

StrucId

タイプ: MQCHAR4 - 入力

構造体 ID 値は次のとおりです。

MQZAC_STRUC_ID

アプリケーション・コンテキスト構造体の ID。

C 言語の場合、定数 MQZAC_STRUC_ID_ARRAY も定義されます。これは MQZAC_STRUC_ID と同じ値ですが、ストリングではなく文字の配列です。

バージョン

タイプ: MQLONG - 入力

構造体のバージョン番号。値は次のとおりです。

MQZAC_VERSION_1

バージョン 1 のアプリケーション・コンテキスト構造体。定数 MQZAC_CURRENT_VERSION は、現行のバージョン番号を指定しています。

ProcessId

タイプ: MQPID - 入力

アプリケーションのプロセス ID。

ThreadId

タイプ: MQTID - 入力

アプリケーションのスレッド ID。

ApplName

タイプ: MQCHAR28 - 入力

アプリケーション名。

UserID

タイプ: MQCHAR12 - 入力

ユーザー ID。AIX and Linux では、このフィールドにアプリケーションの実ユーザー ID を指定します。Windows では、このフィールドにアプリケーションのユーザー ID を指定します。

EffectiveUserID

タイプ: MQCHAR12 - 入力

有効ユーザー ID。AIX and Linux では、このフィールドにアプリケーションの有効ユーザー ID を指定します。On Windows では、このフィールドはブランクです。

Environment

タイプ: MQLONG - 入力

環境。このフィールドで、呼び出しの実行元の環境を指定します。このフィールドの値は以下のいずれかです。

MQXE_COMMAND_SERVER

コマンド・サーバー

MQXE_MQSC

runmqsc コマンド・インタープリター

MQXE_MCA

メッセージ・チャンネル・エージェント

MQXE_OTHER

未定義の環境

CallerType

タイプ: MQLONG - 入力

呼び出し元のタイプ。このフィールドで、呼び出しの実行元プログラムのタイプを指定します。このフィールドの値は以下のいずれかです。

MQXACT_EXTERNAL

呼び出しはキュー・マネージャーの外部です。

MQXACT_INTERNAL

呼び出しはキュー・マネージャーの内部です。

AuthenticationType

タイプ: MQLONG - 入力

認証のタイプ。このフィールドで、実行される認証のタイプを指定します。このフィールドの値は以下のいずれかです。

MQZAT_INITIAL_CONTEXT

認証呼び出しは、ユーザー・コンテキストが初期化されることに起因します。この値は、MQCONN または MQCONNX 呼び出しの間に使用されます。

MQZAT_CHANGE_CONTEXT

この認証呼び出しは、ユーザー・コンテキストが変更されることに起因します。この値は、MCA がユーザー・コンテキストを変更するときに使用されます。親トピック: MQZAC -

BindType

タイプ: MQLONG - 入力

バインドのタイプ。このフィールドは、使用中のバインディング・タイプを指定します。このフィールドの値は以下のいずれかです。

MQCNO_FASTPATH_BINDING

ファースト・パス・バインディング。

MQCNO_SHARED_BINDING

共有バインディング。

MQCNO_ISOLATED_BINDING

分離されたバインディング。

C 宣言

構造体のフィールドを次のように宣言します。

```
typedef struct tagMQZAC MQZAC;
struct tagMQZAC {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQPID      ProcessId;        /* Process identifier */
    MQTID      ThreadId;         /* Thread identifier */
    MQCHAR28   ApplName;         /* Application name */
    MQCHAR12   UserID;           /* User identifier */
    MQCHAR12   EffectiveUserID;   /* Effective user identifier */
    MQLONG     Environment;      /* Environment */
    MQLONG     CallerType;       /* Caller type */
    MQLONG     AuthenticationType; /* Authentication type */
}
```

```

    MQLONG    BindType;          /* Bind type */
};

```

MQZAD - 権限データ

MQZAD 構造体は、MQZ_ENUMERATE_AUTHORITY_DATA 呼び出しの 2 つのパラメーター (1 つは入力で、もう 1 つは出力) に使用します。

Filter および **AuthorityBuffer** パラメーターについては、[1641 ページの『MQZ_ENUMERATE_AUTHORITY_DATA - 権限データの列挙』](#)を参照してください。

- MQZAD は、呼び出しへの入力である **Filter** パラメーターで使用されます。このパラメーターは、この呼び出しによって戻される権限データの選択で使用される選択基準を指定します。
- MQZAD は、呼び出しからの出力である **AuthorityBuffer** パラメーターでも使用されます。このパラメーターでは、プロファイル名、オブジェクト・タイプ、およびエンティティの 1 つの組み合わせの許可を指定します。

表 1. 構造内のフィールドを要約します。

表 875. MQZAD のフィールド	
フィールド	説明
StrucId	構造体 ID
バージョン	構造体のバージョン番号
ProfileName	プロファイル名
ObjectType	オブジェクト・タイプ
Authority	権限
EntityDataPtr	エンティティ・データへのポインター
EntityType	エンティティ・タイプ
オプション	オプション

フィールド

StrucId

タイプ: MQCHAR4 - 入力

構造体 ID 値は次のとおりです。

MQZAD_STRUC_ID

権限データ構造体の ID。

C 言語の場合、定数 MQZAD_STRUC_ID_ARRAY も定義されます。これは MQZAD_STRUC_ID と同じ値ですが、ストリングではなく文字の配列です。

バージョン

タイプ: MQLONG - 入力

構造体のバージョン番号。値は次のとおりです。

MQZAD_VERSION_1

バージョン 1 のアプリケーション・コンテキスト構造体。定数 MQZAD_CURRENT_VERSION は、現行バージョンのバージョン番号を指定します。

以下の定数は、現行バージョンのバージョン番号を指定しています。

MQZAD_CURRENT_VERSION

権限データ構造体の現行バージョン。

ProfileName

タイプ: MQCHAR48 - 入力

プロファイル名。

Filter パラメーターの場合、このフィールドは権限データを必要とするプロファイル名です。名前がフィールドの最後までまたは最初のヌル文字までブランクの場合、すべてのプロファイル名の権限データが返されます。

AuthorityBuffer パラメーターの場合、このフィールドは、指定された選択基準と突き合わせるプロファイルの名前です。

ObjectType

タイプ: MQLONG - 入力

オブジェクト・タイプ

Filter パラメーターの場合、このフィールドは、権限データを必要とするオブジェクト・タイプです。値が MQOT_ALL の場合、すべてのオブジェクト・タイプの権限データが返されます。

AuthorityBuffer パラメーターの場合、このフィールドは、**ProfileName** パラメーターで識別されるプロファイルを適用するオブジェクト・タイプです。

値は以下のいずれかです。**Filter** パラメーターの場合、値 MQOT_ALL も有効です。

MQOT_AUTH_INFO

認証情報

MQOT_CHANNEL

チャンネル

MQOT_CLNTCONN_CHANNEL

クライアント接続チャンネル

MQOT_LISTENER

リスナー

MQOT_NAMELIST

名前リスト

MQOT_PROCESS

プロセス定義

MQOT_Q

キュー

MQOT_Q_MGR

キュー・マネージャー

MQOT_SERVICE

サービス

権限

タイプ: MQLONG - 入力

権限。

Filter パラメーターの場合、このフィールドは無視されます。

AuthorityBuffer パラメーターの場合、このフィールドは **ProfileName** および **ObjectType** で識別されるオブジェクトに対してエンティティーが保持している許可を表します。エンティティーの権限が1つのみである場合、このフィールドは該当する許可値 (MQZAO_* 定数) に等しくなります。エンティティーが複数の権限を持つ場合、このフィールドは対応する MQZAO_* 定数のビット単位 OR になります。

EntityDataPtr

タイプ: PMQZED - 入力

エンティティーを識別する MQZED 構造体のアドレス。

Filter パラメーターの場合、このフィールドは、権限データを必要とするエンティティを識別する MQZED 構造体を指します。 **EntityDataPtr** がヌル・ポインターの場合、すべてのエンティティの権限データが返されます。

AuthorityBuffer パラメーターの場合、このフィールドは、権限データが戻されたエンティティを識別する MQZED 構造体を指します。

EntityType

タイプ: MQLONG - 入力

エンティティ・タイプ。

Filter パラメーターの場合、このフィールドは、権限データを必要とするエンティティ・タイプを指定します。 値が MQZAET_NONE の場合、すべてのエンティティ・タイプの権限データが戻されます。

AuthorityBuffer パラメーターの場合、このフィールドは、 **EntityDataPtr** パラメーターが指している MQZED 構造体によって識別されるエンティティのタイプを指定します。

値は以下のいずれかです。 **Filter** パラメーターの場合、値 MQZAET_NONE も有効です。

MQZAET_PRINCIPAL

プリンシパル

MQZAET_GROUP

グループ

オプション

タイプ: MQAUTHOPT - 入力

オプション。 このフィールドは、表示されるプロファイルに制御を与えるオプションを指定します。 以下のいずれかの値を指定する必要があります。

MQAUTHOPT_NAME_ALL_MATCHING

すべてのプロファイルを表示します。

MQAUTHOPT_NAME_EXPLICIT

ProfileName フィールドで指定されたものと完全に一致する名前を持つプロファイルを表示します。

さらに、次のいずれか 1 つも指定する必要があります。

MQAUTHOPT_ENTITY_SET

ProfileName パラメーターで指定されたオブジェクトに対してエンティティが保持する累積の権限を計算するときに使用されるすべてのプロファイルを表示します。 **ProfileName** パラメーターにワイルドカード文字を含めることはできません。

- 指定されたエンティティがプリンシパルの場合、セット {エンティティ、グループ} のそれぞれのメンバーごとに、オブジェクトに適合する最も適当なプロファイルが表示されます。
- 指定されたエンティティがグループの場合、オブジェクトに適合するグループの中から、最も適当なプロファイルが表示されます。
- この値を指定する場合、 **ProfileName**、 **ObjectType**、 **EntityType**、 および **EntityDataPtr** MQZED 構造体に指定するエンティティ名の値は、すべて非ブランクでなければなりません。

MQAUTHOPT_NAME_ALL_MATCHING を指定した場合は、次の値も指定できます。

MQAUTHOPT_ENTITY_EXPLICIT

EntityDataPtr MQZED 構造体に指定されたエンティティ名と完全に一致するエンティティ名を持つプロファイルを表示します。

C 宣言

```
typedef struct tagMQZAD MQZAD;  
struct tagMQZAD {  
    MQCHAR4    StrucId;        /* Structure identifier */  
    MQLONG     Version;        /* Structure version number */  
};
```

```

MQCHAR48  ProfileName;    /* Profile name */
MQLONG    ObjectType;     /* Object type */
MQLONG    Authority;      /* Authority */
PMQZED    EntityDataPtr;  /* Address of MQZED structure identifying an
                           entity */
MQLONG    EntityType;     /* Entity type */
MQAUTHOPT Options;       /* Options */
};

```

MQZED - エンティティ記述子

MQZED 構造体は、いくつかの許可サービス呼び出しの中で、許可を検査するエンティティを指定するために使用されます。

表 1. は構造体のフィールドを要約している。

表 876. MQZED のフィールド	
フィールド	説明
StrucId	構造体 ID
バージョン	バージョン
EntityName Ptr	エンティティ名
EntityDomainPtr	エンティティ・ドメイン・ポインター
SecurityId	セキュリティ ID
CorrelationPtr	関連ポインター

フィールド

StrucId

タイプ: MQCHAR4 - 入力

構造体 ID 値は次のとおりです。

MQZED_STRUC_ID

エンティティ記述子構造体の ID。

C 言語の場合、定数 MQZED_STRUC_ID_ARRAY も定義されます。これは MQZED_STRUC_ID と同じ値ですが、ストリングではなく文字の配列です。

バージョン

タイプ: MQLONG - 入力

構造体のバージョン番号。値は次のとおりです。

MQZED_VERSION_1

バージョン 1 エンティティ記述子構造体

以下の定数は、現行バージョンのバージョン番号を指定しています。

MQZED_CURRENT_VERSION

エンティティ記述子構造体の現行バージョン。

EntityNamePtr

タイプ: PMQCHAR - 入力

プロファイル名。

エンティティ名のアドレス。これは、許可が検査されるエンティティの名前を指すポインターです。

EntityDomainPtr

タイプ: PMQCHAR - 入力

エンティティ・ドメイン・ネームのアドレス。これは、許可が検査されるエンティティの定義を含むドメインの名前を指すポインターです。

SecurityId

タイプ: MQBYTE40 - 入力

権限。

セキュリティ ID。これは、許可が検査されるセキュリティ ID です。

CorrelationPtr

タイプ: MQPTR - 入力

相関ポインター。これを使用すると、認証ユーザー関数とその他の適切な OAM 関数との間で相関データを引き渡しやすくなります。

C 宣言

```
typedef struct tagMQZED MQZED;  
struct tagMQZED {  
    MQCHAR4    StrucId;           /* Structure identifier */  
    MQLONG     Version;          /* Structure version number */  
    PMQCHAR    EntityNamePtr;    /* Address of entity name */  
    PMQCHAR    EntityDomainPtr;  /* Address of entity domain name */  
    MQBYTE40   SecurityId;       /* Security identifier */  
    MQPTR      CorrelationPtr;   /* Address of correlation data */  
};
```

MQZEP - コンポーネントのエントリー・ポイントの追加

サービス・コンポーネントでは、この関数を初期化時に開始して、そのサービス・コンポーネントのエントリー・ポイント・ベクトルにエントリー・ポイントを追加します。

構文

MQZEP (*Hconfig*, *Function*, *EntryPoint*, *CompCode*, *Reason*)

パラメーター

Hconfig

タイプ: MQHCONFIG - 入力

構成ハンドル。このハンドルは、この特定のインストール可能サービスに構成されるコンポーネントを表します。このコンポーネントは、コンポーネント初期化呼び出し時にキュー・マネージャーによってコンポーネント構成関数に渡されるコンポーネントと同じである必要があります。

Function

タイプ: MQLONG - 入力

関数 ID。これについて有効な値は、インストール可能サービスごとに定義されます。

同じ関数に対して MQZEP が複数回呼び出されると、最後の呼び出しでのエントリー・ポイントが使用されます。

EntryPoint

タイプ: PMQFUNC - 入力

関数エントリー・ポイント。これは、関数を実行するためにコンポーネントで用意されたエントリー・ポイントのアドレスです。

NULL 値は有効で、このコンポーネントで関数は用意されないことを示します。MQZEP を使用して定義されていないエントリー・ポイントには、NULL が想定されます。

CompCode

タイプ: MQLONG - 出力

完了コード。値は、次のいずれかでなければなりません。

MQCC_OK

正常終了。

MQCC_FAILED

呼び出し失敗。

理由

タイプ: MQLONG - 出力

CompCode を限定する理由コード。

CompCode が MQCC_OK の場合、次のようになります。

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_FAILED の場合、次のようになります。

MQRC_FUNCTION_ERROR

(2281, X'8E9') 関数 ID が無効です。

MQRC_HCONFIG_ERROR

(2280, X'8E8') 構成ハンドルが無効です。

これらの理由コードについて詳しくは、[API 完了コードと理由コード](#)を参照してください。

C 言語での呼び出し

```
MQZEP (Hconfig, Function, EntryPoint, &CompCode, &Reason);
```

パラメーターを次のように宣言します。

```
MQHCONFIG  Hconfig;      /* Configuration handle */
MQLONG     Function;     /* Function identifier */
PMQFUNC    EntryPoint;   /* Function entry point */
MQLONG     CompCode;     /* Completion code */
MQLONG     Reason;       /* Reason code qualifying CompCode */
```

MQZFP - 解放パラメーター

MQZFP 構造体は、MQZ_FREE_USER 呼び出しの *FreeParms* パラメーターに使用します。このパラメーターは、解放されるリソースに関連するデータを指定します。

表 1. は構造体のフィールドを要約している。

表 877. MQZFP のフィールド	
フィールド	説明
StrucId	構造体 ID
バージョン	バージョン
Reserved	予約フィールド
CorrelationPtr	相関ポインター

フィールド

StrucId

タイプ: MQCHAR4 - 入力

構造体 ID 値は次のとおりです。

MQZIC_STRUC_ID

ID コンテキスト構造体の ID。C 言語の場合、定数 MQZIC_STRUC_ID_ARRAY も定義されます。これは MQZIC_STRUC_ID と同じ値ですが、文字列ではなく文字の配列です。

バージョン

タイプ: MQLONG - 入力

構造体のバージョン番号。値は次のとおりです。

MQZFP_VERSION_1

バージョン 1 解放パラメーター構造体。

以下の定数は、現行バージョンのバージョン番号を指定しています。

MQZFP_CURRENT_VERSION

解放パラメーター構造体の現行バージョン。

Reserved

タイプ: MQBYTE8 - 入力

予約フィールド。初期値はヌルです。

CorrelationPtr

タイプ: MQPTR - 入力

相関ポインター。解放されるリソースに関連する相関データのアドレス。

C 宣言

```
typedef struct tagMQZFP MQZFP;  
struct tagMQZFP {  
    MQCHAR4    StrucId;           /* Structure identifier */  
    MQLONG     Version;          /* Structure version number */  
    MQBYTE8    Reserved;         /* Reserved field */  
    MQPTR      CorrelationPtr;   /* Address of correlation data */  
};
```

MQZIC - ID コンテキスト

MQZIC 構造体は、MQZ_AUTHENTICATE_USER 呼び出しの *IdentityContext* パラメーターに使用します。

MQZIC 構造体は ID コンテキスト情報を含み、最初にメッセージをキューに書き込んだアプリケーションのユーザーを示します。

- キュー・マネージャーは *UserIdentifier* フィールドにユーザーを識別する名前を入力しますが、キュー・マネージャーがこれを実行する方法は、アプリケーションが実行される環境によって異なります。
- キュー・マネージャーは、*AccountingToken* フィールドに、そのメッセージを書き込んだアプリケーションから判別したトークンまたは番号を入力します。
- アプリケーションは、ユーザーに関して追加する必要がある追加の情報 (例えば、暗号化されたパスワード) 入力用として *ApplIdentityData* フィールドを使用できます。

適切に許可を与えられたアプリケーションは、MQZ_AUTHENTICATE_USER 関数を使用して ID コンテキストを設定できます。

メッセージがの下で作成されると IBM MQ for Windows、システムセキュリティ識別子 (Windows SID) がフィールド *AccountingToken* に格納される。SID の使用目的は、*UserIdentifier* フィールドの補足とユーザーの資格情報の確立です。

表 1. 構造体内のフィールドを要約します。

表 878. MQZIC のフィールド	
フィールド	説明
<u>StrucId</u>	構造体 ID
<u>バージョン</u>	バージョン
<u>UserIdentifier</u>	ユーザー ID
<u>AccountingToken</u>	アカウント・トークン

表 878. MQZIC のフィールド (続き)	
フィールド	説明
ApplIdentityData	アプリケーション識別データ

フィールド

StrucId

タイプ: MQCHAR4 - 入力

構造体 ID 値は次のとおりです。

MQZIC_STRUC_ID

ID コンテキスト構造体の ID。C 言語の場合、定数 MQZIC_STRUC_ID_ARRAY も定義されます。これは MQZIC_STRUC_ID と同じ値ですが、ストリングではなく文字の配列です。

バージョン

タイプ: MQLONG - 入力

構造体のバージョン番号。値は次のとおりです。

MQZIC_VERSION_1

バージョン 1 の ID コンテキスト構造体。

以下の定数は、現行バージョンのバージョン番号を指定しています。

MQZIC_CURRENT_VERSION

ID コンテキスト構造体の現行バージョン。

UserIdentifier

タイプ: MQCHAR12 - 入力

ユーザー ID。これは、メッセージの ID コンテキストの一部です。 *UserIdentifier* には、メッセージを発信したアプリケーションのユーザー ID を指定します。キュー・マネージャーはこの情報を文字データとして扱いますが、そのフォーマットの定義はしません。 *UserIdentifier* フィールドについて詳しくは、424 ページの『[UserIdentifier \(MQCHAR12\)](#)』を参照してください。

AccountingToken

タイプ: MQBYTE32 - 入力

アカウントリング・トークン。これは、メッセージの ID コンテキストの一部です。 *AccountingToken* を指定すると、メッセージに適切な課金が行われた結果として、アプリケーションでの作業完了が許可されます。キュー・マネージャーはこの情報をビット・ストリングとして処理し、その内容は検査しません。 *AccountingToken* フィールドについて詳しくは、425 ページの『[AccountingToken \(MQBYTE32\)](#)』を参照してください。

ApplIdentityData

タイプ: MQCHAR32 - 入力

ID に関連するアプリケーション・データ。これは、メッセージの ID コンテキストの一部です。 *ApplIdentityData* は、アプリケーション・スイートにより定義される情報で、メッセージの発信元についての追加情報を提供するのに使用できます。例えば、ID データが信頼できるかどうかを示すために、適切なユーザー権限で実行されているアプリケーションにより設定が可能です。 *ApplIdentityData* フィールドの詳細については、427 ページの『[ApplIdentityData \(MQCHAR32\)](#)』を参照してください。

C 宣言

```
typedef struct tagMQZED MQZED;
struct tagMQZED {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQCHAR12   UserIdentifier;    /* User identifier */
    MQBYTE32   AccountingToken;  /* Accounting token */
    MQCHAR32   ApplIdentityData; /* Application data relating to identity */
};
```

IBM i IBM i でのインストール可能サービス・インターフェースの参照情報

ここでは、IBM i 用インストール可能サービスに関する参照情報を記載します。

各関数には、関数 ID などの記述があります (MQZEP の場合)。

表示されているパラメーターは、指定順になっていて、すべて指定する必要があります。

各パラメーターの直後の括弧内にデータ・タイプを示します。これらは、[1002 ページの『基本データ型について IBM i』](#)に説明されている、基本的なデータ・タイプです。

パラメーターの説明の後に、C 言語による呼び出しも示します。

関連概念

IBM i [IBM i 用のインストール可能サービスとコンポーネント](#)

ALW [UNIX、Linux、および Windows 用のインストール可能サービスとコンポーネント](#)

関連資料

[1623 ページの『インストール可能サービス・インターフェースの参照情報』](#)

このトピック集では、インストール可能サービスの参照情報を記載します。

IBM i IBM i での MQZEP (コンポーネントのエントリー・ポイントの追加)

この関数は、初期化時にサービス・コンポーネントから呼び出されて、エントリー・ポイントをそのサービス・コンポーネントのエントリー・ポイント・ベクターに追加します。

構文

```
MQZEP (Hconfig, Function, EntryPoint, CompCode, Reason)
```

パラメーター

MQZEP 呼び出しには、以下のパラメーターがあります。

Hconfig (MQHCONFIG) - 入力

構成ハンドル。

このハンドルは、この特定のインストール可能サービスに構成されるコンポーネントを表します。これは、コンポーネント初期化呼び出しでキュー・マネージャーによりコンポーネント構成関数に渡されるハンドルと同じである必要があります。

Function (MQLONG) - 入力

関数 ID。

これについて有効な値は、インストール可能サービスごとに定義されます。同じ関数に対して MQZEP が複数回呼び出されると、最後の呼び出しでのエントリー・ポイントが使用されます。

EntryPoint (PMQFUNC) - 入力

関数エントリー・ポイント。

これは、関数を実行するためにコンポーネントで用意されたエントリー・ポイントのアドレスです。NULL 値は有効で、このコンポーネントで関数は用意されないことを示します。NULL は、MQZEP を使用して定義されていないエントリー・ポイントの場合に想定されます。

CompCode (MQLONG) - 出力

完了コード。

これは、以下のいずれかになります。

MQCC_OK

正常終了。

MQCC_FAILED

呼び出し失敗。

Reason (MQLONG) - 出力

CompCode を限定する理由コード。

CompCode が MQCC_OK の場合:

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_FAILED の場合:

MQRC_FUNCTION_ERROR

(2281, X'8E9') 関数 ID が無効です。

MQRC_HCONFIG_ERROR

(2280, X'8E8') 構成ハンドルが無効です。

これらの理由コードについて詳しくは、[メッセージおよび理由コード](#)を参照してください。

C 言語での呼び出し

```
MQZEP (Hconfig, Function, EntryPoint, &CompCode, &Reason);
```

パラメーターを次のように宣言します。

```
MQHCONFIG  Hconfig;      /* Configuration handle */
MQLONG     Function;     /* Function identifier */
PMQFUNC    EntryPoint;   /* Function entry point */
MQLONG     CompCode;     /* Completion code */
MQLONG     Reason;       /* Reason code qualifying CompCode */
```

IBM i MQHCONFIG (構成ハンドル)

MQHCONFIG データ・タイプは、構成ハンドル、つまり特定のインストール可能なサービス用に構成されているコンポーネントを表します。構成ハンドルは、その本来の境界に位置合わせされる必要があります。このタイプの変数については、値が等しいかどうかのみをアプリケーションでテストする必要があります。

C 宣言

```
typedef void MQPOINTER MQHCONFIG;
```

IBM i MQPMQFUNC (関数へのポインター)

関数へのポインター。

C 宣言

```
typedef void MQPOINTER PMQFUNC;
```

IBM i MQZ_AUTHENTICATE_USER (ユーザーの認証)

この関数は、MQZAS_VERSION_5 許可サービス・コンポーネントに用意されています。これは、ユーザーを認証したり、ID コンテキスト・フィールドを設定したりするために、キュー・マネージャーにより呼び出されます。

この関数は、IBM MQ のユーザー・アプリケーション・コンテキストが設定されると呼び出されます。これは、アプリケーションのユーザー・コンテキストが初期化されるポイント、およびアプリケーションのユーザー・コンテキストが変更されるポイントでの接続呼び出しの間に起こります。アプリケーションのユーザー・コンテキスト情報は、接続呼び出しが実行されるたびに *IdentityContext* フィールドに再取得されます。

この関数の関数 ID (MQZEP 用) は、MQZID_AUTHENTICATE_USER です。

構文

MQZ_AUTHENTICATE_USER (*QMGrName, SecurityParms, ApplicationContext, IdentityContext, CorrelationPtr, ComponentData, Continuation, CompCode, Reason*)

パラメーター

MQZ_AUTHENTICATE_USER 呼び出しには、以下のパラメーターがあります。

QMGrName (MQCHAR48) - 入力

キュー・マネージャー名。

コンポーネントを呼び出すキュー・マネージャーの名前。この名前は、パラメーターがすべて埋まるまで空白が埋め込まれます。名前の最後をヌル文字にすることはできません。キュー・マネージャー名は、情報としてコンポーネントに渡されます。許可サービス・インターフェースでは、コンポーネントは定義されている方法でこの情報を使用する必要はありません。

SecurityParms (MQCSP) - 入力

セキュリティー・パラメーター。

ユーザー ID、パスワード、および認証タイプに関するデータ。

MQCONN MQI 呼び出しの間は、このパラメーターは、ヌルまたはデフォルト値を含んでいます。

ApplicationContext (MQZAC) - 入力

アプリケーション・コンテキスト。

呼び出しアプリケーションに関連するデータ。詳細は [1718 ページの『IBM iでの MQZAC \(アプリケーション・コンテキスト\)』](#)を参照してください。MQCONN または MQCONNx MQI 呼び出しの際は常に、MQZAC 構造体内のユーザー・コンテキスト情報は再度取得されます。

IdentityContext (MQZIC) - 入出力

ID コンテキスト。

ユーザー認証関数への入力時に、これは現在の ID コンテキストを示します。ユーザー認証関数はこれを変更でき、この場合はキュー・マネージャーは新しい ID コンテキストを採用します。MQZIC 構造体の詳細については、[1725 ページの『IBM iでの MQZIC \(ID コンテキスト\)』](#)を参照してください。

CorrelationPtr (MQPTR) - 出力

相関ポインター。

相関データのアドレスを指定します。このポインターは、その後他の OAM 呼び出しに渡されます。

ComponentData (MQBYTE x ComponentDataLength) - 入出力

コンポーネント・データ。

このデータは、この特定のコンポーネントのためにキュー・マネージャーによって保持されます。このコンポーネントによって提供される関数のいずれかによってそのデータに変更が加えられると、その変更は保存され、次回このコンポーネントの関数の 1 つが呼び出されたときに提供されます。このデータ域の長さは、キュー・マネージャーにより MQZ_INIT_AUTHORITY 呼び出しの **ComponentDataLength** パラメーターに入れられて渡されます。

Continuation (MQLONG) - 出力

継続フラグ。

指定可能な値は、次のとおりです。

MQZCI_DEFAULT

他のコンポーネントに依存する継続。

MQZCI_STOP

次のコンポーネントに継続しない。

CompCode (MQLONG) - 出力

完了コード。

これは、以下のいずれかになります。

MQCC_OK

正常終了。

MQCC_FAILED

呼び出し失敗。

Reason (MQLONG) - 出力

CompCode を限定する理由コード。

CompCode が MQCC_OK の場合:

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_FAILED の場合:

MQRC_SERVICE_ERROR

(2289, X'8F1') サービスのアクセスで、予期しないエラーが発生しました。

これらの理由コードについて詳しくは、[メッセージおよび理由コード](#)を参照してください。

C 言語での呼び出し

```
MQZ_AUTHENTICATE_USER (QMgrName, SecurityParms, ApplicationContext,  
                        IdentityContext, &CorrelationPtr, ComponentData,  
                        &Continuation, &CompCode, &Reason);
```

サービスに渡されるパラメーターは次のように宣言されます。

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQCSP     SecurityParms;      /* Security parameters */  
MQZAC     ApplicationContext; /* Application context */  
MQZIC     IdentityContext;    /* Identity context */  
MQPTR     CorrelationPtr;     /* Correlation pointer */  
MQBYTE    ComponentData[n];  /* Component data */  
MQLONG    Continuation;      /* Continuation indicator set by  
                             component */  
MQLONG    CompCode;          /* Completion code */  
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

IBM i IBM i での MQZ_CHECK_AUTHORITY (権限の検査)

この関数は、MQZAS_VERSION_1 許可サービス・コンポーネントにより提供され、キュー・マネージャーから呼び出されて、指定されたオブジェクトに特定のアクションを実行する権限をエンティティーが持っているかどうかを検査します。

この関数の関数 ID (MQZEP 用) は、MQZID_CHECK_AUTHORITY です。

構文

MQZ_CHECK_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName, ObjectType, Authority, ComponentData, Continuation, CompCode, Reason)

パラメーター

MQZ_CHECK_AUTHORITY 呼び出しには、以下のパラメーターがあります。

QMgrName (MQCHAR48) - 入力

キュー・マネージャー名。

コンポーネントを呼び出すキュー・マネージャーの名前。この名前は、パラメーターがすべて埋まるまでブランクが埋め込まれます。名前の最後をヌル文字にすることはできません。キュー・マネー

ャー名は、情報としてコンポーネントに渡されます。許可サービス・インターフェースでは、コンポーネントは定義されている方法でこの情報を使用する必要はありません。

EntityName (MQCHAR12) - 入力

エンティティ名。

オブジェクトに対する許可を検査するエンティティの名前。ストリングの長さは最大で 12 文字です。12 文字未満の場合、その名前の右側が空白で埋められます。名前の最後をヌル文字にすることはできません。

このエンティティは、その下にあるセキュリティー・サービスに既知である必要はありません。既知でない場合、特殊な **nobody** グループ (すべてのエンティティはこのグループに所属していると想定されます) の権限が検査に使用されます。すべて空白の名前は有効で、このように使用できます。

EntityType (MQLONG) - 入力

エンティティ・タイプ。

EntityName によって指定されるエンティティのタイプ。これは、以下のいずれかになります。

MQZAET_PRINCIPAL

プリンシパル。

MQZAET_GROUP

グループ。

ObjectName (MQCHAR48) - 入力

オブジェクト名

アクセスが必要なオブジェクトの名前。ストリングの長さは最大で 48 文字です。48 文字未満の場合、その名前の右側が空白で埋められます。名前の最後をヌル文字にすることはできません。

ObjectType が MQOT_Q_MGR の場合、この名前は *QMgrName* と同じになります。

ObjectType (MQLONG) - 入力

オブジェクト・タイプ

ObjectName によって指定されるエンティティのタイプ。これは、以下のいずれかになります。

MQOT_AUTH_INFO

認証情報

MQOT_CHANNEL

チャネル。

MQOT_CLNTCONN_CHANNEL

クライアント接続チャネル。

MQOT_LISTENER

リスナー

MQOT_NAMELIST

名前リスト。

MQOT_PROCESS

プロセス定義。

MQOT_Q

キュー。

MQOT_Q_MGR

キュー・マネージャー。

MQOT_SERVICE

サービス

Authority (MQLONG) - 入力

検査される権限。

1つの許可を検査する場合、このフィールドは該当する許可操作と同じです (MQZAO_* 定数)。複数の許可を検査する場合、対応する MQZAO_* 定数とビットごとに OR 演算します。

MQI 呼び出しの使用に次の許可が適用されます。

MQZAO_CONNECT

MQCONN 呼び出しを使用する機能。

MQZAO_BROWSE

ブラウズ・オプションを使用して、MQGET 呼び出しを使用する機能。

これにより、MQGMO_BROWSE_FIRST、MQGMO_BROWSE_MSG_UNDER_CURSOR、または MQGMO_BROWSE_NEXT オプションを MQGET 呼び出しで指定できます。

MQZAO_INPUT

入力オプションを使用して、MQGET 呼び出しを使用する機能。

これにより、MQOO_INPUT_SHARED、MQOO_INPUT_EXCLUSIVE、または MQOO_INPUT_AS_Q_DEF オプションを MQOPEN 呼び出しで指定できます。

MQZAO_OUTPUT

MQPUT 呼び出しを使用する機能。

これにより、MQOO_OUTPUT オプションを MQOPEN 呼び出しで指定できます。

MQZAO_INQUIRE

MQINQ 呼び出しを使用する機能。

これにより、MQOO_INQUIRE オプションを MQOPEN 呼び出しで指定できます。

MQZAO_SET

MQSET 呼び出しを使用する機能。

これにより、MQOO_SET オプションを MQOPEN 呼び出しで指定できます。

MQZAO_PASS_IDENTITY_CONTEXT

識別コンテキストを渡す機能。

これにより、MQOO_PASS_IDENTITY_CONTEXT オプションを MQOPEN 呼び出しで指定でき、MQPMO_PASS_IDENTITY_CONTEXT オプションを MQPUT および MQPUT1 呼び出しで指定できます。

MQZAO_PASS_ALL_CONTEXT

すべてのコンテキストを渡す機能。

これにより、MQOO_PASS_ALL_CONTEXT オプションを MQOPEN 呼び出しで指定でき、MQPMO_PASS_ALL_CONTEXT オプションを MQPUT および MQPUT1 呼び出しで指定できます。

MQZAO_SET_IDENTITY_CONTEXT

識別コンテキストを設定する機能。

これにより、MQOO_SET_IDENTITY_CONTEXT オプションを MQOPEN 呼び出しで指定でき、MQPMO_SET_IDENTITY_CONTEXT オプションを MQPUT および MQPUT1 呼び出しで指定できます。

MQZAO_SET_ALL_CONTEXT

すべてのコンテキストを設定する機能。

これにより、MQOO_SET_ALL_CONTEXT オプションを MQOPEN 呼び出しで指定でき、MQPMO_SET_ALL_CONTEXT オプションを MQPUT および MQPUT1 呼び出しで指定できます。

MQZAO_ALTERNATE_USER_AUTHORITY

代替ユーザー権限を使用する機能。

これにより、MQOO_ALTERNATE_USER_AUTHORITY オプションを MQOPEN 呼び出しで指定でき、MQPMO_ALTERNATE_USER_AUTHORITY オプションを MQPUT1 呼び出しで指定できます。

MQZAO_ALL_MQI

すべての MQI 許可。

これにより、前述のすべての許可が使用可能になります。

次の許可がキュー・マネージャーの管理に適用されます。

MQZAO_CREATE

指定されたタイプのオブジェクトを作成する機能。

MQZAO_DELETE

指定されたオブジェクトを削除する機能。

MQZAO_DISPLAY

指定されたオブジェクトの属性を表示する機能。

MQZAO_CHANGE

指定されたオブジェクトの属性を変更する機能。

MQZAO_CLEAR

指定されたキューからすべてのメッセージを削除する機能。

MQZAO_AUTHORIZE

その他のユーザーに指定されたオブジェクトを許可する機能。

MQZAO_CONTROL

非クライアント・チャンネル・オブジェクトを開始、停止、または ping する機能。

MQZAO_CONTROL_EXTENDED

シーケンス番号をリセット、または非クライアント・チャンネル・オブジェクト上の未確定メッセージを解決する機能。

MQZAO_ALL_ADMIN

MQZAO_CREATE 以外のすべての管理許可。

次の許可が、MQI の両方の使用とキュー・マネージャーの管理に適用されます。

MQZAO_ALL

MQZAO_CREATE 以外のすべての許可。

MQZAO_NONE

許可なし。

ComponentData (MQBYTE x ComponentDataLength) - 入出力

コンポーネント・データ。

このデータは、この特定のコンポーネントのためにキュー・マネージャーによって保持されます。このコンポーネントによって提供される関数のいずれかによってそのデータに変更が加えられると、その変更は保存され、次回このコンポーネントの関数の 1 つが呼び出されたときに提供されます。

このデータ域の長さは、キュー・マネージャーにより MQZ_INIT_AUTHORITY 呼び出しの **ComponentDataLength** パラメーターに入れられて渡されます。

Continuation (MQLONG) - 出力

コンポーネントによって設定される継続標識。

指定可能な値は、次のとおりです。

MQZCI_DEFAULT

キュー・マネージャーに依存する継続。

MQZ_CHECK_AUTHORITY の場合、MQZCI_STOP と同じ効果があります。

MQZCI_CONTINUE

次のコンポーネントに継続。

MQZCI_STOP

次のコンポーネントに継続しない。

CompCode (MQLONG) - 出力

完了コード。

これは、以下のいずれかになります。

MQCC_OK

正常終了。

MQCC_FAILED

呼び出し失敗。

Reason (MQLONG) - 出力

CompCode を限定する理由コード。

CompCode が MQCC_OK の場合:

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_FAILED の場合:

MQRC_NOT_AUTHORIZED

(2035, X'7F3') アクセスは許可されません。

MQRC_SERVICE_ERROR

(2289, X'8F1') サービスのアクセスで、予期しないエラーが発生しました。

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') 基本サービスを使用できません。

これらの理由コードについて詳しくは、[メッセージおよび理由コード](#)を参照してください。

C 言語での呼び出し

```
MQZ_CHECK_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName,  
                     ObjectType, Authority, ComponentData,  
                     &Continuation, &CompCode, &Reason);
```

サービスに渡されるパラメーターは次のように宣言されます。

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQCHAR12  EntityName;         /* Entity name */  
MQLONG    EntityType;         /* Entity type */  
MQCHAR48  ObjectName;         /* Object name */  
MQLONG    ObjectType;         /* Object type */  
MQLONG    Authority;          /* Authority to be checked */  
MQBYTE    ComponentData[n];   /* Component data */  
MQLONG    Continuation;       /* Continuation indicator set by  
                               component */  
MQLONG    CompCode;           /* Completion code */  
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

MQZ_CHECK_PRIVILEGED - ユーザーが特権ユーザーかどうかの検査

この関数は MQZAS_VERSION_6 許可サービス・コンポーネントによって提供されています。この関数は、指定されたユーザーが特権ユーザーかどうかを判別するためにキュー・マネージャーによって呼び出されます。

この関数の関数 ID (MQZEP 用) は、MQZID_CHECK_PRIVILEGED です。

構文

```
MQZ_CHECK_PRIVILEGED( QMgrName , EntityData , EntityType , ComponentData ,  
Continuation , CompCode , Reason )
```

パラメーター

QMgrName

タイプ: MQCHAR48 - 入力

キュー・マネージャー名。コンポーネントを呼び出すキュー・マネージャーの名前。この名前は、パラメーターがすべて埋まるまでブランクが埋め込まれます。名前の最後をヌル文字にすることはできません。

キュー・マネージャー名は、情報としてコンポーネントに渡されます。許可サービス・インターフェースでは、コンポーネントは定義されている方法でこの情報を使用する必要はありません。

EntityData

タイプ: MQZED - 入力

エンティティ・データ。検査されるエンティティに関連しているデータ。詳しくは、[1682 ページ](#)の『MQZED - エンティティ記述子』を参照してください。

EntityType

タイプ: MQLONG - 入力

エンティティ・タイプ。EntityData によって指定されるエンティティのタイプ。値は、次のいずれかでなければなりません。

MQZAET_PRINCIPAL

プリンシパル。

MQZAET_GROUP

グループ。

ComponentData

タイプ: MQBYTEExComponentDataLength - 入出力

コンポーネント・データ。このデータは、この特定のコンポーネントのためにキュー・マネージャーで保持されます。このコンポーネントに用意されているいずれかの関数で変更された内容は保持され、これらのコンポーネントのいずれかの関数が次回に呼び出されると提示されます。

このデータ域の長さは、キュー・マネージャーにより MQZ_INIT_AUTHORITY 呼び出しの **ComponentDataLength** パラメーターに入れられて渡されます。

Continuation

タイプ: MQLONG - 出力

コンポーネントによって設定される継続標識。指定可能な値は、次のとおりです。

MQZCI_DEFAULT

キュー・マネージャーに依存する継続。

MQZ_CHECK_AUTHORITY の場合、MQZCI_STOP と同じ効果があります。

MQZCI_CONTINUE

次のコンポーネントに継続。

MQZCI_STOP

次のコンポーネントに継続しない。

コンポーネントの呼び出しに失敗し (つまり、CompCode が MQCC_FAILED を返す)、Continuation パラメーターが MQZCI_DEFAULT または MQZCI_CONTINUE の場合、キュー・マネージャーは他のコンポーネントの呼び出しを続けます (存在する場合)。

呼び出しに成功すると (つまり、CompCode が MQCC_OK を返す)、Continuation の設定に関係なく、コンポーネントの呼び出しはそれ以上行われなくなります。

呼び出しに失敗し、Continuation パラメーターが MQZCI_STOP の場合、その他のコンポーネントの呼び出しは行われず、エラーがキュー・マネージャーに返されます。コンポーネントは以前の呼び出しを認識していないため、呼び出しの前に Continuation パラメーターが常に MQZCI_DEFAULT に設定されます。

CompCode

タイプ: MQLONG - 出力

完了コード。値は、次のいずれかでなければなりません。

MQCC_OK

正常終了。

MQCC_FAILED

呼び出し失敗。

理由

タイプ: MQLONG - 出力

CompCode を限定する理由コード。

CompCode が MQCC_OK の場合、次のようになります。

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_FAILED の場合、次のようになります。

MQRC_NOT_PRIVILEGED

(2584, X'A18') このユーザーは特権ユーザー ID ではありません。

MQRC_UNKNOWN_ENTITY

(2292, X'8F4') サービスに対して不明なエンティティです。

MQRC_SERVICE_ERROR

(2289, X'8F1') サービスのアクセスで、予期しないエラーが発生しました。

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') 基本サービスを使用できません。

これらの理由コードについて詳しくは、[API 完了コードと理由コード](#)を参照してください。

C 言語での呼び出し

```
MQZ_CHECK_PRIVILEGED (QMgrName, &EntityData, EntityType,  
                      ComponentData, &Continuation,  
                      &CompCode, &Reason);
```

サービスに渡されるパラメーターは次のように宣言されます。

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQZED     EntityData;         /* Entity name */  
MQLONG    EntityType;         /* Entity type */  
MQBYTE    ComponentData[n];   /* Component data */  
MQLONG    Continuation;       /* Continuation indicator set by  
                               component */  
MQLONG    CompCode;           /* Completion code */  
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

IBM i IBM i での MQZ_COPY_ALL_AUTHORITY (全権限のコピー)

この関数は、許可サービス・コンポーネントに用意されています。キュー・マネージャーから呼び出され、参照オブジェクトに現在適用されているすべての許可を別のオブジェクトにコピーします。

この関数の関数 ID (MQZEP 用) は、MQZID_COPY_ALL_AUTHORITY です。

構文

MQZ_COPY_ALL_AUTHORITY (*QMgrName*, *RefObjectName*, *ObjectName*,
ObjectType, *ComponentData*, *Continuation*, *CompCode*, *Reason*)

パラメーター

MQZ_COPY_ALL_AUTHORITY 呼び出しには、以下のパラメーターがあります。

QMgrName (MQCHAR48) - 入力

キュー・マネージャー名。

コンポーネントを呼び出すキュー・マネージャーの名前。この名前は、パラメーターがすべて埋まるまで空白が埋め込まれます。名前の最後をヌル文字にすることはできません。

キュー・マネージャー名は、情報としてコンポーネントに渡されます。許可サービス・インターフェースでは、コンポーネントは定義されている方法でこの情報を使用する必要はありません。

RefObjectName (MQCHAR48) - 入力

参照オブジェクト名。

参照オブジェクトの名前で、その許可がコピーされます。ストリングの長さは最大で 48 文字です。48 文字未満の場合、その名前の右側が空白で埋められます。名前の最後をヌル文字にすることはできません。

ObjectName (MQCHAR48) - 入力

オブジェクト名

オブジェクトの名前で、それに対するアクセスが設定されます。ストリングの長さは最大で 48 文字です。48 文字未満の場合、その名前の右側が空白で埋められます。名前の最後をヌル文字にすることはできません。

ObjectType (MQLONG) - 入力

オブジェクト・タイプ

RefObjectName および *ObjectName* によって指定されるオブジェクトのタイプ。これは、以下のいずれかになります。

MQOT_AUTH_INFO

認証情報

MQOT_CHANNEL

チャンネル。

MQOT_CLNTCONN_CHANNEL

クライアント接続チャンネル。

MQOT_LISTENER

リスナー

MQOT_NAMELIST

名前リスト。

MQOT_PROCESS

プロセス定義。

MQOT_Q

キュー。

MQOT_Q_MGR

キュー・マネージャー。

MQOT_SERVICE

サービス

ComponentData (MQBYTE x ComponentDataLength) - 入出力

コンポーネント・データ。

このデータは、この特定のコンポーネントのためにキュー・マネージャーによって保持されます。このコンポーネントによって提供される関数のいずれかによってそのデータに変更が加えられると、その変更は保存され、次回このコンポーネントの関数の 1 つが呼び出されたときに提供されます。

このデータ域の長さは、キュー・マネージャーにより MQZ_INIT_AUTHORITY 呼び出しの **ComponentDataLength** パラメーターに入れられて渡されます。

Continuation (MQLONG) - 出力

コンポーネントによって設定される継続標識。

指定可能な値は、次のとおりです。

MQZCI_DEFAULT

キュー・マネージャーに依存する継続。

MQZ_COPY_ALL_AUTHORITY の場合、これは MQZCI_STOP と同じ効果があります。

MQZCI_CONTINUE

次のコンポーネントに継続。

MQZCI_STOP

次のコンポーネントに継続しない。

CompCode (MQLONG) - 出力

完了コード。

これは、以下のいずれかになります。

MQCC_OK

正常終了。

MQCC_FAILED

呼び出し失敗。

Reason (MQLONG) - 出力

CompCode を限定する理由コード。

CompCode が MQCC_OK の場合:

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_FAILED の場合:

MQRC_SERVICE_ERROR

(2289, X'8F1') サービスのアクセスで、予期しないエラーが発生しました。

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') 基本サービスを使用できません。

MQRC_UNKNOWN_REF_OBJECT

(2294, X'8F6') 参照オブジェクトが不明です。

これらの理由コードについて詳しくは、[メッセージおよび理由コード](#)を参照してください。

C 言語での呼び出し

```
MQZ_COPY_ALL_AUTHORITY (QMgrName, RefObjectName, ObjectName, ObjectType,  
                          ComponentData, &Continuation, &CompCode,  
                          &Reason);
```

サービスに渡されるパラメーターは次のように宣言されます。

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQCHAR48  RefObjectName;      /* Reference object name */  
MQCHAR48  ObjectName;        /* Object name */  
MQLONG    ObjectType;         /* Object type */  
MQBYTE    ComponentData[n];  /* Component data */  
MQLONG    Continuation;       /* Continuation indicator set by  
                               component */  
MQLONG    CompCode;           /* Completion code */  
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

IBM i IBM i での MQZ_DELETE_AUTHORITY (権限の削除)

この関数は、許可サービス・コンポーネントにより提供され、キュー・マネージャーから呼び出されて、指定されたオブジェクトに関連するすべての許可を削除します。

この関数の関数 ID (MQZEP 用) は、MQZID_DELETE_AUTHORITY です。

構文

**MQZ_DELETE_AUTHORITY (QMgrName, ObjectName, ObjectType,
ComponentData, Continuation, CompCode, Reason)**

パラメーター

MQZ_DELETE_AUTHORITY 呼び出しには、以下のパラメーターがあります。

QMgrName (MQCHAR48) - 入力

キュー・マネージャー名。

コンポーネントを呼び出すキュー・マネージャーの名前。この名前は、パラメーターがすべて埋まるまで空白が埋め込まれます。名前の最後をヌル文字にすることはできません。

キュー・マネージャー名は、情報としてコンポーネントに渡されます。許可サービス・インターフェースでは、コンポーネントは定義されている方法でこの情報を使用する必要はありません。

ObjectName (MQCHAR48) - 入力

オブジェクト名

オブジェクトの名前で、それに対するアクセスが削除されます。stringの長さは最大で 48 文字です。48 文字未満の場合、その名前の右側が空白で埋められます。名前の最後をヌル文字にすることはできません。

ObjectType が MQOT_Q_MGR の場合、この名前は *QMgrName* と同じになります。

ObjectType (MQLONG) - 入力

オブジェクト・タイプ

ObjectName によって指定されるエンティティのタイプ。これは、以下のいずれかになります。

MQOT_AUTH_INFO

認証情報

MQOT_CHANNEL

チャンネル。

MQOT_CLNTCONN_CHANNEL

クライアント接続チャンネル。

MQOT_LISTENER

リスナー

MQOT_NAMELIST

名前リスト。

MQOT_PROCESS

プロセス定義。

MQOT_Q

キュー。

MQOT_Q_MGR

キュー・マネージャー。

MQOT_SERVICE

サービス

ComponentData (MQBYTE x ComponentDataLength) - 入出力

コンポーネント・データ。

このデータは、この特定のコンポーネントのためにキュー・マネージャーによって保持されます。このコンポーネントによって提供される関数のいずれかによってそのデータに変更が加えられると、その変更は保存され、次回このコンポーネントの関数の 1 つが呼び出されたときに提供されます。

このデータ域の長さは、キュー・マネージャーにより MQZ_INIT_AUTHORITY 呼び出しの **ComponentDataLength** パラメーターに入れられて渡されます。

Continuation (MQLONG) - 出力

コンポーネントによって設定される継続標識。

指定可能な値は、次のとおりです。

MQZCI_DEFAULT

キュー・マネージャーに依存する継続。

MQZ_DELETE_AUTHORITY の場合、これは MQZCI_STOP と同じ効果があります。

MQZCI_CONTINUE

次のコンポーネントに継続。

MQZCI_STOP

次のコンポーネントに継続しない。

CompCode (MQLONG) - 出力

完了コード。

これは、以下のいずれかになります。

MQCC_OK

正常終了。

MQCC_FAILED

呼び出し失敗。

Reason (MQLONG) - 出力

CompCode を限定する理由コード。

CompCode が MQCC_OK の場合:

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_FAILED の場合:

MQRC_SERVICE_ERROR

(2289, X'8F1') サービスのアクセスで、予期しないエラーが発生しました。

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') 基本サービスを使用できません。

これらの理由コードについて詳しくは、[メッセージおよび理由コード](#)を参照してください。

C 言語での呼び出し

```
MQZ_DELETE_AUTHORITY (QMgrName, ObjectName, ObjectType, ComponentData,  
                      &Continuation, &CompCode, &Reason);
```

サービスに渡されるパラメーターは次のように宣言されます。

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQCHAR48  ObjectName;         /* Object name */  
MQLONG    ObjectType;         /* Object type */  
MQBYTE    ComponentData[n];   /* Component data */  
MQLONG    Continuation;       /* Continuation indicator set by  
                               component */  
MQLONG    CompCode;           /* Completion code */  
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

IBM i での MQZ_ENUMERATE_AUTHORITY_DATA (権限データの列挙)

この関数は、MQZAS_VERSION_4 許可サービス・コンポーネントに用意されており、キュー・マネージャーから繰り返し呼び出されて、最初の呼び出しで指定された選択基準に合致するすべての権限データを検索します。

この関数の関数 ID (MQZEP 用) は、MQZID_ENUMERATE_AUTHORITY_DATA です。

構文

**MQZ_ENUMERATE_AUTHORITY_DATA (QMgrName, StartEnumeration,
Filter, AuthorityBufferLength, AuthorityBuffer, AuthorityDataLength,
ComponentData, Continuation, CompCode, Reason)**

パラメーター

MQZ_ENUMERATE_AUTHORITY_DATA 呼び出しには、以下のパラメーターがあります。

QMgrName (MQCHAR48) - 入力

キュー・マネージャー名。

コンポーネントを呼び出すキュー・マネージャーの名前。この名前は、パラメーターがすべて埋まるまで空白が埋め込まれます。名前の最後をヌル文字にすることはできません。

キュー・マネージャー名は、情報としてコンポーネントに渡されます。許可サービス・インターフェースでは、コンポーネントは定義されている方法でこの情報を使用する必要はありません。

StartEnumeration (MQLONG) - 入力

呼び出して列挙を開始するかどうかを示すフラグ。

これは、この呼び出しによって権限データの列挙を開始できるか、それとも MQZ_ENUMERATE_AUTHORITY_DATA に対する前回の呼び出しによって開始された権限データの列挙を継続するかを示します。値は、次のいずれか 1 つです。

MQZSE_START

列挙を開始します。

この値を使用して呼び出されると、権限データの列挙を開始します。**Filter** パラメーターに、この呼び出しとその後の呼び出しで返される権限データを選択するのに使用する選択基準を指定します。

MQZSE_CONTINUE

列挙を継続します。

この値を使用して呼び出されると、権限データの列挙を継続します。この場合、**Filter** パラメーターは無視され、ヌル・ポインターとして指定できます (選択基準は、*StartEnumeration* が MQZSE_START に設定された呼び出しによって指定された **Filter** パラメーターによって決定されます)。

Filter (MQZAD) - 入力

フィルター。

StartEnumeration が MQZSE_START の場合、**Filter** は、返される権限データを選択するために使用する選択基準を指定します。**Filter** がヌル・ポインターの場合、選択基準は使用されず、すべての権限データが返されます。使用できる選択基準の詳細については、[1720 ページの『IBM i での MQZAD \(権限データ\)』](#)を参照してください。

StartEnumeration が MQZSE_CONTINUE の場合、**Filter** は無視され、ヌル・ポインターとして指定できます。

AuthorityBufferLength (MQLONG) - 入力

AuthorityBuffer の長さ。

これは、**AuthorityBuffer** パラメーターの長さ (バイト単位) です。権限バッファは、返されるデータを格納できる大きさでなければなりません。

AuthorityBuffer (MQZAD) - 出力

権限データ。

権限データが返されるバッファです。このバッファは、MQZAD 構造体、MQZED 構造体、および定義された最長のエンティティ名とドメイン名を格納できる大きさでなければなりません。

注: MQZAD は常にバッファの開始時に作成されるので、このパラメーターは MQZAD として定義されます。ただし、バッファを実際に MQZAD として宣言する場合、バッファは小さすぎます。MQZAD より大きくして、MQZAD、MQZED、およびエンティティ名とドメイン名が収まるようにする必要があります。

AuthorityDataLength (MQLONG) - 出力

AuthorityBuffer に返されるデータ長。

AuthorityBuffer に返されるデータの長さです。権限バッファが小さすぎると、*AuthorityDataLength* は必要なバッファ長に設定され、呼び出しで完了コード *MQCC_FAILED* と理由コード *MQRC_BUFFER_LENGTH_ERROR* が返されます。

ComponentData (MQBYTE x ComponentDataLength) - 入出力

コンポーネント・データ。

このデータは、この特定のコンポーネントのためにキュー・マネージャーによって保持されます。このコンポーネントによって提供される関数のいずれかによってそのデータに変更が加えられると、その変更は保存され、次回このコンポーネントの関数の 1 つが呼び出されたときに提供されます。

このデータ域の長さは、キュー・マネージャーにより *MQZ_INIT_AUTHORITY* 呼び出しの *ComponentDataLength* パラメーターに入れられて渡されます。

Continuation (MQLONG) - 出力

コンポーネントによって設定される継続標識。

指定可能な値は、次のとおりです。

MQZCI_DEFAULT

キュー・マネージャーに依存する継続。

MQZ_ENUMERATE_AUTHORITY_DATA の場合、これは *MQZCI_CONTINUE* と同じ効果があります。

MQZCI_CONTINUE

次のコンポーネントに継続。

MQZCI_STOP

次のコンポーネントに継続しない。

CompCode (MQLONG) - 出力

完了コード。

これは、以下のいずれかになります。

MQCC_OK

正常終了。

MQCC_FAILED

呼び出し失敗。

Reason (MQLONG) - 出力

CompCode を限定する理由コード。

CompCode が *MQCC_OK* の場合:

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が *MQCC_FAILED* の場合:

MQRC_BUFFER_LENGTH_ERROR

(2005, X'7D5') バッファ長パラメーターは無効です。

MQRC_NO_DATA_AVAILABLE

(2379, X'94B') 使用可能なデータはありません。

MQRC_SERVICE_ERROR

(2289, X'8F1') サービスのアクセスで、予期しないエラーが発生しました。

これらの理由コードについて詳しくは、[メッセージおよび理由コード](#)を参照してください。

C 言語での呼び出し

```
MQZ_ENUMERATE_AUTHORITY_DATA (QMgrName, StartEnumeration, &Filter,  
                               AuthorityBufferLength,  
                               &AuthorityBuffer,  
                               &AuthorityDataLength, ComponentData,
```

```
&Continuation, &CompCode,  
&Reason);
```

サービスに渡されるパラメーターは次のように宣言されます。

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQLONG    StartEnumeration;   /* Flag indicating whether call should  
                                start enumeration */  
MQZAD     Filter;             /* Filter */  
MQLONG    AuthorityBufferLength; /* Length of AuthorityBuffer */  
MQZAD     AuthorityBuffer;    /* Authority data */  
MQLONG    AuthorityDataLength; /* Length of data returned in  
                                AuthorityBuffer */  
MQBYTE    ComponentData[n];   /* Component data */  
MQLONG    Continuation;       /* Continuation indicator set by  
                                component */  
MQLONG    CompCode;           /* Completion code */  
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

MQZ_FREE_USER - ユーザーの解放

この関数は、MQZAS_VERSION_5 許可サービス・コンポーネントに用意されており、キュー・マネージャーから呼び出されて、関連している割り振り済みのリソースを解放します。この関数は、アプリケーションがすべてのユーザー・コンテキストの下で (例えば、MQDISC MQI 呼び出し時に) 実行を終了すると呼び出されます。

この関数の関数 ID (MQZEP 用) は、MQZID_FREE_USER です。

IBM i IBM i での MQZ_GET_AUTHORITY (権限の取得)

この関数は、MQZAS_VERSION_1 許可サービス・コンポーネントにより提供され、キュー・マネージャーから呼び出されて、エンティティーが持っている指定されたオブジェクトへのアクセス権限を検索します。

この関数の関数 ID (MQZEP 用) は、MQZID_GET_AUTHORITY です。

構文

MQZ_GET_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName, ObjectType, Authority, ComponentData, Continuation, CompCode, Reason)

パラメーター

MQZ_GET_AUTHORITY 呼び出しには、以下のパラメーターがあります。

QMgrName (MQCHAR48) - 入力

キュー・マネージャー名。

コンポーネントを呼び出すキュー・マネージャーの名前。この名前は、パラメーターがすべて埋まるまでブランクが埋め込まれます。名前の最後をヌル文字にすることはできません。

キュー・マネージャー名は、情報としてコンポーネントに渡されます。許可サービス・インターフェースでは、コンポーネントは定義されている方法でこの情報を使用する必要はありません。

EntityName (MQCHAR12) - 入力

エンティティー名。

オブジェクトに対するアクセス権限を検索するエンティティーの名前。ストリングの長さは最大で 12 文字です。12 文字未満の場合、その名前の右側がブランクで埋められます。名前の最後をヌル文字にすることはできません。

EntityType (MQLONG) - 入力

エンティティー・タイプ。

EntityName によって指定されるエンティティーのタイプ。次の値を指定できます。

MQZAET_PRINCIPAL

プリンシパル。

MQZAET_GROUP

グループ。

ObjectName (MQCHAR48) - 入力

オブジェクト名

オブジェクトの名前で、それに対するエンティティの権限が検索されます。ストリングの長さは最大で 48 文字です。48 文字未満の場合、その名前の右側が空白で埋められます。名前の最後をヌル文字にすることはできません。

ObjectType が MQOT_Q_MGR の場合、この名前は *QMgrName* と同じになります。

ObjectType (MQLONG) - 入力

オブジェクト・タイプ

ObjectName によって指定されるエンティティのタイプ。これは、以下のいずれかになります。

MQOT_AUTH_INFO

認証情報

MQOT_CHANNEL

チャンネル。

MQOT_CLNTCONN_CHANNEL

クライアント接続チャンネル。

MQOT_LISTENER

リスナー

MQOT_NAMELIST

名前リスト。

MQOT_PROCESS

プロセス定義。

MQOT_Q

キュー。

MQOT_Q_MGR

キュー・マネージャー。

MQOT_SERVICE

サービス

Authority (MQLONG) - 出力

エンティティの権限。

エンティティの権限が 1 つの場合、このフィールドは該当する許可演算 (MQZAO_* 定数) に等しくなります。複数の権限を持つ場合、このフィールドは対応する MQZAO_* 定数のビット単位 OR になります。

ComponentData (MQBYTE x ComponentDataLength) - 入出力

コンポーネント・データ。

このデータは、この特定のコンポーネントのためにキュー・マネージャーによって保持されます。このコンポーネントによって提供される関数のいずれかによってそのデータに変更が加えられると、その変更は保存され、次回このコンポーネントの関数の 1 つが呼び出されたときに提供されます。

このデータ域の長さは、キュー・マネージャーにより MQZ_INIT_AUTHORITY 呼び出しの **ComponentDataLength** パラメーターに入れられて渡されます。

Continuation (MQLONG) - 出力

コンポーネントによって設定される継続標識。

指定可能な値は、次のとおりです。

MQZCI_DEFAULT

キュー・マネージャーに依存する継続。

MQZ_GET_AUTHORITY の場合、これは MQZCI_CONTINUE と同じ効果があります。

MQZCI_CONTINUE

次のコンポーネントに継続。

MQZCI_STOP

次のコンポーネントに継続しない。

CompCode (MQLONG) - 出力

完了コード。

これは、以下のいずれかになります。

MQCC_OK

正常終了。

MQCC_FAILED

呼び出し失敗。

Reason (MQLONG) - 出力

CompCode を限定する理由コード。

CompCode が MQCC_OK の場合:

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_FAILED の場合:

MQRC_NOT_AUTHORIZED

(2035, X'7F3') アクセスは許可されません。

MQRC_SERVICE_ERROR

(2289, X'8F1') サービスのアクセスで、予期しないエラーが発生しました。

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') 基本サービスを使用できません。

MQRC_UNKNOWN_ENTITY

(2292, X'8F4') サービスに対して不明なエンティティです。

これらの理由コードについて詳しくは、[メッセージおよび理由コード](#)を参照してください。

C 言語での呼び出し

```
MQZ_GET_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName,  
                   ObjectType, &Authority, ComponentData,  
                   &Continuation, &CompCode, &Reason);
```

サービスに渡されるパラメーターは次のように宣言されます。

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQCHAR12  EntityName;         /* Entity name */  
MQLONG    EntityType;         /* Entity type */  
MQCHAR48  ObjectName;         /* Object name */  
MQLONG    ObjectType;         /* Object type */  
MQLONG    Authority;          /* Authority of entity */  
MQBYTE    ComponentData[n];   /* Component data */  
MQLONG    Continuation;       /* Continuation indicator set by  
                               component */  
MQLONG    CompCode;           /* Completion code */  
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

IBM i IBM i での MQZ_GET_EXPLICIT_AUTHORITY (明示権限の取得)

この関数は、MQZAS_VERSION_1 許可サービス・コンポーネントに用意されており、キュー・マネージャーから呼び出されて、名前付きグループが持っている指定されたオブジェクトへのアクセス権限 (ただし、なしグループの追加権限は除く) を検索するか、または名前付きプリンシパルの主グループが持っている指定されたオブジェクトへのアクセス権限を検索します。

この関数の関数 ID (MQZEP 用) は、MQZID_GET_EXPLICIT_AUTHORITY です。

構文

MQZ_GET_EXPLICIT_AUTHORITY (*QMgrName, EntityName, EntityType, ObjectName, ObjectType, Authority, ComponentData, Continuation, CompCode, Reason*)

パラメーター

MQZ_GET_EXPLICIT_AUTHORITY 呼び出しには、以下のパラメーターがあります。

QMgrName (MQCHAR48) - 入力

キュー・マネージャー名。

コンポーネントを呼び出すキュー・マネージャーの名前。この名前は、パラメーターがすべて埋まるまでブランクが埋め込まれます。名前の最後をヌル文字にすることはできません。

キュー・マネージャー名は、情報としてコンポーネントに渡されます。許可サービス・インターフェースでは、コンポーネントは定義されている方法でこの情報を使用する必要はありません。

EntityName (MQCHAR12) - 入力

エンティティ名。

オブジェクトに対するアクセス権限を検索するエンティティの名前。ストリングの長さは最大で 12 文字です。12 文字未満の場合、その名前の右側がブランクで埋められます。名前の最後をヌル文字にすることはできません。

EntityType (MQLONG) - 入力

エンティティ・タイプ。

EntityName によって指定されるエンティティのタイプ。次の値を指定できます。

MQZAET_PRINCIPAL

プリンシパル。

MQZAET_GROUP

グループ。

ObjectName (MQCHAR48) - 入力

オブジェクト名

オブジェクトの名前で、それに対するエンティティの権限が検索されます。ストリングの長さは最大で 48 文字です。48 文字未満の場合、その名前の右側がブランクで埋められます。名前の最後をヌル文字にすることはできません。

ObjectType が MQOT_Q_MGR の場合、この名前は *QMgrName* と同じになります。

ObjectType (MQLONG) - 入力

オブジェクト・タイプ

ObjectName によって指定されるエンティティのタイプ。これは、以下のいずれかになります。

MQOT_AUTH_INFO

認証情報

MQOT_CHANNEL

チャネル。

MQOT_CLNTCONN_CHANNEL

クライアント接続チャネル。

MQOT_LISTENER

リスナー

MQOT_NAMELIST

名前リスト。

MQOT_PROCESS

プロセス定義。

MQOT_Q

キュー。

MQOT_Q_MGR

キュー・マネージャー。

MQOT_SERVICE

サービス

Authority (MQLONG) - 出力

エンティティの権限。

エンティティの権限が1つの場合、このフィールドは該当する許可演算 (MQZAO_* 定数) に等しくなります。複数の権限を持つ場合、このフィールドは対応する MQZAO_* 定数のビット単位 OR になります。

ComponentData (MQBYTE x ComponentDataLength) - 入出力

コンポーネント・データ。

このデータは、この特定のコンポーネントのためにキュー・マネージャーによって保持されます。このコンポーネントによって提供される関数のいずれかによってそのデータに変更が加えられると、その変更は保存され、次回このコンポーネントの関数の1つが呼び出されたときに提供されます。

このデータ域の長さは、キュー・マネージャーにより MQZ_INIT_AUTHORITY 呼び出しの **ComponentDataLength** パラメーターに入れられて渡されます。

Continuation (MQLONG) - 出力

コンポーネントによって設定される継続標識。

指定可能な値は、次のとおりです。

MQZCI_DEFAULT

キュー・マネージャーに依存する継続。

MQZ_GET_EXPLICIT_AUTHORITY の場合、これは MQZCI_CONTINUE と同じ効果があります。

MQZCI_CONTINUE

次のコンポーネントに継続。

MQZCI_STOP

次のコンポーネントに継続しない。

CompCode (MQLONG) - 出力

完了コード。

これは、以下のいずれかになります。

MQCC_OK

正常終了。

MQCC_FAILED

呼び出し失敗。

Reason (MQLONG) - 出力*CompCode* を限定する理由コード。*CompCode* が MQCC_OK の場合:**MQRC_NONE**

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_FAILED の場合:**MQRC_NOT_AUTHORIZED**

(2035, X'7F3') アクセスは許可されません。

MQRC_SERVICE_ERROR

(2289, X'8F1') サービスのアクセスで、予期しないエラーが発生しました。

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') 基本サービスを使用できません。

MQRC_UNKNOWN_ENTITY

(2292, X'8F4') サービスに対して不明なエンティティです。

これらの理由コードについて詳しくは、[メッセージおよび理由コード](#)を参照してください。

C 言語での呼び出し

```
MQZ_GET_EXPLICIT_AUTHORITY (QMgrName, EntityName, EntityType,  
                             ObjectName, ObjectType, &Authority,  
                             ComponentData, &Continuation,  
                             &CompCode, &Reason);
```

サービスに渡されるパラメーターは次のように宣言されます。

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQCHAR12  EntityName;         /* Entity name */  
MQLONG    EntityType;         /* Entity type */  
MQCHAR48  ObjectName;         /* Object name */  
MQLONG    ObjectType;         /* Object type */  
MQLONG    Authority;          /* Authority of entity */  
MQBYTE    ComponentData[n];   /* Component data */  
MQLONG    Continuation;       /* Continuation indicator set by  
                               component */  
MQLONG    CompCode;           /* Completion code */  
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

IBM i IBM i での MQZ_INIT_AUTHORITY (許可サービスの初期化)

この関数は許可サービス・コンポーネントに用意されています。キュー・マネージャーはコンポーネントの構成時にこの関数を呼び出します。キュー・マネージャーに情報を提供するために、MQZEP を呼び出すことが想定されます。

この関数の関数 ID (MQZEP 用) は、MQZID_INIT_AUTHORITY です。

構文

**MQZ_INIT_AUTHORITY (Hconfig, Options, QMgrName, ComponentDataLength,
ComponentData, Version, CompCode, Reason)**

パラメーター

MQZ_INIT_AUTHORITY 呼び出しには、以下のパラメーターがあります。

Hconfig (MQHCONFIG) - 入力

構成ハンドル。

このハンドルは、初期化される特定のコンポーネントを表します。これは、MQZEP 関数を使用してキュー・マネージャーを呼び出す際に、コンポーネントにより使用されます。

Options (MQLONG) - 入力

初期化オプション。

これは、以下のいずれかになります。

MQZIO_PRIMARY

1 次初期化。

MQZIO_SECONDARY

2 次初期化。

QMgrName (MQCHAR48) - 入力

キュー・マネージャー名。

コンポーネントを呼び出すキュー・マネージャーの名前。この名前は、パラメーターがすべて埋まるまでブランクが埋め込まれます。名前の最後をヌル文字にすることはできません。

キュー・マネージャー名は、情報としてコンポーネントに渡されます。許可サービス・インターフェースでは、コンポーネントは定義されている方法でこの情報を使用する必要はありません。

ComponentDataLength (MQLONG) - 入力

コンポーネント・データの長さ。

ComponentData 域の長さ (バイト単位)。この長さは、コンポーネント構成データに定義されます。

ComponentData (MQBYTE x ComponentDataLength) - 入出力

コンポーネント・データ。

コンポーネントの 1 次初期化関数が呼び出されるまで、すべてゼロに初期化されています。このデータは、この特定のコンポーネントのためにキュー・マネージャーで保持されます。このコンポーネントに用意されているいずれかの関数 (初期化関数を含む) で変更された内容は保持され、このコンポーネントのいずれかの関数が次回に呼び出されると提示されます。

Version (MQLONG) - 入出力

バージョン番号。

初期化関数への入力時に、これによりそのキュー・マネージャーがサポートする最新のバージョン番号を識別します。初期化関数は、必要により、キュー・マネージャーがサポートするインターフェースのバージョンに変更する必要があります。キュー・マネージャーがコンポーネントから返されるバージョンをサポートしないことが返されると、キュー・マネージャーはコンポーネントの MQZ_TERM_AUTHORITY 関数を呼び出し、その後はこのコンポーネントを使用しません。

次の値がサポートされます。

MQZAS_VERSION_1

バージョン 1。

MQZAS_VERSION_2

バージョン 2。

MQZAS_VERSION_3

バージョン 3。

MQZAS_VERSION_4

バージョン 4。

MQZAS_VERSION_5

バージョン 5。

MQZAS_VERSION_6

IBM WebSphere MQ 6.

CompCode (MQLONG) - 出力

完了コード。

これは、以下のいずれかになります。

MQCC_OK

正常終了。

MQCC_FAILED

呼び出し失敗。

Reason (MQLONG) - 出力

CompCode を限定する理由コード。

CompCode が MQCC_OK の場合:

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_FAILED の場合:

MQRC_INITIALIZATION_FAILED

(2286, X'8EE') 未定義の理由で初期化に失敗しました。

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') 基本サービスを使用できません。

これらの理由コードについて詳しくは、[メッセージおよび理由コード](#)を参照してください。

C 言語での呼び出し

```
MQZ_INIT_AUTHORITY (Hconfig, Options, QMgrName, ComponentDataLength,  
                    ComponentData, &Version, &CompCode,  
                    &Reason);
```

サービスに渡されるパラメーターは次のように宣言されます。

```
MQHCONFIG  Hconfig;           /* Configuration handle */  
MQLONG     Options;           /* Initialization options */  
MQCHAR48   QMgrName;         /* Queue manager name */  
MQLONG     ComponentDataLength; /* Length of component data */  
MQBYTE     ComponentData[n]; /* Component data */  
MQLONG     Version;          /* Version number */  
MQLONG     CompCode;         /* Completion code */  
MQLONG     Reason;           /* Reason code qualifying CompCode */
```

IBM i IBM i での MQZ_INQUIRE (許可サービスの照会)

この関数は、MQZAS_VERSION_5 許可サービス・コンポーネントにより提供され、キュー・マネージャーから呼び出されて、サポートされている機能を照会します。複数のサービス・コンポーネントが使用されている場合、サービス・コンポーネントは、インストールされたときの順序とは逆の順序で呼び出されます。

この関数の関数 ID (MQZEP 用) は、MQZID_INQUIRE です。

構文

MQZ_INQUIRE

(QMgrName, SelectorCount, Selectors, IntAttrCount, IntAttrs, CharAttrLength, CharAttrs, SelectorReturned, ComponentData, Continuation, CompCode, Reason)

パラメーター

MQZ_INQUIRE 呼び出しには、以下のパラメーターがあります。

QMgrName (MQCHAR48) - 入力

キュー・マネージャー名。

コンポーネントを呼び出すキュー・マネージャーの名前。この名前は、パラメーターがすべて埋まるまでブランクが埋め込まれます。名前の最後をヌル文字にすることはできません。

キュー・マネージャー名は、情報としてコンポーネントに渡されます。許可サービス・インターフェースでは、コンポーネントは定義されている方法でこの情報を使用する必要はありません。

SelectorCount (MQLONG) - 入力

セレクターの数。

Selectors パラメーターで提供されているセレクターの数。

値は 0 と 256 の間の必要があります。

Selectors (MQLONG x SelectorCount) - 入力

セレクター。

セレクターの配列。それぞれのセレクターは、必要な属性を示し、次のタイプの内の 1 つである必要があります。

- MQIACF_* (整数)
- MQCACF_* (文字)

選択子は任意の順序で指定できます。配列内のセクター数は、SelectorCount パラメーターで指定します。

セクターによって識別される整数属性は、Selectors パラメーターでの指定順序と同じ順序で IntAttrs パラメーターに戻されます。

セクターによって示される文字属性は、Selectors に現れるのと同じ順序で、CharAttrs パラメーターに戻されます。

IntAttrCount (MQLONG) - 入力

整数属性の数。

IntAttrs パラメーターで提供される整数属性の数。

値は 0 から 256 の範囲でなければなりません。

IntAttrs (MQLONG x IntAttrCount) - 出力

整数属性。

整数属性の配列。整数属性は、Selectors 配列内の対応する整数セクターと同じ順序で戻されます。

CharAttrCount (MQLONG) - 入力

文字属性バッファの長さ。

CharAttrs パラメーターの長さ (バイト数)。

値は、要求された文字属性の長さの合計以上でなければなりません。文字属性の要求がない場合は、ゼロが有効な値です。

CharAttrs (MQLONG x CharAttrCount) - 出力

文字属性バッファ。

連結された文字属性が入るバッファ。文字属性は、Selectors 配列内の対応する文字セクターと同じ順序で戻されます。

バッファ長は、CharAttrCount パラメーターで与えられます。

SelectorReturned (MQLONGxSelectorCount) - 入力

戻されたセクター。

Selectors パラメーター内のセクターにより要求されたセットの中から、どの属性が戻されたかを示す値の配列。この配列内の値の数は、SelectorCount パラメーターにより示されます。この配列内のそれぞれの値は、Selectors 配列の対応する位置にあるセクターに関係付けられています。それぞれの値は以下のいずれかです。

MQZSL_RETURNED

Selectors パラメーター内の対応するセクターにより要求された属性が戻されました。

MQZSL_NOT_RETURNED

Selectors パラメーター内の対応するセクターにより要求された属性が戻されません。

この配列では、すべての値は *MQZSL_NOT_RETURNED* として初期化されます。許可サービス・コンポーネントが属性を戻すときに、配列中の値を *MQZSL_RETURNED* に設定します。このようにすることで、照会呼び出しがなされた他の許可サービス・コンポーネントは、どの属性が既に返されたかを識別できます。

ComponentData (MQBYTE x ComponentDataLength) - 入出力

コンポーネント・データ。

このデータは、この特定のコンポーネントのためにキュー・マネージャーによって保持されます。このコンポーネントによって提供される関数のいずれかによってそのデータに変更が加えられると、その変更は保存され、次回このコンポーネントの関数の 1 つが呼び出されたときに提供されます。

このデータ域の長さは、キュー・マネージャーにより MQZ_INIT_AUTHORITY 呼び出しの

ComponentDataLength パラメーターに入れられて渡されます。

Continuation (MQLONG) - 出力

継続フラグ。

指定可能な値は、次のとおりです。

MQZCI_DEFAULT

他のコンポーネントに依存する継続。

MQZCI_STOP

次のコンポーネントに継続しない。

CompCode (MQLONG) - 出力

完了コード。

これは、以下のいずれかになります。

MQCC_OK

正常終了。

MQCC_WARNING

一部完了。

MQCC_FAILED

呼び出し失敗。

Reason (MQLONG) - 出力

CompCode を限定する理由コード。

CompCode が MQCC_OK の場合:

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_WARNING の場合:

MQRC_CHAR_ATTRS_TOO_SHORT

文字属性用のスペースが十分ではありません。

MQRC_INT_COUNT_TOO_SMALL

整数属性用のスペースが十分ではありません。

CompCode が MQCC_FAILED の場合:

MQRC_SELECTOR_COUNT_ERROR

セレクターの数が無効です。

MQRC_SELECTOR_ERROR

属性セレクターが無効です。

MQRC_SELECTOR_LIMIT_EXCEEDED

指定されたセレクターの数が多すぎます。

MQRC_INT_ATTR_COUNT_ERROR

整数属性の数が無効です。

MQRC_INT_ATTRS_ARRAY_ERROR

整数属性配列が無効です。

MQRC_CHAR_ATTR_LENGTH_ERROR

文字属性の数が無効です。

MQRC_CHAR_ATTRS_ERROR

文字属性ストリングが無効です。

MQRC_SERVICE_ERROR

(2289, X'8F1') サービスのアクセスで、予期しないエラーが発生しました。

C 言語での呼び出し

```
MQZ_INQUIRE (QMgrName, SelectorCount, Selectors, IntAttrCount,  
              &IntAttrs, CharAttrLength, &CharAttrs,  
              SelectorReturned, ComponentData, &Continuation,  
              &CompCode, &Reason);
```

サービスに渡されるパラメーターは次のように宣言されます。

```
MQCHAR48  QMgrName;           /* Queue manager name */
MQLONG    SelectorCount;      /* Selector count */
MQLONG    Selectors[n];       /* Selectors */
MQLONG    IntAttrCount;       /* IntAttrs count */
MQLONG    IntAttrs[n];        /* Integer attributes */
MQLONG    CharAttrCount;      /* CharAttrs count */
MQLONG    CharAttrs[n];       /* Character attributes */
MQLONG    SelectorReturned[n]; /* Selector returned */
MQBYTE    ComponentData[n];   /* Component data */
MQLONG    Continuation;       /* Continuation indicator set by
                               component */
MQLONG    CompCode;           /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

IBM i IBM i での MQZ_REFRESH_CACHE (すべての許可の最新表示)

この関数は、MQZAS_VERSION_3 許可サービス・コンポーネントに用意されています。この関数は、キュー・マネージャーから呼び出されて、コンポーネントの内部に保持されている許可リストをリフレッシュします。

この関数の関数 ID (MQZEP 用) は MQZID_REFRESH_CACHE (8L) です。

構文

MQZ_REFRESH_CACHE

(QMgrName, ComponentData, Continuation, CompCode, Reason)

パラメーター

QMgrName (MQCHAR48) - 入力

キュー・マネージャー名。

コンポーネントを呼び出すキュー・マネージャーの名前。この名前は、パラメーターがすべて埋まるまで空白が埋め込まれます。名前の最後をヌル文字にすることはできません。

キュー・マネージャー名は、情報としてコンポーネントに渡されます。許可サービス・インターフェースでは、コンポーネントは定義されている方法でこの情報を使用する必要はありません。

ComponentData (MQBYTE x ComponentDataLength) - 入出力

コンポーネント・データ。

このデータは、この特定のコンポーネントのためにキュー・マネージャーで保持されます。このコンポーネントに用意されている関数によって変更された内容がすべて保持され、このコンポーネントのいずれかの関数が次回に呼び出されると提示されます。

このデータ域の長さは、キュー・マネージャーにより MQZ_INIT_AUTHORITY 呼び出しの ComponentDataLength パラメーターに入れられて渡されます。

Continuation (MQLONG) - 出力

コンポーネントによって設定される継続標識。

指定可能な値は、次のとおりです。

MQZCI_DEFAULT

キュー・マネージャーに依存する継続。

MQZ_REFRESH_CACHE の場合、MQZCI_CONTINUE と同じ効果があります。

MQZCI_CONTINUE

次のコンポーネントに継続。

MQZCI_STOP

次のコンポーネントに継続しない。

CompCode (MQLONG) - 出力

完了コード。

これは、以下のいずれかになります。

MQCC_OK

正常終了。

MQCC_FAILED

呼び出し失敗。

Reason (MQLONG) - 出力

CompCode を限定する理由コード。

CompCode が MQCC_OK の場合、次のようになります。

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_FAILED の場合、次のようになります。

MQRC_SERVICE_ERROR

(2289, X'8F1') サービスのアクセスで、予期しないエラーが発生しました。

C 言語での呼び出し

```
MQZ_REFRESH_CACHE (QMgrName, ComponentData,  
                   &Continuation, &CompCode, &Reason);
```

パラメーターを次のように宣言します。

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQBYTE    ComponentData[n];   /* Component data */  
MQLONG     Continuation;      /* Continuation indicator set by  
                               component */  
MQLONG     CompCode;          /* Completion code */  
MQLONG     Reason;           /* Reason code qualifying CompCode */
```

IBM i での MQZ_SET_AUTHORITY (権限の設定)

この関数は、MQZAS_VERSION_1 許可サービス・コンポーネントに用意されています。キュー・マネージャーはこの関数を呼び出して、指定されたオブジェクトにエンティティーがアクセスするための権限を設定します。

この関数の関数 ID (MQZEP 用) は、MQZID_SET_AUTHORITY です。

注: この関数は、既存のすべての権限を指定変更します。既存の権限を維持するには、この関数を使用して再度設定する必要があります。

構文

MQZ_SET_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName, ObjectType, Authority, ComponentData, Continuation, CompCode, Reason)

パラメーター

MQZ_SET_AUTHORITY 呼び出しには、以下のパラメーターがあります。

QMgrName (MQCHAR48) - 入力

キュー・マネージャー名。

コンポーネントを呼び出すキュー・マネージャーの名前。この名前は、パラメーターがすべて埋まるまでブランクが埋め込まれます。名前の最後をヌル文字にすることはできません。

キュー・マネージャー名は、情報としてコンポーネントに渡されます。許可サービス・インターフェースでは、コンポーネントは定義されている方法でこの情報を使用する必要はありません。

EntityName (MQCHAR12) - 入力

エンティティー名。

オブジェクトに対するアクセス権を設定する エンティティの名前。ストリングの長さは最大で 12 文字です。12 文字未満の場合、その名前の右側が空白で埋められます。名前の最後をヌル文字にすることはできません。

EntityType (MQLONG) - 入力

エンティティ・タイプ。

EntityName によって指定されるエンティティのタイプ。次の値を指定できます。

MQZAET_PRINCIPAL

プリンシパル。

MQZAET_GROUP

グループ。

ObjectName (MQCHAR48) - 入力

オブジェクト名

アクセスが必要なオブジェクトの名前。ストリングの長さは最大で 48 文字です。48 文字未満の場合、その名前の右側が空白で埋められます。名前の最後をヌル文字にすることはできません。

ObjectType が MQOT_Q_MGR の場合、この名前は *QMgrName* と同じになります。

ObjectType (MQLONG) - 入力

オブジェクト・タイプ

ObjectName によって指定されるエンティティのタイプ。これは、以下のいずれかになります。

MQOT_AUTH_INFO

認証情報

MQOT_CHANNEL

チャンネル。

MQOT_CLNTCONN_CHANNEL

クライアント接続チャンネル。

MQOT_LISTENER

リスナー

MQOT_NAMELIST

名前リスト。

MQOT_PROCESS

プロセス定義。

MQOT_Q

キュー。

MQOT_Q_MGR

キュー・マネージャー。

MQOT_SERVICE

サービス

Authority (MQLONG) - 入力

検査される権限。

1 つの許可を設定する場合、このフィールドは該当する許可操作と同じです (MQZAO_* 定数)。複数の許可を設定する場合、対応する MQZAO_* 定数とビットごとに OR 演算します。

ComponentData (MQBYTE x ComponentDataLength) - 入出力

コンポーネント・データ。

このデータは、この特定のコンポーネントのためにキュー・マネージャーによって保持されます。このコンポーネントによって提供される関数のいずれかによってそのデータに変更が加えられると、その変更は保存され、次回このコンポーネントの関数の 1 つが呼び出されたときに提供されます。

このデータ域の長さは、キュー・マネージャーにより MQZ_INIT_AUTHORITY 呼び出しの **ComponentDataLength** パラメーターに入れられて渡されます。

Continuation (MQLONG) - 出力

コンポーネントによって設定される継続標識。

指定可能な値は、次のとおりです。

MQZCI_DEFAULT

キュー・マネージャーに依存する継続。

MQZ_SET_AUTHORITY の場合、MQZCI_STOP と同じ効果があります。

MQZCI_CONTINUE

次のコンポーネントに継続。

MQZCI_STOP

次のコンポーネントに継続しない。

CompCode (MQLONG) - 出力

完了コード。

これは、以下のいずれかになります。

MQCC_OK

正常終了。

MQCC_FAILED

呼び出し失敗。

Reason (MQLONG) - 出力

CompCode を限定する理由コード。

CompCode が MQCC_OK の場合:

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_FAILED の場合:

MQRC_NOT_AUTHORIZED

(2035, X'7F3') アクセスは許可されません。

MQRC_SERVICE_ERROR

(2289, X'8F1') サービスのアクセスで、予期しないエラーが発生しました。

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') 基本サービスを使用できません。

MQRC_UNKNOWN_ENTITY

(2292, X'8F4') サービスに対して不明なエンティティです。

C 言語での呼び出し

```
MQZ_SET_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName,  
                   ObjectType, Authority, ComponentData,  
                   &Continuation, &CompCode, &Reason);
```

サービスに渡されるパラメーターは次のように宣言されます。

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQCHAR12  EntityName;         /* Entity name */  
MQLONG    EntityType;         /* Entity type */  
MQCHAR48  ObjectName;         /* Object name */  
MQLONG    ObjectType;         /* Object type */  
MQLONG    Authority;          /* Authority to be checked */  
MQBYTE    ComponentData[n];   /* Component data */  
MQLONG    Continuation;       /* Continuation indicator set by  
                               component */  
MQLONG    CompCode;           /* Completion code */  
MQLONG    Reason;             /* Reason code qualifying CompCode */
```


MQZ_TERM_AUTHORITY - 許可サービスの終了

この関数は許可サービス・コンポーネントに用意されています。キュー・マネージャーはこのコンポーネントのサービスがなくなった時に、この関数を呼び出します。この関数はコンポーネントに必要なすべてのクリーンアップを実行する必要があります。

この関数の関数 ID (MQZEP 用) は、MQZID_TERM_AUTHORITY です。

構文

MQZ_TERM_AUTHORITY (*Hconfig, Options, QMgrName, ComponentData, CompCode, Reason*)

パラメーター

MQZ_TERM_AUTHORITY 呼び出しには、以下のパラメーターがあります。

Hconfig (MQHCONFIG) - 入力

構成ハンドル。

このハンドルは、終了する特定のコンポーネントを表します。

Options (MQLONG) - 入力

終了オプション。

これは、以下のいずれかになります。

MQZTO_PRIMARY

1 次終了。

MQZTO_SECONDARY

2 次終了。

QMgrName (MQCHAR48) - 入力

キュー・マネージャー名。

コンポーネントを呼び出すキュー・マネージャーの名前。この名前は、パラメーターがすべて埋まるまでブランクが埋め込まれます。名前の最後をヌル文字にすることはできません。

キュー・マネージャー名は、情報としてコンポーネントに渡されます。許可サービス・インターフェースでは、コンポーネントは定義されている方法でこの情報を使用する必要はありません。

ComponentData (MQBYTE x ComponentDataLength) - 入出力

コンポーネント・データ。

このデータは、この特定のコンポーネントのためにキュー・マネージャーによって保持されます。このコンポーネントによって提供される関数のいずれかによってそのデータに変更が加えられると、その変更は保存され、次回このコンポーネントの関数の 1 つが呼び出されたときに提供されます。

このデータ域の長さは、キュー・マネージャーにより MQZ_INIT_AUTHORITY 呼び出しの **ComponentDataLength** パラメーター内に渡されます。

MQZ_TERM_AUTHORITY 呼び出しが完了すると、キュー・マネージャーはこのデータを廃棄します。

CompCode (MQLONG) - 出力

完了コード。

これは、以下のいずれかになります。

MQCC_OK

正常終了。

MQCC_FAILED

呼び出し失敗。

Reason (MQLONG) - 出力

CompCode を限定する理由コード。

CompCode が MQCC_OK の場合:

MQRC_NONE

(0, X'000') レポートする理由コードはありません。

CompCode が MQCC_FAILED の場合:

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') 基本サービスを使用できません。

MQRC_TERMINATION_FAILED

(2287, X'8FF') 未定義の理由で終了に失敗しました。

これらの理由コードについて詳しくは、[メッセージおよび理由コード](#)を参照してください。

C 言語での呼び出し

```
MQZ_TERM_AUTHORITY (Hconfig, Options, QMgrName, ComponentData,  
                    &CompCode, &Reason);
```

サービスに渡されるパラメーターは次のように宣言されます。

```
MQHCONFIG  Hconfig;          /* Configuration handle */  
MQLONG     Options;          /* Termination options */  
MQCHAR48   QMgrName;         /* Queue manager name */  
MQBYTE     ComponentData[n]; /* Component data */  
MQLONG     CompCode;         /* Completion code */  
MQLONG     Reason;           /* Reason code qualifying CompCode */
```

IBM i IBM i での MQZAC (アプリケーション・コンテキスト)

このパラメーターは、呼び出し側アプリケーションに関連するデータを指定します。

MQZAC 構造体は、MQZ_AUTHENTICATE_USER 呼び出しの **ApplicationContext** パラメーターに使用します。

フィールド

StrucId (MQCHAR4)

構造体 ID

その値は、以下のものです。

MQZAC_STRUC_ID

アプリケーション・コンテキスト構造体の ID。

C 言語の場合、定数 MQZAC_STRUC_ID_ARRAY も定義されます。これは MQZAC_STRUC_ID と同じ値ですが、ストリングではなく文字の配列です。

これは、サービスの入力フィールドです。

Version (MQLONG)

構造体のバージョン番号。

その値は、以下のものです。

MQZAC_VERSION_1

バージョン 1 のアプリケーション・コンテキスト構造体。

以下の定数は、現行バージョンのバージョン番号を指定しています。

MQZAC_CURRENT_VERSION

アプリケーション・コンテキスト構造体の現行バージョン。

これは、サービスの入力フィールドです。

ProcessId (MQPID)

プロセス ID。

アプリケーションのプロセス ID。

ThreadId (MQTID)

スレッド ID。

アプリケーションのスレッド ID。

ApplName (MQCHAR28)

アプリケーション名。

アプリケーション名。

UserID (MQCHAR12)

ユーザー ID。

IBM i システムの場合、アプリケーション・ジョブの作成時に使用されたユーザー・プロファイル。
(IBM i では、アプリケーション・ジョブで QWTSETP API によってプロファイル・スワップが実行されると、現行ユーザー・プロファイルが返されます。)

EffectiveUserID (MQCHAR12)

有効ユーザー ID。

IBM i システムの場合は、アプリケーション・ジョブの現行ユーザー・プロファイル。

Environment (MQLONG)

環境。

このフィールドで、呼び出しの実行元の環境を指定します。

これは、以下の値のいずれかになります。

MQXE_COMMAND_SERVER

コマンド・サーバー。

MQXE_MQSC

runmqsc コマンド・インタープリター。

MQXE_MCA

MSG チャネル・エージェント

MQXE_OTHER

未定義の環境

CallerType (MQLONG)

呼び出し元のタイプ。

このフィールドで、呼び出しの実行元プログラムのタイプを指定します。

これは、以下の値のいずれかになります。

MQXACT_EXTERNAL

呼び出しはキュー・マネージャーの外部です。

MQXACT_INTERNAL

呼び出しはキュー・マネージャーの内部です。

AuthenticationType (MQLONG)

認証のタイプ。

このフィールドで、実行される認証のタイプを指定します。

これは、以下の値のいずれかになります。

MQZAT_INITIAL_CONTEXT

認証呼び出しは、ユーザー・コンテキストが初期化されることに起因します。この値は、MQCONN または MQCONNX 呼び出しの間に使用されます。

MQZAT_CHANGE_CONTEXT

この認証呼び出しは、ユーザー・コンテキストが変更されることに起因します。この値は、MCA がユーザー・コンテキストを変更するときに使用されます。

V

BindType (MQLONG)

バインドのタイプ。

このフィールドは、使用中のバインディング・タイプを指定します。

これは、以下の値のいずれかになります。

MQCNO_FASTPATH_BINDING

ファースト・パス・バインディング。

MQCNO_SHARED_BINDING

共有バインディング。

MQCNO_ISOLATED_BINDING

分離されたバインディング。

C 宣言

```
typedef struct tagMQZAC MQZAC;  
struct tagMQZAC {  
    MQCHAR4    StrucId;           /* Structure identifier */  
    MQLONG     Version;           /* Structure version number */  
    MQPID      ProcessId;         /* Process identifier */  
    MQTID      ThreadId;          /* Thread identifier */  
    MQCHAR28    ApplName;         /* Application name */  
    MQCHAR12    UserID;           /* User identifier */  
    MQCHAR12    EffectiveUserID;   /* Effective user identifier */  
    MQLONG      Environment;       /* Environment */  
    MQLONG      CallerType;        /* Caller type */  
    MQLONG      AuthenticationType; /* Authentication type */  
    MQLONG      BindType;          /* Bind type */  
};
```

IBM i IBM i での MQZAD (権限データ)

MQZAD 構造体は、MQZ_ENUMERATE_AUTHORITY_DATA 呼び出しの 2 つのパラメーターに使用します。

Filter および **AuthorityBuffer** パラメーターについて詳しくは、[1700 ページの『IBM i での MQZ_ENUMERATE_AUTHORITY_DATA \(権限データの列挙\)』](#)を参照してください。

- MQZAD は、呼び出しへの入力である **Filter** パラメーターで使用されます。このパラメーターは、この呼び出しによって戻される権限データの選択で使用される選択基準を指定します。
- MQZAD は、呼び出しからの出力である **AuthorityBuffer** パラメーターでも使用されます。このパラメーターでは、プロファイル名、オブジェクト・タイプ、およびエンティティの 1 つの組み合わせの許可を指定します。

フィールド

StrucId (MQCHAR4)

構造体 ID

その値は、以下のものです。

MQZAD_STRUC_ID

権限データ構造体の ID。

C 言語の場合、定数 MQZAD_STRUC_ID_ARRAY も定義されます。これは MQZAD_STRUC_ID と同じ値ですが、ストリングではなく文字の配列です。

これは、サービスの入力フィールドです。

Version (MQLONG)

構造体のバージョン番号。

その値は、以下のものです。

MQZAD_VERSION_1

バージョン 1 の権限データ構造体。

以下の定数は、現行バージョンのバージョン番号を指定しています。

MQZAD_CURRENT_VERSION

権限データ構造体の現行バージョン。

これは、サービスの入力フィールドです。

ProfileName (MQCHAR48)

プロファイル名。

Filter パラメーターの場合、このフィールドには権限データを必要とするプロファイル名が入ります。名前がフィールドの最後までまたは最初のヌル文字までブランクの場合、すべてのプロファイル名の権限データが返されます。

AuthorityBuffer パラメーターの場合、このフィールドは、指定された選択基準と突き合わせるプロファイルの名前です。

ObjectType (MQLONG)

オブジェクト・タイプ

Filter パラメーターの場合、このフィールドは、権限データを必要とするオブジェクト・タイプです。値が MQOT_ALL の場合、すべてのオブジェクト・タイプの権限データが返されます。

AuthorityBuffer パラメーターの場合、このフィールドは、**ProfileName** によって識別されるプロファイルが適用されるオブジェクト・タイプです。

値は以下のいずれかです。**Filter** パラメーターの場合、値 MQOT_ALL も有効です。

MQOT_AUTH_INFO

認証情報

MQOT_CHANNEL

チャンネル。

MQOT_CLNTCONN_CHANNEL

クライアント接続チャンネル。

MQOT_LISTENER

リスナー

MQOT_NAMELIST

名前リスト。

MQOT_PROCESS

プロセス定義。

MQOT_Q

キュー。

MQOT_Q_MGR

キュー・マネージャー。

MQOT_SERVICE

サービス

Authority (MQLONG)

権限。

Filter パラメーターの場合、このフィールドは無視されます。

AuthorityBuffer パラメーターの場合、このフィールドは **ProfileName** および **ObjectType** で識別されるオブジェクトに対してエンティティが保持している許可を表します。エンティティの権限が 1 つのみである場合、このフィールドは該当する許可値 (MQZAO_* 定数) に等しくなります。エンティティが複数の権限を持つ場合、このフィールドは対応する MQZAO_* 定数のビット単位 OR になります。

EntityDataPtr (PMQZED)

エンティティを識別する MQZED 構造体のアドレス。

Filter パラメーターの場合、このフィールドは権限データを必要とするエンティティを識別する MQZED 構造体を指します。 **EntityDataPtr** がヌル・ポインターの場合、すべてのエンティティの権限データが返されます。

AuthorityBuffer パラメーターの場合、このフィールドは権限データが返される元のエンティティを識別する MQZED 構造体を指します。

EntityType (MQLONG)

エンティティ・タイプ。

Filter パラメーターの場合、このフィールドは、権限データを必要とするエンティティ・タイプを指定します。値が MQZAET_NONE の場合、すべてのエンティティ・タイプの権限データが戻されます。

AuthorityBuffer パラメーターの場合、このフィールドは、 **EntityDataPtr** が指す MQZED 構造体によって識別されるエンティティのタイプを指定します。

値は以下のいずれかです。 **Filter** パラメーターの場合、値 MQZAET_NONE も有効です。

MQZAET_PRINCIPAL

プリンシパル。

MQZAET_GROUP

グループ。

Options (MQAUTHOPT)

オプション。

このフィールドは、表示されるプロファイルに制御を与えるオプションを指定します。

以下のいずれかを指定しなければなりません。

MQAUTHOPT_NAME_ALL_MATCHING

すべてのプロファイルを表示します。

MQAUTHOPT_NAME_EXPLICIT

ProfileName フィールドで指定されたものと完全に一致する名前を持つプロファイルを表示します。

さらに、次のいずれか 1 つも指定する必要があります。

MQAUTHOPT_ENTITY_SET

ProfileName で指定されたオブジェクトに対してエンティティが持つ累積権限を計算するために使用されるすべてのプロファイルを表示します。 **ProfileName** フィールドにワイルドカード文字を含めることはできません。

- 指定されたエンティティがプリンシパルの場合、セット {エンティティ、グループ} のそれぞれのメンバーごとに、オブジェクトに適合する最も適当なプロファイルが表示されます。
- 指定されたエンティティがグループの場合、オブジェクトに適合するグループの中から、最も適当なプロファイルが表示されます。
- この値を指定する場合、 **ProfileName**、 **ObjectType**、 **EntityType**、および **EntityDataPtr** MQZED 構造体に指定するエンティティ名の値は、すべて非ブランクでなければなりません。

MQAUTHOPT_NAME_ALL_MATCHING を指定した場合は、以下も指定できます。

MQAUTHOPT_ENTITY_EXPLICIT

EntityDataPtr MQZED 構造体に指定されたエンティティ名と完全に一致するエンティティ名を持つプロファイルを表示します。

C 宣言

```
typedef struct tagMQZAD MQZAD;  
struct tagMQZAD {
```

```

MQCHAR4   StrucId;           /* Structure identifier */
MQLONG    Version;          /* Structure version number */
MQCHAR48   ProfileName;     /* Profile name */
MQLONG     ObjectType;      /* Object type */
MQLONG     Authority;       /* Authority */
PMQZED     EntityDataPtr;    /* Address of MQZED structure identifying an
                               entity */
MQLONG     EntityType;      /* Entity type */
MQAUTHOPT  Options;         /* Options */
};

```

IBM i IBM i での MQZED (エンティティ記述子)

MQZED 構造体は、いくつかの許可サービス呼び出しの中で、許可を検査するエンティティを指定するために使用されます。

フィールド

StrucId (MQCHAR4)

構造体 ID

その値は、以下のものです。

MQZED_STRUC_ID

エンティティ記述子構造体の ID。

C 言語の場合、定数 MQZED_STRUC_ID_ARRAY も定義されます。これは MQZED_STRUC_ID と同じ値ですが、ストリングではなく文字の配列です。

これは、サービスの入力フィールドです。

Version (MQLONG)

構造体のバージョン番号。

その値は、以下のものです。

MQZED_VERSION_1

バージョン 1 エンティティ記述子構造体

以下の定数は、現行バージョンのバージョン番号を指定しています。

MQZED_CURRENT_VERSION

エンティティ記述子構造体の現行バージョン。

これは、サービスの入力フィールドです。

EntityNamePtr (PMQCHAR)

エンティティ名のアドレス。

これは、許可が検査されるエンティティの名前を指すポインターです。

EntityDomainPtr (PMQCHAR)

エンティティ・ドメイン・ネームのアドレス。

これは、許可が検査されるエンティティの定義を含むドメインの名前を指すポインターです。

SecurityId (MQBYTE40)

セキュリティ ID。

これは、許可が検査されるセキュリティ ID です。

CorrelationPtr (MQPTR)

相関ポインター。

これを使用すると、認証ユーザー関数とその他の適切な OAM 関数との間で相関データを引き渡しやすくなります。

C 宣言

```
typedef struct tagMQZED MQZED;
struct tagMQZED {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    PMQCHAR    EntityNamePtr;    /* Address of entity name */
    PMQCHAR    EntityDomainPtr; /* Address of entity domain name */
    MQBYTE40   SecurityId;       /* Security identifier */
    MQPTR      CorrelationPtr;   /* Address of correlation data */
};
```

IBM i IBM i での MQZFP (解放パラメーター)

このパラメーターは、解放されるリソースに関連するデータを指定します。

MQZFP 構造体は、MQZ_FREE_USER 呼び出しの **FreeParms** パラメーターに使用します。

フィールド

StrucId (MQCHAR4)

構造体 ID

その値は、以下のものです。

MQZFP_STRUC_ID

解放パラメーター構造体の ID。

C 言語の場合、定数 MQZFP_STRUC_ID_ARRAY も定義されます。これは MQZFP_STRUC_ID と同じ値ですが、ストリングではなく文字の配列です。

これは、サービスの入力フィールドです。

Version (MQLONG)

構造体のバージョン番号。

その値は、以下のものです。

MQZFP_VERSION_1

バージョン 1 解放パラメーター構造体。

以下の定数は、現行バージョンのバージョン番号を指定しています。

MQZFP_CURRENT_VERSION

解放パラメーター構造体の現行バージョン。

これは、サービスの入力フィールドです。

Reserved (MQBYTE8)

予約フィールド。

初期値はヌルです。

CorrelationPtr (MQPTR)

相関ポインター。

解放されるリソースに関連する相関データのアドレス。

C 宣言

```
typedef struct tagMQZFP MQZFP;
struct tagMQZFP {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQBYTE8     Reserved;        /* Reserved field */
    MQPTR      CorrelationPtr;   /* Address of correlation data */
};
```


MQZIC 構造体は、MQZ_AUTHENTICATE_USER 呼び出しの **IdentityContext** パラメーターに使用します。

MQZIC 構造体には、メッセージを最初にキューに書き込んだアプリケーション・ユーザーを識別する ID コンテキスト情報が格納されます。

- キュー・マネージャーは **UserIdentifier** フィールドにユーザーを識別する名前を入力しますが、キュー・マネージャーがこれを実行する方法は、アプリケーションが実行される環境によって異なります。
- キュー・マネージャーは、**AccountingToken** フィールドに、そのメッセージを書き込んだアプリケーションから判別したトークンまたは番号を入力します。
- アプリケーションは、ユーザーに関して追加する必要がある追加の情報 (例えば、暗号化されたパスワード) 入力用として **ApplIdentityData** フィールドを使用できます。

適切な権限があるアプリケーションは、MQZ_AUTHENTICATE_USER 関数を使用して ID コンテキストを設定できます。

メッセージが **AccountingToken** で作成されると、システムセキュリティ識別子 IBM MQ for Windows (Windows SID) がフィールドに格納されます。SID の使用目的は、**UserIdentifier** フィールドの補足とユーザーの資格情報の確立です。

フィールド

StrucId (MQCHAR4)

構造体 ID

その値は、以下のものです。

MQZIC_STRUC_ID

ID コンテキスト構造体の ID。

C 言語の場合、定数 **MQZIC_STRUC_ID_ARRAY** も定義されます。これは **MQZIC_STRUC_ID** と同じ値ですが、ストリングではなく文字の配列です。

これは、サービスの入力フィールドです。

Version (MQLONG)

構造体のバージョン番号。

その値は、以下のものです。

MQZIC_VERSION_1

バージョン 1 の ID コンテキスト構造体。

以下の定数は、現行バージョンのバージョン番号を指定しています。

MQZIC_CURRENT_VERSION

ID コンテキスト構造体の現行バージョン。

これは、サービスの入力フィールドです。

UserIdentifier (MQCHAR12)

ユーザー ID。

これは、メッセージの **ID コンテキスト** の一部です。

UserIdentifier には、メッセージを発信したアプリケーションのユーザー ID を指定します。キュー・マネージャーはこの情報を文字データとして扱いますが、そのフォーマットの定義はしません。*UserIdentifier* フィールドについて詳しくは、[424 ページの『UserIdentifier \(MQCHAR12\)』](#)を参照してください。

AccountingToken (MQBYTE32)

アカウントリング・トークン。

これは、メッセージの **ID コンテキスト** の一部です。

AccountingToken を指定すると、メッセージに適切な課金が行われた結果として、アプリケーションでの作業完了が許可されます。キュー・マネージャーはこの情報をビット・ストリングとして処理し、その内容は検査しません。*AccountingToken* フィールドについて詳しくは、[425 ページの『AccountingToken \(MQBYTE32\)』](#)を参照してください。

ApplIdentityData (MQCHAR32)

ID に関連するアプリケーション・データ。

これは、メッセージの **ID コンテキスト**の一部です。

ApplIdentityData は、アプリケーション・スイートにより定義される情報で、メッセージの発信元についての追加情報を提供するのに使用できます。例えば、ID データが信頼 できるかどうかを示すために、適切なユーザー権限で実行されているアプリケーションにより設定が可能です。

ApplIdentityData フィールドについて詳しくは、[427 ページの『ApplIdentityData \(MQCHAR32\)』](#)を参照してください。

C 宣言

```
typedef struct tagMQZED MQZED;
struct tagMQZED {
    MQCHAR4    StrucId;          /* Structure identifier */
    MQLONG     Version;         /* Structure version number */
    MQCHAR12   UserIdentifier;   /* User identifier */
    MQBYTE32   AccountingToken; /* Accounting token */
    MQCHAR32   ApplIdentityData; /* Application data relating to identity */
};
```

IBM MQ .NET クラスとインターフェース

IBM MQ .NET クラスとインターフェースはアルファベット順にリストされています。プロパティ、メソッド、およびコンストラクターについての説明があります。

MQAsyncStatus.NET クラス

MQAsyncStatus を使用して、以前の MQI アクティビティの状況を照会します。例えば、以前の非同期 PUT 操作の成功について照会します。*MQAsyncStatus* は、MQSTS データ構造のフィーチャーをカプセル化します。

クラス

```
System.Object
├── IBM.WMQ.MQBase
│   └── IBM.WMQ.MQBaseObject
│       └── IBM.WMQ.MQAsyncStatus
```

```
public class IBM.WMQ.MQAsyncStatus extends IBM.WMQ.MQBaseObject;
```

- [1726 ページの『プロパティ』](#)
- [1727 ページの『コンストラクター』](#)

プロパティ

プロパティの取得時にスローされた *MQException* をテストします。

```
public static int CompCode {get;}
```

最初のエラーまたは警告からの完了コード。

```
public static int Reason {get;}
```

The reason code from the first error or warning.

```
public static int PutSuccessCount {get;}
```

成功した非同期 MQI 書き込み呼び出しの回数。

```
public static int PutWarningCount {get;}
```

警告付きで成功した非同期 MQI 書き込み呼び出しの回数。

```
public static int PutFailureCount {get;}
```

失敗した非同期 MQI 書き込み呼び出しの回数。

```
public static int ObjectType {get;}
```

最初のエラーのオブジェクト・タイプ。属性の値は以下のとおりです。

- MQC.MQOT_ALIAS_Q
- MQC.MQOT_LOCAL_Q
- MQC.MQOT_MODEL_Q
- MQC.MQOT_Q
- MQC.MQOT_REMOTE_Q
- MQC.MQOT_TOPIC
- 0、オブジェクトが返されないことを示しています

```
public static string ObjectName {get;}
```

オブジェクト名。

```
public static string ObjectQMgrName {get;}
```

オブジェクト・キュー・マネージャー名。

```
public static string ResolvedObjectName {get;}
```

解決済みオブジェクト名。

```
public static string ResolvedObjectQMgrName {get;}
```

解決済みオブジェクト・キュー・マネージャー名。

コンストラクター

```
public MQAsyncStatus() throws MQException;
```

コンストラクター・メソッド。必要に応じてフィールドをゼロまたはブランクに初期化して、オブジェクトを作成します。

MQAuthenticationInformationRecord.NET クラス

MQAuthenticationInformationRecord は、IBM MQ TLS クライアント接続で使用する認証子に関する情報を指定するために使用します。MQAuthenticationInformationRecord は、認証情報レコード、MQAIR をカプセル化します。

クラス

```
System.Object
├─ IBM.WMQ.MQAuthenticationInformationRecord
```

```
public class IBM.WMQ.MQAuthenticationInformationRecord extends System.Object;
```

- [1728 ページの『プロパティ』](#)
- [1728 ページの『コンストラクター』](#)

プロパティ

プロパティの取得時にスローされた `MQException` をテストします。

`public long Version {get; set;}`

構造体のバージョン番号。

`public long AuthInfoType {get; set;}`

認証情報のタイプ。この属性は、次のいずれかの値に設定する必要があります。

- OCSP - 証明書取り消し状況の検査は、OCSP を使用して行われます。
- CRLLDAP - 証明書取り消し状況の検査は、LDAP サーバー上の証明書取り消しリストを使用して行われます。

`public string AuthInfoConnName {get; set;}`

LDAP サーバーが動作しているホストの DNS 名または IP アドレス。オプションでポート番号が付きます。このキーワードは必須です。

`public string LDAPPassword {get; set;}`

LDAP サーバーにアクセスしているユーザーの識別名に関連付けられるパスワード。このプロパティは、**`AuthInfoType`** が CRLLDAP に設定されている場合にのみ適用されます。

`public string LDAPUserName {get; set;}`

LDAP サーバーにアクセスしているユーザーの識別名。このプロパティを設定する場合、**`LDAPUserNameLength`** および **`LDAPUserNamePtr`** は自動的に正しく設定されます。このプロパティは、**`AuthInfoType`** が CRLLDAP に設定されている場合にのみ適用されます。

`public string OCSPResponderURL {get; set;}`

OCSP 応答側に接続可能な URL。このプロパティは、**`AuthInfoType`** が OCSP に設定されている場合にのみ適用されます。

このフィールドでは大文字と小文字が区別されます。先頭は、小文字のストリング `http://` にする必要があります。URL の残りの部分では、OCSP サーバー実装環境によっては、大文字小文字が区別されることがあります。

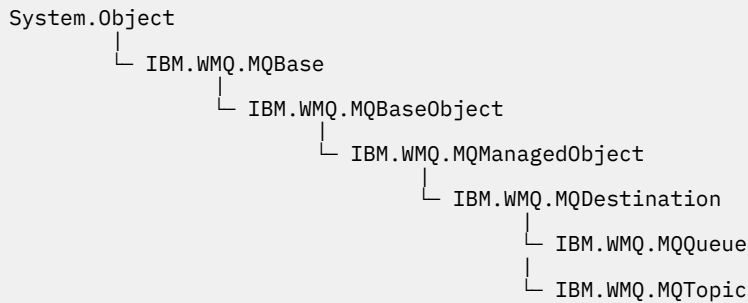
コンストラクター

`MQAuthenticationInformationRecord();`

MQDestination.NET クラス

`MQDestination` を使用して、`MQQueue` および `MQTopic` に共通のメソッドにアクセスします。`MQDestination` は、抽象基本クラスであり、インスタンス化することはできません。

クラス



```
public class IBM.WMQ.MQDestination extends IBM.WMQ.MQManagedObject;
```

- [1729 ページの『プロパティ』](#)
- [1729 ページの『方法』](#)
- [1731 ページの『コンストラクター』](#)

プロパティ

プロパティの取得時にスローされた `MQException` をテストします。

```
public DateTime CreationDateTime {get;}
```

キューまたはトピックが作成された日時。元は、`MQQueue` の中に含まれていたこのプロパティは、基本の `MQDestination` クラスに移動されました。

デフォルト値はありません。

```
public int DestinationType {get;}
```

使用される宛先のタイプを記述する整数値。サブクラス・コンストラクター `MQQueue` または `MQTopic` から初期化され、この値は、次の値のいずれかをとることができます。

- `MQOT_Q`
- `MQOT_TOPIC`

デフォルト値はありません。

方法

```
public void Get(MQMessage message);  
public void Get(MQMessage message, MQGetMessageOptions getMessageOptions);  
public void Get(MQMessage message, MQGetMessageOptions getMessageOptions, int  
MaxMsgSize);
```

`MQException` をスローします。

宛先が `MQQueue` オブジェクトである場合はキューから、宛先が `MQTopic` オブジェクトである場合はトピックから、読み取り用の `MQGetMessageOptions` のデフォルト・インスタンスを使用してメッセージを読み取ります。

読み取りに失敗した場合は、`MQMessage` オブジェクトは変更されません。それが成功した場合は、`MQMessage` のメッセージ記述子とメッセージ・データ部分が、着信メッセージのメッセージ記述子とメッセージ・データに置き換わります。

IBM MQ への、特定の `MQQueueManager` からの呼び出しは、すべて同期されています。そのため読み取り待機を実行すると、同じ `MQQueueManager` を使用しているすべてのスレッドについて、読み取りの呼び出しが確立するまで、次の IBM MQ の呼び出しが行われません。同時に IBM MQ にアクセスするために複数のスレッドが必要である場合、各スレッドがそれ自身の `MQQueueManager` オブジェクトを作成する必要があります。

メッセージ

メッセージ記述子と返されたメッセージ・データを含んでいます。メッセージ記述子のいくつかのフィールドは、入力パラメーターです。MessageId および CorrelationId 入力パラメーターが、必須として設定されているのを確認することが重要です。

再接続可能なクライアントは再接続に成功すると、理由コード MQRC_BACKED_OUT を、MQGM_SYNCPOINT で受信したメッセージに対して返します。

getMessageOptions

読み取りのアクションを制御するオプション。

オプション MQC.MQGM_CONVERT を使用すると、結果として、1 バイト文字コードを 2 バイトコードに変換するときに、理由コード MQC.MQRC_CONVERTED_STRING_TOO_BIG の例外になる可能性があります。この場合、メッセージは変換されずにバッファーにコピーされます。

getMessageOptions が指定されていない場合、使用されるメッセージ・オプションは MQGM_NOWAIT です。

再接続可能なクライアントで MQGM_LOGICAL_ORDER オプションを使用すると、MQRC_RECONNECT_INCOMPATIBLE 理由コードが返されます。

MaxMsgSize

このメッセージ・オブジェクトで受信する最大メッセージ。キューのメッセージがこのサイズより大きい場合、次の 2 つのいずれかが行われます。

- MQGM_ACCEPT_TRUNCATED_MSG フラグが MQGetMessageOptions オブジェクトに設定された場合、メッセージは、最大限までメッセージ・データで埋められます。例外が、MQCC_WARNING 完了コードと MQRC_TRUNCATED_MSG_ACCEPTED 理由コードと共にスローされます。
- MQGM_ACCEPT_TRUNCATED_MSG フラグが設定されていない場合、メッセージはキューに残されます。例外が、MQCC_WARNING 完了コードと MQRC_TRUNCATED_MSG_FAILED 理由コードと共にスローされます。

MaxMsgSize が指定されていない場合、メッセージ全体が取得されます。

```
public void Put(MQMessage message);  
public void Put(MQMessage message, MQPutMessageOptions putMessageOptions);
```

MQException をスローします。

宛先が MQQueue オブジェクトである場合はメッセージをキューに書き込みます。宛先が MQTopic オブジェクトである場合はトピックにメッセージを公開します。

書き込み呼び出しが成功した後に MQMessage オブジェクトを変更しても、IBM MQ キューまたはパブリケーション・トピックの実際のメッセージには影響がありません。

Put により、MQMessage の MessageId プロパティと CorrelationId プロパティを更新します。メッセージ・データはクリアされません。Put または Get 呼び出しは、MQMessage オブジェクト内の更新情報を参照します。例えば、次のコード・スニペットでは、最初のメッセージが a を含み、2 番目のメッセージが ab を含んでいます。

```
msg.WriteString("a");  
q.Put(msg, pmo);  
msg.WriteString("b");  
q.Put(msg, pmo);
```

メッセージ

メッセージ記述子データを含む MQMessage オブジェクト、および送信されるメッセージ。このメソッドの結果、メッセージ記述子は変更することができます。このメソッドの完了直後のメッセージ記述子の値は、キューに書き込みされた値か、トピックに公開された値です。

再接続可能なクライアントには、次の理由コードが返されます。

- MQRC_CALL_INTERRUPTED 持続メッセージで書き込みの呼び出しをしていて再接続が成功している間に、接続に失敗した場合。

- MQRC_NONE 非持続メッセージで書き込みの呼び出しをしている間、接続が成功している場合 (『Application Recovery』を参照)。

putMessageOptions

書き込みのアクションを制御するオプション。

putMessageOptions が指定されていない場合、MQPutMessageOptions のデフォルト・インスタンスが使用されます。

再接続可能なクライアントで MQPMO_LOGICAL_ORDER オプションを使用すると、MQRC_RECONNECT_INCOMPATIBLE 理由コードが返されます。

注: 簡単かつ効率的に単一メッセージをキューに書き込みするには、MQQueueManager.Put オブジェクトを使用します。このためには MQQueue オブジェクトを持っている必要があります。

コンストラクター

MQDestination は、抽象基本クラスであり、インスタンス化することはできません。MQQueue および MQTopic コンストラクターを使用するか、MQQueueManager.AccessQueue および MQQueueManager.AccessTopic methods を使用して、宛先にアクセスします。

MQEnvironment.NET クラス

MQEnvironment を使用して、MQQueueManager コンストラクターの呼び出し方法を制御し、IBM MQ MQI client 接続を選択します。MQEnvironment クラスには、IBM MQ の動作を制御するプロパティーが含まれます。

クラス

```
System.Object
└─ IBM.WMQ.MQEnvironment
```

```
public class IBM.WMQ.MQEnvironment extends System.Object;
```

- [1731 ページの『プロパティー - クライアントのみ』](#)
- [1732 ページの『プロパティー』](#)
- [1733 ページの『コンストラクター』](#)

プロパティー - クライアントのみ

プロパティーの取得時にスローされた MQException をテストします。

```
public static int CertificateValPolicy {get; set;}
```

リモート・パートナー・システムから受け取ったデジタル証明書を妥当性検査するために、どの TLS 証明書妥当性検査ポリシーを使用するかを設定します。有効な値は以下のとおりです。

- MQC.CERTIFICATE_VALIDATION_POLICY_ANY
- MQC.CERTIFICATE_VALIDATION_POLICY_RFC5280

```
public static ArrayList EncryptionPolicySuiteB {get; set;}
```

Suite B 準拠暗号方式のレベルを設定します。有効な値は以下のとおりです。

- MQC.MQ_SUITE_B_NONE - これはデフォルト値です。
- MQC.MQ_SUITE_B_128_BIT
- MQC.MQ_SUITE_B_192_BIT

public static string Channel {get; set;}

宛先キュー・マネージャーに接続するチャンネルの名前。クライアント・モードで MQQueueManager インスタンスを生成する前に、チャンネルのプロパティを設定する必要があります。

public static int FipsRequired {get; set;}

暗号処理が実行される場合、認証済みアルゴリズム FIPSIBM MQ のみを使用するように MQC.MQSSL_FIPS_YES 指定する。デフォルトは MQC.MQSSL_FIPS_NO です。

暗号ハードウェアが構成されている場合は、そのハードウェア製品に用意されている暗号モジュールが使用されます。使用しているハードウェアによっては、これらは特定のレベルで認証 FIPS されていない可能性があります。

public static string Hostname {get; set;}

IBM MQ サーバーが存在するコンピューターの TCP/IP ホスト名。ホスト名が設定されず、指定変更用のプロパティが設定されていない場合は、ローカル・キュー・マネージャーの接続にサーバー・バイインディング・モードが使用されます。

public static int Port {get; set;}

接続するポート。これは、IBM MQ サーバーが着信接続要求を listen するポートです。デフォルト値は 1414 です。

public static string SSLCipherSpec {get; set;}

接続に TLS を有効にするには、SSLCipherSpec を SVRCONN チャンネルで設定された CipherSpec の値に設定します。デフォルトはヌルで、接続に対して TLS は有効ではありません。

public static string sslPeerName {get; set;}

識別名パターン。sslCipherSpec が設定されている場合は、この変数を使用して、正しいキュー・マネージャーを確実に使用することができます。ヌル (デフォルト) に設定すると、キュー・マネージャーの DN は実行されません。sslPeerName は、sslCipherSpec がヌルの場合、無視されます。

public static string SSLKeyRepository {get; set;}

キーリポジトリにアクセスするための平文または暗号化されたパスフレーズ。キーリポジトリのパスフレーズは、クライアントアプリケーションが `runmqicred` ユーティリティを使用して利用するために暗号化されます。

null (デフォルト) に設定 SSLKeyRepositoryPassword されている場合、`MQKEYRPWD` 環境変数の値、またはクライアント設定ファイル内の属性 `SSLKeyRepositoryPassword` が使用されます。

public static string InitialKey {get; set;}

. で指定されたキーリポジトリのパス SSLKeyRepositoryPassword フレーズを暗号化するために使用された初期キー。

キーリポジトリのパスフレーズを `runmqicred` ユーティリティで暗号化する際に初期キーファイルを指定した場合、初期キーを指定する必要があります。

プロパティ

プロパティの取得時にスローされた MQException をテストします。

public static ArrayList HdrCompList {get; set;}

ヘッダー・データ圧縮リスト。

public static int KeyResetCount {get; set;}

秘密鍵が再折衝される前に、TLS 会話内で送受信する非暗号化バイト数を示します。

public static ArrayList MQAIRArray {get; set;}

MQAuthenticationInformationRecord オブジェクトの配列。

public static ArrayList MsgCompList {get; set;}

メッセージ・データ圧縮リスト。

public static string Password {get; set;}

認証されるパスワード。MQCSP 構造体から参照されるパスワードは、この Password プロパティを設定すると入力されます。

public static string ReceiveExit {get; set;}

受信出口により、キュー・マネージャーから受信されるデータを調べ、変更することができます。通常、キュー・マネージャーの対応する送信出口と組み合わせて使用します。ReceiveExit をヌルに設定すると、受信出口は呼び出されません。

public static string ReceiveUserData {get; set;}

受信出口に関連付けられたユーザー・データ。最大 32 文字です。

public static string SecurityExit {get; set;}

セキュリティ出口により、キュー・マネージャーへの接続を試みるときに発生するセキュリティ・フローをカスタマイズできます。SecurityExit をヌルに設定すると、セキュリティ出口は呼び出されません。

public static string SecurityUserData {get; set;}

セキュリティ出口に関連付けられたユーザー・データ。最大 32 文字です。

public static string SendExit {get; set;}

送信出口を使用することで、キュー・マネージャーに送信されたデータを調べたり、変更したりすることができます。通常、キュー・マネージャーの対応する受信出口と組み合わせて使用します。SendExit をヌルに設定すると、送信出口は呼び出されません。

public static string SendUserData {get; set;}

送信出口に関連付けられたユーザー・データ。最大 32 文字です。

public static string SharingConversations {get; set;}

この SharingConversations フィールドは、アプリケーション .NET からの接続において、それらのアプリケーションがクライアントチャンネル定義テーブル (CCDT) を使用していない場合に使用されます。

SharingConversations により、この接続に関連付けられたソケットで共有できる会話の最大数が決定されます。

値 0 は、会話の共有、先読み、およびハートビートに関して、IBM WebSphere MQ 7.0 以前と同じ動作をチャンネルが行うことを意味します。

フィールドは、キュー IBM MQ マネージャをインスタンス化する際に、SHARING_CONVERSATIONS_PROPERTY プロパティのハッシュテーブル内でとして渡されます。

SharingConversations を指定しないと、デフォルト値 10 が使用されます。

public static string SSLCryptoHardware {get; set;}

システム上に存在する暗号ハードウェアの構成に必要なパラメーター・ストリングを設定します。SSLCryptoHardware は、sslCipherSpec がヌルの場合、無視されます。

public static string SSLKeyRepository {get; set;}

キー・リポジトリの完全修飾ファイル名を設定します。

ファイル拡張子が指定されていない場合、デフォルトで .kdb

SSLKeyRepository をヌル (デフォルト) に設定した場合は、証明書の MQSSLKEYR 環境変数を使用して、キー・リポジトリの場所が探索されます。

public static string UserId {get; set;}

認証対象のユーザー ID。MQCSP 構造体から参照されるユーザー ID は、UserId を設定すると入力されます。API 出口またはセキュリティ出口を使用して、UserId を認証します。

コンストラクター

public MQEnvironment()

MQException.NET クラス

MQException は、障害が起きた IBM MQ 関数の完了コードおよび理由コードを検出するために使用します。IBM MQ エラーが発生するたびに、MQException がスローされます。

クラス

```
System.Object
├── System.Exception
│   └── System.ApplicationException
│       └── IBM.WMQ.MQException
```

```
public class IBM.WMQ.MQException extends System.ApplicationException;
```

- [1734 ページの『プロパティ』](#)
- [1734 ページの『コンストラクター』](#)

プロパティ

```
public int CompletionCode {get; set;}
```

エラーに関連付けられた IBM MQ 完了コード。指定できる値は以下のとおりです。

- MQException.MQCC_OK
- MQException.MQCC_WARNING
- MQException.MQCC_FAILED

```
public int ReasonCode {get; set;}
```

エラーを説明する IBM MQ 理由コード。

コンストラクター

```
public MQException(int completionCode, int reasonCode)
```

completionCode

IBM MQ 完了コード。

reasonCode

IBM MQ 完了コード。

MQGetMessageOptions.NET クラス

MQGetMessageOptions を使用して、メッセージを取得する方法を指定します。それによって、MQDestination.Get の動作を変更します。

クラス

```
System.Object
├── IBM.WMQ.MQBase
│   └── IBM.WMQ.MQBaseObject
│       └── IBM.WMQ.MQGetMessageOptions
```

```
public class IBM.WMQ.MQGetMessageOptions extends IBM.WMQ.MQBaseObject;
```

- [1735 ページの『プロパティ』](#)
- [1737 ページの『コンストラクター』](#)

プロパティ

注: このクラスで使用可能な一部のオプションの動作は、それらのオプションが使用される環境によって異なります。これらのエレメントは、アスタリスク (*) でマークを付けられます。

プロパティの取得時にスローされた `MQException` をテストします。

public int GroupStatus {get;}*

`GroupStatus` は、取り出されたメッセージが 1 つのグループに属しているかどうか、およびそれがグループの最後にあるものかどうかを示します。指定可能な値は以下のとおりです。

MQC.MQGS_LAST_MSG_IN_GROUP

メッセージはグループの最後にあるか、またはグループで唯一のメッセージである。

MQC.MQGS_MSG_IN_GROUP

メッセージは 1 つのグループに属しているが、グループの最後にあるものではない。

MQC.MQGS_NOT_IN_GROUP

メッセージは 1 つのグループに属していない。

public int MatchOptions {get; set;}*

`MatchOptions` は、メッセージの選択方法を決定します。以下のマッチング・オプションを設定できます。

MQC.MQMO_MATCH_CORREL_ID

一致する相関 ID。

MQC.MQMO_MATCH_GROUP_ID

一致するグループ ID。

MQC.MQMO_MATCH_MSG_ID

一致するメッセージ ID。

MQC.MQMO_MATCH_MSG_SEQ_NUMBER

一致するメッセージ順序番号。

MQC.MQMO_NONE

一致は不要です。

public int Options {get; set;}*

`Options` は、`MQQueue.get` のアクションを制御します。以下のいずれかの値を指定できます。複数のオプションが必要な場合は、値を追加するか、ビット単位 OR 演算子を使用して結合できます。

MQC.MQGMO_ACCEPT_TRUNCATED_MSG

メッセージ・データの切り捨てを許可します。

MQC.MQGMO_ALL_MSGS_AVAILABLE*

グループ内のすべてのメッセージが使用可能な場合に限り、そのグループからメッセージを取得します。

MQC.MQGMO_ALL_SEGMENTS_AVAILABLE*

グループ内のすべてのセグメントが使用可能な場合に限り、論理メッセージのセグメントを取得します。

MQC.MQGMO_BROWSE_FIRST

キューの先頭からブラウズします。

MQC.MQGMO_BROWSE_MSG_UNDER_CURSOR*

ブラウズ・カーソルの下のメッセージをブラウズします。

MQC.MQGMO_BROWSE_NEXT

キューの現在位置からブラウズします。

MQC.MQGMO_COMPLETE_MSG*

完全な論理メッセージのみを取り出します。

MQC.MQGMO_CONVERT

アプリケーション・データを、変換して `MQMessage` の `CharacterSet` および `Encoding` 属性に準拠させてからメッセージ・バッファにコピーすることを要求します。データがメッセージ・バ

ッファーから取り出されるときにデータ変換も適用されるため、アプリケーションはこのオプションを設定しません。

このオプションを使用すると、1 バイト文字セットから 2 バイト文字セットへの変換時に問題が発生する可能性があります。それで代替方法として、メッセージの送信後に、`readString`、`readLine`、および `writeString` メソッドを使用して変換を実行してください。

MQC.MQGM0_FAIL_IF_QUIESCING

キュー・マネージャーが静止中の場合は失敗します。

MQC.MQGM0_LOCK*

ブラウズされたメッセージをロックします。

MQC.MQGM0_LOGICAL_ORDER*

グループ内のメッセージと論理メッセージ内のセグメントを論理順序で戻します。

MQGM0_LOGICAL_ORDER オプションを再接続可能なクライアントで使用する場合、MQRC_RECONNECT_INCOMPATIBLE 理由コードがアプリケーションに返されます。

MQC.MQGM0_MARK_SKIP_BACKOUT*

キューのメッセージを復元することなく、作業単位をバックアウトできます。

MQC.MQGM0_MSG_UNDER_CURSOR

ブラウズ・カーソルが置かれているメッセージを読み取ります。

MQC.MQGM0_NONE

他のオプションは指定されません。すべてのオプションはそれぞれのデフォルト値を使用します。

MQC.MQGM0_NO_PROPERTIES

メッセージ記述子(または拡張)に含まれるプロパティを除き、メッセージのプロパティは取得されません。

MQC.MQGM0_NO_SYNCPOINT

同期点制御を使用しないでメッセージを読み取ります。

MQC.MQGM0_NO_WAIT

適切なメッセージがない場合はすぐに戻ります。

MQC.MQGM0_PROPERTIES_AS_Q_DEF

MQQueue の PropertyControl 属性によって定義されているメッセージ・プロパティを取得します。メッセージ記述子または拡張内のメッセージ・プロパティへのアクセスは、PropertyControl 属性による影響は受けません。

MQC.MQGM0_PROPERTIES_COMPATIBILITY

MQRFH2 ヘッダ中の接頭部が mcd、jms、usr、または mqext のメッセージ・プロパティを取得します。その他のメッセージのプロパティは廃棄されます(ただしメッセージ記述子または拡張に含まれるプロパティを除きます)。

MQC.MQGM0_PROPERTIES_FORCE_MQRFH2

MQRFH2 ヘッダー内のメッセージ・プロパティを取得します(ただしメッセージ記述子または拡張に含まれるプロパティを除きます)。MQC.MQGM0_PROPERTIES_FORCE_MQRFH2 は、プロパティを取得しようとしているものの、メッセージ・ハンドルを使用するように変更できないアプリケーションで使

MQC.MQGM0_PROPERTIES_IN_HANDLE

MsgHandle を使用してメッセージ・プロパティを取得します。

MQC.MQGM0_SYNCPOINT

同期点制御を使用してメッセージを読み取ります。メッセージには、他のアプリケーションでは使用できないものとしてマークが付けられますが、作業単位がコミットされたときにのみ、キューから削除されます。作業単位がバックアウトされると、メッセージは再び使用可能になります。

MQC.MQGM0_SYNCPOINT_IF_PERSISTENT*

メッセージが持続する場合、同期点制御を使用してメッセージを読み取ります。

MQC.MQGM0_UNLOCK*

既にロックされたメッセージをアンロックします。

MQC.MQGMO_WAIT

メッセージの到着を待ちます。

public string ResolvedQueueName {get;}

キュー・マネージャーは、ResolvedQueueName をメッセージが取り出されたキューのローカル名に設定します。ResolvedQueueName は、別名キューまたはモデル・キューがオープンされていた場合のキューのオープンに使用された名前とは異なります。

public char Segmentation {get;}*

Segmentation は、取り出されたメッセージに対するセグメンテーションを許可できるかどうかを示します。指定可能な値は以下のとおりです。

MQC.MQSEG_INHIBITED

セグメンテーションを許可しません。

MQC.MQSEG_ALLOWED

セグメンテーションを許可します

public byte SegmentStatus {get;}*

SegmentStatus は、取り出されたメッセージが論理メッセージの 1 つのセグメントであるかどうかを示す出力フィールドです。メッセージがセグメントである場合は、フラグによって、そのセグメントが最後のセグメントかどうかを示します。指定可能な値は以下のとおりです。

MQC.MQSS_LAST_SEGMENT

メッセージは論理メッセージの最後のまたは唯一のセグメントである。

MQC.MQSS_NOT_A_SEGMENT

メッセージは 1 つのセグメントではない。

MQC.MQSS_SEGMENT

メッセージは 1 つのセグメントであるが、論理メッセージの最後のセグメントではない。

public int WaitInterval {get; set;}

WaitInterval は、MQQueue.get 呼び出しが適切なメッセージの着信を待機する、最大時間 (ミリ秒単位) です。WaitInterval は、MQC.MQGMO_WAIT とともに使用します。MQC.MQWI_UNLIMITED の値を設定すると、メッセージを時間制限なしに待機することになります。

コンストラクター

public MQGetMessageOptions()

新規 MQGetMessageOptions オブジェクトについて、Options を MQC.MQGMO_NO_WAIT に、WaitInterval をゼロに、ResolvedQueueName をブランクにそれぞれ設定して構成します。

MQManagedObject.NET クラス

MQManagedObject を使用して、MQDestination、MQProcess、MQQueueManager、および MQSubscription の属性を照会、設定します。MQManagedObject は、これらのクラスのスーパークラスです。

クラス

```

System.Object
├── IBM.WMQ.MQBase
│   └── IBM.WMQ.MQBaseObject
│       └── IBM.WMQ.MQManagedObject
│           ├── IBM.WMQ.MQProcess
│           ├── IBM.WMQ.MQQueueManager
│           ├── IBM.WMQ.MQSubscription
│           └── IBM.WMQ.MQDestination

```

```
public class IBM.WMQ.MQManagedObject extends IBM.WMQ.MQBaseObject;
```

- [1738 ページの『プロパティ』](#)
- [1739 ページの『方法』](#)
- [1740 ページの『コンストラクター』](#)

プロパティ

プロパティの取得時にスローされた `MQException` をテストします。

```
public string AlternateUserId {get; set;}
```

リソースのオープン時に設定された代替ユーザー ID (もしあれば)。AlternateUserId.set は、オープンしているオブジェクトに対して出される場合は無視されます。AlternateUserId は、サブスクリプションには無効です。

```
public int CloseOptions {get; set;}
```

この属性は、リソースのクローズ方法を制御するために設定します。デフォルト値は `MQC.MQCO_NONE` です。MQC.MQCO_NONE は、永続動的キュー、一時動的キュー、サブスクリプション、およびそれらを作成したオブジェクトによってアクセスされているトピック以外のすべてのリソースに対して許可される唯一の値です。

キューおよびトピックの場合、以下の追加値を指定できます。

MQC.MQCO_DELETE

メッセージがない場合は、キューを削除します。

MQC.MQCO_DELETE_PURGE

キューを削除し、そこに入っているすべてのメッセージを除去します。

MQC.MQCO_QUIESCE

キューをクローズし、残っているメッセージがあれば警告を受け取ることを要求します (最終クローズの前にメッセージを取り出すことが可能になります)。

サブスクリプションの場合、以下の追加値を指定できます。

MQC.MQCO_KEEP_SUB

サブスクリプションは削除されません。このオプションは、元のサブスクリプションが永続的である場合のみに有効です。MQC.MQCO_KEEP_SUB は、永続的トピックのデフォルト値です。

MQC.MQCO_REMOVE_SUB

サブスクリプションは削除されます。MQC.MQCO_REMOVE_SUB は、非永続的な非管理トピックのデフォルト値です。

MQC.MQCO_PURGE_SUB

サブスクリプションは削除されます。MQC.MQCO_PURGE_SUB は、非永続的な管理トピックのデフォルト値です。

```
public MQQueueManager ConnectionReference {get;}
```

このリソースが属しているキュー・マネージャー。

public string MQDescription {get;}

キュー・マネージャーによって保持されるリソースの説明。サブスクリプションおよびトピックの場合、MQDescription は空ストリングを返します。

public boolean IsOpen {get;}

リソースが現在オープンしているかどうかを示します。

public string Name {get;}

リソースの名前。この名前は、アクセス方式で指定されたものか、キュー・マネージャーによって動的キューに割り振られたもののいずれかです。

public int OpenOptions {get; set;}

IBM MQ オブジェクトがオープンされるときに、OpenOptions が設定されます。OpenOptions.set メソッドは無視されるので、エラーは起こりません。サブスクリプションには OpenOptions がありません。

方法

public virtual void Close();

MQException をスローします。

オブジェクトをクローズします。Close を呼び出した後、このリソースに対してさらに操作を行うことは許可されません。Close メソッドの動作を変更するには、closeOptions 属性を設定します。

public string GetAttributeString(int selector, int length);

MQException をスローします。

属性ストリングを取得します。

選択子

照会する属性を示す整数。

length

必要なストリングの長さを示す整数。

public void Inquire(int[] selectors, int[] intAttrs, byte[] charAttrs);

MQException をスローします。

キュー、プロセス、またはキュー・マネージャーの属性が入っている整数の配列と一連の文字ストリングを返します。照会する属性は、セクター配列で指定します。

注: より一般的な属性の多くは、MQManagedObject、MQQueue、および MQQueueManager で定義された Get メソッドを使用して照会できます。

selectors

照会する値を持つ属性を指定する整数の配列。

intAttrs

整数属性値が戻される配列。整数属性値は、セクター配列の整数属性セクターと同じ順序で戻されます。

charAttrs

文字属性が連結されて戻されるバッファー。文字属性は、セクター配列の文字属性セクターと同じ順序で戻されます。各属性ストリングの長さは、属性ごとに固定されています。

public void Set(int[] selectors, int[] intAttrs, byte[] charAttrs);

MQException をスローします。

セクターのベクトルで定義されている属性を設定します。設定する属性は、セクター配列で指定します。

selectors

設定する値を持つ属性を指定する整数の配列。

intAttrs

設定する整数属性値の配列。これらの値は、セクター配列の整数属性セクターと同じ順序にする必要があります。

charAttrs

設定する文字属性が連結されるバッファー。これらの値は、セクター配列の文字属性セクターと同じ順序にする必要があります。各文字属性の長さは固定されています。

public void SetAttributeString(int selector, string value, int length);

MQException をスローします。

属性ストリングを設定します。

選択子

設定する属性を示す整数。

値

属性値として設定するストリング。

length

必要なストリングの長さを示す整数。

コンストラクター

protected MQManagedObject()

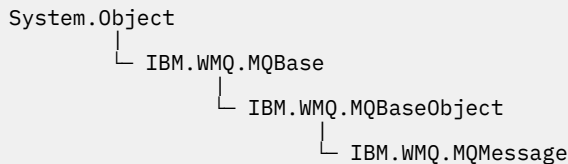
コンストラクター・メソッド。このオブジェクトは、単独ではインスタンス化できない抽象基本クラスです。

MQMessage.NET クラス

MQMessage を使用して、IBM MQ メッセージのメッセージ記述子およびデータにアクセスします。

MQMessage は、IBM MQ メッセージをカプセル化します。

クラス



```
public class IBM.WMQ.MQMessage extends IBM.WMQ.MQBaseObject;
```

MQMessage オブジェクトを作成してから Read メソッドおよび Write メソッドを使用して、ご使用のアプリケーション内でメッセージと他のオブジェクト間のデータを転送します。MQDestination クラス、MQQueue クラス、および MQTopic クラスの Put メソッドおよび Get メソッドを使用して、MQMessage オブジェクトを送受信します。

MQMessage のプロパティを使用して、メッセージ記述子のプロパティを取得および設定します。 SetProperty メソッドおよび GetProperty メソッドを使用して、拡張メッセージのプロパティを設定および取得します。

- [1740 ページの『プロパティ』](#)
- [1746 ページの『Read および Write のメッセージ・メソッド』](#)
- [1749 ページの『バッファー・メソッド』](#)
- [1750 ページの『プロパティ・メソッド』](#)
- [1752 ページの『コンストラクター』](#)

プロパティ

プロパティの取得時にスローされた MQException をテストします。

public string AccountingToken {get; set;}

メッセージの識別コンテキストの一部。これにより、メッセージの結果として実行される作業に対してアプリケーションから課金できるようになります。デフォルト値は MQC.MQACT_NONE です。

public string ApplicationIdData {get; set;}

メッセージの識別コンテキストの一部。ApplicationIdData は、アプリケーション・スイートによって定義される情報であり、メッセージやその発信元に関する追加情報を提供するために使用できます。デフォルト値は "" です。

public string ApplicationOriginData {get; set;}

この情報はアプリケーションによって定義され、メッセージ発信元に関する追加情報を提供するために使用できます。デフォルト値は "" です。

public int BackoutCount {get;}

メッセージが以前、作業単位の一貫として MQQueue.Get 呼び出しによって返され、その後バックアウトされた回数です。デフォルト値はゼロです。

public int CharacterSet {get; set;}

メッセージ内の文字データのコード化文字セット ID。

メッセージ内の文字データの文字セットを特定するには、CharacterSet を設定します。メッセージ内の文字データにどのような文字セットが使用されてエンコードされているかを調べるには、CharacterSet を取得します。

他の環境では、アプリケーションがキュー・マネージャーと同じ文字セットで実行されるのに対して、.NET アプリケーションは常に Unicode で実行されます。

ReadString メソッドおよび ReadLine メソッドは、ユーザー向けに、メッセージ内の文字データを Unicode に変換します。

WriteString メソッドは、Unicode から、CharacterSet でエンコードした文字セットに変換します。CharacterSet をデフォルト値 MQC.MQCCSI_Q_MGR (値は 0) に設定した場合、変換は行われず、CharacterSet が 1200 に設定されます。CharacterSet をその他の値に設定した場合、WriteString は Unicode からその代替値に変換されます。

注: その他の読み取りメソッドおよび書き込みメソッドでは、CharacterSet は使用されません。

- ReadChar および WriteChar は、変換せずに、Unicode 文字をメッセージ・バッファーに対して読み取りおよび書き込みします。
- ReadUTF および WriteUTF は、アプリケーションの Unicode スtring と、メッセージ・バッファー内の 2 バイト長の接頭部フィールド付き UTF-8 スtring 間の変換を行います。
- Byte メソッドは、アプリケーションとメッセージ・バッファー間のバイトの転送を変更せずにを行います。

public byte[] CorrelationId {get; set;}

• MQQueue.Get 呼び出しの場合は、取り出されるメッセージの相関 ID。キュー・マネージャーは、メッセージ記述子フィールドと一致するメッセージ ID および相関 ID を持つ最初のメッセージを返します。デフォルト値 MQC.MQCI_NONE は、相関 ID のマッチングに役立ちます。

• MQQueue.Put 呼び出しの場合は、設定する相関 ID。

public int DataLength {get;}

まだ読み取られていないメッセージ・データのバイト数。

public int DataOffset {get; set;}

メッセージ・データ内の現行カーソル位置。読み取りと書き込みは、現在位置で有効になります。

public int Encoding {get; set;}

アプリケーション・メッセージ・データで使用する数値表記。Encoding は、2 進データ、パック 10 進データ、および浮動小数点データに適用されます。これらの数値形式に従って、読み取りメソッドおよび書き込みメソッドの動作は異なります。これら 3 つの各セクションの値を合計して、encoding フィールドの値を構成します。または、ビット単位の OR 演算子を使用し、これら 3 つのセクションそれぞれの値を結合して値を構成します。

1. 2 進整数

MQC.MQENC_INTEGER_NORMAL

ビッグ・エンディアン形式の整数。

MQC.MQENC_INTEGER_REVERSED

Intel アーキテクチャーで使用されるリトル・エンディアン形式の整数。

2. パック 10 進数

MQC.MQENC_DECIMAL_NORMAL

z/OS で使用されるビッグ・エンディアン形式のパック 10 進数。

MQC.MQENC_DECIMAL_REVERSED

リトル・エンディアン形式のパック 10 進数。

3. 浮動小数点数

MQC.MQENC_FLOAT_IEEE_NORMAL

ビッグ・エンディアン形式の IEEE float 型。

MQC.MQENC_FLOAT_IEEE_REVERSED

Intel アーキテクチャーで使用されるリトル・エンディアン形式の IEEE float 型。

MQC.MQENC_FLOAT_S390

z/OS 形式の浮動小数点。

デフォルト値は次のとおりです。

```
MQC.MQENC_INTEGER_REVERSED |  
MQC.MQENC_DECIMAL_REVERSED |  
MQC.MQENC_FLOAT_IEEE_REVERSED
```

デフォルト設定により、WriteInt はリトル・エンディアン形式の整数を書き込み、ReadInt はリトル・エンディアン形式の整数を読み取るようになります。代わりに MQC.MQENC_INTEGER_NORMAL フラグを設定すると、WriteInt はビッグ・エンディアン形式の整数を書き込み、ReadInt はビッグ・エンディアン形式の整数を読み取ります。

注: IEEE 形式の浮動小数点を zSeries 形式の浮動小数点に変換すると、精度が低下する可能性があります。

public int Expiry {get; set;}

メッセージを書き込むアプリケーションによって設定される有効期限時刻。これは 0.1 秒単位で表されます。メッセージの有効期限時間が経過すると、そのメッセージはキュー・マネージャーによる廃棄の対象になります。メッセージにいずれかの MQC.MQRO_EXPIRATION フラグが指定されている場合は、メッセージの廃棄時にレポートが生成されます。デフォルト値は MQC.MQEI_UNLIMITED で、メッセージの有効期限が切れないことを意味します。

public int Feedback {get; set;}

メッセージ・タイプ MQC.MQMT_REPORT の Feedback を使用して、レポートの性質を示します。以下のフィードバック・コードがシステムで定義されています。

- MQC.MQFB_EXPIRATION
- MQC.MQFB_COA
- MQC.MQFB_COD
- MQC.MQFB_QUIT
- MQC.MQFB_PAN
- MQC.MQFB_NAN
- MQC.MQFB_DATA_LENGTH_ZERO
- MQC.MQFB_DATA_LENGTH_NEGATIVE
- MQC.MQFB_DATA_LENGTH_TOO_BIG
- MQC.MQFB_BUFFER_OVERFLOW

- MQC.MQFB_LENGTH_OFF_BY_ONE
- MQC.MQFB_IIH_ERROR

MQC.MQFB_APPL_FIRST から MQC.MQFB_APPL_LAST の範囲内のアプリケーションで定義済みフィードバック値を使用することもできます。このフィールドのデフォルト値は MQC.MQFB_NONE です。これはフィードバックを提供しないことを示します。

public string Format {get; set;}

メッセージの送信側が使用する形式名。メッセージのデータの性質を受信側に示します。独自の形式名を使用できますが、文字 MQ で始まる名前には、キュー・マネージャーによって定義された意味があります。キュー・マネージャーの組み込み形式は次のとおりです。

MQC.MQFMT_ADMIN

コマンド・サーバー要求/応答メッセージ。

MQC.MQFMT_COMMAND_1

タイプ 1 のコマンド応答メッセージ。

MQC.MQFMT_COMMAND_2

タイプ 2 のコマンド応答メッセージ。

MQC.MQFMT_DEAD_LETTER_HEADER

送達不能ヘッダー。

MQC.MQFMT_EVENT

イベント・メッセージ。

MQC.MQFMT_NONE

形式名がありません。

MQC.MQFMT_PCF

プログラマブル・コマンド・フォーマットのユーザー定義メッセージ。

MQC.MQFMT_STRING

全体が文字で構成されているメッセージ。

MQC.MQFMT_TRIGGER

トリガー・メッセージ

MQC.MQFMT_XMIT_Q_HEADER

伝送キュー・ヘッダー。

デフォルト値は MQC.MQFMT_NONE です。

public byte[] GroupId {get; set;}

物理メッセージが属するメッセージ・グループを識別するバイト・ストリング。デフォルト値は MQC.MQGI_NONE です。

public int MessageFlags {get; set;}

メッセージの分割と状況を制御するフラグ。

public byte[] MessageId {get; set;}

MQQueue.Get 呼び出しの場合は、取り出されるメッセージのメッセージ ID をこのフィールドに指定します。通常、キュー・マネージャーは、メッセージ記述子フィールドと一致するメッセージ ID および相関 ID を持つ最初のメッセージを返します。特殊値 MQC.MQMI_NONE を使用して、任意のメッセージ ID をマッチングできます。

MQQueue.Put 呼び出しの場合は、このフィールドは使用するメッセージ ID を指定します。

MQC.MQMI_NONE を指定すると、キュー・マネージャーはメッセージの書き込み時に一意のメッセージ ID を生成します。このメンバー変数の値は書き込みの後で更新され、使用されたメッセージ ID が示されます。デフォルト値は MQC.MQMI_NONE です。

public int MessageLength {get;}

MQMessage オブジェクトのメッセージ・データのバイト数。

public int MessageSequenceNumber {get; set;}

グループ内の論理メッセージの順序番号。

public int MessageType {get; set;}

メッセージのタイプを示します。以下の値がシステムで現在定義されています。

- MQC.MQMT_DATAGRAM
- MQC.MQMT_REPLY
- MQC.MQMT_REPORT
- MQC.MQMT_REQUEST

MQC.MQMT_APPL_FIRST から MQC.MQMT_APPL_LAST の範囲で、アプリケーション定義の値も使用できます。このフィールドのデフォルト値は MQC.MQMT_DATAGRAM です。

public int Offset {get; set;}

セグメント化されたメッセージにおける、論理メッセージの先頭からの物理メッセージのデータのオフセット。

public int OriginalLength {get; set;}

分割されたメッセージの元の長さ。

public int Persistence {get; set;}

メッセージの持続性。以下の値が定義されます。

- MQC.MQPER_NOT_PERSISTENT

再接続可能なクライアントでこのオプションを設定した場合は、接続に成功すると、アプリケーションに対して MQRC_NONE 理由コードが返されます。

- MQC.MQPER_PERSISTENT

再接続可能なクライアントでこのオプションを設定した場合は、接続に成功すると、アプリケーションに対して MQRC_CALL_INTERRUPTED 理由コードが返されます。

- MQC.MQPER_PERSISTENCE_AS_Q_DEF

デフォルト値は MQC.MQPER_PERSISTENCE_AS_Q_DEF です。これは、宛先キューのデフォルトの永続性属性からメッセージの永続性を取得します。

public int Priority {get; set;}

メッセージ優先順位。アウトバウンド・メッセージでは、特殊値 MQC.MQPRI_PRIORITY_AS_Q_DEF も設定できます。この場合、メッセージの優先順位は、宛先キューのデフォルトの優先順位属性から取得されます。デフォルト値は MQC.MQPRI_PRIORITY_AS_Q_DEF です。

public int PropertyValidation {get; set;}

メッセージのプロパティの設定時に、プロパティの妥当性検査を行うかどうかを指定します。指定可能な値は以下のとおりです。

- MQCMHO_DEFAULT_VALIDATION
- MQCMHO_VALIDATE
- MQCMHO_NO_VALIDATION

デフォルト値は MQCMHO_DEFAULT_VALIDATION です。

public string PutApplicationName {get; set;}

メッセージを書き込むアプリケーションの名前。デフォルト値は "" です。

public int PutApplicationType {get; set;}

メッセージを書き込むアプリケーションのタイプ。PutApplicationType はシステム定義またはユーザー定義の値です。以下の値がシステムで定義されています。

- MQC.MQAT_AIX
- MQC.MQAT_CICS
- MQC.MQAT_DOS
- MQC.MQAT_IMS

- MQC.MQAT_MVS
- MQC.MQAT_OS2
- MQC.MQAT_OS400
- MQC.MQAT_QMGR
- MQC.MQAT_UNIX
- MQC.MQAT_WINDOWS
- MQC.MQAT_JAVA

デフォルト値は MQC.MQAT_NO_CONTEXT で、これはコンテキスト情報がメッセージ中に存在しないことを示します。

public DateTime PutDateTime {get; set;}

メッセージが書き込まれた日付と時刻。

public string ReplyToQueueManagerName {get; set;}

応答メッセージまたはレポート・メッセージを送信するキュー・マネージャーの名前。デフォルト値は "" です。キュー・マネージャーでは ReplyToQueueManagerName が指定されます。

public string ReplyToQueueName {get; set;}

メッセージの取得要求を発行したアプリケーションが MQC.MQMT_REPLY または MQC.MQMT_REPORT メッセージを送信するメッセージ・キューの名前。デフォルトの ReplyToQueueName は "" です。

public int Report {get; set;}

Report を使用して、レポート・メッセージおよび応答メッセージに関するオプションを指定します。

- レポートを必須にするかどうか。
- アプリケーション・メッセージ・データをレポートに含めるかどうか。
- レポートまたは応答でのメッセージ ID および相関 ID の設定方法。

以下の 4 種類のレポートを組み合わせることで要求できます。

- 4 種類のレポートを任意に組み合わせることで指定します。アプリケーション・メッセージ・データをレポート・メッセージに含めるかどうかに応じて、3 つのオプションから任意のオプションを、レポート・タイプごとに選択します。

1. 到着時の確認

- MQC.MQRO_COA
- MQC.MQRO_COA_WITH_DATA
- MQC.MQRO_COA_WITH_FULL_DATA **

2. 送達時の確認

- MQC.MQRO_COD
- MQC.MQRO_COD_WITH_DATA
- MQC.MQRO_COD_WITH_FULL_DATA **

3. 例外

- MQC.MQRO_EXCEPTION
- MQC.MQRO_EXCEPTION_WITH_DATA
- MQC.MQRO_EXCEPTION_WITH_FULL_DATA **

4. 有効期限

- MQC.MQRO_EXPIRATION
- MQC.MQRO_EXPIRATION_WITH_DATA
- MQC.MQRO_EXPIRATION_WITH_FULL_DATA **

注: リスト内で ** のマークが付いた値は、z/OS キュー・マネージャーではサポートされません。アプリケーションが実行されているプラットフォームにかかわらず、アプリケーションが z/OS キュー・マネージャーにアクセスする可能性がある場合は使用しないでください。

- レポート・メッセージまたは応答メッセージのメッセージ ID の生成方法を制御するには、以下のいずれかを指定します。
 - MQC.MQRO_NEW_MSG_ID
 - MQC.MQRO_PASS_MSG_ID
- レポート・メッセージまたは応答メッセージの相関 ID を設定する方法を制御するには、以下のいずれかを指定します。
 - MQC.MQRO_COPY_MSG_ID_TO_CORREL_ID
 - MQC.MQRO_PASS_CORREL_ID
- 元のメッセージを宛先キューに送達できない場合に、元のメッセージの後処理を制御するには、以下のいずれかを指定します。
 - MQC.MQRO_DEAD_LETTER_Q
 - MQC.MQRO_DISCARD_MSG **
- レポート・オプションが指定されない場合、デフォルトは次のとおりです。

```
MQC.MQRO_NEW_MSG_ID |  
MQC.MQRO_COPY_MSG_ID_TO_CORREL_ID |  
MQC.MQRO_DEAD_LETTER_Q
```

- 受信側アプリケーションが肯定アクションまたは否定アクションのレポート・メッセージを送信するよう要求するため、次のいずれかまたは両方を指定できます。
 - MQC.MQRO_PAN
 - MQC.MQRO_NAN

public int TotalMessageLength {get;}

このメッセージの受信元のメッセージ・キューに格納されるメッセージのバイト総数。

public string UserId {get; set;}

UserId は、メッセージの識別コンテキストの一部です。通常はキュー・マネージャーで値が指定されます。識別コンテキストを設定する権限があれば、値をオーバーライドできます。

public int Version {get; set;}

使用中の MQMD 構造体のバージョン。

Read および Write のメッセージ・メソッド

Read メソッドと Write メソッドは、.NET System.IO 名前空間内の BinaryReader クラスと BinaryWriter クラスのメンバーと同じ機能を実行します。この言語のすべての構文と使用例については、MSDN を参照してください。これらのメソッドは、メッセージ・バッファー内の現在位置から読み取りおよび書き込みを行います。これらのメソッドは、読み取りまたは書き込みのバイト数分だけ現在位置を進めます。

注: メッセージ・データに MQRFH または MQRFH2 ヘッダーが含まれる場合は、ReadBytes メソッドを使用してデータを読み取る必要があります。

- メソッドはすべて、IOException をスローします。
- ReadFully メソッドは、メッセージにちょうど合うように、宛先の byte 配列または sbyte 配列を自動的にサイズ変更します。ヌルの配列もサイズ変更されます。
- Read メソッドは、EndOfStreamException をスローします。
- WriteDecimal メソッドは、MQException をスローします。
- ReadString メソッド、ReadLine メソッド、および WriteString メソッドは、Unicode とメッセージの文字セット間の変換を行います。[CharacterSet](#) を参照してください。

- Decimal メソッドは、Encoding の値に従って、ビッグ・エンディアンの MQC.MQENC_DECIMAL_NORMAL、またはリトル・エンディアンの MQC.MQENC_DECIMAL_REVERSE のいずれかの形式にエンコードされたパック 10 進数を読み取りおよび書き込みします。10 進数の範囲および対応する .NET タイプは、以下のとおりです。

Decimal2/short

-999 から 999

Decimal4/int

-9999999 から 9999999

Decimal8/long

-9999999999999999 から 9999999999999999

- Double メソッドおよび Float メソッドは、Encoding の値に従って、IEE ビッグ・エンディアン形式とリトル・エンディアン形式の MQC.MQENC_FLOAT_IEEE_NORMAL と MQC.MQENC_FLOAT_IEEE_REVERSED でエンコードされているか、または S/390 形式の MQC.MQENC_FLOAT_S390 でエンコードされた浮動小数点値を読み取りおよび書き込みします。
- Int メソッドは、Encoding の値に従って、ビッグ・エンディアンの MQC.MQENC_INTEGER_NORMAL、またはリトル・エンディアンの MQC.MQENC_INTEGER_REVERSED のいずれかの形式でエンコードされた整数値を読み取りおよび書き込みします。符号なしの 2 バイト整数型を追加する場合を除き、整数にはすべて符号が付きます。整数のサイズ、.NET および IBM MQ タイプは、以下のとおりです。

2 バイト

short, Int2, ushort, UInt2

4 バイト

int, Int4

8 バイト

long, Int8

- WriteObject は、オブジェクトのクラス、その非一時フィールドと非静的フィールドの値、そのスーパータイプのフィールドをメッセージ・バッファに転送します。
- ReadObject は、オブジェクトのクラス、クラスのシグニチャー、その非一時フィールドと非静的フィールドの値、およびそのスーパータイプのフィールドからオブジェクトを作成します。

表 879. Read および Write のメッセージ・メソッド

ターゲット・タイプ	Method signatures
Boolean	<pre>public bool ReadBoolean(); public void WriteBoolean(bool value);</pre>
Byte	<pre>public byte ReadByte() public byte ReadUnsignedByte() public void Write(int value) public void WriteByte(int value) public void WriteByte(byte value) public void WriteByte(sbyte value)</pre>

表 879. *Read* および *Write* のメッセージ・メソッド (続き)

ターゲット・タイプ	Method signatures
Bytes	<pre> public byte[] ReadBytes(int count) public void ReadFully(ref byte[] value) public void ReadFully(ref sbyte[] value) public void ReadFully(ref byte[] value, int offset,int length) public void ReadFully(ref sbyte[] value, int offset,int length) public void Write(byte[] value) public void Write(sbyte[] value) public void Write(byte[] value, int offset,int length) public void Write(sbyte[] value, int offset,int length) public void WriteBytes(string value) </pre>
Decimal2	<pre> public void WriteDecimal2(short value) </pre>
Decimal4	<pre> public void WriteDecimal4(short value) </pre>
Decimal8	<pre> public void WriteDecimal8(short value) </pre>
Double	<pre> public double ReadDouble() public void WriteDouble(double value) </pre>
Float	<pre> public float ReadFloat() public void WriteFloat(float value) </pre>
Int2	<pre> public void WriteInt2(int value) </pre>
Int4	<pre> public int readDecimal4() public int ReadInt() public int ReadInt4() public void WriteInt(int value) public void WriteInt4(int value) </pre>
Int8	<pre> public void WriteInt8(long value) </pre>
Long	<pre> public long ReadDecimal8() public long ReadLong() public long ReadInt8() public void WriteLong(long value) </pre>
Object	<pre> public Object ReadObject() public void WriteObject(Object object) </pre>

表 879. *Read* および *Write* のメッセージ・メソッド (続き)

ターゲット・タイプ	Method signatures
Short	<pre>public short ReadShort() public short ReadDecimal2() public short ReadInt2() public void WriteShort(int value)</pre>
string	<pre>public string ReadString(int length) public void WriteString(string string)</pre>
Unsigned Short	<pre>public ushort ReadUnsignedShort() public ushort ReadUInt2()</pre>
Unicode	<pre>public string ReadLine() public char ReadChar() public void WriteChar(int value) public void WriteChars(string string)</pre>
UTF	<pre>public string ReadUTF() public void WriteUTF(string string)</pre>

バッファ・メソッド

public void ClearMessage();

IOException をスローします。

メッセージ・バッファのデータを廃棄し、データ・オフセットをゼロに戻します。

public void ResizeBuffer(int size)

IOException をスローします。

MQMessage オブジェクトに対する、バッファのサイズに関するヒント。後続の取得操作で必要な場合があります。現在、メッセージにデータが含まれていて、新しいサイズが現在のサイズより小さい場合、メッセージ・データは切り捨てられます。

public void Seek(int pos)

IOException、*ArgumentOutOfRangeException*、*ArgumentException* をスローします。

pos で指定されたメッセージ・バッファの絶対位置にカーソルを移動します。後続の読み取りと書き込みは、バッファのこの位置から行われます。

public int SkipBytes(int i)

IOException、*EndOfStreamException* をスローします。

メッセージ・バッファ内を *n* バイト分進め、スキップしたバイト数である *n* を返します。

SkipBytes メソッドは、以下のいずれかのイベントが発生するまでブロックします。

- すべてのバイトがスキップされた。
- メッセージ・バッファの最後が検出された。

- 例外がスローされた。

プロパティー・メソッド

public void DeleteProperty(string name);

MQException をスローします。

指定した名前のプロパティーをメッセージから削除します。

名前

削除するプロパティーの名前。

public System.Collections.IEnumerator GetPropertyNames(string name)

MQException をスローします。

指定した名前と一致するすべてのプロパティー名の IEnumerator を返します。 名前の末尾にパーセント記号 '%' をワイルドカード文字として使用し、ゼロ文字以上 (ピリオド (.) を含む) が一致するメッセージのプロパティーをフィルターに掛けることができます。

名前

突き合わせるプロパティーの名前。

SetProperty メソッドと GetProperty メソッド

SetProperty メソッドおよび GetProperty メソッドはすべて、MQException をスローします。

MQMessage.NET クラスの SetProperty メソッドは、プロパティーがまだ存在しない場合に新規プロパティーを追加します。 ただし、プロパティーが既に存在する場合は、指定されたプロパティー値がリストの末尾に追加されます。 SetProperty を使用してプロパティー名に複数の値が設定されている場合、その名前に対して GetProperty を呼び出すと、値が設定順に返されます。

この動作は、GetLongProperty、SetLongProperty、GetBooleanProperty、SetBooleanProperty、GetStringProperty、および SetStringProperty など、すべての Set*Property 型付きメソッドおよび Get*Property 型付きメソッドで同じです。

表 880. SetProperty メソッドと GetProperty メソッド	
タイプ	Method signatures
Boolean	<pre>public boolean GetBooleanProperty(string name); public boolean GetBooleanProperty(string name, MQPropertyDescriptor pd); public void SetBooleanProperty(string name, boolean value); public void SetBooleanProperty(string name, MQPropertyDescriptor pd, boolean value);</pre>
Byte	<pre>public sbyte GetByteProperty(string name); public sbyte GetByteProperty(string name, MQPropertyDescriptor pd); public void SetByteProperty(string name, sbyte value); public void SetByteProperty(string name, MQPropertyDescriptor pd, sbyte value);</pre>
Bytes	<pre>public sbyte[] GetBytesProperty(string name); public sbyte[] GetBytesProperty(string name, MQPropertyDescriptor pd); public void SetBytesProperty(string name, sbyte[] value); public void SetBytesProperty(string name, MQPropertyDescriptor pd, sbyte[] value);</pre>

表 880. SetProperty メソッドと GetProperty メソッド (続き)

タイプ	Method signatures
Double	<pre>public double GetDoubleProperty(string name); public double GetDoubleProperty(string name, MQPropertyDescriptor pd); public void SetDoubleProperty(string name, double value); public void SetDoubleProperty(string name, MQPropertyDescriptor pd, double value);</pre>
Float	<pre>public float GetFloatProperty(string name); public float GetFloatProperty(string name, MQPropertyDescriptor pd); public void SetFloatProperty(string name, float value); public void SetFloatProperty(string name, MQPropertyDescriptor pd, float value);</pre>
Int2	<pre>public short GetInt2Property(string name); public short GetInt2Property(string name, MQPropertyDescriptor pd); public void SetInt2Property(string name, short value); public void SetInt2Property(string name, MQPropertyDescriptor pd, short value);</pre>
Int4	<pre>public int GetInt4Property(string name); public int GetInt4Property(string name, MQPropertyDescriptor pd); public void SetInt4Property(string name, int value); public void SetInt4Property(string name, MQPropertyDescriptor pd, int value);</pre>
Int8	<pre>public long GetInt8Property(string name); public long GetInt8Property(string name, MQPropertyDescriptor pd); public void SetInt8Property(string name, long value); public void SetInt8Property(string name, MQPropertyDescriptor pd, long value);</pre>
Long	<pre>public long GetLongProperty(string name); public long GetLongProperty(string name, MQPropertyDescriptor pd); public void SetLongProperty(string name, long value); public void SetLongProperty(string name, MQPropertyDescriptor pd, long value);</pre>
Object	<pre>public Object GetObjectProperty(string name); public Object GetObjectProperty(string name, MQPropertyDescriptor pd); public void SetObjectProperty(string name, Object value); public void SetObjectProperty(string name, MQPropertyDescriptor pd, Object value);</pre>
Short	<pre>public short GetShortProperty(string name); public short GetShortProperty(string name, MQPropertyDescriptor pd); public void SetShortProperty(string name, short value); public void SetShortProperty(string name, MQPropertyDescriptor pd, short value);</pre>

表 880. SetProperty メソッドと GetProperty メソッド (続き)

タイプ	Method signatures
string	<pre>public string GetStringProperty(string name); public string GetStringProperty(string name, MQPropertyDescriptor pd);</pre>
	<pre>public void SetStringProperty(string name, string value); public void SetStringProperty(string name, MQPropertyDescriptor pd, string value);</pre>

コンストラクター

public MQMessage();

デフォルトのメッセージ記述子情報と空のメッセージ・バッファーを持つ MQMessage オブジェクトを作成します。

MQProcess.NET クラス

MQProcess を使用して IBM MQ プロセスの属性を照会します。コンストラクター、または MQQueueManager AccessProcess を使用して MQProcess オブジェクトを作成します。

クラス

```

System.Object
├── IBM.WMQ.MQBase
│   └── IBM.WMQ.MQBaseObject
│       └── IBM.WMQ.MQManagedObject
│           └── IBM.WMQ.MQProcess

```

```
public class IBM.WMQ.MQProcess extends IBM.WMQ.MQManagedObject;
```

- [1752 ページの『プロパティ』](#)
- [1753 ページの『コンストラクター』](#)

プロパティ

プロパティの取得時にスローされた MQException をテストします。

public string ApplicationId {get;}

開始するアプリケーションを識別する文字ストリングを取得します。ApplicationId は、トリガー・モニター・アプリケーションによって使用されます。ApplicationId は、トリガー・メッセージの一部として開始キューに送られます。

デフォルト値はヌルです。

public int ApplicationType {get;}

トリガー・モニター・アプリケーションによって開始されるプロセスのタイプを識別します。標準のタイプは定義されており、その他は次のタイプが使用できます。

- MQAT_AIX
- MQAT_CICS
- MQAT_IMS
- MQAT_MVS
- MQAT_NATIVE

- MQAT_OS400
- MQAT_UNIX
- MQAT_WINDOWS
- MQAT_JAVA
- MQAT_USER_FIRST
- MQAT_USER_LAST

デフォルト値は MQAT_NATIVE です。

public string EnvironmentData {get;}

開始されるアプリケーションの環境に関する情報を取得します。

デフォルト値はヌルです。

public string UserData {get;}

開始されるアプリケーションについてユーザーから提供された情報を取得します。

デフォルト値はヌルです。

コンストラクター

public MQProcess(MQQueueManager queueManager, string processName, int openOptions);

public MQProcess(MQQueueManager qMgr, string processName, int openOptions, string queueManagerName, string alternateUserId);

MQException をスローします。

キュー・マネージャー *qMgr* 上の IBM MQ プロセスにアクセスして、プロセス属性を照会します。

qMgr

アクセスするキュー・マネージャー。

processName

オープンするプロセスの名前。

openOptions

プロセスのオープンを制御するオプション。追加またはビット単位 OR を使用した結合が可能です。有効なオプションは次の通りです。

- MQC.MQ00_FAIL_IF QUIESCING
- MQC.MQ00_INQUIRE
- MQC.MQ00_SET
- MQC.MQ00_ALTERNATE_USER_AUTHORITY

queueManagerName

プロセスが定義されたキュー・マネージャーの名前。キュー・マネージャーが、プロセスがアクセスしているものと同じである場合、そのキュー・マネージャーの名前は空白またはヌルのままにすることができます。

alternateUserId

openOptions パラメーターで MQC.MQ00_ALTERNATE_USER_AUTHORITY が指定されている場合、*alternateUserId* は、アクションの許可を確認するために使用される代替ユーザー ID を指定します。MQ00_ALTERNATE_USER_AUTHORITY が指定されていない場合、*alternateUserId* は空白またはヌルのままにすることができます。

MQC.MQ00_ALTERNATE_USER_AUTHORITY が指定されていない場合、デフォルトのユーザー権限を使用してキュー・マネージャーに接続します。

```
public MQProcess MQQueueManager.AccessProcess(string processName, int
openOptions);
public MQProcess MQQueueManager.AccessProcess(string processName, int
openOptions, string queueManagerName, string alternateUserId);
```

MQException をスローします。

プロセス属性を照会するために、このキュー・マネージャーの IBM MQ プロセスにアクセスします。

processName

オープンするプロセスの名前。

openOptions

プロセスのオープンを制御するオプション。追加またはビット単位 OR を使用した結合が可能である、有効なオプションは次の通りです。

- MQC.MQOO_FAIL_IF_QUIESCING
- MQC.MQOO_INQUIRE
- MQC.MQOO_SET
- MQC.MQOO_ALTERNATE_USER_AUTHORITY

queueManagerName

プロセスが定義されたキュー・マネージャーの名前。キュー・マネージャーが、プロセスがアクセスしているものと同じである場合、そのキュー・マネージャーの名前はブランクまたはヌルのままにすることができます。

alternateUserId

openOptions パラメーターで MQC.MQOO_ALTERNATE_USER_AUTHORITY が指定されている場合、*alternateUserId* は、アクションの許可を確認するために使用される代替ユーザー ID を指定します。MQOO_ALTERNATE_USER_AUTHORITY が指定されていない場合、*alternateUserId* はブランクまたはヌルのままにすることができます。

MQC.MQOO_ALTERNATE_USER_AUTHORITY が指定されていない場合、デフォルトのユーザー権限を使用してキュー・マネージャーに接続します。

MQPropertyDescriptor.NET クラス

MQPropertyDescriptor を、MQMessage GetProperty メソッドおよび SetProperty メソッドのパラメーターとして使用します。MQPropertyDescriptor は、MQMessage プロパティの説明です。

クラス

```
System.Object
└─ IBM.WMQ.MQPropertyDescriptor
```

```
public class IBM.WMQ.MQPropertyDescriptor extends System.Object;
```

- [1754 ページの『プロパティ』](#)
- [1756 ページの『コンストラクター』](#)

プロパティ

プロパティの取得時にスローされた MQException をテストします。

```
public int Context {get; set;}
```

プロパティが属するメッセージ・コンテキストを指定します。指定可能な値は以下のとおりです。

MQC.MQPD_NO_CONTEXT

プロパティはメッセージ・コンテキストに関連付けられません。

MQC.MQPD_USER_CONTEXT

プロパティは user コンテキストに関連付けられます。

ユーザーが許可されている場合は、メッセージの取得時に、ユーザー・コンテキストに関連付けられたプロパティが保存されます。保存されたコンテキストを参照している後続の Put メソッドは、プロパティを新しいメッセージに渡すことができます。

public int CopyOptions {get; set;}

CopyOptions は、プロパティのコピー先とすることができるメッセージ・タイプについて説明します。

キュー・マネージャーが不正と認識した IBM MQ 定義のプロパティを含むメッセージを、キュー・マネージャーが受信した場合、キュー・マネージャーは、CopyOptions フィールドの値を訂正します。

次のオプションの組み合わせのいずれかを指定することができます。値を追加するか、ビット単位の OR を使用して、オプションを結合します。

MQC.MQCOPY_ALL

そのプロパティはすべてのタイプの後続メッセージにコピーされます。

MQC.MQCOPY_FORWARD

そのプロパティは、転送されるメッセージにコピーされます。

MQC.MQCOPY_PUBLISH

そのプロパティは、メッセージのパブリッシュ中にサブスクライバーが受信したメッセージにコピーされます。

MQC.MQCOPY_REPLY

そのプロパティは応答メッセージにコピーされます。

MQC.MQCOPY_REPORT

そのプロパティはレポート・メッセージにコピーされます。

MQC.MQCOPY_DEFAULT

その値は、他のコピー・オプションが指定されていないことを示していました。プロパティと後続のメッセージの間には、全く関連するものが存在していません。MQC.MQCOPY_DEFAULT は、常にメッセージ記述子プロパティに対して返されます。

MQC.MQCOPY_NONE

MQC.MQCOPY_DEFAULT と同じです。

public int Options { set; }

Options は、デフォルトの CMQC.MQPD_NONE になります。他の値は設定できません。

public int Support { get; set; }

Support を設定して、IBM MQ の定義メッセージ・プロパティに必要なサポートのレベルを指定します。他のすべてのプロパティに対するサポートはオプションです。以下のいずれかの値を指定できます。値を指定しなくてもかまいません。

MQC.MQPD_SUPPORT_OPTIONAL

プロパティはサポートされていなくても、キュー・マネージャーに受け入れられます。メッセージ・プロパティをサポートしていないキュー・マネージャーにメッセージをフローするために、このプロパティは破棄される場合があります。この値は、IBM MQ 定義ではないプロパティにも割り当てられます。

MQC.MQPD_SUPPORT_REQUIRED

プロパティに対するサポートは必須です。IBM MQ 定義のプロパティをサポートしていないキュー・マネージャーにメッセージを書き込んだ場合、メソッドは失敗します。完了コード MQC.MQCC_FAILED と理由コード MQC.MQRC_UNSUPPORTED_PROPERTY が返されます。

MQC.MQPD_SUPPORT_REQUIRED_IF_LOCAL

メッセージの宛先が、ローカル・キュー・マネージャーである場合、プロパティに対するサポートは必須です。IBM MQ 定義のプロパティをサポートしていないキュー・マネージャーのローカル・キューにメッセージを書き込んだ場合、メソッドは失敗します。完了コード MQC.MQCC_FAILED と理由コード MQC.MQRC_UNSUPPORTED_PROPERTY が返されます。

メッセージがリモート・キュー・マネージャーに書き込まれた場合、チェックはされません。

コンストラクター

PropertyDescriptor();

プロパティ記述子を作成します。

MQPutMessageOptions.NET クラス

MQPutMessageOptions を使用してメッセージの送信方法を指定します。それによって、MQDestination.Put の動作を変更します。

クラス

```
System.Object
├── IBM.WMQ.MQBase
│   └── IBM.WMQ.MQBaseObject
│       └── IBM.WMQ.MQPutMessageOptions
```

```
public class IBM.WMQ.MQPutMessageOptions extends IBM.WMQ.MQBaseObject;
```

• [1756 ページの『プロパティ』](#) [1758 ページの『コンストラクター』](#)

プロパティ

プロパティの取得時にスローされた MQException をテストします。

注: このクラスで使用可能な一部のオプションの動作は、それらのオプションが使用される環境によって異なります。これらのエレメントはアスタリスク (*) でマーク付けられます。

public MQQueue ContextReference {get; set;}

options フィールドに MQC.MQPMO_PASS_IDENTITY_CONTEXT または MQC.MQPMO_PASS_ALL_CONTEXT が含まれている場合、このフィールドを、コンテキスト情報を取得する場所として MQQueue を参照するように設定します。

このフィールドの初期値はヌルです。

public int InvalidDestCount {get;} *

通常は、配布リストとして使用され、InvalidDestCount は、配布リスト中のキューに送信できなかったメッセージの数を示しています。この数には、オープンに失敗したキューの数、およびオープンには成功したが PUT 操作には失敗したキューの数も含まれています。

.NET は配布リストをサポートしていませんが、単一のキューをオープンするときには InvalidDestCount が設定されています。

public int KnownDestCount {get;} *

通常は配布リストとして使用され、KnownDestCount は、現在の呼び出しがローカル・キューを分解するキューへの送信に成功したメッセージの数を示しています。

.NET は配布リストをサポートしていませんが、単一のキューをオープンするときには InvalidDestCount が設定されています。

public int Options {get; set;}

MQDestination.put および MQQueueManager.put のアクションを制御するオプション。以下のいずれかの値を指定できます。値を指定しなくてもかまいません。複数のオプションが必要な場合は、ビット単位の OR オペレーターを使用して、その値を追加または結合することができます。

MQC.MQPMO_ASYNC_RESPONSE

このオプションによって、MQDestination.put 呼び出しが、いくつかの応答データと同時に行われます。

MQC.MQPMO_DEFAULT_CONTEXT

デフォルトのコンテキストをメッセージに関連付けます。

MQC.MQPMO_FAIL_IF QUIESCING

キュー・マネージャーが静止中の場合は失敗します。

MQC.MQPMO_LOGICAL_ORDER *

論理メッセージとセグメントを論理順序でメッセージ・グループに書き込みます。
再接続可能なクライアントで MQPMO_LOGICAL_ORDER オプションを使用すると、MQRC_RECONNECT_INCOMPATIBLE 理由コードがアプリケーションに返されます。

MQC.MQPMO_NEW_CORREL_ID *

送信されるメッセージごとに新しい相関 ID を生成します。

MQC.MQPMO_NEW_MSG_ID *

送信されるメッセージごとに新しいメッセージ ID を生成します。

MQC.MQPMO_NONE

指定されるオプションはありません。他のオプションと一緒に使用しないでください。

MQC.MQPMO_NO_CONTEXT

このメッセージに関連するコンテキストはありません。

MQC.MQPMO_NO_SYNCPOINT

同期点制御を持たないメッセージを書き込みます。同期点制御オプションが指定されていない場合、同期点制御なしがデフォルトとみなされます。

MQC.MQPMO_PASS_ALL_CONTEXT

入力キュー・ハンドルからすべてのコンテキストを渡します。

MQC.MQPMO_PASS_IDENTITY_CONTEXT

入力キュー・ハンドルから識別コンテキストを渡します。

MQC.MQPMO_RESPONSE_AS_Q_DEF

MQDestination.put 呼び出しについては、このオプションで、キューの DEFPRESP 属性から PUT 応答タイプを取得します。

MQQueueManager.put 呼び出しについては、このオプションによって、呼び出しが同時に行われます。

MQC.MQPMO_RESPONSE_AS_TOPIC_DEF

MQC.MQPMO_RESPONSE_AS_TOPIC_DEF は、トピック・オブジェクトで使用される MQC.MQPMO_RESPONSE_AS_Q_DEF と同義です。

MQC.MQPMO_RETAIN

送信されたパブリケーションがキュー・マネージャーによって保存されます。このオプションが使用され、パブリケーションを保存できない場合、メッセージは公開されずに呼び出しが失敗し、MQC.MQRC_PUT_NOT_RETAINED が戻されます。

このパブリケーションが公開された後、MQSubscription.RequestPublicationUpdate メソッドを呼び出しすることによりそのコピーを要求します。保存されたパブリケーションは、MQC.MQSO_NEW_PUBLICATIONS_ONLY オプションの設定なしでサブスクリプションを作成するアプリケーションに対して送信されます。パブリケーションを受信したら、その MQIsRetained メッセージ・プロパティーを確認し、それが保存パブリケーションかどうかを検証します。

保存パブリケーションがサブスクライバーによって要求される場合、使用されるサブスクリプションのトピック・ストリングにワイルドカードが含まれていることがあります。トピック・ツリーに、そのサブスクリプションと一致する複数の保存パブリケーションがある場合、それらはすべて送信されます。

MQC.MQPMO_SET_ALL_CONTEXT

アプリケーションから、すべてのコンテキストを設定します。

MQC.MQPMO_SET_IDENTITY_CONTEXT

アプリケーションからすべての識別コンテキストを設定します。

MQC.MQPMO_SYNC_RESPONSE

このオプションによって、MQDestination.put または MQQueueManager.put 呼び出しが、完全応答データと同時に行われます。

MQC.MQPMO_SUPPRESS_REPLYTO

パブリケーションの ReplyToQueueName および ReplyToQueueManagerName フィールドに入力される情報は、サブスクライバーに渡されません。このオプションが、ReplyToQueueName を必要とするレポート・オプションと組み合わせて使用されると、呼び出しは失敗し、MQC.MQRC_MISSING_REPLY_TO_Q が戻されます。

MQC.MQPMO_SYNCPOINT

同期点制御を持つメッセージを書き込みます。メッセージは、作業単位がコミットされるまで、作業単位の外側には表示されません。作業単位がバックアウトされると、メッセージは除去されます。

```
public int RecordFields {get; set;} *
```

配布リストの情報。配布リストは .NET ではサポートされていません。

```
public string ResolvedQueueManagerName {get;}
```

キュー・マネージャーによって、リモート・キュー名で指定されるキューを所有するキュー・マネージャーの名前に設定される出力フィールド。ResolvedQueueManagerName は、キューがリモート・キューであるときに、キューにアクセスするキュー・マネージャーの名前とは異なる場合があります。

非空白値は、オブジェクトが単一のキューである場合のみ返されます。オブジェクトが配布リストまたはトピックである場合、返される値は未定義です。

```
public string ResolvedQueueName {get;}
```

キュー・マネージャーによって、メッセージが入れられるキューの名前に設定される出力フィールド。開かれたキューが別名またはモデル・キューであった場合、ResolvedQueueName は、キューをオープンするのに使用される名前とは異なるようです。

非空白値は、オブジェクトが単一のキューである場合のみ返されます。オブジェクトが配布リストまたはトピックである場合、返される値は未定義です。

```
public int UnknownDestCount {get;} *
```

通常は配布リストに使用される UnknownDestCount は、キュー・マネージャーによって設定される出力フィールドです。それによって、現在の呼び出しがリモート・キューを解決するキューへの送信に成功したメッセージの数が示されます

.NET は配布リストをサポートしていませんが、単一のキューをオープンするときには InvalidDestCount が設定されています。

コンストラクター

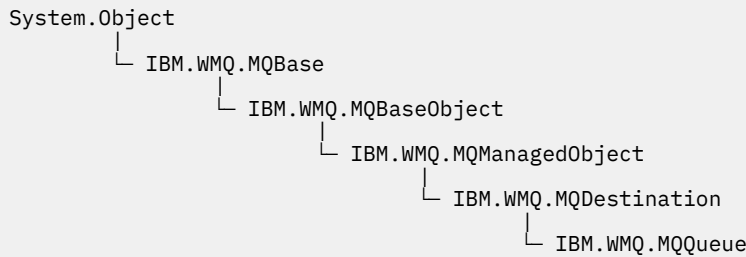
```
public MQPutMessageOptions();
```

オプション・セットの無い新しい MQPutMessageOptions オブジェクトと、空白の ResolvedQueueName および ResolvedQueueManagerName を構成します。

MQQueue.NET クラス

MQQueue を使用して、メッセージの送受信、および IBM MQ キューの属性を照会します。コンストラクター、または MQQueueManager.AccessProcess メソッドを使用して MQQueue オブジェクトを作成します。

クラス



```
public class IBM.WMQ.MQQueue extends IBM.WMQ.MQDestination;
```

- [1759 ページの『プロパティ』](#)
- [1761 ページの『方法』](#)
- [1764 ページの『コンストラクター』](#)

プロパティ

プロパティの取得時にスローされた MQException をテストします。

public int ClusterWorkLoadPriority {get;}

キューの優先順位を指定します。このパラメーターはローカル、リモート、および別名キューにのみ有効です。

public int ClusterWorkLoadRank {get;}

キューのランクを指定します。このパラメーターはローカル、リモート、および別名キューにのみ有効です。

public int ClusterWorkLoadUseQ {get;}

宛先キューにローカル・インスタンスと最低 1 つのリモート・クラスター・インスタンスがある場合に、MQPUT 操作の動作を指定します。MQPUT がクラスター・チャネルから出された場合、このパラメーターは適用されません。このパラメーターは、ローカル・キューにのみ有効です。

public DateTime CreationDateTime {get;}

このキューが作成された日時。

public int CurrentDepth {get;}

現在キューにあるメッセージの数を取得します。この値は、書き込み呼び出し中、および取得呼び出しのバックアウト中に増分されます。また、ブラウズ以外の取得呼び出し中、および書き込み呼び出しのバックアウト中に減分されます。

public int DefinitionType {get;}

キューが定義された方法。指定できる値は以下のとおりです。

- MQC.MQQDT_PREDEFINED
- MQC.MQQDT_PERMANENT_DYNAMIC
- MQC.MQQDT_TEMPORARY_DYNAMIC

public int InhibitGet {get; set;}

このキューまたはこのトピックの、どちらでメッセージを読み取るかを制御します。指定できる値は以下のとおりです。

- MQC.MQQA_GET_INHIBITED
- MQC.MQQA_GET_ALLOWED

public int InhibitPut {get; set;}

このキューまたはこのトピックの、どちらにメッセージを書き込むかを制御します。指定できる値は以下のとおりです。

- MQQA_PUT_INHIBITED
- MQQA_PUT_ALLOWED

public int MaximumDepth {get;}

同時にキューに存在できるメッセージの最大数。すでにこれだけ多くのメッセージが入っているキューにさらにメッセージを書き込もうとすると、理由コード MQC.MQRC_Q_FULL で失敗します。

public int MaximumMessageLength {get;}

このキューの各メッセージに含めることができるアプリケーション・データの最大長。この値より大きいメッセージを書き込もうとすると、理由コード MQC.MQRC_MSG_TOO_BIG_FOR_Q で失敗します。

public int NonPersistentMessageClass {get;}

このキューに書き込まれる非持続メッセージの信頼性のレベル。

public int OpenInputCount {get;}

キューからメッセージを削除するために現在有効なハンドルの数。OpenInputCount は、アプリケーションによって作成されたハンドルのみではなく、ローカル・キュー・マネージャーに認識されている有効な入力ハンドルの総数です。

public int OpenOutputCount {get;}

キューにメッセージを追加するために現在有効なハンドルの数。OpenOutputCount は、アプリケーションによって作成されたハンドルのみではなく、ローカル・キュー・マネージャーに認識されている有効な出力ハンドルの総数です。

public int QueueAccounting {get;}

キューのアカウントング情報のコレクションを使用可能に設定できるようにするかどうかを指定します。

public int QueueMonitoring {get;}

キューのモニターを使用可能に設定できるようにするかどうかを指定します。

public int QueueStatistics {get;}

キューの統計のコレクションを使用可能に設定できるようにするかどうかを指定します。

public int QueueType {get;}

このキューのタイプを次のいずれかの値で戻します。

- MQC.MQQT_ALIAS
- MQC.MQQT_LOCAL
- MQC.MQQT_REMOTE
- MQC.MQQT_CLUSTER

public int Shareability {get;}

入力のためにキューを複数回オープンできるかどうか。指定できる値は以下のとおりです。

- MQC.MQQA_SHAREABLE
- MQC.MQQA_NOT_SHAREABLE

public string TPIPE {get;}

IMSTPIPEIBM MQ ブリッジを使用して OTMA と通信する際に使用する名前。

public int TriggerControl {get; set;}

アプリケーションのキュー・サービスを開始するために、トリガー・メッセージが開始キューに書き込まれるかどうか。指定できる値は以下のとおりです。

- MQC.MQTC_OFF
- MQC.MQTC_ON

public string TriggerData {get; set;}

キュー・マネージャーがトリガー・メッセージに挿入する自由形式のデータ。このキューに到着したメッセージによって、トリガー・メッセージが開始キューに書き込まれることになった場合に、TriggerData が挿入されます。ストリングの可能な最大長は、MQC.MQ_TRIGGER_DATA_LENGTH によって示されます。

public int TriggerDepth {get; set;}

トリガー・タイプが MQC.MQTT_DEPTH に設定されているときに、トリガー・メッセージが書き込まれる前に、キューに入れなければならないメッセージ数。

```
public int TriggerMessagePriority {get; set;}
```

メッセージによってトリガー・メッセージが生成されない場合のメッセージ優先順位。すなわち、キュー・マネージャーは、トリガーを生成するかどうかを判定するときに、これらのメッセージを無視します。値がゼロの場合、すべてのメッセージによってトリガー・メッセージが生成されます。

```
public int TriggerType {get; set;}
```

このキューにメッセージが到着する結果として、トリガー・メッセージが書き込まれる条件。指定できる値は以下のとおりです。

- MQC.MQTT_NONE
- MQC.MQTT_FIRST
- MQC.MQTT EVERY
- MQC.MQTT_DEPTH

方法

```
public void Get(MQMessage message);
```

```
public void Get(MQMessage message, MQGetMessageOptions getMessageOptions);
```

```
public void Get(MQMessage message, MQGetMessageOptions getMessageOptions, int MaxMsgSize);
```

MQException をスローします。

キューからメッセージを取得します。

読み取りに失敗した場合は、MQMessage オブジェクトは変更されません。それが成功した場合は、MQMessage のメッセージ記述子とメッセージ・データ部分が、着信メッセージのメッセージ記述子とメッセージ・データに置き換わります。

IBM MQ への、特定の MQQueueManager からの呼び出しは、すべて同期されています。そのため読み取り待機を実行すると、同じ MQQueueManager を使用しているすべてのスレッドについて、読み取りの呼び出しが確立するまで、次の IBM MQ の呼び出しが行われません。同時に IBM MQ にアクセスするために複数のスレッドが必要である場合、各スレッドがそれ自身の MQQueueManager オブジェクトを作成する必要があります。

メッセージ

メッセージ記述子と返されたメッセージ・データを含んでいます。メッセージ記述子のいくつかのフィールドは、入力パラメーターです。MessageId および CorrelationId 入力パラメーターが、必須として設定されているのを確認することが重要です。

再接続可能なクライアントは再接続に成功すると、理由コード MQRC_BACKED_OUT を、MQGM_SYNCPOINT で受信したメッセージに対して返します。

getMessageOptions

読み取りのアクションを制御するオプション。

オプション MQC.MQGM_CONVERT を使用すると、結果として、1 バイト文字コードを 2 バイトコードに変換するときに、理由コード MQC.MQRC_CONVERTED_STRING_TOO_BIG の例外になる可能性があります。この場合、メッセージは変換されずにバッファーにコピーされます。

getMessageOptions が指定されていない場合、使用されるメッセージ・オプションは MQGM_NOWAIT です。

再接続可能なクライアントで MQGM_LOGICAL_ORDER オプションを使用すると、MQRC_RECONNECT_INCOMPATIBLE 理由コードが返されます。

MaxMsgSize

このメッセージ・オブジェクトで受信する最大メッセージ。キューのメッセージがこのサイズより大きい場合、次の 2 つのいずれかが行われます。

- MQGMO_ACCEPT_TRUNCATED_MSG フラグが MQGetMessageOptions オブジェクトに設定された場合、メッセージは、最大限までメッセージ・データで埋められます。例外が、MQCC_WARNING 完了コードと MQRC_TRUNCATED_MSG_ACCEPTED 理由コードと共にスローされます。
- MQGMO_ACCEPT_TRUNCATED_MSG フラグが設定されていない場合、メッセージはキューに残されます。例外が、MQCC_WARNING 完了コードと MQRC_TRUNCATED_MSG_FAILED 理由コードと共にスローされます。

MaxMsgSize が指定されていない場合、メッセージ全体が取得されます。

```
public void Put(MQMessage message);  
public void Put(MQMessage message, MQPutMessageOptions putMessageOptions);
```

MQException をスローします。

メッセージをキューに書き込みます。

書き込み呼び出しが成功した後に MQMessage オブジェクトを変更しても、IBM MQ キューまたはパブリケーション・トピックの実際のメッセージには影響がありません。

Put により、MQMessage の MessageId プロパティと CorrelationId プロパティを更新します。メッセージ・データはクリアされません。Put または Get 呼び出しは、MQMessage オブジェクト内の更新情報を参照します。例えば、次のコード・スニペットでは、最初のメッセージが a を含み、2 番目のメッセージが ab を含んでいます。

```
msg.WriteString("a");  
q.Put(msg, pmo);  
msg.WriteString("b");  
q.Put(msg, pmo);
```

メッセージ

メッセージ記述子データを含む MQMessage オブジェクト、および送信されるメッセージ。このメソッドの結果、メッセージ記述子は変更することができます。このメソッドの完了直後のメッセージ記述子の値は、キューに書き込みされた値か、トピックに公開された値です。

再接続可能なクライアントには、次の理由コードが返されます。

- MQRC_CALL_INTERRUPTED 持続メッセージで書き込みの呼び出しをしていて再接続が成功している間に、接続に失敗した場合。
- MQRC_NONE 非持続メッセージで書き込みの呼び出しをしている間、接続が成功している場合 (『[Application Recovery](#)』を参照)。

putMessageOptions

書き込みのアクションを制御するオプション。

putMessageOptions が指定されていない場合、MQPutMessageOptions のデフォルト・インスタンスが使用されます。

再接続可能なクライアントで MQPMO_LOGICAL_ORDER オプションを使用すると、MQRC_RECONNECT_INCOMPATIBLE 理由コードが返されます。

注: 簡単かつ効率的に単一メッセージをキューに書き込みするには、MQQueueManager.Put オブジェクトを使用します。このためには MQQueue オブジェクトを持っている必要があります。

```
public void PutForwardMessage(MQMessage message);  
public void PutForwardMessage(MQMessage message, MQPutMessageOptions  
putMessageOptions);
```

MQException をスローします

message が元のメッセージである場合のキューに対してメッセージが書き込まれます。

メッセージ

メッセージ記述子データを含む MQMessage オブジェクト、および送信されるメッセージ。このメソッドの結果、メッセージ記述子は変更することができます。このメソッドの完了直後のメッセージ記述子の値は、キューに書き込みされた値か、トピックに公開された値です。

再接続可能なクライアントには、次の理由コードが返されます。

- MQRC_CALL_INTERRUPTED 持続メッセージで書き込みの呼び出しをしていて再接続が成功している間に、接続に失敗した場合。
- MQRC_NONE 非持続メッセージで書き込みの呼び出しをしている間、接続が成功している場合 (『Application Recovery』を参照)。

putMessageOptions

書き込みのアクションを制御するオプション。

putMessageOptions が指定されていない場合、MQPutMessageOptions のデフォルト・インスタンスが使用されます。

再接続可能なクライアントで MQPMO_LOGICAL_ORDER オプションを使用すると、MQRC_RECONNECT_INCOMPATIBLE 理由コードが返されます。

```
public void PutReplyMessage(MQMessage message)  
public void PutReplyMessage(MQMessage message, MQPutMessageOptions  
putMessageOptions)
```

MQException をスローします。

message が元のメッセージである場合のキューに対して応答メッセージが書き込まれます。

メッセージ

メッセージ記述子と返されたメッセージ・データを含んでいます。メッセージ記述子のいくつかのフィールドは、入力パラメーターです。MessageId および CorrelationId 入力パラメーターが、必須として設定されているのを確認することが重要です。

再接続可能なクライアントは再接続に成功すると、理由コード MQRC_BACKED_OUT を、MQGM_SYNCPOINT で受信したメッセージに対して返します。

putMessageOptions

書き込みのアクションを制御するオプション。

putMessageOptions が指定されていない場合、MQPutMessageOptions のデフォルト・インスタンスが使用されます。

再接続可能なクライアントで MQPMO_LOGICAL_ORDER オプションを使用すると、MQRC_RECONNECT_INCOMPATIBLE 理由コードが返されます。

```
public void PutReportMessage(MQMessage message)  
public void PutReportMessage(MQMessage message, MQPutMessageOptions  
putMessageOptions)
```

MQException をスローします。

message が元のメッセージである場合のキューに対してレポート・メッセージが書き込まれます。

メッセージ

メッセージ記述子と返されたメッセージ・データを含んでいます。メッセージ記述子のいくつかのフィールドは、入力パラメーターです。MessageId および CorrelationId 入力パラメーターが、必須として設定されているのを確認することが重要です。

再接続可能なクライアントは再接続に成功すると、理由コード MQRC_BACKED_OUT を、MQGM_SYNCPOINT で受信したメッセージに対して返します。

putMessageOptions

書き込みのアクションを制御するオプション。

putMessageOptions が指定されていない場合、MQPutMessageOptions のデフォルト・インスタンスが使用されます。

再接続可能なクライアントで MQPMO_LOGICAL_ORDER オプションを使用すると、MQRC_RECONNECT_INCOMPATIBLE 理由コードが返されます。

コンストラクター

```
public MQQueue MQQueueManager.AccessQueue(string queueName, int openOptions);  
public MQQueue MQQueueManager.AccessQueue(string queueName, int openOptions,  
string queueManagerName, string dynamicQueueName, string alternateUserId);
```

MQException をスローします。

このキュー・マネージャーのキューにアクセスします。

メッセージの取得または参照、メッセージの書き込み、キューの属性を照会するか、またはキューの属性を設定する。指定されたキューがモデル・キューである場合は、動的ローカル・キューが作成されます。MQQueue 結果オブジェクトの name 属性を照会し、動的キューの名前を検索します。

queueName

オープンするキューの名前。

openOptions

キューのオープンを制御するオプション。

MQC.MQOO_ALTERNATE_USER_AUTHORITY

指定したユーザー ID で検証します。

MQC.MQOO_BIND_AS_QDEF

キューのデフォルトのバインディングを使用します。

MQC.MQOO_BIND_NOT_FIXED

特定の宛先にバインドしません。

MQC.MQOO_BIND_ON_OPEN

キューがオープンされたときに、ハンドルを宛先にバインドします。

MQC.MQOO_BROWSE

メッセージのブラウズ用にオープンします。

MQC.MQOO_FAIL_IF QUIESCING

キュー・マネージャーが静止中の場合は失敗します。

MQC.MQOO_INPUT_AS_Q_DEF

キュー定義のデフォルトを使用したメッセージの読み取り用にオープンします。

MQC.MQOO_INPUT_SHARED

共有アクセスによるメッセージの読み取り用にオープンします。

MQC.MQOO_INPUT_EXCLUSIVE

排他的アクセスによるメッセージの読み取り用にオープンします。

MQC.MQOO_INQUIRE

照会用にオープンします。プロパティを照会する場合に必要です。

MQC.MQOO_OUTPUT

メッセージの書き込み用にオープンします。

MQC.MQOO_PASS_ALL_CONTEXT

すべてのコンテキストを渡すことができますようにします。

MQC.MQOO_PASS_IDENTITY_CONTEXT

識別コンテキストを渡すことができます。

MQC.MQOO_SAVE_ALL_CONTEXT

メッセージが取り出されるときにコンテキストを保管します。

MQC.MQOO_SET

属性の設定のためにオープンします。プロパティを設定するときに必要です。

MQC.MQOO_SET_ALL_CONTEXT

すべてのコンテキストの設定を許可します。

MQC.MQOO_SET_IDENTITY_CONTEXT

ID コンテキストの設定を許可します。

queueManagerName

キューが定義されているキュー・マネージャーの名前。全体がブランクまたはヌルである名前は、MQQueueManager オブジェクトが接続されているキュー・マネージャーを表します。

dynamicQueueName

dynamicQueueName は、*queueName* がモデル・キュー名を指定しなければ、無視されます。指定された場合、*dynamicQueueName* は、作成する動的キューの名前を指定します。ブランクまたはヌルである名前は、*queueName* がモデル・キュー名を指定している場合は無効になります。名前の最後の空白でない文字がアスタリスクの場合、キューマネージャはアスタリスク * を文字列に置換します。文字によって、キューのために生成される名前が、このキュー・マネージャー上で確実に固有のものとなります。

alternateUserId

MQC.MQOO_ALTERNATE_USER_AUTHORITY が *openOptions* パラメーターで指定されている場合、*alternateUserId* は、オープンに関する権限を確認するために使用される代替ユーザー ID を指定します。MQC.MQOO_ALTERNATE_USER_AUTHORITY が指定されていない場合、*alternateUserId* はブランクまたはヌルのままにすることができます。

```
public MQQueue(MQQueueManager queueManager, string queueName, int openOptions,  
string queueManagerName, string dynamicQueueName, string alternateUserId);
```

MQException をスローします。

queueManager のキューにアクセスします。

メッセージの取得または参照、メッセージの書き込み、キューの属性を照会するか、またはキューの属性を設定する。指定されたキューがモデル・キューである場合は、動的ローカル・キューが作成されます。MQQueue 結果オブジェクトの *name* 属性を照会し、動的キューの名前を検索します。

queueManager

アクセス対象のキューがあるキュー・マネージャー。

queueName

オープンするキューの名前。

openOptions

キューのオープンを制御するオプション。

MQC.MQOO_ALTERNATE_USER_AUTHORITY

指定したユーザー ID で検証します。

MQC.MQOO_BIND_AS_QDEF

キューのデフォルトのバインディングを使用します。

MQC.MQOO_BIND_NOT_FIXED

特定の宛先にバインドしません。

MQC.MQOO_BIND_ON_OPEN

キューがオープンされたときに、ハンドルを宛先にバインドします。

MQC.MQOO_BROWSE

メッセージのブラウズ用にオープンします。

MQC.MQOO_FAIL_IF QUIESCING

キュー・マネージャーが静止中の場合は失敗します。

MQC.MQOO_INPUT_AS_Q_DEF

キュー定義のデフォルトを使用したメッセージの読み取り用にオープンします。

MQC.MQOO_INPUT_SHARED

共有アクセスによるメッセージの読み取り用にオープンします。

MQC.MQOO_INPUT_EXCLUSIVE

排他的アクセスによるメッセージの読み取り用にオープンします。

MQC.MQOO_INQUIRE

照会用にオープンします。プロパティを照会する場合に必要です。

MQC.MQOO_OUTPUT

メッセージの書き込み用にオープンします。

MQC.MQOO_PASS_ALL_CONTEXT

すべてのコンテキストを渡すことができますようにします。

MQC.MQOO_PASS_IDENTITY_CONTEXT

識別コンテキストを渡すことができます。

MQC.MQOO_SAVE_ALL_CONTEXT

メッセージが取り出されるときにコンテキストを保管します。

MQC.MQOO_SET

属性の設定のためにオープンします。プロパティを設定するときに必要です。

MQC.MQOO_SET_ALL_CONTEXT

すべてのコンテキストの設定を許可します。

MQC.MQOO_SET_IDENTITY_CONTEXT

ID コンテキストの設定を許可します。

queueManagerName

キューが定義されているキュー・マネージャーの名前。全体がブランクまたはヌルである名前は、MQQueueManager オブジェクトが接続されているキュー・マネージャーを表します。

dynamicQueueName

dynamicQueueName は、*queueName* がモデル・キュー名を指定しなければ、無視されます。指定された場合、*dynamicQueueName* は、作成する動的キューの名前を指定します。ブランクまたはヌルである名前は、*queueName* がモデル・キュー名を指定している場合は無効になります。名前の最後の空白でない文字がアスタリスクの場合、キューマネージャはアスタリスク * を文字列に置換します。文字によって、キューのために生成される名前が、このキュー・マネージャー上で確実に固有のものとなります。

alternateUserId

MQC.MQOO_ALTERNATE_USER_AUTHORITY が *openOptions* パラメーターで指定されている場合、*alternateUserId* は、オープンに関する権限を確認するために使用される代替ユーザー ID を指定します。MQC.MQOO_ALTERNATE_USER_AUTHORITY が指定されていない場合、*alternateUserId* はブランクまたはヌルのままにすることができます。

MQQueueManager.NET クラス

MQQueueManager を使用してキュー・マネージャーに接続し、キュー・マネージャー・オブジェクトにアクセスします。これにより、トランザクションも制御されます。MQQueueManager コンストラクターによって、クライアントまたはサーバー接続が作成されます。

クラス

```
System.Object
├── IBM.WMQ.MQBase
│   └── IBM.WMQ.MQBaseObject
│       └── IBM.WMQ.ManagedObject
│           └── IBM.WMQ.MQQueueManager
```

```
public class IBM.WMQ.MQQueueManager extends IBM.WMQ.MQManagedObject;
```

- [1767 ページの『プロパティ』](#)
- [1770 ページの『方法』](#)
- [1776 ページの『コンストラクター』](#)

プロパティ

プロパティの取得時にスローされた MQException をテストします。

public int AccountingConnOverride {get;}

アプリケーションが、MQI アカウンティングとキュー・アカウンティングの値の設定を指定変更できるかどうかを指定します。

public int AccountingInterval {get;}

中間アカウンティング・レコードが書き込まれるまでの時間 (秒)。

public int ActivityRecording {get;}

活動レポートの生成を制御します。

public int AdoptNewMCACheck {get;}

新しいインバウンド・チャンネルが検出されたときに MCA を採用するかどうかを判断するために検査するエレメントを指定します。採用するには、MCA 名がアクティブな MCA 名と一致する必要があります。

public int AdoptNewMCAInterval {get;}

孤立したチャンネルが終了するまで新しいチャンネルが待機する時間 (秒)。

public int AdoptNewMCAType {get;}

AdoptNewMCACheck 値と一致する新しいインバウンド・チャンネル要求が検出されたときに、孤立した MCA インスタンスを採用 (再始動) するかどうかを指定します。

public int BridgeEvent {get;}

IMS ブリッジ・イベントを生成するかどうか。

public int ChannelEvent {get;}

チャンネル・イベントを生成するかどうか。

public int ChannelInitiatorControl {get;}

キュー・マネージャーが開始するときに、チャンネル・イニシエーターが自動的に開始するかどうかを指定します。

public int ChannelInitiatorAdapters {get;}

IBM MQ 呼び出しを処理するためのアダプター・サブタスクの数。

public int ChannelInitiatorDispatchers {get;}

チャンネル・イニシエーターで使用するディスパッチャーの数。

public int ChannelInitiatorTraceAutoStart {get;}

チャンネル・イニシエーター・トレースが自動的に開始されるかどうかを指定します。

public int ChannelInitiatorTraceTableSize {get;}

チャンネル・イニシエーターのトレース・データ・スペースのサイズ (メガバイト)。

public int ChannelMonitoring {get;}

チャンネル・モニターを使用するかどうかを指定します。

public int ChannelStatistics {get;}

チャンネルの統計データの収集を制御します。

public int CharacterSet {get;}

キュー・マネージャーのコード化文字セット ID (CCSID) を返します。CharacterSet は、アプリケーション・プログラミング・インターフェースのすべての文字ストリング・フィールドについて、キュー・マネージャーによって使用されます。

public int ClusterSenderMonitoring {get;}

自動的に定義されたクラスター送信側チャンネルに関するオンライン・モニター・データの収集を制御します。

public int ClusterSenderStatistics {get;}

自動的に定義されたクラスター送信側チャンネルの統計データの収集を制御します。

public int ClusterWorkLoadMRU {get;}

アウトバウンド・クラスター・チャンネルの最大数。

public int ClusterWorkLoadUseQ {get;}

QMGR の値が指定されている場合、ClusterWorkLoadUseQ、MQQueue プロパティのデフォルト値。

public intCommandEvent {get;}

コマンド・イベントを生成するかどうかを指定します。

public string CommandInputQueueName {get;}

キュー・マネージャーで定義されているコマンド入力キューの名前を返します。アプリケーションは、許可されていれば、このキューにコマンドを送信することができます。

public int CommandLevel {get;}

キュー・マネージャーの関数レベルを表します。特定の関数レベルに対応する関数のセットは、プラットフォームに依存します。特定のプラットフォームでは、すべてのキュー・マネージャーに共通である最低限の関数レベルで、関数をサポートするキュー・マネージャーのどれにでも依存することができます。

public int CommandLevel {get;}

キュー・マネージャーが開始するときに、コマンド・サーバーが自動的に開始するかどうかを指定します。

public string DNSGroup {get;}

使用されなくなりました。

public int DNSWLM {get;}

使用されなくなりました。

public int IPAddressVersion {get;}

チャンネル接続に使用する IP プロトコル (IPv4 または IPv6)。

public boolean IsConnected {get;}

isConnected の値を返します。

true の場合、キュー・マネージャーとの接続は確立済みですが、障害が発生しているかどうかは不明です。キュー・マネージャーに到達するための IsConnected への接続試行が現在行われていないため、物理接続に障害が発生した可能性があります。ただし、その場合でも、IsConnected は true を返すはずです。IsConnected 状態は、メッセージの書き込みや取得がキュー・マネージャーで実際に行われた場合にのみ更新されます。

false は、キュー・マネージャーとの接続は確立されていないこと、接続に障害が発生したこと、あるいは切断されたことを示します。

public int KeepAlive {get;}

接続の他方の終端がまだ使用可能であることを確認するために、TCP KEEPALIVE 機能を使用するかどうかを指定します。使用不可の場合は、チャンネルが閉じられます。

public int ListenerTimer {get;}

APPC または TCP/IP で障害が発生した後に IBM MQ がリスナーの再始動を試行する秒単位の時間間隔です。

public int LoggerEvent {get;}

ロガー・イベントを生成するかどうかを指定します。

public string LU62ARMSuffix {get;}

SYS1.PARMLIB の APPCPM メンバーの接尾部。この接尾部は、このチャンネル・イニシエーターの LUADD を指名します。自動リスタート・マネージャー (ARM) がチャンネル・イニシエーターを再始動すると、z/OS コマンド SET APPC=xx が発行されます。

public string LUGroupName {get; z/os}

キュー共有グループの インバウンド伝送を処理する LU 6.2 リスナーに使用する総称 LU 名。

public string LUName {get;}

アウトバウンド LU 6.2 伝送に使用する LU の名前。

public int MaximumActiveChannels {get;}

任意の時点でアクティブなチャンネルの最大数。

public int MaximumCurrentChannels {get;}

いつでも現行チャンネルにできるチャンネルの最大数 (接続されたクライアントとのサーバー接続チャンネルも含まれます)。

public int MaximumLU62Channels {get;}

LU 6.2 伝送プロトコルを使用する、現行チャンネルにすることが可能なチャンネルの最大数、または接続できるクライアントの最大数。

public int MaximumMessageLength {get;}

キュー・マネージャーが処理できるメッセージの最大長 (バイト) を返します。メッセージ長が MaximumMessageLength を超える場合、キューは定義できません。

public int MaximumPriority {get;}

キュー・マネージャーによってサポートされる最大メッセージ優先順位を返します。優先順位の範囲はゼロ (最低) からこの値までです。キュー・マネージャーから切断した後にこのメソッドを呼び出すと、MQException をスローします。

public int MaximumTCPChannels {get;}

TCP/IP 伝送プロトコルを使用する、現行チャンネルの最大数、または接続可能なクライアントの最大数。

public int MQIAccounting {get;}

MQI データに関するアカウンティング情報の収集を制御します。

public int MQIStatistics {get;}

キュー・マネージャーに関する統計モニター情報の収集を制御します。

public int OutboundPortMax {get;}

発信チャンネルのバインディング時に使用されるポート番号の範囲の最大値。

public int OutboundPortMin {get;}

発信チャンネルのバインディング時に使用されるポート番号の範囲の最小値。

public int QueueAccounting {get;}

クラス 3 アカウンティング (スレッド・レベルとキュー・レベルのアカウンティング) のデータをすべてのキューで使用するかどうかを指定します。

public int QueueMonitoring {get;}

キューに関するオンライン・モニター・データの収集を制御します。

public int QueueStatistics {get;}

キューに関する統計データの収集を制御します。

public int ReceiveTimeout {get;}

TCP/IP チャンネルが、非アクティブ状態に戻る前に、そのパートナーからの (ハートビートを含む) データの受信を待機する 時間です。

public int ReceiveTimeoutMin {get;}

非アクティブ状態に戻る前に、パートナーからハートビートを含むデータを受信するために、TCP/IP チャンネルが待機する最小時間。

public int ReceiveTimeoutType {get;}

ReceiveTimeout の値に適用する修飾子。

public int SharedQueueQueueManagerName {get;}

共有キューへのメッセージの送信方法を指定します。ターゲット・キュー・マネージャーとして同じキュー共有グループとは異なるキュー・マネージャーを、書き込み指定した場合、メッセージは次の 2 とおりの方法で送信されます。

MQC.MQSQQM_USE

メッセージは、共有キューに入れられる前に、オブジェクト・キュー・マネージャーに送信されます。

MQCMQSQQM_IGNORE

メッセージは、共有キューに直接入れられます。

public int SSLEvent {get;}

TLS イベントを生成するかどうか。

public int SSLFips {get;}

暗号処理を暗号ハードウェアではなくソフトウェア IBM MQ で行う場合、認定アルゴリズム FIPS のみを使用すべきかどうか。

public int SSLKeyResetCount {get;}

秘密鍵が再折衝される前に、TLS 会話内で送受信する非暗号化バイト数を示します。

public int ClusterSenderStatistics {get;}

連続した統計収集と統計収集の間隔を分単位で指定します。

public int SyncpointAvailability {get;}

キュー・マネージャーが、MQQueue.get および MQQueue.put メソッドとの作業単位と同期点をサポートするかどうかを示します。

public string TCPName {get;}

TCPStackType の値に応じて、使用される唯一の、あるいはデフォルトの TCP/IP システムの名前。

public int TCPStackType {get;}

チャンネル・イニシエーターが、TCPName で指定された TCP/IP アドレス・スペースのみを使用するかどうかを指定します。あるいはチャンネル・イニシエーターが、いずれかの TCP/IP アドレスにバインドすることもできます。

public int TraceRouteRecording {get;}

経路トレース情報の記録を制御します。

方法

public MQProcess AccessProcess(string processName, int openOptions);

public MQProcess AccessProcess(string processName, int openOptions, string queueManagerName, string alternateUserId);

MQException をスローします。

プロセス属性を照会するために、このキュー・マネージャーの IBM MQ プロセスにアクセスします。

processName

オープンするプロセスの名前。

openOptions

プロセスのオープンを制御するオプション。追加またはビット単位 OR を使用した結合が可能である、有効なオプションは次の通りです。

- MQC.MQ00_FAIL_IF_QUIESCING
- MQC.MQ00_INQUIRE
- MQC.MQ00_SET
- MQC.MQ00_ALTERNATE_USER_AUTHORITY

queueManagerName

プロセスが定義されたキュー・マネージャーの名前。キュー・マネージャーが、プロセスがアクセスしているものと同じである場合、そのキュー・マネージャーの名前はブランクまたはヌルのままにすることができます。

alternateUserId

openOptions パラメーターで MQC.MQ00_ALTERNATE_USER_AUTHORITY が指定されている場合、*alternateUserId* は、アクションの許可を確認するために使用される代替ユーザー ID を指定します。MQ00_ALTERNATE_USER_AUTHORITY が指定されていない場合、*alternateUserId* はブランクまたはヌルのままにすることができます。

MQC.MQ00_ALTERNATE_USER_AUTHORITY が指定されていない場合、デフォルトのユーザー権限を使用してキュー・マネージャーに接続します。

public MQQueue AccessQueue(string queueName, int openOptions);

public MQQueue AccessQueue(string queueName, int openOptions, string queueManagerName, string dynamicQueueName, string alternateUserId);

MQException をスローします。

このキュー・マネージャーのキューにアクセスします。

メッセージの取得または参照、メッセージの書き込み、キューの属性を照会するか、またはキューの属性を設定する。指定されたキューがモデル・キューである場合は、動的ローカル・キューが作成されます。MQQueue 結果オブジェクトの name 属性を照会し、動的キューの名前を検索します。

queueName

オープンするキューの名前。

openOptions

キューのオープンを制御するオプション。

MQC.MQOO_ALTERNATE_USER_AUTHORITY

指定したユーザー ID で検証します。

MQC.MQOO_BIND_AS_QDEF

キューのデフォルトのバインディングを使用します。

MQC.MQOO_BIND_NOT_FIXED

特定の宛先にバインドしません。

MQC.MQOO_BIND_ON_OPEN

キューがオープンされたときに、ハンドルを宛先にバインドします。

MQC.MQOO_BROWSE

メッセージのブラウズ用にオープンします。

MQC.MQOO_FAIL_IF QUIESCING

キュー・マネージャーが静止中の場合は失敗します。

MQC.MQOO_INPUT_AS_Q_DEF

キュー定義のデフォルトを使用したメッセージの読み取り用にオープンします。

MQC.MQOO_INPUT_SHARED

共有アクセスによるメッセージの読み取り用にオープンします。

MQC.MQOO_INPUT_EXCLUSIVE

排他的アクセスによるメッセージの読み取り用にオープンします。

MQC.MQOO_INQUIRE

照会用にオープンします。プロパティを照会する場合に必要です。

MQC.MQOO_OUTPUT

メッセージの書き込み用にオープンします。

MQC.MQOO_PASS_ALL_CONTEXT

すべてのコンテキストを渡すことができますようにします。

MQC.MQOO_PASS_IDENTITY_CONTEXT

識別コンテキストを渡すことができます。

MQC.MQOO_SAVE_ALL_CONTEXT

メッセージが取り出されるときにコンテキストを保管します。

MQC.MQOO_SET

属性の設定のためにオープンします。プロパティを設定するときに必要です。

MQC.MQOO_SET_ALL_CONTEXT

すべてのコンテキストの設定を許可します。

MQC.MQOO_SET_IDENTITY_CONTEXT

ID コンテキストの設定を許可します。

queueManagerName

キューが定義されているキュー・マネージャーの名前。全体がブランクまたはヌルである名前は、MQQueueManager オブジェクトが接続されているキュー・マネージャーを表します。

dynamicQueueName

dynamicQueueName は、*queueName* がモデル・キュー名を指定しなければ、無視されます。指定された場合、*dynamicQueueName* は、作成する動的キューの名前を指定します。ブランクまたはヌルである名前は、*queueName* がモデル・キュー名を指定している場合は無効になります。名前の最後の空白でない文字がアスタリスクの場合、キューマネージャはアスタリスク * を文字列に

置換します。文字によって、キューのために生成される名前が、このキュー・マネージャー上で確実に固有のものとなります。

alternateUserId

MQC.MQOO_ALTERNATE_USER_AUTHORITY が openOptions パラメーターで指定されている場合、*alternateUserId* は、オープンに関する権限を確認するために使用される代替ユーザー ID を指定します。MQC.MQOO_ALTERNATE_USER_AUTHORITY が指定されていない場合、*alternateUserId* はブランクまたはヌルのままにすることができます。

```
public MQTopic AccessTopic( MQDestination destination, string topicName, string
topicObject, int options);
public MQTopic AccessTopic( MQDestination destination, string topicName, string
topicObject, int options, string alternateUserId);
public MQTopic AccessTopic( MQDestination destination, string topicName, string
topicObject, int options, string alternateUserId, string subscriptionName);
public MQTopic AccessTopic( MQDestination destination, string topicName, string
topicObject, int options, string alternateUserId, string subscriptionName,
System.Collections.Hashtable properties);
public MQTopic AccessTopic(string topicName, string topicObject, int openAs,
int options);
public MQTopic AccessTopic(string topicName, string topicObject, int openAs,
int options, string alternateUserId);
public MQTopic AccessTopic(string topicName, string topicObject, int options,
string alternateUserId, string subscriptionName);
public MQTopic AccessTopic(string topicName, string topicObject, int options,
string alternateUserId, string subscriptionName, System.Collections.Hashtable
properties);
```

このキュー・マネージャーのトピックにアクセスします。

MQTopic オブジェクトは、トピック・オブジェクトと呼ばれることもある管理トピック・オブジェクトに密接に関連しています。入力では、*topicObject* は、管理トピック・オブジェクトを指します。MQTopic コンストラクターは、トピック・オブジェクトからトピック・ストリングを取得し、それを *topicName* と結合してトピック名を作成します。*topicObject*、*topicName*、またはその両方をヌルに設定できます。トピック名は、トピック・ツリーに対して突き合わせされ、管理トピック・オブジェクトに最も類似する名前が *topicObject* で返されます。

MQTopic オブジェクトに関連付けられたトピックは、2つのトピック・ストリングを結合した結果です。最初のトピック・ストリングは、*topicObject* によって特定された管理トピック・オブジェクトによって定義されます。2番目のトピック・ストリングは、*topicString* です。MQTopic オブジェクトに関連付けられた、結果のトピック・ストリングは、ワイルドカードを含めることにより、複数のトピックを特定できます。

トピックがオープンされているのがパブリッシュ用か、サブスクライブ用かに応じて、MQTopic.Put メソッドを使用してトピックにパブリッシュしたり、MQTopic.Get メソッドを使用してトピックのパブリケーションを受け取ったりします。同じトピックに対してパブリッシュおよびサブスクライブをする場合は、トピックに2回アクセスする必要があります。1回はパブリッシュ用、もう1回はサブスクライブ用です。

MQDestination オブジェクトを指定せずに、サブスクリプション用に MQTopic オブジェクトを作成する場合は、管理対象サブスクリプションが指定されます。MQDestination オブジェクトとしてキューを引き渡す場合は、管理対象外のサブスクリプションが使用されます。設定したサブスクリプション・オプションは、サブスクリプションの管理状態 (管理対象か、管理対象外か) と矛盾しないようにする必要があります。

destination

destination は、MQQueue インスタンスです。*destination* を指定することにより、MQTopic は管理対象外のサブスクリプションとしてオープンされます。トピックに関するパブリケーションは、*destination* としてアクセスされるキューに配信されます。

topicName

トピック名の 2 番目の部分であるトピック・ストリング。 *topicName* は、 *topicObject* 管理トピック・オブジェクトで定義されたトピック・ストリングと連結されます。 *topicName* をヌルに設定できます。 この場合、トピック名は、 *topicObject* のトピック・ストリングによって定義されます。

topicObject

入力では、 *topicObject* は、トピック名の最初の部分を構成するトピック・ストリングを含むトピック・オブジェクトの名前です。 *topicObject* のトピック・ストリングは、 *topicName* と連結されます。 トピック・ストリングの構成に関する規則は、 [トピック・ストリングの結合](#) で定義されます。

出力では、 *topicObject* には、トピック・ストリングによって特定されるトピックへのトピック・ツリー内で最も類似する管理トピック・オブジェクトの名前が含まれます。

openAs

パブリッシュまたはサブスクライブするトピックにアクセスします。 パラメーターには、以下のオプションのうちいずれか 1 つのみ含めることができます。

- MQC.MQTOPIC_OPEN_AS_SUBSCRIPTION
- MQC.MQTOPIC_OPEN_AS_PUBLICATION

オプション

パブリッシュ用またはサブスクリプション用のトピックのオープンを制御するオプションを結合します。 MQC.MQSO_* 定数を使用してサブスクリプション用のトピックにアクセスし、MQC.MQOO_* 定数を使用してパブリケーション用のトピックにアクセスします。

複数のオプションが必要な場合は、値を合計するか、ビット単位の OR 演算子を使用してオプションの値を結合します。

alternateUserId

操作の完了に必要な権限を検査するために使用する代替ユーザー ID を指定します。 options パラメーターに MQC.MQOO_ALTERNATE_USER_AUTHORITY または MQC.MQSO_ALTERNATE_USER_AUTHORITY を設定した場合は、 *alternateUserId* を指定する必要があります。

subscriptionName

オプション MQC.MQSO_DURABLE または MQC.MQSO_ALTER が指定されている場合、 *subscriptionName* は必須です。 いずれの場合でも、MQTopic がサブスクリプション用に暗黙的にオープンされます。 MQC.MQSO_DURABLE が設定され、サブスクリプションが存在する場合、または MQC.MQSO_ALTER が設定され、サブスクリプションが存在しない場合は、例外がスローされます。

プロパティー

ハッシュ・テーブルを使用してリストされた特別なサブスクリプション・プロパティーを設定します。 ハッシュ・テーブルで指定されたエントリーは、出力値によりアップデートされます。 出力値をレポートするためにエントリーがハッシュ・テーブルに追加されることはありません。

- MQC.MQSUB_PROP_ALTERNATE_SECURITY_ID
- MQC.MQSUB_PROP_SUBSCRIPTION_EXPIRY
- MQC.MQSUB_PROP_SUBSCRIPTION_USER_DATA
- MQC.MQSUB_PROP_SUBSCRIPTION_CORRELATION_ID
- MQC.MQSUB_PROP_PUBLICATION_PRIORITY
- MQC.MQSUB_PROP_PUBLICATION_ACCOUNTING_TOKEN
- MQC.MQSUB_PROP_PUBLICATION_APPLICATIONID_DATA

public MQAsyncStatus GetAsyncStatus();

MQException をスローします

キュー・マネージャー接続の非同期アクティビティーを示す、MQAsyncStatus オブジェクトを返します。

public void Backout();

MQException をスローします。

前の同期点以降の同期点内で読み取りまたは書き込みされたメッセージをすべてバックアウトします。

MQC.MQPMO_SYNCPOINT フラグ・セットを使用して書き込まれたメッセージは、キューから除去されます。MQC.MQGMO_SYNCPOINT フラグで読み取られたメッセージは、元のキューに復元されます。持続メッセージである場合は変更がログに記録されます。

再接続可能クライアントの場合、再接続が成功すると、MQRC_NONE 理由コードがクライアントに戻されます。

public void Begin();

MQException をスローします。

Begin は、サーバー・バインディング・モードでのみサポートされます。それによってグローバルな作業単位が始動されます。

public void Commit();

MQException をスローします。

前の同期点以降の同期点内で読み取りまたは書き込みされたメッセージをすべてコミットします。

MQC.MQPMO_SYNCPOINT フラグ・セットを使用して書き込まれたメッセージは、他のアプリケーションで使用可能になります。MQC.MQGMO_SYNCPOINT フラグ・セットを使用して取得されたメッセージは削除されます。持続メッセージである場合は変更がログに記録されます。

再接続可能なクライアントには、次の理由コードが返されます。

- コミット呼び出しの実行中に接続が失われた場合は MQRC_CALL_INTERRUPTED。
- 再接続後にコミット呼び出しが発行された場合は MQRC_BACKED_OUT。

Disconnect();

MQException をスローします。

キュー・マネージャーへの接続をクローズします。このキュー・マネージャーでアクセスされるすべてのオブジェクトは、このアプリケーションにアクセスできなくなりました。オブジェクトに再度アクセスするには、MQQueueManager オブジェクトを作成します。

通常、作業単位の一部として実行された作業は、すべてコミットされます。ただし、作業単位が .NET によって管理されている場合は、作業単位がロールバックされる可能性があります。

```
public void Put(int type, string destinationName, MQMessage message);  
public void Put(int type, string destinationName, MQMessage message  
MQPutMessageOptions putMessageOptions);  
public void Put(int type, string destinationName, string queueManagerName,  
string topicString, MQMessage message);  
public void Put(string queueName, MQMessage message);  
public void Put(string queueName, MQMessage message, MQPutMessageOptions  
putMessageOptions);  
public void Put(string queueName, string queueManagerName, MQMessage message);  
public void Put(string queueName, string queueManagerName, MQMessage message,  
MQPutMessageOptions putMessageOptions);  
public void Put(string queueName, string queueManagerName, MQMessage message,  
MQPutMessageOptions putMessageOptions, string alternateUserId);
```

MQException をスローします。

MQQueue オブジェクトまたは MQTopic オブジェクトをあらかじめ作成しないで、キューまたはトピックに単一メッセージを入れます。

queueName

メッセージを入れるキューの名前。

destinationName

宛先オブジェクトの名前。 *type* の値に応じて、キューまたはトピックのいずれかになります。

タイプ

宛先オブジェクトのタイプ。 オプションは結合してはいけません。

MQC.MQOT_Q

キュー

MQC.MQOT_TOPIC

トピック

queueManagerName

キューが定義されたキュー・マネージャーの名前またはキュー・マネージャーの別名。 タイプ **MQC.MQOT_TOPIC** が指定されている場合、このパラメーターは無視されます。

キューがモデル・キューで、解決されたキュー・マネージャーの名前がこのキュー・マネージャーではない場合、MQException がスローされます。

topicString

topicString は、*destinationName* トピック・オブジェクト内のトピック名と結合されます。

destinationName がキューの場合、*topicString* は無視されます。

メッセージ

送信するメッセージ。 メッセージは、入出力のオブジェクトです。

再接続可能なクライアントには、次の理由コードが返されます。

- 持続メッセージに対する Put 呼び出しの実行中に接続が切断された場合は **MQRC_CALL_INTERRUPTED**。
- 非持続メッセージに対して Put 呼び出しを実行中に接続が成功した場合は **MQRC_NONE** ([アプリケーション・リカバリー](#)を参照)。

putMessageOptions

書き込みのアクションを制御するオプション。

putMessageOptions を省略すると、*putMessageOptions* のデフォルト・インスタンスが作成されます。 *putMessageOptions* は入出力オブジェクトです。

再接続可能なクライアントで **MQPMO_LOGICAL_ORDER** オプションを使用すると、**MQRC_RECONNECT_INCOMPATIBLE** 理由コードが返されます。

alternateUserId

キューにメッセージを入れるときに、許可を確認するために使用する代替ユーザー ID を指定します。

putMessageOptions で **MQC.MQOO_ALTERNATE_USER_AUTHORITY** を設定しない場合は、*alternateUserId* を省略できます。 **MQC.MQOO_ALTERNATE_USER_AUTHORITY** を設定する場合は、*alternateUserId* も設定する必要があります。 *alternateUserId* は、**MQC.MQOO_ALTERNATE_USER_AUTHORITY** も設定しない限り有効になりません。

コンストラクター

```
public MQQueueManager();  
public MQQueueManager(string queueManagerName);  
public MQQueueManager(string queueManagerName, Int options);  
public MQQueueManager(string queueManagerName, Int options, string channel,  
string connName);  
public MQQueueManager(string queueManagerName, string channel, string  
connName);  
public MQQueueManager(string queueManagerName, System.Collections.Hashtable  
properties);
```

MQException をスローします。

キュー・マネージャーへの接続を作成します。クライアント接続またはサーバー接続のどちらを作成するか選択します。

キュー・マネージャーに接続するには、キュー・マネージャーに対する照会 (inq) 権限が必要です。照会権限がない場合、接続試行は失敗します。

次の条件のいずれか 1 つに該当する場合、クライアント接続が作成されます。

1. `channel` または `connName` はコンストラクターで指定されます。
2. `HostName`、`Port`、または `Channel` は、`properties` で指定されます。
3. `MQEnvironment.HostName`、`MQEnvironment.Port`、または `MQEnvironment.Channel` が指定されています。

接続プロパティーの値が、示された順序でデフォルトにされます。コンストラクター内の `channel` および `connName` は、コンストラクター内のプロパティー値よりも優先されます。コンストラクター・プロパティーの値は、`MQEnvironment` プロパティーよりも優先されます。

ホスト名、チャンネル名、およびポートは、`MQEnvironment` クラスで定義されます。

queueManagerName

キュー・マネージャーの名前、または接続先のキュー・マネージャー・グループ。

パラメーターを省略する、またはそれをヌルのままにする、またはデフォルトのキュー・マネージャー・セレクションを作成するためにブランクにする。サーバーでのデフォルトのキュー・マネージャー接続は、サーバーでのデフォルトのキュー・マネージャーに対するものです。クライアント接続でのデフォルトのキュー・マネージャー接続は、リスナーが接続されたキュー・マネージャーに対するものです。

オプション

MQCNO 接続オプションを指定します。その値は、作成される接続のタイプに適合するものである必要があります。例えば、次のサーバー接続プロパティーをクライアント接続として指定すると、MQException がスローされます。

- `MQC.MQCNO_FASTPATH_BINDING`
- `MQC.MQCNO_STANDARD_BINDING`

プロパティー

プロパティー・パラメーターは、`MQEnvironment` によってプロパティー設定を指定変更する、一連のキー値のペアを取ります。[1779 ページの『MQEnvironment プロパティーの指定変更』](#)の例を参照してください。次のプロパティーは指定変更できます。

- `MQC.CONNECT_OPTIONS_PROPERTY`
- `MQC.CONNECTION_NAME_PROPERTY`
- `MQC.ENCRYPTION_POLICY_SUITE_B`
- `MQC.HOST_NAME_PROPERTY`
- `MQC.PORT_PROPERTY`

- MQC.CHANNEL_PROPERTY
- MQC.SSL_CIPHER_SPEC_PROPERTY
- MQC.SSL_PEER_NAME_PROPERTY
- MQC.SSL_CERT_STORE_PROPERTY
- MQC.SSL_CRYPTO_HARDWARE_PROPERTY
- MQC.SECURITY_EXIT_PROPERTY
- MQC.SECURITY_USERDATA_PROPERTY
- MQC.SEND_EXIT_PROPERTY
- MQC.SEND_USERDATA_PROPERTY
- MQC.RECEIVE_EXIT_PROPERTY
- MQC.RECEIVE_USERDATA_PROPERTY
- MQC.USER_ID_PROPERTY
- MQC.PASSWORD_PROPERTY
- MQC.MQAIR_ARRAY
- MQC.KEY_RESET_COUNT
- MQC.FIPS_REQUIRED
- MQC.HDR_CMP_LIST
- MQC.MSG_CMP_LIST
- MQC.TRANSPORT_PROPERTY

channel

サーバー接続チャンネルの名前

connName

接続名 (ポート) *HostName* の形式

CONNECTION_NAME_PROPERTY を使用して、コン MQQueueManager(String queueManagerName, Hashtable properties) ストラクタの引数としてホスト名とポートのリストを指定できます。

以下に例を示します。

```

ConnectionName = "fred.mq.com(2344),nick.mq.com(3746),tom.mq.com(4288)";
Hashtable Properties=new Hashtable();
properties.Add(MQC.CONNECTION_NAME_PROPERTY,ConnectionName);
MQQueueManager qmgr=new MQQueue Manager("qmgrname",properties);

```

接続が試行される時、接続名のリストは順序に従って処理されます。最初のホスト名とポートへの接続の試行が失敗すると、2 番目の属性のペアへの接続が試行されます。クライアントは、接続に成功するか、リストを使用し尽くすまで、このプロセスを繰り返します。リストを使用し尽くした場合、適切な理由コードと完了コードが、クライアント・アプリケーションに返されます。

接続名にポート番号が指定されていない場合、デフォルト・ポート (mqclient.ini で構成) が使用されます。

接続リストを設定する

自動クライアント再接続オプションが設定されている場合、次のメソッドを使用して接続リストを設定することができます。

MQSERVER による接続リストの設定

コマンド・プロンプトによって接続リストを設定します。

コマンド・プロンプトで、次のコマンドを設定します。

```
MQSERVER=SYSTEM.DEF.SVRCONN/TCP/Hostname1(Port1),Hostname2(Por2),Hostname3(Port3)
```

以下に例を示します。

```
MQSERVER=SYSTEM.DEF.SVRCONN/TCP/fred.mq.com(5266),nick.mq.com(6566),jack.mq.com(8413)
```

接続を MQSERVER で設定するときには、アプリケーション内で設定しないでください。

接続リストをアプリケーション内で設定すると、MQSERVER 環境変数で設定されているすべてが、アプリケーションによって上書きされてしまいます。

アプリケーションによる接続リストの設定

ホスト名とポートのプロパティを指定して、アプリケーションに接続リストを設定することができます。

```
String connName = "fred.mq.com(2344), nick.mq.com(3746), chris.mq.com(4288)";  
MQQueueManager qm = new MQQueueManager("QM1", "TestChannel", connName);
```

app.config を使用して接続リストを設定します

App.config は、キーと値のペアを指定する XML ファイルです。

接続リストでは次のように指定します。

```
<app.Settings>  
<add key="Connection1" value="Hostname1(Port1)"/>  
<add key="Connection2" value="Hostname2(Port2)"/>  
</app.Settings>
```

以下に例を示します。

```
<app.Settings>  
<add key="Connection1" value="fred.mq.com(2966)"/>  
<add key="Connection2" value="alex.mq.com(6533)"/>  
</app.Settings>
```

app.config ファイル内の接続リストを直接変更することができます。

MQEnvironment を使用して接続リストを設定します

MQEnvironment で接続リストを設定するには、*ConnectionName* プロパティを使用します。

```
MQEnvironment.ConnectionName = "fred.mq.com(4288),alex.mq.com(5211);
```

ConnectionName プロパティは、MQEnvironment のホスト名とポートのプロパティ設定を上書きします。

クライアント接続を作成する

次の例では、キュー・マネージャーへのクライアント接続の作成方法を示しています。新しい MQQueueManager オブジェクトを作成する前に、MQEnvironment 変数を設定してクライアント接続を作成できます。

```
MQEnvironment.Hostname = "fred.mq.com"; // host to connect to  
MQEnvironment.Port     = 1414;          // port to connect to  
                                // If not explicitly set,  
                                // defaults to 1414  
                                // (the default IBM MQ port)  
MQEnvironment.Channel  = "channel.name"; // the case sensitive  
                                // name of the  
                                // SVR CONN channel on  
                                // the queue manager  
MQQueueManager qMgr    = new MQQueueManager("MYQM");
```

図 10. クライアント接続

MQEnvironment プロパティの指定変更

次の例では、キュー・マネージャーを、ハッシュ・テーブルで定義したそのユーザー ID とパスワードで作成する方法が示されています。

```
Hashtable properties = new Hashtable();

properties.Add( MQC.USER_ID_PROPERTY, "ExampleUserId" );
properties.Add( MQC.PASSWORD_PROPERTY, "ExamplePassword" );

try
{
    MQQueueManager qMgr = new MQQueueManager("qmgrname", properties);
}
catch (MQException mqe)
{
    System.Console.WriteLine("Connect failed with " + mqe.Message);
    return((int)mqe.Reason);
}
```

図 11. MQEnvironment プロパティの指定変更

再接続可能な接続を作成する

次の例では、クライアントをキュー・マネージャーに、自動的に再接続する方法を示しています。

```
Hashtable properties = new Hashtable(); // The queue manager name and the
                                         // properties how it has to be connected

properties.Add(MQC.CONNECT_OPTIONS_PROPERTY, MQC.MQCNO_RECONNECT); // Options
                                         // through which reconnection happens

properties.Add(MQC.CONNECTION_NAME_PROPERTY, "fred.mq.com(4789),nick.mq.com(4790)"); // The list
                                         // of queue managers through which reconnection happens

MQ QueueManager qmgr = new MQQueueManager("qmgrname", properties);
```

図 12. クライアントを自動的にキュー・マネージャーと再接続

MQSubscription.NET クラス

MQSubscription を使用して、保存パブリケーションが、サブスクライバーに対して送信されるように要求します。MQSubscription は、公開されたサブスクリプションである MQTopic オブジェクトのプロパティです。

クラス

```
System.Object
├── IBM.WMQ.MQBase
│   └── IBM.WMQ.MQBaseObject
│       └── IBM.WMQ.MQManagedObject
│           └── IBM.WMQ.MQSubscription
```

```
public class IBM.WMQ.MQSubscription extends IBM.WMQ.MQManagedObject;
```

- [1779 ページの『プロパティ』](#)
- [1780 ページの『方法』](#)
- [1780 ページの『コンストラクター』](#)

プロパティ

MQManagedObject クラスを使用して、サブスクリプション・プロパティにアクセスします。[1738 ページの『プロパティ』](#)を参照してください。

方法

MQManagedObject クラスを使用して、サブスクリプション Inquire、Set および Get メソッドにアクセスします。[1739 ページの『方法』](#)を参照してください。

public int RequestPublicationUpdate(int options);

MQException をスローします。

現在のトピックの更新されたパブリケーションを要求します。トピックに関する保存パブリケーションがキュー・マネージャー中にある場合、それらはサブスクライバーに送信されます。

RequestPublicationUpdate を呼び出しする前に、トピックをサブスクリプションに対してオープンにして、MQSubscription オブジェクトを取得します。

通常は、MQC.MQSO_PUBLICATIONS_ON_REQUEST オプションでサブスクリプションをオープンにします。トピック・ストリング中にワイルドカードがない場合は、この呼び出しの結果として1つのパブリケーションのみが送信されます。トピック・ストリング中にワイルドカードがある場合は、多くのパブリケーションが送信されます。メソッドは、サブスクリプション・キューに送信される保存パブリケーションの数を返します。特にそれらが非永続メッセージであった場合、これだけ多くのパブリケーションを受信できるという保証はありません。

オプション

MQC.MQSO_FAIL_IF QUIESCING

キュー・マネージャーが静止状態である場合、メソッドは失敗します。z/OS では、CICS または IMS アプリケーションの場合、接続が静止状態であれば、

MQC.MQSO_FAIL_IF QUIESCING はメソッドを強制的に失敗させます。

MQC.MQSO_NONE

オプションが指定されていません。

コンストラクター

public コンストラクターはありません。

MQSubscription オブジェクトが、公開されたサブスクリプションである、MQTopic オブジェクトの SubscriptionReference プロパティに返されます。

RequestPublicationUpdate メソッドを呼び出します。MQSubscription は、MQManagedObject のサブクラスです。参照を使用して、MQManagedObject のプロパティとメソッドにアクセスします。

MQTopic.NET クラス

MQTopic を使用して、トピックのメッセージをパブリッシュまたはサブスクライブしたり、トピックの属性を照会または設定したりします。コンストラクターまたは MQQueueManager.AccessTopic メソッドを使用し、MQTopic オブジェクトを作成してパブリッシュまたはサブスクライブします。

クラス

```
System.Object
├── IBM.WMQ.MQBase
│   └── IBM.WMQ.MQBaseObject
│       └── IBM.WMQ.MQManagedObject
│           └── IBM.WMQ.MQDestination
│               └── IBM.WMQ.MQTopic
```

public class IBM.WMQ.MQTopic extends IBM.WMQ.MQDestination;

- [1781 ページの『プロパティ』](#)

- [1781 ページの『方法』](#)
- [1783 ページの『コンストラクター』](#)

プロパティ

プロパティの取得時にスローされた `MQException` をテストします。

public Boolean IsDurable {get;}

サブスクリプションが永続である場合は `True` を返し、その他の場合は `False` を返す読み取り専用プロパティ。トピックがパブリケーション用にオープンされていた場合、プロパティは無視され、常に `False` が返されます。

public Boolean IsManaged {get;};

サブスクリプションがキュー・マネージャーによって管理されている場合は `True` を返し、その他の場合は `False` を返す読み取り専用プロパティ。トピックがパブリケーション用にオープンされていた場合、プロパティは無視され、常に `False` が返されます。

public Boolean IsSubscribed {get;};

トピックがサブスクリプション用にオープンされていた場合は `True` を返し、トピックがパブリケーション用にオープンされていた場合は `False` を返す読み取り専用プロパティ。

public MQSubscription SubscriptionReference {get;};

サブスクリプション用にオープンされたトピック・オブジェクトに関連する `MQSubscription` オブジェクトを返す読み取り専用プロパティ。close オプションを変更する場合、またはオブジェクトのメソッドを開始する場合は、この参照を使用できます。

public MQDestination UnmanagedDestinationReference {get;};

管理対象外のサブスクリプションに関連する `MQQueue` を返す読み取り専用プロパティ。これは、トピック・オブジェクトが作成されたときに指定された宛先です。このプロパティでは、パブリケーション用にオープンされたトピック・オブジェクト、または管理対象のサブスクリプションでオープンされたトピック・オブジェクトの場合は `Null` が返されます。

方法

public void Put(MQMessage message);

public void Put(MQMessage message, MQPutMessageOptions putMessageOptions);

`MQException` をスローします。

メッセージをトピックにパブリッシュします。

書き込み呼び出しが成功した後に `MQMessage` オブジェクトを変更しても、IBM MQ キューまたはパブリケーション・トピックの実際のメッセージには影響がありません。

`Put` により、`MQMessage` の `MessageId` プロパティと `CorrelationId` プロパティを更新します。メッセージ・データはクリアされません。`Put` または `Get` 呼び出しは、`MQMessage` オブジェクト内の更新情報を参照します。例えば、次のコード・スニペットでは、最初のメッセージが `a` を含み、2 番目のメッセージが `ab` を含んでいます。

```
msg.WriteString("a");
q.Put(msg,pmo);
msg.WriteString("b");
q.Put(msg,pmo);
```

メッセージ

メッセージ記述子データを含む `MQMessage` オブジェクト、および送信されるメッセージ。このメソッドの結果、メッセージ記述子を変更することができます。このメソッドの完了直後のメッセージ記述子の値は、キューに書き込みされた値か、トピックに公開された値です。

再接続可能なクライアントには、次の理由コードが返されます。

- `MQRC_CALL_INTERRUPTED` 持続メッセージで書き込みの呼び出しをしていて再接続が成功している間に、接続に失敗した場合。

- MQRC_NONE 非持続メッセージで書き込みの呼び出しをしている間、接続が成功している場合 (『Application Recovery』を参照)。

putMessageOptions

書き込みのアクションを制御するオプション。

putMessageOptions が指定されていない場合、MQPutMessageOptions のデフォルト・インスタンスが使用されます。

再接続可能なクライアントで MQPMO_LOGICAL_ORDER オプションを使用すると、MQRC_RECONNECT_INCOMPATIBLE 理由コードが返されます。

注: 簡単かつ効率的に単一メッセージをキューに書き込みするには、MQQueueManager.Put オブジェクトを使用します。このためには MQQueue オブジェクトを持っている必要があります。

```
public void Get(MQMessage message);
public void Get(MQMessage message, MQGetMessageOptions getMessageOptions);
public void Get(MQMessage message, MQGetMessageOptions getMessageOptions, int
MaxMsgSize);
```

MQException をスローします。

メッセージをトピックから取り出します。

このメソッドは、MQGetMessageOptions のデフォルト・インスタンスを使用して取得を実行します。使用されるメッセージ・オプションは MQGMO_NOWAIT です。

読み取りに失敗した場合は、MQMessage オブジェクトは変更されません。それが成功した場合は、MQMessage のメッセージ記述子とメッセージ・データ部分が、着信メッセージのメッセージ記述子とメッセージ・データに置き換わります。

IBM MQ への、特定の MQQueueManager からの呼び出しは、すべて同期されています。そのため読み取り待機を実行すると、同じ MQQueueManager を使用しているすべてのスレッドについて、読み取りの呼び出しが確立するまで、次の IBM MQ の呼び出しが行われません。同時に IBM MQ にアクセスするために複数のスレッドが必要である場合、各スレッドがそれ自身の MQQueueManager オブジェクトを作成する必要があります。

メッセージ

メッセージ記述子と返されたメッセージ・データを含んでいます。メッセージ記述子のいくつかのフィールドは、入力パラメーターです。MessageId および CorrelationId 入力パラメーターが、必須として設定されているのを確認することが重要です。

再接続可能なクライアントは再接続に成功すると、理由コード MQRC_BACKED_OUT を、MQGM_SYNCPOINT で受信したメッセージに対して返します。

getMessageOptions

読み取りのアクションを制御するオプション。

オプション MQC.MQGMO_CONVERT を使用すると、結果として、1 バイト文字コードを 2 バイトコードに変換するときに、理由コード MQC.MQRC_CONVERTED_STRING_TOO_BIG の例外になる可能性があります。この場合、メッセージは変換されずにバッファーにコピーされます。

getMessageOptions が指定されていない場合、使用されるメッセージ・オプションは MQGMO_NOWAIT です。

再接続可能なクライアントで MQGMO_LOGICAL_ORDER オプションを使用すると、MQRC_RECONNECT_INCOMPATIBLE 理由コードが返されます。

MaxMsgSize

このメッセージ・オブジェクトで受信する最大メッセージ。キューのメッセージがこのサイズより大きい場合、次の 2 つのいずれかが行われます。

- MQGMO_ACCEPT_TRUNCATED_MSG フラグが MQGetMessageOptions オブジェクトに設定された場合、メッセージは、最大限までメッセージ・データで埋められます。例外が、MQCC_WARNING 完了コードと MQRC_TRUNCATED_MSG_ACCEPTED 理由コードと共にスローされます。

- MQGMO_ACCEPT_TRUNCATED_MSG フラグが設定されていない場合、メッセージはキューに残されます。例外が、MQCC_WARNING 完了コードと MQRC_TRUNCATED_MSG_FAILED 理由コードと共にスローされます。

MaxMsgSize が指定されていない場合、メッセージ全体が取得されます。

コンストラクター

```
public MQTopic(MQQueueManager queueManager, MQDestination destination, string
topicName, string topicObject, int options);
public MQTopic(MQQueueManager queueManager, MQDestination destination, string
topicName, string topicObject, int options, string alternateUserId);
public MQTopic(MQQueueManager queueManager, MQDestination destination, string
topicName, string topicObject, int options, string alternateUserId, string
subscriptionName);
public MQTopic(MQQueueManager queueManager, MQDestination destination, string
topicName, string topicObject, int options, string alternateUserId, string
subscriptionName, System.Collections.Hashtable properties);
public MQTopic(MQQueueManager queueManager, string topicName, string
topicObject, int openAs, int options);
public MQTopic(MQQueueManager queueManager, string topicName, string
topicObject, int openAs, int options, string alternateUserId);
public MQTopic(MQQueueManager queueManager, string topicName, string
topicObject, int options, string alternateUserId, string subscriptionName);
public MQTopic(MQQueueManager queueManager, string topicName, string
topicObject, int options, string alternateUserId, string subscriptionName,
System.Collections.Hashtable properties);
```

queueManager のトピックにアクセスします。

MQTopic オブジェクトは、トピック・オブジェクトと呼ばれることもある管理トピック・オブジェクトに密接に関連しています。入力では、*topicObject* は、管理トピック・オブジェクトを指します。MQTopic コンストラクターは、トピック・オブジェクトからトピック・ストリングを取得し、それを *topicName* と結合してトピック名を作成します。*topicObject*、*topicName*、またはその両方をヌルに設定できます。トピック名は、トピック・ツリーに対して突き合わせされ、管理トピック・オブジェクトに最も類似する名前が *topicObject* で返されます。

MQTopic オブジェクトに関連付けられたトピックは、2つのトピック・ストリングを結合した結果です。最初のトピック・ストリングは、*topicObject* によって特定された管理トピック・オブジェクトによって定義されます。2番目のトピック・ストリングは、*topicString* です。MQTopic オブジェクトに関連付けられた、結果のトピック・ストリングは、ワイルドカードを含めることにより、複数のトピックを特定できます。

トピックがオープンされているのがパブリッシュ用か、サブスクライブ用かに応じて、MQTopic.Put メソッドを使用してトピックにパブリッシュしたり、MQTopic.Get メソッドを使用してトピックのパブリケーションを受け取ったりします。同じトピックに対してパブリッシュおよびサブスクライブをする場合は、トピックに2回アクセスする必要があります。1回はパブリッシュ用、もう1回はサブスクライブ用です。

MQDestination オブジェクトを指定せずに、サブスクリプション用に MQTopic オブジェクトを作成する場合は、管理対象サブスクリプションが指定されます。MQDestination オブジェクトとしてキューを引き渡す場合は、管理対象外のサブスクリプションが使用されます。設定したサブスクリプション・オプションは、サブスクリプションの管理状態 (管理対象か、管理対象外か) と矛盾しないようにする必要があります。

queueManager

トピックにアクセスするキュー・マネージャー。

destination

destination は、MQQueue インスタンスです。 *destination* を指定することにより、MQTopic は管理対象外のサブスクリプションとしてオープンされます。 トピックに関するパブリケーションは、 *destination* としてアクセスされるキューに配信されます。

topicName

トピック名の 2 番目の部分であるトピック・ストリング。 *topicName* は、 *topicObject* 管理トピック・オブジェクトで定義されたトピック・ストリングと連結されます。 *topicName* をヌルに設定できます。 この場合、トピック名は、 *topicObject* のトピック・ストリングによって定義されます。

topicObject

入力では、 *topicObject* は、トピック名の最初の部分を構成するトピック・ストリングを含むトピック・オブジェクトの名前です。 *topicObject* のトピック・ストリングは、 *topicName* と連結されます。 トピック・ストリングの構成に関する規則は、 [トピック・ストリングの結合](#) で定義されます。

出力では、 *topicObject* には、トピック・ストリングによって特定されるトピックへのトピック・ツリー内で最も類似する管理トピック・オブジェクトの名前が含まれます。

openAs

パブリッシュまたはサブスクライブするトピックにアクセスします。 パラメーターには、以下のオプションのうちいずれか 1 つのみ含めることができます。

- MQC.MQTOPIC_OPEN_AS_SUBSCRIPTION
- MQC.MQTOPIC_OPEN_AS_PUBLICATION

オプション

パブリッシュ用またはサブスクリプション用のトピックのオープンを制御するオプションを結合します。 MQC.MQSO_* 定数を使用してサブスクリプション用のトピックにアクセスし、MQC.MQOO_* 定数を使用してパブリケーション用のトピックにアクセスします。

複数のオプションが必要な場合は、値を合計するか、ビット単位の OR 演算子を使用してオプションの値を結合します。

alternateUserId

操作の完了に必要な権限を検査するために使用する代替ユーザー ID を指定します。 options パラメーターに MQC.MQOO_ALTERNATE_USER_AUTHORITY または MQC.MQSO_ALTERNATE_USER_AUTHORITY を設定した場合は、 *alternateUserId* を指定する必要があります。

subscriptionName

オプション MQC.MQSO_DURABLE または MQC.MQSO_ALTER が指定されている場合、 *subscriptionName* は必須です。 いずれの場合でも、MQTopic がサブスクリプション用に暗黙的にオープンされます。 MQC.MQSO_DURABLE が設定され、サブスクリプションが存在する場合、または MQC.MQSO_ALTER が設定され、サブスクリプションが存在しない場合は、例外がスローされます。

プロパティー

ハッシュ・テーブルを使用してリストされた特別なサブスクリプション・プロパティーを設定します。 ハッシュ・テーブルで指定されたエントリーは、出力値によりアップデートされます。 出力値をレポートするためにエントリーがハッシュ・テーブルに追加されることはありません。

- MQC.MQSUB_PROP_ALTERNATE_SECURITY_ID
- MQC.MQSUB_PROP_SUBSCRIPTION_EXPIRY
- MQC.MQSUB_PROP_SUBSCRIPTION_USER_DATA
- MQC.MQSUB_PROP_SUBSCRIPTION_CORRELATION_ID
- MQC.MQSUB_PROP_PUBLICATION_PRIORITY
- MQC.MQSUB_PROP_PUBLICATION_ACCOUNTING_TOKEN

• MQC.MQSUB_PROP_PUBLICATION_APPLICATIONID_DATA

```
public MQTopic MQQueueManager.AccessTopic(MQDestination destination, string
topicName, string topicObject, int options);
public MQTopic MQQueueManager.AccessTopic(MQDestination destination, string
topicName, string topicObject, int options, string alternateUserId);
public MQTopic MQQueueManager.AccessTopic(MQDestination destination, string
topicName, string topicObject, int options, string alternateUserId, string
subscriptionName);
public MQTopic MQQueueManager.AccessTopic(MQDestination destination, string
topicName, string topicObject, int options, string alternateUserId, string
subscriptionName, System.Collections.Hashtable properties);
public MQTopic MQQueueManager.AccessTopic(string topicName, string topicObject,
int openAs, int options);
public MQTopic MQQueueManager.AccessTopic(string topicName, string topicObject,
int openAs, int options, string alternateUserId);
public MQTopic MQQueueManager.AccessTopic(string topicName, string topicObject,
int options, string alternateUserId, string subscriptionName);
public MQTopic MQQueueManager.AccessTopic(string topicName, string topicObject,
int options, string alternateUserId, string subscriptionName,
System.Collections.Hashtable properties);
```

このキュー・マネージャーのトピックにアクセスします。

MQTopic オブジェクトは、トピック・オブジェクトと呼ばれることもある管理トピック・オブジェクトに密接に関連しています。入力では、topicObject は、管理トピック・オブジェクトを指します。MQTopic コンストラクターは、トピック・オブジェクトからトピック・ストリングを取得し、それを topicName と結合してトピック名を作成します。topicObject、topicName、またはその両方をヌルに設定できます。トピック名は、トピック・ツリーに対して突き合わせされ、管理トピック・オブジェクトに最も類似する名前が topicObject で返されます。

MQTopic オブジェクトに関連付けられたトピックは、2つのトピック・ストリングを結合した結果です。最初のトピック・ストリングは、topicObject によって特定された管理トピック・オブジェクトによって定義されます。2番目のトピック・ストリングは、topicString です。MQTopic オブジェクトに関連付けられた、結果のトピック・ストリングは、ワイルドカードを含めることにより、複数のトピックを特定できます。

トピックがオープンされているのがパブリッシュ用か、サブスクライブ用かに応じて、MQTopic.Put メソッドを使用してトピックにパブリッシュしたり、MQTopic.Get メソッドを使用してトピックのパブリケーションを受け取ったりします。同じトピックに対してパブリッシュおよびサブスクライブをする場合は、トピックに2回アクセスする必要があります。1回はパブリッシュ用、もう1回はサブスクライブ用です。

MQDestination オブジェクトを指定せずに、サブスクリプション用に MQTopic オブジェクトを作成する場合は、管理対象サブスクリプションが指定されます。MQDestination オブジェクトとしてキューを引き渡す場合は、管理対象外のサブスクリプションが使用されます。設定したサブスクリプション・オプションは、サブスクリプションの管理状態 (管理対象か、管理対象外か) と矛盾しないようにする必要があります。

destination

destination は、MQQueue インスタンスです。*destination* を指定することにより、MQTopic は管理対象外のサブスクリプションとしてオープンされます。トピックに関するパブリケーションは、*destination* としてアクセスされるキューに配信されます。

topicName

トピック名の2番目の部分であるトピック・ストリング。*topicName* は、*topicObject* 管理トピック・オブジェクトで定義されたトピック・ストリングと連結されます。*topicName* をヌルに設定できます。この場合、トピック名は、*topicObject* のトピック・ストリングによって定義されます。

topicObject

入力では、*topicObject* は、トピック名の最初の部分を構成するトピック・ストリングを含むトピック・オブジェクトの名前です。*topicObject* のトピック・ストリングは、*topicName* と連結されます。トピック・ストリングの構成に関する規則は、[トピック・ストリングの結合](#)で定義されます。

出力では、*topicObject* には、トピック・ストリングによって特定されるトピックへのトピック・ツリー内で最も類似する管理トピック・オブジェクトの名前が含まれます。

openAs

パブリッシュまたはサブスクライブするトピックにアクセスします。パラメーターには、以下のオプションのうちいずれか1つのみ含めることができます。

- MQC.MQTOPIC_OPEN_AS_SUBSCRIPTION
- MQC.MQTOPIC_OPEN_AS_PUBLICATION

オプション

パブリッシュ用またはサブスクリプション用のトピックのオープンを制御するオプションを結合します。MQC.MQSO_* 定数を使用してサブスクリプション用のトピックにアクセスし、MQC.MQOO_* 定数を使用してパブリケーション用のトピックにアクセスします。

複数のオプションが必要な場合は、値を合計するか、ビット単位の OR 演算子を使用してオプションの値を結合します。

alternateUserId

操作の完了に必要な権限を検査するために使用する代替ユーザー ID を指定します。options パラメーターに MQC.MQOO_ALTERNATE_USER_AUTHORITY または MQC.MQSO_ALTERNATE_USER_AUTHORITY を設定した場合は、*alternateUserId* を指定する必要があります。

subscriptionName

オプション MQC.MQSO_DURABLE または MQC.MQSO_ALTER が指定されている場合、*subscriptionName* は必須です。いずれの場合でも、MQTopic がサブスクリプション用に暗黙的にオープンされます。MQC.MQSO_DURABLE が設定され、サブスクリプションが存在する場合、または MQC.MQSO_ALTER が設定され、サブスクリプションが存在しない場合は、例外がスローされます。

プロパティー

ハッシュ・テーブルを使用してリストされた特別なサブスクリプション・プロパティーを設定します。ハッシュ・テーブルで指定されたエントリーは、出力値によりアップデートされます。出力値をレポートするためにエントリーがハッシュ・テーブルに追加されることはありません。

- MQC.MQSUB_PROP_ALTERNATE_SECURITY_ID
- MQC.MQSUB_PROP_SUBSCRIPTION_EXPIRY
- MQC.MQSUB_PROP_SUBSCRIPTION_USER_DATA
- MQC.MQSUB_PROP_SUBSCRIPTION_CORRELATION_ID
- MQC.MQSUB_PROP_PUBLICATION_PRIORITY
- MQC.MQSUB_PROP_PUBLICATION_ACCOUNTING_TOKEN
- MQC.MQSUB_PROP_PUBLICATION_APPLICATIONID_DATA

IMQObjectTrigger.NET インターフェース

IMQObjectTrigger を実装して、**runmqdmn.NET** モニターによって渡されたメッセージを処理します。

インターフェース

```
public interface IBM.WMQMonitor.IMQObjectTrigger();
```

同期点制御が **runmqdmn** コマンドで指定されているかどうかによって、メッセージは、Execute メソッドが戻る前か後に、キューから削除されます。

方法

void Execute (MQQueueManager queueManager, MQQueue queue, MQMessage message, string param);

queueManager

モニターされているキューをホスティングしているキュー・マネージャー。

キュー

モニターされているキュー。

メッセージ

キューから読み取りされたメッセージ。

param

UserParameter から渡されたデータ。

MQC.NET インターフェース

定数名に接頭部 **MQC.** を付けることによって、MQI 定数を参照します。MQC は、MQI で使用するすべての定数を定義します。

インターフェース

```
System.Object
├── IBM.WMQ.MQC
```

```
public interface IBM.WMQ.MQC extends System.Object;
```

例

```
MQQueue queue;  
queue.closeOptions = MQC.MQCO_DELETE;
```

.NET アプリケーション用の文字セット ID

.NET IBM MQ メッセージのエンコード用に選択可能な文字セットの説明

文字セット	説明
37	ibm037
437	ibm437 / PC オリジナル
500	ibm500
819	iso-8859-1 / latin1 / ibm819
1200	Unicode
1208	UTF-8
273	ibm273
277	ibm277
278	ibm278
280	ibm280

文字セット	説明
284	ibm284
285	ibm285
297	ibm297
420	ibm420
424	ibm424
737	ibm737 / PC ギリシャ語
775	ibm775 / PC バルト語
813	iso-8859-7 / ギリシャ語 / ibm813
838	ibm838
850	ibm850 / PC Latin 1
852	ibm852 / PC Latin 2
855	ibm855 / PC キリル文字
856	ibm856
857	ibm857 / PC トルコ語
860	ibm860 / PC ポルトガル語
861	ibm861 / PC アイスランド語
862	ibm862 / PC ヘブライ語
863	ibm863 / PC カナダ・フランス語
864	ibm864 / PC アラビア語
865	ibm865 / PC 北欧ゲルマン系言語
868	ibm868
869	ibm869 / PC 現代ギリシャ語
870	ibm870
871	ibm871
874	ibm874
875	ibm875
912	iso-8859-2 / latin2 / ibm912
913	iso-8859-3 / latin3 / ibm913
914	iso-8859-4 / latin4 / ibm914
915	iso-8859-5 / キリル文字 / ibm915
916	iso-8859-8 / ヘブライ語 / ibm916
918	ibm918
920	iso-8859-9 / latin5 / ibm920
921	ibm921
922	ibm922

文字セット	説明
930	ibm930
932	PC 日本語
933	ibm933
935	ibm935
937	ibm937
939	ibm939
942	ibm942
943	ibm943
948	ibm948
949	ibm949
950	ibm950 / Big 5 中国語 (繁体字)
954	EUCJIS
964	ibm964 / CNS 11643 中国語 (繁体字)
970	ibm970
1006	ibm1006
1025	ibm1025
1026	ibm1026
1089	iso-8859-6 / アラビア語 / ibm1089
1097	ibm1097
1098	ibm1098
1112	ibm1112
1122	ibm1122
1123	ibm1123
1124	ibm1124
1250	Windows Latin 2
1251	Windows キリル文字
1252	Windows Latin 1
1253	Windows ギリシャ語
1254	Windows トルコ語
1255	Windows ヘブライ語
1256	Windows アラビア語
1257	Windows バルト語
1258	Windows ベトナム語
1381	ibm1381
1383	ibm1383

文字セット	説明
2022	JIS
5601	ksc-5601 ハングル
33722	ibm33722

IBM MQ C++ クラス

IBM MQ C++ クラスは、IBM MQ Message Queue Interface (MQI) をカプセル化します。このクラスをすべて扱う単一の C++ ヘッダー・ファイルとして、**imqi.hpp** があります。

各クラスごとに、次の情報が示されます。

クラス階層ダイアグラム

クラスをその直属の親クラス (ある場合) との継承関係で示すクラス・ダイアグラム。

その他の関連クラス

親クラスなどのその他の関連クラス、ならびにメソッド・シグニチャーで使用されるオブジェクトのクラスへの文書リンク。

オブジェクトの属性

そのクラスの属性。これらは、いずれの親クラスについても定義されている属性に加えられるものです。多くの属性が、IBM MQ データ構造体メンバー (1791 ページの『C++ と MQI の相互参照』を参照) を反映しています。詳細な説明については、791 ページの『オブジェクトの属性』を参照してください。

コンストラクター

そのクラスのオブジェクトを生成するのに使用される特別なメソッドのシグニチャー。

オブジェクト・メソッド (共有)

その命令についてクラスのインスタンスを必要とし、しかも使用上の制限のないメソッドのシグニチャー。

上記の情報が適用される場合は、次の情報も示されます。

クラス・メソッド (共有)

その命令についてクラスのインスタンスを必要とせず、しかも使用上の制限のないメソッドのシグニチャー。

多重定義された (親クラス) メソッド

親クラスで定義されているが、親クラスとは別の多様動作を実行する仮想メソッドのシグニチャー。

オブジェクト・メソッド (保護)

その命令についてクラスのインスタンスを必要とし、しかも派生したクラスの実装でできるように予約されているメソッドのシグニチャー。ここは、クラスの利用者でなくクラスの作成者だけに関係するセクションです。

オブジェクト・データ (保護)

派生したクラスの実装に使用できるオブジェクト・インスタンス・データについての実装の詳細。ここは、クラスの利用者でなくクラスの作成者だけに関係するセクションです。

理由コード

失敗したメソッドから予期される MQRC_* 値 (API 完了コードと理由コードを参照)。クラスのオブジェクトについて起こり得る理由コードを網羅したリストについては、親クラスの文書を参照してください。クラスの理由コードの文書リストには、親クラスの理由コードは記載されていません。

注:

- これらのクラスのオブジェクトは、スレッド・セーフではありません。このため、最適なパフォーマンスは保証されますが、複数のスレッドからオブジェクトにアクセスしないように注意してください。
- マルチスレッド・プログラムの場合は、スレッドごとに別個の ImqQueueManager オブジェクトを使用することをお勧めします。各マネージャー・オブジェクトには、異なるスレッド内のオブジェクトが相互に分離されるようにする、独自の独立した他のオブジェクト・コレクションが必要です。

クラスは、以下のとおりです。

- [1806 ページの『ImqAuthenticationRecord C++ クラス』](#)
- [1809 ページの『ImqBinary C++ クラス』](#)
- [1810 ページの『ImqCache C++ クラス』](#)
- [1813 ページの『ImqChannel C++ クラス』](#)
- [1819 ページの『ImqCICSBridgeHeader C++ クラス』](#)
- [1825 ページの『ImqDeadLetterHeader C++ クラス』](#)
- [1828 ページの『ImqDistributionList C++ クラス』](#)
- [1829 ページの『ImqError C++ クラス』](#)
- [1830 ページの『ImqGetMessageOptions C++ クラス』](#)
- [1833 ページの『ImqHeader C++ クラス』](#)
- [1835 ページの『ImqIMSBridgeHeader C++ クラス』](#)
- [1838 ページの『ImqItem C++ クラス』](#)
- [1839 ページの『ImqMessage C++ クラス』](#)
- [1846 ページの『ImqMessageTracker C++ クラス』](#)
- [1848 ページの『ImqNamelist C++ クラス』](#)
- [1850 ページの『ImqObject C++ クラス』](#)
- [1855 ページの『ImqProcess C++ クラス』](#)
- [1857 ページの『ImqPutMessageOptions C++ クラス』](#)
- [1859 ページの『ImqQueue C++ クラス』](#)
- [1869 ページの『ImqQueueManager C++ クラス』](#)
- [1885 ページの『ImqReferenceHeader C++ クラス』](#)
- [1888 ページの『ImqString C++ クラス』](#)
- [1893 ページの『ImqTrigger C++ クラス』](#)
- [1896 ページの『ImqWorkHeader C++ クラス』](#)

C++ と MQI の相互参照

この一連のトピックでは、C++ と MQI の関係について説明します。

この情報は、[177 ページの『MQI で使用されるデータ・タイプ』](#)と共にお読みください。

この表では、MQI のデータ構造体を C++ クラスおよびインクルード・ファイルに関連付けます。以下のトピックでは、各 C++ クラスの相互参照情報を示します。この相互参照は、基礎となる IBM MQ 手続き型インターフェースの使用法と関連しています。クラス ImqBinary、ImqDistributionList、および ImqString にはこのカテゴリーに分類される属性がなく、除外されます。

表 881. データ構造体、クラス、および組み込みファイルの相互参照		
データ構造体	クラス	組み込みファイル
MQAIR	ImqAuthenticationRecord	imqair.hpp
	ImqBinary	imqbin.hpp
	ImqCache	imqcac.hpp
MQCD	ImqChannel	imqchl.hpp
MQCIH	ImqBridgeCICS ヘッダー	imqcih.hpp
MQDLH	ImqDeadLetterHeader	imqdlh.hpp
MQOR	ImqDistributionList	imqdst.hpp
	ImqError	imqerr.hpp

表 881. データ構造体、クラス、および組み込みファイルの相互参照 (続き)

データ構造体	クラス	組み込みファイル
MQGMO	ImqGetMessageOptions	imqgmo.hpp
	ImqHeader	imqhdr.hpp
MQIIH	ImqBridgeIMS ヘッダー	imqihi.hpp
	ImqItem	imqitm.hpp
MQMD	ImqMessage	imqmsg.hpp
	ImqMessageTracker	imqmtr.hpp
	ImqNamelist	imqnml.hpp
MQOD、MQRR	ImqObject	imqobj.hpp
MQPMO、MQPMR、MQRR	ImqPutMessageOptions	imqpmo.hpp
	ImqProcess	imqpro.hpp
	ImqQueue	imqque.hpp
MQBO、MQCNO、MQCSP	ImqQueueManager	imqmgr.hpp
MQRMH	ImqReferenceHeader	imqrfh.hpp
	ImqString	imqstr.hpp
MQTM	ImqTrigger	imqtrg.hpp
MQTMC		
MQTMC2	ImqTrigger	imqtrg.hpp
MQXQH		
MQWIH	ImqWorkHeader	imqwih.hpp

ImqAuthenticationRecord クラスの属性の相互参照

ImqAuthenticationRecord C++ クラスの属性、データ構造体、フィールド、および呼び出しの相互参照。

表 882. 属性、データ構造、フィールド、および呼び出し

属性	データ構造体	フィールド	コール
接続名	MQAIR	AuthInfoConnName	MQCONN
パスワード	MQAIR	LDAPPassword	MQCONN
タイプ	MQAIR	AuthInfoType	MQCONN
ユーザー名	MQAIR	LDAPUserNamePtr	MQCONN
	MQAIR	LDAPUserNameOffset	MQCONN
	MQAIR	LDAPUserNameLength	MQCONN

ImqCache クラスの属性の相互参照

ImqCache C++ クラスの属性および呼び出しの相互参照。

表 883. 属性および呼び出し	
属性	コール
automatic buffer	MQGET
buffer length	MQGET
buffer pointer	MQGET、MQPUT
data length	MQGET
data offset	MQGET
data pointer	MQGET
message length	MQGET、MQPUT

ImqChannel の相互参照

ImqChannel C++ クラスの属性、データ構造、フィールド、および呼び出しの相互参照。

表 884. 属性、データ構造、フィールド、および呼び出し			
属性	データ構造体	フィールド	コール
batch heart-beat	MQCD	BatchHeartbeat	MQCONN
チャンネル名	MQCD	ChannelName	MQCONN
接続名	MQCD	ConnectionName	MQCONN
	MQCD	ShortConnectionName	MQCONN
ヘッダー圧縮	MQCD	HdrCompList	MQCONN
heart-beat interval	MQCD	HeartbeatInterval	MQCONN
keep alive interval	MQCD	KeepAliveInterval	MQCONN
local address	MQCD	LocalAddress	MQCONN
最大メッセージ長	MQCD	MaxMsgLength	MQCONN
メッセージ圧縮	MQCD	MsgCompList	MQCONN
モード名	MQCD	ModeName	MQCONN
パスワード	MQCD	パスワード	MQCONN
receive exit count	MQCD		MQCONN
receive exit names	MQCD	ReceiveExit	MQCONN
	MQCD	ReceiveExitsDefined	MQCONN
	MQCD	ReceiveExitPtr	MQCONN
receive user data	MQCD	ReceiveUserData	MQCONN
	MQCD	ReceiveUserDataPtr	MQCONN
セキュリティ出口名	MQCD	SecurityExit	MQCONN
security user data	MQCD	SecurityUserData	MQCONN
send exit count	MQCD		MQCONN
send exit names	MQCD	SendExit	MQCONN
	MQCD	SendExitsDefined	MQCONN

表 884. 属性、データ構造、フィールド、および呼び出し (続き)

属性	データ構造体	フィールド	コール
	MQCD	SendExitPtr	MQCONN
send user data	MQCD	SendUserData	MQCONN
	MQCD	SendUserDataPtr	MQCONN
SSL CipherSpec	MQCD	sslCipherSpecification	MQCONN
SSL client authentication type	MQCD	sslClientAuthentication	MQCONN
SSL ピア名	MQCD	sslPeerName	MQCONN
トランザクション・プログラム名	MQCD	TpName	MQCONN
トランスポート・タイプ	MQCD	TransportType	MQCONN
ユーザー ID	MQCD	UserIdentifier	MQCONN

ImqCICSBridgeHeader 相互参照

ImqCICSBridgeHeader C++ クラスの属性、データ構造体、およびフィールドの相互参照。

表 885. 属性、データ構造体、およびフィールド間のマッピング

属性	データ構造体	フィールド
bridge abend code	MQCIH	AbendCode
ADS descriptor	MQCIH	AdsDescriptor
attention identifier	MQCIH	AttentionId
authenticator	MQCIH	Authenticator
bridge completion code	MQCIH	BridgeCompletionCode
bridge error offset	MQCIH	ErrorOffset
bridge reason code	MQCIH	BridgeReason
bridge cancel code	MQCIH	CancelCode
conversational task	MQCIH	ConversationalTask
cursor position	MQCIH	CursorPosition
facility token	MQCIH	Facility
facility keep time	MQCIH	FacilityKeepTime
facility like	MQCIH	FacilityLike
function	MQCIH	Function
get wait interval	MQCIH	GetWaitInterval
link type	MQCIH	LinkType
next transaction identifier	MQCIH	NextTransactionId
output data length	MQCIH	OutputDataLength
reply-to format	MQCIH	ReplyToFormat
bridge return code	MQCIH	ReturnCode
start code	MQCIH	StartCode

表 885. 属性、データ構造体、およびフィールド間のマッピング (続き)		
属性	データ構造体	フィールド
task end status	MQCIH	TaskEndStatus
transaction identifier	MQCIH	TransactionId
uow control	MQCIH	UowControl
バージョン	MQCIH	バージョン

ImqDeadLetterHeader 相互参照

ImqDeadLetterHeader C++ クラスの属性、データ構造体、およびフィールドの相互参照。

表 886. 属性、データ構造体、およびフィールド間のマッピング		
属性	データ構造体	フィールド
dead-letter reason code	MQDLH	理由
destination queue manager name	MQDLH	DestQMgrName
destination queue name	MQDLH	DestQName
put application name	MQDLH	PutApplName
put application type	MQDLH	PutApplType
put date	MQDLH	PutDate
put time	MQDLH	PutTime

ImqError 相互参照

ImqError C++ クラスの属性および呼び出しの相互参照。

表 887. 属性および呼び出し	
属性	コール
完了コード	MQBACK、MQBEGIN、MQCLOSE、MQCMIT、MQCONN、MQCONNX、MQDISC、MQGET、MQINQ、MQOPEN、MQPUT、MQSET
理由コード	MQBACK、MQBEGIN、MQCLOSE、MQCMIT、MQCONN、MQCONNX、MQDISC、MQGET、MQINQ、MQOPEN、MQPUT、MQSET

ImqGetMessageOptions 相互参照

ImqGetMessageOptions C++ クラスの属性、データ構造体、およびフィールドの相互参照。

表 888. 属性、データ構造体、およびフィールド間のマッピング		
属性	データ構造体	フィールド
group status	MQGMO	GroupStatus
match options	MQGMO	MatchOptions
メッセージ・トークン (message token)	MQGMO	MessageToken
オプション	MQGMO	オプション
resolved queue name	MQGMO	ResolvedQName
returned length	MQGMO	ReturnedLength

表 888. 属性、データ構造体、およびフィールド間のマッピング (続き)		
属性	データ構造体	フィールド
セグメント化 (segmentation)	MQGMO	Segmentation
segment status	MQGMO	SegmentStatus
	MQGMO	Signal1
	MQGMO	Signal2
syncpoint participation	MQGMO	オプション
wait interval	MQGMO	WaitInterval

ImqHeader 相互参照

ImqHeader C++ クラスの属性、データ構造体、およびフィールドの相互参照。

表 889. 属性、データ構造体、およびフィールド間のマッピング		
属性	データ構造体	フィールド
character set	MQDLH、MQIIH	CodedCharSetId
encoding	MQDLH、MQIIH	Encoding
形式	MQDLH、MQIIH	Format
header flags	MQIIH、MQRMH	フラグ

ImqIMSBridgeHeader 相互参照

ImqAuthenticationRecord C++ クラスの属性、データ構造体、およびフィールドの相互参照。

表 890. 属性、データ構造体、およびフィールド間のマッピング		
属性	データ構造体	フィールド
authenticator	MQIIH	Authenticator
commit mode	MQIIH	CommitMode
logical terminal override	MQIIH	LTermOverride
message format services map name	MQIIH	MFSMapName
reply-to format	MQIIH	ReplyToFormat
security scope	MQIIH	SecurityScope
transaction instance id	MQIIH	TranInstanceId
transaction state	MQIIH	TranState

ImqItem 相互参照

ImqItem C++ クラスの属性および呼び出しの相互参照。

表 891. 属性および呼び出し	
属性	コール
structure id	MQGET

ImqMessage 相互参照

ImqMessage C++ クラスの属性、データ構造、フィールド、および呼び出しの相互参照。

表 892. 属性、データ構造、フィールド、および呼び出し			
属性	データ構造体	フィールド	コール
application ID data	MQMD	ApplIdentityData	
application origin data	MQMD	ApplOriginData	
backout count	MQMD	BackoutCount	
character set	MQMD	CodedCharSetId	
encoding	MQMD	Encoding	
expiry	MQMD	Expiry	
形式	MQMD	Format	
message flags	MQMD	MsgFlags	
メッセージ・タイプ	MQMD	MsgType	
offset	MQMD	オフセット	
original length	MQMD	OriginalLength	
persistence	MQMD	Persistence	
priority	MQMD	優先順位	
put application name	MQMD	PutApplName	
put application type	MQMD	PutApplType	
put date	MQMD	PutDate	
put time	MQMD	PutTime	
reply-to queue manager name	MQMD	ReplyToQMgr	
reply-to queue name	MQMD	ReplyToQ	
レポート	MQMD	レポート	
sequence number	MQMD	MsgSeqNumber	
total message length		DataLength	MQGET
ユーザー ID	MQMD	UserIdentifier	

ImqMessageTracker 相互参照

ImqMessageTracker C++ クラスの属性、データ構造体、およびフィールドの相互参照。

表 893. 属性、データ構造体、およびフィールド間のマッピング		
属性	データ構造体	フィールド
accounting token	MQMD	AccountingToken
correlation id	MQMD	CorrelId
feedback	MQMD	Feedback
group id	MQMD	GroupId
message id	MQMD	MsgId

ImqNamelist 相互参照

ImqNamelist C++ クラスの属性、照会、および呼び出しの相互参照。

表 894. 属性、照会、および呼び出し。		
属性	照会	コール
name count	MQIA_NAME_COUNT	MQINQ
namelist name	MQCA_NAMELIST_NAME	MQINQ

ImqObject 相互参照

ImqObject C++ クラスの属性、データ構造、フィールド、照会、および呼び出しの相互参照。

表 895. 属性、データ構造、フィールド、照会、および呼び出し。				
属性	データ構造体	フィールド	照会	コール
alteration date			MQCA_ALTERATION_DATE	MQINQ
alteration time			MQCA_ALTERATION_TIME	MQINQ
alternate user id	MQOD	AlternateUserId		
alternate security id				
close options				MQCLOSE
description			MQCA_Q_DESC、MQCA_Q_MGR_DESC、MQCA_PROCESS_DESC	MQINQ
名前	MQOD	ObjectName	MQCA_Q_MGR_NAME、MQCQ_Q_NAME、MQCA_PROCESS_NAME	MQINQ
open options				MQOPEN
open status				MQOPEN、MQCLOSE
queue manager identifier	queue manager identifier		MQCA_Q_MGR_IDENTIFIER	MQINQ

ImqProcess クラスの属性の相互参照

ImqAuthenticationRecord C++ クラスの属性、照会、および呼び出しの相互参照。

表 896. 属性、照会、および呼び出し。		
属性	照会	コール
application id	MQCA_APPL_ID	MQINQ
application type	MQIA_APPL_TYPE	MQINQ
environment data	MQCA_ENV_DATA	MQINQ
ユーザー・データ	MQCA_USER_DATA	MQINQ

ImqPutMessageOptions クラスの属性の相互参照

ImqAuthenticationRecord C++ クラスの属性、データ構造体、およびフィールドの相互参照。

表 897. 属性、データ構造体、およびフィールド間のマッピング		
属性	データ構造体	フィールド
context reference	MQPMO	Context
	MQPMO	InvalidDestCount
	MQPMO	KnownDestCount
オプション	MQPMO	オプション
record fields	MQPMO	PutMsgRecFields
resolved queue manager name	MQPMO	ResolvedQMgrName
resolved queue name	MQPMO	ResolvedQName
	MQPMO	タイムアウト
	MQPMO	UnknownDestCount
syncpoint participation	MQPMO	オプション

ImqQueue の相互参照

ImqQueue C++ クラスの属性、データ構造、フィールド、照会、および呼び出しの相互参照。

表 898. ImqQueue の相互参照				
属性	データ構造体	フィールド	照会	コール
backout requeue name			MQCA_BACKOUT_REQ_Q_NAME	MQINQ
バックアウトしきい値			MQIA_BACKOUT_THRESHOLD	MQINQ
base queue name			MQCA_BASE_Q_NAME	MQINQ
cluster name			MQCA_CLUSTER_NAME	MQINQ
cluster namelist name			MQCA_CLUSTER_NAMELIST	MQINQ
cluster workload rank			MQIA_CLWL_Q_RANK	MQINQ
cluster workload priority			MQIA_CLWL_Q_PRIORITY	MQINQ
cluster workload use queue			MQIA_CLWL_USEQ	MQINQ
creation date			MQCA_CREATION_DATE	MQINQ
creation time			MQCA_CREATION_TIME	MQINQ
current depth			MQIA_CURRENT_Q_DEPTH	MQINQ
default bind			MQIA_DEF_BIND	MQINQ
default input open option			MQIA_DEF_INPUT_OPEN_OPTION	MQINQ
default persistence			MQIA_DEF_PERSISTENCE	MQINQ
default priority			MQIA_DEF_PRIORITY	MQINQ

表 898. ImqQueue の相互参照 (続き)

属性	データ構造体	フィールド	照会	コール
definition type			MQIA_DEFINITION_TYPE	MQINQ
depth high event			MQIA_Q_DEPTH_HIGH_EVENT	MQINQ
depth high limit			MQIA_Q_DEPTH_HIGH_LIMIT	MQINQ
depth low event			MQIA_Q_DEPTH_LOW_EVENT	MQINQ
depth low limit			MQIA_Q_DEPTH_LOW_LIMIT	MQINQ
depth maximum event			MQIA_Q_DEPTH_MAX_LIMIT	MQINQ
distribution lists			MQIA_DIST_LISTS	MQINQ、MQSET
dynamic queue name	MQOD	DynamicQName		
harden get backout			MQIA_HARDEN_GET_BACKOUT	MQINQ
index type			MQIA_INDEX_TYPE	MQINQ
inhibit get			MQIA_INHIBIT_GET	MQINQ、MQSET
inhibit put			MQIA_INHIBIT_PUT	MQINQ、MQSET
initiation queue name			MQCA_INITIATION_Q_NAME	MQINQ
maximum depth			MQIA_MAX_Q_DEPTH	MQINQ
最大メッセージ長			MQIA_MAX_MSG_LENGTH	MQINQ
message delivery sequence			MQIA_MSG_DELIVERY_SEQUENCE	MQINQ
next distributed queue				
non persistent message class			MQIA_NPM_CLASS	MQINQ
インプット・カウントを開く			MQIA_OPEN_INPUT_COUNT	MQINQ
アウトプット・カウントを開く			MQIA_OPEN_OUTPUT_COUNT	MQINQ
previous distributed queue				
process name			MQCA_PROCESS_NAME	MQINQ
queue accounting			MQIA_ACCOUNTING_Q	MQINQ
キュー・マネージャー名	MQOD	ObjectQMgrName		
queue monitoring			MQIA_MONITORING_Q	MQINQ
queue statistics			MQIA_STATISTICS_Q	MQINQ

表 898. ImqQueue の相互参照 (続き)				
属性	データ構造体	フィールド	照会	コール
キュー・タイプ (queue type)			MQIA_Q_TYPE	MQINQ
リモート・キュー・マネージャー名			MQCA_REMOTE_Q_MGR_NAME	MQINQ
remote queue name			MQCA_REMOTE_Q_NAME	MQINQ
resolved queue manager name	MQOD	ResolvedQMgrName		
resolved queue name	MQOD	ResolvedQName		
retention interval			MQIA_RETENTION_INTERVAL	MQINQ
scope			MQIA_SCOPE	MQINQ
サービス間隔 (service interval)			MQIA_Q_SERVICE_INTERVAL	MQINQ
サービス・インターバル・イベント (service interval event)			MQIA_Q_SERVICE_INTERVAL_EVENT	MQINQ
shareability			MQIA_SHAREABILITY	MQINQ
ストレージ・クラス			MQCA_STORAGE_CLASS	MQINQ
伝送キュー名			MQCA_XMIT_Q_NAME	MQINQ
trigger control			MQIA_TRIGGER_CONTROL	MQINQ、MQSET
trigger data			MQCA_TRIGGER_DATA	MQINQ、MQSET
trigger depth			MQIA_TRIGGER_DEPTH	MQINQ、MQSET
trigger message priority			MQIA_TRIGGER_MSG_PRIORITY	MQINQ、MQSET
trigger type			MQIA_TRIGGER_TYPE	MQINQ、MQSET
usage			MQIA_USAGE	MQINQ

ImqQueueManager の相互参照

ImqQueueManager C++ クラスの属性、データ構造、フィールド、照会、および呼び出しの相互参照。

表 899. 属性、データ構造、フィールド、照会、および呼び出し。				
属性	データ構造体	フィールド	照会	コール
accounting connections override			MQIA_ACCOUNTING_CONN_OVERRIDE	MQINQ
accounting interval			MQIA_ACCOUNTING_INTERVAL	MQINQ

表 899. 属性、データ構造、フィールド、照会、および呼び出し。(続き)				
属性	データ 構造体	フィールド	照会	コール
activity recording			MQIA_ACTIVITY_RECORDING	MQINQ
adopt new mca check			MQIA_ADOPTNEWMCA_CHECK	MQINQ
adopt new mca type			MQIA_ADOPTNEWMCA_TYPE	MQINQ
authentication type	MQCSP	AuthenticationType		MQCONN
authority event			MQIA_AUTHORITY_EVENT	MQINQ
begin options	MQBO	オプション		MQBEGIN
bridge event			MQIA_BRIDGE_EVENT	MQINQ
channel auto definition			MQIA_CHANNEL_AUTO_DEF	MQINQ
channel auto definition event			MQIA_CHANNEL_AUTO_EVENT	MQIA
channel auto definition exit			MQIA_CHANNEL_AUTO_EXIT	MQIA
チャンネル・イベント (channel event)			MQIA_CHANNEL_EVENT	MQINQ
channel initiator adapters			MQIA_CHINIT_ADAPTERS	MQINQ
channel initiator control			MQIA_CHINIT_CONTROL	MQINQ
channel initiator dispatchers			MQIA_CHINIT_DISPATCHERS	MQINQ
channel initiator trace auto start			MQIA_CHINIT_TRACE_AUTO_START	MQINQ
channel initiator trace table size			MQIA_CHINIT_TRACE_TABLE_SIZE	MQINQ
チャンネル・モニター			MQIA_MONITORING_CHANNEL	MQINQ
channel reference	MQCD	ChannelType		MQCONN
チャンネル 統計			MQIA_STATISTICS_CHANNEL	MQINQ
character set			MQIA_CODED_CHAR_SET_ID	MQINQ
cluster sender monitoring			MQIA_MONITORING_AUTO_CLUSSDR	MQINQ
cluster sender statistics			MQIA_STATISTICS_AUTO_CLUSSDR	MQINQ
cluster workload data			MQCA_CLUSTER_WORKLOAD_DATA	MQINQ
クラスター・ワーク ロード出口			MQCA_CLUSTER_WORKLOAD_EXIT	MQINQ

表 899. 属性、データ構造、フィールド、照会、および呼び出し。(続き)				
属性	データ 構造体	フィールド	照会	コール
cluster workload length			MQIA_CLUSTER_WORKLOAD_LENGTH	MQINQ
cluster workload mru			MQIA_CLWL_MRU_CHANNELS	MQINQ
cluster workload use queue			MQIA_CLWL_USEQ	MQINQ
コマンド・イベント (command event)			MQIA_COMMAND_EVENT	MQINQ
command input queue name			MQCA_COMMAND_INPUT_Q_NAME	MQINQ
コマンド・レベル			MQIA_COMMAND_LEVEL	MQINQ
command server control			MQIA_CMD_SERVER_CONTROL	MQINQ
connect options	MQCNO	オプション		MQCONN、 MQCONNXX
connection id	MQCNO	ConnectionId		MQCONNXX
connection status				MQCONN、 MQCONNXX、 MQDISC
connection tag	MQCD	ConnTag		MQCONNXX
cryptographic hardware	MQSCO	CryptoHardware		MQCONNXX
dead-letter queue name			MQCA_DEAD_LETTER_Q_NAME	MQINQ
default transmission queue name			MQCA_DEF_XMIT_Q_NAME	MQINQ
distribution lists			MQIA_DIST_LISTS	MQINQ
dns group			MQCA_DNS_GROUP	MQINQ
dns wlm			MQIA_DNS_WLM	MQINQ
first authentication record	MQSCO	AuthInfoRecOffset		MQCONNXX
	MQSCO	AuthInfoRecPtr		MQCONNXX
inhibit event			MQIA_INHIBIT_EVENT	MQINQ
ip address version			MQIA_IP_ADDRESS_VERSION	MQINQ
鍵リポジトリ (key repository)	MQSCO	KeyRepository		MQCONNXX
key reset count	MQSCO	KeyResetCount		MQCONNXX
listener timer			MQIA_LISTENER_TIMER	MQINQ

表 899. 属性、データ構造、フィールド、照会、および呼び出し。(続き)

属性	データ 構造体	フィールド	照会	コール
local event			MQIA_LOCAL_EVENT	MQINQ
logger event			MQIA_LOGGER_EVENT	MQINQ
lu group name			MQCA_LU_GROUP_NAME	MQINQ
lu name			MQCA_LU_NAME	MQINQ
lu62 arm suffix			MQCA_LU62_ARM_SUFFIX	MQINQ
lu62 channels			MQIA_LU62_CHANNELS	MQINQ
maximum active channels			MQIA_ACTIVE_CHANNELS	MQINQ
maximum channels			MQIA_MAX_CHANNELS	MQINQ
maximum handles			MQIA_MAX_HANDLES	MQINQ
最大メッセージ長			MQIA_MAX_MSG_LENGTH	MQINQ
maximum priority			MQIA_MAX_PRIORITY	MQINQ
maximum uncommitted messages			MQIA_MAX_UNCOMMITTED_MSGS	MQINQ
mqi accounting			MQIA_ACCOUNTING_MQI	MQINQ
mqi statistics			MQIA_STATISTICS_MQI	MQINQ
outbound port maximum			MQIA_OUTBOUND_PORT_MAX	MQINQ
outbound port minimum			MQIA_OUTBOUND_PORT_MIN	MQINQ
パスワード	MQCSP	CSPPasswordPtr		MQCONN
	MQCSP	CSPPasswordOffset		MQCONN
	MQCSP	CSPPasswordLength		MQCONN
パフォーマンス・イ ベント (performance event)			MQIA_PERFORMANCE_EVENT	MQINQ
platform			MQIA_PLATFORM	MQINQ
queue accounting			MQIA_ACCOUNTING_Q	MQINQ
queue monitoring			MQIA_MONITORING_Q	MQINQ
queue statistics			MQIA_STATISTICS_Q	MQINQ
receive timeout			MQIA_RECEIVE_TIMEOUT	MQINQ
receive timeout minimum			MQIA_RECEIVE_TIMEOUT_MIN	MQINQ

表 899. 属性、データ構造、フィールド、照会、および呼び出し。(続き)				
属性	データ構造体	フィールド	照会	コール
receive timeout type			MQIA_RECEIVE_TIMEOUT_TYPE	MQINQ
remote event			MQIA_REMOTE_EVENT	MQINQ
repository name			MQCA_REPOSITORY_NAME	MQINQ
repository namelist			MQCA_REPOSITORY_NAMELIST	MQINQ
shared queue queue manager name			MQIA_SHARED_Q_Q_MGR_NAME	MQINQ
ssl event			MQIA_SSL_EVENT	MQINQ
ssl fips			MQIA_SSL_FIPS_REQUIRED	MQINQ
ssl key reset count			MQIA_SSL_RESET_COUNT	MQINQ
start-stop event			MQIA_START_STOP_EVENT	MQINQ
statistics interval			MQIA_STATISTICS_INTERVAL	MQINQ
syncpoint availability			MQIA_SYNCPOINT	MQINQ
tcp channels			MQIA_TCP_CHANNELS	MQINQ
tcp keep alive			MQIA_TCP_KEEP_ALIVE	MQINQ
tcp name			MQCA_TCP_NAME	MQINQ
tcp stack type			MQIA_TCP_STACK_TYPE	MQINQ
trace route recording			MQIA_TRACE_ROUTE_RECORDING	MQINQ
trigger interval			MQIA_TRIGGER_INTERVAL	MQINQ
ユーザー ID	MQCSP	CSPUserIdPtr		MQCONN
	MQCSP	CSPUserIdOffset		MQCONN
	MQCSP	CSPUserIdLength		MQCONN

ImqReferenceHeader の相互参照

ImqAuthenticationRecord C++ クラスの属性、データ構造体、およびフィールドの相互参照。

表 900. 属性、データ構造体、およびフィールド間のマッピング		
属性	データ構造体	フィールド
destination environment	MQRMH	DestEnvLength, DestEnvOffset
destination name	MQRMH	DestNameLength, DestNameOffset
instance id	MQRMH	ObjectInstanceId
logical length	MQRMH	DataLogicalLength
logical offset	MQRMH	DataLogicalOffset

表 900. 属性、データ構造体、およびフィールド間のマッピング (続き)		
属性	データ構造体	フィールド
logical offset 2	MQRMH	DataLogicalOffset2
reference type	MQRMH	ObjectType
source environment	MQRMH	SrcEnvLength, SrcEnvOffset
source name	MQRMH	SrcNameLength, SrcNameOffset

ImqTrigger クラスの属性の相互参照

ImqAuthenticationRecord C++ クラスの属性、データ構造体、およびフィールドの相互参照。

表 901. 属性、データ構造体、およびフィールド間のマッピング		
属性	データ構造体	フィールド
application id	MQTM	ApplId
application type	MQTM	ApplType
environment data	MQTM	EnvData
process name	MQTM	ProcessName
キュー名	MQTM	QName
trigger data	MQTM	TriggerData
ユーザー・データ	MQTM	UserData

ImqWorkHeader クラスの属性の相互参照

ImqAuthenticationRecord C++ クラスの属性、データ構造体、およびフィールドの相互参照。

表 902. 属性、データ構造体、およびフィールド間のマッピング		
属性	データ構造体	フィールド
メッセージ・トークン (message token)	MQWIIH	MessageToken
サービス名	MQWIIH	ServiceName
service step	MQWIIH	ServiceStep

ImqAuthenticationRecord C++ クラス

このクラスは、カスタム TLS クライアント接続で、ImqQueueManager::connect メソッドを実行するときに使用するための認証情報レコード (MQAIR) をカプセル化します。

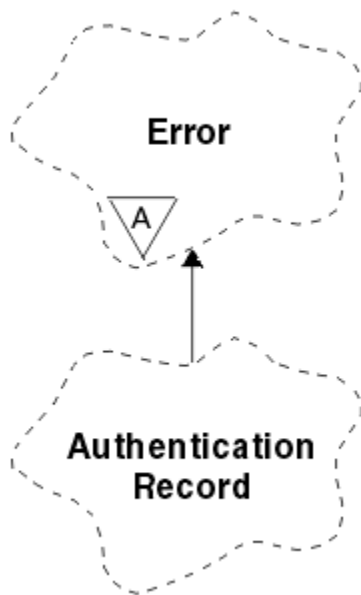


図 13. *ImqAuthenticationRecord* クラス

詳細については、`ImqQueueManager::connect` メソッドの説明を参照してください。このクラスは、z/OS プラットフォームでは使用できません。

- [1807 ページの『オブジェクトの属性』](#)
- [1807 ページの『コンストラクター』](#)
- [1808 ページの『オブジェクト・メソッド \(共有\)』](#)
- [1808 ページの『オブジェクト・メソッド \(保護\)』](#)

オブジェクトの属性

接続名

LDAP CRL サーバーへの接続の名前。IP アドレスまたは DNS 名で、オプションでポート番号を括弧に入れてこの後ろに続けることができます。

connection reference

必要に応じて (ローカル) キュー・マネージャーに接続できるようにする `ImqQueueManager` オブジェクトに対する参照。初期値はゼロです。これを、指定のキューのキュー・マネージャー (場合によっては、リモートにある) を識別する `queue manager name` と混同しないでください。

next authentication record

特定の順序はなく、このオブジェクトと同じ **connection reference** を持つ、このクラスの次のオブジェクト。初期値はゼロです。

パスワード

LDAP CRL サーバーへの接続の認証用に指定するパスワード。

previous authentication record

特定の順序はなく、このオブジェクトと同じ **connection reference** を持つ、このクラスの直前のオブジェクト。初期値はゼロです。

タイプ

レコードに含まれる認証情報のタイプ。

ユーザー名

LDAP CRL サーバーに対する許可のために指定するユーザー ID。

コンストラクター

`ImqAuthenticationRecord()`;

デフォルトのコンストラクター。

オブジェクト・メソッド (共有)

void operator = (const ImqAuthenticationRecord & *air*);

インスタンス・データを *air* からコピーし、既存のインスタンス・データを置き換えます。

const ImqString & connectionName () const ;

connection name を返します。

void setConnectionName (const ImqString & *name*);

connection name を設定します。

void setConnectionName (const char * *name* = 0);

connection name を設定します。

ImqQueueManager * connectionReference () const ;

connection reference を返します。

void setConnectionReference (ImqQueueManager & *manager*);

connection reference を設定します。

void setConnectionReference (ImqQueueManager * *manager* = 0);

connection reference を設定します。

void copyOut (MQAIR * *pAir*);

インスタンス・データを *pAir* にコピーし、既存のインスタンス・データを置き換えます。 このことには、従属ストレージの割り振りが関係することがあります。

void clear (MQAIR * *pAir*);

構造体を消去し、*pAir* によって参照される従属ストレージを解放します。

ImqAuthenticationRecord * nextAuthenticationRecord () const ;

next authentication record を返します。

const ImqString & password () const ;

password を返します。

void setPassword (const ImqString & *password*);

password を設定します。

void setPassword (const char * *password* = 0);

password を設定します。

ImqAuthenticationRecord * previousAuthenticationRecord () const ;

previous authentication record を返します。

MQLONG type () const ;

type を返します。

void setType (const MQLONG *type*);

type を設定します。

const ImqString & userName () const ;

user name を返します。

void setUserName (const ImqString & *name*);

user name を設定します。

void setUserName (const char * *name* = 0);

user name を設定します。

オブジェクト・メソッド (保護)

void setNextAuthenticationRecord (ImqAuthenticationRecord * *pAir* = 0);

next authentication record を設定します。

重要: この関数は、認証レコード・リストを損傷しないことが確かな場合に限り使用してください。

void setPreviousAuthenticationRecord (ImqAuthenticationRecord * *pAir* = 0);

previous authentication record を設定します。

重要: この関数は、認証レコード・リストを損傷しないことが確かな場合に限り使用してください。

ImqBinary C++ クラス

このクラスは、ImqMessage の **accounting token**、**correlation id**、および **message id** 値に使用できる 2 進バイト・アレイをカプセル化します。これにより割り当て、コピー、および比較が容易になります。

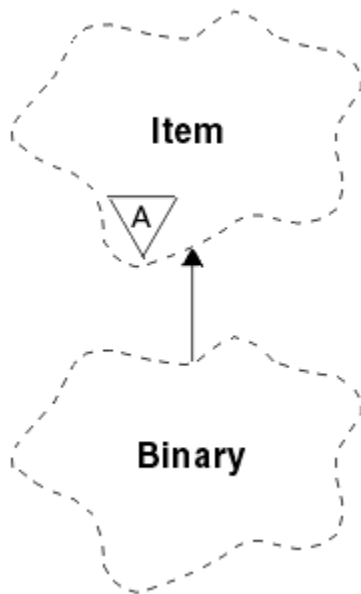


図 14. *ImqBinary* クラス

- [1809 ページの『オブジェクトの属性』](#)
- [1809 ページの『コンストラクター』](#)
- [1810 ページの『多重定義された ImqItem メソッド』](#)
- [1810 ページの『オブジェクト・メソッド \(共有\)』](#)
- [1810 ページの『オブジェクト・メソッド \(保護\)』](#)
- [1810 ページの『理由コード』](#)

オブジェクトの属性

データ

2 進データのバイト・アレイ。初期値はヌルです。

data length

バイト数。初期値はゼロです。

data pointer

data の最初のバイトのアドレス。初期値はゼロです。

コンストラクター

ImqBinary();

デフォルトのコンストラクター。

ImqBinary(const ImqBinary & binary);

コピー・コンストラクター。

ImqBinary(const void * data, const size_t length);

data から **length** 分のバイトをコピーします。

多重定義された ImqItem メソッド

virtual ImqBoolean copyOut (ImqMessage & msg);

data をメッセージ・バッファにコピーし、既存の内容があれば置き換えます。また、*msg* の形式を MQFMT_NONE に設定します。

詳細については、ImqItem クラス・メソッドの説明を参照してください。

virtual ImqBoolean pasteIn (ImqMessage & msg);

メッセージ・バッファから残っているデータを転送し、既存の **data** を置き換えることにより **data** を設定します。

正常に実行されるためには、ImqMessage の **format** が MQFMT_NONE でなければなりません。

詳細については、ImqItem クラス・メソッドの説明を参照してください。

オブジェクト・メソッド (共有)

void operator = (const ImqBinary & binary);

binary からバイトをコピーします。

ImqBoolean operator == (const ImqBinary & binary);

このオブジェクトを *binary* と比較します。等しくない場合は FALSE を返し、それ以外の場合は TRUE を返します。2つのオブジェクトが同じ **data length** をもっており、バイト数が一致する場合、2つのオブジェクトは等しいといえます。

ImqBoolean copyOut(void * buffer, const size_t length, const char pad = 0);

最大 *length* 分のバイトを **data pointer** から *buffer* へコピーします。**data length** が不十分であると、*buffer* 内の残りのスペースには *pad* 分のバイトが埋め込まれます。*length* がゼロの場合は、*buffer* もゼロで構いません。*length* が負であってはなりません。正常に終了した場合は TRUE を返します。

size_t dataLength() const ;

data length を返します。

ImqBoolean setDataLength(const size_t length);

data length を設定します。このメソッドの結果として **data length** が変更された場合、オブジェクト内のデータの初期設定は解除されます。正常に終了した場合は TRUE を返します。

void * dataPointer() const ;

data pointer を返します。

ImqBoolean isNull() const ;

data length がゼロである場合、あるいは **data** バイトがすべてゼロである場合は TRUE を返します。それ以外の場合は FALSE を返します。

ImqBoolean set(const void * buffer, const size_t length);

buffer から *length* 分のバイトをコピーします。正常に終了した場合は TRUE を返します。

オブジェクト・メソッド (保護)

void clear();

data length を減らしてゼロにします。

理由コード

- MQRC_NO_BUFFER
- MQRC_STORAGE_NOT_AVAILABLE
- MQRC_INCONSISTENT_FORMAT

ImqCache C++ クラス

このクラスは、データをメモリーに保持したり、マーシャルするのに使用します。

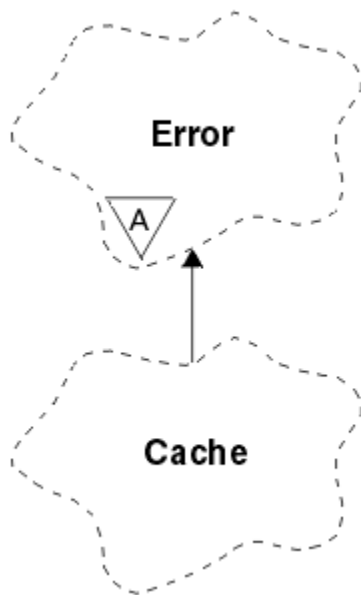


図 15. *ImqCache* クラス

このクラスは、データをメモリに保持したり、マーシャルするのに使用します。ユーザーが固定サイズのメモリのバッファを指定することも、システムによる自動的かつ柔軟なメモリ設定に任せることもできます。このクラスは、1792 ページの『[ImqCache クラスの属性の相互参照](#)』にリストされている MQI 呼び出しと関連があります。

- [1811 ページの『オブジェクトの属性』](#)
- [1812 ページの『コンストラクター』](#)
- [1812 ページの『オブジェクト・メソッド \(共有\)』](#)
- [1813 ページの『理由コード』](#)

オブジェクトの属性

automatic buffer

バッファ・メモリがシステムによって自動的に管理される (TRUE) のか、ユーザーによって提供される (FALSE) のかを示します。初期設定は TRUE です。

この属性は直接設定されることはありません。 **useEmptyBuffer** または **useFullBuffer** のどちらかのメソッドを使用して、間接的に設定されます。

ユーザー・ストレージが提供されている場合、この属性は FALSE になり、バッファ・メモリの増設はできません。さらに、バッファ・オーバーフロー・エラーが発生する可能性があります。バッファのアドレスと長さは変わりません。

ユーザー・ストレージが提供されていない場合、この属性は TRUE になり、バッファ・メモリは任意の量のメッセージ・データを収容できる大きさになるまで決まった容量ずつ増設できます。ただし、バッファが大きくなると、そのバッファのアドレスが変わる可能性があります。このため、バッファ・ポインターとデータ・ポインターを使用する際には、十分に注意してください。

buffer length

バッファ内のメモリのバイト数。初期値はゼロです。

buffer pointer

バッファ・メモリのアドレス。初期値はヌルです。

data length

data pointer より後のバイト数。 **message length** に等しいか、それより小さい数値である必要があります。初期値はゼロです。

data offset

data pointer より前のバイト数。 **message length** に等しいか、それより小さい数値である必要があります。 初期値はゼロです。

data pointer

次に読み取りまたは書き込みが行われるバッファー部分のアドレス。 初期値はヌルです。

message length

バッファー内の有効なデータのバイト数。 初期値はゼロです。

コンストラクター

ImqCache();

デフォルトのコンストラクター。

ImqCache(const ImqCache & cache);

コピー・コンストラクター。

オブジェクト・メソッド (共有)

void operator = (const ImqCache & cache);

message length 分のバイトのデータを *cache* オブジェクトから当該オブジェクトへコピーします。
automatic buffer が FALSE である場合は、**buffer length** が、コピーされたデータを収容できるだけの大きさでなければなりません。

ImqBoolean automaticBuffer() const ;

automatic buffer の値を返します。

size_t bufferLength() const ;

buffer length を返します。

char * bufferPointer() const ;

buffer pointer を返します。

void clearMessage();

message length および **data offset** をゼロに設定します。

size_t dataLength() const ;

data length を返します。

size_t dataOffset() const ;

data offset を返します。

ImqBoolean setDataOffset(const size_t offset);

data offset を設定します。 メッセージ長は、データ・オフセットより小さくならないように、必要に応じて増加されます。 このメソッドは、正常に終了した場合には TRUE を返します。

char * dataPointer() const ;

data pointer のコピーを返します。

size_t messageLength() const ;

message length を返します。

ImqBoolean setMessageLength(const size_t length);

message length を設定します。 **message length** が **buffer length** 以下であることが必要であれば、**buffer length** を増やします。 **message length** 以上ではないことが必要であれば、**data offset** を減らします。 正常に終了した場合は TRUE を返します。

ImqBoolean moreBytes(const size_t bytes-required);

data pointer からバッファーの終わりまでの間でさらに *bytes-required* 分のバイトが (書き込みに) 使用できるようにします。 正常に終了した場合は TRUE を返します。

automatic buffer が TRUE である場合は、必要に応じて、さらに多くのメモリーが取得されます。
FALSE の場合は、**buffer length** が十分な大きさでなければなりません。

ImqBoolean read (const size_t length, char * & external-buffer);

length 分のバイトを、**data pointer** 位置で始まるバッファから *external-buffer* へコピーします。データがコピーされたあとで、**data offset** は *length* だけ増やされます。このメソッドは、正常に終了した場合には TRUE を返します。

ImqBoolean resizeBuffer(const size_t length);

automatic buffer が TRUE である場合は、**buffer length** を変えます。これは、バッファ・メモリを再度割り振ることによって行います。既存のバッファから最大 **message length** 分のバイトのデータが新しいバッファにコピーされます。コピーされる最大数は *length* バイトです。**buffer pointer** は変更されます。**message length** と **data offset** は、新しいバッファの範囲内で可能な限り接近して保存されます。正常に終了した場合は TRUE を返し、**automatic buffer** が FALSE である場合は FALSE を返します。

注：システム・リソースに問題があると、このメソッドが失敗し、理由コード MQRC_STORAGE_NOT_AVAILABLE が返される場合があります。

ImqBoolean useEmptyBuffer(const char * external-buffer, const size_t length);

空のユーザー・バッファを識別し、**buffer pointer** が *external-buffer* を示すように設定し、**buffer length** を *length* に、また **message length** をゼロにそれぞれ設定します。**clearMessage** を実行します。バッファがデータで完全に満たされている場合は、代わりに **useFullBuffer** メソッドを使用してください。バッファの一部にデータが入っている場合は、**setMessageLength** メソッドを使用して、正しい量を示してください。このメソッドは、正常に終了した場合には TRUE を返します。

前述のとおり、このメソッドを使用して、固定量のメモリ (*external-buffer* が非ヌルで、*length* が非ゼロ) を識別することができます。この場合、**automatic buffer** は FALSE に設定されます。もしくは、システムによる柔軟なメモリ管理 (*external-buffer* がヌルで、しかも *length* がゼロ) に戻すこともできます。この場合、**automatic buffer** は TRUE に設定されます。

ImqBoolean useFullBuffer(const char * externalBuffer, const size_t length);

useEmptyBuffer については、**message length** が *length* に設定されている場合を除きます。正常に終了した場合は TRUE を返します。

ImqBoolean write(const size_t length, const char * external-buffer);

length 分のバイトを、*external-buffer* から、**data pointer** 位置で始まるバッファへコピーします。データがコピーされた後、**データ・オフセット**は長さによって増加され、必要に応じて**メッセージ長**が増加し、それが新しい**データ・オフセット**値よりも小さい値にならないようにします。このメソッドは、正常に終了した場合には TRUE を返します。

automatic buffer が TRUE である場合は、十分な量のメモリが保証されます。そうでない場合は、最終的な **data offset** が **buffer length** を超えてはなりません。

理由コード

- MQRC_BUFFER_NOT_AUTOMATIC
- MQRC_DATA_TRUNCATED
- MQRC_INSUFFICIENT_BUFFER
- MQRC_INSUFFICIENT_DATA
- MQRC_NULL_POINTER
- MQRC_STORAGE_NOT_AVAILABLE
- MQRC_ZERO_LENGTH

ImqChannel C++ クラス

このクラスは、カスタム・クライアント接続で、Manager::connect メソッドを実行するときに使用するためのチャンネル定義 (MQCD) をカプセル化します。

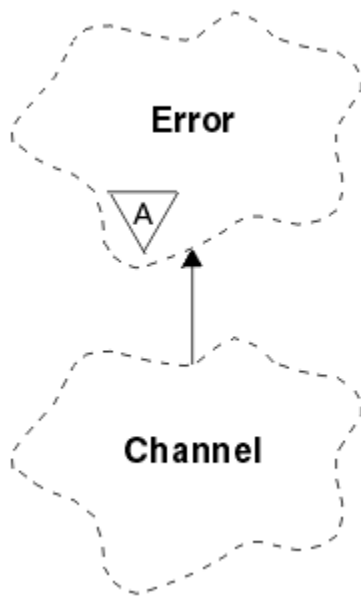


図 16. *ImqChannel* クラス

詳細については、`Manager::connect` メソッドの説明と、[サンプル・プログラム HELLO WORLD \(imqwrlld.cpp\)](#) を参照してください。

リストにあるすべてのメソッドをすべてのプラットフォームに適用できるわけではありません。詳しくは、[DEFINE CHANNEL](#) および [ALTER CHANNEL](#) コマンドの説明を参照してください。

ImqChannel クラスは、z/OS ではサポートされていません。

- [1814 ページの『オブジェクトの属性』](#)
- [1815 ページの『コンストラクター』](#)
- [1815 ページの『オブジェクト・メソッド \(共有\)』](#)
- [1819 ページの『理由コード』](#)

オブジェクトの属性

batch heart-beat

リモート・チャンネルがアクティブとなっているチェック間の時間 (ミリ秒)。初期値は 0 です。

チャンネル名

チャンネルの名前。初期値はヌルです。

接続名

接続の名前。例えば、ホスト・コンピューターの IP アドレス。初期値はヌルです。

ヘッダー圧縮

チャンネルでサポートされるヘッダー・データ圧縮技法のリスト。初期値はすべて、`MQCOMPRESS_NOT_AVAILABLE` に設定されます。

heart-beat interval

接続が作動を続けているかどうかを検査する頻度 (秒数)。初期値は 300 です。

keep alive interval

チャンネルのキープアライブ時間を指定する、通信スタックに渡される時間 (秒)。初期値は `MQKAI_AUTO` です。

local address

チャンネル用のローカル通信アドレス。

最大メッセージ長

単一の通信において、チャンネルがサポートするメッセージの最大長。初期値は 4 194 304 です。

メッセージ圧縮

チャンネルでサポートされるメッセージ・データ圧縮技法のリスト。初期値はすべて、MQCOMPRESS_NOT_AVAILABLE に設定されます。

モード名

モードの名前。初期値はヌルです。

パスワード

接続の認証用に指定するパスワード。初期値はヌルです。

receive exit count

受信出口の数。初期値はゼロです。この属性は読み取り専用です。

receive exit names

受信出口の名前。

receive user data

受信出口に関連したデータ。

セキュリティー出口名

接続のサーバー側で起動されるセキュリティー出口の名前。初期値はヌルです。

security user data

セキュリティー出口に渡されるデータ。初期値はヌルです。

send exit count

送信出口の数。初期値はゼロです。この属性は読み取り専用です。

send exit names

送信出口の名前。

send user data

送信出口に関連したデータ。

SSL CipherSpec

TLS とともに使用するための CipherSpec。

SSL client authentication type

TLS とともに使用するためのクライアント認証タイプ。

SSL ピア名

TLS とともに使用するためのピア名。

トランザクション・プログラム名

トランザクション・プログラムの名前。初期値はヌルです。

トランスポート・タイプ

接続のトランスポートのタイプ。初期値は MQXPT_LU62 です。

ユーザー ID

許可のために指定するユーザー ID。初期値はヌルです。

コンストラクター

ImqChannel();

デフォルトのコンストラクター。

ImqChannel(const ImqChannel & channel);

コピー・コンストラクター。

オブジェクト・メソッド (共有)

void operator = (const ImqChannel & channel);

インスタンス・データを *channel* からコピーし、既存の任意のインスタンス・データを置き換えます。

MQLONG batchHeartBeat() const ;

batch heart-beat を返します。

ImqBoolean setBatchHeartBeat(const MQLONG heartbeat = 0L);

batch heart-beat を設定します。このメソッドは、正常に終了した場合には TRUE を返します。

ImqString channelName() const ;
channel name を返します。

ImqBoolean setChannelName(const char * name = 0);
channel name を設定します。このメソッドは、正常に終了した場合には TRUE を返します。

ImqString connectionName() const ;
connection name を返します。

ImqBoolean setConnectionName(const char * name = 0);
connection name を設定します。このメソッドは、正常に終了した場合には TRUE を返します。

size_t headerCompressionCount() const ;
サポートされているヘッダー・データ圧縮技法カウントを返します。

ImqBoolean headerCompression(const size_t count, MQLONG compress []) const ;
compress の、サポートされているヘッダー・データ圧縮技法のコピーを返します。このメソッドは、正常に終了した場合には TRUE を返します。

ImqBoolean setHeaderCompression(const size_t count, const MQLONG compress []);
compress にサポートされているヘッダー・データ圧縮技法を設定します。
count にサポートされているヘッダー・データ圧縮技法カウントを設定します。
このメソッドは、正常に終了した場合には TRUE を返します。

MQLONG heartBeatInterval() const ;
heart-beat interval を返します。

ImqBoolean setHeartBeatInterval(const MQLONG interval = 300L);
heart-beat interval を設定します。このメソッドは、正常に終了した場合には TRUE を返します。

MQLONG keepAliveInterval() const ;
keep alive interval を返します。

ImqBoolean setKeepAliveInterval(const MQLONG interval = MQKAI_AUTO);
keep alive interval を設定します。このメソッドは、正常に終了した場合には TRUE を返します。

ImqString localAddress() const ;
local address を返します。

ImqBoolean setLocalAddress (const char * address = 0);
local address を設定します。このメソッドは、正常に終了した場合には TRUE を返します。

MQLONG maximumMessageLength() const ;
maximum message length を返します。

ImqBoolean setMaximumMessageLength(const MQLONG length = 4194304L);
maximum message length を設定します。このメソッドは、正常に終了した場合には TRUE を返します。

size_t messageCompressionCount() const ;
サポートされているメッセージ・データ圧縮技法カウントを返します。

ImqBoolean messageCompression(const size_t count, MQLONG compress []) const ;
compress の、サポートされているメッセージ・データ圧縮技法のコピーを返します。このメソッドは、正常に終了した場合には TRUE を返します。

ImqBoolean setMessageCompression(const size_t count, const MQLONG compress []);
compress にサポートされているメッセージ・データ圧縮技法を設定します。
count にサポートされているメッセージ・データ圧縮技法カウントを設定します。
このメソッドは、正常に終了した場合には TRUE を返します。

ImqString modeName() const ;
mode name を返します。

ImqBoolean setModeName(const char * name = 0);
mode name を設定します。このメソッドは、正常に終了した場合には TRUE を返します。

ImqString password() const ;

password を返します。

ImqBoolean setPassword(const char * password = 0);

password を設定します。このメソッドは、正常に終了した場合には TRUE を返します。

size_t receiveExitCount() const ;

receive exit count を返します。

ImqString receiveExitName();

receive exit names があれば、その最初のものを返します。**receive exit count** がゼロの場合は、空ストリングを返します。

ImqBoolean receiveExitNames(const size_t count, ImqString * names []);

names に **receive exit names** のコピーを返します。**receive exit count** を上回る **names** をすべてヌル・ストリングに設定します。このメソッドは、正常に終了した場合には TRUE を返します。

ImqBoolean setReceiveExitName(const char * name = 0);

receive exit names を単一の **name** に設定します。**name** は、ブランクまたはヌルにすることができます。**receive exit count** を 1 またはゼロに設定します。**receive user data** を消去します。このメソッドは、正常に終了した場合には TRUE を返します。

ImqBoolean setReceiveExitNames(const size_t count, const char * names []);

receive exit names を **names** に設定します。個々の **names** 値は、ブランクまたはヌルにできません。**receive exit count** を **count** に設定します。**receive user data** を消去します。このメソッドは、正常に終了した場合には TRUE を返します。

ImqBoolean setReceiveExitNames(const size_t count, const ImqString * names []);

receive exit names を **names** に設定します。個々の **names** 値は、ブランクまたはヌルにできません。**receive exit count** を **count** に設定します。**receive user data** を消去します。このメソッドは、正常に終了した場合には TRUE を返します。

ImqString receiveUserData();

receive user data 項目があれば、その最初の項目を返します。**receive exit count** がゼロの場合は、空ストリングを返します。

ImqBoolean receiveUserData(const size_t count, ImqString * data []);

receive user data 項目のコピーを **data** に返します。**receive exit count** を上回る **data** をすべてヌル・ストリングに設定します。このメソッドは、正常に終了した場合には TRUE を返します。

ImqBoolean setReceiveUserData(const char * data = 0);

receive user data を、単一の項目 **data** に設定します。データがヌルでない場合、**receive exit count** は少なくとも 1 でなければなりません。このメソッドは、正常に終了した場合には TRUE を返します。

ImqBoolean setReceiveUserData(const size_t count, const char * data []);

receive user data を **data** に設定します。**count** は、**receive exit count** より大きい値であってはなりません。このメソッドは、正常に終了した場合には TRUE を返します。

ImqBoolean setReceiveUserData(const size_t count, const ImqString * data []);

receive user data を **data** に設定します。**count** は、**receive exit count** より大きい値であってはなりません。このメソッドは、正常に終了した場合には TRUE を返します。

ImqString securityExitName() const ;

security exit name を返します。

ImqBoolean setSecurityExitName(const char * name = 0);

security exit name を設定します。このメソッドは、正常に終了した場合には TRUE を返します。

ImqString securityUserData() const ;

security user data を返します。

ImqBoolean setSecurityUserData(const char * data = 0);

security user data を設定します。このメソッドは、正常に終了した場合には TRUE を返します。

size_t sendExitCount() const ;

send exit count を返します。

ImqString sendExitName();

send exit names があれば、その最初のものを返します。**send exit count** がゼロの場合は、空ストリングを返します。

ImqBoolean sendExitNames(const size_t count, ImqString * names []);
names に **send exit names** のコピーを返します。 **send exit count** を上回る *names* をすべてヌル・ストリングに設定します。このメソッドは、正常に終了した場合には TRUE を返します。

ImqBoolean setSendExitName(const char * name = 0);
send exit names を単一の *name* に設定します。 *name* は、ブランクまたはヌルにすることができます。 **send exit count** を 1 またはゼロに設定します。 **send user data** を消去します。このメソッドは、正常に終了した場合には TRUE を返します

ImqBoolean setSendExitNames(const size_t count, const char * names []);
send exit names を *names* に設定します。個々の *names* 値は、ブランクまたはヌルにすることができません。 **send exit count** を *count* に設定します。 **send user data** を消去します。このメソッドは、正常に終了した場合には TRUE を返します。

ImqBoolean setSendExitNames(const size_t count, const ImqString * names []);
send exit names を *names* に設定します。個々の *names* 値は、ブランクまたはヌルにすることができません。 **send exit count** を *count* に設定します。 **send user data** を消去します。このメソッドは、正常に終了した場合には TRUE を返します。

ImqString sendUserData();
send user data 項目がある場合には、その項目の最初の項目を返します。 **send exit count** がゼロの場合は、空ストリングを返します。

ImqBoolean sendUserData(const size_t count, ImqString * data []);
data で **send user data** 項目のコピーを返します。 **send exit count** を上回る *data* をすべてヌル・ストリングに設定します。このメソッドは、正常に終了した場合には TRUE を返します。

ImqBoolean setSendUserData(const char * data = 0);
send user data を、単一の項目 *data* に設定します。データがヌルでない場合、 **send exit count** は少なくとも 1 でなければなりません。このメソッドは、正常に終了した場合には TRUE を返します。

ImqBoolean setSendUserData(const size_t count, const char * data []);
send user data を *data* に設定します。 *count* は、 **send exit count** より大きい値であってはなりません。このメソッドは、正常に終了した場合には TRUE を返します。

ImqBoolean setSendUserData(const size_t count, const ImqString * data []);
send user data を *data* に設定します。 *count* は、 **send exit count** より大きい値であってはなりません。このメソッドは、正常に終了した場合には TRUE を返します。

ImqString sslCipherSpecification() const ;
TLS 暗号仕様を返します。

ImqBoolean setSslCipherSpecification(const char * name = 0);
TLS 暗号仕様を設定します。このメソッドは、正常に終了した場合には TRUE を返します。

MQLONG sslClientAuthentication() const ;
TLS クライアント認証タイプを返します。

ImqBoolean setSslClientAuthentication(const MQLONG auth = MQSCA_REQUIRED);
TLS クライアント認証タイプを設定します。このメソッドは、正常に終了した場合には TRUE を返します。

ImqString sslPeerName() const ;
TLS ピア名を返します。

ImqBoolean setSslPeerName(const char * name = 0);
TLS ピア名を設定します。このメソッドは、正常に終了した場合には TRUE を返します。

ImqString transactionProgramName() const ;
transaction program name を返します。

ImqBoolean setTransactionProgramName(const char * name = 0);
transaction program name を設定します。このメソッドは、正常に終了した場合には TRUE を返します。

MQLONG transportType() const ;
transport type を返します。

ImqBoolean setTransportType(const MQLONG type = MQXPT_LU62);
transport type を設定します。このメソッドは、正常に終了した場合には TRUE を返します。

ImqString userId() const ;

user id を返します。

ImqBoolean setUserId(const char * id = 0);

user id を設定します。このメソッドは、正常に終了した場合には TRUE を返します。

理由コード

- MQRC_DATA_LENGTH_ERROR
- MQRC_ITEM_COUNT_ERROR
- MQRC_NULL_POINTER
- MQRC_SOURCE_BUFFER_ERROR

ImqCICSBridgeHeader C++ クラス

このクラスは、MQCIH データ構造体の特定の機能をカプセル化します。

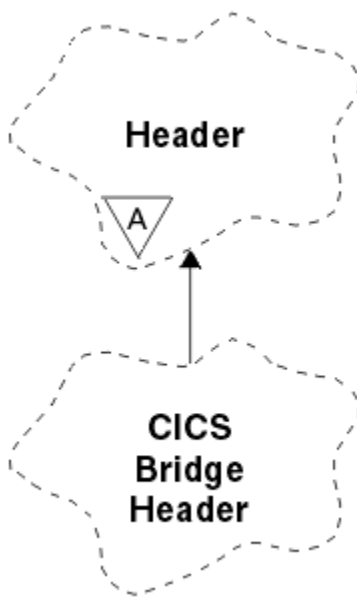


図 17. *ImqCICSBridgeHeader* クラス

このクラスのオブジェクトは、IBM MQ for z/OS を介して CICS bridge にメッセージを送信するアプリケーションによって使用されます。

- [1819 ページの『オブジェクトの属性』](#)
- [1822 ページの『コンストラクター』](#)
- [1822 ページの『多重定義された ImqItem メソッド』](#)
- [1822 ページの『オブジェクト・メソッド \(共有\)』](#)
- [1824 ページの『オブジェクト・データ \(保護\)』](#)
- [1824 ページの『理由コード』](#)
- [1825 ページの『戻りコード』](#)

オブジェクトの属性

ADS descriptor

ADS 記述子の送受信を行います。このフィールドの値の設定には、MQCADSD_NONE を使用します。初期値は MQCADSD_NONE です。以下の値が取得可能な値です。

- MQCADSD_NONE

- MQCADSD_SEND
- MQCADSD_RECV
- MQCADSD_MSGFORMAT

attention identifier

AID キー。このフィールドの長さは MQ_ATTENTION_ID_LENGTH でなければなりません。

authenticator

RACF パスワードまたはパスチケット。初期値はブランクになります。値の長さは MQ_AUTHENTICATOR_LENGTH です。

bridge abend code

ブリッジ異常終了コード。長さは MQ_ABEND_CODE_LENGTH です。初期値は 4 つのブランク文字になります。このフィールドに返される値は、戻りコードによって異なります。詳しくは、[1825 ページの表 903](#) を参照してください。

bridge cancel code

ブリッジ異常終了トランザクション・コード。このフィールドは予約フィールドなので、値は必ずブランクになります。値の長さは MQ_CANCEL_CODE_LENGTH でなければなりません。

bridge completion code

完了コード。このフィールドの値は、IBM MQ 完了コードまたは CICS EIBRESP 値のどちらかになります。このフィールドの初期値は MQCC_OK です。このフィールドに返される値は、戻りコードによって異なります。詳細については、[1825 ページの表 903](#) を参照してください。

bridge error offset

ブリッジ・エラー・オフセット。初期値はゼロです。この属性は読み取り専用です。

bridge reason code

理由コード。このフィールドの値は、IBM MQ 理由コードまたは CICS EIBRESP2 値のどちらかになります。このフィールドの初期値は MQRC_NONE です。このフィールドに返される値は、戻りコードによって異なります。詳細については、[1825 ページの表 903](#) を参照してください。

bridge return code

CICS bridge からの戻りコード。初期値は MQCRC_OK です。

conversational task

タスクが会話型タスクになっているかを示します。初期値は MQCCT_NO です。以下の値が取得可能な値です。

- MQCCT_YES
- MQCCT_NO

cursor position

カーソル位置。初期値はゼロです。

facility keep time

CICS bridge のファシリティ・リリース時間。

facility like

端末でエミュレートされた属性。このフィールドの長さは MQ_FACILITY_LIKE_LENGTH でなければなりません。

facility token

BVT トークン値。このフィールドの長さは MQ_FACILITY_LENGTH でなければなりません。初期値は MQCFAC_NONE です。

function

関数。このフィールドの値は、IBM MQ 呼び出し名または CICS EIBFN 関数のどちらかになります。このフィールドの初期値は MQCFUNC_NONE です。値の長さは MQ_FUNCTION_LENGTH です。このフィールドに返される値は、戻りコードによって異なります。詳細については、[1825 ページの表 903](#) を参照してください。

function に IBM MQ 呼び出し名が含まれている場合に、以下の値を追加で使用できます。

- MQCFUNC_MQCONN
- MQCFUNC_MQGET

- MQCFUNC_MQINQ
- MQCFUNC_NONE
- MQCFUNC_MQOPEN
- MQCFUNC_PUT
- MQCFUNC_MQPUT1

get wait interval

CICS bridge・タスクによって MQGET 呼び出しが発行されるまでの待機間隔。初期値は MQCGWI_DEFAULT です。このフィールドが適用されるのは、**uow control** の値が MQCUOWC_FIRST の場合だけです。以下の値が取得可能な値です。

- MQCGWI_DEFAULT
- MQWI_UNLIMITED

link type

リンクのタイプ。初期値は MQCLT_PROGRAM です。以下の値が取得可能な値です。

- MQCLT_PROGRAM
- MQCLT_TRANSACTION

next transaction identifier

次に関連付けるトランザクションの ID。このフィールドの長さは MQ_TRANSACTION_ID_LENGTH でなければなりません。

output data length

COMMAREA データの長さ。初期値は MQCODL_AS_INPUT です。

reply-to format

応答メッセージの形式名。初期値は MQFMT_NONE です。値の長さは MQ_FORMAT_LENGTH です。

start code

トランザクション開始コード。このフィールドの長さは MQ_START_CODE_LENGTH でなければなりません。初期値は MQCSC_NONE です。以下の値が取得可能な値です。

- MQCSC_START
- MQCSC_STARTDATA
- MQCSC_TERMINPUT
- MQCSC_NONE

task end status

タスク終了状況。初期値は MQCTES_NOSYNC です。以下の値が取得可能な値です。

- MQCTES_COMMIT
- MQCTES_BACKOUT
- MQCTES_ENDTASK
- MQCTES_NOSYNC

transaction identifier

関連付けるトランザクションの ID。初期値は必ずブランクになります。値の長さは MQ_TRANSACTION_ID_LENGTH でなければなりません。このフィールドが適用されるのは、**uow control** の値が MQCUOWC_FIRST または MQCUOWC_ONLY の場合だけです。

UOW control

UOW (作業単位) 制御。初期値は MQCUOWC_ONLY です。以下の値が取得可能な値です。

- MQCUOWC_FIRST
- MQCUOWC_MIDDLE
- MQCUOWC_LAST
- MQCUOWC_ONLY
- MQCUOWC_COMMIT

- MQCUOWC_BACKOUT
- MQCUOWC_CONTINUE

バージョン

MQCIH バージョン番号。初期値は MQCIH_VERSION_2 です。このほかにサポートされている値は MQCIH_VERSION_1 だけです。

コンストラクター

ImqCICSBridgeHeader() ;

デフォルトのコンストラクター。

ImqCICSBridgeHeader(const ImqCICSBridgeHeader & header);

コピー・コンストラクター。

多重定義された ImqItem メソッド

virtual ImqBoolean copyOut(ImqMessage & msg);

MQCIH データ構造体をメッセージ・バッファの始めに挿入し、既存のメッセージ・データを後ろにずらし、メッセージ形式を MQFMT_CICS に設定します。

詳細については、親クラス・メソッドの説明を参照してください。

virtual ImqBoolean pasteIn(ImqMessage & msg);

メッセージ・バッファから MQCIH データ構造体を読み取ります。読み取りが正しく実行されるようにするには、msg オブジェクトのエンコードを MQENC_NATIVE に設定する必要があります。

MQGMO_CONVERT を使用してメッセージを MQENC_NATIVE に取り出します。検索が正しく実行されるようにするには、ImqMessage の形式を MQFMT_CICS に設定する必要があります。

詳細については、親クラス・メソッドの説明を参照してください。

オブジェクト・メソッド (共有)

void operator = (const ImqCICSBridgeHeader & header);

インスタンス・データを header からコピーし、既存のインスタンス・データを置き換えます。

MQLONG ADSDescriptor() const;

ADS descriptor のコピーを返します。

void setADSDescriptor(const MQLONG descriptor = MQCADSD_NONE);

ADS descriptor を設定します。

ImqString attentionIdentifier() const;

attention identifier のコピーを、長さ MQ_ATTENTION_ID_LENGTH まで後書きブランクを埋め込んで返します。

void setAttentionIdentifier(const char * data = 0);

attention identifier を、長さ MQ_ATTENTION_ID_LENGTH まで後書きブランクを埋め込んで設定します。data が設定されていないと、attention identifier を初期値に戻します。

ImqString authenticator() const;

authenticator のコピーを、長さ MQ_AUTHENTICATOR_LENGTH まで後書きブランクを埋め込んで返します。

void setAuthenticator(const char * data = 0);

authenticator を、長さ MQ_AUTHENTICATOR_LENGTH まで後書きブランクを埋め込んで返します。data が設定されていないと、authenticator を初期値に戻します。

ImqString bridgeAbendCode() const;

bridge abend code のコピーを、長さ MQ_ABEND_CODE_LENGTH まで後書きブランクを埋め込んで返します。

ImqString bridgeCancelCode() const;

bridge cancel code のコピーを、長さ MQ_CANCEL_CODE_LENGTH まで後書きブランクを埋め込んで返します。

void setBridgeCancelCode(const char * data = 0);
bridge cancel code を、長さ MQ_CANCEL_CODE_LENGTH まで後書きブランクを埋め込んで設定します。**data** が設定されていないと、**bridge cancel code** を初期値に戻します。

MQLONG bridgeCompletionCode() const;
bridge completion code のコピーを返します。

MQLONG bridgeErrorOffset() const ;
bridge error offset のコピーを返します。

MQLONG bridgeReasonCode() const;
bridge reason code のコピーを返します。

MQLONG bridgeReturnCode() const;
bridge return code を返します。

MQLONG conversationalTask() const;
conversational task のコピーを返します。

void setConversationalTask(const MQLONG task = MQCCT_NO);
conversational task を設定します。

MQLONG cursorPosition() const ;
cursor position のコピーを返します。

void setCursorPosition(const MQLONG position = 0);
cursor position を設定します。

MQLONG facilityKeepTime() const;
facility keep time のコピーを返します。

void setFacilityKeepTime(const MQLONG time = 0);
facility keep time を設定します。

ImqString facilityLike() const;
facility like のコピーを、長さ MQ_FACILITY_LIKE_LENGTH まで後書きブランクを埋め込んで返します。

void setFacilityLike(const char * name = 0);
facility like を、長さ MQ_FACILITY_LIKE_LENGTH まで後書きブランクを埋め込んで設定します。**name** が設定されていないと、**facility like** を初期値に戻します。

ImqBinary facilityToken() const;
facility token のコピーを返します。

ImqBoolean setFacilityToken(const ImqBinary & token);
facility token を設定します。**token** の **data length** は、ゼロまたは、MQ_FACILITY_LENGTH のいずれかでなければなりません。正常に終了した場合は TRUE を返します。

void setFacilityToken(const MQBYTE8 token = 0);
facility token を設定します。**token** はゼロであっても構いません。これは MQCFAC_NONE を指定するのと同じです。**token** がゼロ以外の場合には、MQ_FACILITY_LENGTH バイトの 2 進データをアドレッシングするものでなければなりません。MQCFAC_NONE などの事前定義値を使用する場合は、シグニチャーが一致することを確認するためにキャストを行う必要がある場合があります。例えば、(MQBYTE *)MQCFAC_NONE などです。

ImqString function() const;
function のコピーを、長さ MQ_FUNCTION_LENGTH まで後書きブランクを埋め込んで返します。

MQLONG getWaitInterval() const;
get wait interval のコピーを返します。

void setGetWaitInterval(const MQLONG interval = MQCGWI_DEFA
get wait interval を設定します。

MQLONG linkType() const;
link type のコピーを返します。

void setLinkType(const MQLONG type = MQCLT_PROGRAM);
link type を設定します。

ImqString nextTransactionIdentifier() const ;

next transaction identifier のコピーを、長さ MQ_TRANSACTION_ID_LENGTH まで後書きブランクを埋め込んで返します。

MQLONG outputDataLength() const;

output data length のコピーを返します。

void setOutputDataLength(const MQLONG length = MQCODL_AS_INPUT);

output data length を設定します。

ImqString replyToFormat() const;

reply-to format 名のコピーを、長さ MQ_FORMAT_LENGTH まで後書きブランクを埋め込んで返します。

void setReplyToFormat(const char * name = 0);

reply-to format を、長さ MQ_FORMAT_LENGTH まで後書きブランクを埋め込んで設定します。 *name* が設定されていないと、**reply-to format** を初期値に戻します。

ImqString startCode() const;

start のコピーを、長さ MQ_START_CODE_LENGTH まで後書きブランクを埋め込んで返します。

void setStartCode(const char * data = 0);

start code を、長さ MQ_START_CODE_LENGTH まで後書きブランクを埋め込んで設定します。 *data* が設定されていないと、**start code** を初期値に戻します。

MQLONG taskEndStatus() const;

task end status のコピーを返します。

ImqString transactionIdentifier() const;

transaction identifier データのコピーを、長さ MQ_TRANSACTION_ID_LENGTH まで後書きブランクを埋め込んで返します。

void setTransactionIdentifier(const char * data = 0);

transaction identifier を、長さ MQ_TRANSACTION_ID_LENGTH まで後書きブランクを埋め込んで設定します。 *data* が設定されていないと、**transaction identifier** を初期値に戻します。

MQLONG UOWControl() const;

UOW control のコピーを返します。

void setUOWControl(const MQLONG control = MQCUOWC_ONLY);

UOW control を設定します。

MQLONG version() const;

version 番号を返します。

ImqBoolean setVersion(const MQLONG version = MQCIH_VERSION_2);

version 番号を設定します。正常に終了した場合は TRUE を返します。

オブジェクト・データ (保護)

MQLONG olVersion

opcih 用に割り振られたストレージに指定できる最大 MQCIH バージョン番号。

PMQCIH opcih

MQCIH データ構造体のアドレス。割り振られたストレージ容量は *olVersion* で示されます。

理由コード

- MQRC_BINARY_DATA_LENGTH_ERROR
- MQRC_WRONG_VERSION

戻りコード

表 903. <i>ImqCICSBridgeHeader</i> クラスの戻りコード				
リターン・コード	Function	CompCode	理由	異常終了コード
MQCRC_OK				
MQCRC_BRIDGE_ERROR			MQFB_CICS	
MQCRC_MQ_API_ERROR	IBM MQ 呼び出し名	IBM MQ CompCode	IBM MQ Reason	
MQCRC_BRIDGE_TIMEOUT	IBM MQ 呼び出し名	IBM MQ CompCode	IBM MQ Reason	
MQCRC_CICS_EXEC_ERROR	CICS EIBFN (EIBFN)	CICS EIBRESP (EIBRESP)	CICS EIBRESP2	
MQCRC_SECURITY_ERROR	CICS EIBFN (EIBFN)	CICS EIBRESP (EIBRESP)	CICS EIBRESP2	
MQCRC_PROGRAM_NOT_AVAILABLE	CICS EIBFN (EIBFN)	CICS EIBRESP (EIBRESP)	CICS EIBRESP2	
MQCRC_TRANSID_NOT_AVAILABLE	CICS EIBFN (EIBFN)	CICS EIBRESP (EIBRESP)	CICS EIBRESP2	
MQCRC_BRIDGE_ABEND				CICS 異常終了コード
MQCRC_APPLICATION_ABEND				CICS 異常終了コード

ImqDeadLetterHeader C++ クラス

このクラスは MQDLH データ構造体の機能をカプセル化します。

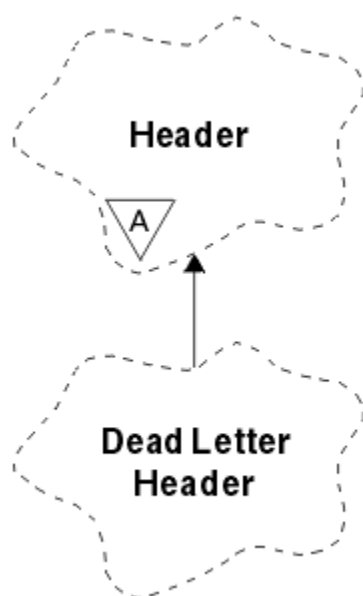


図 18. *ImqDeadLetterHeader* クラス

このクラスのオブジェクトは、一般的に、処理不能なメッセージを検出するアプリケーションによって使用されます。送達不能ヘッダーとメッセージ内容とで構成される新規メッセージは送達不能キューに入れられ、そのメッセージは破棄されます。

- [1826 ページの『オブジェクトの属性』](#)
- [1826 ページの『コンストラクター』](#)
- [1826 ページの『多重定義された ImqItem メソッド』](#)
- [1827 ページの『オブジェクト・メソッド \(共有\)』](#)
- [1827 ページの『オブジェクト・データ \(保護\)』](#)
- [1827 ページの『理由コード』](#)

オブジェクトの属性

dead-letter reason code

メッセージが送達不能キューに届いた理由。初期値は MQRC_NONE です。

destination queue manager name

元の宛先キュー・マネージャーの名前。この名前は、長さ MQ_Q_MGR_NAME_LENGTH のstring です。初期値はヌルです。

destination queue name

元の宛先キューの名前。この名前は、長さ MQ_Q_NAME_LENGTH のstring です。初期値はヌルです。

put application name

メッセージを送達不能キューに書き込んだアプリケーションの名前。この名前は、長さ MQ_PUT_APPL_NAME_LENGTH のstring です。初期値はヌルです。

put application type

メッセージを送達不能キューに書き込んだアプリケーションのタイプ。初期値はゼロです。

put date

メッセージが送達不能キューに書き込まれた日付。この日付は、長さ MQ_PUT_DATE_LENGTH のstring です。初期値はヌル・string です。

put time

メッセージが送達不能キューに書き込まれた時刻。この時刻は、長さ MQ_PUT_TIME_LENGTH のstring です。初期値はヌル・string です。

コンストラクター

ImqDeadLetterHeader();

デフォルトのコンストラクター。

ImqDeadLetterHeader(const ImqDeadLetterHeader & header);

コピー・コンストラクター。

多重定義された ImqItem メソッド

virtual ImqBoolean copyOut (ImqMessage & msg);

MQDLH データ構造体をメッセージ・バッファの始めに挿入し、既存のメッセージ・データを後ろにずらします。また、msg の形式を MQFMT_DEAD_LETTER_HEADER に設定します。

詳細については、[1833 ページの『ImqHeader C++ クラス』](#)の ImqHeader クラス・メソッドの説明を参照してください。

virtual ImqBoolean pasteIn (ImqMessage & msg);

メッセージ・バッファから MQDLH データ構造体を読み取ります。

正常に実行されるためには、ImqMessage の format が MQFMT_DEAD_LETTER_HEADER でなければなりません。

詳細については、1833 ページの『ImqHeader C++ クラス』の ImqHeader クラス・メソッドの説明を参照してください。

オブジェクト・メソッド (共有)

void operator = (const ImqDeadLetterHeader & header);

インスタンス・データを header からコピーし、既存のインスタンス・データを置き換えます。

MQLONG deadLetterReasonCode () const ;

dead-letter reason code を返します。

void setDeadLetterReasonCode (const MQLONG reason);

dead-letter reason code を設定します。

ImqString destinationQueueManagerName () const ;

末尾ブランクを除去した destination queue manager name を返します。

void setDestinationQueueManagerName (const char * name);

destination queue manager name を設定します。 MQ_Q_MGR_NAME_LENGTH (48 文字) より長いデータを切り捨てます。

ImqString destinationQueueName () const ;

末尾ブランクを除去した destination queue name のコピーを返します。

void setDestinationQueueName (const char * name);

destination queue name を設定します。 MQ_Q_NAME_LENGTH (48 文字) より長いデータを切り捨てます。

ImqString putApplicationName () const ;

末尾ブランクを除去した put application name のコピーを返します。

void setPutApplicationName (const char * name = 0);

put application name を設定します。 MQ_PUT_APPL_NAME_LENGTH (28 文字) より長いデータを切り捨てます。

MQLONG putApplicationType () const ;

put application type を返します。

void setPutApplicationType (const MQLONG type = MQAT_NO_CONTEXT);

put application type を設定します。

ImqString putDate () const ;

末尾ブランクを除去した put date のコピーを返します。

void setPutDate (const char * date = 0);

put date を設定します。 MQ_PUT_DATE_LENGTH (8 文字) より長いデータを切り捨てます。

ImqString putTime () const ;

末尾ブランクを除去した put time のコピーを返します。

void setPutTime (const char * time = 0);

put time を設定します。 MQ_PUT_TIME_LENGTH (8 文字) より長いデータを切り捨てます。

オブジェクト・データ (保護)

MQDLH omqdlh

MQDLH データ構造体。

理由コード

- MQRC_INCONSISTENT_FORMAT
- MQRC_STRUC_ID_ERROR
- MQRC_ENCODING_ERROR

ImqDistributionList C++ クラス

このクラスは、複数の宛先に1つまたは複数のメッセージを送信するために1つまたは複数のキューを参照する動的配布リストをカプセル化します。

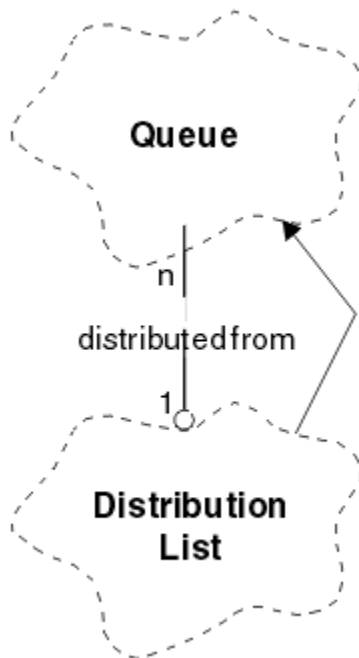


図 19. ImqDistributionList クラス

- [1828 ページの『オブジェクトの属性』](#)
- [1828 ページの『コンストラクター』](#)
- [1828 ページの『オブジェクト・メソッド \(共有\)』](#)
- [1829 ページの『オブジェクト・メソッド \(保護\)』](#)

オブジェクトの属性

first distributed queue

クラスの1つ以上のオブジェクトのうち、特定の順序ではなく、**distribution list reference**がこのオブジェクトをアドレッシングする順序で最初のもの。

このオブジェクトは最初から存在しているわけではありません。ImqDistributionList を正常にオープンするには、必ずこのオブジェクトが1つ以上存在していなければなりません。

注: ImqDistributionList オブジェクトがオープンされているときに、それを参照するオープン・オブジェクトは、自動的にクローズされます。

コンストラクター

ImqDistributionList();

デフォルトのコンストラクター。

ImqDistributionList(const ImqDistributionList & list);

コピー・コンストラクター。

オブジェクト・メソッド (共有)

void operator = (const ImqDistributionList & list);

このオブジェクトを参照するオブジェクトはすべて、コピーの前に参照の対象から外されます。このメソッドが呼び出された後にこのオブジェクトを参照するオブジェクトはありません。

*** firstDistributedQueue() const ;**
first distributed queue を返します。

オブジェクト・メソッド (保護)

void setFirstDistributedQueue(* queue = 0);
first distributed queue を設定します。

ImqError C++ クラス

この抽象クラスは、あるオブジェクトに関連付けられたエラーに関する情報を提供します。

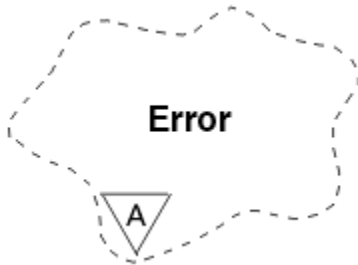


図 20. ImqError クラス

- [1829 ページの『オブジェクトの属性』](#)
- [1829 ページの『コンストラクター』](#)
- [1829 ページの『オブジェクト・メソッド \(共有\)』](#)
- [1830 ページの『オブジェクト・メソッド \(保護\)』](#)
- [1830 ページの『理由コード』](#)

オブジェクトの属性

完了コード

最新の完了コード。初期値はゼロです。以下の値が取得可能な値です。

- MQCC_OK
- MQCC_WARNING
- MQCC_FAILED

理由コード

最新の理由コード。初期値はゼロです。

コンストラクター

ImqError();
デフォルトのコンストラクター。

ImqError(const ImqError & error);
コピー・コンストラクター。

オブジェクト・メソッド (共有)

void operator = (const ImqError & error);
インスタンス・データを error からコピーし、既存のインスタンス・データを置き換えます。

void clearErrorCodes();
completion code と reason code を両方ともゼロに設定します。

MQLONG completionCode() const ;
completion code を返します。

MQLONG reasonCode() const ;
reason code を返します。

オブジェクト・メソッド (保護)

ImqBoolean checkReadPointer(const void * *pointer*, const size_t *length*);
pointer と length の組み合わせが読み取り専用アクセスに有効であるか調べます。正常に実行されると、TRUE を返します。

ImqBoolean checkWritePointer(const void * *pointer*, const size_t *length*);
pointer と length の組み合わせが読み取り及び書き込みアクセスに有効であるか調べます。正常に実行されると、TRUE を返します。

void setCompletionCode(const MQLONG code = 0);
completion code を設定します。

void setReasonCode(const MQLONG code = 0);
reason code を設定します。

理由コード

- MQRC_BUFFER_ERROR

ImqGetMessageOptions C++ クラス

このクラスは MQGMO データ構造体をカプセル化します。

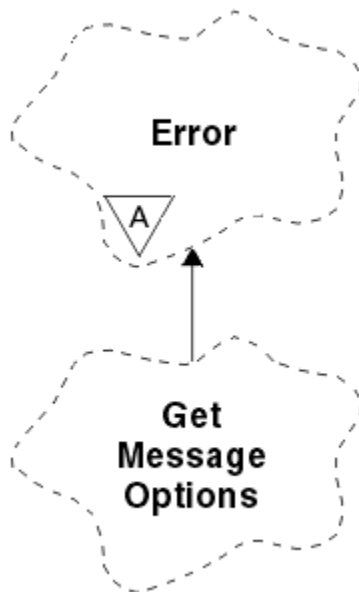


図 21. ImqGetMessageOptions クラス

- [1831 ページの『オブジェクトの属性』](#)
- [1832 ページの『コンストラクター』](#)
- [1832 ページの『オブジェクト・メソッド \(共有\)』](#)
- [1833 ページの『オブジェクト・メソッド \(保護\)』](#)
- [1833 ページの『オブジェクト・データ \(保護\)』](#)
- [1833 ページの『理由コード』](#)

オブジェクトの属性

group status

メッセージのグループについてのメッセージの状況。初期値は MQGS_NOT_IN_GROUP です。以下の値が取得可能な値です。

- MQGS_MSG_IN_GROUP
- MQGS_LAST_MSG_IN_GROUP

match options

着信メッセージを選択するためのオプション。初期値は MQMO_MATCH_MSG_ID | MQMO_MATCH_CORREL_ID です。以下の値が取得可能な値です。

- MQMO_GROUP_ID
- MQMO_MATCH_MSG_SEQ_NUMBER
- MQMO_MATCH_OFFSET
- MQMO_MSG_TOKEN
- MQMO_NONE

メッセージ・トークン (message token)

メッセージ・トークン。長さ MQ_MSG_TOKEN_LENGTH の 2 進値 (MQBYTE16)。初期値は MQMTOK_NONE です。

オプション

メッセージに適用可能なオプション。初期値は MQGMO_NO_WAIT です。以下の値が取得可能な値です。

- MQGMO_WAIT
- MQGMO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT
- MQGMO_NO_SYNCPOINT
- MQGMO_MARK_SKIP_BACKOUT
- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_NEXT
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_MSG_UNDER_CURSOR
- MQGMO_LOCK
- MQGMO_UNLOCK
- MQGMO_ACCEPT_TRUNCATED_MSG
- MQGMO_SET_SIGNAL
- MQGMO_FAIL_IF QUIESCING
- MQGMO_CONVERT
- MQGMO_LOGICAL_ORDER
- MQGMO_COMPLETE_MSG
- MQGMO_ALL_MSGS_AVAILABLE
- MQGMO_ALL_SEGMENTS_AVAILABLE
- MQGMO_NONE

resolved queue name

解決済みのキューの名前。この属性は読み取り専用です。名前は、長さが 48 文字以下でなければならず、その長さになるまでヌルが埋め込まれます。初期値はヌル・ストリングです。

returned length

戻された長さ。初期値は MQRL_UNDEFINED です。この属性は読み取り専用です。

セグメント化 (segmentation)

メッセージを分割する機能です。初期値は MQSEG_INHIBITED です。取得可能な値は MQSEG_ALLOWED です。

segment status

メッセージの分割状況です。初期値は MQSS_NOT_A_SEGMENT です。以下の値が取得可能な値です。

- MQSS_SEGMENT
- MQSS_LAST_SEGMENT

syncpoint participation

同期点制御を受けてメッセージが検索される場合は TRUE です。

wait interval

適切なメッセージがまだ使用できるようになっていない場合に、クラスの get メソッドが、適切なメッセージの着信を待機している間に一時停止している時間の長さ。初期値はゼロで、これは、無期限の待機を有効にします。取得可能な値は MQWI_UNLIMITED です。options に MQGMO_WAIT が組み込まれていない場合は、この属性は無視されます。

コンストラクター

ImqGetMessageOptions();

デフォルトのコンストラクター。

ImqGetMessageOptions(const ImqGetMessageOptions & gmo);

コピー・コンストラクター。

オブジェクト・メソッド (共有)

void operator = (const ImqGetMessageOptions & gmo);

インスタンス・データを gmo からコピーし、既存のインスタンス・データを置き換えます。

MQCHAR groupStatus () const ;

group status を返します。

void setGroupStatus (const MQCHAR status);

group status を設定します。

MQLONG matchOptions () const ;

match options を返します。

void setMatchOptions (const MQLONG options);

match options を設定します。

ImqBinary messageToken () const;

message token を返します。

ImqBoolean setMessageToken(const ImqBinary & token);

message token を設定します。token の data length は、ゼロまたは、MQ_MSG_TOKEN_LENGTH のいずれかでなければなりません。このメソッドは、正常に終了した場合には TRUE を返します。

void setMessageToken(const MQBYTE16 token = 0);

message token を設定します。token はゼロであっても構いません。これは MQMTOK_NONE を指定するのと同じです。token が非ゼロの場合には、MQ_MSG_TOKEN_LENGTH バイトの 2 進データをアドレッシングするものでなければなりません。

MQMTOK_NONE などの事前定義値を使用する場合は、確実にシグニチャーが一致するようにキャスト (例えば、(MQBYTE *)MQMTOK_NONE) を作成する必要がないことがあります。

MQLONG options () const ;

options を返します。

void setOptions (const MQLONG options);

syncpoint participation 値を組み込んで、options を設定します。

ImqString resolvedQueueName () const ;
resolved queue name のコピーを返します。

MQLONG returnedLength() const;
returned length を返します。

MQCHAR segmentation () const ;
segmentation を返します。

void setSegmentation (const MQCHAR value);
segmentation を設定します。

MQCHAR segmentStatus () const ;
segment status を返します。

void setSegmentStatus (const MQCHAR status);
segment status を設定します。

ImqBoolean syncPointParticipation () const ;
syncpoint participation 値を返します。options に MQGMO_SYNCPOINT または MQGMO_SYNCPOINT_IF_PERSISTENT のいずれかが組み込まれている場合は、TRUE を返します。

void setSyncPointParticipation (const ImqBoolean sync);
syncpoint participation 値を設定します。sync の指定値により options の変更方法が異なります。TRUE の場合、MQGMO_SYNCPOINT を設定し、MQGMO_NO_SYNCPOINT および MQGMO_SYNCPOINT_IF_PERSISTENT を除外します。sync が FALSE の場合、MQGMO_NO_SYNCPOINT は含まれ、MQGMO_SYNCPOINT および MQGMO_SYNCPOINT_IF_PERSISTENT はいずれも除外されるように options を変更します。

MQLONG waitInterval () const ;
wait interval を返します。

void setWaitInterval (const MQLONG interval);
wait interval を設定します。

オブジェクト・メソッド (保護)

static void setVersionSupported (const MQLONG);
MQGMO version を設定します。デフォルトは、MQGMO_VERSION_3 です。

オブジェクト・データ (保護)

MQGMO omqgmo
MQGMO バージョン 2 のデータ構造体。必ず MQGMO_VERSION_2 専用にサポートされている MQGMO フィールドにアクセスしてください。

PMQGMO opgmo
MQGMO データ構造体のアドレス。このアドレスのバージョン番号は、olVersion に示されています。MQGMO フィールドが存在していることを確認するには、それらのフィールドにアクセスする前に、必ずバージョン番号を調べてください。

MQLONG olVersion
opgmo でアドレッシングされた MQGMO データ構造体のバージョン番号。

理由コード

- MQRC_BINARY_DATA_LENGTH_ERROR

ImqHeader C++ クラス

この抽象クラスは MQDLH データ構造体の共通機能をカプセル化します。

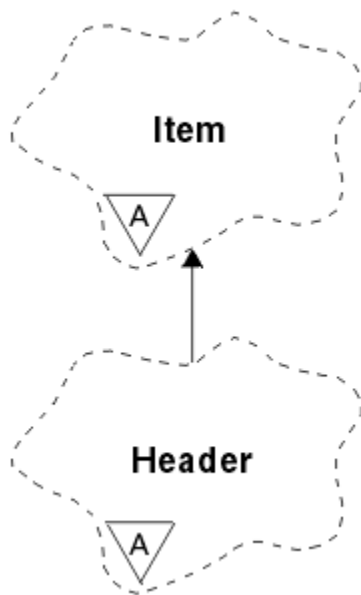


図 22. *ImqHeader* クラス

- [1834 ページの『オブジェクトの属性』](#)
- [1834 ページの『コンストラクター』](#)
- [1834 ページの『オブジェクト・メソッド \(共有\)』](#)

オブジェクトの属性

character set

元のコード化文字セット ID。初期値は MQCCSI_Q_MGR です。

encoding

元のエンコード。初期値は MQENC_NATIVE です。

形式

元の形式。初期値は MQFMT_NONE です。

header flags

初期値は次のとおりです。

- *ImqDeadLetterHeader* クラスのオブジェクトでは、ゼロ
- *ImqIMSBridgeHeader* クラスのオブジェクトでは、MQIIH_NONE
- *ImqReferenceHeader* クラスのオブジェクトでは、MQRMHF_LAST
- *ImqCICSBridgeHeader* クラスのオブジェクトでは、MQCIH_NONE
- *ImqWorkHeader* クラスのオブジェクトでは、MQWIH_NONE

コンストラクター

ImqHeader();

デフォルトのコンストラクター。

ImqHeader(const ImqHeader & header);

コピー・コンストラクター。

オブジェクト・メソッド (共有)

void operator = (const ImqHeader & header);

インスタンス・データを *header* からコピーし、既存のインスタンス・データを置き換えます。

virtual MQLONG characterSet() const ;
character set を返します。

virtual void setCharacterSet(const MQLONG ccsid = MQCCSI_Q_MGR);
character set を設定します。

virtual MQLONG encoding() const ;
encoding を返します。

virtual void setEncoding(const MQLONG encoding = MQENC_NATIVE);
encoding を設定します。

virtual ImqString format() const ;
 後書きブランクを含め、**format** のコピーを返します。

virtual void setFormat(const char * name = 0);
format を設定し、後書きブランクで 8 文字まで埋め込みます。

virtual MQLONG headerFlags() const ;
header flags を返します。

virtual void setHeaderFlags(const MQLONG flags = 0);
header flags を設定します。

ImqIMSBridgeHeader C++ クラス

このクラスは MQIIH データ構造体の機能をカプセル化します。

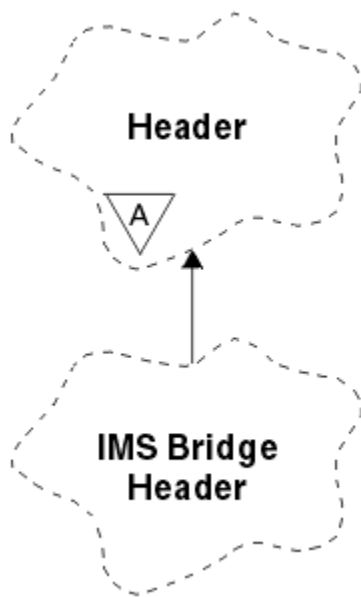


図 23. ImqIMSBridgeHeader クラス

このクラスのオブジェクトは、IBM MQ for z/OS を介して IMS ブリッジにメッセージを送信するアプリケーションによって使用されます。

注：ImqHeader の **character set** および **encoding** にはデフォルトの値が定められており、それ以外の値に設定することはできません。

- [1836 ページの『オブジェクトの属性』](#)
- [1836 ページの『コンストラクター』](#)
- [1836 ページの『多重定義された ImqItem メソッド』](#)
- [1836 ページの『オブジェクト・メソッド \(共有\)』](#)
- [1837 ページの『オブジェクト・データ \(保護\)』](#)
- [1837 ページの『理由コード』](#)

オブジェクトの属性

authenticator

RACF パスワードまたはパスチケット。長さは MQ_AUTHENTICATOR_LENGTH です。初期値は MQIAUT_NONE です。

commit mode

コミット・モード。IMS コミット・モードについて詳しくは、「OTMA ユーザーズ・ガイド」を参照してください。初期値は MQICM_COMMIT_THEN_SEND です。取得可能な値は MQICM_SEND_THEN_COMMIT です。

logical terminal override

論理端末の指定変更。長さは MQ_LTERM_OVERRIDE_LENGTH です。初期値はヌル・ストリングです。

message format services map name

MFS マップ名。長さは MQ_MFS_MAP_NAME_LENGTH です。初期値はヌル・ストリングです。

reply-to format

任意の応答の形式。長さは MQ_FORMAT_LENGTH です。初期値は MQFMT_NONE です。

security scope

IMS セキュリティー処理の有効範囲。初期値は MQISS_CHECK です。取得可能な値は MQISS_FULL です。

transaction instance id

トランザクション・インスタンス ID。長さが MQ_TRAN_INSTANCE_ID_LENGTH の 2 進 (MQBYTE16) 値です。初期値は MQITII_NONE です。

transaction state

IMS 会話の状態。初期値は MQITS_NOT_IN_CONVERSATION です。取得可能な値は MQITS_IN_CONVERSATION です。

コンストラクター

ImqIMSBridgeHeader() ;

デフォルトのコンストラクター。

ImqIMSBridgeHeader(const ImqIMSBridgeHeader & header);

コピー・コンストラクター。

多重定義された ImqItem メソッド

virtual ImqBoolean copyOut (ImqMessage & msg);

MQIIH データ構造体をメッセージ・バッファーの始めに挿入し、既存のメッセージ・データを後ろにずらします。また、*msg* の形式を MQFMT_IMS に設定します。

詳細については、親クラス・メソッドの説明を参照してください。

virtual ImqBoolean pasteIn (ImqMessage & msg);

メッセージ・バッファーから MQIIH データ構造体を読み取ります。

読み取りが正しく実行されるようにするには、*msg* オブジェクトのエンコードを MQENC_NATIVE に設定する必要があります。MQGMO_CONVERT を使用してメッセージを MQENC_NATIVE に取り出します。

正常に実行されるためには、ImqMessage の format が MQFMT_IMS でなければなりません。

詳細については、親クラス・メソッドの説明を参照してください。

オブジェクト・メソッド (共有)

void operator = (const ImqIMSBridgeHeader & header);

インスタンス・データを *header* からコピーし、既存のインスタンス・データを置き換えます。

ImqString authenticator() const;

authenticator のコピーを、長さ MQ_AUTHENTICATOR_LENGTH まで後書きブランクを埋め込んで返します。

void setAuthenticator (const char * name);

authenticator を設定します。

MQCHAR commitMode () const ;

commit mode を返します。

void setCommitMode (const MQCHAR mode);

commit mode を設定します。

ImqString logicalTerminalOverride () const ;

logical terminal override のコピーを返します。

void setLogicalTerminalOverride (const char * override);

logical terminal override を設定します。

ImqString messageFormatServicesMapName () const ;

message format services map name のコピーを返します。

void setMessageFormatServicesMapName (const char * name);

message format services map name を設定します。

ImqString replyToFormat() const;

reply-to format のコピーを、長さ MQ_FORMAT_LENGTH まで後書きブランクを埋め込んで返します。

void setReplyToFormat (const char * format);

reply-to format を、長さ MQ_FORMAT_LENGTH まで後書きブランクを埋め込んで設定します。

MQCHAR securityScope () const ;

security scope を返します。

void setSecurityScope (const MQCHAR scope);

security scope を設定します。

ImqBinary transactionInstanceId () const ;

transaction instance id のコピーを返します。

ImqBoolean setTransactionInstanceId (const ImqBinary & id);

transaction instance id を設定します。 *token* の data length は、ゼロまたは MQ_TRAN_INSTANCE_ID_LENGTH のいずれかでなければなりません。このメソッドは、正常に終了した場合には TRUE を返します。

void setTransactionInstanceId (const MQBYTE16 id = 0);

transaction instance id を設定します。 *id* はゼロであっても構いません。これは MQITII_NONE を指定するのと同じです。 *id* が非ゼロの場合には、MQ_TRAN_INSTANCE_ID_LENGTH バイトの 2 進データをアドレッシングするものでなければなりません。MQITII_NONE などの事前定義値を使用する場合は、確実にシグニチャーが一致するようにキャスト (例えば、(MQBYTE *)MQITII_NONE) を作成しなければなりません。

MQCHAR transactionState () const ;

transaction state を返します。

void setTransactionState (const MQCHAR state);

transaction state を設定します。

オブジェクト・データ (保護)

MQIIH omqiih

MQIIH データ構造体。

理由コード

- MQRC_BINARY_DATA_LENGTH_ERROR
- MQRC_INCONSISTENT_FORMAT
- MQRC_ENCODING_ERROR
- MQRC_STRUC_ID_ERROR

ImqItem C++ クラス

この抽象クラスは、1つのメッセージ内の1つの項目 (おそらく複数の項目のうちの1つ) を表します。

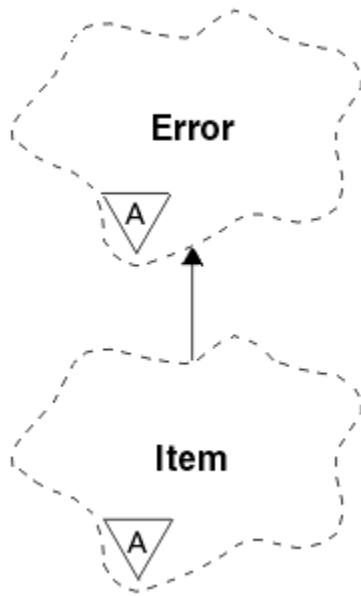


図 24. *ImqItem* クラス

項目は、メッセージ・バッファ内で1つに連結されます。いずれの特殊化も、1つの構造体 ID で始まる特定のデータ構造体と関連付けられます。

この抽象クラス内で多様メソッドを使用することにより、メッセージとの間で項目をコピーできます。*ImqMessage* クラスの **readItem** および **writeItem** メソッドでは、アプリケーション・プログラムにとって無理のない別の方法でこれらの多様メソッドを呼び出せるようにします。

- [1838 ページの『オブジェクトの属性』](#)
- [1838 ページの『コンストラクター』](#)
- [1838 ページの『クラス・メソッド \(共有\)』](#)
- [1839 ページの『オブジェクト・メソッド \(共有\)』](#)
- [1839 ページの『理由コード』](#)

オブジェクトの属性

structure id

データ構造体の始めにある、4文字のストリングです。この属性は読み取り専用です。この属性は、派生クラスへの使用を検討してください。この属性は、自動的に組み込まれません。

コンストラクター

ImqItem();

デフォルトのコンストラクター。

ImqItem(const ImqItem & item);

コピー・コンストラクター。

クラス・メソッド (共有)

static ImqBoolean structureIdIs (const char * structure-id-to-test, const ImqMessage & msg);

着信 *msg* 内の次の *ImqItem* の **structure id** が、*structure-id-to-test* と同じであれば、TRUE を返します。次の項目は、現在 *ImqCache* の **data pointer** によってアドレッシングされているメッセージ・バッファの部分として識別されます。このメソッドは、**structure id** に依存しているので、すべての *ImqItem* 派生クラスにうまくいくという保証はありません。

オブジェクト・メソッド (共有)

void operator = (const ImqItem & item);

インスタンス・データを *item* からコピーし、既存のインスタンス・データを置き換えます。

virtual ImqBoolean copyOut (ImqMessage & msg) = 0 ;

このオブジェクトを次の項目として出力メッセージ・バッファに書き込み、既存の項目があればそれに付加します。書き込み操作が正常に実行されると、ImqCache の **data length** は増やされます。このメソッドは、正常に終了した場合には TRUE を返します。

特定のサブクラスを扱うには、このメソッドを上書きしてください。

virtual ImqBoolean pasteIn (ImqMessage & msg) = 0 ;

このオブジェクトを着信メッセージ・バッファから破壊読み取りします。ここで破壊読み取りといっているのは、ImqCache の **data pointer** が移動するという意味です。ただし、バッファの内容は変わらないため、ImqCache の **data pointer** を再設定すればデータの読み取りが再び可能になります。

このオブジェクトの (サブ) クラスは、*msg* オブジェクトのメッセージ・バッファ内で次に検出された **structure id** と矛盾しないものでなければなりません。

msg オブジェクトの **encoding** は、MQENC_NATIVE でなければなりません。ImqMessage の **encoding** を MQENC_NATIVE に設定し、ImqGetMessageOptions の **options** に MQGMO_CONVERT を含めた上でメッセージの検索を行うようにしてください。

読み取り操作が正常に行われると、ImqCache の **data length** は減少します。このメソッドは、正常に終了した場合には TRUE を返します。

特定のサブクラスを扱うには、このメソッドを上書きしてください。

理由コード

- MQRC_ENCODING_ERROR
- MQRC_STRUC_ID_ERROR
- MQRC_INCONSISTENT_FORMAT
- MQRC_INSUFFICIENT_BUFFER
- MQRC_INSUFFICIENT_DATA

ImqMessage C++ クラス

このクラスは、MQMD データ構造体をカプセル化し、メッセージ・データの構築と再構築を扱います。

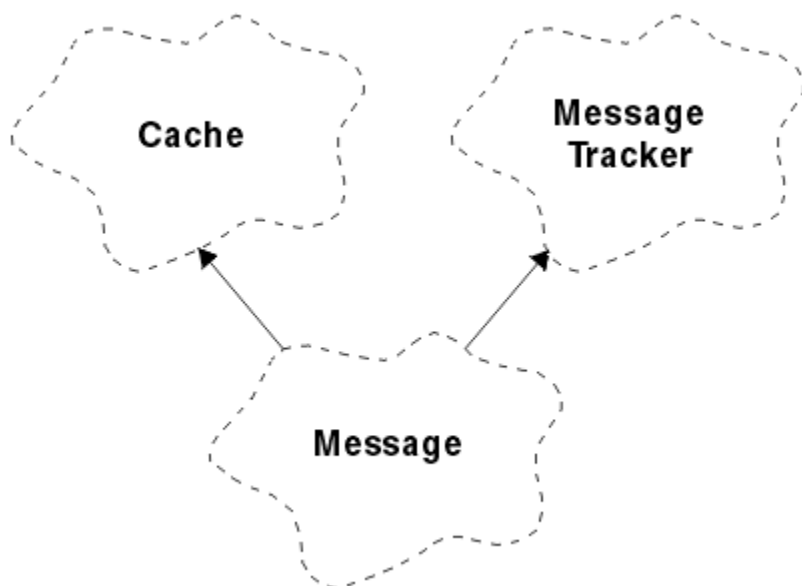


図 25. ImqMessage クラス

- [1840 ページの『オブジェクトの属性』](#)
- [1843 ページの『コンストラクター』](#)
- [1843 ページの『オブジェクト・メソッド \(共有\)』](#)
- [1845 ページの『オブジェクト・メソッド \(保護\)』](#)
- [1845 ページの『オブジェクト・データ \(保護\)』](#)

オブジェクトの属性

application ID data

メッセージに関連付けられた ID 情報。初期値はヌル・ストリングです。

application origin data

メッセージに関連付けられた起点情報。初期値はヌル・ストリングです。

backout count

メッセージが試験的に取り出され、次にバックアウトされた回数。初期値はゼロです。この属性は読み取り専用です。

character set

コード化文字セット ID。初期値は MQCCSI_Q_MGR です。以下の値が取得可能な値です。

- MQCCSI_INHERIT
- MQCCSI_EMBEDDED

自分で選択したコード化文字セット ID を使用することもできます。詳細は、[935 ページの『コード・ページ変換』](#)を参照してください。

encoding

メッセージ・データのマシン・エンコード。初期値は MQENC_NATIVE です。

expiry

IBM MQ が未検索メッセージを破棄する前に保持する期間を制御する時間依存の数量。初期値は MQEI_UNLIMITED です。

形式

バッファ内のデータのレイアウトを記述する形式 (テンプレート) の名前。8 文字を超える名前は、8 文字に切り捨てられます。名前は、必ず 8 文字の長さまでブランクで埋め込まれます。初期の定数値は MQFMT_NONE です。以下の追加の定数も可能です。

- MQFMT_ADMIN
- MQFMT_CICS
- MQFMT_COMMAND_1
- MQFMT_COMMAND_2
- MQFMT_DEAD_LETTER_HEADER
- MQFMT_DIST_HEADER
- MQFMT_EVENT
- MQFMT_IMS
- MQFMT_IMS_VAR_STRING
- MQFMT_MD_EXTENSION
- MQFMT_PCF
- MQFMT_REF_MSG_HEADER
- MQFMT_RF_HEADER
- MQFMT_STRING
- MQFMT_TRIGGER
- MQFMT_WORK_INFO_HEADER
- MQFMT_XMIT_Q_HEADER

自分で選択したアプリケーション特有のストリングを使用することもできます。詳細については、メッセージ記述子 (MQMD) の [413 ページ](#) の『[Format \(MQCHAR8\)](#)』フィールドを参照してください。

message flags

分割制御情報。初期値は、MQMF_SEGMENTATION_INHIBITED です。以下の値が取得可能な値です。

- MQMF_SEGMENTATION_ALLOWED
- MQMF_MSG_IN_GROUP
- MQMF_LAST_MSG_IN_GROUP
- MQMF_SEGMENT
- MQMF_LAST_SEGMENT
- MQMF_NONE

メッセージ・タイプ

メッセージの広範囲なカテゴリー化。初期値は MQMT_DATAGRAM です。以下の値が取得可能な値です。

- MQMT_SYSTEM_FIRST
- MQMT_SYSTEM_LAST
- MQMT_DATAGRAM
- MQMT_REQUEST
- MQMT_REPLY
- MQMT_REPORT
- MQMT_APPL_FIRST
- MQMT_APPL_LAST

自分で選択したアプリケーション特有の値を使用することもできます。詳細については、メッセージ記述子 (MQMD) の [403 ページ](#) の『[MsgType \(MQLONG\)](#)』フィールドを参照してください。

offset

オフセット情報。初期値はゼロです。

original length

分割されたメッセージの元の長さ。初期値は MQOL_UNDEFINED です。

persistence

当該メッセージが重大であり、持続ストレージを使用して常にバックアップを取っておく必要があることを指示します。このオプションは、パフォーマンスの低下を含意します。初期値は MQPER_PERSISTENCE_AS_Q_DEF です。以下の値が取得可能な値です。

- MQPER_PERSISTENT
- MQPER_NOT_PERSISTENT

priority

伝送および送達の実相優先順位。同じ優先順位のメッセージは、通常、提供されたのと同じ順序で送達されます (ただし、これが確実に行われるようにするには、満たさなければならない基準がいくつかあります)。初期値は MQPRI_PRIORITY_AS_Q_DEF です。

property validation

メッセージのプロパティが設定されているときにプロパティを検証するかどうかを指定します。初期値は MQCMHO_DEFAULT_VALIDATION です。以下の値が取得可能な値です。

- MQCMHO_VALIDATE
- MQCMHO_NO_VALIDATION

以下のメソッドは、**property validation** に作用します。

MQLONG propertyValidation() const ;

property validation オプションを返します。

void setPropertyValidation(const MQLONG option);
property validation オプションを設定します。

put application name

メッセージを書き込むアプリケーションの名前。初期値はヌル・ストリングです。

put application type

メッセージを書き込むアプリケーションのタイプ。初期値は MQAT_NO_CONTEXT です。以下の値が取得可能な値です。

- MQAT_AIX
- MQAT_CICS
- MQAT_CICS_BRIDGE
- MQAT_DOS
- MQAT_IMS
- MQAT_IMS_BRIDGE
- MQAT_MVS
- MQAT_NOTES_AGENT
- MQAT_OS2
- MQAT_OS390
- MQAT_OS400
- MQAT_QMGR
- MQAT_UNIX
- MQAT_WINDOWS
- MQAT_WINDOWS_NT
- MQAT_XCF
- MQAT_DEFAULT
- MQAT_UNKNOWN
- MQAT_USER_FIRST
- MQAT_USER_LAST

自分で選択したアプリケーション特有のストリングを使用することもできます。詳細については、メッセージ記述子 (MQMD) の [428 ページ](#)の『PutApplType (MQLONG)』フィールドを参照してください。

put date

メッセージが書き込まれた日付。初期値はヌル・ストリングです。

put time

メッセージが書き込まれた時刻。初期値はヌル・ストリングです。

reply-to queue manager name

応答が送られるキュー・マネージャーの名前。初期値はヌル・ストリングです。

reply-to queue name

応答が送られるキューの名前。初期値はヌル・ストリングです。

レポート

メッセージと関連付けられているフィードバック情報。初期値は MQRO_NONE です。以下の値が取得可能な値です。

- MQRO_EXCEPTION
- MQRO_EXCEPTION_WITH_DATA
- MQRO_EXCEPTION_WITH_FULL_DATA *
- MQRO_EXPIRATION
- MQRO_EXPIRATION_WITH_DATA

- MQRO_EXPIRATION_WITH_FULL_DATA *
- MQRO_COA
- MQRO_COA_WITH_DATA
- MQRO_COA_WITH_FULL_DATA *
- MQRO_COD
- MQRO_COD_WITH_DATA
- MQRO_COD_WITH_FULL_DATA *
- MQRO_PAN
- MQRO_NAN
- MQRO_NEW_MSG_ID
- MQRO_NEW_CORREL_ID
- MQRO_COPY_MSG_ID_TO_CORREL_ID
- MQRO_PASS_CORREL_ID
- MQRO_DEAD_LETTER_Q
- MQRO_DISCARD_MSG

* は、IBM MQ for z/OS ではサポートされていない値を示しています。

sequence number

グループ内のメッセージを識別するシーケンス情報。初期値は 1 です。

total message length

メッセージを最後に読み取ろうとしたときに有効だったバイト数。最後のメッセージに切り捨てが発生した場合、あるいは切り捨ての発生を理由として読み取りが行われなかった場合、この値は ImqCache の **message length** より大きくなります。この属性は読み取り専用です。初期値はゼロです。

この属性は、メッセージの切り捨てに関連した状況で有用です。

ユーザー ID

メッセージと関連付けられているユーザー ID。初期値はヌル・ストリングです。

コンストラクター

ImqMessage();

デフォルトのコンストラクター。

ImqMessage(const ImqMessage & msg);

コピー・コンストラクター。詳細については、**operator =** メソッドを参照してください。

オブジェクト・メソッド (共有)

void operator = (const ImqMessage & msg);

msg から MQMD およびメッセージ・データをコピーします。このオブジェクトのユーザーによってバッファが提供されている場合は、コピーされるデータの量は有効なバッファ・サイズまでに制限されます。バッファが提供されていない場合は、システムは、必ず十分なサイズのバッファがコピーされたデータに使用できるようにします。

ImqString applicationIdData() const ;

application ID data のコピーを返します。

void setApplicationIdData(const char * data = 0);

application ID data を設定します。

ImqString applicationOriginData() const ;

application origin data のコピーを返します。

void setApplicationOriginData(const char * data = 0);

application origin data を設定します。

MQLONG backoutCount() const ;
backout count を返します。

MQLONG characterSet() const ;
character set を返します。

void setCharacterSet(const MQLONG ccsid = MQCCSI_Q_MGR);
character set を設定します。

MQLONG encoding() const ;
encoding を返します。

void setEncoding(const MQLONG encoding = MQENC_NATIVE);
encoding を設定します。

MQLONG expiry() const ;
expiry を返します。

void setExpiry(const MQLONG expiry);
expiry を設定します。

ImqString format() const ;
後書きブランクを含め、**format** のコピーを返します。

ImqBoolean formatIs(const char * *format-to-test*) const ;
format が *format-to-test* と同じであれば、TRUE を返します。

void setFormat(const char * *name* = 0);
format を設定し、後書きブランクで 8 文字まで埋め込みます。

MQLONG messageFlags() const ;
message flags を返します。

void setMessageFlags(const MQLONG *flags*);
message flags を設定します。

MQLONG messageType() const ;
message type を返します。

void setMessageType(const MQLONG *type*);
message type を設定します。

MQLONG offset() const ;
offset を返します。

void setOffset(const MQLONG *offset*);
offset を設定します。

MQLONG originalLength() const ;
original length を返します。

void setOriginalLength(const MQLONG *length*);
original length を設定します。

MQLONG persistence() const ;
persistence を返します。

void setPersistence(const MQLONG *persistence*);
persistence を設定します。

MQLONG priority() const ;
priority を返します。

void setPriority(const MQLONG *priority*);
priority を設定します。

ImqString putApplicationName() const ;
put application name のコピーを返します。

void setPutApplicationName(const char * *name* = 0);
put application name を設定します。

MQLONG putApplicationType() const ;
put application type を返します。

void setPutApplicationType(const MQLONG type = MQAT_NO_CONTEXT);
 put application type を設定します。

ImqString putDate() const ;
 put date のコピーを返します。

void setPutDate(const char * date = 0);
 put date を設定します。

ImqString putTime() const ;
 put time のコピーを返します。

void setPutTime(const char * time = 0);
 put time を設定します。

ImqBoolean readItem (ImqItem & item);
 ImqItem の **pasteIn** メソッドを使用して、メッセージ・バッファから *item* オブジェクトへ読み込みます。正常に終了した場合は TRUE を返します。

ImqString replyToQueueManagerName() const ;
 reply-to queue manager name のコピーを返します。

void setReplyToQueueManagerName(const char * name = 0);
 reply-to queue manager name を設定します。

ImqString replyToQueueName() const ;
 reply-to queue name のコピーを返します。

void setReplyToQueueName(const char * name = 0);
 reply-to queue name を設定します。

MQLONG report() const ;
 report を返します。

void setReport(const MQLONG report);
 report を設定します。

MQLONG sequenceNumber() const ;
 sequence number を返します。

void setSequenceNumber(const MQLONG number);
 sequence number を設定します。

size_t totalMessageLength() const ;
 total message length を返します。

ImqString userId() const ;
 user id のコピーを返します。

void setUserId(const char * id = 0);
 user id を設定します。

ImqBoolean writeItem (ImqItem & item);
 ImqItem の **copyOut** メソッドを使用して、*item* オブジェクトからメッセージ・バッファへ書き込みます。書き込みは、挿入、置換、または付加の形態をとる場合がありますが、これは *item* オブジェクトのクラスによって決まります。このメソッドは、正常に終了した場合には TRUE を返します。

オブジェクト・メソッド (保護)

static void setVersionSupported(const MQLONG);
 MQMD version を設定します。デフォルトは、MQMD_VERSION_2 です。

オブジェクト・データ (保護)

 **MQMD1 omqmd**
 z/OS での MQMD データ構造体。

 **MQMD2 omqmd**
[マルチプラットフォーム](#) での MQMD データ構造体。

ImqMessageTracker C++ クラス

このクラスは、ImqMessage オブジェクトまたは ImqQueue オブジェクトのいずれかに関連付けることができる、この 2 つのオブジェクトの属性をカプセル化します。

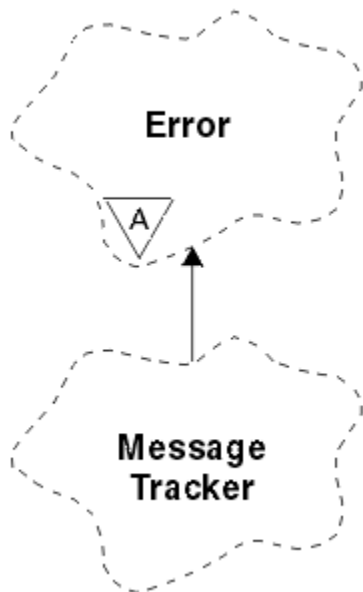


図 26. ImqMessageTracker クラス

このクラスは、[1797 ページの『ImqMessageTracker 相互参照』](#)にリストされている MQI 呼び出しと関連があります。

- [1846 ページの『オブジェクトの属性』](#)
- [1847 ページの『コンストラクター』](#)
- [1847 ページの『オブジェクト・メソッド \(共有\)』](#)
- [1848 ページの『理由コード』](#)

オブジェクトの属性

accounting token

長さ MQ_ACCOUNTING_TOKEN_LENGTH の 2 進値 (MQBYTE32)。初期値は MQACT_NONE です。

correlation id

メッセージを相互に関連付ける目的でユーザーが割り当てた、長さが MQ_CORREL_ID_LENGTH の 2 進値 (MQBYTE24)。初期値は MQCI_NONE です。取得可能な値は MQCI_NEW_SESSION です。

feedback

メッセージとともに送られるフィードバック情報。初期値は MQFB_NONE です。以下の値が取得可能な値です。

- MQFB_SYSTEM_FIRST
- MQFB_SYSTEM_LAST
- MQFB_APPL_FIRST
- MQFB_APPL_LAST
- MQFB_COA
- MQFB_COD
- MQFB_EXPIRATION
- MQFB_PAN
- MQFB_NAN

- MQFB_QUIT
- MQFB_DATA_LENGTH_ZERO
- MQFB_DATA_LENGTH_NEGATIVE
- MQFB_DATA_LENGTH_TOO_BIG
- MQFB_BUFFER_OVERFLOW
- MQFB_LENGTH_OFF_BY_ONE
- MQFB_IIH_ERROR
- MQFB_NOT_AUTHORIZED_FOR_IMS
- MQFB_IMS_ERROR
- MQFB_IMS_FIRST
- MQFB_IMS_LAST
- MQFB_CICS_APPL_ABENDED
- MQFB_CICS_APPL_NOT_STARTED
- MQFB_CICS_BRIDGE_FAILURE
- MQFB_CICS_CCSID_ERROR
- MQFB_CICS_CIH_ERROR
- MQFB_CICS_COMMAREA_ERROR
- MQFB_CICS_CORREL_ID_ERROR
- MQFB_CICS_DLQ_ERROR
- MQFB_CICS_ENCODING_ERROR
- MQFB_CICS_INTERNAL_ERROR
- MQFB_CICS_NOT_AUTHORIZED
- MQFB_CICS_UOW_BACKED_OUT
- MQFB_CICS_UOW_ERROR

自分で選択したアプリケーション特有のストリングを使用することもできます。詳細については、メッセージ記述子 (MQMD) の [407 ページ](#) の『[Feedback \(MQLONG\)](#)』フィールドを参照してください。

group id

キュー内で固有の長さ MQ_GROUP_ID_LENGTH の 2 進値 (MQBYTE24)。初期値は MQGI_NONE です。

message id

キュー内で固有の長さ MQ_MSG_ID_LENGTH の 2 進値 (MQBYTE24)。初期値は MQMI_NONE です。

コンストラクター

ImqMessageTracker();

デフォルトのコンストラクター。

ImqMessageTracker(const ImqMessageTracker & tracker);

コピー・コンストラクター。詳細については、**operator =** メソッドを参照してください。

オブジェクト・メソッド (共有)

void operator = (const ImqMessageTracker & tracker);

インスタンス・データを *tracker* からコピーし、既存のインスタンス・データを置き換えます。

ImqBinary accountingToken() const ;

accounting token のコピーを返します。

ImqBoolean setAccountingToken (const ImqBinary & token);

accounting token を設定します。 *token* の **data length** は、ゼロまたは、MQ_ACCOUNTING_TOKEN_LENGTH のいずれかでなければなりません。このメソッドは、正常に終了した場合には TRUE を返します。

void setAccountingToken(const MQBYTE32 token = 0);

accounting token を設定します。 *token* はゼロであっても構いません。これは MQACT_NONE を指定するのと同じです。 *token* が非ゼロの場合には、MQ_ACCOUNTING_TOKEN_LENGTH バイトの 2 進データをアドレッシングするものでなければなりません。MQACT_NONE などの事前定義値を使用する場合は、確実にシグニチャーが一致するようにキャスト (例えば、(MQBYTE *)MQACT_NONE) を作成しなければならないことがあります。

ImqBinary correlationId() const ;

correlation id のコピーを返します。

ImqBoolean setCorrelationId (const ImqBinary & token);

correlation id を設定します。 *token* の **data length** は、ゼロまたは、MQ_CORREL_ID_LENGTH のいずれかでなければなりません。このメソッドは、正常に終了した場合には TRUE を返します。

void setCorrelationId(const MQBYTE24 id = 0);

correlation id を設定します。 *id* はゼロであっても構いません。これは MQCI_NONE を指定するのと同じです。 *id* が非ゼロの場合には、MQ_CORREL_ID_LENGTH バイトの 2 進データをアドレッシングするものでなければなりません。MQCI_NONE などの事前定義値を使用する場合は、確実にシグニチャーが一致するようにキャスト (例えば、(MQBYTE *)MQCI_NONE) を作成しなければならないことがあります。

MQLONG feedback() const ;

feedback を返します。

void setFeedback(const MQLONG feedback);

feedback を設定します。

ImqBinary groupId() const ;

group id のコピーを返します。

ImqBoolean setGroupId (const ImqBinary & token);

group id を設定します。 *token* の **data length** は、ゼロまたは、MQ_GROUP_ID_LENGTH のいずれかでなければなりません。このメソッドは、正常に終了した場合には TRUE を返します。

void setGroupId(const MQBYTE24 id = 0);

group id を設定します。 *id* はゼロであっても構いません。これは MQGI_NONE を指定するのと同じです。 *id* が非ゼロの場合には、MQ_GROUP_ID_LENGTH バイトの 2 進データをアドレッシングするものでなければなりません。MQGI_NONE などの事前定義値を使用する場合は、確実にシグニチャーが一致するようにキャスト (例えば、(MQBYTE *)MQGI_NONE) を作成しなければならないことがあります。

ImqBinary messageId() const ;

message id のコピーを返します。

ImqBoolean setMessageId (const ImqBinary & token);

message id を設定します。 *token* の **data length** は、ゼロまたは、MQ_MSG_ID_LENGTH のいずれかでなければなりません。このメソッドは、正常に終了した場合には TRUE を返します。

void setMessageId(const MQBYTE24 id = 0);

message id を設定します。 *id* はゼロであっても構いません。これは MQMI_NONE を指定するのと同じです。 *id* が非ゼロの場合には、MQ_MSG_ID_LENGTH バイトの 2 進データをアドレッシングするものでなければなりません。MQMI_NONE などの事前定義値を使用する場合は、確実にシグニチャーが一致するようにキャスト (例えば、(MQBYTE *)MQMI_NONE) を作成しなければならないことがあります。

理由コード

- MQRC_BINARY_DATA_LENGTH_ERROR

ImqNamelist C++ クラス

このクラスは、名前リストをカプセル化します。

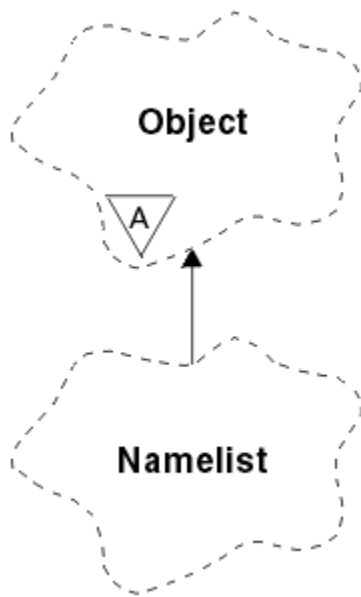


図 27. *ImqNamelist* クラス

このクラスは、[1798 ページの『ImqNamelist 相互参照』](#) にリストされている MQI 呼び出しと関連があります。

- [1849 ページの『オブジェクトの属性』](#)
- [1849 ページの『コンストラクター』](#)
- [1849 ページの『オブジェクト・メソッド \(共有\)』](#)
- [1850 ページの『理由コード』](#)

オブジェクトの属性

name count

namelist names に含まれているオブジェクト名の数。この属性は読み取り専用です。

namelist names

オブジェクト名。これらのオブジェクト名は、**name count** に示されます。この属性は読み取り専用です。

コンストラクター

ImqNamelist();

デフォルトのコンストラクター。

ImqNamelist(const ImqNamelist & list);

コピー・コンストラクター。ImqObject の **open status** は FALSE です。

ImqNamelist(const char * name);

ImqObject 名を **name** に設定します。

オブジェクト・メソッド (共有)

void operator = (const ImqNamelist & list);

インスタンス・データを *list* からコピーし、既存のインスタンス・データを置き換えます。ImqObject の **open status** は FALSE です。

ImqBoolean nameCount(MQLONG & count);

name count のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG nameCount ();

起こり得るエラーを指示せずに、**name count** を返します。

ImqBoolean namelistName (const MQLONG index、 ImqString & name);

ゼロを基準とした索引に従って **namelist names** のいずれかのコピーを提供します。正常に終了した場合は TRUE を返します。

ImqString namelistName (const MQLONG index);

起こり得るエラーを示さずに、ゼロを基準とした索引に従って **namelist names** のいずれかを返します。

理由コード

- MQRC_INDEX_ERROR
- MQRC_INDEX_NOT_PRESENT

ImqObject C++ クラス

このクラスは抽象クラスです。このクラスのオブジェクトは破棄されると自動的にクローズされ、その ImqQueueManager 接続は切断されます。

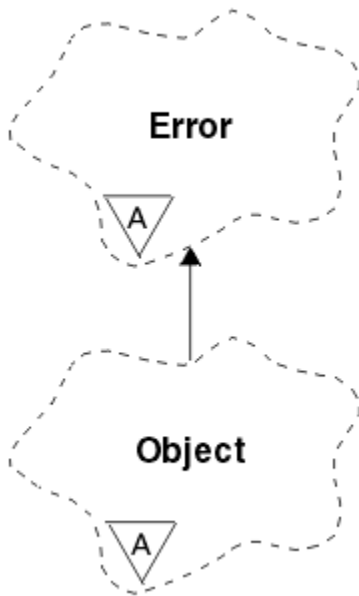


図 28. ImqObject クラス

このクラスは、[1798 ページの『ImqObject 相互参照』](#) にリストされている MQI 呼び出しと関連があります。

- [1850 ページの『クラス属性』](#)
- [1851 ページの『オブジェクトの属性』](#)
- [1852 ページの『コンストラクター』](#)
- [1852 ページの『クラス・メソッド \(共有\)』](#)
- [1852 ページの『オブジェクト・メソッド \(共有\)』](#)
- [1854 ページの『オブジェクト・メソッド \(保護\)』](#)
- [1855 ページの『オブジェクト・データ \(保護\)』](#)
- [1855 ページの『理由コード』](#)
-

クラス属性

動作 (behavior)

暗黙オープンの振る舞いを制御します。

IMQ_IMPL_OPEN (8L)

暗黙オープンが許可されます。これはデフォルトです。

オブジェクトの属性

alteration date

変更日。この属性は読み取り専用です。

alteration time

変更時刻。この属性は読み取り専用です。

alternate user id

代替ユーザーの ID。最大 MQ_USER_ID_LENGTH 文字です。初期値はヌル・ストリングです。

alternate security id

代替セキュリティの ID。長さ MQ_SECURITY_ID_LENGTH の 2 進値 (MQBYTE40)。初期値は MQSID_NONE です。

close options

オブジェクトのクローズ時に適用されるオプション。初期値は MQCO_NONE です。この属性は、暗黙の再オープン操作時には無視されます。再オープン時は、必ず値 MQCO_NONE が使用されるためです。

connection reference

必要に応じて (ローカル) キュー・マネージャーに接続できるようにする ImqQueueManager オブジェクトに対する参照。ImqQueueManager オブジェクトの場合、これはオブジェクト自身です。初期値はゼロです。

注: これを、指定のキューのキュー・マネージャー (場合によっては、リモートにある) を識別する queue manager name と混同しないでください。

description

キュー・マネージャー、キュー、名前リスト、またはプロセスの記述名 (最大 64 文字)。この属性は読み取り専用です。

名前

キュー・マネージャー、キュー、名前リスト、またはプロセスの名前 (最大 48 文字)。初期値はヌル・ストリングです。結果として生じた動的キューの名前に対する **open** のあとのモデル・キュー変更の名前です。

注: ImqQueueManager は、ヌル名にしても構いません。その場合、デフォルトのキュー・マネージャーを指定したことになります。open が正常に実行されると、ここに実際のキュー・マネージャーの名前が表示されます。ImqDistributionList は動的なので、必ずヌル名にしなければなりません。

next managed object

これは、特定の順序でなく、このクラスの次のオブジェクトで、このオブジェクトと同じ connection reference を持つものです。初期値はゼロです。

open options

オブジェクトのオープン時に適用されるオプション。初期値は MQOO_INQUIRE です。適切な値を設定する方法は、次の 2 通りあります。

1. open options を設定せず、open メソッドも使用しないでください。IBM MQ は、open options を自動的に調整し、必要に応じてオブジェクトを自動的にオープン、再オープン、およびクローズします。この結果、再オープン操作が不要に行われる場合があります。これは、IBM MQ が openFor メソッドを使用していて、それにより open options が繰り返し追加されるためです。
2. 結果的に MQI 呼び出し (1791 ページの『C++ と MQI の相互参照』を参照) が発生する任意のメソッドを使用する前に、open options を設定してください。こうすると、不要な再オープン操作が行われなくなります。潜在的な再開の問題が発生する可能性がある場合は、明示的にオープンオプションを設定してください (C++ 言語に関する考慮事項の「再開」を参照)。

open メソッドを使用する場合は、open options が適切であるかどうかを最初に確認する必要があります。ただし、必ずしも open メソッドを使用する必要はありません。IBM MQ は、引き続き 1 の場合と同じ動作を示しますが、2 の状況では動作は効率的になっています。

ゼロは有効な値ではないため、オブジェクトのオープンを行う前に適切な値を設定してください。それには、**setOpenOptions(IOpenOptions)** を指定してから **open()** を使用するか、または **openFor(IRequiredOpenOption)** を使用します。

注:

1. 配布リストに対して **open** メソッドを使用している間は、MQOO_INQUIRE の代わりに MQOO_OUTPUT が使用されます。これは、MQOO_OUTPUT がこの期間中に有効なただ 1 つの **open option** であるからです。しかし、**open** メソッドを使用しているアプリケーション・プログラムでは、必ず MQOO_OUTPUT を明示的に設定することをお勧めします。
2. クラスのおよび **resolved queue name resolved queue manager name** 属性を使用したい場合は、MQOO_RESOLVE_NAMES を指定してください。

open status

オブジェクトがオープンである (TRUE) か、クローズされている (FALSE) かを示します。初期値は FALSE です。この属性は読み取り専用です。

previous managed object

これは、特定の順序ではなく、このクラスの直前のオブジェクトで、このオブジェクトと同じ connection reference を持つものです。初期値はゼロです。

queue-manager-identifier

キュー・マネージャーの ID。この属性は読み取り専用です。

コンストラクター

ImqObject();

デフォルトのコンストラクター。

ImqObject(const ImqObject & object);

コピー・コンストラクター。open status は FALSE となります。

クラス・メソッド (共有)

static MQLONG behavior();

behavior を返します。

void setBehavior(const MQLONG behavior = 0);

behavior を設定します。

オブジェクト・メソッド (共有)

void operator = (const ImqObject & object);

必要に応じてクローズを実行し、object からインスタンス・データをコピーします。open status は FALSE となります。

ImqBoolean alterationDate(ImqString & date);

alteration date のコピーを提供します。正常に終了した場合は TRUE を返します。

ImqString alterationDate();

起こり得るエラーを示さずに、alteration date を返します。

ImqBoolean alterationTime(ImqString & time);

alteration time のコピーを提供します。正常に終了した場合は TRUE を返します。

ImqString alterationTime();

起こり得るエラーを示さずに、alteration time を返します。

ImqString alternateUserId () const ;

alternate user id のコピーを返します。

ImqBoolean setAlternateUserId (const char * id);

alternate user id を設定します。alternate user id を設定できるのは、open status が FALSE の間だけです。このメソッドは、正常に終了した場合には TRUE を返します。

ImqBinary alternateSecurityId() const ;

alternate security ID のコピーを返します。

ImqBoolean setAlternateSecurityId(const ImqBinary & token);

alternate security id を設定します。 alternate security id を設定できるのは、open status が FALSE の間だけです。 *token* の data length は、ゼロまたは MQ_SECURITY_ID_LENGTH のいずれかでなければなりません。 正常に終了した場合は TRUE を返します。

ImqBoolean setAlternateSecurityId(const MQBYTE* token = 0);

alternate security id を設定します。 *token* はゼロであっても構いません。これは MQSID_NONE を指定するのと同じです。 *token* が非ゼロの場合には、MQ_SECURITY_ID_LENGTH バイトの 2 進データをアドレッシングするものでなければなりません。 MQSID_NONE などの事前定義値を使用する場合は、確実にシングニチャーが一致するようにキャスト (例えば、(MQBYTE *)MQSID_NONE) を作成する必要があります。

alternate security id を設定できるのは、open status が TRUE の間だけです。 正常に終了した場合は TRUE を返します。

ImqBoolean setAlternateSecurityId(const unsigned char * id = 0);

alternate security id を設定します。

ImqBoolean close ();

open status を FALSE に設定します。 正常に終了した場合は TRUE を返します。

MQLONG closeOptions () const ;

close options を返します。

void setCloseOptions (const MQLONG options);

close options を設定します。

ImqQueueManager * connectionReference () const ;

connection reference を返します。

void setConnectionReference (ImqQueueManager & manager);

connection reference を設定します。

void setConnectionReference (ImqQueueManager * manager = 0);

connection reference を設定します。

virtual ImqBoolean description (ImqString & description) = 0 ;

description のコピーを提供します。 正常に終了した場合は TRUE を返します。

ImqString description ();

起り得るエラーを示さずに、description のコピーを返します。

virtual ImqBoolean name (ImqString & name);

name のコピーを提供します。 正常に終了した場合は TRUE を返します。

ImqString name ();

起り得るエラーを示さずに、name のコピーを返します。

ImqBoolean setName (const char * name = 0);

name を設定します。 name を設定できるのは、open status が FALSE であるとき、および connection status が FALSE であるとき (ImqQueueManager の場合) のみです。 正常に終了した場合は TRUE を返します。

ImqObject * nextManagedObject () const ;

next managed object を返します。

ImqBoolean open ();

さまざまな属性のうち特に open options と name を使用して必要に応じてオブジェクトをオープンすることにより、open status を TRUE に変更します。 ImqQueueManager の connection status を確実に TRUE にする必要がある場合、このメソッドは connection reference 情報と、ImqQueueManager の connect メソッドを使用します。 open status を返します。

ImqBoolean openFor (const MQLONG required-options = 0);

open options、または *required-options* パラメーター値によって示された振る舞いを保証する open options を指定して、オブジェクトがオープンされるようにします。

required-options がゼロである場合は、入力必須であり、いずれの入力オプションでも有効です。したがって、*open options* に既に、次のいずれかが含まれている場合は、

- MQOO_INPUT_AS_Q_DEF
- MQOO_INPUT_SHARED
- MQOO_INPUT_EXCLUSIVE

open options は既に有効であり、変更されません。*open options* に上記のいずれのオプションも含まれていない場合は、*open options* に MQOO_INPUT_AS_Q_DEF が設定されます。

required-options が非ゼロである場合は、必須指定のオプションが *open options* に追加されます。*required-options* が上記のいずれかのオプションである場合、それ以外のものはリセットされます。

open options のいずれかが変更され、オブジェクトが既にオープンになっている場合は、*open options* を調整するためにオブジェクトは一時的にクローズされ、再オープンされます。

正常に終了した場合は TRUE を返します。正常に行われたということは、オブジェクトが適切なオプションでオープンになっていることを示します。

MQLONG openOptions () const ;

open options を返します。

ImqBoolean setOpenOptions (const MQLONG options);

open options を設定します。*open options* を設定できるのは、*open status* が FALSE の間だけです。正常に終了した場合は TRUE を返します。

ImqBoolean openStatus () const ;

open status を返します。

ImqObject * previousManagedObject () const ;

previous managed object を返します。

ImqBoolean queueManagerIdentifier(ImqString & id);

queue manager identifier のコピーを提供します。正常に終了した場合は TRUE を返します。

ImqString queueManagerIdentifier();

起こり得るエラーを示さずに、*queue manager identifier* を返します。

オブジェクト・メソッド (保護)

virtual ImqBoolean closeTemporarily ();

再オープンの前にオブジェクトを安全にクローズします。正常に終了した場合は TRUE を返します。このメソッドでは、*open status* が TRUE であると想定しています。

MQHCONN connectionHandle () const ;

connection reference と関連付けられた MQHCONN を返します。この値は、*connection reference* がいない場合、または Manager が接続されていない場合はゼロです。

ImqBoolean inquire (const MQLONG int-attr, MQLONG & value);

整数値を返します。この索引は MQIA_* 値です。エラーがあった場合、値は MQIAV_UNDEFINED に設定されます。

ImqBoolean inquire (const MQLONG char-attr, char * & buffer, const size_t length);

文字ストリングを返します。この索引は MQCA_* 値です。

注：これらのメソッドは、いずれも 1 つの属性値しか返しません。複数の値についてスナップショットが必要であり、その値が 1 つのインスタンスについて互いに一貫している場合、IBM MQ C++ にはこの機能が備わっていないため、適切なパラメーターを指定して MQINQ 呼び出しを使用する必要があります。

virtual void openInformationDisperse ();

MQOPEN 呼び出しの直後に MQOD データ構造体の可変部分から情報を分散させます。

virtual ImqBoolean openInformationPrepare ();

MQOPEN 呼び出しの直前に MQOD データ構造体の可変部分の情報を準備し、正常に終了した場合は TRUE を返します。

ImqBoolean set (const MQLONG int-attr, const MQLONG value);

IBM MQ 整数属性を設定します。

ImqBoolean set (const MQLONG char-attr, const char * buffer, const size_t required-length);

IBM MQ 文字属性を設定します。

void setNextManagedObject (const ImqObject * object = 0);

next managed object を設定します。

重要: この関数は、管理対象オブジェクト・リストを損傷しないことが確かな場合に限り使用してください。

void setPreviousManagedObject (const ImqObject * object = 0);

previous managed object を設定します。

重要: この関数は、管理対象オブジェクト・リストを損傷しないことが確かな場合に限り使用してください。

オブジェクト・データ (保護)

MQHOBJ ohobj

IBM MQ オブジェクト・ハンドル (open status が TRUE の場合にのみ有効)。

MQOD omqod

組み込み MQOD データ構造体。このデータ構造体に割り振られているストレージ容量が、MQOD バージョン 2 に必要なストレージ容量となります。Inspect the version number (*omqod.Version*) and access the other fields as follows:

MQOD_VERSION_1

omqod に含まれているほかの全フィールドにアクセスできます。

MQOD_VERSION_2

omqod に含まれているほかの全フィールドにアクセスできます。

MQOD_VERSION_3

omqod.pmqod は、動的に割り振られた、ほかよりも容量の大きい MQOD を指すポインターです。*omqod* に含まれているほかのフィールドにはアクセスできません。*omqod.pmqod* でアドレスされた全フィールドにアクセスできます。

注: *omqod.pmqod.Version* が *omqod.Version* より小さくなる場合があります。これは、IBM MQ MQI client の機能が IBM MQ サーバーの機能より豊富であることを示しています。

理由コード

- MQRC_ATTRIBUTE_LOCKED
- MQRC_INCONSISTENT_OBJECT_STATE
- MQRC_NO_CONNECTION_REFERENCE
- MQRC_STORAGE_NOT_AVAILABLE
- MQRC_REOPEN_SAVED_CONTEXT_ERR
- (MQCLOSE からの理由コード)
- (MQCONN からの理由コード)
- (MQINQ からの理由コード)
- (MQOPEN からの理由コード)
- (MQSET からの理由コード)

ImqProcess C++ クラス

このクラスは、トリガー・モニターにより起動されることがあるアプリケーション・プロセス (タイプ MQOT_PROCESS の IBM MQ オブジェクト) をカプセル化します。

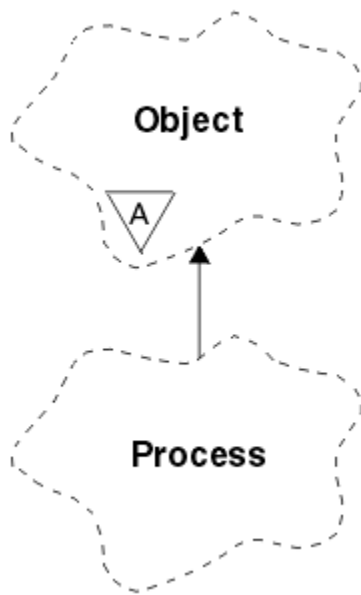


図 29. *ImqProcess* クラス

- [1856 ページの『オブジェクトの属性』](#)
- [1856 ページの『コンストラクター』](#)
- [1856 ページの『オブジェクト・メソッド \(共有\)』](#)

オブジェクトの属性

application id

アプリケーション・プロセスの ID。この属性は読み取り専用です。

application type

アプリケーション・プロセスのタイプ。この属性は読み取り専用です。

environment data

これは、プロセスの環境情報です。この属性は読み取り専用です。

ユーザー・データ

プロセスのユーザー・データ。この属性は読み取り専用です。

コンストラクター

ImqProcess();

デフォルトのコンストラクター。

ImqProcess(const ImqProcess & process);

コピー・コンストラクター。ImqObject の **open status** は FALSE です。

ImqProcess(const char * name);

ImqObject の **name** を設定します。

オブジェクト・メソッド (共有)

void operator = (const ImqProcess & process);

必要に応じてクローズを実行し、*process* からインスタンス・データをコピーします。ImqObject の **open status** は FALSE です。

ImqBoolean applicationId (ImqString & id);

application id のコピーを提供します。正常に終了した場合は TRUE を返します。

ImqString applicationId ();

起り得るエラーを示さずに、**application id** を返します。

ImqBoolean applicationType (MQLONG & type);

application type のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG applicationType();

起こり得るエラーを示さずに、**application type** を返します。

ImqBoolean environmentData (ImqString & data);

environment data のコピーを提供します。正常に終了した場合は TRUE を返します。

ImqString environmentData ();

起こり得るエラーを示さずに、**environment data** を返します。

ImqBoolean userData (ImqString & data);

user data のコピーを提供します。正常に終了した場合は TRUE を返します。

ImqString userData ();

起こり得るエラーを示さずに、**user data** を返します。

ImqPutMessageOptions C++ クラス

このクラスは、MQPMO データ構造体をカプセル化します。

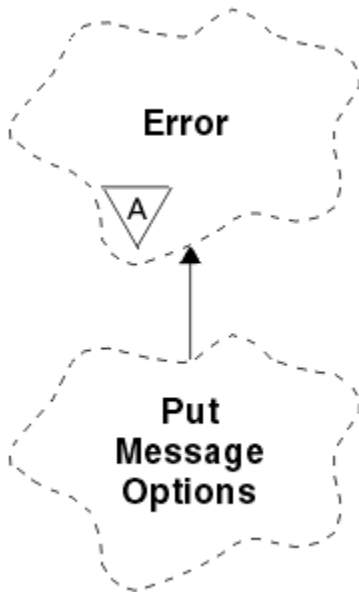


図 30. *ImqPutMessageOptions* クラス

- [1857 ページの『オブジェクトの属性』](#)
- [1858 ページの『コンストラクター』](#)
- [1858 ページの『オブジェクト・メソッド \(共有\)』](#)
- [1859 ページの『オブジェクト・データ \(保護\)』](#)
- [1859 ページの『理由コード』](#)

オブジェクトの属性

context reference

メッセージにコンテキストを提供する *ImqQueue*。初期設定時には、参照は存在しません。

オプション

書き込みメッセージ・オプション。初期値は *MQPMO_NONE* です。以下の値が取得可能な値です。

- *MQPMO_SYNCPOINT*
- *MQPMO_NO_SYNCPOINT*
- *MQPMO_NEW_MSG_ID*

- MQPMO_NEW_CORREL_ID
- MQPMO_LOGICAL_ORDER
- MQPMO_NO_CONTEXT
- MQPMO_DEFAULT_CONTEXT
- MQPMO_PASS_IDENTITY_CONTEXT
- MQPMO_PASS_ALL_CONTEXT
- MQPMO_SET_IDENTITY_CONTEXT
- MQPMO_SET_ALL_CONTEXT
- MQPMO_ALTERNATE_USER_AUTHORITY
- MQPMO_FAIL_IF QUIESCING

record fields

メッセージが書き込まれるときに PUT メッセージ・レコードの組み込みを制御するフラグ。初期値は MQPMRF_NONE です。以下の値が取得可能な値です。

- MQPMRF_MSG_ID
- MQPMRF_CORREL_ID
- MQPMRF_GROUP_ID
- MQPMRF_FEEDBACK
- MQPMRF_ACCOUNTING_TOKEN

ImqMessageTracker 属性は、指定されたフィールドについてはオブジェクトから取得されます。指定されないフィールドについては、ImqMessageTracker 属性は ImqMessage オブジェクトから取得されます。

resolved queue manager name

書き込み中に判別された宛先キュー・マネージャーの名前。初期値はヌルです。この属性は読み取り専用です。

resolved queue name

書き込み中に判別された宛先キューの名前。初期値はヌルです。この属性は読み取り専用です。

syncpoint participation

同期点制御を受けてメッセージが書き込まれる場合は TRUE です。

コンストラクター

ImqPutMessageOptions();

デフォルトのコンストラクター。

ImqPutMessageOptions(const ImqPutMessageOptions & pmo);

コピー・コンストラクター。

オブジェクト・メソッド (共有)

void operator = (const ImqPutMessageOptions & pmo);

インスタンス・データを pmo からコピーし、既存のインスタンス・データを置き換えます。

ImqQueue * contextReference () const ;

context reference を返します。

void setContextReference (const ImqQueue & queue);

context reference を設定します。

void setContextReference (const ImqQueue * queue = 0);

context reference を設定します。

MQLONG options () const ;

options を返します。

void setOptions (const MQLONG options);

syncpoint participation 値を組み込んで、options を設定します。

MQLONG recordFields () const ;

record fields を返します。

void setRecordFields (const MQLONG fields);

record fields を設定します。

ImqString resolvedQueueManagerName () const ;

resolved queue manager name のコピーを返します。

ImqString resolvedQueueName () const ;

resolved queue name のコピーを返します。

ImqBoolean syncPointParticipation () const ;

syncpoint participation 値を返します。options に MQPMO_SYNCPOINT が組み込まれている場合は、TRUE を返します。

void setSyncPointParticipation (const ImqBoolean sync);

syncpoint participation 値を設定します。sync が TRUE の場合、options は、MQPMO_SYNCPOINT が設定され、MQPMO_NO_SYNCPOINT が除外されるように変更されます。sync が FALSE の場合、options は、MQPMO_NO_SYNCPOINT を含み、MQPMO_SYNCPOINT が除外されるように変更されます。

オブジェクト・データ (保護)

MQPMO omqpmo

MQPMO データ構造体。

理由コード

- MQRC_STORAGE_NOT_AVAILABLE

ImqQueue C++ クラス

このクラスは、メッセージ・キュー (タイプ MQOT_Q の IBM MQ オブジェクト) をカプセル化します。

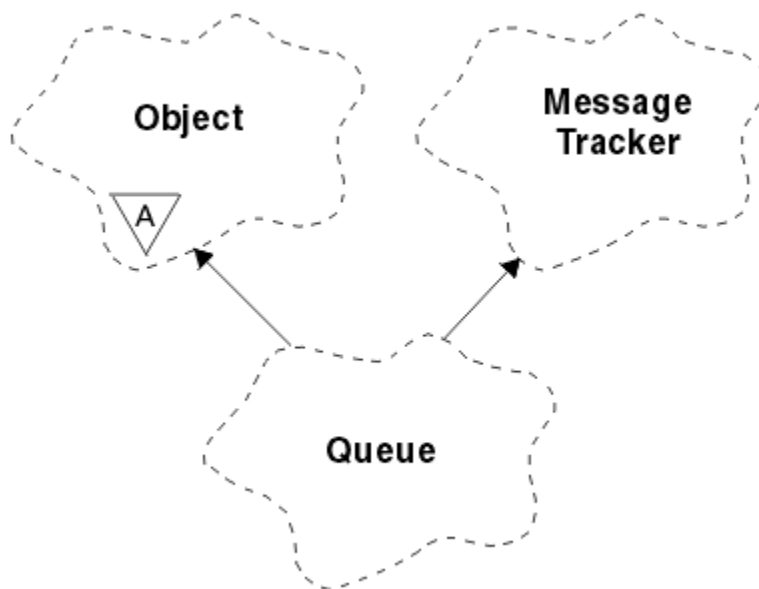


図 31. ImqQueue クラス

このクラスは、[1799 ページの表 898](#) にリストされている MQI 呼び出しと関連があります。

- [1860 ページの『オブジェクトの属性』](#)

- [1863 ページの『コンストラクター』](#)
- [1863 ページの『オブジェクト・メソッド \(共有\)』](#)
- [1869 ページの『オブジェクト・メソッド \(保護\)』](#)
- [1869 ページの『理由コード』](#)

オブジェクトの属性

backout requeue name

過剰バックアウト再キューイングの名前。この属性は読み取り専用です。

バックアウトしきい値

バックアウトしきい値。この属性は読み取り専用です。

base queue name

別名の元となるキューの名前。この属性は読み取り専用です。

cluster name

クラスターの名前。この属性は読み取り専用です。

cluster namelist name

クラスター名前リストの名前。この属性は読み取り専用です。

cluster workload rank

クラスター・ワークロード・ランク。この属性は読み取り専用です。

cluster workload priority

クラスター・ワークロード優先順位。この属性は読み取り専用です。

cluster workload use queue

クラスター・ワークロードの使用キューの値。この属性は読み取り専用です。

creation date

キューの作成日。この属性は読み取り専用です。

creation time

キュー作成時刻。この属性は読み取り専用です。

current depth

キュー上にあるメッセージの数。この属性は読み取り専用です。

default bind

デフォルトのバインド。この属性は読み取り専用です。

default input open option

デフォルトの入力用オープンのオプション。この属性は読み取り専用です。

default persistence

デフォルトのメッセージ持続性。この属性は読み取り専用です。

default priority

デフォルトのメッセージ優先度。この属性は読み取り専用です。

definition type

キュー定義タイプ。この属性は読み取り専用です。

depth high event

キューのサイズ上位イベントの制御属性。この属性は読み取り専用です。

depth high limit

キューのサイズの上限。この属性は読み取り専用です。

depth low event

キューのサイズ下位イベントの制御属性。この属性は読み取り専用です。

depth low limit

キューのサイズの下限。この属性は読み取り専用です。

depth maximum event

キューのサイズ最大イベントの制御属性。この属性は読み取り専用です。

distribution list reference

このキューを含め、複数のキューにメッセージを配布するのに使用できる ImqDistributionList へのオプション参照。初期値はヌルです。

注: ImqQueue オブジェクトがオープンされると、それが参照する任意のオープンの ImqDistributionList オブジェクトは、自動的にクローズされます。

distribution lists

配布リストをサポートするための伝送キューの機能。この属性は読み取り専用です。

dynamic queue name

動的キューの名前。初期値は AMQ.* です。すべての AIX, Linux, and Windows プラットフォームの場合。

harden get backout

バックアウト・カウントを固定するかどうかを判別します。この属性は読み取り専用です。

index type

索引タイプ。この属性は読み取り専用です。

inhibit get

読み取り操作が許されているかどうかを判別します。初期値はキューの定義により異なります。この属性は、別名またはローカル・キューについてのみ有効です。

inhibit put

書き込み操作が許されているかどうかを判別します。初期値はキューの定義により異なります。

initiation queue name

開始キューの名前。この属性は読み取り専用です。

maximum depth

キューで許されるメッセージの最大数。この属性は読み取り専用です。

最大メッセージ長

このキュー上のあらゆるメッセージの最大長。これは、関連付けられているキュー・マネージャーによって管理されるどのキューの最大長の値より小さいことがあります。この属性は読み取り専用です。

message delivery sequence

メッセージ優先順位が関係あるかどうかを判別します。この属性は読み取り専用です。

next distributed queue

このクラスの次のオブジェクト。特定の順序はなく、このオブジェクトと同じ **distribution list reference** を持つものです。初期値はゼロです。

チェーン内のオブジェクトが削除された場合は、その直前のオブジェクトと次のオブジェクトがアップデートされ、それらの分散キュー・リンクが、削除されたオブジェクトを指さないように変更されます。

non-persistent message class

このキューに入れられる非持続メッセージの信頼性のレベル。この属性は読み取り専用です。

インプット・カウントを開く

入力できるようにオープンになっている ImqQueue オブジェクトの数。この属性は読み取り専用です。

アウトプット・カウントを開く

出力できるようにオープンになっている ImqQueue オブジェクトの数。この属性は読み取り専用です。

previous distributed queue

このクラスの直前のオブジェクト。特定の順序はなく、このオブジェクトと同じ **distribution list reference** を持つものです。初期値はゼロです。

チェーン内のオブジェクトが削除された場合は、その直前のオブジェクトと次のオブジェクトがアップデートされ、それらの分散キュー・リンクが、削除されたオブジェクトを指さないように変更されます。

process name

プロセス定義の名前。この属性は読み取り専用です。

queue accounting

キューのアカウントリング情報のレベル。この属性は読み取り専用です。

キュー・マネージャー名

キューが入っているキュー・マネージャー (おそらくリモート) の名前。ここに指定するキュー・マネージャーを、接続を提供する (ローカル) キュー・マネージャーを参照する `ImqObject` の **connection reference** と混同しないでください。初期値はヌルです。

queue monitoring

キューのモニター・データ収集のレベル。この属性は読み取り専用です。

queue statistics

キューの統計データのレベル。この属性は読み取り専用です。

キュー・タイプ (queue type)

キュー・タイプ。この属性は読み取り専用です。

リモート・キュー・マネージャー名

リモート・キュー・マネージャーの名前。この属性は読み取り専用です。

remote queue name

リモート・キュー・マネージャーで認識されているとおりのリモート・キューの名前。この属性は読み取り専用です。

resolved queue manager name

解決済みのキュー・マネージャーの名前。この属性は読み取り専用です。

resolved queue name

解決済みのキューの名前。この属性は読み取り専用です。

retention interval

キュー保持期間間隔。この属性は読み取り専用です。

scope

キュー定義の有効範囲。この属性は読み取り専用です。

サービス間隔 (service interval)

サービス間隔。この属性は読み取り専用です。

サービス・インターバル・イベント (service interval event)

サービス間隔イベントの制御属性。この属性は読み取り専用です。

shareability

キューを共用できるかどうか。この属性は読み取り専用です。

ストレージ・クラス

ストレージ・クラス。この属性は読み取り専用です。

伝送キュー名

伝送キューの名前。この属性は読み取り専用です。

trigger control

トリガー制御。初期値は、キュー定義により異なります。この属性は、ローカル・キューについてのみ有効です。

trigger data

トリガー・データです。初期値は、キュー定義により異なります。この属性は、ローカル・キューについてのみ有効です。

trigger depth

トリガー項目数。初期値は、キュー定義により異なります。この属性は、ローカル・キューについてのみ有効です。

trigger message priority

トリガーのしきい値メッセージ優先順位。初期値は、キュー定義により異なります。この属性は、ローカル・キューについてのみ有効です。

trigger type

トリガー・タイプ。初期値は、キュー定義により異なります。この属性は、ローカル・キューについてのみ有効です。

usage

使用法 この属性は読み取り専用です。

コンストラクター

ImqQueue();

デフォルトのコンストラクター。

ImqQueue(const ImqQueue & queue);

コピー・コンストラクター。ImqObject の **open status** は FALSE です。

ImqQueue(const char * name);

ImqObject の **name** を設定します。

オブジェクト・メソッド (共有)

void operator = (const ImqQueue & queue);

必要に応じてクローズを実行し、*queue* からインスタンス・データをコピーします。ImqObject の **open status** は FALSE です。

ImqBoolean backoutRequeueName (ImqString & name);

backout requeue name のコピーを提供します。正常に終了した場合は TRUE を返します。

ImqString backoutRequeueName ();

起り得るエラーを示さずに、**backout requeue name** を返します。

ImqBoolean backoutThreshold (MQLONG & threshold);

backout threshold のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG backoutThreshold();

起り得るエラーを示さずに、**backout threshold** 値を返します。

ImqBoolean baseQueueName (ImqString & name);

base queue name のコピーを提供します。正常に終了した場合は TRUE を返します。

ImqString baseQueueName ();

起り得るエラーを示さずに、**base queue name** を返します。

ImqBoolean clusterName(ImqString & name);

cluster name のコピーを提供します。正常に終了した場合は TRUE を返します。

ImqString clusterName();

起り得るエラーを示さずに、**cluster name** を返します。

ImqBoolean clusterNamelistName(ImqString & name);

cluster namelist name のコピーを提供します。正常に終了した場合は TRUE を返します。

ImqString clusterNamelistName();

起り得るエラーを示さずに、**cluster namelist name** を返します。

ImqBoolean clusterWorkLoadPriority (MQLONG & priority);

cluster workload priority 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG clusterWorkLoadPriority ();

起り得るエラーを示さずに、**cluster workload priority** 値を返します。

ImqBoolean clusterWorkLoadRank (MQLONG & rank);

cluster workload rank 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG clusterWorkLoadRank ();

起り得るエラーを示さずに、**cluster workload rank** 値を返します。

ImqBoolean clusterWorkLoadUseQ (MQLONG & useq);

cluster workload use queue 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG clusterWorkLoadUseQ ();

起り得るエラーを示さずに、**cluster workload use queue** 値を返します。

ImqBoolean creationDate (ImqString & date);

creation date のコピーを提供します。正常に終了した場合は TRUE を返します。

ImqString creationDate ();

起こり得るエラーを示さずに、**creation date** を返します。

ImqBoolean creationTime (ImqString & time);

creation time のコピーを提供します。正常に終了した場合は TRUE を返します。

ImqString creationTime ();

起こり得るエラーを示さずに、**creation time** を返します。

ImqBoolean currentDepth (MQLONG & depth);

current depth のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG currentDepth();

起こり得るエラーを示さずに、**current depth** を返します。

ImqBoolean defaultInputOpenOption (MQLONG & option);

default input open option のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG defaultInputOpenOption();

起こり得るエラーを示さずに、**default input open option** を返します。

ImqBoolean defaultPersistence (MQLONG & persistence);

default persistence のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG defaultPersistence();

起こり得るエラーを示さずに、**default persistence** を返します。

ImqBoolean defaultPriority (MQLONG & priority);

default priority のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG defaultPriority();

起こり得るエラーを示さずに、**default priority** を返します。

ImqBoolean defaultBind (MQLONG & bind);

default bind のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG defaultBind();

起こり得るエラーを示さずに、**default bind** を返します。

ImqBoolean definitionType (MQLONG & type);

definition type のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG definitionType();

起こり得るエラーを示さずに、**definition type** を返します。

ImqBoolean depthHighEvent (MQLONG & event);

depth high event の有効化状態のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG depthHighEvent();

起こり得るエラーを示さずに、**depth high event** の有効化状態を返します。

ImqBoolean depthHighLimit (MQLONG & limit);

depth high limit のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG depthHighLimit();

起こり得るエラーを示さずに、**depth high limit** 値を返します。

ImqBoolean depthLowEvent (MQLONG & event);

depth low event の有効化状態のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG depthLowEvent();

起こり得るエラーを示さずに、**depth low event** の有効化状態を返します。

ImqBoolean depthLowLimit (MQLONG & limit);

depth low limit のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG depthLowLimit();

起こり得るエラーを示さずに、**depth low limit** 値を返します。

ImqBoolean depthMaximumEvent (MQLONG & event);

depth maximum event の有効化状態のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG depthMaximumEvent();

起こり得るエラーを示さずに、**depth maximum event** の有効化状態を返します。

ImqDistributionList * distributionListReference() const ;

distribution list reference を返します。

void setDistributionListReference (ImqDistributionList & list);

distribution list reference を設定します。

void setDistributionListReference(ImqDistributionList * list = 0);

distribution list reference を設定します。

ImqBoolean distributionLists (MQLONG & support);

distribution lists 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG distributionLists();

起こり得るエラーを示さずに、**distribution lists** 値を返します。

ImqBoolean setDistributionLists(const MQLONG support);

distribution lists 値を設定します。正常に終了した場合は TRUE を返します。

ImqString dynamicQueueName() const ;

dynamic queue name のコピーを返します。

ImqBoolean setDynamicQueueName(const char * name);

dynamic queue name を設定します。**dynamic queue name** を設定できるのは、ImqObject の **open status** が FALSE の間だけです。正常に終了した場合は TRUE を返します。

ImqBoolean get (ImqMessage & msg, ImqGetMessageOptions & options);

指定された **options** を使用して、キューからメッセージを検索します。**options** に応じて、ImqObject の **open options** にいずれかの MQOO_INPUT_* 値 または MQOO_BROWSE 値が必ず含まれるようにする必要のある場合は、ImqObject の **openFor** メソッドが呼び出されます。**msg** オブジェクトに ImqCache の **automatic buffer** が指定されている場合は、そのバッファは検索されたメッセージに合わせて大きくなります。検索の前に、**msg** オブジェクトに対して **clearMessage** メソッドが呼び出されます。

このメソッドは、正常に終了した場合には TRUE を返します。

注: ImqObject の **reason code** が MQRC_TRUNCATED_MSG_FAILED である場合、この **reason code** が警告であっても、メソッド呼び出しの結果は FALSE になります。切り捨てられたメッセージが受け入れられる場合、ImqCache の **message length** は切り捨てられた長さを反映します。どちらの場合も、ImqMessage の **total message length** は、有効であったバイト数を示します。

ImqBoolean get (ImqMessage & msg);

デフォルトの読み取りメッセージ・オプションが使用されること以外は、上記のメソッドと同じです。

ImqBoolean get (ImqMessage & msg, ImqGetMessageOptions & options, const size_t buffer-size);

buffer size による指定変更ができる以外は、上記の 2 つのメソッドと同じです。**msg** オブジェクトが ImqCache の **automatic buffer** を使用している場合、メッセージの検索の前に **msg** オブジェクトに対して **resizeBuffer** メソッドが呼び出されます。このバッファは検索されたメッセージに合わせて大きくはなりません。

ImqBoolean get (ImqMessage & msg, const size_t buffer-size);

デフォルトの読み取りメッセージ・オプションが使用されること以外は、上記のメソッドと同じです。

ImqBoolean hardenGetBackout (MQLONG & harden);

harden get backout 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG hardenGetBackout();

起こり得るエラーを示さずに、**harden get backout** 値を返します。

ImqBoolean indexType(MQLONG & type);

index type のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG indexType();

起こり得るエラーを示さずに、**index type** を返します。

ImqBoolean inhibitGet (MQLONG & inhibit);

inhibit get 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG inhibitGet();
 起り得るエラーを示さずに、**inhibit get** 値を返します。

ImqBoolean setInhibitGet(const MQLONG *inhibit*);
inhibit get 値を設定します。正常に終了した場合は TRUE を返します。

ImqBoolean inhibitPut (MQLONG & *inhibit*);
inhibit put 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG inhibitPut();
 起り得るエラーを示さずに、**inhibit put** 値を返します。

ImqBoolean setInhibitPut(const MQLONG *inhibit*);
inhibit put 値を設定します。正常に終了した場合は TRUE を返します。

ImqBoolean initiationQueueName (ImqString & *name*);
initiation queue name のコピーを提供します。正常に終了した場合は TRUE を返します。

ImqString initiationQueueName ();
 起り得るエラーを示さずに、**initiation queue name** を返します。

ImqBoolean maximumDepth (MQLONG & *depth*);
maximum depth のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG maximumDepth();
 起り得るエラーを示さずに、**maximum depth** を返します。

ImqBoolean maximumMessageLength (MQLONG & *length*);
maximum message length のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG maximumMessageLength();
 起り得るエラーを示さずに、**maximum message length** を返します。

ImqBoolean messageDeliverySequence (MQLONG & *sequence*);
message delivery sequence のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG messageDeliverySequence();
 起り得るエラーを示さずに、**message delivery sequence** 値を返します。

ImqQueue * nextDistributedQueue() const ;
next distributed queue を返します。

ImqBoolean nonPersistentMessageClass (MQLONG & *monq*);
non persistent message class 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG nonPersistentMessageClass ();
 起り得るエラーを示さずに、**non persistent message class** 値を返します。

ImqBoolean openInputCount (MQLONG & *count*);
open input count のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG openInputCount();
 起り得るエラーを示さずに、**open input count** を返します。

ImqBoolean openOutputCount (MQLONG & *count*);
open output count のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG openOutputCount();
 起り得るエラーを示さずに、**open output count** を返します。

ImqQueue * previousDistributedQueue() const ;
previous distributed queue を返します。

ImqBoolean processName (ImqString & *name*);
process name のコピーを提供します。正常に終了した場合は TRUE を返します。

ImqString processName ();
 起り得るエラーを示さずに、**process name** を返します。

ImqBoolean put (ImqMessage & *msg*);
 デフォルトの書き込みメッセージ・オプションを使用して、キューにメッセージを配置します。
 ImqObject の **open options** に MQOO_OUTPUT 値が必ず含まれるようにする必要がある場合は、
 ImqObject の **openFor** メソッドを使用します。

このメソッドは、正常に終了した場合には TRUE を返します。

ImqBoolean put (ImqMessage & msg, ImqPutMessageOptions & pmo);

指定された *pmo* を使用して、キューにメッセージを配置します。ImqObject の **open options** に MQOO_OUTPUT が含まれ、また、(*pmo options* に MQPMO_PASS_IDENTITY_CONTEXT、MQPMO_PASS_ALL_CONTEXT、MQPMO_SET_IDENTITY_CONTEXT、または MQPMO_SET_ALL_CONTEXT のいずれかが含まれている場合には) 対応する MQOO_*_CONTEXT 値が必ず含まれるようにする必要がある場合は、ImqObject の **openFor** メソッドを使用します。

このメソッドは、正常に終了した場合には TRUE を返します。

注 : *pmo* に **context reference** が含まれている場合、必要に応じて、コンテキストを提供するために参照対象のオブジェクトがオープンされます。

ImqBoolean queueAccounting (MQLONG & acctq);

queue accounting 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG queueAccounting ();

起こり得るエラーを示さずに、queue accounting 値を返します。

ImqString queueManagerName () const ;

queue manager name を返します。

ImqBoolean setQueueManagerName(const char * name);

queue manager name を設定します。queue manager name を設定できるのは、ImqObject の **open status** が FALSE の間だけです。このメソッドは、正常に終了した場合には TRUE を返します。

ImqBoolean queueMonitoring (MQLONG & monq);

queue monitoring 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG queueMonitoring ();

起こり得るエラーを示さずに、queue monitoring 値を返します。

ImqBoolean queueStatistics (MQLONG & statq);

queue statistics 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG queueStatistics ();

起こり得るエラーを示さずに、queue statistics 値を返します。

ImqBoolean queueType (MQLONG & type);

queue type 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG queueType ();

起こり得るエラーを示さずに、queue type を返します。

ImqBoolean remoteQueueManagerName (ImqString & name);

remote queue manager name のコピーを提供します。正常に終了した場合は TRUE を返します。

ImqString remoteQueueManagerName ();

起こり得るエラーを示さずに、remote queue manager name を返します。

ImqBoolean remoteQueueName (ImqString & name);

remote queue name のコピーを提供します。正常に終了した場合は TRUE を返します。

ImqString remoteQueueName ();

起こり得るエラーを示さずに、remote queue name を返します。

ImqBoolean resolvedQueueManagerName(ImqString & name);

resolved queue manager name のコピーを提供します。正常に終了した場合は TRUE を返します。

注 : MQOO_RESOLVE_NAMES が ImqObject の **open options** に含まれていないと、このメソッドは正常に実行されません。

ImqString resolvedQueueManagerName ();

起こり得るエラーを示さずに、resolved queue manager name を返します。

ImqBoolean resolvedQueueName(ImqString & name);

resolved queue name のコピーを提供します。正常に終了した場合は TRUE を返します。

注 : MQOO_RESOLVE_NAMES が ImqObject の **open options** に含まれていないと、このメソッドは正常に実行されません。

ImqString resolvedQueueName();

起こり得るエラーを示さずに、**resolved queue name** を返します。

ImqBoolean retentionInterval (MQLONG & interval);

retention interval のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG retentionInterval();

起こり得るエラーを示さずに、**retention interval** を返します。

ImqBoolean scope (MQLONG & scope);

scope のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG scope();

起こり得るエラーを示さずに、**scope** を返します。

ImqBoolean serviceInterval (MQLONG & interval);

service interval のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG serviceInterval();

起こり得るエラーを示さずに、**service interval** を返します。

ImqBoolean serviceIntervalEvent (MQLONG & event);

service interval event の有効化状態のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG serviceIntervalEvent();

起こり得るエラーを示さずに、**service interval event** の有効化状態を返します。

ImqBoolean shareability (MQLONG & shareability);

shareability 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG shareability();

起こり得るエラーを示さずに、**shareability** 値を返します。

ImqBoolean storageClass(ImqString & class);

storage class のコピーを提供します。正常に終了した場合は TRUE を返します。

ImqString storageClass();

起こり得るエラーを示さずに、**storage class** を返します。

ImqBoolean transmissionQueueName (ImqString & name);

transmission queue name のコピーを提供します。正常に終了した場合は TRUE を返します。

ImqString transmissionQueueName ();

起こり得るエラーを示さずに、**transmission queue name** を返します。

ImqBoolean triggerControl (MQLONG & control);

trigger control 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG triggerControl();

起こり得るエラーを示さずに、**trigger control** 値を返します。

ImqBoolean setTriggerControl(const MQLONG control);

trigger control 値を設定します。正常に終了した場合は TRUE を返します。

ImqBoolean triggerData (ImqString & data);

trigger data のコピーを提供します。正常に終了した場合は TRUE を返します。

ImqString triggerData ();

起こり得るエラーを示さずに、**trigger data** のコピーを返します。

ImqBoolean setTriggerData(const char * data);

trigger data を設定します。正常に終了した場合は TRUE を返します。

ImqBoolean triggerDepth (MQLONG & depth);

trigger depth のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG triggerDepth();

起こり得るエラーを示さずに、**trigger depth** を返します。

ImqBoolean setTriggerDepth(const MQLONG depth);

trigger depth を設定します。正常に終了した場合は TRUE を返します。

ImqBoolean triggerMessagePriority (MQLONG & *priority*);
trigger message priority のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG triggerMessagePriority();
起こり得るエラーを示さずに、**trigger message priority** を返します。

ImqBoolean setTriggerMessagePriority(const MQLONG *priority*);
trigger message priority を設定します。正常に終了した場合は TRUE を返します。

ImqBoolean triggerType (MQLONG & *type*);
trigger type のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG triggerType();
起こり得るエラーを示さずに、**trigger type** を返します。

ImqBoolean setTriggerType(const MQLONG *type*);
trigger type を設定します。正常に終了した場合は TRUE を返します。

ImqBoolean usage (MQLONG & *usage*);
usage 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG usage();
起こり得るエラーを示さずに、**usage** 値を返します。

オブジェクト・メソッド (保護)

void setNextDistributedQueue(ImqQueue * *queue* = 0);
next distributed queue を設定します。

重要: この関数は、分散キュー・リストを損傷しないことが確かな場合に限り使用してください。

void setPreviousDistributedQueue(ImqQueue * *queue* = 0);
previous distributed queue を設定します。

重要: この関数は、分散キュー・リストを損傷しないことが確かな場合に限り使用してください。

理由コード

- MQRC_ATTRIBUTE_LOCKED
- MQRC_CONTEXT_OBJECT_NOT_VALID
- MQRC_CONTEXT_OPEN_ERROR
- MQRC_CURSOR_NOT_VALID
- MQRC_NO_BUFFER
- MQRC_REOPEN_EXCL_INPUT_ERROR
- MQRC_REOPEN_INQUIRE_ERROR
- MQRC_REOPEN_TEMPORARY_Q_ERROR
- (MQGET からの理由コード)
- (MQPUT からの理由コード)

ImqQueueManager C++ クラス

このクラスは、キュー・マネージャー (タイプ MQOT_Q_MGR の IBM MQ オブジェクト) をカプセル化します。

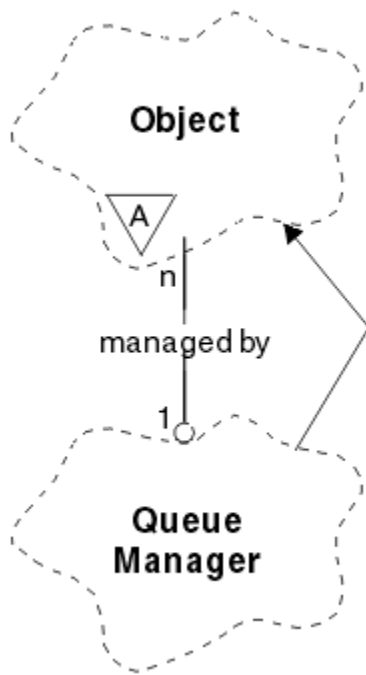


図 32. *ImqQueueManager* クラス

このクラスは、1801 ページの『[ImqQueueManager の相互参照](#)』にリストされている MQI 呼び出しと関連があります。リストされたすべてのメソッドがすべてのプラットフォームに適用できるわけではありません。詳しくは、[ALTER QMGR](#) を参照してください。

- [1870 ページの『クラス属性』](#)
- [1871 ページの『オブジェクトの属性』](#)
- [1876 ページの『コンストラクター』](#)
- [1876 ページの『デストラクター』](#)
- [1876 ページの『クラス・メソッド \(共有\)』](#)
- [1876 ページの『オブジェクト・メソッド \(共有\)』](#)
- [1885 ページの『オブジェクト・メソッド \(保護\)』](#)
- [1885 ページの『オブジェクト・データ \(保護\)』](#)
- [1885 ページの『理由コード』](#)

クラス属性

動作 (behavior)

暗黙接続と切断の振る舞いを制御します。

IMQ_EXPL_DISC_BACKOUT (0L)

disconnect メソッドを明示的に呼び出して、バックアウトを暗黙指定します。この属性を IMQ_EXPL_DISC_COMMIT と同時に指定することはできません。

IMQ_EXPL_DISC_COMMIT (1L)

disconnect メソッドを明示的に呼び出して、コミットを暗黙指定します (デフォルト)。この属性を IMQ_EXPL_DISC_BACKOUT と同時に指定することはできません。

IMQ_IMPL_CONN (2L)

暗黙接続を許可します (デフォルト)。

IMQ_IMPL_DISC_BACKOUT (0L)

disconnect メソッドを暗黙的に呼び出して、バックアウトを暗黙指定します。この呼び出しは、オブジェクトの消滅時に実行できます。この属性を IMQ_IMPL_DISC_COMMIT と同時に指定することはできません。

IMQ_IMPL_DISC_COMMIT (4L)

disconnect メソッドを暗黙的に呼び出して、コミットを暗黙指定します (デフォルト)。この呼び出しは、オブジェクトの消滅時に実行できます。この属性を IMQ_IMPL_DISC_BACKOUT と同時に指定することはできません。

IBM MQ V7.0 以降の場合、暗黙接続を使用する C++ アプリケーションは、クラス ImqQueueManager のオブジェクトの setBehavior() メソッドで指定される他のすべてのオプションとともに IMQ_IMPL_CONN を指定する必要があります。アプリケーションで setBehavior() メソッドを使用して動作オプションを明示的に設定しない場合には、以下のようになります。

```
ImqQueueManager_object.setBehavior(IMQ_IMPL_DISC_COMMIT)
```

MQ_IMPL_CONN がデフォルトで有効であるため、この変更による影響はありません。

アプリケーションで、次の例のように動作オプションを明示的に設定する場合には、影響があります。

```
ImqQueueManager_object.setBehavior(IMQ_IMPL_DISC_COMMIT)
```

この場合、次のように、setBehavior() メソッドに IMQ_IMPL_CONN を含めて、アプリケーションが暗黙接続を完了できるようにする必要があります。

```
ImqQueueManager_object.setBehavior(IMQ_IMPL_CONN | IMQ_IMPL_DISC_COMMIT)
```

オブジェクトの属性

accounting connections override

MQI accounting 値および queue accounting 値の設定をアプリケーションが指定変更できるようにします。この属性は読み取り専用です。

accounting interval

中間アカウンティング・レコードが書き込まれるまでの時間 (秒)。この属性は読み取り専用です。

activity recording

活動レポートの生成を制御します。この属性は読み取り専用です。

adopt new mca check

既にアクティブになっている MCA と同じ名前のインバウンド・チャネルが新たに検出されたときに、MCA を取り入れるかどうかを判別するために検査される エlement。この属性は読み取り専用です。

adopt new mca type

adopt new mca check パラメーターと一致するインバウンド・チャネル要求が新たに検出されたときに、特定のチャネル・タイプを持つ MCA の孤立インスタンスを自動的に再始動するかどうか。この属性は読み取り専用です。

authentication type

実行される認証のタイプを指示します。

authority event

許可イベントを制御します。この属性は読み取り専用です。

begin options

begin メソッドに適用されるオプション。初期値は MQBO_NONE です。

bridge event

IMS ブリッジ・イベントを生成するかどうか。この属性は読み取り専用です。

channel auto definition

チャネル自動定義値。この属性は読み取り専用です。

channel auto definition event

チャネル自動定義イベント値。この属性は読み取り専用です。

channel auto definition exit

チャネル自動定義出口の名前。この属性は読み取り専用です。

チャネル・イベント (channel event)

チャネル・イベントを生成するかどうか。この属性は読み取り専用です。

channel initiator adapters

IBM MQ 呼び出しを処理するために使用するアダプターのサブタスク数です。この属性は読み取り専用です。

channel initiator control

キュー・マネージャーが開始されたときにチャンネル開始プログラムを自動的に開始するかどうか。この属性は読み取り専用です。

channel initiator dispatchers

チャンネル・イニシエーターで使用するディスパッチャーの数。この属性は読み取り専用です。

channel initiator trace autostart

チャンネル開始プログラム・トレースを自動的に開始するかどうか。この属性は読み取り専用です。

channel initiator trace table size

チャンネル開始プログラムのトレース・データ・スペースのサイズ (MB)。この属性は読み取り専用です。

チャンネル・モニター

チャンネルに関するオンライン・モニター・データの収集を制御します。この属性は読み取り専用です。

channel reference

クライアント接続中に使用するための、チャンネル定義への参照。接続中、この属性をヌルに設定することはできますが、他の値に変更することはできません。初期値はヌルです。

チャンネル 統計

チャンネルの統計データの収集を制御します。この属性は読み取り専用です。

character set

コード化文字セットの ID (CCSID)。この属性は読み取り専用です。

cluster sender monitoring

自動的に定義されたクラスター送信側チャンネルのオンライン・モニター・データの収集を制御します。この属性は読み取り専用です。

cluster sender statistics

自動的に定義されたクラスター送信側チャンネルの統計データの収集を制御します。この属性は読み取り専用です。

cluster workload data

クラスター・ワークロード出口データ。この属性は読み取り専用です。

クラスター・ワークロード出口

クラスター・ワークロード出口名。この属性は読み取り専用です。

cluster workload length

クラスター・ワークロードの長さ。この属性は読み取り専用です。

cluster workload mru

クラスター・ワークロードの、最近使用されたチャンネルの値。この属性は読み取り専用です。

cluster workload use queue

クラスター・ワークロードの使用キューの値。この属性は読み取り専用です。

コマンド・イベント (command event)

コマンド・イベントを生成するかどうか。この属性は読み取り専用です。

command input queue name

システム・コマンド入力キュー名です。この属性は読み取り専用です。

コマンド・レベル

キュー・マネージャーによってサポートされているコマンド・レベルです。この属性は読み取り専用です。

command server control

キュー・マネージャーが開始されたときにコマンド・サーバーを自動的に開始するかどうか。この属性は読み取り専用です。

connect options

connect メソッドに適用されるオプションです。初期値は MQCNO_NONE です。プラットフォームに応じて、以下の値を追加で使えます。

- MQCNO_STANDARD_BINDING
- MQCNO_FASTPATH_BINDING
- MQCNO_HANDLE_SHARE_NONE
- MQCNO_HANDLE_SHARE_BLOCK
- MQCNO_HANDLE_SHARE_NO_BLOCK
- MQCNO_SERIALIZE_CONN_TAG_Q_MGR
- MQCNO_SERIALIZE_CONN_TAG_QSG
- MQCNO_RESTRICT_CONN_TAG_Q_MGR
- MQCNO_RESTRICT_CONN_TAG_QSG

connection id

MQ でアプリケーションを確実に特定できるようにする固有 ID。

connection status

キュー・マネージャーに接続されている場合は、TRUE。この属性は読み取り専用です。

connection tag

接続に関連付けられるタグ。この属性を設定できるのは、接続していない時だけです。初期値はヌルです。

cryptographic hardware

暗号ハードウェア用の構成の詳細。MQ MQI クライアント接続用。

dead-letter queue name

送達不能キューの名前です。この属性は読み取り専用です。

default transmission queue name

デフォルト伝送キュー名。この属性は読み取り専用です。

distribution lists

配布リストをサポートするためのキュー・マネージャーの機能。

dns group

キュー共有グループのインバウンド伝送を処理する TCP リスナーが、Workload Manager for Dynamic Domain Name Services サポートの使用時に参加する必要があるグループの名前。この属性は読み取り専用です。

dns wlm

キュー共有グループのインバウンド伝送を処理する TCP リスナーを Workload Manager for Dynamic Domain Name Services に登録するかどうか。この属性は読み取り専用です。

first authentication record

クラス ImqAuthenticationRecord の 1 つまたは複数のオブジェクトのうち、特定の順序ではなく、ImqAuthenticationRecord の connection reference がこのオブジェクトをアドレッシングする順序のうち最初のもの。MQ MQI クライアント接続用。

first managed object

クラス ImqObject の 1 つまたは複数のオブジェクトのうち、特定の順序ではなく、ImqObject の connection reference がこのオブジェクトをアドレッシングする順序のうち最初のもの。初期値はゼロです。

inhibit event

禁止イベントを制御します。この属性は読み取り専用です。

ip address version

チャネル接続に使用する IP プロトコル (IPv4 または IPv6)。この属性は読み取り専用です。

鍵リポジトリ (key repository)

キーおよび証明書が保管されているキー・データベース・ファイルの場所。IBM MQ MQI client 接続用。

key reset count

秘密鍵が再折衝される前に TLS 会話内で暗号化されずに送受信されるバイト数。この属性は、MQCONN を使用したクライアント接続に対してのみ適用されます。『[ssl key reset count](#)』も参照してください。

listener timer

APPC または TCP/IP の障害が発生した場合に IBM MQ がリスナーの再始動を繰り返し試みる時間間隔 (秒)。この属性は読み取り専用です。

local event

ローカル・イベントを制御します。この属性は読み取り専用です。

logger event

回復ログ・イベントを生成するかどうかを制御します。この属性は読み取り専用です。

lu group name

キュー共有グループのインバウンド伝送を処理する LU 6.2 リスナーが使用する総称 LU 名。この属性は読み取り専用です。

lu name

アウトバウンド LU 6.2 伝送に使用する LU の名前。この属性は読み取り専用です。

lu62 arm suffix

SYS1.PARMLIB メンバー APPCPMxx のサフィックス。このチャンネル開始プログラムとして LUADD を指定します。この属性は読み取り専用です。

lu62 channels

現行チャンネルにすることができるチャンネルの最大数、または接続可能なクライアントの最大数 (LU 6.2 伝送プロトコルを使用するもの)。この属性は読み取り専用です。

maximum active channels

任意の時点でアクティブなチャンネルの最大数。この属性は読み取り専用です。

maximum channels

現行チャンネルにすることが可能なチャンネルの最大数 (クライアントが接続されているサーバー接続チャンネルを含む)。この属性は読み取り専用です。

maximum handles

ハンドルの最大数です。この属性は読み取り専用です。

最大メッセージ長

このキュー・マネージャーによって管理される任意のキュー上の任意のメッセージの可能な最大長です。この属性は読み取り専用です。

maximum priority

最高のメッセージ優先順位。この属性は読み取り専用です。

maximum uncommitted messages

1つの作業単位内のコミットされていないメッセージの最大数。この属性は読み取り専用です。

mqi accounting

MQI データに関するアカウントリング情報の収集を制御します。この属性は読み取り専用です。

mqi statistics

キュー・マネージャーに関する統計モニター情報の収集を制御します。この属性は読み取り専用です。

outbound port maximum

出力チャンネルのバインド時に使用されるポート番号範囲の中で最も大きい番号。この属性は読み取り専用です。

outbound port minimum

出力チャンネルのバインド時に使用されるポート番号範囲の中で最も小さい番号。この属性は読み取り専用です。

パスワード

ユーザー ID に関連付けられたパスワード。

パフォーマンス・イベント (performance event)

パフォーマンス・イベントを制御します。この属性は読み取り専用です。

platform

キュー・マネージャーがあるプラットフォーム。この属性は読み取り専用です。

queue accounting

キューに関するアカウントリング情報の収集を制御します。この属性は読み取り専用です。

queue monitoring

キューに関するオンライン・モニター・データの収集を制御します。この属性は読み取り専用です。

queue statistics

キューに関する統計データの収集を制御します。この属性は読み取り専用です。

receive timeout

TCP/IP メッセージ・チャネルが、非アクティブ状態に戻るまでにパートナーからのデータ (ハートビートを含む) の受信を待機するおおよその時間。この属性は読み取り専用です。

receive timeout minimum

TCP/IP チャネルが、非アクティブ状態に戻るまでにパートナーからのデータ (ハートビートを含む) の受信を待機する最小時間。この属性は読み取り専用です。

receive timeout type

receive timeout に適用される修飾子。この属性は読み取り専用です。

remote event

リモート・イベントを制御します。この属性は読み取り専用です。

repository name

リポジトリの名前。この属性は読み取り専用です。

repository namelist

リポジトリ名前リストの名前。この属性は読み取り専用です。

shared queue manager name

共有キューで ObjectQMGrName がキュー共有グループの別のキュー・マネージャーである場合に、ローカル・キュー・マネージャーで共有キューをオープンするように、共有キューの MQOPEN を解決するかどうか。この属性は読み取り専用です。

ssl event

SSL イベントを生成するかどうか。この属性は読み取り専用です。

SSLFIPS 必須

ソフトウェア IBM MQ で暗号処理を実行する場合、認定されたアルゴリズム FIPS のみを使用すべきかどうか。この属性は読み取り専用です。

ssl key reset count

秘密鍵が再折衝される前に SSL 会話内で暗号化されずに送受信されるバイト数。この属性は読み取り専用です。

start-stop event

開始/終了イベントを制御します。この属性は読み取り専用です。

statistics interval

統計モニター・データをモニター・キューに入れる頻度。この属性は読み取り専用です。

syncpoint availability

同期点参加の可/不可。この属性は読み取り専用です。

注: IBM i プラットフォームでは、キュー・マネージャーによって整合されたグローバル作業単位はサポートされません。IBM i  _Rcommit および _Rback ネイティブ・システム呼び出しを使用すれば、IBM i によって外部的に整合された作業単位をプログラミングできます。このタイプの作業単位を開始するには、STRCMTCTL コマンドを使用して、ジョブ・レベルのコミットメント制御を行いながら、IBM MQ アプリケーションを開始します。詳しくは、[IBM i 外部同期点マネージャーへのインターフェース](#)を参照してください。IBM i プラットフォームでは、キュー・マネージャーによって整合されたローカル作業単位に対して backout および commit がサポートされています。

tcp channels

現行チャネルにすることができるチャネルの最大数、または接続可能なクライアントの最大数 (TCP/IP 伝送プロトコルを使用するもの)。この属性は読み取り専用です。

tcp keepalive

接続先がまだ使用可能であるかどうかを検査するために TCP KEEPALIVE 機能を使用するかどうか。この属性は読み取り専用です。

tcp name

使用する単一またはデフォルトの TCP/IP システムの名前。tcp stack type の値によって異なります。
この属性は読み取り専用です。

tcp stack type

チャネル開始プログラムが、tcp name で指定された TCP/IP アドレス・スペースのみを使用できるようにするのか、任意に選択された TCP/IP アドレスにバインドできるようにするのか。この属性は読み取り専用です。

trace route recording

経路トレース情報の記録を制御します。この属性は読み取り専用です。

trigger interval

トリガー間隔。この属性は読み取り専用です。

ユーザー ID

AIX and Linux プラットフォームでは、アプリケーションの実ユーザー ID。Windows プラットフォームでは、アプリケーションのユーザー ID。

コンストラクター**ImqQueueManager();**

デフォルトのコンストラクター。

ImqQueueManager(const ImqQueueManager & manager);

コピー・コンストラクター。connection status は FALSE です。

ImqQueueManager(const char * name);

ImqObject 名を *name* に設定します。

デストラクター

ImqQueueManager オブジェクトは、破棄されると自動的に切断されます。

クラス・メソッド (共有)**static MQLONG behavior();**

behavior を返します。

void setBehavior(const MQLONG behavior = 0);

behavior を設定します。

オブジェクト・メソッド (共有)**void operator = (const ImqQueueManager & mgr);**

必要な場合は切断し、インスタンス・データを *mgr* からコピーします。接続状態は FALSE となります。

ImqBoolean accountingConnOverride (MQLONG & statint);

accounting connections override 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG accountingConnOverride ();

起こり得るエラーを示さずに、accounting connections override 値を返します。

ImqBoolean accountingInterval (MQLONG & statint);

accounting interval 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG accountingInterval ();

起こり得るエラーを示さずに、accounting interval 値を返します。

ImqBoolean activityRecording (MQLONG & rec);

activity recording 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG activityRecording ();

起こり得るエラーを示さずに、activity recording 値を返します。

ImqBoolean adoptNewMCACheck (MQLONG & check);

adopt new MCA check 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG adoptNewMCACheck ();

起こり得るエラーを示さずに、adopt new MCA check 値を返します。

ImqBoolean adoptNewMCAType (MQLONG & type);

adopt new MCA type のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG adoptNewMCAType ();

起こり得るエラーを示さずに、adopt new MCA type を返します。

QLONG authenticationType () const;

authentication type を返します。

void setAuthenticationType (const MQLONG type = MQCSP_AUTH_NONE);

authentication type を設定します。

ImqBoolean authorityEvent(MQLONG & event);

authority event の有効化状態のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG authorityEvent();

起こり得るエラーを示さずに、authority event の有効化状態を返します。

ImqBoolean backout();

コミットされていない変更内容をバックアウトします。正常に終了した場合は TRUE を返します。

ImqBoolean begin();

1つの作業単位を始めます。開始オプションは、このメソッドの動作に影響します。正常に終了した場合は TRUE を返しますが、基礎となる MQBEGIN 呼び出しが MQRC_NO_EXTERNAL_PARTICIPANTS または MQRC_PARTICIPANT_NOT_AVAILABLE (どちらも MQCC_WARNING に関連している) を返した場合にも TRUE を返します。

MQLONG beginOptions() const ;

begin options を返します。

void setBeginOptions(const MQLONG options = MQBO_NONE);

begin options を設定します。

ImqBoolean bridgeEvent (MQLONG & event);

bridge event 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG bridgeEvent ();

起こり得るエラーを示さずに、bridge event 値を返します。

ImqBoolean channelAutoDefinition(MQLONG & value);

channel auto definition 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG channelAutoDefinition();

起こり得るエラーを示さずに、channel auto definition 値を返します。

ImqBoolean channelAutoDefinitionEvent(MQLONG & value);

channel auto definition event 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG channelAutoDefinitionEvent();

起こり得るエラーを示さずに、channel auto definition event 値を返します。

ImqBoolean channelAutoDefinitionExit(ImqString & name);

channel auto definition exit 名のコピーを提供します。正常に終了した場合は TRUE を返します。

ImqString channelAutoDefinitionExit();

起こり得るエラーを示さずに、channel auto definition exit 名を返します。

ImqBoolean channelEvent (MQLONG & event);

channel event 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG channelEvent();

起こり得るエラーを示さずに、channel event 値を返します。

MQLONG channelInitiatorAdapters ();

起こり得るエラーを示さずに、channel initiator adapters 値を返します。

ImqBoolean channelInitiatorAdapters (MQLONG & adapters);

channel initiator adapters 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG channelInitiatorControl ();

起こり得るエラーを示さずに、channel initiator startup 値を返します。

ImqBoolean channelInitiatorControl (MQLONG & init);

channel initiator control startup 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG channelInitiatorDispatchers ();

起こり得るエラーを示さずに、channel initiator dispatchers 値を返します。

ImqBoolean channelInitiatorDispatchers (MQLONG & dispatchers);

channel initiator dispatchers 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG channelInitiatorTraceAutoStart ();

起こり得るエラーを示さずに、channel initiator trace auto start 値を返します。

ImqBoolean channelInitiatorTraceAutoStart (MQLONG & auto);

channel initiator trace auto start 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG channelInitiatorTraceTableSize ();

起こり得るエラーを示さずに、channel initiator trace table size 値を返します。

ImqBoolean channelInitiatorTraceTableSize (MQLONG & size);

channel initiator trace table size 値のコピーを提供します。正常に終了した場合は TRUE を返します。

ImqBoolean channelMonitoring (MQLONG & monchl);

channel monitoring value 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG channelMonitoring ();

起こり得るエラーを示さずに、channel monitoring 値を返します。

ImqBoolean channelReference(ImqChannel * & pchannel);

channel reference のコピーを提供します。channel reference が無効の場合、*pchannel* はヌルに設定されます。このメソッドは、正常に終了した場合には TRUE を返します。

ImqChannel * channelReference();

起こり得るエラーを示さずに、channel reference を返します。

ImqBoolean setChannelReference(ImqChannel & channel);

channel reference を設定します。このメソッドは、正常に終了した場合には TRUE を返します。

ImqBoolean setChannelReference(ImqChannel * channel = 0);

channel reference を設定またはリセットします。このメソッドは、正常に終了した場合には TRUE を返します。

ImqBoolean channelStatistics (MQLONG & statchl);

channel statistics 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG channelStatistics ();

起こり得るエラーを示さずに、channel statistics 値を返します。

ImqBoolean characterSet(MQLONG & ccscid);

character set のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG characterSet();

起こり得るエラーを示さずに、character set のコピーを返します。

MQLONG clientSslKeyResetCount () const;

クライアント接続で使用する SSL key reset count 値を返します。

void setClientSslKeyResetCount(const MQLONG count);

クライアント接続で使用する SSL key reset count を設定します。

ImqBoolean clusterSenderMonitoring (MQLONG & monacsl);

cluster sender monitoring のデフォルト値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG clusterSenderMonitoring ();

起こり得るエラーを示さずに、cluster sender monitoring のデフォルト値を返します。

ImqBoolean clusterSenderStatistics (MQLONG & statacls);

cluster sender statistics 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG clusterSenderStatistics ();

起こり得るエラーを示さずに、cluster sender statistics 値を返します。

ImqBoolean clusterWorkloadData(ImqString & data);

cluster workload exit data のコピーを提供します。正常に終了した場合は TRUE を返します。

ImqString clusterWorkloadData();

起こり得るエラーを示さずに、cluster workload exit data を返します。

ImqBoolean clusterWorkloadExit(ImqString & name);

cluster workload exit name のコピーを提供します。正常に終了した場合は TRUE を返します。

ImqString clusterWorkloadExit();

起こり得るエラーを示さずに、cluster workload exit name を返します。

ImqBoolean clusterWorkloadLength(MQLONG & length);

cluster workload length のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG clusterWorkloadLength();

起こり得るエラーを示さずに、cluster workload length を返します。

ImqBoolean clusterWorkLoadMRU (MQLONG & mru);

cluster workload most recently used channels 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG clusterWorkLoadMRU ();

起こり得るエラーを示さずに、cluster workload most recently used channels 値を返します。

ImqBoolean clusterWorkLoadUseQ (MQLONG & useq);

cluster workload use queue 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG clusterWorkLoadUseQ ();

起こり得るエラーを示さずに、cluster workload use queue 値を返します。

ImqBoolean commandEvent (MQLONG & event);

command event 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG commandEvent ();

起こり得るエラーを示さずに、command event 値を返します。

ImqBoolean commandInputQueueName(ImqString & name);

command input queue name のコピーを提供します。正常に終了した場合は TRUE を返します。

ImqString commandInputQueueName();

起こり得るエラーを示さずに、command input queue name を返します。

ImqBoolean commandLevel(MQLONG & level);

command level のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG commandLevel();

起こり得るエラーを示さずに、command level を返します。

MQLONG commandServerControl ();

起こり得るエラーを示さずに、command server startup 値を返します。

ImqBoolean commandServerControl (MQLONG & server);

command server control startup 値のコピーを提供します。正常に終了した場合は TRUE を返します。

ImqBoolean commit();

コミットされていない変更内容をコミットします。正常に終了した場合は TRUE を返します。

ImqBoolean connect();

与えられた ImqObject の name をもつキュー・マネージャーへ接続します。デフォルトはローカル・キュー・マネージャーです。特定のキュー・マネージャーに接続したい場合は、接続前に ImqObject の setName メソッドを使用してください。channel reference がある場合は、それが MQCD の MQCONN によってチャンネル定義に関する情報を渡すために使用されることになります。MQCD の ChannelType は MQCHT_CLNTCONN に設定されます。channel reference 情報が意味を持つのはクライアント接続に関してだけなので、サーバー接続の場合は無視されます。connect options は、このメ

ソッドの動作に影響します。このメソッドは正常に終了した場合は `connection status` を `TRUE` に設定します。また、このメソッドは、新規の接続状況を返します。

最初の認証レコードがある場合には、認証レコードのチェーンを使用して、セキュア・クライアント・チャンネル用のデジタル証明書を認証します。

同一のキュー・マネージャーに複数の `ImqQueueManager` オブジェクトを接続できます。それらのすべてのオブジェクトは同じ `MQHCONN` 接続ハンドルを使用し、スレッドと関連した接続の `UOW` の機能を共有します。接続する最初の `ImqQueueManager` は `MQHCONN` ハンドルを取得します。切断する最後の `ImqQueueManager` は `MQDISC` を実行します。

マルチスレッド・プログラムの場合、各スレッドに対して別個の `ImqQueueManager` オブジェクトを使用することが推奨されます。

`ImqBinary connectionId ( ) const ;`

`connection ID` を返します。

`ImqBinary connectionTag ( ) const ;`

`connection tag` を返します。

`ImqBoolean setConnectionTag ( const MQBYTE128 tag = 0 );`

`connection tag` を設定します。`tag` がゼロの場合、`connection tag` はクリアされます。このメソッドは、正常に終了した場合には `TRUE` を返します。

`ImqBoolean setConnectionTag ( const ImqBinary & tag );`

`connection tag` を設定します。`tag` の `data length` は、ゼロ (`connection tag` をクリアする場合) か、`MQ_CONN_TAG_LENGTH` のどちらかでなければなりません。このメソッドは、正常に終了した場合には `TRUE` を返します。

`MQLONG connectOptions( ) const ;`

`connect options` を返します。

`void setConnectOptions( const MQLONG options = MQCNO_NONE );`

`connect options` を設定します。

`ImqBoolean connectionStatus( ) const ;`

`connection status` を返します。

`ImqString cryptographicHardware ( );`

`cryptographic hardware` を返します。

`ImqBoolean setCryptographicHardware ( const char * hardware = 0 );`

`cryptographic hardware` を設定します。このメソッドは、正常に終了した場合には `TRUE` を返します。

`ImqBoolean deadLetterQueueName( ImqString & name );`

`dead-letter queue name` のコピーを提供します。正常に終了した場合は `TRUE` を返します。

`ImqString deadLetterQueueName( );`

起り得るエラーを示さずに、`dead-letter queue name` のコピーを返します。

`ImqBoolean defaultTransmissionQueueName( ImqString & name );`

`default transmission queue name` のコピーを提供します。正常に終了した場合は `TRUE` を返します。

`ImqString defaultTransmissionQueueName( );`

起り得るエラーを示さずに、`default transmission queue name` を返します。

`ImqBoolean disconnect( );`

キュー・マネージャーから切断し、`connection status` を `FALSE` に設定します。このオブジェクトと関連付けられている `ImqProcess` オブジェクトおよび `ImqQueue` オブジェクトをすべてプログラムの切断前にクローズし、それぞれの `connection reference` を外します。同一のキュー・マネージャーに複数の `ImqQueueManager` オブジェクトが接続されている場合、最後に切断されるオブジェクトだけが物理的な切断を実行します。それ以外のオブジェクトは論理的な切断を行います。コミットされていない変更内容は、物理的な切断時のみコミットされます。

このメソッドは、正常に終了した場合には `TRUE` を返します。これは、既存の接続がない場合に呼び出され、戻りコードも `TRUE` になります。

`ImqBoolean distributionLists( MQLONG & support );`

`distribution lists` 値のコピーを提供します。正常に終了した場合は `TRUE` を返します。

MQLONG distributionLists();

起こり得るエラーを示さずに、distribution lists 値を返します。

ImqBoolean dnsGroup (ImqString & group);

DNS group name のコピーを提供します。正常に終了した場合は TRUE を返します。

ImqString dnsGroup ();

起こり得るエラーを示さずに、DNS group name を返します。

ImqBoolean dnsWlm (MQLONG & wlm);

DNS WLM 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG dnsWlm ();

起こり得るエラーを示さずに、DNS WLM 値を返します。

ImqAuthenticationRecord * firstAuthenticationRecord () const ;

first authentication record を返します。

void setFirstAuthenticationRecord (const ImqAuthenticationRecord * air = 0);

first authentication record を設定します。

ImqObject * firstManagedObject() const ;

first managed object を返します。

ImqBoolean inhibitEvent(MQLONG & event);

inhibit event の有効化状態のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG inhibitEvent();

起こり得るエラーを示さずに、inhibit event の有効化状態を返します。

ImqBoolean ipAddressVersion (MQLONG & version);

IP address version 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG ipAddressVersion ();

起こり得るエラーを示さずに、IP address version 値を返します。

ImqBoolean keepAlive (MQLONG & keepalive);

keep alive 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG keepAlive ();

起こり得るエラーを示さずに、keep alive 値を返します。

ImqString keyRepository ();

key repository を返します。

ImqBoolean setKeyRepository (const char * repository = 0);

key repository を設定します。正常に終了した場合は TRUE を返します。

ImqBoolean listenerTimer (MQLONG & timer);

listener timer 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG listenerTimer ();

起こり得るエラーを示さずに、listener timer 値を返します。

ImqBoolean localEvent(MQLONG & event);

local event の有効化状態のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG localEvent();

起こり得るエラーを示さずに、local event の有効化状態を返します。

ImqBoolean loggerEvent (MQLONG & count);

logger event 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG loggerEvent ();

起こり得るエラーを示さずに、logger event 値を返します。

ImqBoolean luGroupName (ImqString & name);

LU group name のコピーを提供します。正常に終了した場合は TRUE を返します。

ImqString luGroupName ();

起こり得るエラーを示さずに、LU group name を返します。

ImqBoolean lu62ARMSuffix (ImqString & suffix);

LU62 ARM suffix のコピーを提供します。正常に終了した場合は TRUE を返します。

ImqString lu62ARMSuffix ();

起こり得るエラーを示さずに、LU62 ARM suffix を返します。

ImqBoolean luName (ImqString & name);

LU name のコピーを提供します。正常に終了した場合は TRUE を返します。

ImqString luName ();

起こり得るエラーを示さずに、LU name を返します。

ImqBoolean maximumActiveChannels (MQLONG & channels);

maximum active channels 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG maximumActiveChannels ();

起こり得るエラーを示さずに、maximum active channels 値を返します。

ImqBoolean maximumCurrentChannels (MQLONG & channels);

maximum current channels 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG maximumCurrentChannels ();

起こり得るエラーを示さずに、maximum current channels 値を返します。

ImqBoolean maximumHandles(MQLONG & number);

maximum handles のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG maximumHandles();

起こり得るエラーを示さずに、maximum handles を返します。

ImqBoolean maximumLu62Channels (MQLONG & channels);

maximum LU62 channels 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG maximumLu62Channels ();

起こり得るエラーを示さずに、maximum LU62 channels 値を返します。

ImqBoolean maximumMessageLength(MQLONG & length);

maximum message length のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG maximumMessageLength();

起こり得るエラーを示さずに、maximum message length を返します。

ImqBoolean maximumPriority(MQLONG & priority);

maximum priority のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG maximumPriority();

起こり得るエラーを示さずに、maximum priority のコピーを返します。

ImqBoolean maximumTcpChannels (MQLONG & channels);

maximum TCP channels 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG maximumTcpChannels ();

起こり得るエラーを示さずに、maximum TCP channels 値を返します。

ImqBoolean maximumUncommittedMessages(MQLONG & number);

maximum uncommitted messages のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG maximumUncommittedMessages();

起こり得るエラーを示さずに、maximum uncommitted messages を返します。

ImqBoolean mqiAccounting (MQLONG & statint);

MQI accounting 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG mqiAccounting ();

起こり得るエラーを示さずに、MQI accounting 値を返します。

ImqBoolean mqiStatistics (MQLONG & statmqi);

MQI statistics 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG mqiStatistics ();

起こり得るエラーを示さずに、MQI statistics 値を返します。

ImqBoolean outboundPortMax (MQLONG & max);

maximum outbound port 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG outboundPortMax ();

起こり得るエラーを示さずに、maximum outbound port 値を返します。

ImqBoolean outboundPortMin (MQLONG & min);
minimum outbound port 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG outboundPortMin ();
起こり得るエラーを示さずに、minimum outbound port 値を返します。

ImqBinary password () const;
クライアント接続で使用される password を返します。

ImqBoolean setPassword (const ImqString & password);
クライアント接続で使用される password を設定します。

ImqBoolean setPassword (const char * = 0 password);
クライアント接続で使用される password を設定します。

ImqBoolean setPassword (const ImqBinary & password);
クライアント接続で使用される password を設定します。

ImqBoolean performanceEvent(MQLONG & event);
performance event の有効化状態のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG performanceEvent();
起こり得るエラーを示さずに、performance event の有効化状態を返します。

ImqBoolean platform(MQLONG & platform);
platform のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG platform();
起こり得るエラーを示さずに、platform を返します。

ImqBoolean queueAccounting (MQLONG & acctq);
queue accounting 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG queueAccounting ();
起こり得るエラーを示さずに、queue accounting 値を返します。

ImqBoolean queueMonitoring (MQLONG & monq);
queue monitoring 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG queueMonitoring ();
起こり得るエラーを示さずに、queue monitoring 値を返します。

ImqBoolean queueStatistics (MQLONG & statq);
queue statistics 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG queueStatistics ();
起こり得るエラーを示さずに、queue statistics 値を返します。

ImqBoolean receiveTimeout (MQLONG & timeout);
receive timeout 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG receiveTimeout ();
起こり得るエラーを示さずに、receive timeout 値を返します。

ImqBoolean receiveTimeoutMin (MQLONG & min);
minimum receive timeout 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG receiveTimeoutMin ();
起こり得るエラーを示さずに、minimum receive timeout 値を返します。

ImqBoolean receiveTimeoutType (MQLONG & type);
receive timeout type のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG receiveTimeoutType ();
起こり得るエラーを示さずに、receive timeout type を返します。

ImqBoolean remoteEvent(MQLONG & event);
remote event の有効化状態のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG remoteEvent();
起こり得るエラーを示さずに、remote event の有効化状態を返します。

ImqBoolean repositoryName(ImqString & name);
repository name のコピーを提供します。正常に終了した場合は TRUE を返します。

ImqString repositoryName();

起こり得るエラーを示さずに、repository name を返します。

ImqBoolean repositoryNamelistName(ImqString & name);

repository namelist name のコピーを提供します。正常に終了した場合は TRUE を返します。

ImqString repositoryNamelistName();

起こり得るエラーを示さずに、repository namelist name のコピーを返します。

ImqBoolean sharedQueueQueueManagerName (MQLONG & name);

shared queue queue manager name 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG sharedQueueQueueManagerName ();

起こり得るエラーを示さずに、shared queue queue manager name 値を返します。

ImqBoolean sslEvent (MQLONG & event);

SSL event 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG sslEvent ();

起こり得るエラーを示さずに、SSL event 値を返します。

ImqBoolean sslFips (MQLONG & sslfips);

値 **SSLFIPS** のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG sslFips ();

SSLFIPS の値を、エラーの可能性を示す表示なしに返します。

ImqBoolean sslKeyResetCount (MQLONG & count);

SSL key reset count 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG sslKeyResetCount ();

起こり得るエラーを示さずに、SSL key reset count 値を返します。

ImqBoolean startStopEvent(MQLONG & event);

start-stop event の有効化状態のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG startStopEvent();

起こり得るエラーを示さずに、start-stop event の有効化状態を返します。

ImqBoolean statisticsInterval (MQLONG & statint);

statistics interval 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG statisticsInterval ();

起こり得るエラーを示さずに、statistics interval 値を返します。

ImqBoolean syncPointAvailability(MQLONG & sync);

syncpoint availability 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG syncPointAvailability();

起こり得るエラーを示さずに、syncpoint availability 値のコピーを返します。

ImqBoolean tcpName (ImqString & name);

TCP system name のコピーを提供します。正常に終了した場合は TRUE を返します。

ImqString tcpName ();

起こり得るエラーを示さずに、TCP system name を返します。

ImqBoolean tcpStackType (MQLONG & type);

TCP stack type のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG tcpStackType ();

起こり得るエラーを示さずに、TCP stack type を返します。

ImqBoolean traceRouteRecording (MQLONG & routerec);

trace route recording 値のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG traceRouteRecording ();

起こり得るエラーを示さずに、trace route recording 値を返します。

ImqBoolean triggerInterval(MQLONG & interval);

trigger interval のコピーを提供します。正常に終了した場合は TRUE を返します。

MQLONG triggerInterval();

起こり得るエラーを示さずに、trigger interval を返します。

ImqBinary userId () const;

クライアント接続で使用する user ID を返します。

ImqBoolean setUserId (const ImqString & id);

クライアント接続で使用する user ID を設定します。

ImqBoolean setUserId (const char * = 0 id);

クライアント接続で使用する user ID を設定します。

ImqBoolean setUserId (const ImqBinary & id);

クライアント接続で使用する user ID を設定します。

オブジェクト・メソッド (保護)

void setFirstManagedObject (const ImqObject * object = 0);

first managed object を設定します。

オブジェクト・データ (保護)

MQHCONN ohconn

IBM MQ 接続ハンドル (connection status が TRUE の場合にのみ有効)。

理由コード

- MQRC_ATTRIBUTE_LOCKED
- MQRC_ENVIRONMENT_ERROR
- MQRC_FUNCTION_NOT_SUPPORTED
- MQRC_REFERENCE_ERROR
- (MQBACK の理由コード)
- (MQBEGIN の理由コード)
- (MQCMIT の理由コード)
- (MQCONN の理由コード)
- (MQCONNX の理由コード)
- (MQDISC の理由コード)
- (MQCONN の理由コード)

ImqReferenceHeader C++ クラス

このクラスは MQRMH データ構造体の機能をカプセル化します。

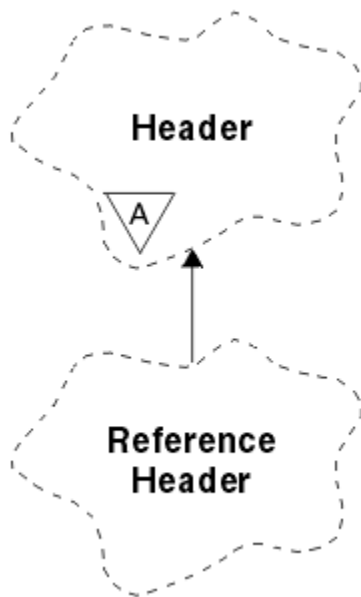


図 33. *ImqReferenceHeader* クラス

このクラスは、[1805 ページの『ImqReferenceHeader の相互参照』](#)にリストされている MQI 呼び出しと関連があります。

- [1886 ページの『オブジェクトの属性』](#)
- [1887 ページの『コンストラクター』](#)
- [1887 ページの『多重定義された ImqItem メソッド』](#)
- [1887 ページの『オブジェクト・メソッド \(共有\)』](#)
- [1888 ページの『オブジェクト・データ \(保護\)』](#)
- [1888 ページの『理由コード』](#)

オブジェクトの属性

destination environment

宛先の環境。初期値はヌル・ストリングです。

destination name

データ宛先の名前。初期値はヌル・ストリングです。

instance id

インスタンスの ID。長さ MQ_OBJECT_INSTANCE_ID_LENGTH の 2 進値 (MQBYTE24)。初期値は MQOII_NONE です。

logical length

このヘッダーに続くメッセージ・データの論理上の、つまり使用する予定の長さです。初期値はゼロです。

logical offset

後続のメッセージ・データの論理オフセットであり、最終宛先で、全体としてデータのコンテキストで解釈されます。初期値はゼロです。

logical offset 2

logical offset への高位拡張。初期値はゼロです。

reference type

参照タイプ。初期値はヌル・ストリングです。

source environment

発信側の環境。初期値はヌル・ストリングです。

source name

データ送信側の名前。初期値はヌル・ストリングです。

コンストラクター

ImqReferenceHeader();

デフォルトのコンストラクター。

ImqReferenceHeader(const ImqReferenceHeader & header);

コピー・コンストラクター。

多重定義された ImqItem メソッド

virtual ImqBoolean copyOut (ImqMessage & msg);

MQRMH データ構造体をメッセージ・バッファの始めに挿入し、既存のメッセージ・データを後ろにずらし、msg format を MQFMT_REF_MSG_HEADER に設定します。

詳細については、[1833 ページの『ImqHeader C++ クラス』](#)の ImqHeader クラス・メソッドの説明を参照してください。

virtual ImqBoolean pasteIn (ImqMessage & msg);

メッセージ・バッファから MQRMH データ構造体を読み取ります。

正常に実行されるためには、ImqMessage の format が MQFMT_REF_MSG_HEADER でなければなりません。

詳細については、[1833 ページの『ImqHeader C++ クラス』](#)の ImqHeader クラス・メソッドの説明を参照してください。

オブジェクト・メソッド (共有)

void operator = (const ImqReferenceHeader & header);

インスタンス・データを header からコピーし、既存のインスタンス・データを置き換えます。

ImqString destinationEnvironment () const ;

destination environment のコピーを返します。

void setDestinationEnvironment (const char * environment = 0);

destination environment を設定します。

ImqString destinationName () const ;

destination name のコピーを返します。

void setDestinationName (const char * name = 0);

destination name を設定します。

ImqBinary instanceId () const ;

instance id のコピーを返します。

ImqBoolean setInstanceId (const ImqBinary & id);

instance id を設定します。token の data length は、ゼロまたは MQ_OBJECT_INSTANCE_ID_LENGTH のいずれかでなければなりません。このメソッドは、正常に終了した場合には TRUE を返します。

void setInstanceId (const MQBYTE24 id = 0);

instance id を設定します。id はゼロであっても構いません。これは MQOII_NONE を指定するのと同じです。id が非ゼロの場合には、MQ_OBJECT_INSTANCE_ID_LENGTH バイトの 2 進データをアドレッシングするものでなければなりません。MQOII_NONE などの事前定義値を使用する場合は、確実にシグニチャーが一致するようにキャスト、例えば、(MQBYTE *)MQOII_NONE を作成する必要があります。

MQLONG logicalLength () const ;

logical length を返します。

void setLogicalLength (const MQLONG length);

logical length を設定します。

MQLONG logicalOffset () const ;

logical offset を返します。

void setLogicalOffset (const MQLONG offset);

logical offset を設定します。

MQLONG logicalOffset2 () const ;

logical offset 2 を返します。

void setLogicalOffset2 (const MQLONG offset);

logical offset 2 を設定します。

ImqString referenceType () const ;

reference type のコピーを返します。

void setReferenceType (const char * name = 0);

reference type を設定します。

ImqString sourceEnvironment () const ;

source environment のコピーを返します。

void setSourceEnvironment (const char * environment = 0);

source environment を設定します。

ImqString sourceName () const ;

source name のコピーを返します。

void setSourceName (const char * name = 0);

source name を設定します。

オブジェクト・データ (保護)

MQRMH omqrmh

MQRMH データ構造体。

理由コード

- MQRC_BINARY_DATA_LENGTH_ERROR
- MQRC_STRUC_LENGTH_ERROR
- MQRC_STRUC_ID_ERROR
- MQRC_INSUFFICIENT_DATA
- MQRC_INCONSISTENT_FORMAT
- MQRC_ENCODING_ERROR

ImqString C++ クラス

このクラスは、ヌル終了ストリングに文字ストリング・ストレージと操作を提供します。

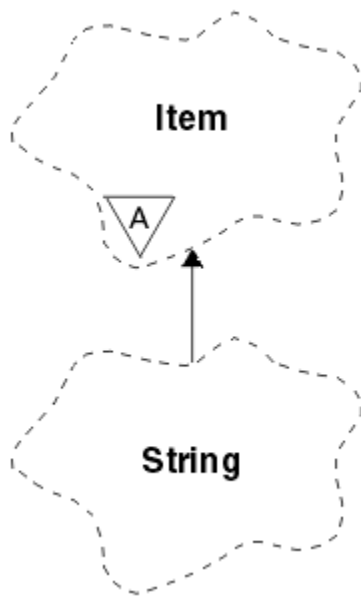


図 34. ImqString クラス

パラメーターが **char *** を呼び出す状態では、たいていの場合、**char *** の代わりに ImqString を使用します。

- [1889 ページの『オブジェクトの属性』](#)
- [1889 ページの『コンストラクター』](#)
- [1890 ページの『クラス・メソッド \(共有\)』](#)
- [1890 ページの『多重定義された ImqItem メソッド』](#)
- [1890 ページの『オブジェクト・メソッド \(共有\)』](#)
- [1893 ページの『オブジェクト・メソッド \(保護\)』](#)
- [1893 ページの『理由コード』](#)

オブジェクトの属性

characters

後書きヌルの前にある **storage** 内の文字。

length

characters 内のバイト数。 **storage** がない場合、**length** はゼロです。 初期値はゼロです。

storage

任意のサイズの、バイトの揮発性アレイ。 **characters** の終わりを検出できるように、**characters** の後ろの **storage** には、必ず後書きヌルが存在しなければなりません。 メソッドは、この状態が必ず保持されるようにしますが、アレイにバイトを直接設定する際には、必ず、変更の後に後書きヌルがあるようにしてください。 初期には、**storage** 属性は存在しません。

コンストラクター

ImqString();

デフォルトのコンストラクター。

ImqString(const ImqString & string);

コピー・コンストラクター。

ImqString(const char c);

characters は **c** から成ります。

ImqString(const char * text);

characters は **text** からコピーされます。

ImqString(const void * *buffer*, const size_t *length*);

buffer の最初の *length* 分のバイトをコピーし、それらを **characters** に割り当てます。コピーされたあらゆるヌル文字に対して置換が行われます。置換文字はピリオド(.)です。その他の印刷不能文字または表示不能文字がコピーされた場合は、特別な考慮事項はありません。

クラス・メソッド (共有)

static ImqBoolean copy(char * *destination-buffer*, const size_t *length*, const char * *source-buffer*, const char *pad* = 0);

最大 *length* 分のバイトを *source-buffer* から *destination-buffer* へコピーします。 *source-buffer* の文字数が不十分な場合、*destination-buffer* 内の残りのスペースに *pad* 文字を埋め込みます。 *source-buffer* はゼロでも構いません。 *length* がゼロの場合は、*destination-buffer* もゼロで構いません。エラー・コードはすべて失われます。このメソッドは、正常に終了した場合には TRUE を返します。

static ImqBoolean copy (char * *destination-buffer*, const size_t *length*, const char * *source-buffer*, ImqError & *error-object*, const char *pad* = 0);

最大 *length* 分のバイトを *source-buffer* から *destination-buffer* へコピーします。 *source-buffer* の文字数が不十分な場合、*destination-buffer* 内の残りのスペースに *pad* 文字を埋め込みます。 *source-buffer* はゼロでも構いません。 *length* がゼロの場合は、*destination-buffer* もゼロで構いません。エラー・コードはすべて、*error-object* に設定されます。このメソッドは、正常に終了した場合には TRUE を返します。

多重定義された ImqItem メソッド

virtual ImqBoolean copyOut (ImqMessage & *msg*);

characters をメッセージ・バッファへコピーし、既存の内容と置き換えます。 *msg format* を MQFMT_STRING に設定します。

詳細については、親クラス・メソッドの説明を参照してください。

virtual ImqBoolean pasteIn (ImqMessage & *msg*);

メッセージ・バッファから残りのデータを転送することにより **characters** を設定し、既存の **characters** を置き換えます。

読み取りが正しく実行されるようにするには、*msg* オブジェクトのエンコードを MQENC_NATIVE に設定する必要があります。MQGMO_CONVERT を使用してメッセージを MQENC_NATIVE に取り出します。

正常に実行されるためには、ImqMessage **format** が MQFMT_STRING でなければなりません。

詳細については、親クラス・メソッドの説明を参照してください。

オブジェクト・メソッド (共有)

char & operator [] (const size_t *offset*) const ;

storage 内のオフセット *offset* にある文字を参照します。関係のあるバイトが必ず存在し、アドレス可能であるようにしてください。

ImqString operator () (const size_t *offset*, const size_t *length* = 1) const ;

offset から始まる **characters** からバイトをコピーすることによってサブストリングを返します。

length がゼロの場合は、**characters** の残りを返します。 *offset* と *length* を組み合わせても **characters** 内で参照が行われない場合には、空の ImqString を返します。

void operator = (const ImqString & *string*);

インスタンス・データを *string* からコピーし、既存のインスタンス・データを置き換えます。

ImqString operator + (const char *c*) const ;

c を **characters** に付加した結果を返します。

ImqString operator + (const char * text) const ;

text を **characters** に付加した結果を返します。これは逆になっても構いません。以下に例を示します。

```
strOne + "string two" ;
"string one" + strTwo ;
```

注：多くのコンパイラーは **strOne + "string two"** ; を受け入れますが、Microsoft Visual C++ は **strOne + (char *) "string two"** ; を必要とします。

ImqString operator + (const ImqString & string1) const ;

string1 を **characters** に付加した結果を返します。

ImqString operator + (const double number) const ;

テキストに変換後 *number* を **characters** に付加した結果を返します。

ImqString operator + (const long number) const ;

テキストに変換後 *number* を **characters** に付加した結果を返します。

void operator += (const char c);

c を **characters** に付加します。

void operator += (const char * text);

text を **characters** に付加します。

void operator += (const ImqString & string);

string を **characters** に付加します。

void operator += (const double number);

テキストに変換後 *number* を **characters** に付加します。

void operator += (const long number);

テキストに変換後 *number* を **characters** に付加します。

operator char * () const ;

storage 内の最初のバイトのアドレスを返します。この値はゼロであっても構わず、揮発性です。このメソッドは、読み取り専用で使用してください。

ImqBoolean operator < (const ImqString & string) const ;

compare メソッドを使用して、**characters** を *string* のものと比較します。結果は、「未満」の場合は TRUE になり、「以上」の場合は FALSE になります。

ImqBoolean operator > (const ImqString & string) const ;

compare メソッドを使用して、**characters** を *string* のものと比較します。結果は、「より大」の場合は TRUE になり、「以下」の場合は FALSE になります。

ImqBoolean operator <= (const ImqString & string) const ;

compare メソッドを使用して、**characters** を *string* のものと比較します。結果は、「以下」の場合は TRUE になり、「より大」の場合は FALSE になります。

ImqBoolean operator >= (const ImqString & string) const ;

compare メソッドを使用して、**characters** を *string* のものと比較します。結果は、「以上」の場合は TRUE になり、「未満」の場合は FALSE になります。

ImqBoolean operator == (const ImqString & string) const ;

compare メソッドを使用して、**characters** を *string* のものと比較します。TRUE か FALSE のどちらかを返します。

ImqBoolean operator != (const ImqString & string) const ;

compare メソッドを使用して、**characters** を *string* のものと比較します。TRUE か FALSE のどちらかを返します。

short compare(const ImqString & string) const ;

characters を *string* のものと比較します。結果は、両方の **characters** が等しい場合はゼロ、前者の方が「小さい」場合は負、「大きい」場合は正です。比較には、大文字小文字の区別があります。ヌルの ImqString は、ヌルでない ImqString より「小さい」と見なされます。

ImqBoolean copyOut(char * *buffer*, const size_t *length*, const char *pad* = 0);

最大 *length* 分のバイトを **characters** から *buffer* へコピーします。 **characters** の文字数が不十分である場合、*buffer* 内の残りのスペースに *pad* 文字が埋め込まれます。 *length* がゼロの場合は、*buffer* もゼロで構いません。 正常に終了した場合は TRUE を返します。

size_t copyOut(long & *number*) const ;

テキストからの変換後に **characters** から *number* を設定し、変換にかかわった文字数を返します。これがゼロの場合、変換は行われておらず、*number* は設定されません。 変換可能文字シーケンスは、次の値で始まる必要があります。

```
<blank(s)>
<+|->
digit(s)
```

size_t copyOut(ImqString & *token*, const char *c* = ' ') const ;

characters に *c* と異なる 1 つまたは複数の文字が含まれている場合、トークンを、そのような文字の最初の連続するシーケンスとして識別します。 この場合、*token* はそのシーケンスに設定され、返された値は先行文字 *c* の数とシーケンス中のバイトの数との合計です。 それ以外の場合はゼロを返し、*token* を設定しません。

size_t cutOut(long & *number*);

copy メソッドの場合と同様に *number* を設定しますが、さらに、戻り値によって指示されたバイト数を **characters** から除去します。 例えば、次の例に示すストリングは、**cutOut(*number*)** を 3 回使用することにより、3 つの数値に分割できます。

```
strNumbers = "-1 0    +55 "

while ( strNumbers.cutOut( number ) );
number becomes -1, then 0, then 55
leaving strNumbers == " "
```

size_t cutOut(ImqString & *token*, const char *c* = ' ')

copyOut メソッドの場合と同様に *token* を設定し、**characters** から *strToken* 分の文字を除去し、さらに、*token* 分の文字の前にある任意の文字 *c* も除去します。 *c* がブランクでない場合、*token* 分の文字の直後に続く文字 *c* を除去します。 除去された文字数を返します。 例えば、次の例に示すストリングは、**cutOut(*token*)** を 3 回使用することにより、3 つのトークンに分割できます。

```
strText = " Program Version 1.1 "

while ( strText.cutOut( token ) );

// token becomes "Program", then "Version",
// then "1.1" leaving strText == " "
```

次に、DOS パス名を解析する方法について示します。

```
strPath = "C:\OS2\BITMAP\OS2LOGO.BMP"

strPath.cutOut( strDrive, ':' );
strPath.stripLeading( ':' );
while ( strPath.cutOut( strFile, '\\' ) );

// strDrive becomes "C".
// strFile becomes "OS2", then "BITMAP",
// then "OS2LOGO.BMP" leaving strPath empty.
```

ImqBoolean find(const ImqString & *string*);

characters 内のどこかに *string* の完全一致がないか探索します。 一致が見つからない場合は、FALSE を返します。 見つかった場合は、TRUE を返します。 *string* がヌルの場合は TRUE を返します。

ImqBoolean find(const ImqString & *string*, size_t & *offset*);

オフセット *offset* から先の **characters** 内のどこかに *string* の完全一致がないか探索します。 *string* がヌルの場合は、*offset* をアップデートせずに TRUE を返します。 一致が見つからない場合は FALSE を返します (*offset* の値が増やされている場合もあります)。 一致が見つかった場合は、TRUE を返し、*offset* を **characters** 内の *string* のオフセットにアップデートします。

size_t length() const ;
length を返します。

ImqBoolean pasteIn(const double number, const char * format = "%f");
テキストに変換後 *number* を **characters** に付加します。正常に終了した場合は TRUE を返します。

浮動小数点変換を形式設定するために、指定 *format* が使用されます。これを指定する場合は、**printf** および浮動小数点数 (例えば、**%.3f**) とともに使用するのに適したものでなければなりません。

ImqBoolean pasteIn(const long number);
テキストに変換後 *number* を **characters** に付加します。正常に終了した場合は TRUE を返します。

ImqBoolean pasteIn(const void * buffer, const size_t length);
buffer から **characters** に *length* 分のバイトを付加し、最後の後書きヌルを追加します。コピーされたあらゆるヌル文字を置換します。置換文字はピリオド (.) です。その他の印刷不能文字、またはコピーされた表示不能文字には、特別な考慮事項は与えられません。このメソッドは、正常に終了した場合には TRUE を返します。

ImqBoolean set(const char * buffer, const size_t length);
固定長文字フィールドからの **characters** を設定します。これには、ヌルが含まれている場合があります。必要に応じて固定長フィールドからの文字にヌルを付加します。このメソッドは、正常に終了した場合には TRUE を返します。

ImqBoolean setStorage(const size_t length);
storage を (再度) 割り振ります。後書きヌルを含め、元の **characters** が収まるだけの余裕がある場合には、それらを保存しますが、追加のストレージは初期化されません。
このメソッドは、正常に終了した場合には TRUE を返します。

size_t storage() const ;
storage 内のバイト数を返します。

size_t stripLeading(const char c = ' ');
characters から先行文字 *c* を除去し、除去された数を返します。

size_t stripTrailing(const char c = ' ');
characters から後書き文字 *c* を除去し、除去された数を返します。

ImqString upperCase() const ;
characters の大文字のコピーを返します。

オブジェクト・メソッド (保護)

ImqBoolean assign (const ImqString & string);
対応する **operator =** メソッドと同じ処理を行います。ただし、仮想メソッドではありません。正常に終了した場合は TRUE を返します。

理由コード

- MQRC_DATA_TRUNCATED
- MQRC_NULL_POINTER
- MQRC_STORAGE_NOT_AVAILABLE
- MQRC_BUFFER_ERROR
- MQRC_INCONSISTENT_FORMAT

ImqTrigger C++ クラス

このクラスは、MQTM (トリガー・メッセージ) データ構造をカプセル化します。

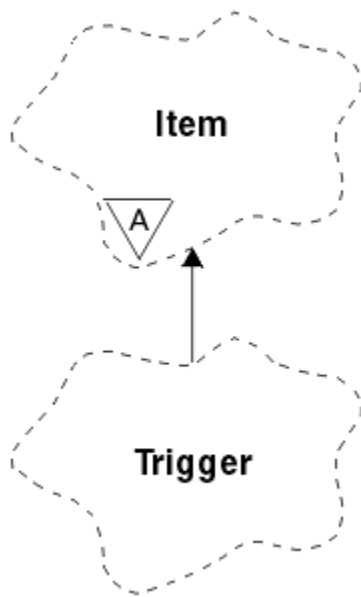


図 35. *ImqTrigger* クラス

このクラスのオブジェクトは通常、トリガー・モニター・プログラムによって使用されます。トリガー・モニター・プログラムのタスクは、特定のメッセージを待機し、メッセージが他の IBM MQ アプリケーションを待機している場合にそのアプリケーションが確実に開始されるようにメッセージを処理することです。

使用法の例については、IMQSTRG サンプル・プログラムを参照してください。

- [1894 ページの『オブジェクトの属性』](#)
- [1895 ページの『コンストラクター』](#)
- [1895 ページの『多重定義された *ImqItem* メソッド』](#)
- [1895 ページの『オブジェクト・メソッド \(共有\)』](#)
- [1896 ページの『オブジェクト・データ \(保護\)』](#)
- [1896 ページの『理由コード』](#)

オブジェクトの属性

application id

当該メッセージを送信したアプリケーションの ID。初期値はヌル・ストリングです。

application type

当該メッセージを送信したアプリケーションのタイプ。初期値はゼロです。以下の値が取得可能な値です。

- MQAT_AIX
- MQAT_CICS
- MQAT_DOS
- MQAT_IMS
- MQAT_MVS
- MQAT_NOTES_AGENT
- MQAT_OS2
- MQAT_OS390
- MQAT_OS400
- MQAT_UNIX

- MQAT_WINDOWS
- MQAT_WINDOWS_NT
- MQAT_USER_FIRST
- MQAT_USER_LAST

environment data

プロセスの環境データ。初期値はヌル・ストリングです。

process name

プロセス名。初期値はヌル・ストリングです。

キュー名

開始されるキューの名前。初期値はヌル・ストリングです。

trigger data

プロセスのトリガー・データ。初期値はヌル・ストリングです。

ユーザー・データ

プロセスのユーザー・データ。初期値はヌル・ストリングです。

コンストラクター

ImqTrigger();

デフォルトのコンストラクター。

ImqTrigger(const ImqTrigger & trigger ());

コピー・コンストラクター。

多重定義された ImqItem メソッド

virtual ImqBoolean copyOut (ImqMessage & msg);

MQTM データ構造体をメッセージ・バッファーに書き込み、既存の内容と置き換えます。また、*msg* の形式を MQFMT_TRIGGER に設定します。

詳細については、[1838 ページの『ImqItem C++ クラス』](#)の ImqItem クラス・メソッドの説明を参照してください。

virtual ImqBoolean pasteIn (ImqMessage & msg);

メッセージ・バッファーから MQTM データ構造体を読み取ります。

正常に実行されるためには、ImqMessage の形式が MQFMT_TRIGGER でなければなりません。

詳細については、[1838 ページの『ImqItem C++ クラス』](#)の ImqItem クラス・メソッドの説明を参照してください。

オブジェクト・メソッド (共有)

void operator = (const ImqTrigger & trigger ());

インスタンス・データを *trigger* からコピーし、既存のインスタンス・データを置き換えます。

ImqString applicationId () const ;

application id のコピーを返します。

void setApplicationId (const char * id);

application id を設定します。

MQLONG applicationType () const ;

application type を返します。

void setApplicationType (const MQLONG type);

application type を設定します。

ImqBoolean copyOut (MQTMC2 * ptmc2);

開始キューで受信された MQTM データ構造体をカプセル化します。呼び出し側によって提供された同等な MQTMC2 データ構造体に値を入力し、QMgrName フィールド (MQTM データ構造体には存在しません) をすべてブランクに設定します。MQTMC2 データ構造体は、従来、トリガー・モニターによって

開始されたアプリケーションに対するパラメーターとして使用されています。このメソッドは、正常に終了した場合には TRUE を返します。

ImqString environmentData () const ;

environment data のコピーを返します。

void setEnvironmentData (const char * data);

environment data を設定します。

ImqString processName () const ;

process name のコピーを返します。

void setProcessName (const char * name);

process name を設定します。長さが 48 文字になるまでブランクが埋め込まれます。

ImqString queueName () const ;

queue name のコピーを返します。

void setQueueName (const char * name);

queue name を設定します。長さが 48 文字になるまでブランクが埋め込まれます。

ImqString triggerData () const ;

trigger data のコピーを返します。

void setTriggerData (const char * data);

trigger data を設定します。

ImqString userData () const ;

user data のコピーを返します。

void setUserData (const char * data);

user data を設定します。

オブジェクト・データ (保護)

MQTM omqtm

MQTM データ構造体。

理由コード

- MQRC_NULL_POINTER
- MQRC_INCONSISTENT_FORMAT
- MQRC_ENCODING_ERROR
- MQRC_STRUC_ID_ERROR

ImqWorkHeader C++ クラス

このクラスは、MQWIH データ構造体の特定の機能をカプセル化します。

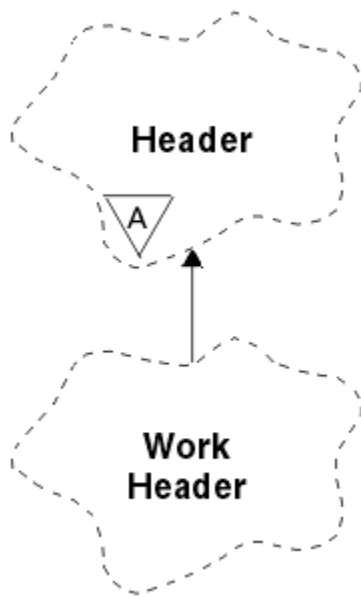


図 36. *ImqWorkHeader* クラス

このクラスのオブジェクトは、z/OS Workload Manager が管理するキューにメッセージを書き込むアプリケーションによって使用されます。

- [1897 ページの『オブジェクトの属性』](#)
- [1897 ページの『コンストラクター』](#)
- [1897 ページの『多重定義された *ImqItem* メソッド』](#)
- [1898 ページの『オブジェクト・メソッド \(共有\)』](#)
- [1898 ページの『オブジェクト・データ \(保護\)』](#)
- [1898 ページの『理由コード』](#)

オブジェクトの属性

メッセージ・トークン (message token)

z/OS Workload Manager 用のメッセージ・トークン。長さは MQ_MSG_TOKEN_LENGTH です。初期値は MQMTOK_NONE です。

サービス名

プロセスの名前 (32 文字)。初期値はブランクです。

service step

プロセス内の 1 つのステップの名前 (8 文字)。初期値はブランクです。

コンストラクター

ImqWorkHeader();

デフォルトのコンストラクター。

ImqWorkHeader(const *ImqWorkHeader* & *header*);

コピー・コンストラクター。

多重定義された *ImqItem* メソッド

virtual *ImqBoolean* copyOut(*ImqMessage* & *msg*);

MQWIH データ構造体をメッセージ・バッファの始めに挿入し、既存のメッセージ・データを後ろにずらし、*msg format* を MQFMT_WORK_INFO_HEADER に設定します。

詳細については、親クラス・メソッドの説明を参照してください。

virtual ImqBoolean pasteIn(ImqMessage & msg);

メッセージ・バッファーから MQWIH データ構造体を読み取ります。

読み取りが正しく実行されるようにするには、*msg* オブジェクトのエンコードを MQENC_NATIVE に設定する必要があります。MQGMO_CONVERT を使用してメッセージを MQENC_NATIVE に取り出します。

ImqMessage 形式は、必ず MQFMT_WORK_INFO_HEADER でなければなりません。

詳細については、親クラス・メソッドの説明を参照してください。

オブジェクト・メソッド (共有)

void operator = (const ImqWorkHeader & header);

インスタンス・データを *header* からコピーし、既存のインスタンス・データを置き換えます。

ImqBinary messageToken () const;

message token を返します。

ImqBoolean setMessageToken(const ImqBinary & token);

message token を設定します。 *token* の data length は、ゼロまたは、MQ_MSG_TOKEN_LENGTH のいずれかでなければなりません。正常に終了した場合は TRUE を返します。

void setMessageToken(const MQBYTE16 token = 0);

message token を設定します。 *token* はゼロであっても構いません。これは MQMTOK_NONE を指定するのと同じです。 *token* が非ゼロの場合には、MQ_MSG_TOKEN_LENGTH バイトの 2 進データをアドレッシングするものでなければなりません。

MQMTOK_NONE などの事前定義値を使用する場合は、確実にシグニチャーが一致するようにキャスト (例えば、(MQBYTE *)MQMTOK_NONE) を作成しなければならないことがあります。

ImqString serviceName () const;

service name を後書きブランクを埋め込んで返します。

void setServiceName(const char * name);

service name を設定します。

ImqString serviceStep () const;

service step を後書きブランクを埋め込んで返します。

void setServiceStep(const char * step);

service step を設定します。

オブジェクト・データ (保護)

MQWIH omqwih

MQWIH データ構造体。

理由コード

- MQRC_BINARY_DATA_LENGTH_ERROR

IBM MQ classes for JMS オブジェクトのプロパティー

IBM MQ classes for JMS のすべてのオブジェクトはプロパティーを持ちます。各種プロパティーがさまざまなオブジェクト・タイプに適用されます。プロパティーが持つ許容値もそれぞれ異なります。また、シンボリック・プロパティーの値も管理ツールとプログラム・コードの間で異なります。

IBM MQ classes for JMS には、IBM MQ JMS 管理ツールや、IBM MQ エクスプローラーを使用するか、アプリケーションで、オブジェクトのプロパティーを設定および照会する機能が備わっています。プロパティーの多くは、特定のオブジェクト・タイプのサブセットにのみ関係します。

IBM MQ JMS 管理ツールの使い方については、[Configuring JMS objects using the administration tool](#) を参照してください。

1899 ページの表 904 に、各プロパティの概要と、適用されるオブジェクト・タイプを各プロパティごとに示します。オブジェクト・タイプはキーワードを使用して識別されます。これらのオブジェクトの説明については、[Configuring JMS objects using the administration tool](#) を参照してください。

数字は、表の最後にある注記を示しています。1902 ページの『IBM MQ classes for JMS オブジェクトのプロパティ間の依存関係』も参照してください。

プロパティは、次の形式の名前/値のペアから構成されています。

```
PROPERTY_NAME(property_value)
```

このセクションのトピックでは、各プロパティについて、プロパティの名前と概要をリストし、管理ツールで使用される有効なプロパティ値を示します。また、アプリケーションでプロパティの値を設定するために使用する set メソッドを示します。これらのトピックではまた、各プロパティで有効なプロパティ値、およびツールで使用されるシンボリック・プロパティ値とそれに相当するプログラマブル・プロパティ値との間のマッピングも示しています。

プロパティの名前は、大文字小文字を区別せず、これらのトピックで示す認識された一連の名前に限定されます。

表 904. プロパティの名前と適用可能なオブジェクト・タイプ									
Property	短縮形	オブジェクト・タイプ							
		CF	QCF	TCF	Q	T	XACF	XAQCF	XATCF
1904 ページの『APPLICATIONNAME』	APPNAME	Y	Y	Y			Y	Y	Y
1905 ページの『ASYNCEXCEPTION』	AEX	Y	Y	Y			Y	Y	Y
1906 ページの『バルオプション』	オプション	Y	Y	Y			Y	Y	Y
1906 ページの『BALTYPE』	タイプ	Y	Y	Y			Y	Y	Y
1907 ページの『BALTIMEOUT』	TIMEOUT	Y	Y	Y			Y	Y	Y
1907 ページの『BROKERCCDURSUBQ』 ¹	CCDSUB					Y			
1908 ページの『BROKERCCSUBQ』 ¹	CCSUB	Y		Y			Y		Y
1908 ページの『BROKERCONQ』 ¹	BCON	Y		Y			Y		Y
1909 ページの『BROKERDURSUBQ』 ¹	BDSUB					Y			
1909 ページの『BROKERPUBQ』 ¹	BPUB	Y		Y		Y	Y		Y
1910 ページの『BROKERPUBQMGR』 ¹	BPQM					Y			
1910 ページの『BROKERQMGR』 ¹	BQM	Y		Y			Y		Y
1910 ページの『BROKERSUBQ』 ¹	BSUB	Y		Y			Y		Y
1911 ページの『BROKERVER』 ¹	BVER	Y ²		Y ²		Y	Y		Y
1912 ページの『CCDTURL』 ³	CCDT	Y	Y	Y			Y	Y	Y
1913 ページの『CERTVALPO』	CVAP	Y							
1912 ページの『CCSID』	CCS	Y	Y	Y	Y	Y	Y	Y	Y
1913 ページの『CHANNEL』 ³	CHAN	Y	Y	Y			Y	Y	Y
1914 ページの『CLEANUP』 ¹	CL	Y		Y			Y		Y
1914 ページの『CLEANUPINT』 ¹	CLINT	Y		Y			Y		Y

表 904. プロパティの名前と適用可能なオブジェクト・タイプ (続き)

Property	短縮形	オブジェクト・タイプ							
		CF	QCF	TCF	Q	T	XACF	XAQCF	XATCF
1915 ページの『CONNECTIONNAMELIST』	CNLIST	Y	Y	Y					
1915 ページの『CLIENTRECONNECTOPTIONS』	CROPT	Y	Y	Y					
1916 ページの『CLIENTRECONNECTTIMEOUT』	CRT	Y	Y	Y					
1916 ページの『CLIENTID』	CID	Y ²	Y	Y ²			Y	Y	Y
1917 ページの『CLONESUPP』	CLS	Y		Y			Y		Y
1917 ページの『COMPHDR』	HC	Y		Y			Y		Y
1918 ページの『COMPMSG』	MC	Y	Y	Y			Y	Y	Y
1918 ページの『CONNOPT』	CNOPT	Y	Y	Y			Y	Y	Y
1919 ページの『CONNTAG』	CNTAG	Y	Y	Y			Y	Y	Y
1920 ページの『説明』	DESC	Y ²	Y	Y ²	Y	Y	Y	Y	Y
1920 ページの『DIRECTAUTH』	DAUTH	Y ²		Y ²					
1921 ページの『ENCODING』	ENC				Y	Y			
1922 ページの『EXPIRY』	EXP				Y	Y			
1922 ページの『FAILIFQUIESCE』	FIQ	Y	Y	Y	Y	Y	Y	Y	Y
1923 ページの『HOSTNAME』	HOST	Y ²	Y	Y ²			Y	Y	Y
1923 ページの『LOCALADDRESS』	LA	Y ²	Y	Y ²			Y	Y	Y
1924 ページの『MAPNAMESTYLE』	MNST	Y	Y	Y			Y	Y	Y
1925 ページの『MAXBUFFSIZE』	MBSZ	Y ²		Y ²					
1925 ページの『MDREAD』	MDR				Y	Y			
1926 ページの『MDWRITE』	MDW				Y	Y			
1926 ページの『MDMSGCTX』	MDCTX				Y	Y			
1927 ページの『MSGBATCHSZ』¹	MBS	Y	Y	Y			Y	Y	Y
1927 ページの『MSGBODY』	MBODY				Y	Y			
1928 ページの『MSGRETENTION』	MRET	Y	Y				Y	Y	
1928 ページの『MSGSELECTION』¹	MSEL	Y		Y			Y		Y
1929 ページの『MULTICAST』	MCAST	Y ²		Y ²		Y			
1930 ページの『OPTIMISTICPUBLICATION』¹	OPTPUB	Y		Y					
1930 ページの『OUTCOMENOTIFICATION』¹	NOTIFY	Y		Y					
1931 ページの『PERSISTENCE』	PER				Y	Y			
1931 ページの『POLLINGINT』¹	PINT	Y	Y	Y			Y	Y	Y

表 904. プロパティの名前と適用可能なオブジェクト・タイプ (続き)

Property	短縮形	オブジェクト・タイプ							
		CF	QCF	TCF	Q	T	XACF	XAQCF	XATCF
1932 ページの『PORT』	PORT	Y ²	Y	Y ²			Y	Y	Y
1932 ページの『PRIORITY』	PRI				Y	Y			
1933 ページの『PROCESSDURATION』 1	PROCUR	Y		Y					
1933 ページの『PROVIDERVERSION』	PVER	Y	Y	Y			Y	Y	Y
1935 ページの『PROXYHOSTNAME』	PHOST	Y ²		Y ²					
1936 ページの『PROXYPORT』	PPORT	Y ²		Y ²					
1936 ページの『PUBACKINT』 ¹	PAI	Y		Y			Y		Y
1937 ページの『PUTASYNCALLOWED』	PAALD				Y	Y			
1937 ページの『QMANAGER』	QMGR	Y	Y	Y	Y		Y	Y	Y
1938 ページの『QUEUE』	QU				Y				
1938 ページの『READAHEADALLOWED』	RAALD				Y	Y			
1939 ページの『READAHEADCLOSEPOLICY』	RACP				Y	Y			
1939 ページの『RECEIVECCSID』	RCCS				Y	Y			
1940 ページの『RECEIVECONVERSION』	RCNV				Y	Y			
1940 ページの『RECEIVEISOLATION』 1	RCVISOL	Y		Y					
1941 ページの『RECEXIT』	RCX	Y	Y	Y			Y	Y	Y
1941 ページの『RECEXITINIT』	RCXI	Y	Y	Y			Y	Y	Y
1942 ページの『REPLYTOSTYLE』	RTOST				Y	Y			
1942 ページの『RESCANINT』 ¹	RINT	Y	Y				Y	Y	
1943 ページの『SECEXIT』	SCX	Y	Y	Y			Y	Y	Y
1943 ページの『SECEXITINIT』	SCXI	Y	Y	Y			Y	Y	Y
1944 ページの『SENDCHECKCOUNT』	SCC	Y	Y	Y			Y	Y	Y
1944 ページの『SENDEXIT』	SDX	Y	Y	Y			Y	Y	Y
1945 ページの『SENDEXITINIT』	SDXI	Y	Y	Y			Y	Y	Y
1945 ページの『SHARECONVALLOWED』	SCALD	Y	Y	Y			Y	Y	Y
1946 ページの『SPARSESUBS』 ¹	SSUBS	Y		Y					
1946 ページの『SSLCIPHERSUITE』	SCPHS	Y	Y	Y			Y	Y	Y
1947 ページの『SSLCRL』	SCRL	Y	Y	Y			Y	Y	Y
1947 ページの『SSLFIPSREQUIRED』	SFIPS	Y	Y	Y			Y	Y	Y

表 904. プロパティの名前と適用可能なオブジェクト・タイプ (続き)

Property	短縮形	オブジェクト・タイプ							
		CF	QCF	TCF	Q	T	XACF	XAQCF	XATCF
1948 ページの『SSLPEERNAME』	SPEER	Y	Y	Y			Y	Y	Y
1949 ページの『SSLRESETCOUNT』	SRC	Y	Y	Y			Y	Y	Y
1949 ページの『STATREFRESHINT』 ¹	SRI	Y		Y			Y		Y
1950 ページの『SUBSTORE』 ¹	SS	Y		Y			Y		Y
1950 ページの『SYNCPOINTALLGETS』	SPAG	Y	Y	Y			Y	Y	Y
1951 ページの『TARGCLIENT』	TC				Y	Y			
1951 ページの『TARGCLIENTMATCHING』	TCM	Y	Y				Y	Y	
1952 ページの『TEMPMODEL』	TM	Y	Y				Y	Y	
1952 ページの『TEMPQPREFIX』	TQP	Y	Y				Y	Y	
1953 ページの『TEMPTOPICPREFIX』	TTP	Y		Y			Y		Y
v 7.0.0 トークンクライアント ID	TKCD	Y	Y	Y			Y	Y	Y
v 7.0.0 トークンクライアントシークレット	TKST	Y	Y	Y			Y	Y	Y
v 7.0.0 トークンエンドポイント	TKEP	Y	Y	Y			Y	Y	Y
v 7.0.0 トークン HTTPS 証明書検証ポリシー	TCVP	Y	Y	Y			Y	Y	Y
1955 ページの『トピック』	TOP					Y			
1956 ページの『TRANSPORT』	TRAN	Y ²	Y	Y ²			Y	Y	Y
1957 ページの『WILDCARDFORMAT』	WCfmt	Y		Y			Y		Y
v 7.0.0 WMQ CCDT HTTPS 証明書検証ポリシー	CCVP	Y	Y	Y			Y	Y	Y

注:

1. このプロパティはバージョン 70 で使用可能ですが IBM MQ classes for JMS、接続ファクトリの ``プロパティ **PROVIDERVERSION** がバージョン 7 未満に設定されていない限り、 ``キュー IBM WebSphere MQ 7.0 マネージャに接続されたアプリケーションでは効果を持ちません。
2. ブローカーへのリアルタイム接続を使用する場合、ConnectionFactory または TopicConnectionFactory オブジェクトでは、**CLIENTID**, **DESCRIPTION**, **DIRECTAUTH HOSTNAME**, **LOCALADDRESS**, **MAXBUFFSIZE**, **MULTICAST**, **PORT**, **PROXYHOSTNAME**, **PROXYPORT**, **BROKERVER**, **TRANSPORT** およびプロパティのみがサポートされます。
3. オブジェクトのプロパティと **CHANNEL CCDTURL** プロパティは、同時に両方設定してはならない。

関連タスク

[JMS オブジェクトの作成](#)

IBM MQ classes for JMS オブジェクトのプロパティ間の依存関係

一部のプロパティの妥当性は、他のプロパティの特定の値に依存します。

この依存関係は、以下のグループのプロパティに生じます。

- クライアント・プロパティ
- ブローカーとのリアルタイム接続用のプロパティ
- 出口初期化ストリング

クライアント・プロパティ

キュー・マネージャーとの接続の場合、TRANSPORT の値が CLIENT のときにのみ、以下のプロパティが使用されます。

- HOSTNAME
- PORT
- CHANNEL
- LOCALADDRESS
- CCDTURL
- CCSID
- COMPHDR
- COMPMSG
- RECEXIT
- RECEXITINIT
- SECEXIT
- SECEXITINIT
- SENDEXIT
- SENDEXITINIT
- SHARECONVALLOWED
- SSLCIPHERSUITE
- SSLCRL
- SSLFIPSREQUIRED
- SSLPEERNAME
- SSLRESETCOUNT
- APPLICATIONNAME

TRANSPORT の値が BIND の場合は、管理ツールを使用してこれらのプロパティに値を設定することはできません。

TRANSPORT の値が CLIENT の場合は、BROKERVER プロパティのデフォルト値は V1 であり、PORT プロパティのデフォルト値は 1414 です。 BROKERVER または PORT の値を明示的に設定すると、後で TRANSPORT の値を変更してもユーザーの選択項目はオーバーライドされません。

ブローカーとのリアルタイム接続用のプロパティ

TRANSPORT の値が DIRECT または DIRECTHTTP の場合は、以下のプロパティのみが関係します。

- BROKERVER
- CLIENTID
- 説明
- DIRECTAUTH
- HOSTNAME
- LOCALADDRESS
- MAXBUFFSIZE
- MULTICAST (DIRECT でのみサポートされています)
- PORT

- PROXYHOSTNAME (DIRECT でのみサポートされています)
- PROXYPORT (DIRECT でのみサポートされています)

TRANSPORT の値が DIRECT または DIRECTHTTP の場合は、BROKERVER プロパティのデフォルト値は V2 であり、PORT プロパティのデフォルト値は 1506 です。BROKERVER または PORT の値を明示的に設定すると、後で TRANSPORT の値を変更してもユーザーの選択項目はオーバーライドされません。

出口初期化ストリング

どのような出口初期化ストリングを設定する場合にも、該当する出口名を必ず指定してください。出口初期化プロパティには、以下のプロパティが含まれます。

- REEXITINIT
- SEEXITINIT
- SENDEXITINIT

例えば、REEXIT(some.exit.classname) を指定せずに REEXITINIT(myString) を指定すると、エラーが発生します。

関連資料

[1956 ページの『TRANSPORT』](#)

キュー・マネージャーまたはブローカーとの接続の性質。

APPLICATIONNAME

アプリケーションに名前を設定して、アプリケーションからキュー・マネージャーへの接続を識別することができます。このアプリケーション名は、コマンド **DISPLAY CONN MQSC/PCF** (フィールド名は **APPLTAG**) で表示されるか、または IBM MQ エクスプローラーのアプリケーション接続表示 (フィールド名は **App name**) で表示されます。

適用可能なオブジェクト

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理ツールのロング・ネーム: APPLICATIONNAME

JMS 管理ツールのショート・ネーム: APPNAME

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setAppName()
- MQConnectionFactory.getAppName()

値

28 文字以内の有効なストリング。それより長い名前は、必要に応じて、制限内に収まるように、それより前の位置にあるパッケージ名を削除して調整されます。例えば、起動クラスが `com.example.MainApp` である場合は完全な名前が使用されますが、起動クラスが `com.example.dictionaryAndThesaurus.multilingual.mainApp` である場合は `multilingual.mainApp` が使用されます。これが、有効な長さに収まるクラス名と右端のパッケージ名との最長の組み合わせだからです。

クラス名自体が 28 文字より長い場合は、収まるように切り捨てられます。例えば、`com.example.mainApplicationForSecondTestCase` は `mainApplicationForSecondTest` となります。

 z/OS では、APPNAME は以下のようになります。

- バインド・モードでは、設定されている場合は無視され、ブランクにしか設定できません。
- クライアント・モードでは、設定して使用することができます。

ASYNCEXCEPTION

このプロパティは、IBM MQ classes for JMS が ExceptionListener への通知を、接続が切断されたときのみ行うか、または JMS API 呼び出しに対して非同期に例外が発生したときに行うかを決定します。これは ExceptionListener が登録されているこの ConnectionFactory から作成されたすべての接続に対して適用されます。

適用可能なオブジェクト

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理ツールのロング・ネーム: ASYNCEXCEPTION

JMS 管理ツールのショート・ネーム: AEX

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setAsyncExceptions()
- MQConnectionFactory.getAsyncExceptions()

値

ASYNCEXCEPTIONS_ALL

同期 API 呼び出しの有効範囲外で非同期に検出されたすべての例外、および接続切断の例外はすべて ExceptionListener に送信されます。

表 905. すべての非同期例外: 環境とそれに関連する定数名	
Environment	値
JMS 管理ツール	ALL
プログラムの使用	WMQCONSTANTS.ASYNC_EXCEPTIONS_ALL = -1
IBM MQ Explorer	すべて

ASYNCEXCEPTIONS_CONNECTIONBROKEN

切断された接続を示す例外のみが ExceptionListener に送信されます。非同期処理中に起きる他のすべての例外は ExceptionListener には報告されず、そのためアプリケーションにはそれらの例外は通知されません。これは IBM MQ 8.0.0 Fix Pack 2 からのデフォルト値です。 [JMS: IBM MQ 8.0 での例外リスナーの変更点](#)を参照してください。

表 906. 切断された接続を示す例外: 環境とそれに関連する定数名	
Environment	値
JMS 管理ツール	CONNECTIONBROKEN
プログラムの使用	WMQCONSTANTS.ASYNC_EXCEPTIONS_CONNECTIONBROKEN = 1
IBM MQ Explorer	接続切断

次の追加の定数が定義されました。

- IBM MQ 8.0.0 Fix Pack 2 以降: WMQCONSTANTS.ASYNC_EXCEPTIONS_DEFAULT = ASYNCEXCEPTIONS_CONNECTIONBROKEN

- IBM MQ 8.0.0 Fix Pack 2 より前: WMQCONSTANTS.ASYNC_EXCEPTIONS_DEFAULT = ASYNC_EXCEPTIONS_ALL

関連概念

[IBM MQ classes for JMS での例外](#)

バルオプション

クライアントトランスポートを使用するアプリケーションが、均一 IBM MQ classes for JMS クラスター内どのように再バランスされるかを制御します。

適用可能なオブジェクト

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory.

JMS 管理ツールの長い名前: **BALOPTIONS**

JMS 管理ツールの略称: **OPTIONS**

プログラムによるアクセス

セッター/ゲッター

- `MQConnectionFactory.setBalancingOptions()`
- `MQConnectionFactory.getBalancingOptions()`

値

IGNNONE

通常のトランザクション処理が適用され、トランザクション実行中にアプリケーションが移動するように要求されることはありません。

この値は IBM `MQBalancingOption` `MQBNO_OPTIONS_NONE` にマップされます。

IGNTRANS

トランザクション中にアプリケーションの移動が要求される場合があります。

この値は IBM `MQBalancingOption` `MQBNO_OPTIONS_IGNORE_TRANS` にマップされます。

BALTYPE

クライアント転送を使用するアプリケーションが、IBM MQ classes for JMS ユニフォームクラスター内どのように再バランスされるかを制御します。

適用可能なオブジェクト

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory.

JMS 管理ツールの長い名前: **BALTYPE**

JMS 管理ツールの略称: **TYPE**

プログラムによるアクセス

セッター/ゲッター

- `MQConnectionFactory.setBalancingApplicationType()`
- `MQConnectionFactory.getBalancingApplicationType()`

値

SIMPLE

均一クラスタにおけるアプリケーションのデフォルト処理が適用されます。

この値は IBM *MQBalancingOption* `MQBNO_BALTYPE_SIMPLE` にマップされます。

リクエスト・リプライ

タイムアウト期間が経過しない限り、によってが **MQPUT** 相殺されていない場合、アプリケーション **MQGET** は再接続を要求されません。

この値は IBM *MQBalancingOption* `MQBNO_BALTYPE_REQREP` にマッピングされます

BALTIMEOUT

クライアントトランスポートを使用するアプリケーションが、ユニフォーム IBM MQ classes for JMS クラスタ内でどのように再バランスされるかを制御します。

適用可能なオブジェクト

`ConnectionFactory`, `QueueConnectionFactory`, `TopicConnectionFactory`,
`XAConnectionFactory`, `XAQueueConnectionFactory`, `XATopicConnectionFactory`.

JMS 管理ツールの長い名前: **BALTIMEOUT**

JMS 管理ツールの略称: **TIMEOUT**

プログラムによるアクセス

セッター/ゲッター

- `MQConnectionFactory.setBalancingTimeout()`
- `MQConnectionFactory.getBalancingTimeout()`

値

NEVER

均一なクラスターにおいて、リバランスを目的としてアプリケーションがタイムアウトすることは決してない。

この値は IBM *MQBalancingOption* `MQBNO_TIMEOUT_NEVER` にマップされます。

Immediate

アプリケーションは、均一なクラスターにおけるリバランスを目的として、直ちにタイムアウトする。

この値は IBM *MQBalancingOption* `MQBNO_TIMEOUT_IMMEDIATE` にマップされます。

デフォルト

アプリケーションは、デフォルトの 10 秒経過後、均一クラスター内でのリバランス目的でタイムアウトする。

この値は IBM *MQBalancingOption* `MQBNO_TIMEOUT_AS_DEFAULT` にマッピングされます。

nn

アプリケーションは、均一クラスター内でのリバランスを目的として、*nn* 秒経過後にタイムアウトする。

nn は 1 から 999999999 の間の値を取ることができます。

BROKERCCDURSUBQ

`ConnectionConsumer` のために永続サブスクライブ・メッセージを取り出すキューの名前。

適用可能なオブジェクト

トピック

JMS 管理ツールのロング・ネーム: BROKERCCDURSUBQ

JMS 管理ツールのショート・ネーム: CCDSUB

プログラムによるアクセス

セッター/ゲッター

- MQTopic.setBrokerCCDurSubQueue()
- MQTopic.getBrokerCCDurSubQueue()

値

SYSTEM.JMS.D.CC.SUBSCRIBER.QUEUE

これがデフォルト値です。

任意の有効なストリング

BROKERCCSUBQ

ConnectionConsumer の非永続サブスクリプション・メッセージを取り出すキューの名前。

適用可能なオブジェクト

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

JMS 管理ツールのロング・ネーム: BROKERCCSUBQ

JMS 管理ツールのショート・ネーム: CCSUB

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setBrokerCCSubQueue()
- MQConnectionFactory.getBrokerCCSubQueue()

値

SYSTEM.JMS.ND.CC.SUBSCRIBER.QUEUE

これがデフォルト値です。

任意の有効なストリング

BROKERCONQ

ブローカーの制御キュー名。

適用可能なオブジェクト

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

JMS 管理ツールのロング・ネーム: BROKERCONQ

JMS 管理ツールのショート・ネーム: BCON

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setBrokerControlQueue()
- MQConnectionFactory.getBrokerControlQueue()

値

SYSTEM.BROKER.CONTROL.QUEUE

これがデフォルト値です。

任意の有効なストリング

BROKERDURSUBQ

IBM MQ classes for JMS が IBM MQ メッセージング・プロバイダー移行モードで使用されている場合、このプロパティは永続サブスクリプション・メッセージが取得されるキューの名前を指定します。

適用可能なオブジェクト

トピック

JMS 管理ツールのロング・ネーム: BROKERDURSUBQ

JMS 管理ツールのショート・ネーム: BDSUB

プログラムによるアクセス

セッター/ゲッター

- MQTopic.setBrokerDurSubQueue()
- MQTopic.getBrokerDurSubQueue()

値

SYSTEM.JMS.D.SUBSCRIBER.QUEUE

これがデフォルト値です。

任意の有効なストリング

SYSTEM.JMS.D で始まるもの

関連タスク

JMS PROVIDERVERSION プロパティの構成

BROKERPUBQ

パブリッシュ済みメッセージが送信されたキュー (ストリーム・キュー) の名前。

適用可能なオブジェクト

ConnectionFactory、TopicConnectionFactory、Topic、XAConnectionFactory、XATopicConnectionFactory

JMS 管理ツールのロング・ネーム: BROKERPUBQ

JMS 管理ツールのショート・ネーム: BPUB

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setBrokerPubQueue
- MQConnectionFactory.getBrokerPubQueue

値

SYSTEM.BROKER.DEFAULT.STREAM

これがデフォルト値です。

任意の有効なストリング

BROKERPUBQMGR

トピックに関して公開されたメッセージが送信されるキューを所有するキュー・マネージャーの名前。

適用可能なオブジェクト

トピック

JMS 管理ツールのロング・ネーム: BROKERPUBQMGR

JMS 管理ツールのショート・ネーム: BPQM

プログラムによるアクセス

セッター/ゲッター

- MQTopic.setBrokerPubQueueManager()
- MQTopic.getBrokerPubQueueManager()

値

null

これがデフォルト値です。

任意の有効なストリング

BROKERQMGR

ブローカーが稼働しているキュー・マネージャーの名前。

適用可能なオブジェクト

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

JMS 管理ツールのロング・ネーム: BROKERQMGR

JMS 管理ツールのショート・ネーム: BQM

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setBrokerQueueManager()
- MQConnectionFactory.getBrokerQueueManager()

値

null

これがデフォルト値です。

任意の有効なストリング

BROKERSUBQ

IBM MQ classes for JMS が IBM MQ メッセージング・プロバイダー移行モードで使用されている場合、このプロパティは非永続サブスクリプション・メッセージが取得されるキューの名前を指定します。

適用可能なオブジェクト

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

JMS 管理ツールのロング・ネーム: BROKERSUBQ

JMS 管理ツールのショート・ネーム: BSUB

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setBrokerSubQueue()
- MQConnectionFactory.getBrokerSubQueue()

値

SYSTEM.JMS.ND.SUBSCRIBER.QUEUE

これがデフォルト値です。

任意の有効なストリング

SYSTEM.JMS.ND で始まるもの

関連タスク

JMS PROVIDERVERSION [プロパティの構成](#)

BROKERVER

使用されているブローカーのバージョン。

適用可能なオブジェクト

ConnectionFactory, TopicConnectionFactory, Topic, XAConnectionFactory, XATopicConnectionFactory

JMS 管理ツールのロング・ネーム: BROKERVER

JMS 管理ツールのショート・ネーム: BVER

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setBrokerVersion()
- MQConnectionFactory.getBrokerVersion()

値

V1

IBM MQ パブリッシュ/サブスクライブ・ブローカーを使用する場合、または IBM MQ Integrator、WebSphere Event Broker、WebSphere Business Integration Event Broker、WebSphere Business Integration Message Broker のいずれかのブローカーを互換モードで使用する場合。これは、TRANSPORT が BIND または CLIENT に設定されている場合のデフォルト値です。

V2

IBM MQ Integrator、WebSphere Event Broker、WebSphere Business Integration Event Broker、WebSphere Business Integration Message Broker のいずれかのブローカーをネイティブ・モードで使用する場合。これは、TRANSPORT が DIRECT または DIRECTHTTP に設定されている場合のデフォルト値です。

指定なし

ブローカーを V6 から V7 に移行した後、このプロパティを設定し、RFH2 ヘッダーが使用されないようにします。移行した後、このプロパティは関係なくなります。

CCDTURL

クライアント・チャネル定義テーブルが格納されているファイルの名前と場所を識別し、このファイルのアクセス方法を指定する Uniform Resource Locator (URL)。

適用可能なオブジェクト

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理ツールのロング・ネーム: CCDTURL

JMS 管理ツールのショート・ネーム: CCDT

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setCCDTURL()
- MQConnectionFactory.getCCDTURL()

値

null

これがデフォルト値です。

Uniform Resource Locator (URL)

注: [V 10.0.0](#) から IBM MQ 10.0.0、カスタムの SSLSocketFactory 設定を `application.security.ssl.ssl\_certificate\_validation` プロパティ **WMQ_CCDT_SSL_SOCKET_FACTORY** で追加できます。これにより、アプリケーションは独自の実装を通じて HTTPS 接続のセキュリティと証明書検証を管理できます。クライアントアプリケーションが **WMQ_CCDT_SSL_SOCKET_FACTORY** プロパティを提供する場合、プロパティ **CCDTURL** は必須となり、それが欠落している場合はエラーメッセージがクライアントアプリケーションに返される。詳細については、[/ IBM MQ classes for JMS 内のカスタマイズされた SSLSocketFactoryJakarta Messaging の提供を参照してください。](#)

CCSID

接続ファクトリにおいて、このプロパティはキューマネージャとの内部データフローに使用するコード化文字セット ID (CCSID) を指定します。CCSID は、チャネル名やキューマネージャ名などの文字データ型について、キューマネージャへの通信フローにおけるデータの符号化方法を制御します。これらのフローは固定幅データ構造を使用するため、最大許容フィールド長を使用するには、指定された CCSID 値がシングルバイト文字セットでなければならない。宛先に対して、プロパティは、その宛先に送信される MapMessages、StreamMessages、および TextMessages の文字列データをエンコードするために使用する CCSID を定義します。

注: 通常、接続ファクトリに対してこのプロパティを変更する必要はありません。

適用可能なオブジェクト

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, Queue, Topic, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理ツールのロング・ネーム: CCSID

JMS 管理ツールのショート・ネーム: CCS

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setCCSID()

- MQConnectionFactory.getCCSID()

値

819

接続ファクトリのデフォルト値。

1208

宛先のデフォルト値。

任意の正整数

関連概念

[JMS メッセージ変換](#)

CERTVALPO

TLS 接続に使用する証明書検証ポリシー。

適用可能なオブジェクト

ConnectionFactory

JMS 管理ツールの長い名前: CERTVALPO

JMS 管理ツール略称: CVAP

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setCertificateValPolicy()
- MQConnectionFactory.getCertificateValPolicy()

値

0/ANY

これは、接続ファクトリーのデフォルト値です。

2 / なし

このポリシーには証明書検証が含まれていません。クライアントアプリケーションでのみ設定可能です。

関連情報

[IBM MQ での証明書妥当性検査ポリシーの構成](#)

CHANNEL

使用されているクライアント接続チャンネルの名前。

適用可能なオブジェクト

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理ツールのロング・ネーム: CHANNEL

JMS 管理ツールのショート・ネーム: CHAN

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setChannel()

- MQConnectionFactory.getChannel()

値

SYSTEM.DEF.SVRCONN

これがデフォルト値です。

任意の有効なストリング

CLEANUP

BROKER または MIGRATE サブスクリプション・ストアのクリーンアップ・レベル。

適用可能なオブジェクト

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

JMS 管理ツールのロング・ネーム: CLEANUP

JMS 管理ツールのショート・ネーム: CL

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setCleanupLevel()
- MQConnectionFactory.getCleanupLevel()

値

SAFE

安全なクリーンアップを使用します。これがデフォルト値です。

ASPROP

Java コマンド行で設定されたプロパティに応じて、安全なクリーンアップ、強力なクリーンアップのいずれかを使用するか、またはクリーンアップを使用しません。

NONE

クリーンアップを使用しません。

strong

強力なクリーンアップを使用します。

CLEANUPINT

パブリッシュ/サブスクライブ・クリーンアップ・ユーティリティーのバックグラウンド実行の間隔 (ミリ秒)。

適用可能なオブジェクト

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

JMS 管理ツールのロング・ネーム: CLEANUPINT

JMS 管理ツールのショート・ネーム: CLINT

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setCleanupInterval()
- MQConnectionFactory.getCleanupInterval()

値

3600000

これがデフォルト値です。

任意の正整数

CONNECTIONNAMELIST

TCP/IP 接続名のリスト。このリストは、再接続が再試行されるたびに並び替えが行われます。

適用可能なオブジェクト

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory

JMS 管理ツールのロング・ネーム: CONNECTIONNAMELIST

JMS 管理ツールのショート・ネーム: CNLIST

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setconnectionNameList()
- MQConnectionFactory.getconnectionNameList()

値

HOSTNAME(PORT) のコンマ区切りリスト。HOSTNAME には、DNS 名または IP アドレスのいずれかを指定できます。

デフォルトの PORT は 1414 です。

CLIENTRECONNECTOPTIONS

再接続を制御するオプション。

適用可能なオブジェクト

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory

JMS 管理ツールのロング・ネーム: CLIENTRECONNECTOPTIONS

JMS 管理ツールのショート・ネーム: CROPT

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setClientReconnectOptions()
- MQConnectionFactory.getClientReconnectOptions()

値

QMGR

アプリケーションは、元の接続先と同じキュー・マネージャーに再接続できます。

接続名リストに指定されているように、アプリケーションが接続しようとしているキュー・マネージャーの QMID が、元の接続先のキュー・マネージャーの QMID と異なる場合は、理由コード MQRC_RECONNECT_QMID_MISMATCH のエラーが返されます。

アプリケーションが再接続可能であるが、IBM MQ classes for JMS アプリケーションと、それが最初に接続を確立したキュー・マネージャーとの間に類縁性がある場合には、この値を使用してください。

アプリケーションを可用性の高いキュー・マネージャーのスタンバイ・インスタンスに自動的に再接続したい場合、この値を選択してください。

この値をプログラムで使用するには、定数 `WMQConstants.WMQ_CLIENT_RECONNECT_Q_MGR` を使用します。

ANY

アプリケーションは、接続名リストに指定されているどのキュー・マネージャーにも再接続できます。

この再接続オプションは、IBM MQ classes for JMS アプリケーションと、それが最初に接続を確立したキュー・マネージャーとの間に類縁性がない場合にのみ使用してください。

この値をプログラムから使用するには、定数 `WMQConstants.WMQ_CLIENT_RECONNECT` を使用します。

DISABLED

アプリケーションは再接続されません。

この値をプログラムで使用するには、定数 `WMQConstants.WMQ_CLIENT_RECONNECT_DISABLED` を使用します。

ASDEF

アプリケーションが自動的に再接続されるかどうかは、IBM MQ チャンネル属性 `DefReconnect` によって決まります。

これがデフォルト値です。

この値をプログラムから使用するには、定数 `WMQConstants.WMQ_CLIENT_RECONNECT_AS_DEF` を使用します。

CLIENTRECONNECTTIMEOUT

再接続の再試行が行われなくなるまでの時間。

適用可能なオブジェクト

`ConnectionFactory`, `QueueConnectionFactory`, `TopicConnectionFactory`

JMS 管理ツールのロング・ネーム: `CLIENTRECONNECTTIMEOUT`

JMS 管理ツールのショート・ネーム: `CRT`

プログラムによるアクセス

セッター/ゲッター

- `MQConnectionFactory.setClientReconnectTimeout()`
- `MQConnectionFactory.setClientReconnectTimeout()`

値

インターバル (秒単位)。デフォルトは 1800 (30 分) です。

CLIENTID

クライアント ID は、永続サブスクリプション用のアプリケーション接続を一意的に識別するために使用されます。

適用可能なオブジェクト

`ConnectionFactory`, `QueueConnectionFactory`, `TopicConnectionFactory`, `XAConnectionFactory`, `XAQueueConnectionFactory`, `XATopicConnectionFactory`

JMS 管理ツールのロング・ネーム: CLIENTID

JMS 管理ツールのショート・ネーム: CID

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setClientId()
- MQConnectionFactory.getClientId()

値

null

これがデフォルト値です。

任意の有効なストリング

CLONESUPP

1 つの永続トピック・サブスクライバーの複数のインスタンスを同時に稼働できるかどうか。

適用可能なオブジェクト

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

JMS 管理ツールのロング・ネーム: CLONESUPP

JMS 管理ツールのショート・ネーム: CLS

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setCloneSupport()
- MQConnectionFactory.getCloneSupport()

値

DISABLED

一度に稼働できる永続トピック・サブスクライバーのインスタンスは 1 つだけです。これがデフォルト値です。

ENABLED

1 つの永続トピック・サブスクライバーの 2 つ以上のインスタンスを同時に稼働できますが、各インスタンスを別々の Java 仮想マシン (JVM) で稼働させる必要があります。

COMPHDR

接続のヘッダー・データを圧縮するために使用できる技法のリスト。

適用可能なオブジェクト

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

JMS 管理ツールのロング・ネーム: COMPHDR

JMS 管理ツールのショート・ネーム: HC

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setHdrCompList()
- MQConnectionFactory.getHdrCompList()

値

NONE

これがデフォルト値です。

SYSTEM

RLE メッセージ・ヘッダーの圧縮が実行される

COMPMSG

接続のメッセージ・データを圧縮するために使用できる技法のリスト。

適用可能なオブジェクト

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理ツールのロング・ネーム: COMPMSG

JMS 管理ツールのショート・ネーム: MC

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setMsgCompList()
- MQConnectionFactory.getMsgCompList()

値

NONE

これがデフォルト値です。

ブランク文字で区切られた以下の値の 1 つ以上のリスト。

RLE ZLIBFAST ZLIBHIGH LZ4FAST LZ4HIGH

CONNOPT

バインディング・トランスポートを使用する IBM MQ classes for JMS アプリケーションが、どのようにキュー・マネージャーに接続するかを制御します。

適用可能なオブジェクト

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory.

JMS 管理ツールのロング・ネーム: CONNOPT

JMS 管理ツールのショート・ネーム: CNOPT

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setMQConnectionOptions()
- MQConnectionFactory.getMQConnectionOptions()

値

STANDARD

アプリケーションとキュー・マネージャーの間のバインディングの性質は、キュー・マネージャーの *DefaultBindType* 属性の値によって異なります。STANDARD という値は、IBM MQ *ConnectOption* MQCNO_STANDARD_BINDING にマップします。

SHARED

アプリケーションとローカル・キュー・マネージャー・エージェントは、別個の実行単位で実行されますが、いくつかのリソースを共有します。この値は、IBM MQ *ConnectOption* MQCNO_SHARED_BINDING にマップします。

ISOLATED

アプリケーションとローカル・キュー・マネージャー・エージェントは、別個の実行単位で実行され、リソースを共有しません。ISOLATED という値は、IBM MQ *ConnectOption* MQCNO_ISOLATED_BINDING にマップします。

FASTPATH

アプリケーションとローカル・キュー・マネージャー・エージェントは、同じ実行単位で実行されます。この値は、IBM MQ *ConnectOption* MQCNO_FASTPATH_BINDING にマップします。

SERIALQM

アプリケーションは、キュー・マネージャーの有効範囲内で接続タグの排他使用を要求します。この値は、IBM MQ *ConnectOption* MQCNO_SERIALIZE_CONN_TAG_Q_MGR にマップします。

SERIALQSG

アプリケーションは、キュー・マネージャーが属しているキュー共有グループの有効範囲内で接続タグの排他使用を要求します。SERIALQSG という値は、IBM MQ *ConnectOption* MQCNO_SERIALIZE_CONN_TAG_QSG にマップします。

RESTRICTQM

アプリケーションは接続タグの共有を要求します。ただし、キュー・マネージャーの有効範囲内での接続タグの共有には制約があります。この値は、IBM MQ *ConnectOption* MQCNO_RESTRICT_CONN_TAG_Q_MGR にマップします。

RESTRICTQSG

アプリケーションは接続タグの共有を要求します。ただし、キュー・マネージャーが属しているキュー共有グループの有効範囲内での接続タグの共有には制約があります。この値は、IBM MQ *ConnectOption* MQCNO_RESTRICT_CONN_TAG_QSG にマップします。

IBM MQ 接続オプションの詳細については、[MQCONNEX 呼び出しを使用したキュー・マネージャーへの接続](#)を参照してください。

CONNTAG

アプリケーションがキュー・マネージャーに接続されているときに、キュー・マネージャーが、作業単位の範囲内でアプリケーションによって更新されたリソースと関連付けるタグ。

適用可能なオブジェクト

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理ツールのロング・ネーム: CONNTAG

JMS 管理ツールのショート・ネーム: CNTAG

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setConnTag()
- MQConnectionFactory.getConnTag()

値

128 個の要素のバイト配列。要素はそれぞれ **0** です。
これがデフォルト値です。

任意のストリング

この値は、128 バイトよりも長い場合、切り捨てられます。

説明

保管オブジェクトの説明。

適用可能なオブジェクト

ConnectionFactory、QueueConnectionFactory、TopicConnectionFactory、Queue、Topic、XAConnectionFactory、XAQueueConnectionFactory、XATopicConnectionFactory

JMS 管理ツールのロング・ネーム: DESCRIPTION

JMS 管理ツールのショート・ネーム: DESC

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setDescription()
- MQConnectionFactory.getDescription()

値

null

これがデフォルト値です。

任意の有効なストリング

DIRECTAUTH

ブローカーへのリアルタイム接続で TLS 認証が使用されるかどうか。

適用可能なオブジェクト

ConnectionFactory、TopicConnectionFactory

JMS 管理ツールのロング・ネーム: DIRECTAUTH

JMS 管理ツールのショート・ネーム: DAUTH

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setDirectAuth()
- MQConnectionFactory.getDirectAuth()

値

BASIC

認証、ユーザー名認証またはパスワード認証なし。これがデフォルト値です。

CERTIFICATE

公開鍵証明書の認証。

ENCODING

メッセージがこの宛先に送信されたときのメッセージ本体の数値データの表記方法。このプロパティは、2 進整数、パック 10 進数、および浮動小数点数の表記を指定します。

適用可能なオブジェクト

キュー、トピック

JMS 管理ツールのロング・ネーム: ENCODING

JMS 管理ツールのショート・ネーム: ENC

プログラムによるアクセス

セッター/ゲッター

- `MQDestination.setEncoding()`
- `MQDestination.getEncoding()`

値

ENCODING プロパティ

ENCODING プロパティが取ることのできる有効値は、以下の 3 つのサブプロパティから構成されます。

整数エンコード

NORMAL または REVERSED のいずれか。

10 進数エンコード

NORMAL または REVERSED のいずれか。

浮動小数点エンコード

IEEE normal、IEEE reversed、または z/OS があります

ENCODING プロパティは、次の構文で、3 文字のストリングとして表されます。

```
{N|R}{N|R}{N|R|3}
```

このストリングにおける各部の意味は以下のとおりです。

- N は NORMAL
- R は REVERSED
- 3 は z/OS
- 最初の文字は整数エンコード
- 2 番目の文字は 10 進数エンコード
- 3 番目の文字は浮動小数点エンコード

これらの文字の組み合わせで、ENCODING プロパティには 12 の値を設定できます。

これらに加えて、NATIVE というストリングもあります。この値を指定すると、Java プラットフォームに合ったエンコード値が設定されます。

有効な ENCODING の組み合わせの例を以下に示します。

```
ENCODING(NNR)  
ENCODING(NATIVE)  
ENCODING(RR3)
```

EXPIRY

宛先に送られたメッセージの有効期限が切れる経過時間。

適用可能なオブジェクト

キュー、トピック

JMS 管理ツールのロング・ネーム: EXPIRY

JMS 管理ツールのショート・ネーム: EXP

プログラムによるアクセス

セッター/ゲッター

- `MQDestination.setExpiry()`
- `MQDestination.getExpiry()`

値

APP

有効期限を JMS アプリケーションで定義できる。これがデフォルト値です。

UNLIM

有効期限を設けない。

0

有効期限を設けない。

有効期限を表す正整数 (ミリ秒)

FAILIFQUIESCE

このプロパティは、キュー・マネージャーが静止状態の場合、またはアプリケーションが **CLIENT** トランSPORTを使用してキュー・マネージャーに接続していて、アプリケーションが使用しているチャンネルが **STOP CHANNEL** や **STOP CHANNEL MODE(QUIESCE)** MQSC コマンドなどを使用して静止状態になっている場合に、特定のメソッドの呼び出しが失敗するかどうかを決定します。

適用可能なオブジェクト

`ConnectionFactory`、`QueueConnectionFactory`、`TopicConnectionFactory`、`Queue`、`Topic`、`XAConnectionFactory`、`XAQueueConnectionFactory`、`XATopicConnectionFactory`

JMS 管理ツールのロング・ネーム: FAILIFQUIESCE

JMS 管理ツールのショート・ネーム: FIQ

プログラムによるアクセス

セッター/ゲッター

- `MQConnectionFactory.setFailIfQuiesce()`
- `MQConnectionFactory.getFailIfQuiesce()`

値

YES

キュー・マネージャーが静止状態の場合や、キュー・マネージャーに接続するために使用中のチャンネルが静止状態になっている場合に、特定のメソッドの呼び出しが失敗します。これらのいずれかの状態をアプリケーションが検出すると、アプリケーションは現在のタスクを完了して接続を切断し、キュー・マネージャーまたはチャンネル・インスタンスが停止できるようにします。これがデフォルト値です。

NO

キュー・マネージャーやキュー・マネージャーに接続するために使用中のチャンネルが静止状態になっていても、それが理由でメソッドの呼び出しが失敗することはありません。この値を指定した場合、アプリケーションはキュー・マネージャーまたはチャンネルが静止状態であることを検出できません。アプリケーションは、このキュー・マネージャーに対してオペレーションを実行し続けることができるため、キュー・マネージャーを停止できません。

HOSTNAME

キュー・マネージャーとの接続の場合、キュー・マネージャーが稼働しているシステムのホスト名または IP アドレス。ブローカーとのリアルタイム接続の場合、ブローカーが稼働しているシステムのホスト名または IP アドレス。

適用可能なオブジェクト

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理ツールのロング・ネーム: HOSTNAME

JMS 管理ツールのショート・ネーム: HOST

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setHostName()
- MQConnectionFactory.getHostName()

値

localhost

これがデフォルト値です。

任意の有効なストリング

LOCALADDRESS

キュー・マネージャーとの接続場合、このプロパティは、使用されるローカル・ネットワーク・インターフェース、あるいは使用されるローカル・ポートまたはローカル・ポートの範囲のいずれかを指定します。

適用可能なオブジェクト

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理ツールのロング・ネーム: LOCALADDRESS

JMS 管理ツールのショート・ネーム: LA

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setLocalAddress()
- MQConnectionFactory.getLocalAddress()

値

"" (空ストリング)

これがデフォルト値です。

[ip-addr][(low-port[,high-port])] 形式のストリングです。

例えば、次のとおりです。

192.0.2.0

アドレス 192.0.2.0 にローカルにバインドするチャネル。

192.0.2.0(1000)

アドレス 192.0.2.0 にローカルにバインドし、ポート 1000 を使うチャネル。

192.0.2.0(1000,2000)

アドレス 192.0.2.0 にローカルにバインドし、1000 から 2000 の範囲のポートを使うチャネル。

(1000)

ポート 1000 にローカルにバインドするチャネル。

(1000,2000)

1000 から 2000 の範囲のポートにバインドするチャネル。

IP アドレスの代わりにホスト名を指定することができます。ブローカーとのリアルタイム接続の場合、このプロパティは、マルチキャストが使用される場合にのみ使用され、プロパティの値にはポート番号またはポート番号の範囲を指定できません。この場合、このプロパティの有効な値は、ヌル、IP アドレス、またはホスト名のみです。

MAPNAMESTYLE

互換性スタイルを MapMessage エlement 名に使用できます。

適用可能なオブジェクト

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理ツールのロング・ネーム: MAPNAMESTYLE

JMS 管理ツールのショート・ネーム: MNST

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setMapNameStyle()
- MQConnectionFactory.getMapNameStyle()

値

STANDARD

標準の com.ibm.jms.JMSMapMessage Element の命名形式が使用されます。これがデフォルト値であり、Java の識別子としては無効な値も Element 名として使用できます。

COMPATIBLE

以前の com.ibm.jms.JMSMapMessage Element の命名形式が使用されます。Java の識別子として有効な値だけを Element 名として使用できます。これが必要なのは、マップ・メッセージを、IBM MQ classes for JMS の 5.3 より前のバージョンを使用するアプリケーションに送信する場合だけです。

MAXBUFFSIZE

アプリケーションによって処理されるのを待つ間に、内部のメッセージ・バッファーに格納できる受信メッセージの最大数。このプロパティは、TRANSPORT の値が DIRECT または DIRECTHTTP の場合にのみ適用されます。

適用可能なオブジェクト

ConnectionFactory, TopicConnectionFactory

JMS 管理ツールのロング・ネーム: MAXBUFFSIZE

JMS 管理ツールのショート・ネーム: MBSZ

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setMaxBufferSize()
- MQConnectionFactory.getMaxBufferSize()

値

1000

これがデフォルト値です。

任意の正整数

MDREAD

このプロパティは、JMS アプリケーションが MQMD フィールドの値を抽出できるかどうかを決定します。

適用可能なオブジェクト

JMS 管理ツールのロング・ネーム: MDREAD

JMS 管理ツールのショート・ネーム: MDR

プログラムによるアクセス

セッター/ゲッター

- MQDestination.setMQMDReadEnabled()
- MQDestination.getMQMDReadEnabled()

値

NO

メッセージの送信時に、送信されたメッセージの JMS_IBM_MQMD* プロパティは、MQMD での更新済みのフィールド値を反映するように更新されません。メッセージを受信するときに、送信側が JMS_IBM_MQMD* プロパティの一部またはすべてを設定した場合でも、受信されたメッセージでそのプロパティを使用できません。これは、管理ツールのデフォルト値です。

プログラムの場合は False を使用します。

はい

メッセージの送信時に、送信されたメッセージの JMS_IBM_MQMD* プロパティはすべて、送信者が明示的に設定しなかったプロパティを含め、MQMD での更新済みのフィールド値を反映するように更新されます。メッセージを受信するときに、受信されたメッセージで、送信者が明示的に設定しなかったプロパティを含め、すべての JMS_IBM_MQMD* プロパティを使用できます。

プログラムの場合は True を使用します。

MDWRITE

このプロパティは、JMS アプリケーションが MQMD フィールドの値を設定できるかどうかを決定します。

適用可能なオブジェクト

キュー、トピック

JMS 管理ツールのロング・ネーム: MDWRITE

JMS 管理ツールのショート・ネーム: MDR

プログラムによるアクセス

セッター/ゲッター

- `MQDestination.setMQMDWriteEnabled()`
- `MQDestination.getMQMDWriteEnabled()`

値

NO

JMS_IBM_MQMD* プロパティはすべて無視され、その値は基礎となる MQMD 構造にコピーされません。これは、管理ツールのデフォルト値です。

プログラムの場合は False を使用します。

YES

JMS_IBM_MQMD* プロパティは処理されます。それらの値は基礎となる MQMD 構造にコピーされます。

プログラムの場合は True を使用します。

MDMSGCTX

JMS アプリケーションによって、メッセージ・コンテキストのどのレベルが設定されるか。このプロパティが有効となるためには、アプリケーションが適切なコンテキスト権限を持って実行されていなければなりません。

適用可能なオブジェクト

JMS 管理ツールのロング・ネーム: MDMSGCTX

JMS 管理ツールのショート・ネーム: MDCTX

プログラムによるアクセス

セッター/ゲッター

- `MQDestination.setMQMDMessageContext()`
- `MQDestination.getMQMDMessageContext()`

値

デフォルト

MQOPEN API 呼び出しおよび MQPMO 構造は、メッセージ・コンテキスト・オプションを明示的に指定しません。これは、管理ツールのデフォルト値です。

プログラムの場合は WMQ_MDCTX_DEFAULT を使用します。

SET_IDENTITY_CONTEXT

MQOPEN API 呼び出しは、メッセージ・コンテキスト・オプション MQOO_SET_IDENTITY_CONTEXT を指定し、MQPMO 構造は MQPMO_SET_IDENTITY_CONTEXT を指定します。

プログラムの場合は WMQ_MDCTX_SET_IDENTITY_CONTEXT を使用します。

SET_ALL_CONTEXT

MQOPEN API 呼び出しは メッセージ・コンテキスト・オプション MQOO_SET_ALL_CONTEXT を指定し、MQPMO 構造は MQPMO_SET_ALL_CONTEXT を指定します。

プログラムの場合は WMQ_MDCTX_SET_ALL_CONTEXT を使用します。

MSGBATCHSZ

非同期メッセージ送達を使用する際に、1つのパケット内のキューから取得するメッセージの最大数。

適用可能なオブジェクト

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理ツールのロング・ネーム: MAXBUFFSIZE

JMS 管理ツールのショート・ネーム: MBSZ

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setMsgBatchSize()
- MQConnectionFactory.getMsgBatchSize()

値

10

これがデフォルト値です。

任意の正整数

MSGBODY

JMS アプリケーションが IBM MQ メッセージの MQRFH2 にメッセージ・ペイロードの一部としてアクセスするかどうかを決定します。

適用可能なオブジェクト

キュー、トピック

JMS 管理ツールのロング・ネーム: WMQ_MESSAGE_BODY

JMS 管理ツールのショート・ネーム: MBODY

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setMessageBodyStyle()
- MQConnectionFactory.getMessageBodyStyle()

値

UNSPECIFIED

送信の際に、IBM MQ classes for JMS が MQRFH2 ヘッダーを生成して組み込むかどうかは、WMQ_TARGET_CLIENT の値によって異なります。受信の際には、JMS の値に従って作動します。

JMS

送信の際に、IBM MQ classes for JMS は自動的に MQRFH2 ヘッダーを生成して IBM MQ メッセージに組み込みます。

受信の際に、IBM MQ classes for JMS は、MQRFH2 (存在する場合) の値に従って JMS メッセージ・プロパティを設定します。MQRFH2 を JMS メッセージ本体の一部として表示しません。

MQ

送信の際に、IBM MQ classes for JMS は MQRFH2 を生成しません。

受信の際に、IBM MQ classes for JMS は MQRFH2 を JMS メッセージ本体の一部として表示します。

MSGRETENTION

入力キューにある未配布メッセージを接続のコンシューマーに保持させるかどうか。

適用可能なオブジェクト

ConnectionFactory, QueueConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory,

JMS 管理ツールのロング・ネーム: MSGRETENTION

JMS 管理ツールのショート・ネーム: MRET

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setMessageRetention()
- MQConnectionFactory.getMessageRetention()

値

はい

未配布メッセージは入力キューに残ります。これがデフォルト値です。

いいえ

未配布メッセージは、その後処理オプションに従って処理されます。

MSGSELECTION

メッセージ選択を IBM MQ classes for JMS またはブローカーのどちらが行うかを決定します。

TRANSPORT の値が DIRECT の場合、メッセージ選択は常にブローカーにより実行され、MSGSELECTION の値は無視されます。ブローカーによるメッセージ選択は、BROKERVER の値が V1 の場合、サポートされません。

適用可能なオブジェクト

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

JMS 管理ツールのロング・ネーム: MSGSELECTION

JMS 管理ツールのショート・ネーム: MSEL

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setMessageSelection()
- MQConnectionFactory.getMessageSelection()

値

CLIENT

IBM MQ classes for JMS がメッセージ選択を行う。これがデフォルト値です。

BROKER

ブローカーがメッセージ選択を行う。

MULTICAST

ブローカーとのリアルタイム接続でマルチキャストを使用可能にする場合、また、使用可能に設定されているときは、ブローカーからメッセージ・コンシューマーへのメッセージの送達にマルチキャストを使用する厳密な方法を指定する場合。このプロパティを指定しても、メッセージ・プロデューサーによるブローカーへのメッセージの送信方法に影響はありません。

適用可能なオブジェクト

ConnectionFactory、TopicConnectionFactory、Topic

JMS 管理ツールのロング・ネーム: MULTICAST

JMS 管理ツールのショート・ネーム: MCAST

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setMulticast()
- MQConnectionFactory.getMulticast()

値

DISABLED

マルチキャスト・トランスポートを使用したメッセージ・コンシューマーへのメッセージの送達は行われません。これは、ConnectionFactory および TopicConnectionFactory オブジェクトのデフォルト値です。

ASCF

メッセージは、メッセージ・コンシューマーに関連付けられた接続ファクトリーのマルチキャスト設定に応じて、メッセージ・コンシューマーに送達されます。接続ファクトリーのマルチキャスト設定は、メッセージ・コンシューマーの作成時に確認されます。この値は、トピック・オブジェクトにのみ妥当で、トピック・オブジェクトのデフォルト値です。

ENABLED

このトピックがブローカーでマルチキャスト用に構成されている場合、メッセージはマルチキャスト・トランスポートを使用してメッセージ・コンシューマーに送達されます。信頼性の高いマルチキャストに合わせてトピックを構成した場合は、信頼性の高いサービス品質が使用されます。

RELIABLE

このトピックがブローカーで信頼性の高いマルチキャスト用に構成されている場合、メッセージは信頼性の高いサービス品質のマルチキャスト・トランスポートを使用してメッセージ・コンシューマーに送達されます。信頼性の高いマルチキャストに合わせてトピックを構成しなかった場合は、このトピックに対してメッセージ・コンシューマーを作成することはできません。

NOTR

トピックがブローカーでマルチキャスト用に構成されている場合、メッセージはマルチキャスト・トランスポートを使用してメッセージ・コンシューマーに送達されます。信頼性の高いマルチキャストに合わせてトピックを構成した場合でも、信頼性の高いサービス品質は使用されません。

OPTIMISTICPUBLICATION

このプロパティは、IBM MQ classes for JMS が、メッセージを公開したパブリッシャーに制御を即座に戻すか、または呼び出しに関連するすべての処理を完了し、結果をパブリッシャーに報告できた後にのみ制御を戻すかを決定します。

適用可能なオブジェクト

ConnectionFactory, TopicConnectionFactory

JMS 管理ツールのロング・ネーム: OPTIMISTICPUBLICATION

JMS 管理ツールのショート・ネーム: OPTPUB

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setOptimisticPublication()
- MQConnectionFactory.getOptimisticPublication()

値

NO

パブリッシャーがメッセージを公開すると、IBM MQ classes for JMS は、呼び出しに関連するすべての処理を完了し、結果をパブリッシャーに報告できるようになるまでパブリッシャーに制御を戻しません。これがデフォルト値です。

YES

パブリッシャーがメッセージを公開すると、IBM MQ classes for JMS は制御をパブリッシャーに即座に戻します。その後、呼び出しに関連するすべての処理を完了し、結果をパブリッシャーに報告できるようになります。IBM MQ classes for JMS が結果を報告するのは、パブリッシャーがメッセージをコミットした場合のみです。

OUTCOMENOTIFICATION

このプロパティは、IBM MQ classes for JMS が、メッセージを確認したかコミットした直後のサブスクライバーに制御を即座に戻すか、または呼び出しに関連するすべての処理を完了し、結果をサブスクライバーに報告できた後にのみ制御を戻すかを決定します。

適用可能なオブジェクト

ConnectionFactory, TopicConnectionFactory

JMS 管理ツールのロング・ネーム: OUTCOMENOTIFICATION

JMS 管理ツールのショート・ネーム: NOTIFY

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setOutcomeNotification()
- MQConnectionFactory.getOutcomeNotification()

値

YES

サブスクライバーがメッセージを確認するかコミットしても、IBM MQ classes for JMS は、呼び出しに関連するすべての処理を完了し、結果をサブスクライバーに報告できるようになるまでサブスクライバーに制御を戻しません。これがデフォルト値です。

NO

サブスクライバーがメッセージを確認するかコミットすると、IBM MQ classes for JMS は制御をサブスクライバーに即座に戻します。その後、呼び出しに関連するすべての処理を完了し、結果をサブスクライバーに報告できるようになります。

PERSISTENCE

宛先に送信されたメッセージの持続性。

適用可能なオブジェクト

キュー、トピック

JMS 管理ツールのロング・ネーム: PERSISTENCE

JMS 管理ツールのショート・ネーム: PER

プログラムによるアクセス

セッター/ゲッター

- MQDestination.setPersistence()
- MQDestination.getPersistence()

値

APP

JMS アプリケーションによって持続性を定義する。これがデフォルト値です。

QDEF

キューのデフォルトの値から持続性を決定する。

PERS

メッセージに持続性を与える

NON

メッセージに持続性を与えない

HIGH

この値の使用方法について詳しくは、[JMS 持続メッセージ](#)を参照してください。

POLLINGINT

これは、各セッション内のメッセージ・リスナーのキューに適切なメッセージがない場合に、各メッセージ・リスナーがキューからメッセージの取得を再度試みるまでの最大の時間間隔 (ミリ秒) です。セッション内のいずれのメッセージ・リスナーでも適切なメッセージがない状態が頻繁に発生する場合は、このプロパティの値を大きくすることを考えてください。このプロパティは、TRANSPORT の値が BIND または CLIENT の場合にのみ使用されます。

適用可能なオブジェクト

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理ツールのロング・ネーム: POLLINGINT

JMS 管理ツールのショート・ネーム: PINT

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setPollingInterval()

- MQConnectionFactory.getPollingInterval()

値

5000

これがデフォルト値です。

任意の正整数

PORT

キュー・マネージャーとの接続の場合、キュー・マネージャーが listen しているポートの番号。ブローカーとのリアルタイム接続の場合、ブローカーがリアルタイム接続を listen しているポートの番号。

適用可能なオブジェクト

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理ツールのロング・ネーム: PORT

JMS 管理ツールのショート・ネーム: PORT

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setPort()
- MQConnectionFactory.getPort()

値

1414

これは、TRANSPORT が CLIENT に設定されている場合のデフォルト値です。

1506

これは、TRANSPORT が DIRECT または DIRECTHTTP に設定されている場合のデフォルト値です。

任意の正整数

PRIORITY

宛先に送信されたメッセージの優先順位。

適用可能なオブジェクト

キュー、トピック

JMS 管理ツールのロング・ネーム: PRIORITY

JMS 管理ツールのショート・ネーム: PRI

プログラムによるアクセス

セッター/ゲッター

- MQDestination.setPriority()
- MQDestination.getPriority()

値

APP

JMS アプリケーションによって優先順位を定義する。これがデフォルト値です。

QDEF

キューのデフォルトの値を優先順位にする

0 から 9 の範囲の任意の整数
昇順。

PROCESSDURATION

このプロパティは、サブスクライバーが、IBM MQ classes for JMS に制御を戻す前に、受信したメッセージを素早く処理することを保証するかどうかを決定します。

適用可能なオブジェクト

ConnectionFactory, TopicConnectionFactory

JMS 管理ツールのロング・ネーム: PROCESSDURATION

JMS 管理ツールのショート・ネーム: PROC DUR

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setProcessDuration()
- MQConnectionFactory.getProcessDuration()

値

UNKNOWN

サブスクライバーは、受信したメッセージをどれだけ迅速に処理できるかについては、保証できません。これがデフォルト値です。

SHORT

サブスクライバーは、IBM MQ classes for JMS に制御を戻す前に、受信したメッセージを素早く処理することを保証します。

PROVIDERVERSION

このプロパティは、IBM MQ メッセージングの 3 つの操作モードである IBM MQ メッセージング・プロバイダー通常モード、IBM MQ メッセージング・プロバイダー通常モード (制限付き)、および IBM MQ メッセージング・プロバイダー移行モードを区別します。

IBM MQ メッセージング・プロバイダーの通常モードでは、IBM MQ キュー・マネージャーのすべての機能を使用して JMS が実装されます。このモードは、JMS 2.0 の API と機能を使用するように最適化されています。IBM MQ メッセージング・プロバイダー制限付き通常モードは JMS 2.0 API を使用しますが、共有サブスクリプション、遅延送達、非同期送信などの新機能は使用しません。

適用可能なオブジェクト

ConnectionFactory、QueueConnectionFactory、TopicConnectionFactory、XAConnectionFactory、XAQueueConnectionFactory、XATopicConnectionFactory

JMS 管理ツールのロング・ネーム: PROVIDERVERSION

JMS 管理ツールのショート・ネーム: PVER

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setProviderVersion()
- MQConnectionFactory.getProviderVersion()

値

PROVIDERVERSION プロパティは、値 8 (通常モード)、7 (制限付き通常モード)、6 (移行モード)、または **unspecified** (デフォルト値) のいずれかに設定できます。 **PROVIDERVERSION** プロパティに指定する値は、ストリングでなければなりません。 オプション 8、7、または 6 を指定する場合、次のいずれかのフォーマットでこれを行えます。

- V.R.M.F
- V.R.M
- V.R
- V

ここで、V、R、M、および F は、ゼロ以上の整数値です。 追加の R、M、および F の値は任意指定で、細かい制御が必要な場合に使用できます。 例えば、 **PROVIDERVERSION** レベル 7 を使用する場合は、 **PROVIDERVERSION=7**、 **7.0**、 **7.0.0**、または **7.0.0.0** を設定できます。

8 - 通常モード

JMS アプリケーションは、IBM MQ メッセージング・プロバイダー通常モードを使用します。 通常モードは、IBM MQ キュー・マネージャーのすべての機能を使用して JMS を実装します。 このモードは、JMS 2.0 の API と機能を使用するように最適化されています。

コマンドレベルが 800 以降のキューマネージャーに接続する場合、非同期送信、遅延配信、共有サブスクリプションなどのすべての JMS 2.0 API および機能を利用できます。

接続ファクトリの設定で指定されたキューマネージャーが以降の IBM MQ 8.0.0 バージョンでない場合、メソッド `createConnection` は例外を発生 `JMSFMQ0003` させて失敗します。

IBM MQ メッセージング・プロバイダー通常モードでは、共用会話機能を使用して、共用可能な会話の数がサーバー接続チャンネルの **SHARECNV()** プロパティによって制御されます。 このプロパティを 0 に設定した場合、IBM MQ メッセージング・プロバイダー通常モードは使用できず、`createConnection` メソッドは例外 `JMSCC5007` で失敗します。

7 - 制限付き通常モード

JMS アプリケーションは、IBM MQ メッセージング・プロバイダーの制限付き通常モードを使用します。 このモードは JMS 2.0 API を使用しますが、共用サブスクリプション、遅延送達、非同期送信などの新機能は使用しません。

PROVIDERVERSION を 7 に設定すると、IBM MQ メッセージング・プロバイダーで操作の「制限付き通常」モードのみ使用可能になります。

制限付きの通常モードで、コマンドレベルが 800 未満のキューマネージャーに接続する場合、API JMS 2.0 は使用できますが、非同期送信、遅延配信、および共有サブスクリプション機能は使用できません。

IBM MQ メッセージング・プロバイダーの「制限付き通常」モードでは、共用会話機能を使用して、共用可能な会話の数がサーバー接続チャンネルの **SHARECNV()** プロパティによって制御されます。 このプロパティを 0 に設定した場合、IBM MQ メッセージング・プロバイダーの制限付き通常モードは使用できず、`createConnection` メソッドは例外 `JMSCC5007` で失敗します。

6 - 移行モード

JMS アプリケーションは、IBM MQ メッセージング・プロバイダーの移行モードを使用します。

このモードを使用すると、バージョン IBM MQ 8.0 またはそれ以降のキューマネージャーに接続できますが、読み込み先読みやストリーミングなど、バージョン IBM MQ classes for JMS 以降のキューマネージャーの新機能は一切使用されません。

IBM MQ 8.0 以降のクライアントが IBM MQ 8.0 以降のキュー・マネージャーに接続している場合、メッセージの選択はクライアント・システムではなく、キュー・マネージャーによって行われます。

IBM MQ メッセージング・プロバイダー移行モードを指定していて、いずれかの JMS 2.0 API を使用しようとした場合、API メソッド呼び出しは例外 `JMSCC5007` で失敗します。

unspecified (デフォルト)

PROVIDERVERSION プロパティは、デフォルトで *unspecified* に設定されています。

前のバージョンの IBM MQ classes for JMS (JNDI) で作成された接続ファクトリーは、新しいバージョンの IBM MQ classes for JMS で使用されるときにこの値を引き継ぎます。使用する操作モードは、以下のアルゴリズムで決定されます。このアルゴリズムは `createConnection` メソッドが呼び出されるときに使用され、接続ファクトリーの他の特性を使用して IBM MQ メッセージング・プロバイダー通常モード、制限付き通常モード、IBM MQ メッセージング・プロバイダー移行モードのどれが必要かを決定します。

1. 最初に、IBM MQ メッセージング・プロバイダー通常モードを使用しようとします。
2. 接続されているキュー・マネージャーが IBM MQ 8.0 以降でない場合、IBM MQ メッセージング・プロバイダー制限付き通常モードを使用しようとします。
3. 接続されているキュー・マネージャーが IBM WebSphere MQ 7.0.1 以降ではない場合、接続はクローズされ、代わりに IBM MQ メッセージング・プロバイダー移行モードが使用されます。
4. サーバー接続チャンネルの **SHARECNV** プロパティが 0 に設定されている場合、接続は終了し、代わりに IBM MQ メッセージング・プロバイダー移行モードが使用されます。
5. が V1 に設定 **BROKERVER** されているか、デフォルトの未指定値になっている場合、IBM MQ メッセージングプロバイダの通常モードが引き続き使用されます。

ALTER QMGR コマンドの PSMODE パラメーターの互換性の詳細については、[ALTER QMGR](#) を参照してください。

6. **BROKERVER** が V2 に設定されている場合、実行されるアクションは、以下のように **BROKERQMGR** の値によって異なります。

- **BROKERQMGR** が ブランクである場合:

BROKERCONQ プロパティで指定されたキューを出力用に開くことが可能で (つまり、出力用の MQOPEN が成功する)、キュー・マネージャーの **PSMODE** が COMPAT または DISABLED に設定されている場合は、IBM MQ メッセージング・プロバイダー移行モードが使用されます。

- **BROKERCONQ** プロパティで指定されたキューを出力のために開くことができない場合、または **PSMODE** 属性が ENABLED に設定されている場合:

IBM MQ メッセージング・プロバイダー通常モードが使用されます。

- **BROKERQMGR** が 非ブランクである場合:

IBM MQ メッセージング・プロバイダー移行モードが使用されます。

使用している接続ファクトリーを変更できない場合は、`com.ibm.msg.client.wmq.overrideProviderVersion` プロパティを使用して、その接続ファクトリーの設定をオーバーライドできます。この指定変更は JVM 中のすべての接続ファクトリーに適用されますが、実際の接続ファクトリー・オブジェクトは変更されません。

関連タスク

[JMS PROVIDERVERSION](#) プロパティの構成

PROXYHOSTNAME

プロキシ・サーバーを介してブローカーとのリアルタイム接続を使用しているときの、プロキシ・サーバーが稼働しているシステムのホスト名または IP アドレス。

適用可能なオブジェクト

ConnectionFactory, TopicConnectionFactory

JMS 管理ツールのロング・ネーム: PROXYHOSTNAME

JMS 管理ツールのショート・ネーム: PHOST

プログラムによるアクセス

セッター/ゲッター

- `MQConnectionFactory.setProxyHostName()`
- `MQConnectionFactory.getProxyHostName()`

値

`null`

プロキシ・サーバーのホスト名。これがデフォルト値です。

PROXYPORT

プロキシ・サーバーを介してブローカーとのリアルタイム接続を使用しているときの、プロキシ・サーバーが `listen` しているポートの番号。

適用可能なオブジェクト

`ConnectionFactory`, `TopicConnectionFactory`

JMS 管理ツールのロング・ネーム: `PROXYPORT`

JMS 管理ツールのショート・ネーム: `PPORT`

プログラムによるアクセス

セッター/ゲッター

`MQConnectionFactory.setProxyPort()`

`MQConnectionFactory.getProxyPort()`

値

`443`

プロキシ・サーバーのポート番号。これがデフォルト値です。

PUBACKINT

IBM MQ classes for JMS がブローカーからの確認通知を要求するまでに、パブリッシャーによって公開されるメッセージの数。

このプロパティの値を小さくすると、IBM MQ classes for JMS による確認通知の要求頻度が増すため、パブリッシャーのパフォーマンスが低下します。この値を大きくすると、ブローカーに障害が発生した場合に IBM MQ classes for JMS が例外をスローするために要する時間が長くなります。このプロパティは、`TRANSPORT` の値が `BIND` または `CLIENT` の場合にのみ使用されます。

適用可能なオブジェクト

`ConnectionFactory`, `TopicConnectionFactory`, `XAConnectionFactory`, `XATopicConnectionFactory`

JMS 管理ツールのロング・ネーム: `PROXYPORT`

JMS 管理ツールのショート・ネーム: `PPORT`

プログラムによるアクセス

セッター/ゲッター

`MQConnectionFactory.setPubAckInterval()`

`MQConnectionFactory.getPubAckInterval()`

値

25

任意の正整数をデフォルト値にすることができます。

PUTASYNCALLOWED

このプロパティは、メッセージ・プロデューサーが非同期書き込みを使用してこの宛先にメッセージを送信することが許可されるかどうかを決定します。

適用可能なオブジェクト

キュー、トピック

JMS 管理ツールのロング・ネーム: PUTASYNCALLOWED

JMS 管理ツールのショート・ネーム: PAALD

プログラムによるアクセス

セッター/ゲッター

`MQDestination.setPutAsyncAllowed()`

`MQDestination.getPutAsyncAllowed()`

値

AS_DEST

キュー定義またはトピック定義を参照することによって、非同期書き込みが許可されるかどうかを決定します。これがデフォルト値です。

AS_Q_DEF

キュー定義を参照することによって非同期書き込みが許可されるかどうかを判別します。queue プロパティの名前は です **DefPResp (MQLONG)**。

AS_TOPIC_DEF

トピック定義を参照することによって非同期書き込みが許可されるかどうかを判別します。

NO

非同期書き込みは許可されない。

YES

非同期書き込みは許可される。

QMANAGER

接続するキュー・マネージャーの名前。

ただし、アプリケーションがクライアント・チャンネル定義テーブルを使用してキュー・マネージャーと接続する場合は、IBM MQ classes for JMS でのクライアント・チャンネル定義テーブルの使用を参照してください。

適用可能なオブジェクト

ConnectionFactory、QueueConnectionFactory、TopicConnectionFactory、Queue、XAConnectionFactory、XAQueueConnectionFactory、XATopicConnectionFactory

JMS 管理ツールのロング・ネーム: QMANAGER

JMS 管理ツールのショート・ネーム: QMGR

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setQueueManager()
- MQConnectionFactory.getQueueManager()

値

"" (空ストリング)

任意のストリングをデフォルト値にできます。

QUEUE

JMS キュー宛先の名前。これは、キュー・マネージャーで使用されるキューの名前と一致します。

適用可能なオブジェクト

キュー

JMS 管理ツールのロング・ネーム: QUEUE

JMS 管理ツールのショート・ネーム: QU

値

任意のストリング

有効な IBM MQ キュー名。

関連資料

[IBM MQ オブジェクトの命名規則](#)>

READAHEADALLOWED

このプロパティは、メッセージ・コンシューマーおよびキュー・ブラウザーが先読みを使用してこの宛先から内部バッファーへの非持続メッセージを受信する前に、それらを取得することを許可されるかどうかを決定します。

適用可能なオブジェクト

キュー、トピック

JMS 管理ツールのロング・ネーム: READAHEADALLOWED

JMS 管理ツールのショート・ネーム: RAALD

プログラムによるアクセス

セッター/ゲッター

- MQDestination.setReadAheadAllowed()
- MQDestination.getReadAheadAllowed()

値

AS_DEST

キュー定義またはトピック定義を参照することによって、先行読み取りが許可されるかどうかを決定します。これは、管理ツールでのデフォルト値です。

プログラムで WMQConstants.WMQ_READ_AHEAD_ALLOWED_AS_DEST を使用します。

AS_Q_DEF

キュー定義を参照することによって、先行読み取りが許可されるかどうかを決定します。

プログラムで WMQConstants.WMQ_READ_AHEAD_ALLOWED_AS_Q_DEF を使用します。

AS_TOPIC_DEF

トピック定義を参照することによって、先行読み取りが許可されるかどうかを決定します。
プログラムで `WMQConstants.WMQ_READ_AHEAD_ALLOWED_AS_TOPIC_DEF` を使用します。

NO

先読みは許可されない。
プログラムで `WMQConstants.WMQ_READ_AHEAD_ALLOWED_DISABLED` を使用します。

YES

先読みは許可される。
プログラムで `WMQConstants.WMQ_READ_AHEAD_ALLOWED_ENABLED` を使用します。

READAHEADCLOSEPOLICY

非同期メッセージ・リスナーに送信されるメッセージの場合、メッセージ・コンシューマーがクローズされるときに内部先読みバッファのメッセージに対して何が行われるか

適用可能なオブジェクト

キュー、トピック

JMS 管理ツールのロング・ネーム: `READAHEADCLOSEPOLICY`

JMS 管理ツールのショート・ネーム: `RACP`

プログラムによるアクセス

セッター/ゲッター

- `MQDestination.setReadAheadClosePolicy()`
- `MQDestination.getReadAheadClosePolicy()`

値

DELIVER_ALL

内部先読みバッファ内のすべてのメッセージは、戻る前にアプリケーションのメッセージ・リスナーに送達されます。これは、管理ツールでのデフォルト値です。

プログラムで `WMQConstants.WMQ_READ_AHEAD_DELIVERALL` を使用します。

DELIVER_CURRENT

現行のメッセージ・リスナーの呼び出しのみが完了して戻ります。この場合、内部先行読み取りバッファにメッセージが残る可能性があり、それらは廃棄されます。

プログラムで `WMQConstants.WMQ_READ_AHEAD_DELIVERCURRENT` を使用します。

RECEIVECCSID

キュー・マネージャー・メッセージ変換のターゲット CCSID を設定する宛先プロパティ。
`RECEIVECONVERSION` が `WMQ_RECEIVE_CONVERSION_QMGR` に設定されていない場合、値は無視されます。

適用可能なオブジェクト

キュー、トピック

JMS 管理ツールのロング・ネーム: `RECEIVECCSID`

JMS 管理ツールのショート・ネーム: `RCCS`

プログラムによるアクセス

セッター/ゲッター

- `MQDestination.setReceiveCCSID`
- `MQDestination.getReceiveCCSID`

値

WMQConstants.WMQ_RECEIVE_CC_SID_JVM_DEFAULT

0 - JVM `Charset.defaultCharset` を使用します。

1208

UTF-8

CCSID

サポートされているコード化文字セット ID。

RECEIVECONVERSION

データ変換が、キュー・マネージャーによって実行されるかどうかを決定する 宛先プロパティ。

適用可能なオブジェクト

キュー、トピック

JMS 管理ツールの ロング・ネーム: RECEIVECONVERSION

JMS 管理ツールの ショート・ネーム: RCNV

プログラムによるアクセス

セッター/ゲッター

- `MQDestination.setReceiveConversion`
- `MQDestination.getReceiveConversion`

値

WMQConstants.WMQ_RECEIVE_CONVERSION_CLIENT_MSG

1 - JMS クライアントでのみデータ変換を実行します。7.0 までと、7.0.1.5 以降のデフォルト値です。

WMQConstants.WMQ_RECEIVE_CONVERSION_QMGR

2 - クライアントにメッセージを送信する前に、キュー・マネージャーでデータ変換を実行します。
APAR IC72897 が適用されている場合を除き、V7.0 から V7.0.1.4 まで (V7.0.1.4 を含む) の デフォルト
値 (かつ唯一の値) です。

RECEIVEISOLATION

このプロパティは、サブスクライバーが、サブスクライバー・キューでコミットされていないメッセージを受信できるかどうかを決定します。

適用可能なオブジェクト

ConnectionFactory, TopicConnectionFactory

JMS 管理ツールの ロング・ネーム: RECEIVEISOLATION

JMS 管理ツールの ショート・ネーム: RCVISOL

値

COMMITTED

サブスクライバーは、コミットされているサブスクライバー・キューにあるメッセージのみを受信します。これは、管理ツールでのデフォルト値です。

プログラムで `WMQConstants.WMQ_RCVISOL_COMMITTED` を使用します。

UNCOMMITTED

サブスクライバーは、サブスクライバー・キューでコミットされていないメッセージを受信できます。

プログラムで `WMQConstants.WMQ_RCVISOL_UNCOMMITTED` を使用します。

RECEXIT

チャンネル受信出口、または連続して実行される一連の受信出口を識別します。

IBM MQ classes for JMS が受信出口を見つけるためには、追加の構成が必要となる場合があります。詳しくは、[IBM MQ classes for JMS のチャンネル出口の割り当て](#)を参照してください。

適用可能なオブジェクト

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理ツールのロング・ネーム: RECEXIT

JMS 管理ツールのショート・ネーム: RCX

プログラムによるアクセス

セッター/ゲッター

- `MQConnectionFactory.setReceiveExit()`
- `MQConnectionFactory.getReceiveExit()`

値

- `null`。これがデフォルト値です。
- コンマで区切られた 1 つ以上の項目から成るストリング。各項目は次のいずれかです。
 - `WMQReceiveExit` インターフェースを実装するクラスの名前 (Java で作成されたチャンネル受信出口の場合)。
 - 形式が `libraryName(entryPointName)` のストリング (Java で作成されていないチャンネル受信出口の場合)。

RECEXITINIT

チャンネル受信出口が呼び出されたときに、その出口に渡されるユーザー・データ。

適用可能なオブジェクト

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理ツールのロング・ネーム: RECEXITINIT

JMS 管理ツールのショート・ネーム: RCXI

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setReceiveExitInit()
- MQConnectionFactory.getReceiveExitInit()

値

null

コンマで区切られた 1 つ以上のユーザー・データ項目から成るストリング。これがデフォルト値です。

REPLYTOSTYLE

受信したメッセージの JMSReplyTo フィールドの構成方法を決定します。

適用可能なオブジェクト

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理ツールのロング・ネーム: REPLYTOSTYLE

JMS 管理ツールのショート・ネーム: RTOST

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setReplyToStyle()
- MQConnectionFactory.getReplyToStyle()

値

デフォルト

MQMD と同等です。

RFH2

RFH2 ヘッダーで指定されている値を使用します。JMSReplyTo 値が送信アプリケーションで設定されている場合はその値を使用します。

MQMD

MQMD により指定された値を使用します。この動作は、IBM WebSphere MQ 6.0.2.4 および 6.0.2.5 のデフォルトの動作と同等です。

送信アプリケーションによって設定されている JMSReplyTo 値にキュー・マネージャー名が含まれない場合、受信側キュー・マネージャーは MQMD に自身の名前を挿入します。このパラメーターを MQMD に設定した場合、使用する応答先キューは受信側キュー・マネージャー上にあります。このパラメーターを RFH2 に設定した場合、使用する応答先キューは、送信アプリケーションで最初に設定されているとおり、送信されるメッセージの RFH2 で指定されているキュー・マネージャー上にあります。

送信アプリケーションによって設定されている JMSReplyTo 値にキュー・マネージャー名が含まれる場合、MQMD と RFH2 の両方に同じ値が含まれるためこのパラメーターの値は重要でなくなります。

RESCANINT

Point-to-Point ドメインのメッセージ・コンシューマーがメッセージ・セクターを使用して受信するメッセージを選択する場合、IBM MQ classes for JMS は IBM MQ キューを検索して、このキューの MsgDeliverySequence 属性によって決定される順序で適切なメッセージを探します。

IBM MQ classes for JMS が適切なメッセージを見つけてコンシューマーに配信した後、IBM MQ classes for JMS は、キュー内の現在位置から次の適切なメッセージの検索を再開します。IBM MQ classes for JMS は、キューの終わりに到達するか、このプロパティの値で決定される時間間隔 (ミリ秒単位) が期限に達するまで、このようにしてキューの検索を続行します。いずれの場合でも、IBM MQ classes for JMS はキューの先頭に戻って検索を続行し、新たな時間間隔が開始されます。

適用可能なオブジェクト

ConnectionFactory, QueueConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory

JMS 管理ツールのロング・ネーム: RESCANINT

JMS 管理ツールのショート・ネーム: RINT

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setRescanInterval()
- MQConnectionFactory.getRescanInterval()

値

5000

任意の正整数をデフォルト値にできます。

SECEXIT

チャンネル・セキュリティ出口を識別します。

IBM MQ classes for JMS がセキュリティ出口を見つけるためには、追加の構成が必要となる場合があります。詳しくは、[IBM MQ classes for JMS のチャンネル出口の割り当て](#)を参照してください。

適用可能なオブジェクト

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理ツールのロング・ネーム: SECEXIT

JMS 管理ツールのショート・ネーム: SXC

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setSecurityExit()
- MQConnectionFactory.getSecurityExit()

値

- null。これがデフォルト値です。
- コンマで区切られた 1 つ以上の項目から成るストリング。各項目は次のいずれかです。
 - WMQSecurityExit インターフェースを実装するクラスの名前 (Java で作成されたチャンネル・セキュリティ出口の場合)。
 - 形式が *libraryName(entryPointName)* のストリング (Java で作成されていないチャンネル・セキュリティ出口の場合)。

SECEXITINIT

チャンネル・セキュリティ出口が呼び出されたときに、その出口プログラムに渡されるユーザー・データ。

適用可能なオブジェクト

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理ツールのロング・ネーム: SECEXITINIT

JMS 管理ツールのショート・ネーム: SCXI

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setSecurityExitInit()
- MQConnectionFactory.getSecurityExitInit()

値

null

任意のストリングをデフォルト値にできます。

SENDCHECKCOUNT

単一の未処理 JMS セッション内で、非同期書き込みエラーの検査が行われてから次の検査が行われるまでに許可される送信呼び出しの数。

適用可能なオブジェクト

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理ツールのロング・ネーム: SENDCHECKCOUNT

JMS 管理ツールのショート・ネーム: SCC

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setSendCheckCount()
- MQConnectionFactory.getSendCheckCount()

値

null

任意のストリングをデフォルト値にできます。

SENDEXIT

チャンネル送信出口、または連続して実行される一連の送信出口を識別します。

IBM MQ classes for JMS が送信出口を見つけるためには、追加の構成が必要となる場合があります。詳しくは、[IBM MQ classes for JMS のチャンネル出口の割り当て](#)を参照してください。

適用可能なオブジェクト

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理ツールのロング・ネーム: SENDEXIT

JMS 管理ツールのショート・ネーム: SDX

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setSendExit()
- MQConnectionFactory.getSendExit()

値

- null。これがデフォルト値です。
- コンマで区切られた 1 つ以上の項目から成るストリング。各項目は次のいずれかです。
 - WMQSendExit インターフェースを実装するクラスの名前 (Java で作成されたチャネル送信出口の場合)。
 - 形式が *libraryName(entryPointName)* のストリング (Java で作成されていないチャネル送信出口の場合)。

SENDEXITINIT

チャネル送信出口が呼び出されたときに、その出口に渡されるユーザー・データ。

適用可能なオブジェクト

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理ツールのロング・ネーム: SENDEXITINIT

JMS 管理ツールのショート・ネーム: SDXI

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setSendExitInit()
- MQConnectionFactory.getSendExitInit()

値

null

コンマで区切られた 1 つ以上のユーザー・データ項目から成る任意のストリングをデフォルト値にできます。

SHARECONVALLOWED

メッセージングプロバイダーの通常モードまたは制限 IBM MQ 付き通常モードを使用するアプリケーションにおいて、このプロパティは、接続ファクトリから作成される接続、セッション、および JMS コンテキストに対して共有会話機能が使用されるかどうかを決定します。

適用可能なオブジェクト

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理ツールのロング・ネーム: SHARECONVALLOWED

JMS 管理ツールのショート・ネーム: SCALD

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setShareConvAllowed()
- MQConnectionFactory.getShareConvAllowed()

値

YES

JMS 同一 JVM 内の接続ファクトリから作成された接続、セッション、およびコンテキストは、適切な場合、チャンネルインスタンス（TCP/IP 接続に対応する）を共有することができます。

これは、管理ツールのデフォルト値です。

プログラムの場合は `WMQConstants.WMQ_SHARE_CONV_ALLOWED_YES` を使用します。

NO

JMS コネクションファクトリから作成されるすべての JMS 接続、およびそれらの接続から作成されるすべての JMS セッションは、キューマネージャへの独自のチャンネルインスタンス（TCP/IP 接続）を持っています。

コンテキストについては JMS、接続ファクトリから最初に作成されるコンテキストが、2 つのチャンネルインスタンス（TCP/IP 接続）を作成します。最初のコンテキストから作成されるその他の JMS コンテキストには、それぞれ独自のチャンネルインスタンス（TCP/IP 接続）があります。

プログラムの場合は `WMQConstants.WMQ_SHARE_CONV_ALLOWED_NO` を使用します。

関連概念

[IBM MQ メッセージング・プロバイダー・モードの操作](#)

[JMS 用 IBM MQ クラスにおける TCP/IP 接続の共有](#)

SPARSESUBS

TopicSubscriber オブジェクトのメッセージ検索ポリシーを制御する。

適用可能なオブジェクト

ConnectionFactory, TopicConnectionFactory

JMS 管理ツールのロング・ネーム: SPARSESUBS

JMS 管理ツールのショート・ネーム: SSUBS

プログラムによるアクセス

セッター/ゲッター

- `MQConnectionFactory.setSparseSubscriptions()`
- `MQConnectionFactory.getSparseSubscriptions()`

値

NO

サブスクリプションが頻繁にマッチング・メッセージを受信する。これは、管理ツールのデフォルト値です。

プログラムの場合は `false` を使用します。

YES

サブスクリプションが頻繁にマッチング・メッセージを受信しない。この値は、サブスクリプション・キューがブラウザ用にオープンできることを必要とします。

プログラムの場合は `true` を使用します。

SSLCIPHERSUITE

TLS 接続で使用する CipherSuite。

適用可能なオブジェクト

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理ツールのロング・ネーム: SSLCIPHERSUITE

JMS 管理ツールのショート・ネーム: SCPHS

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setSSLCipherSuite()
- MQConnectionFactory.getSSLCipherSuite()

値

null

これがデフォルト値です。詳しくは、[JMS オブジェクトの TLS プロパティ](#)を参照してください。

関連資料

[SSLCIPHERSUITE オブジェクトプロパティによる TLS の有効化](#)

SSLCRL

TLS 証明書の失効を検査する CRL サーバー。

適用可能なオブジェクト

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理ツールのロング・ネーム: SSLCRL

JMS 管理ツールのショート・ネーム: SCRL

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setSSLCertStores()
- MQConnectionFactory.getSSLCertStores()

値

null

LDAP URL のスペースで区切られたリスト。これがデフォルト値です。詳しくは、[JMS オブジェクトの TLS プロパティ](#)を参照してください。

SSLFIPSREQUIRED

このプロパティは、TLS 接続が(JSSE) FIPSIBMJavaJava Secure Socket Extension プロバイダ (IBMJSSEFIPS) によってサポートされている CipherSuite を使用する必要があるかどうかを決定します。

注:

-  WindowsAIX、Linux、およびにおいて、IBM MQ は GSKit 9 IBM Crypto for C (ICC) 暗号モジュールを通じてへの準拠を実現 FIPS 140-3 しています。この FIPS 140-3 モジュールに関連する NIST 認証は、<https://csrc.nist.gov/projects/cryptographic-module-validation-program/certificate/4755> で確認できます。

- V 10.0.0 > V 10.0.0 Linux s390xFIPS 現在、.への対応は行っていません。IBM MQ このプラットフォームをご利用のお客様でサポートが必要な FIPS 場合は、以前のバージョンのままご利用ください。このプラットフォーム上で暗号モジュールが IBM Crypto for C (ICC) 認証を取得次第、今後の FixPack でサポートが FIPS 140-3 有効化されます。
- V 10.0.0 このソフトウェアに含まれる暗号 FIPS 140-3 モジュールは、2024 IBM Semeru Runtime 年 8 月に NIST の承認を受けた。IBM Semeru Runtime 11+IBM MQ 10.0.0 TLS を使用した および IBM MQ classes for Java のクライアント接続の IBM MQ classes for JMS 処理機能を、FIPS 140-3 およびに Java 8 追加しました。 <https://csrc.nist.gov/projects/cryptographic-module-validation-program/certificate/4755> この FIPS 140-3 モジュールに関連する NIST 認証は、以下で確認できます。
- Containers では IBM MQ、3.2.0 およびキューマネージャ IBM MQ Operator ーのコンテナイメージ 9.4.0.0 以降は、UBI 9 をベースとしています。FIPS 140-3 Containers における IBM MQ コンプライアンスは、現在審査中です。

V 10.0.0 が有効になっている FIPS 場合、コンテナ制御 IBM MQ プロセスでは、認定 FIPS 140-3 OpenSSL モジュールが使用されます。NIST 認証の詳細については、以下の URL をご覧ください： <https://access.redhat.com/compliance/fips>。IBM MQ コンテナイメージ内で実行されるキューマネージャは、ベースイメージのプラットフォームバージョンと同じ FIPS IBM MQ 認証レベルを持っています。

V 10.0.0 から IBM MQ 10.0.0、接続ファクトリプロパティは **SSLFIPSREQUIRED** でサポートされて IBM Semeru Runtime いません。クライアントアプリケーションによって含まれてもエラーは発生しません。プロパティが設定されているにもかかわらず、他の FIPS IBM Semeru Runtime プロパティが設定されていない場合、新しい JMSEException がスローされます。Java 11+ これにより、既存のクライアントアプリケーションがに移行し、接続を FIPS 期待してしまうという事態を回避できます。

注：これは既存のクライアントには影響せず IBMJava 8、クライアントのみに影響 IBM Semeru RuntimeJava します。

適用可能なオブジェクト

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理ツールのロング・ネーム: SSLFIPSREQUIRED

JMS 管理ツールのショート・ネーム: SFIPS

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setSSLFipsRequired()
- MQConnectionFactory.getSSLFipsRequired()

値

NO

TLS 接続では、プロバイダ（IBMJavaJSSEFIPS IBMJSSEFIPS）がサポートしていない任意の CipherSuite を使用できます。

これがデフォルト値です。プログラムでは false を使用します。

YES

TLS 接続では、IBMJSSEFIPS でサポートされている CipherSuite を使用する必要があります。プログラムでは true を使用します。

関連資料

[SSLFIPSREQUIRED を使用して、接続に認証済み CipherSpecsFIPS のみを使用するかどうかを指定する](#)

SSLPEERNAME

TLS の場合、キュー・マネージャーによって提供されるものと必ず一致する識別名 スケルトン。

適用可能なオブジェクト

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理ツールのロング・ネーム: SSLPEERNAME

JMS 管理ツールのショート・ネーム: SPEER

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setSSLPeerName()
- MQConnectionFactory.getSSLPeerName()

値

null

これがデフォルト値です。詳しくは、[JMS オブジェクトの TLS プロパティ](#)を参照してください。

関連資料

[区別名パターンを指定するための SSLPEERNAME の使用](#)

SSLRESETCOUNT

TLS の場合、暗号化に使用された秘密鍵の再ネゴシエーションの前に、接続で送信および受信したバイトの総数。

適用可能なオブジェクト

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理ツールのロング・ネーム: SSLRESETCOUNT

JMS 管理ツールのショート・ネーム: SRC

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setSSLResetCount()
- MQConnectionFactory.getSSLResetCount()

値

0

ゼロ、または 999,999,999 以下の正整数。これがデフォルト値です。詳しくは、[JMS オブジェクトの TLS プロパティ](#)を参照してください。

関連資料

[SSLRESETCOUNT を使用して秘密鍵の再ネゴシエーションタイミングを指定する](#)

STATREFRESHINT

サブスクライバーがキュー・マネージャーとの接続を失ったときに検出する長期実行トランザクションのリフレッシュの間隔 (ミリ秒)。

このプロパティは、SUBSTORE の値が QUEUE の場合にのみ使用されます。

適用可能なオブジェクト

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

JMS 管理ツールのロング・ネーム: STATREFRESHINT

JMS 管理ツールのショート・ネーム: SRI

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setStatusRefreshInterval()
- MQConnectionFactory.getStatusRefreshInterval()

値

60000

任意の正整数をデフォルト値にできます。詳しくは、[JMS オブジェクトの TLS プロパティ](#)を参照してください。

SUBSTORE

IBM MQ classes for JMS がアクティブ・サブスクリプションに関する永続データを保管する場所。

適用可能なオブジェクト

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

JMS 管理ツールのロング・ネーム: SUBSTORE

JMS 管理ツールのショート・ネーム: SS

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setSubscriptionStore()
- MQConnectionFactory.getSubscriptionStore()

値

BROKER

ブローカー・ベースのサブスクリプション・ストアを使用してサブスクリプションの詳細を保持します。これは、管理ツールのデフォルト値です。

プログラムの場合は WMQConstants.WMQ_SUBSTORE_BROKER を使用します。

MIGRATE

キュー・ベースのサブスクリプション・ストアからブローカー・ベースのサブスクリプション・ストアにサブスクリプション情報を転送します。

プログラムの場合は WMQConstants.WMQ_SUBSTORE_MIGRATE を使用します。

QUEUE

キュー・ベースのサブスクリプション・ストアを使用してサブスクリプションの詳細を保持します。

プログラムの場合は WMQConstants.WMQ_SUBSTORE_QUEUE を使用します。

SYNCPOINTALLGETS

このプロパティは、すべての取得を同期点下で実行するかどうかを 決定します。

適用可能なオブジェクト

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理ツールのロング・ネーム: SYNCPOINTALLGETS

JMS 管理ツールのショート・ネーム: SPAG

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setSyncpointAllGets()
- MQConnectionFactory.getSyncpointAllGets()

値

いいえ

これがデフォルト値です。

はい

TARGCLIENT

このプロパティは、ターゲット・アプリケーションとの情報の交換に IBM MQ RFH2 形式を使用するかどうかを決定します。

適用可能なオブジェクト

キュー、トピック

JMS 管理ツールのロング・ネーム: TARGCLIENT

JMS 管理ツールのショート・ネーム: TC

プログラムによるアクセス

セッター/ゲッター

- MQDestination.setTargetClient()
- MQDestination.getTargetClient()

値

JMS

JMS アプリケーションをメッセージのターゲットにする。これは、管理ツールのデフォルト値です。プログラムの場合は WMQConstants.WMQ_CLIENT_JMS_COMPLIANT を使用します。

MQ

JMS 以外の IBM MQ アプリケーションをメッセージのターゲットにする。プログラムの場合は WMQConstants.WMQ_CLIENT_NONJMS_MQ を使用します。

TARGCLIENTMATCHING

このプロパティは、着信メッセージの JMSReplyTo ヘッダー・フィールドで識別されるキューに送信された応答メッセージに MQRFH2 ヘッダーがあるかどうか (着信メッセージに MQRFH2 ヘッダーがある場合のみ) を決定します。

適用可能なオブジェクト

ConnectionFactory, QueueConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory

JMS 管理ツールのロング・ネーム: TARGCLIENTMATCHING

JMS 管理ツールのショート・ネーム: TCM

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setTargetClientMatching()
- MQConnectionFactory.getTargetClientMatching()

値

YES

着信メッセージに MQRFH2 ヘッダーがない場合、メッセージの JMSReplyTo ヘッダー・フィールドから派生された Queue オブジェクトの TARGCLIENT プロパティーが、MQ に送信されます。メッセージに MQRFH2 ヘッダーがある場合は、TARGCLIENT プロパティーが JMS に設定されます。これは、管理ツールのデフォルト値です。

プログラムの場合は true を使用します。

NO

着信メッセージの JMSReplyTo ヘッダー・フィールドから取得した Queue オブジェクトの TARGCLIENT プロパティーが、常に JMS に設定されます。

プログラムの場合は false を使用します。

TEMPMODEL

JMS 一時キューの作成に使用するモデル・キューの名前。

適用可能なオブジェクト

ConnectionFactory, QueueConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory

JMS 管理ツールのロング・ネーム: TEMPMODEL

JMS 管理ツールのショート・ネーム: TM

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setTemporaryModel()
- MQConnectionFactory.getTemporaryModel()

値

SYSTEM.DEFAULT.MODEL.QUEUE

任意のストリングをデフォルト値にできます。

TEMPQPREFIX

IBM MQ 動的キューの名前を形成するために使用された接頭部。

適用可能なオブジェクト

ConnectionFactory, QueueConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory

JMS 管理ツールのロング・ネーム: TEMPQPREFIX

JMS 管理ツールのショート・ネーム: TQP

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setTempQPrefix()
- MQConnectionFactory.getTempQPrefix()

値

"" (空ストリング)

接頭部として CSQ.* (z/OS の場合) および AMQ.* (その他のすべてのプラットフォームの場合) が使用されます。これらはデフォルト値です。

キュー接頭部

キュー接頭部は、IBM MQ オブジェクト記述子 (構造体 MQOD) の *DynamicQName* フィールドの内容を形成するための規則に準拠する任意のストリングですが、最後の非ブランク文字はアスタリスクでなければなりません。

TEMPTOPICPREFIX

一時トピックの作成時に、JMS は "TEMP/TEMPTOPICPREFIX/unique_id" という形式のトピック・ストリングを生成します。またはこのプロパティがデフォルト値のままになっている場合は、"TEMP/unique_id" のみ生成します。空でない TEMPTOPICPREFIX を指定すると、この接続の下で作成された一時トピックに対して、サブスクライバー用の管理されたキューを作成するための特定のモデル・キューを定義できます。

適用可能なオブジェクト

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

JMS 管理ツールのロング・ネーム: TEMPTOPICPREFIX

JMS 管理ツールのショート・ネーム: TTP

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setTempTopicPrefix()
- MQConnectionFactory.getTempTopicPrefix()

値

IBM MQ トピック・ストリングに有効な文字だけで構成される任意の非ヌル・ストリング。デフォルト値は "" (空ストリング) です。

V 10.0.0 トークンクライアント ID

JSON Web Token (JWT) プロバイダーで定義されるクライアント ID。

適用可能なオブジェクト

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理ツールの長い名前: TOKEN_CLIENT_ID

JMS 管理ツールの略称: TKCD

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setTokenClientId()
- MQConnectionFactory.getTokenClientId()

値

null

これがデフォルト値です。

任意の有効なストリング

注: 接続 **TOKEN_CLIENT_ID** ファクトリプロパティを指定する場合、接続ファクトリプロパティ **TOKEN_ENDPOINT** は必須です。

V 10.0.0 トークンクライアントシークレット

クライアントシークレット、すなわちクライアントアプリケーションとトークンプロバイダーのみが知る秘密鍵。

適用可能なオブジェクト

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理ツールの長い名前: TOKEN_CLIENT_SECRET

JMS 管理ツールの略称: TKST

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setTokenClientSecret()
- MQConnectionFactory.getTokenClientSecret()

値

null

これがデフォルト値です。

任意の有効なストリング

注: 接続 **TOKEN_CLIENT_SECRET** ファクトリプロパティを指定する場合、接続ファクトリプロパティ **TOKEN_ENDPOINT** は必須です。

V 10.0.0 トークンエンドポイント

トークンプロバイダーのエンドポイントの URI。

適用可能なオブジェクト

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理ツールの長い名前: TOKEN_ENDPOINT

JMS 管理ツールの略称: TKEP

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setTokenEndpoint()

- MQConnectionFactory.getTokenEndpoint()

値

null

これがデフォルト値です。

任意の有効なストリング

これは有効なトークンプロバイダーでなければなりません URL。

注: 接続ファクトリ **TOKEN_ENDPOINT** のプロパティを指定する場合:

- と **TOKEN_CLIENT_SECRET** **TOKEN_CLIENT_ID** プロパティは必須です。
- 接続ファクトリ **USER_AUTHENTICATION_MQCSP** のプロパティは true に設定する必要があります。
- 接続ファクトリのプロパティで **UserId** 指定してはいけません。

注: V 10.0.0 SSLSocketFactory のプロパティを使用して **TOKEN_SSL_SOCKET_FACTORY**、アプリケーションが HTTPS のエンドポイントから JWT トークンを安全に取得する方法を制御するための、カスタマイズされた設定を追加できます。 **TOKEN_SSL_SOCKET_FACTORY** クライアントアプリケーションがを提供する場合、プロパティは **TOKEN_ENDPOINT** 必須となり、それが欠落している場合は、クライアントアプリケーションにエラーメッセージが返されます。詳細については、「[カスタマイズされた SSLSocketFactory 提供](#)」を参照してください。 [IBM MQ classes for JMS /Jakarta Messaging](#)。

V 10.0.0 トークン HTTPS 証明書検証ポリシー

クライアントライブラリ IBM MQ が HTTPS 接続から JWT を取得する際に実行する検証レベル。

適用可能なオブジェクト

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理ツールの長い名前: TOKENCERTVALPOL

JMS 管理ツールの略称: TCVP

プログラムによるアクセス

セッター/ゲッター

- setTokenHttpsCertValidationPolicy(int 値)
- getTokenHttpsCertValidationPolicy()

値

ANY

値は 1 です。

NONE

値は 2 です。

ホストネーム CN

値は 3 です。これがデフォルト値です。

トピック

JMS トピック宛先の名前。キュー・マネージャーは、この値をパブリケーションまたはサブスクリプションのトピック・ストリングとして使用します。

適用可能なオブジェクト

トピック

JMS 管理ツールのロング・ネーム: TOPIC

JMS 管理ツールのショート・ネーム: TOP

値

任意のストリング

有効な IBM MQ トピック・ストリングを形成するストリング。を IBM MQ メッセージングプロバイダーとして使用する場合 WebSphere Application Server、トピックが管理目的で内部的に識別される名前と一致する WebSphere Application Server 値を指定してください。

関連資料

[トピック・ストリング](#)

TRANSPORT

キュー・マネージャーまたはブローカーとの接続の性質。

適用可能なオブジェクト

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理ツールのロング・ネーム: TRANSPORT

JMS 管理ツールのショート・ネーム: TRAN

プログラムによるアクセス

セッター/ゲッター

- MQConnectionFactory.setTransportType()
- MQConnectionFactory.getTransportType()

値

BIND

バインディング・モードでのキュー・マネージャーとの接続の場合。これは、管理ツールのデフォルト値です。

このモードでは、QMANAGER プロパティで指定されたキューマネージャへのローカル接続が確立されます。ただし、CCDTURL クライアントモードプロパティも指定した場合、互換性確保のため（以前のバージョンとの互換性のため IBM MQ）、CCDTURL プロパティが優先されます。代わりに、CCDT ファイルで指定されたキューマネージャに対してクライアントモード接続が確立されます。

クライアントモード固有のその他のプロパティ（CHANNEL など）を設定した場合、それらのプロパティは無視されます。

プログラムの場合は WMQConstants.WMQ_CM_BINDINGS を使用します。

CLIENT

クライアント・モードでのキュー・マネージャーとの接続の場合。

接続は、以下のプロパティまたはプロパティの組み合わせのいずれかによって制御されます：CONNECTIONNAMELIST、HOSTNAME と PORT、または CCDTURL。

プログラムの場合は WMQConstants.WMQ_CM_CLIENT を使用します。

DIRECT

HTTP トンネル機能を使用しないブローカーとのリアルタイム接続の場合。

プログラムの場合は WMQConstants.WMQ_CM_DIRECT_TCPIP を使用します。

DIRECTHTTP

HTTP トンネル機能を使用するブローカーとのリアルタイム接続の場合。HTTP 1.0 のみがサポートされています。

プログラムの場合は `WMQConstants.WMQ_CM_DIRECT_HTTP` を使用します。

関連概念

1902 ページの『[IBM MQ classes for JMS オブジェクトのプロパティ間の依存関係](#)』
一部のプロパティの妥当性は、他のプロパティの特定の値に依存します。

WILDCARDFORMAT

このプロパティは、使用されるワイルドカード構文のバージョンを決定します。

適用可能なオブジェクト

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

JMS 管理ツールのロング・ネーム: WILDCARDFORMAT

JMS 管理ツールのショート・ネーム: WCFMT

プログラムによるアクセス

セッター/ゲッター

- `MQConnectionFactory.setWildcardFormat()`
- `MQConnectionFactory.getWildcardFormat()`

値

TOPIC_ONLY

ブローカー・バージョン 2 で使用されるトピック・レベル・ワイルドカードのみを認識します。これは、管理ツールのデフォルト値です。

プログラムの場合は `WMQConstants.WMQ_WILDCARD_TOPIC_ONLY` を使用します。

CHAR_ONLY

ブローカー・バージョン 1 で使用された文字ワイルドカードのみ認識します。

プログラムの場合は `WMQConstants.WMQ_WILDCARD_CHAR_ONLY` を使用します。

WMQ_CCDT_HTTPS_証明書検証ポリシー

クライアントライブラリ IBM MQ が HTTPS 接続から CCDT を取得する際に実行する検証レベル。

適用可能なオブジェクト

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS 管理ツールの長い名前: CCDCERTVALPOL

JMS 管理ツールの略称: CCVP

プログラムによるアクセス

セッター/ゲッター

- `setCCDTHttpsCertValidationPolicy(int 値)`
- `getCCDTHttpsCertValidationPolicy()`

値

ANY

値は 1 です。

NONE

値は 2 です。

ホストネーム CN

値は 3 です。これがデフォルト値です。

ENCODING プロパティ

ENCODING プロパティは 3 つのサブプロパティで構成され、12 とおりの組み合わせが可能です。

ENCODING プロパティが取ることのできる有効値は、以下の 3 つのサブプロパティから構成されます。

整数エンコード

NORMAL または REVERSED のいずれか。

10 進数エンコード

NORMAL または REVERSED のいずれか。

浮動小数点エンコード

IEEE normal、IEEE reversed、または z/OS があります

ENCODING プロパティは、次の構文で、3 文字のストリングとして表されます。

```
{N|R}{N|R}{N|R|3}
```

このストリングにおける各部の意味は以下のとおりです。

- N は NORMAL
- R は REVERSED
- 3 は z/OS
- 最初の文字は整数エンコード
- 2 番目の文字は 10 進数エンコード
- 3 番目の文字は浮動小数点エンコード

これらの文字の組み合わせで、ENCODING プロパティには 12 の値を設定できます。

これらに加えて、NATIVE というストリングもあります。この値を指定すると、Java プラットフォームに合ったエンコード値が設定されます。

有効な ENCODING の組み合わせの例を以下に示します。

```
ENCODING(NNR)  
ENCODING(NATIVE)  
ENCODING(RR3)
```

IBM MQ Message Service Client (XMS) for .NET リファレンス

このリファレンス・セクションでは、IBM MQ Message Service Client (XMS) for .NET (XMS .NET) クラス・インターフェースに関する情報、および XMS で定義されるオブジェクト・プロパティについて説明します。

.NET インターフェース

このセクションでは、.NET クラスのインターフェースとそのプロパティおよびメソッドについて説明します。

以下の表は、IBM.XMS namespace 内で定義されたインターフェースを要約したものです。

表 907. .NET クラス・インターフェースの要約

インターフェース	説明
1961 ページの『IBytesMessage』	バイト・メッセージとは、本体がバイトのストリームからなるメッセージです。
1970 ページの『IConnection』	Connection オブジェクトは、アプリケーションからメッセージング・サーバーへのアクティブな接続を表します。
1973 ページの『IConnectionFactory』	アプリケーションは、接続ファクトリーを使用して接続を作成します。
1974 ページの『IConnectionMetaData』	ConnectionMetaData オブジェクトは、接続に関する情報を提供します。
1975 ページの『IDestination』	宛先とは、アプリケーションがメッセージを送信する場所、またはアプリケーションがメッセージを受信する場合の送信元、あるいはその両方のことです。
1976 ページの『ExceptionListener』	アプリケーションは例外リスナーを使用して、接続の問題に関する通知を非同期に受信します。
1976 ページの『IllegalStateException』	アプリケーションがメソッドを正しくない時刻または不適当な時刻に呼び出した場合、または XMS が要求された操作に適切な状態でない場合、XMS はこの例外をスローします。
1976 ページの『InitialContext』	アプリケーションは、InitialContext オブジェクトを使用して、管理対象オブジェクトのリポジトリから取得したオブジェクト定義によってオブジェクトを作成します。
1978 ページの『InvalidClientIDException』	XMS がこの例外をスローするのは、アプリケーションが接続のクライアント ID を設定しようとしたが、そのクライアント ID が無効かまたは既に使用中である場合です。
1979 ページの『InvalidDestinationException』	XMS がこの例外をスローするのは、アプリケーションが無効な宛先を指定している場合です。
1979 ページの『InvalidSelectorException』	XMS がこの例外をスローするのは、無効な構文のメッセージ・セレクター式をアプリケーションで指定した場合です。
1979 ページの『IMapMessage』	マップ・メッセージとは、本体が名前と値のペアで構成されるメッセージです。それぞれの値に、関連付けられたデータ・タイプがあります。
1988 ページの『IMessage』	メッセージ・オブジェクトは、アプリケーションが送信または受信するメッセージを表します。 IMessage は、IMapMessage などの Message クラスのスーパークラスです。
1993 ページの『IMessageConsumer』	アプリケーションは、メッセージ・コンシューマーを使用して、宛先に送信されたメッセージを受信します。
1996 ページの『MessageEOFException』	XMS がこの例外をスローするのは、アプリケーションがバイト・メッセージの本体を読み取っているときに、XMS がバイト・メッセージ・ストリームの終端を検出した場合です。

表 907. .NET クラス・インターフェースの要約 (続き)

インターフェース	説明
1996 ページの『MessageFormatException』	XMS がこの例外をスローするのは、XMS が無効なフォーマットのメッセージを検出した場合です。
1996 ページの『IMessageListener (代行)』	アプリケーションは、メッセージ・リスナーを使用して、メッセージを非同期に受信します。
1997 ページの『MessageNotReadableException』	XMS がこの例外をスローするのは、書き込み専用になっているメッセージの本体をアプリケーションが読み取ろうとした場合です。
1997 ページの『MessageNotWritableException』	XMS がこの例外をスローするのは、読み取り専用になっているメッセージの本体にアプリケーションが書き込もうとした場合です。
1997 ページの『IMessageProducer』	アプリケーションは、メッセージ・プロデューサーを使用して、メッセージを宛先に送信します。
2002 ページの『IObjectMessage』	オブジェクト・メッセージは、シリアライズされた Java オブジェクトまたは .NET オブジェクトから構成されるメッセージのことです。
2003 ページの『IPropertyContext』	IPropertyContext は、プロパティを取得および設定するメソッドを含む抽象スーパークラスです。これらのメソッドは、その他のクラスによって継承されます。
2012 ページの『IQueueBrowser』	アプリケーションは、キュー・ブラウザーを使用して、キュー上のメッセージを参照します。その際にメッセージは除去されません。
2014 ページの『Requestor』	アプリケーションはリクエスターを使用して、要求メッセージを送信し、応答を待機して受信します。
2015 ページの『ResourceAllocationException』	XMS がこの例外をスローするのは、メソッドが必要とするリソースを XMS が割り振ることができない場合です。
2015 ページの『SecurityException』	XMS は、アプリケーションを認証するために指定されたユーザー ID とパスワードが拒否された場合に、この例外をスローします。XMS は、権限検査が不合格になり、そのためにメソッドを完了できない場合にもこの例外をスローします。
2015 ページの『ISession』	セッションとは、メッセージ送受信のための単一スレッドのコンテキストのことです。
2025 ページの『IStreamMessage』	ストリーム・メッセージとは、本体が値のストリームで構成されるメッセージです。それぞれの値に、関連付けられたデータ・タイプがあります。
2034 ページの『ITextMessage』	テキスト・メッセージとは、本体がストリングからなるメッセージです。
2035 ページの『TransactionInProgressException』	XMS がこの例外をスローするのは、トランザクションが進行中であるために無効になっている操作をアプリケーションが要求した場合です。

表 907. .NET クラス・インターフェースの要約 (続き)	
インターフェース	説明
2035 ページの『 TransactionRolledBackException 』	XMS がこの例外をスローするのは、アプリケーションが現行のトランザクションをコミットするために <code>Session.commit()</code> を呼び出したにもかかわらず、その後このトランザクションがロールバックされた場合です。
XMSC	.NET の場合、XMS プロパティの名前および値は、パブリック定数としてこのクラスに定義されます。詳しくは、2038 ページの『 XMS オブジェクトのプロパティ 』を参照してください。
2035 ページの『 XMSEException 』	XMS が .NET メソッドの呼び出しを処理しているときにエラーを検出すると、XMS は例外をスローします。例外とは、エラーに関する情報をカプセル化するオブジェクトのことです。 XMS 例外にはさまざまなタイプがあり、<code>XMSEException</code> オブジェクトは例外の 1 つのタイプにすぎません。ただし、<code>XMSEException</code> クラスは、その他の XMS 例外クラスのスーパークラスです。XMS は、<code>XMSEException</code> オブジェクト以外のタイプの例外では適切でない状態では、<code>XMSEException</code> オブジェクトをスローします。
2036 ページの『 XMSFactoryFactory 』	アプリケーションが管理対象オブジェクトを使用していない場合は、このクラスを使用して接続ファクトリー、キュー、およびトピックを作成します。

各メソッドの定義では、XMS がメソッドの呼び出しの処理中にエラーを検出した場合に戻す例外コードをリストしています。各例外コードは、その名前付き定数で表されますが、この定数には対応する例外があります。

IBytesMessage

バイト・メッセージとは、本体がバイトのストリームからなるメッセージです。

継承の階層:

```

IBM.XMS.IPropertyContext
|
+----IBM.XMS.IMessage
|
+----IBM.XMS.IBytesMessage

```

.NET プロパティ

BodyLength - 本体の長さの取得

インターフェース:

```

Int64 BodyLength
{
    get;
}

```

メッセージの本体が読み取り専用である場合に、メッセージの本体の長さ (バイト単位) を取得します。

この値は、メッセージを読み取るためのカーソルの現在の位置にかかわらず、本体の全体の長さが戻されます。

例外:

- XMSEException
- MessageNotReadableException

方法

ReadBoolean - ブール値の読み取り

インターフェース:

```
Boolean ReadBoolean();
```

バイト・メッセージ・ストリームからブール値を読み取ります。

パラメーター:

なし

戻り値:

読み取られるブール値。

例外:

- XMSEException
- MessageNotReadableException
- MessageEOFException

ReadSignedByte - バイトの読み取り

インターフェース:

```
Int16 ReadSignedByte();
```

バイト・メッセージ・ストリームから、次のバイトを符号付き 8 ビット整数として読み取ります。

パラメーター:

なし

戻り値:

読み取られるバイト。

例外:

- XMSEException
- MessageNotReadableException
- MessageEOFException

ReadBytes - バイトの読み取り

インターフェース:

```
Int32 ReadBytes(Byte[] array);  
Int32 ReadBytes(Byte[] array, Int32 length);
```

バイト・メッセージ・ストリームから、カーソルの現在位置から始まるバイトの配列を読み取ります。

パラメーター:

array (出力)

読み取られるバイトの配列を含むバッファー。呼び出しの前の、ストリームから読み取られる残りバイト数が、バッファーの長さより大きい場合、バッファーはいっぱいになります。残りバイト数の方が小さい場合は、バッファーに残りのすべてのバイトが格納され、バッファーは部分的に埋まります。

入力に NULL ポインターを指定すると、メソッドはそのバイトを読み取らずにスキップオーバーします。呼び出しの前の、ストリームから読み取られる残りバイト数が、バッファーの長さより大き

いか等しい場合、スキップされるバイト数は、バッファの長さと等しくなります。そうでない場合は、残りのすべてのバイトがスキップされます。カーソルは、バイト・メッセージ・ストリームを読み取るために次の位置に残ります。

length (入力)

バッファの長さ (バイト単位)

戻り値:

バッファに読み取るバイト数。バッファが部分的に埋まっている場合は、値はバッファの長さよりも小さく、読み取るバイトが残っていないことを示します。呼び出しの前にストリームから読み取るバイトが残っていない場合、値は `XMSC_END_OF_STREAM` になります。

入力に `NULL` ポインターを指定すると、メソッドは値を戻しません。

例外:

- `XMSEException`
- `MessageNotReadableException`

ReadChar - 文字の読み取り

インターフェース:

```
Char    ReadChar();
```

バイト・メッセージ・ストリームから、次の 2 バイトを文字として読み取ります。

パラメーター:

なし

戻り値:

読み取られる文字。

例外:

- `XMSEException`
- `MessageNotReadableException`
- `MessageEOFException`

ReadDouble - 倍精度浮動小数点数の読み取り

インターフェース:

```
Double  ReadDouble();
```

バイト・メッセージ・ストリームから、次の 8 バイトを倍精度浮動小数点数として読み取ります。

パラメーター:

なし

戻り値:

読み取られる倍精度浮動小数点数。

例外:

- `XMSEException`
- `MessageNotReadableException`
- `MessageEOFException`

ReadFloat - 浮動小数点数の読み取り

インターフェース:

```
Single  ReadFloat();
```

バイト・メッセージ・ストリームから、次の 4 バイトを浮動小数点数として読み取ります。

パラメーター:

なし

戻り値:

読み取られる浮動小数点数。

例外:

- XMSEException
- MessageNotReadableException
- MessageEOFException

ReadInt - 整数の読み取り

インターフェース:

```
Int32 ReadInt();
```

バイト・メッセージ・ストリームから、次の 4 バイトを符号付き 32 ビット整数として読み取ります。

パラメーター:

なし

戻り値:

読み取られる整数。

例外:

- XMSEException
- MessageNotReadableException
- MessageEOFException

ReadLong - 長整数の読み取り

インターフェース:

```
Int64 ReadLong();
```

バイト・メッセージ・ストリームから、次の 8 バイトを符号付き 64 ビット整数として読み取ります。

パラメーター:

なし

戻り値:

読み取られる長整数。

例外:

- XMSEException
- MessageNotReadableException
- MessageEOFException

ReadShort - 短整数の読み取り

インターフェース:

```
Int16 ReadShort();
```

バイト・メッセージ・ストリームから、次の 2 バイトを符号付き 16 ビット整数として読み取ります。

パラメーター:

なし

戻り値:

読み取られる短整数。

例外:

- XMSEException
- MessageNotReadableException
- MessageEOFException

ReadByte - 符号なしバイトの読み取り

インターフェース:

```
Byte    ReadByte();
```

バイト・メッセージ・ストリームから、次のバイトを符号なし 8 ビット整数として読み取ります。

パラメーター:

なし

戻り値:

読み取られるバイト。

例外:

- XMSEException
- MessageNotReadableException
- MessageEOFException

ReadUnsignedShort - 符号なし短整数の読み取り

インターフェース:

```
Int32    ReadUnsignedShort();
```

バイト・メッセージ・ストリームから、次の 2 バイトを符号なし 16 ビット整数として読み取ります。

パラメーター:

なし

戻り値:

読み取られる符号なし短整数。

例外:

- XMSEException
- MessageNotReadableException
- MessageEOFException

ReadUTF - UTF スtring の読み取り

インターフェース:

```
String    ReadUTF();
```

バイト・メッセージ・ストリームから、UTF-8 でエンコードされた String を読み取ります。

注: ReadUTF() を呼び出す前に、バッファのカーソルがバイト・メッセージ・ストリームの先頭を指すようにしてください。

パラメーター:

なし

戻り値:

読み取られる String をカプセル化している String オブジェクト。

例外:

- XMSEException

- MessageNotReadableException
- MessageEOFException

Reset - リセット

インターフェース:

```
void Reset();
```

メッセージの本体を読み取り専用モードにして、カーソルをバイト・メッセージ・ストリームの先頭に位置変更します。

パラメーター:

なし

戻り値:

Void

例外:

- XMSEException
- MessageNotReadableException

WriteBoolean - ブール値の書き込み

インターフェース:

```
void WriteBoolean(Boolean value);
```

バイト・メッセージ・ストリームへブール値を書き込みます。

パラメーター:

value (入力)

書き込まれるブール値。

戻り値:

Void

例外:

- XMSEException
- MessageNotWritableException

WriteByte - バイトの書き込み

インターフェース:

```
void WriteByte(Byte value);
void WriteSignedByte(Int16 value);
```

バイト・メッセージ・ストリームへバイトを書き込みます。

パラメーター:

value (入力)

書き込まれるバイト。

戻り値:

Void

例外:

- XMSEException
- MessageNotWritableException

WriteBytes - 複数バイトの書き込み

インターフェース:

```
void WriteBytes(Byte[] value);
```

バイト・メッセージ・ストリームへバイトの配列を書き込みます。

パラメーター:

value (入力)

書き込まれるバイト配列。

戻り値:

Void

例外:

- XMSEException
- MessageNotWritableException

WriteBytes - 部分的なバイト配列の書き込み

インターフェース:

```
void WriteBytes(Byte[] value, int offset, int length);
```

部分的なバイト配列を、指定の長さで定義したとおりにバイト・メッセージ・ストリームに書き込みます。

パラメーター:

value (入力)

書き込まれるバイト配列。

offset (入力)

書き込まれるバイト配列の開始点。

length (入力)

書き込むバイト数。

戻り値:

Void

例外:

- XMSEException
- MessageNotWritableException

WriteChar - 文字の書き込み

インターフェース:

```
void WriteChar(Char value);
```

文字を、上位バイトを先にして、2 バイトでバイト・メッセージ・ストリームに書き込みます。

パラメーター:

value (入力)

書き込まれる文字。

戻り値:

Void

例外:

- XMSEException
- MessageNotWritableException

WriteDouble - 倍精度浮動小数点数の書き込み

インターフェース:

```
void WriteDouble(Double value);
```

倍精度浮動小数点数を長整数に変換し、その長整数を、上位バイトを先にして、8 バイトでバイト・メッセージ・ストリームに書き込みます。

パラメーター:

value (入力)

書き込まれる倍精度浮動小数点数。

戻り値:

Void

例外:

- XMSEException
- MessageNotWritableException

WriteFloat - 浮動小数点数の書き込み

インターフェース:

```
void WriteFloat(Single value);
```

浮動小数点数を整数に変換し、その整数を、上位バイトを先にして、4 バイトでバイト・メッセージ・ストリームに書き込みます。

パラメーター:

value (入力)

書き込まれる浮動小数点数。

戻り値:

Void

例外:

- XMSEException
- MessageNotWritableException

WriteInt - 整数の書き込み

インターフェース:

```
void WriteInt(Int32 value);
```

整数を、上位バイトを先にして、4 バイトでバイト・メッセージ・ストリームに書き込みます。

パラメーター:

value (入力)

書き込まれる整数。

戻り値:

Void

例外:

- XMSEException
- MessageNotWritableException

WriteLong - 長整数の書き込み

インターフェース:

```
void WriteLong(Int64 value);
```

長整数を、上位バイトを先にして、8 バイトでバイト・メッセージ・ストリームに書き込みます。

パラメーター:

value (入力)

書き込まれる長整数。

戻り値:

Void

例外:

- XMSEException
- MessageNotWritableException

WriteObject - オブジェクトの書き込み

インターフェース:

```
void WriteObject(Object value);
```

指定したオブジェクトをバイト・メッセージ・ストリームに書き込みます。

パラメーター:

value (入力)

書き込まれるオブジェクト。プリミティブ型への参照である必要があります。

戻り値:

Void

例外:

- XMSEException
- MessageNotWritableException

WriteShort - 短整数の書き込み

インターフェース:

```
void WriteShort(Int16 value);
```

短整数を、上位バイトを先にして、2 バイトでバイト・メッセージ・ストリームに書き込みます。

パラメーター:

value (入力)

書き込まれる短整数。

戻り値:

Void

例外:

- XMSEException
- MessageNotWritableException

WriteUTF - UTF スtringの書き込み

インターフェース:

```
void WriteUTF(String value);
```

バイト・メッセージ・ストリームへ、UTF-8 でエンコードされたストリングを書き込みます。

パラメーター:

value (入力)

書き込まれるストリングをカプセル化している String オブジェクト。

戻り値:

Void

例外:

- XMSEException
- MessageNotWritableException

継承されたプロパティーおよびメソッド

以下のプロパティーは、[IMessage](#) インターフェースから継承されています。

[JMSCorrelationID](#)、[JMSDeliveryMode](#)、[JMSDestination](#)、[JMSExpiration](#)、[JMSMessageID](#)、[JMSPriority](#)、[JMSRedelivered](#)、[JMSReplyTo](#)、[JMSTimestamp](#)、[JMSType](#)、[Properties](#)

以下のメソッドは、[IMessage](#) インターフェースから継承されています。

[clearBody](#)、[clearProperties](#)、[PropertyExists](#)

以下のメソッドは、[IPropertyContext](#) インターフェースから継承されています。

[GetBooleanProperty](#)、[GetByteProperty](#)、[GetBytesProperty](#)、[GetCharProperty](#)、[GetDoubleProperty](#)、[GetFloatProperty](#)、[GetIntProperty](#)、[GetLongProperty](#)、[GetObjectProperty](#)、[GetShortProperty](#)、[GetStringProperty](#)、[SetBooleanProperty](#)、[SetByteProperty](#)、[SetBytesProperty](#)、[SetCharProperty](#)、[SetDoubleProperty](#)、[SetFloatProperty](#)、[SetIntProperty](#)、[SetLongProperty](#)、[SetObjectProperty](#)、[SetShortProperty](#)、[SetStringProperty](#)

IConnection

Connection オブジェクトは、アプリケーションからメッセージング・サーバーへのアクティブな接続を表します。

継承の階層:

```
IBM.XMS.IPropertyContext
|
+----IBM.XMS.IConnection
```

Connection オブジェクトの XMS 定義プロパティーのリストについては、[2038 ページの『Connection のプロパティー』](#)を参照してください。

.NET プロパティー

ClientID - クライアント ID の取得および設定

インターフェース:

```
String ClientID
{
    get;
    set;
}
```

接続のクライアント ID を取得および設定します。

クライアント ID は、ClientID を設定することにより割り当てられることも、管理者が ConnectionFactory で事前に構成することもできます。

クライアント ID は、パブリッシュ/サブスクライブ・ドメイン内の永続サブスクリプションをサポートするためだけに使用され、Point-to-Point ドメインでは無視されます。

アプリケーションが接続のクライアント ID を設定する場合、アプリケーションは、接続の作成の直後に、接続で他の操作を実行する前にこの設定を行う必要があります。アプリケーションが、この時点よりも後に、このクライアント ID の設定を試行すると、呼び出しは例外 `IllegalStateException` をスローします。

ブローカーへのリアルタイム接続の場合、このプロパティは無効です。

例外:

- `XMSEException`
- `IllegalStateException`
- `InvalidClientIDException`

ExceptionListener - 例外リスナーの取得および設定

インターフェース:

```
ExceptionListener ExceptionListener
{
    get;
    set;
}
```

接続に登録されている例外リスナーを取得し、例外リスナーを接続に登録します。

接続に例外リスナーが登録されていない場合、このメソッドでは `NULL` が戻されます。接続に例外リスナーが既に登録されている場合は、この例外リスナーの代わりに `NULL` を指定すれば、登録を取り消すことができます。

例外リスナーの使用について詳しくは、[.NET でのメッセージおよび例外リスナーの使用](#)を参照してください。

例外:

- `XMSEException`

Metadata - メタデータの取得

インターフェース:

```
IConnectionMetaData MetaData
{
    get;
}
```

接続のメタデータを取得します。

例外:

- `XMSEException`

方法

Close - 接続のクローズ

インターフェース:

```
void Close();
```

接続を閉じます。

アプリケーションが、既に閉じている接続を閉じようとした場合、呼び出しは無視されます。

パラメーター:

なし

戻り値:

`Void`

例外:

- XMSEException

CreateSession - セッションの作成

インターフェース:

```
ISession CreateSession(Boolean transacted,  
                        AcknowledgeMode acknowledgeMode);
```

セッションを作成します。

パラメーター:**transacted (入力)**

値 True は、セッションがトランザクション化されていることを意味します。値 False は、セッションがトランザクション化されていないことを意味します。

ブローカーへのリアルタイム接続の場合、値は False である必要があります。

acknowledgeMode (入力)

アプリケーションが受信するメッセージの確認応答の方法を示します。値は、以下の AcknowledgeMode 列挙子のいずれかにする必要があります。

```
AcknowledgeMode.AutoAcknowledge  
AcknowledgeMode.ClientAcknowledge  
AcknowledgeMode.DupsOkAcknowledge
```

ブローカーへのリアルタイム接続の場合、値は AcknowledgeMode.AutoAcknowledge または AcknowledgeMode.DupsOkAcknowledge である必要があります。

セッションがトランザクション化されている場合、このパラメーターは無視されます。肯定応答モードについて詳しくは、[メッセージの肯定応答](#)を参照してください。

戻り値:

Session オブジェクト。

例外:

- XMSEException

Start - 接続の開始

インターフェース:

```
void Start();
```

接続の着信メッセージの配信を開始または再開します。接続が既に開始されている場合、呼び出しは無視されます。

パラメーター:

なし

戻り値:

Void

例外:

- XMSEException

Stop - 接続の停止

インターフェース:

```
void Stop();
```

接続の着信メッセージの配信を停止します。接続が既に停止されている場合、呼び出しは無視されます。

パラメーター:

なし

戻り値:

Void

例外:

- XMSEException

継承されたプロパティーおよびメソッド

以下のメソッドは、[IPropertyContext](#) インターフェースから継承されています。

[GetBooleanProperty](#)、[GetByteProperty](#)、[GetBytesProperty](#)、[GetCharProperty](#)、[GetDoubleProperty](#)、[GetFloatProperty](#)、[GetIntProperty](#)、[GetLongProperty](#)、[GetObjectProperty](#)、[GetShortProperty](#)、[GetStringProperty](#)、[SetBooleanProperty](#)、[SetByteProperty](#)、[SetBytesProperty](#)、[SetCharProperty](#)、[SetDoubleProperty](#)、[SetFloatProperty](#)、[SetIntProperty](#)、[SetLongProperty](#)、[SetObjectProperty](#)、[SetShortProperty](#)、[SetStringProperty](#)

IConnectionFactory

アプリケーションは、接続ファクトリーを使用して接続を作成します。

継承の階層:

```
IBM.XMS.IPropertyContext
|
+---- IBM.XMS.IConnectionFactory
```

ConnectionFactory オブジェクトの XMS 定義プロパティーのリストについては、[2039 ページの『ConnectionFactory のプロパティー』](#)を参照してください。

方法

CreateConnection - 接続ファクトリーの作成 (デフォルト・ユーザー ID を使用)

インターフェース:

```
IConnection CreateConnection();
```

デフォルトのプロパティーを使用して接続ファクトリーを作成します。

IBM MQ に接続しているときに、XMSC_USERID が設定されていない場合、キュー・マネージャーはデフォルトでログオン・ユーザーのユーザー ID を使用します。個々のユーザーの接続レベル認証がさらに必要な場合には、IBM MQ で構成済みのクライアント認証出口を作成できます。

パラメーター:

なし

例外:

- XMSEException

CreateConnection - 接続の作成 (指定されたユーザー ID を使用)

インターフェース:

```
IConnection CreateConnection(String userId, String password);
```

指定されたユーザー ID をを使用して接続を作成します。

IBM MQ に接続しているときに、XMSC_USERID が設定されていない場合、キュー・マネージャーはデフォルトでログオン・ユーザーのユーザー ID を使用します。個々のユーザーの接続レベル認証がさらに必要な場合には、IBM MQ で構成済みのクライアント認証出口を作成できます。

接続は停止済みモードで作成されます。アプリケーションが **Connection.start()** を呼び出すまで、メッセージは配信されません。

パラメーター:

userID (入力)

アプリケーションを認証するときに使用するユーザー ID をカプセル化している String オブジェクト。NULL を指定した場合は、認証のない接続の作成が試行されます。

password (入力)

アプリケーションを認証するときに使用するパスワードをカプセル化している String オブジェクト。NULL を指定した場合は、認証のない接続の作成が試行されます。

戻り値:

Connection オブジェクト。

例外:

- XMSEException
- XMS_X_SECURITY_EXCEPTION

継承されたプロパティおよびメソッド

以下のメソッドは、[IPropertyContext](#) インターフェースから継承されています。

[GetBooleanProperty](#)、[GetByteProperty](#)、[GetBytesProperty](#)、[GetCharProperty](#)、[GetDoubleProperty](#)、[GetFloatProperty](#)、[GetIntProperty](#)、[GetLongProperty](#)、[GetObjectProperty](#)、[GetShortProperty](#)、[GetStringProperty](#)、[SetBooleanProperty](#)、[SetByteProperty](#)、[SetBytesProperty](#)、[SetCharProperty](#)、[SetDoubleProperty](#)、[SetFloatProperty](#)、[SetIntProperty](#)、[SetLongProperty](#)、[SetObjectProperty](#)、[SetShortProperty](#)、[SetStringProperty](#)

IConnectionMetaData

ConnectionMetaData オブジェクトは、接続に関する情報を提供します。

継承の階層:

```
IBM.XMS.IPropertyContext
|
+---- IBM.XMS.IConnectionMetaData
```

ConnectionMetaData オブジェクトの XMS 定義プロパティのリストについては、[2043 ページの『ConnectionMetaData のプロパティ』](#)を参照してください。

.NET プロパティ

JMSXPropertyNames - JMS 定義メッセージ・プロパティの取得

インターフェース:

```
System.Collections.IEnumerator JMSXPropertyNames
{
    get;
}
```

接続でサポートされている JMS 定義メッセージ・プロパティの名前の列挙を戻します。

JMS 定義メッセージ・プロパティは、ブローカーへのリアルタイム接続ではサポートされていません。

例外:

- XMSEException

継承されたプロパティおよびメソッド

以下のメソッドは、[IPropertyContext](#) インターフェースから継承されています。

[GetBooleanProperty](#)、[GetByteProperty](#)、[GetBytesProperty](#)、[GetCharProperty](#)、[GetDoubleProperty](#)、[GetFloatProperty](#)、[GetIntProperty](#)、[GetLongProperty](#)、[GetObjectProperty](#)、[GetShortProperty](#)、[GetStringProperty](#)、[SetBooleanProperty](#)、[SetByteProperty](#)、[SetBytesProperty](#)、[SetCharProperty](#)、[SetDoubleProperty](#)、[SetFloatProperty](#)、[SetIntProperty](#)、[SetLongProperty](#)、[SetObjectProperty](#)、[SetShortProperty](#)、[SetStringProperty](#)

IDestination

宛先とは、アプリケーションがメッセージを送信する場所、またはアプリケーションがメッセージを受信する場合の送信元、あるいはその両方のことです。

継承の階層:

```
IBM.XMS.IPropertyContext
|
+---- IBM.XMS.IDestination
```

Destination オブジェクトの XMS 定義プロパティのリストについては、[2044 ページの『Destination のプロパティ』](#)を参照してください。

.NET プロパティ

Name - 宛先名の取得

インターフェース:

```
String Name
{
    get;
}
```

宛先名を取得します。この名前は、キューの名前またはトピックの名前をカプセル化しているストリングです。

例外:

- [XMSEException](#)

TypeId - 宛先タイプの取得

インターフェース:

```
DestinationType TypeId
{
    get;
}
```

宛先のタイプを取得します。宛先のタイプは、以下の値のいずれかです。

[DestinationType.Queue](#)
[DestinationType.Topic](#)

例外:

- [XMSEException](#)

継承されたプロパティおよびメソッド

以下のメソッドは、[IPropertyContext](#) インターフェースから継承されています。

[GetBooleanProperty](#)、[GetByteProperty](#)、[GetBytesProperty](#)、[GetCharProperty](#)、[GetDoubleProperty](#)、[GetFloatProperty](#)、[GetIntProperty](#)、[GetLongProperty](#)、[GetObjectProperty](#)、[GetShortProperty](#)、[GetStringProperty](#)、[SetBooleanProperty](#)、[SetByteProperty](#)、[SetBytesProperty](#)、[SetCharProperty](#)、[SetDoubleProperty](#)、[SetFloatProperty](#)、[SetIntProperty](#)、[SetLongProperty](#)、[SetObjectProperty](#)、[SetShortProperty](#)、[SetStringProperty](#)

ExceptionHandler

アプリケーションは例外リスナーを使用して、接続の問題に関する通知を非同期に受信します。

継承の階層:

なし

アプリケーションがメッセージを非同期にコンSUMするのためにのみ接続を使用し、他の目的では使用しない場合、アプリケーションが接続の問題を確認することができる唯一の方法は、例外リスナーを使用することです。その他の状況では、例外リスナーは、次の XMS への同期呼び出しを待機するよりも迅速に、接続の問題を確認する方法を提供することができます。

代行

ExceptionHandler - 例外リスナー

インターフェース:

```
public delegate void ExceptionListener(Exception ex)
```

アプリケーションに接続の問題を通知します。

この代行を実装するメソッドは、接続に登録できます。

例外リスナーの使用について詳しくは、[.NET でのメッセージおよび例外リスナーの使用](#)を参照してください。

パラメーター:

exception (入力)

XMS が作成した例外へのポインター。

戻り値:

Void

IllegalStateException

アプリケーションがメソッドを正しくない時刻または不適當な時刻に呼び出した場合、または XMS が要求された操作に適切な状態でない場合、XMS はこの例外をスローします。

継承の階層:

```
IBM.XMS.XMSException
|
+----IBM.XMS.Exception
|
+----IBM.XMS.IllegalStateException
```

継承されたプロパティーおよびメソッド

以下のメソッドは、[XMSException](#) インターフェースから継承されています。

[GetErrorCode](#)、[GetLinkedException](#)

InitialContext

アプリケーションは、InitialContext オブジェクトを使用して、管理対象オブジェクトのリポジトリから取得したオブジェクト定義によってオブジェクトを作成します。

継承の階層:

なし

.NET プロパティー

Environment - 環境の取得

インターフェース:

```
Hashtable Environment
{
    get;
}
```

環境を取得します。

例外:

- 例外は、使用するディレクトリー・サービスに固有のものです。

コンストラクター

InitialContext - 初期コンテキストの作成

インターフェース:

```
InitialContext(Hashtable env);
```

InitialContext オブジェクトを作成します。

パラメーター:

管理対象オブジェクトのリポジトリーへの接続を確立するために必要な情報は、環境 Hashtable 内のコンストラクターに渡されます。

例外:

- XMSEException

方法

AddToEnvironment - 環境への新規プロパティーの追加

インターフェース:

```
Object AddToEnvironment(String propName, Object propVal);
```

新規プロパティーを環境に追加します。

パラメーター:

propName (入力)

追加するプロパティーの名前をカプセル化している String オブジェクト。

propVal (入力)

追加するプロパティーの値。

戻り値:

プロパティーの以前の値。

例外:

- 例外は、使用するディレクトリー・サービスに固有のものです。

Close - このコンテキストのクローズ

インターフェース:

```
void Close()
```

このコンテキストをクローズします。

パラメーター:

なし

戻り値:

なし

例外:

- 例外は、使用するディレクトリー・サービスに固有のものです。

Lookup - 初期コンテキスト内のオブジェクトの検索

インターフェース:

```
Object Lookup(String name);
```

管理対象オブジェクトのリポジトリから取得したオブジェクト定義によって、オブジェクトを作成します。

パラメーター:

name (入力)

検索対象の管理対象オブジェクトの名前をカプセル化している String オブジェクト。この名前は、単純な名前でも複雑な名前でも構いません。詳細については、[管理対象オブジェクトの取得](#)を参照してください。

戻り値:

検索の対象となるオブジェクトのタイプに応じて、*ConnectionFactory* または *IDestination* のいずれか。この関数はディレクトリーにアクセスできますが、必要なオブジェクトを検索できないため、NULL が戻ります。

例外:

- 例外は、使用するディレクトリー・サービスに固有のものです。

RemoveFromEnvironment - 環境からのプロパティーの除去

インターフェース:

```
Object RemoveFromEnvironment(String propName);
```

環境からプロパティーを除去します。

パラメーター:

propName (入力)

除去するプロパティーの名前をカプセル化している String オブジェクト。

戻り値:

除去されたオブジェクト。

例外:

- 例外は、使用するディレクトリー・サービスに固有のものです。

InvalidClientIDException

XMS がこの例外をスローするのは、アプリケーションが接続のクライアント ID を設定しようとしたが、そのクライアント ID が無効かまたは既に使用中である場合です。

継承の階層:

```
IBM.XMS.XMSException
|
+----IBM.XMS.XMSException
|
+----IBM.XMS.InvalidClientIDException
```

継承されたプロパティおよびメソッド

以下のメソッドは、[XMSEException](#) インターフェースから継承されています。

[GetErrorCode](#)、[GetLinkedException](#)

InvalidDestinationException

XMS がこの例外をスローするのは、アプリケーションが無効な宛先を指定している場合です。

継承の階層:

```
IBM.XMS.XMSEException
|
+----IBM.XMS.XMSEException
|
+----IBM.XMS.InvalidDestinationException
```

継承されたプロパティおよびメソッド

以下のメソッドは、[XMSEException](#) インターフェースから継承されています。

[GetErrorCode](#)、[GetLinkedException](#)

InvalidSelectorException

XMS がこの例外をスローするのは、無効な構文のメッセージ・セレクター式をアプリケーションで指定した場合です。

継承の階層:

```
IBM.XMS.XMSEException
|
+----IBM.XMS.XMSEException
|
+----IBM.XMS.InvalidSelectorException
```

継承されたプロパティおよびメソッド

以下のメソッドは、[XMSEException](#) インターフェースから継承されています。

[GetErrorCode](#)、[GetLinkedException](#)

IMapMessage

マップ・メッセージとは、本体が名前と値のペアで構成されるメッセージです。それぞれの値に、関連付けられたデータ・タイプがあります。

継承の階層:

```
IBM.XMS.IPropertyContext
|
+----IBM.XMS.IMessage
|
+----IBM.XMS.IMapMessage
```

アプリケーションが名前と値のペアの値を取得するとき、値は XMS によって別のデータ・タイプに変換される可能性があります。この形式の暗黙の型変換について詳しくは、[XMS メッセージの本体にあるマップ・メッセージに関する情報を参照してください](#)。

.NET プロパティ

MapNames - マップ名の取得

インターフェース:

```
System.Collections.IEnumerator MapNames
{
    get;
}
```

マップ・メッセージの本体に存在する名前の列挙を取得します。

例外:

- XMSEException

方法

GetBoolean - ブール値の取得

インターフェース:

```
Boolean GetBoolean(String name);
```

マップ・メッセージの本体から名前で識別されるブール値を取得します。

パラメーター:

name (入力)

ブール値を識別する名前をカプセル化している String オブジェクト。

戻り値:

マップ・メッセージの本体から検索されたブール値。

例外:

- XMSEException

GetByte - バイトの取得

インターフェース:

```
Byte    GetByte(String name);
Int16   GetSignedByte(String name);
```

マップ・メッセージの本体から名前で識別されるバイトを取得します。

パラメーター:

name (入力)

バイトを識別する名前をカプセル化している String オブジェクト。

戻り値:

マップ・メッセージの本体から検索されたバイト。バイトにはデータ変換は実行されません。

例外:

- XMSEException

GetBytes - 複数バイトの取得

インターフェース:

```
Byte[]  GetBytes(String name);
```

マップ・メッセージの本体から名前で識別されるバイトの配列を取得します。

パラメーター:

name (入力)

バイトの配列を識別する名前をカプセル化している String オブジェクト。

戻り値:

配列のバイト数。

例外:

- XMSEException

GetChar - 文字の取得

インターフェース:

```
Char    GetChar(String name);
```

マップ・メッセージの本体から名前で識別される文字を取得します。

パラメーター:**name (入力)**

文字を識別する名前をカプセル化している String オブジェクト。

戻り値:

マップ・メッセージの本体から検索された文字。

例外:

- XMSEException

GetDouble - 倍精度浮動小数点数の取得

インターフェース:

```
Double  GetDouble(String name);
```

マップ・メッセージの本体から名前で識別される倍精度浮動小数点数を取得します。

パラメーター:**name (入力)**

倍精度浮動小数点数を識別する名前をカプセル化している String オブジェクト。

戻り値:

マップ・メッセージの本体から検索された倍精度浮動小数点数。

例外:

- XMSEException

GetFloat - 浮動小数点数の取得

インターフェース:

```
Single  GetFloat(String name);
```

マップ・メッセージの本体から名前で識別される浮動小数点数を取得します。

パラメーター:**name (入力)**

浮動小数点数を識別する名前をカプセル化している String オブジェクト。

戻り値:

マップ・メッセージの本体から検索された浮動小数点数。

例外:

- XMSEException

GetInt - 整数の取得

インターフェース:

```
Int32  GetInt(String name);
```

マップ・メッセージの本体から名前で識別される整数を取得します。

パラメーター:

name (入力)

整数を識別する名前をカプセル化している String オブジェクト。

戻り値:

マップ・メッセージの本体から検索された整数。

例外:

- XMSEException

GetLong - 長整数の取得

インターフェース:

```
Int64  GetLong(String name);
```

マップ・メッセージの本体から名前で識別される長整数を取得します。

パラメーター:

name (入力)

長整数を識別する名前をカプセル化している String オブジェクト。

戻り値:

マップ・メッセージの本体から検索された長整数。

例外:

- XMSEException

GetObject - オブジェクトの取得

インターフェース:

```
Object  GetObject(String name);
```

マップ・メッセージの本体から、名前と値のペアの値への参照を取得します。名前と値のペアは、名前と値で識別されます。

パラメーター:

name (入力)

名前と値のペアの名前をカプセル化している String オブジェクト。

戻り値:

値。以下のオブジェクト・タイプのいずれかです。

Boolean
Byte
Byte[]
Char
Double
Single
Int32
Int64
Int16
String

例外:

XMSEException

GetShort - 短整数の取得

インターフェース:

```
Int16 GetShort(String name);
```

マップ・メッセージの本体から名前で識別される短整数を取得します。

パラメーター:**name (入力)**

短整数を識別する名前をカプセル化している String オブジェクト。

戻り値:

マップ・メッセージの本体から検索された短整数。

例外:

- XMSEException

GetString - スtringの取得

インターフェース:

```
String GetString(String name);
```

マップ・メッセージの本体から名前で識別されるStringを取得します。

パラメーター:**name (入力)**

マップ・メッセージの本体のStringを識別する名前をカプセル化している String オブジェクト。

戻り値:

マップ・メッセージの本体から取り出したStringをカプセル化している String オブジェクト。データ変換が必要な場合、この値は、変換後のStringになります。

例外:

- XMSEException

ItemExists - 名前と値のペアの存在のチェック

インターフェース:

```
Boolean ItemExists(String name);
```

マップ・メッセージの本体に、指定された名前の名前-値ペアが含まれているかどうかをチェックします。

パラメーター:**name (入力)**

名前と値のペアの名前をカプセル化している String オブジェクト。

戻り値:

- マップ・メッセージの本体に、指定された名前が付けられた名前と値のペアが含まれている場合は、True です。
- マップ・メッセージの本体に、指定された名前が付けられた名前と値のペアが含まれていない場合は、False です。

例外:

- XMSEException

SetBoolean - ブール値の設定

インターフェース:

```
void SetBoolean(String name, Boolean value);
```

マップ・メッセージの本体にブール値を設定します。

パラメーター:

name (入力)

マップ・メッセージの本体に存在するブール値を識別するための名前をカプセル化している String オブジェクト。

value (入力)

設定されるブール値。

戻り値:

Void

例外:

- XMSEException

SetByte - バイトの設定

インターフェース:

```
void SetByte(String name, Byte value);  
void SetSignedByte(String name, Int16 value);
```

マップ・メッセージの本体にバイトを設定します。

パラメーター:

name (入力)

マップ・メッセージの本体に存在するバイトを識別するための名前をカプセル化している String オブジェクト。

value (入力)

設定されるバイト。

戻り値:

Void

例外:

- XMSEException

SetBytes - 複数バイトの設定

インターフェース:

```
void SetBytes(String name, Byte[] value);
```

マップ・メッセージの本体にバイトの配列を設定します。

パラメーター:

name (入力)

マップ・メッセージの本体に存在するバイトの配列を識別するための名前をカプセル化している String オブジェクト。

value (入力)

設定されるバイト配列。

戻り値:

Void

例外:

- XMSEException

SetChar - 文字の設定

インターフェース:

```
void SetChar(String name, Char value);
```

マップ・メッセージの本体に 2 バイト文字を設定します。

パラメーター:

name (入力)

マップ・メッセージの本体に存在する文字を識別するための名前をカプセル化している String オブジェクト。

value (入力)

設定される文字。

戻り値:

Void

例外:

- XMSEException

SetDouble - 倍精度浮動小数点数の設定

インターフェース:

```
void SetDouble(String name, Double value);
```

マップ・メッセージの本体に倍精度浮動小数点数を設定します。

パラメーター:

name (入力)

マップ・メッセージの本体に存在する倍精度浮動小数点数を識別するための名前をカプセル化している String オブジェクト。

value (入力)

設定される倍精度浮動小数点数。

戻り値:

Void

例外:

- XMSEException

SetFloat - 浮動小数点数の設定

インターフェース:

```
void SetFloat(String name, Single value);
```

マップ・メッセージの本体に浮動小数点数を設定します。

パラメーター:

name (入力)

マップ・メッセージの本体に存在する浮動小数点数を識別するための名前をカプセル化している String オブジェクト。

value (入力)

設定される浮動小数点数。

戻り値:

Void

例外:

- XMSEException

SetInt - 整数の設定

インターフェース:

```
void SetInt(String name, Int32 value);
```

マップ・メッセージの本体に整数を設定します。

パラメーター:

name (入力)

マップ・メッセージの本体に存在する整数を識別するための名前をカプセル化している String オブジェクト。

value (入力)

設定される整数。

戻り値:

Void

例外:

- XMSEException

SetLong - 長整数の設定

インターフェース:

```
void SetLong(String name, Int64 value);
```

マップ・メッセージの本体に長整数を設定します。

パラメーター:

name (入力)

マップ・メッセージの本体に存在する長整数を識別するための名前をカプセル化している String オブジェクト。

value (入力)

設定される長整数。

戻り値:

Void

例外:

- XMSEException

SetObject - オブジェクトの設定

インターフェース:

```
void SetObject(String name, Object value);
```

マップ・メッセージの本体に XMS プリミティブ・タイプの値を設定します。

パラメーター:

name (入力)

マップ・メッセージの本体に存在する値を識別するための名前をカプセル化している String オブジェクト。

value (入力)

設定される値を含むバイトの配列。

戻り値:

Void

例外:

- XMSEException

SetShort - 短整数の設定

インターフェース:

```
void SetShort(String name, Int16 value);
```

マップ・メッセージの本体に短整数を設定します。

パラメーター:

name (入力)

マップ・メッセージの本体に存在する短整数を識別するための名前をカプセル化している String オブジェクト。

value (入力)

設定される短整数。

戻り値:

Void

例外:

- XMSEException

SetString - スtringの設定

インターフェース:

```
void SetString(String name, String value);
```

マップ・メッセージの本体にStringを設定します。

パラメーター:

name (入力)

マップ・メッセージの本体に存在するStringを識別するための名前をカプセル化している String オブジェクト。

value (入力)

設定されるStringをカプセル化している String オブジェクト。

戻り値:

Void

例外:

- XMSEException

継承されたプロパティおよびメソッド

以下のプロパティは、IMessage インターフェースから継承されています。

JMSCorrelationID、JMSDeliveryMode、JMSDestination、JMSExpiration、JMSMessageID、JMSPriority、JMSRedelivered、JMSReplyTo、JMSTimestamp、JMSType、Properties

以下のメソッドは、IMessage インターフェースから継承されています。

clearBody、clearProperties、PropertyExists

以下のメソッドは、IPropertyContext インターフェースから継承されています。

GetBooleanProperty、GetByteProperty、GetBytesProperty、GetCharProperty、GetDoubleProperty、GetFloatProperty、GetIntProperty、GetLongProperty、GetObjectProperty、GetShortProperty、GetStringProperty、SetBooleanProperty、SetByteProperty、SetBytesProperty、SetCharProperty、SetDoubleProperty、SetFloatProperty、SetIntProperty、SetLongProperty、SetObjectProperty、SetShortProperty、SetStringProperty

IMessage

メッセージ・オブジェクトは、アプリケーションが送信または受信するメッセージを表します。IMessage は、IMapMessage などの Message クラスのスーパークラスです。

継承の階層:

```
IBM.XMS.IPropertyContext
|
+---- IBM.XMS.IMessage
```

Message オブジェクトの JMS メッセージ・ヘッダー・フィールドのリストについては、[XMS メッセージのヘッダー・フィールド](#)を参照してください。Message オブジェクトの JMS 定義プロパティのリストについては、[メッセージの JMS 定義プロパティ](#)を参照してください。Message オブジェクトの IBM 定義プロパティのリストについては、[メッセージの IBM 定義プロパティ](#)を参照してください。Message オブジェクトの JMS_IBM_MQMD* プロパティのリストについては、[2048 ページの『JMS_IBM_MQMD* プロパティ』](#)を参照してください。

メッセージは、ガーベッジ・コレクターによって削除されます。メッセージが削除されると、これによってメッセージが使用していたリソースが解放されます。

.NET プロパティ

GetJMSCorrelationID - *JMSCorrelationID* の取得および設定

インターフェース:

```
String JMSCorrelationID
{
    get;
    set;
}
```

メッセージの相関 ID を String オブジェクトとして取得および設定します。

例外:

- XMSEException

JMSDeliveryMode - *JMSDeliveryMode* の取得および設定

インターフェース:

```
DeliveryMode JMSDeliveryMode
{
    get;
    set;
}
```

メッセージの送達モードを取得して設定します。

メッセージの送達モードは、以下の値のいずれかです。

```
DeliveryMode.Persistent
DeliveryMode.NonPersistent
```

新規に作成されて送信されていないメッセージの場合、送達モードは `DeliveryMode.Persistent` です。ただし、送達モードが `DeliveryMode.NonPersistent` であるブローカーへのリアルタイム接続の場合を除きます。受信されたメッセージについては、受信側アプリケーションが `JMSDeliveryMode` を設定して送達モードを変更していない限り、このメソッドでは、メッセージ送信時に `IMessageProducer.send()` 呼び出しによって設定された送達モードが戻されます。

例外:

- XMSEException

JMSDestination - JMSDestination の取得および設定

インターフェース:

```
IDestination JMSDestination
{
    get;
    set;
}
```

メッセージの宛先を取得して設定します。

宛先は、メッセージの送信時に `IMessageProducer.send()` 呼び出しによって設定されます。JMSDestination の値は無視されます。ただし、JMSDestination を使用して、受信されたメッセージの宛先を変更することができます。

未送信の新規作成メッセージの場合、送信側アプリケーションが JMSDestination を設定して宛先を設定しない限り、メソッドはヌルの Destination オブジェクトを返します。受信されたメッセージについては、受信側アプリケーションが JMSDestination を設定して宛先を変更していない限り、このメソッドでは、メッセージ送信時に `IMessageProducer.send()` 呼び出しによって設定された宛先の Destination オブジェクトが返されます。

例外:

- XMSEException

JMSExpiration - JMSExpiration の取得および設定

インターフェース:

```
Int64 JMSExpiration
{
    get;
    set;
}
```

メッセージの有効期限切れ時刻を取得および設定します。

有効期限切れ時刻は、メッセージの送信時に `IMessageProducer.send()` 呼び出しによって設定されます。その値は、送信側アプリケーションが指定した存続時間を、メッセージの送信時刻に加算して計算されます。有効期限切れ時刻は、1970 年 1 月 1 日 00:00:00 GMT からのミリ秒で表されます。

新規に作成されて送信されていないメッセージの場合、送信側アプリケーションが JMSExpiration を設定して異なる有効期限切れ時刻を設定していない限り、有効期限切れ時刻は 0 です。受信されたメッセージについては、受信側アプリケーションが JMSExpiration を設定して有効期限切れ時刻を変更していない限り、このメソッドでは、メッセージ送信時に `IMessageProducer.send()` 呼び出しによって設定された有効期限切れ時刻が返されます。

存続時間が 0 の場合、`IMessageProducer.send()` 呼び出しでは、有効期限切れ時刻が 0 に設定されますが、これはメッセージの有効期限がないことを示します。

XMS は、有効期限が切れたメッセージを廃棄し、アプリケーションに配信しません。

例外:

- XMSEException

JMSMessageID - JMSMessageID の取得および設定

インターフェース:

```
String JMSMessageID
{
    get;
    set;
}
```

メッセージのメッセージ ID を、このメッセージ ID をカプセル化している String オブジェクトとして取得し、設定します。

メッセージ ID は、メッセージの送信時に `IMessageProducer.send()` 呼び出しによって設定されます。受信されたメッセージについては、受信側アプリケーションが `JMSMessageID` を設定してメッセージ ID を変更していない限り、このメソッドでは、メッセージ送信時に `IMessageProducer.send()` 呼び出しによって設定されたメッセージ ID が戻されます。

メッセージにメッセージ ID がない場合、このメソッドは `NULL` を戻します。

例外:

- `XMSEException`

JMSPriority - *JMSPriority* の取得および設定

インターフェース:

```
Int32 JMSPriority
{
    get;
    set;
}
```

メッセージの優先順位を取得して設定します。

優先順位は、メッセージの送信時に `IMessageProducer.send()` 呼び出しによって設定されます。値は 0 (最低優先順位) から 9 (最高優先順位) までの整数です。

新規に作成されて送信されていないメッセージの場合、送信側アプリケーションが `JMSPriority` を設定して異なる優先順位を設定していない限り、優先順位は 4 です。受信されたメッセージについては、受信側アプリケーションが `JMSPriority` を設定して優先順位を変更していない限り、このメソッドでは、メッセージ送信時に `IMessageProducer.send()` 呼び出しによって設定された優先順位が戻されます。

例外:

- `XMSEException`

JMSRedelivered - *JMSRedelivered* の取得および設定

インターフェース:

```
Boolean JMSRedelivered
{
    get;
    set;
}
```

メッセージが再配信されるかどうかの標識を取得し、メッセージが再配信されるかどうかを示します。標識は、メッセージの受信時に `IMessageConsumer.receive()` 呼び出しによって設定されます。

このプロパティの値は、以下のとおりです。

- メッセージが再配信される場合は、`True` です。
- メッセージが再配信されない場合は、`False` です。

ブローカーへのリアルタイム接続の場合、常に値は `False` です。

メッセージの送信前に `JMSRedelivered` が設定した再配信の標識は、メッセージの送信時の `IMessageProducer.send()` 呼び出しでは無視され、メッセージ受信時の `IMessageConsumer.receive()` 呼び出しでは無視されて置き換えられます。ただし、`JMSRedelivered` を使用して、受信したメッセージの標識を変更することもできます。

例外:

- `XMSEException`

JMSReplyTo - JMSReplyTo の取得および設定

インターフェース:

```
IDestination JMSReplyTo
{
    get;
    set;
}
```

メッセージに対する応答が送信される宛先を取得および設定します。

このプロパティの値は、メッセージに対する応答が送信される宛先の **Destination** オブジェクトです。Destination オブジェクトがヌルの場合は、応答を想定していないという意味です。

例外:

- XMSEException

JMSTimestamp - JMSTimestamp の取得および設定

インターフェース:

```
Int64 JMSTimestamp
{
    get;
    set;
}
```

メッセージが送信された時刻を取得して設定します。

タイム・スタンプは、メッセージの送信時に `IMessageProducer.send()` 呼び出しによって設定され、1970 年 1 月 1 日 00:00:00 GMT からのミリ秒で表されます。

新規に作成されて送信されていないメッセージの場合、送信側アプリケーションが **JMSTimestamp** を設定して異なるタイム・スタンプを設定していない限り、タイム・スタンプは 0 です。受信されたメッセージについては、受信側アプリケーションが **JMSTimestamp** を設定してタイム・スタンプを変更していない限り、このメソッドでは、メッセージ送信時に `IMessageProducer.send()` 呼び出しによって設定されたタイム・スタンプが戻されます。

例外:

- XMSEException

注:

1. タイム・スタンプが未定義の場合、このメソッドは 0 を返しますが例外はスローしません。

JMSType - JMSType の取得および設定

インターフェース:

```
String JMSType
{
    get;
    set;
}
```

メッセージのタイプを取得および設定します。

JMSType の値は、メッセージのタイプをカプセル化しているストリングです。データ変換が必要な場合、この値は、変換後のタイプになります。

例外:

- XMSEException

PropertyNames - プロパティの取得

インターフェース:

```
System.Collections.IEnumerator PropertyNames
{
    get;
}
```

メッセージの名前プロパティの列挙を取得します。

例外:

- XMSEException

方法

Acknowledge - 応答

インターフェース:

```
void Acknowledge();
```

このメッセージと、それ以前にセッションが受信したすべての未承認メッセージを確認します。

セッションの応答モードが `AcknowledgeMode.ClientAcknowledge` である場合、アプリケーションはこのメソッドを呼び出すことができます。セッションがその他の応答モードであるか、セッションが処理中である場合は、このメソッドの呼び出しは無視されます。

受信されたが、応答されていないメッセージは、再配信される可能性があります。

メッセージの応答について詳しくは、[../develop/xms_cmesak.dita#xms_cmesak](http://develop.xms_cmesak.dita#xms_cmesak) を参照してください。

パラメーター:

なし

戻り値:

Void

例外:

- XMSEException
- IllegalStateException

ClearBody - 本体のクリア

インターフェース:

```
void ClearBody();
```

メッセージの本体をクリアします。ヘッダー・フィールドおよびメッセージ・プロパティはクリアされません。

アプリケーションがメッセージ本体をクリアすると、本体は、新規に作成されたメッセージ内の空の本体と同じ状態で残ります。新規に作成されたメッセージ内の空の本体の状態は、メッセージ本体のタイプによって異なります。詳しくは、[XMS メッセージの本体](#) を参照してください。

アプリケーションは、本体の状態にかかわらず、いつでもメッセージ本体をクリアできます。メッセージ本体が読み取り専用の場合、アプリケーションが本体に書き込むことができる唯一の方法は、まずアプリケーションが本体をクリアすることです。

パラメーター:

なし

戻り値:

Void

例外:

- XMSEException

ClearProperties - プロパティのクリア

インターフェース:

```
void ClearProperties();
```

メッセージのプロパティをクリアします。ヘッダー・フィールドおよびメッセージ本体はクリアされません。

アプリケーションがメッセージのプロパティをクリアした場合、プロパティは読み書き可能になります。

アプリケーションは、プロパティの状態にかかわらず、いつでもメッセージのプロパティをクリアできます。メッセージのプロパティが読み取り専用の場合、プロパティを書き込み可能にすることができ、唯一の方法は、アプリケーションがまずプロパティをクリアすることです。

パラメーター:

なし

戻り値:

Void

例外:

- XMSEException

PropertyExists - プロパティの存在の検査

インターフェース:

```
Boolean PropertyExists(String propertyName);
```

メッセージに、指定された名前のプロパティがあるかどうかをチェックします。

パラメーター:

propertyName (入力)

プロパティの名前をカプセル化している String オブジェクト。

戻り値:

- メッセージに、指定された名前のプロパティがある場合は、True です。
- メッセージに、指定された名前のプロパティがない場合は、False です。

例外:

- XMSEException

継承されたプロパティおよびメソッド

以下のメソッドは、*IPROPERTYContext* インターフェースから継承されています。

[GetBooleanProperty](#)、[GetByteProperty](#)、[GetBytesProperty](#)、[GetCharProperty](#)、[GetDoubleProperty](#)、[GetFloatProperty](#)、[GetIntProperty](#)、[GetLongProperty](#)、[GetObjectProperty](#)、[GetShortProperty](#)、[GetStringProperty](#)、[SetBooleanProperty](#)、[SetByteProperty](#)、[SetBytesProperty](#)、[SetCharProperty](#)、[SetDoubleProperty](#)、[SetFloatProperty](#)、[SetIntProperty](#)、[SetLongProperty](#)、[SetObjectProperty](#)、[SetShortProperty](#)、[SetStringProperty](#)

IMessageConsumer

アプリケーションは、メッセージ・コンシューマーを使用して、宛先に送信されたメッセージを受信します。

継承の階層:

```
IBM.XMS.IPropertyContext
|
+---- IBM.XMS.IMessageConsumer
```

MessageConsumer オブジェクトの XMS 定義プロパティのリストについては、[2051 ページの『MessageConsumer のプロパティ』](#)を参照してください。

.NET プロパティ

MessageListener - メッセージ・リスナーの取得および設定

インターフェース:

```
MessageListener MessageListener
{
    get;
    set;
}
```

メッセージ・コンシューマーに登録されているメッセージ・リスナーを取得し、メッセージ・リスナーをメッセージ・コンシューマーに登録します。

メッセージ・コンシューマーにメッセージ・リスナーが登録されていない場合、*MessageListener* は NULL です。メッセージ・コンシューマーにメッセージ・リスナーが既に登録されている場合、代わりに NULL を指定することによって、登録を取り消すことができます。

メッセージ・リスナーの使用について詳しくは、[.NET でのメッセージおよび例外リスナーの使用](#)を参照してください。

例外:

- *XMSEException*

MessageSelector - メッセージ・セクターの取得

インターフェース:

```
String MessageSelector
{
    get;
}
```

メッセージ・コンシューマーのメッセージ・セクターを取得します。戻り値は、メッセージ・セクター式をカプセル化している String オブジェクトです。データ変換が必要な場合、この値は、変換後のメッセージ・セクター式になります。メッセージ・コンシューマーにメッセージ・セクターが存在しない場合、*MessageSelector* の値はヌルの String オブジェクトです。

例外:

- *XMSEException*

方法

Close - メッセージ・コンシューマーのクローズ

インターフェース:

```
void Close();
```

メッセージ・コンシューマーを閉じます。

アプリケーションが、既に閉じているメッセージ・コンシューマーを閉じようとした場合、呼び出しは無視されます。

パラメーター:

なし

戻り値:

Void

例外:

- XMSEException

Receive - 受信

インターフェース:

```
IMessage Receive();
```

メッセージ・コンシューマーの次のメッセージを受信します。この呼び出しでは、無期限にメッセージを待機し続けるか、メッセージ・コンシューマーがクローズされるまで待機します。

パラメーター:

なし

戻り値:

Message オブジェクトへのポインター。呼び出しがメッセージを待機している間にメッセージ・コンシューマーを閉じると、メソッドは、ヌルの Message オブジェクトを指すポインターを戻します。

例外:

- XMSEException

Receive - 受信 (待機間隔あり)

インターフェース:

```
IMessage Receive(Int64 delay);
```

メッセージ・コンシューマーの次のメッセージを受信します。この呼び出しは、指定の期間だけメッセージを待機するか、メッセージ・コンシューマーがクローズされるまで待機します。

パラメーター:

delay (入力)

呼び出しがメッセージを待機する時間 (ミリ秒単位)。待機間隔を 0 と指定した場合、呼び出しは無期限にメッセージを待機します。

戻り値:

Message オブジェクトへのポインター。待機間隔の間にメッセージが到着しなかった場合や、呼び出しがメッセージを待機している間にメッセージ・コンシューマーを閉じた場合、メソッドは、ヌルの Message オブジェクトを指すポインターを戻しますが、例外はスローしません。

例外:

- XMSEException

ReceiveNoWait - 待機なしの受信

インターフェース:

```
IMessage ReceiveNoWait();
```

メッセージ・コンシューマーの次のメッセージが即時に受信可能である場合に、そのメッセージを受け取ります。

パラメーター:

なし

戻り値:

Message オブジェクトへのポインター。即時に有効なメッセージがない場合、メソッドは、ヌルの Message オブジェクトを指すポインターを戻します。

例外:

- [XMSEException](#)

継承されたプロパティおよびメソッド

以下のメソッドは、[IPropertyContext](#) インターフェースから継承されています。

[GetBooleanProperty](#)、[GetByteProperty](#)、[GetBytesProperty](#)、[GetCharProperty](#)、[GetDoubleProperty](#)、[GetFloatProperty](#)、[GetIntProperty](#)、[GetLongProperty](#)、[GetObjectProperty](#)、[GetShortProperty](#)、[GetStringProperty](#)、[SetBooleanProperty](#)、[SetByteProperty](#)、[SetBytesProperty](#)、[SetCharProperty](#)、[SetDoubleProperty](#)、[SetFloatProperty](#)、[SetIntProperty](#)、[SetLongProperty](#)、[SetObjectProperty](#)、[SetShortProperty](#)、[SetStringProperty](#)

MessageEOFException

XMS がこの例外をスローするのは、アプリケーションがバイト・メッセージの本体を読み取っているときに、XMS がバイト・メッセージ・ストリームの終端を検出した場合です。

継承の階層:

```
IBM.XMS.XMSEException
|
+----IBM.XMS.XMSEException
|
+----IBM.XMS.MessageEOFException
```

継承されたプロパティおよびメソッド

以下のメソッドは、[XMSEException](#) インターフェースから継承されています。

[GetErrorCode](#)、[GetLinkedException](#)

MessageFormatException

XMS がこの例外をスローするのは、XMS が無効なフォーマットのメッセージを検出した場合です。

継承の階層:

```
IBM.XMS.XMSEException
|
+----IBM.XMS.XMSEException
|
+----IBM.XMS.MessageFormatException
```

継承されたプロパティおよびメソッド

以下のメソッドは、[XMSEException](#) インターフェースから継承されています。

[GetErrorCode](#)、[GetLinkedException](#)

IMessageListener (代行)

アプリケーションは、メッセージ・リスナーを使用して、メッセージを非同期に受信します。

継承の階層:

なし

代行

MessageListener - メッセージ・リスナー

インターフェース:

```
public delegate void MessageListener(IMessage msg);
```


メッセージを非同期にメッセージ・コンシューマーに配信します。

この代行を実装するメソッドは、接続に登録できます。

メッセージ・リスナーの使用について詳しくは、[.NET でのメッセージおよび例外リスナーの使用](#)を参照してください。

パラメーター:

mesg (入力)

Message オブジェクト。

戻り値:

Void

MessageNotReadableException

XMS がこの例外をスローするのは、書き込み専用になっているメッセージの本体をアプリケーションが読み取ろうとした場合です。

継承の階層:

```
IBM.XMS.XMSEException
|
+----IBM.XMS.XMSEException
|
+----IBM.XMS.MessageNotReadableException
```

継承されたプロパティおよびメソッド

以下のメソッドは、[XMSEException](#) インターフェースから継承されています。

[GetErrorCode](#)、[GetLinkedException](#)

MessageNotWritableException

XMS がこの例外をスローするのは、読み取り専用になっているメッセージの本体にアプリケーションが書き込もうとした場合です。

継承の階層:

```
IBM.XMS.XMSEException
|
+----IBM.XMS.XMSEException
|
+----IBM.XMS.MessageNotWritableException
```

継承されたプロパティおよびメソッド

以下のメソッドは、[XMSEException](#) インターフェースから継承されています。

[GetErrorCode](#)、[GetLinkedException](#)

IMessageProducer

アプリケーションは、メッセージ・プロデューサーを使用して、メッセージを宛先に送信します。

継承の階層:

```
IBM.XMS.IPropertyContext
|
+----IBM.XMS.IMessageProducer
```

MessageProducer オブジェクトの XMS 定義プロパティのリストについては、[2051 ページの『MessageProducer のプロパティ』](#)を参照してください。

.NET プロパティ

DeliveryMode - デフォルト送達モードの取得および設定

インターフェース:

```
DeliveryMode DeliveryMode
{
    get;
    set;
}
```

メッセージ・プロデューサーによって送信されるメッセージのデフォルト送達モードを取得および設定します。

デフォルトの送達モードは、以下の値のいずれかです。

DeliveryMode.Persistent
DeliveryMode.NonPersistent

ブローカーへのリアルタイム接続の場合、値は DeliveryMode.NonPersistent である必要があります。

デフォルト値は DeliveryMode.Persistent です。ただし、デフォルト値が DeliveryMode.NonPersistent となるブローカーへのリアルタイム接続の場合を除きます。

例外:

- XMSEException

Destination - 宛先の取得

インターフェース:

```
IDestination Destination
{
    get;
}
```

メッセージ・プロデューサーの宛先を取得します。

パラメーター:

なし

戻り値:

Destination オブジェクト。メッセージ・プロデューサーに宛先が存在しない場合、このメソッドはヌルの Destination オブジェクトを返します。

例外:

- XMSEException

DisableMsgID - メッセージ ID の使用不可化フラグの取得および設定

インターフェース:

```
Boolean DisableMessageID
{
    get;
    set;
}
```

受信側アプリケーションにとって、メッセージ・プロデューサーにより送信されるメッセージにメッセージ ID が含まれている必要があるかどうかの標識を取得し、受信側アプリケーションにとって、メッセージ・プロデューサーにより送信されるメッセージにメッセージ ID が含まれている必要があるかどうかを示します。

キュー・マネージャーへの接続またはブローカーへのリアルタイム接続では、このフラグは無視されます。サービス統合バスへの接続では、フラグが尊重されます。

DisabledMsgID の値は、以下のとおりです。

- 受信側アプリケーションにとって、メッセージ・プロデューサーにより送信されるメッセージにメッセージ ID が含まれている必要がない場合は、True です。
- 受信側アプリケーションにとって、メッセージ・プロデューサーにより送信されるメッセージにメッセージ ID が含まれている必要がある場合は、False です。

例外:

- XMSEException

DisableMsgTS - タイム・スタンプの使用不可化フラグの取得および設定

インターフェース:

```
Boolean DisableMessageTimestamp
{
    get;
    set;
}
```

受信側アプリケーションにとって、メッセージ・プロデューサーにより送信されるメッセージにタイム・スタンプが含まれている必要があるかどうかの標識を取得し、受信側アプリケーションにとって、メッセージ・プロデューサーにより送信されるメッセージにタイム・スタンプが含まれている必要があるかどうかを示します。

ブローカーへのリアルタイム接続では、このフラグは無視されます。キュー・マネージャーへの接続またはサービス統合バスへの接続では、フラグは尊重されます。

DisableMsgTS の値は、以下のとおりです。

- 受信側アプリケーションにとって、メッセージ・プロデューサーにより送信されるメッセージにタイム・スタンプが含まれている必要がない場合は、True です。
- 受信側アプリケーションにとって、メッセージ・プロデューサーにより送信されるメッセージにタイム・スタンプが含まれている必要がある場合は、False です。

戻り値:

例外:

- XMSEException

Priority - デフォルト優先順位の取得および設定

インターフェース:

```
Int32 Priority
{
    get;
    set;
}
```

メッセージ・プロデューサーによって送信されるメッセージのデフォルト優先順位を取得および設定します。

デフォルトのメッセージ優先順位の値は、0 (最低優先順位) から 9 (最高優先順位) までの整数です。

ブローカーへのリアルタイム接続では、メッセージの優先順位は無視されます。

例外:

- XMSEException

TimeToLive - デフォルト存続時間の取得および設定

インターフェース:

```
Int64 TimeToLive
{
    get;
}
```

```
    set;  
}
```

メッセージが有効期限切れになるまでのデフォルトの時間の長さを取得して設定します。

時間は、メッセージ・プロデューサーがメッセージを送信した時刻とデフォルトの存続時間を基に、ミリ秒単位で測定します。値 0 は、メッセージの有効期限がないことを意味します。

ブローカーへのリアルタイム接続の場合、この値は常に 0 です。

例外:

- XMSEException

方法

Close - メッセージ・プロデューサーのクローズ

インターフェース:

```
void Close();
```

メッセージ・プロデューサーを閉じます。

アプリケーションが、既に閉じているメッセージ・プロデューサーを閉じようとした場合、呼び出しは無視されます。

パラメーター:

なし

戻り値:

Void

例外:

- XMSEException

Send - 送信

インターフェース:

```
void Send(IMessage msg) ;
```

メッセージ・プロデューサーが作成されたときに指定された宛先にメッセージを送信します。メッセージ・プロデューサーのデフォルト送達モード、優先順位、および存続時間を使用してメッセージを送信します。

パラメーター:

msg (入力)

Message オブジェクト。

戻り値:

Void

例外:

- XMSEException
- MessageFormatException
- InvalidDestinationException

Send - 送信 (送達モード、優先順位、および存続時間を指定)

インターフェース:

```
void Send(IMessage msg,  
          DeliveryMode deliveryMode,
```

```
Int32 priority,  
Int64 timeToLive);
```

メッセージ・プロデューサーが作成されたときに指定された宛先にメッセージを送信します。指定された送達モード、優先順位、および存続時間を使用してメッセージを送信します。

パラメーター:

msg (入力)

Message オブジェクト。

deliveryMode (入力)

メッセージの送達モード。以下の値のいずれかでなければなりません。

`DeliveryMode.Persistent`

`DeliveryMode.NonPersistent`

ブローカーへのリアルタイム接続の場合、値は `DeliveryMode.NonPersistent` である必要があります。

priority (入力)

メッセージの優先順位。値は 0 (最低優先順位) から 9 (最高優先順位) までの整数です。ブローカーへのリアルタイム接続では、値は無視されます。

timeToLive (入力)

メッセージの存続時間 (ミリ秒単位)。値 0 は、メッセージの有効期限がないことを意味します。ブローカーへのリアルタイム接続の場合、値は 0 である必要があります。

戻り値:

Void

例外:

- `XMSEException`
- `MessageFormatException`
- `InvalidDestinationException`
- `IllegalStateException`

送信 (指定された宛先へ)

インターフェース:

```
void Send(IDestination dest, IMessage msg) ;
```

メッセージ・プロデューサーが作成されたときに宛先が指定されていないメッセージ・プロデューサーを使用している場合は、指定された宛先にメッセージを送信します。メッセージ・プロデューサーのデフォルト送達モード、優先順位、および存続時間を使用してメッセージを送信します。

通常、メッセージ・プロデューサーの作成時には宛先を指定しますが、宛先を指定しない場合、メッセージの送信ごとに宛先を指定する必要があります。

パラメーター:

dest (入力)

Destination オブジェクト。

msg (入力)

Message オブジェクト。

戻り値:

Void

例外:

- `XMSEException`
- `MessageFormatException`
- `InvalidDestinationException`

Send - 送信 (送達モード、優先順位、および存続時間を指定して、指定された宛先へ)

インターフェース:

```
void Send(IDestination dest,
          IMessage msg,
          DeliveryMode deliveryMode,
          Int32 priority,
          Int64 timeToLive) ;
```

メッセージ・プロデューサーが作成されたときに宛先が指定されていないメッセージ・プロデューサーを使用している場合は、指定された宛先にメッセージを送信します。指定された送達モード、優先順位、および存続時間を使用してメッセージを送信します。

通常、メッセージ・プロデューサーの作成時には宛先を指定しますが、宛先を指定しない場合、メッセージの送信ごとに宛先を指定する必要があります。

パラメーター:

dest (入力)

Destination オブジェクト。

msg (入力)

Message オブジェクト。

deliveryMode (入力)

メッセージの送達モード。以下の値のいずれかでなければなりません。

`DeliveryMode.Persistent`

`DeliveryMode.NonPersistent`

ブローカーへのリアルタイム接続の場合、値は `DeliveryMode.NonPersistent` である必要があります。

priority (入力)

メッセージの優先順位。値は 0 (最低優先順位) から 9 (最高優先順位) までの整数です。ブローカーへのリアルタイム接続では、値は無視されます。

timeToLive (入力)

メッセージの存続時間 (ミリ秒単位)。値 0 は、メッセージの有効期限がないことを意味します。ブローカーへのリアルタイム接続の場合、値は 0 である必要があります。

戻り値:

`Void`

例外:

- `XMSEException`
- `MessageFormatException`
- `InvalidDestinationException`
- `IllegalStateException`

継承されたプロパティおよびメソッド

以下のメソッドは、`IPROPERTYContext` インターフェースから継承されています。

[`GetBooleanProperty`](#)、[`GetByteProperty`](#)、[`GetBytesProperty`](#)、[`GetCharProperty`](#)、[`GetDoubleProperty`](#)、[`GetFloatProperty`](#)、[`GetIntProperty`](#)、[`GetLongProperty`](#)、[`GetObjectProperty`](#)、[`GetShortProperty`](#)、[`GetStringProperty`](#)、[`SetBooleanProperty`](#)、[`SetByteProperty`](#)、[`SetBytesProperty`](#)、[`SetCharProperty`](#)、[`SetDoubleProperty`](#)、[`SetFloatProperty`](#)、[`SetIntProperty`](#)、[`SetLongProperty`](#)、[`SetObjectProperty`](#)、[`SetShortProperty`](#)、[`SetStringProperty`](#)

IOBJECTMessage

オブジェクト・メッセージは、シリアライズされた Java オブジェクトまたは .NET オブジェクトから構成されるメッセージのことです。

継承の階層:

```
IBM.XMS.IPropertyContext
|
+----IBM.XMS.IMessage
|
+----IBM.XMS.IObjectMessage
```

.NET プロパティー

Object - オブジェクトをバイトとして取得および設定

インターフェース:

```
System.Object Object
{
    get;
    set;
}

Byte[] GetObject();
```

オブジェクト・メッセージの本体を形成するオブジェクトを取得および設定します。

例外:

- [XMSEException](#)
- [MessageNotReadableException](#)
- [MessageEOFException](#)
- [MessageNotWritableException](#)

継承されたプロパティーおよびメソッド

以下のプロパティーは、[IMessage](#) インターフェースから継承されています。

[JMSCorrelationID](#)、[JMSDeliveryMode](#)、[JMSDestination](#)、[JMSEExpiration](#)、[JMSMessageID](#)、[JMSPriority](#)、[JMSRedelivered](#)、[JMSReplyTo](#)、[JMSTimestamp](#)、[JMSType](#)、[Properties](#)

以下のメソッドは、[IMessage](#) インターフェースから継承されています。

[clearBody](#)、[clearProperties](#)、[PropertyExists](#)

以下のメソッドは、[IPropertyContext](#) インターフェースから継承されています。

[GetBooleanProperty](#)、[GetByteProperty](#)、[GetBytesProperty](#)、[GetCharProperty](#)、[GetDoubleProperty](#)、[GetFloatProperty](#)、[GetIntProperty](#)、[GetLongProperty](#)、[GetObjectProperty](#)、[GetShortProperty](#)、[GetStringProperty](#)、[SetBooleanProperty](#)、[SetByteProperty](#)、[SetBytesProperty](#)、[SetCharProperty](#)、[SetDoubleProperty](#)、[SetFloatProperty](#)、[SetIntProperty](#)、[SetLongProperty](#)、[SetObjectProperty](#)、[SetShortProperty](#)、[SetStringProperty](#)

IPropertyContext

[IPropertyContext](#) は、プロパティーを取得および設定するメソッドを含む抽象スーパークラスです。これらのメソッドは、その他のクラスによって継承されます。

継承の階層:

なし

方法

GetBooleanProperty - *boolean* プロパティーの取得

インターフェース:

```
Boolean GetBooleanProperty(String property_name);
```

指定した名前を持つ boolean プロパティの値を取得します。

パラメーター:

property_name (入力)

プロパティの名前をカプセル化している String オブジェクト。

戻り値:

プロパティの値。

スレッド・コンテキスト:

サブクラスによって決定

例外:

- XMSEException

GetByteProperty - バイト・プロパティの取得

インターフェース:

```
Byte    GetByteProperty(String property_name) ;
Int16   GetSignedByteProperty(String property_name) ;
```

名前で識別されるバイト・プロパティの値を取得します。

パラメーター:

property_name (入力)

プロパティの名前をカプセル化している String オブジェクト。

戻り値:

プロパティの値。

スレッド・コンテキスト:

サブクラスによって決定

例外:

- XMSEException

GetBytesProperty - バイト配列プロパティの取得

インターフェース:

```
Byte[]  GetBytesProperty(String property_name) ;
```

名前で識別されるバイト配列プロパティの値を取得します。

パラメーター:

property_name (入力)

プロパティの名前をカプセル化している String オブジェクト。

戻り値:

配列のバイト数。

スレッド・コンテキスト:

サブクラスによって決定

例外:

- XMSEException

GetCharProperty - 文字プロパティの取得

インターフェース:

```
Char    GetCharProperty(String property_name) ;
```

名前で識別される 2 バイト文字プロパティの値を取得します。

パラメーター:

property_name (入力)

プロパティの名前をカプセル化している String オブジェクト。

戻り値:

プロパティの値。

スレッド・コンテキスト:

サブクラスによって決定

例外:

- XMSEException

GetDoubleProperty - 倍精度浮動小数点プロパティの取得

インターフェース:

```
Double GetDoubleProperty(String property_name) ;
```

名前で識別される倍精度浮動小数点プロパティの値を取得します。

パラメーター:

property_name (入力)

プロパティの名前をカプセル化している String オブジェクト。

戻り値:

プロパティの値。

スレッド・コンテキスト:

サブクラスによって決定

例外:

- XMSEException

GetFloatProperty - 浮動小数点プロパティの取得

インターフェース:

```
Single GetFloatProperty(String property_name) ;
```

名前で識別される浮動小数点プロパティの値を取得します。

パラメーター:

property_name (入力)

プロパティの名前をカプセル化している String オブジェクト。

戻り値:

プロパティの値。

スレッド・コンテキスト:

サブクラスによって決定

例外:

- XMSEException

GetIntProperty - *GetIntProperty*

インターフェース:

```
Int32 GetIntProperty(String property_name) ;
```

名前で識別される整数プロパティの値を取得します。

パラメーター:

property_name (入力)

プロパティの名前をカプセル化している String オブジェクト。

戻り値:

プロパティの値。

スレッド・コンテキスト:

サブクラスによって決定

例外:

- XMSEException

GetLongProperty - 長整数プロパティの取得

インターフェース:

```
Int64 GetLongProperty(String property_name) ;
```

名前で識別される長整数プロパティの値を取得します。

パラメーター:

property_name (入力)

プロパティの名前をカプセル化している String オブジェクト。

戻り値:

プロパティの値。

スレッド・コンテキスト:

サブクラスによって決定

例外:

- XMSEException

GetObjectProperty - オブジェクト・プロパティの取得

インターフェース:

```
Object GetObjectProperty( String property_name) ;
```

名前で識別されるプロパティの値およびデータ・タイプを取得します。

パラメーター:

property_name (入力)

プロパティの名前をカプセル化している String オブジェクト。

戻り値:

オブジェクト・タイプが以下のいずれかであるプロパティの値。

Boolean
Byte
Byte[]
Char
Double
Single
Int32
Int64
Int16
String

スレッド・コンテキスト:

サブクラスによって決定

例外:

- XMSEException

GetShortProperty - 短整数プロパティの取得

インターフェース:

```
Int16 GetShortProperty(String property_name) ;
```

名前で識別される短整数プロパティの値を取得します。

パラメーター:

property_name (入力)

プロパティの名前をカプセル化している String オブジェクト。

戻り値:

プロパティの値。

スレッド・コンテキスト:

サブクラスによって決定

例外:

- XMSEException

GetStringProperty - *GetStringProperty*

インターフェース:

```
String GetStringProperty(String property_name) ;
```

名前で識別されるストリング・プロパティの値を取得します。

パラメーター:

property_name (入力)

プロパティの名前をカプセル化している String オブジェクト。

戻り値:

プロパティの値を表すストリングをカプセル化している String オブジェクト。データ変換が必要な場合、この値は、変換後のストリングになります。

スレッド・コンテキスト:

サブクラスによって決定

例外:

- XMSEException

SetBooleanProperty - *boolean* プロパティの設定

インターフェース:

```
void SetBooleanProperty( String property_name, Boolean value) ;
```

名前で識別される boolean プロパティの値を設定します。

パラメーター:

property_name (入力)

プロパティの名前をカプセル化している String オブジェクト。

value (入力)

プロパティの値。

戻り値:

Void

スレッド・コンテキスト:

サブクラスによって決定

例外:

- XMSEException
- MessageNotWritableException

SetByteProperty - バイト・プロパティの設定

インターフェース:

```
void SetByteProperty( String property_name, Byte value) ;  
void SetSignedByteProperty( String property_name, Int16 value) ;
```

名前で識別されるバイト・プロパティの値を設定します。

パラメーター:

property_name (入力)

プロパティの名前をカプセル化している String オブジェクト。

value (入力)

プロパティの値。

戻り値:

Void

スレッド・コンテキスト:

サブクラスによって決定

例外:

- XMSEException
- MessageNotWritableException

SetBytesProperty - バイト配列プロパティの設定

インターフェース:

```
void SetBytesProperty( String property_name, Byte[] value ) ;
```

名前で識別されるバイト配列プロパティの値を設定します。

パラメーター:

property_name (入力)

プロパティの名前をカプセル化している String オブジェクト。

value (入力)

バイトの配列であるプロパティの値。

戻り値:

Void

スレッド・コンテキスト:

サブクラスによって決定

例外:

- XMSEException
- MessageNotWritableException

SetCharProperty - 文字プロパティの設定

インターフェース:

```
void SetCharProperty( String property_name, Char value) ;
```

名前で識別される 2 バイト文字プロパティの値を設定します。

パラメーター:

property_name (入力)

プロパティの名前をカプセル化している String オブジェクト。

value (入力)

プロパティの値。

戻り値:

Void

スレッド・コンテキスト:

サブクラスによって決定

例外:

- XMSEException
- MessageNotWritableException

SetDoubleProperty - 倍精度浮動小数点プロパティの設定

インターフェース:

```
void SetDoubleProperty( String property_name, Double value) ;
```

名前で識別される倍精度浮動小数点プロパティの値を設定します。

パラメーター:

property_name (入力)

プロパティの名前をカプセル化している String オブジェクト。

value (入力)

プロパティの値。

戻り値:

Void

スレッド・コンテキスト:

サブクラスによって決定

例外:

- XMSEException
- MessageNotWritableException

SetFloatProperty - 浮動小数点プロパティの設定

インターフェース:

```
void SetFloatProperty( String property_name, Single value) ;
```

名前で識別される浮動小数点プロパティの値を設定取得します。

パラメーター:

property_name (入力)

プロパティの名前をカプセル化している String オブジェクト。

value (入力)

プロパティの値。

戻り値:

Void

スレッド・コンテキスト:

サブクラスによって決定

例外:

- XMSEException
- MessageNotWritableException

SetIntProperty - 整数プロパティの設定

インターフェース:

```
void SetIntProperty( String property_name, Int32 value) ;
```

名前で識別される整数プロパティの値を設定します。

パラメーター:

property_name (入力)

プロパティの名前をカプセル化している String オブジェクト。

value (入力)

プロパティの値。

戻り値:

Void

スレッド・コンテキスト:

サブクラスによって決定

例外:

- XMSEException
- MessageNotWritableException

SetLongProperty - 長整数プロパティの設定

インターフェース:

```
void SetLongProperty( String property_name, Int64 value) ;
```

名前で識別される長整数プロパティの値を設定します。

パラメーター:

property_name (入力)

プロパティの名前をカプセル化している String オブジェクト。

value (入力)

プロパティの値。

戻り値:

Void

スレッド・コンテキスト:

サブクラスによって決定

例外:

- XMSEException
- MessageNotWritableException

SetObjectProperty - オブジェクト・プロパティの設定

インターフェース:

```
void SetObjectProperty( String property_name, Object value) ;
```

名前で識別されるプロパティの値およびデータ・タイプを設定します。

パラメーター:

property_name (入力)

プロパティの名前をカプセル化している String オブジェクト。

objectType (入力)

プロパティの値。以下のオブジェクト・タイプのいずれかでなければなりません。

Boolean
Byte
Byte[]
Char
Double
Single
Int32
Int64
Int16
String

value (入力)

バイトの配列としてのプロパティの値。

length (入力)

配列のバイト数。

戻り値:

Void

スレッド・コンテキスト:

サブクラスによって決定

例外:

- XMSEException
- MessageNotWritableException

SetShortProperty - 短整数プロパティの設定

インターフェース:

```
void SetShortProperty( String property_name, Int16 value) ;
```

名前で識別される短整数プロパティの値を設定します。

パラメーター:

property_name (入力)

プロパティの名前をカプセル化している String オブジェクト。

value (入力)

プロパティの値。

戻り値:

Void

スレッド・コンテキスト:

サブクラスによって決定

例外:

- XMSEException
- MessageNotWritableException

SetStringProperty - ストリング・プロパティの設定

インターフェース:

```
void SetStringProperty( String property_name, String value);
```

名前で識別されるストリング・プロパティの値を設定します。

パラメーター:

property_name (入力)

プロパティの名前をカプセル化している String オブジェクト。

value (入力)

プロパティの値を表すストリングをカプセル化している String オブジェクト。

戻り値:

Void

スレッド・コンテキスト:

サブクラスによって決定

例外:

- XMSEException
- MessageNotWritableException

IQueueBrowser

アプリケーションは、キュー・ブラウザーを使用して、キュー上のメッセージを参照します。その際にメッセージは除去されません。

継承の階層:

```
IBM.XMS.IPropertyContext
System.Collections.IEnumerable
|
+---- IBM.XMS.IQueueBrowser
```

.NET プロパティ

MessageSelector - メッセージ・セレクターの取得

インターフェース:

```
String MessageSelector
{
    get;
}
```

キュー・ブラウザーのメッセージ・セレクターを取得します。

メッセージ・セレクターは、メッセージ・セレクター式をカプセル化している String オブジェクトです。データ変換が必要な場合、この値は、変換後のメッセージ・セレクター式になります。キュー・ブラウザーにメッセージ・セレクターが存在しない場合、このメソッドはヌルの String オブジェクトを戻します。

例外:

- XMSEException

Queue - キューの取得

インターフェース:

```
IDestination Queue
{
    get;
}
```


キュー・ブラウザーに関連付けられたキューを、キューを表す宛先オブジェクトとして取得します。

例外:

- `XMSEException`

方法

Close - キュー・ブラウザーのクローズ

インターフェース:

```
void Close();
```

キュー・ブラウザーを閉じます。

アプリケーションが、既に閉じているキュー・ブラウザーを閉じようとした場合、呼び出しは無視されます。

パラメーター:

なし

戻り値:

`Void`

例外:

- `XMSEException`

GetEnumerator - メッセージの取得

インターフェース:

```
IEnumerator GetEnumerator();
```

キュー上のメッセージのリストを取得します。

このメソッドは、`Message` オブジェクトのリストをカプセル化する列挙子を返します。`Message` オブジェクトの順序は、メッセージがキューから取り出される順序と同じです。アプリケーションは列挙子を使用して、各メッセージを順番に参照できます。

メッセージがキューに書き込まれたり削除されたりすると、列挙子は動的に更新されます。アプリケーションがキュー上の次のメッセージを参照するために `IEnumerator.MoveNext()` を呼び出すたびに、メッセージはキューの現在の内容を反映しています。

アプリケーションがキュー・ブラウザーに対してこのメソッドを複数回呼び出すと、そのたびに新しい列挙子が返されます。したがって、アプリケーションは複数の列挙子を使用してキューのメッセージを参照し、キュー内の複数の位置を維持することができます。

パラメーター:

なし

戻り値:

`Iterator` オブジェクト。

例外:

- `XMSEException`

継承されたプロパティおよびメソッド

以下のメソッドは、[IPropertyContext](#) インターフェースから継承されています。

[GetBooleanProperty](#)、[GetByteProperty](#)、[GetBytesProperty](#)、[GetCharProperty](#)、[GetDoubleProperty](#)、[GetFloatProperty](#)、[GetIntProperty](#)、[GetLongProperty](#)、[GetObjectProperty](#)、[GetShortProperty](#)、[GetStringProperty](#)、[SetBooleanProperty](#)、[SetByteProperty](#)、[SetBytesProperty](#)、[SetCharProperty](#)、[SetDoubleProperty](#)、[SetFloatProperty](#)、[SetIntProperty](#)、[SetLongProperty](#)、[SetObjectProperty](#)、[SetShortProperty](#)、[SetStringProperty](#)

Requestor

アプリケーションはリクエスターを使用して、要求メッセージを送信し、応答を待機して受信します。

継承の階層:

なし

コンストラクター

Requestor - リクエスターの作成

インターフェース:

```
Requestor(ISession sess, IDestination dest);
```

リクエスターを作成します。

パラメーター:

sess (入力)

Session オブジェクト。セッションはトランザクション化されてはならず、以下の肯定応答モードのいずれかでなければなりません。

`AcknowledgeMode.AutoAcknowledge`

`AcknowledgeMode.DupsOkAcknowledge`

dest (入力)

アプリケーションが要求メッセージを送信できる宛先を表す Destination オブジェクト。

スレッド・コンテキスト:

リクエスターに関連付けられたセッション

例外:

- `XMSEException`

方法

Close - リクエスターのクローズ

インターフェース:

```
void Close();
```

リクエスターをクローズします。

アプリケーションが、既に閉じているリクエスターを閉じようとした場合、呼び出しは無視されます。

注: アプリケーションがリクエスターを閉じたとき、関連したセッションは閉じません。この点で、XMS の動作は JMS とは異なります。

パラメーター:

なし

戻り値:

`Void`

スレッド・コンテキスト:

任意

例外:

- `XMSEException`

Request - 応答の要求

インターフェース:

```
IMessage Request(IMessage requestMessage);
```

要求メッセージを送信し、要求メッセージを受信したアプリケーションからの応答を待ち、受信します。

このメソッドへの呼び出しは、応答が受信されるまで、またはセッションが終了するまでの、いずれか早い方の時点までブロックします。

パラメーター:

requestMessage (入力)

要求メッセージをカプセル化している Message オブジェクト。

戻り値:

応答メッセージをカプセル化している Message オブジェクトを指すポインター。

スレッド・コンテキスト:

リクエストに関連付けられたセッション

例外:

- [XMSEException](#)

ResourceAllocationException

XMS がこの例外をスローするのは、メソッドが必要とするリソースを XMS が割り振ることができない場合です。

継承の階層:

```
IBM.XMS.XMSEException
|
+----IBM.XMS.XMSEException
|
+----IBM.XMS.ResourceAllocationException
```

継承されたプロパティおよびメソッド

以下のメソッドは、[XMSEException](#) インターフェースから継承されています。

[GetErrorCode](#)、[GetLinkedException](#)

SecurityException

XMS は、アプリケーションを認証するために指定されたユーザー ID とパスワードが拒否された場合に、この例外をスローします。XMS は、権限検査が不合格になり、そのためにメソッドを完了できない場合にもこの例外をスローします。

継承の階層:

```
IBM.XMS.XMSEException
|
+----IBM.XMS.XMSEException
|
+----IBM.XMS.SecurityException
```

継承されたプロパティおよびメソッド

以下のメソッドは、[XMSEException](#) インターフェースから継承されています。

[GetErrorCode](#)、[GetLinkedException](#)

ISession

セッションとは、メッセージ送受信のための単一スレッドのコンテキストのことです。

継承の階層:

```
IBM.XMS.IPropertyContext
|
+---- IBM.XMS.ISession
```

Session オブジェクトの XMS 定義プロパティのリストについては、[2051 ページの『Session のプロパティ』](#)を参照してください。

.NET プロパティ

AcknowledgeMode - 肯定応答モードの取得

インターフェース:

```
AcknowledgeMode AcknowledgeMode
{
    get;
}
```

セッションの肯定応答モードを取得します。

肯定応答モードは、セッション作成時に指定されます。

セッションがトランザクション化されていない場合、肯定応答モードは以下の値のいずれかです。

```
AcknowledgeMode.AutoAcknowledge
AcknowledgeMode.ClientAcknowledge
AcknowledgeMode.DupsOkAcknowledge
```

肯定応答モードについて詳しくは、[メッセージの肯定応答](#)を参照してください。

トランザクション化されているセッションには、肯定応答モードはありません。セッションが処理された場合、メソッドは代わりに `AcknowledgeMode.SessionTransacted` を戻します。

例外:

- `XMSEException`

Transacted - 処理済みであるかどうかの判別

インターフェース:

```
Boolean Transacted
{
    get;
}
```

セッションがトランザクション化されているかどうかを判別します。

トランザクション化されている状態は以下のとおりです。

- セッションがトランザクション化されている場合は、`True` です。
- セッションがトランザクション化されていない場合は、`False` です。

ブローカーへのリアルタイム接続の場合、常にメソッドは `False` を戻します。

例外:

- `XMSEException`

方法

Close - セッションのクローズ

インターフェース:

```
void Close();
```

セッションを閉じます。セッションがトランザクション化されている場合、進行中のトランザクションはロールバックされます。

アプリケーションが、既に閉じているセッションを閉じようとした場合、呼び出しは無視されます。

パラメーター:

なし

戻り値:

Void

スレッド・コンテキスト:

任意

例外:

- XMSEException

Commit - コミット

インターフェース:

```
void Commit();
```

現在のトランザクションで処理されたすべてのメッセージをコミットします。

セッションは、トランザクション化セッションでなければなりません。

パラメーター:

なし

戻り値:

Void

例外:

- XMSEException
- IllegalStateException
- TransactionRolledBackException

CreateBrowser - キュー・ブラウザーの作成

インターフェース:

```
IQueueBrowser CreateBrowser(IDestination queue) ;
```

指定されたキューのキュー・ブラウザーを作成します。

パラメーター:

queue (入力)

キューを表す Destination オブジェクト。

戻り値:

QueueBrowser オブジェクト。

例外:

- XMSEException
- InvalidDestinationException

CreateBrowser - キュー・ブラウザーの作成 (メッセージ・セレクトーを使用)

インターフェース:

```
IQueueBrowser CreateBrowser(IDestination queue, String selector) ;
```

メッセージ・セレクトーを使用して、指定されたキューのキュー・ブラウザーを作成します。

パラメーター:

queue (入力)

キューを表す Destination オブジェクト。

selector (入力)

メッセージ・セレクトー式をカプセル化している String オブジェクト。メッセージ・セレクトー式に一致するプロパティを持つメッセージのみが、キュー・ブラウザーに配信されます。

String オブジェクトがヌルであるとは、キュー・ブラウザー用のメッセージ・セレクトーが存在しないという意味です。

戻り値:

QueueBrowser オブジェクト。

例外:

- XMSEException
- InvalidDestinationException
- InvalidSelectorException

CreateBytesMessage - バイト・メッセージの作成

インターフェース:

```
IBytesMessage CreateBytesMessage();
```

バイト・メッセージを作成します。

パラメーター:

なし

戻り値:

BytesMessage オブジェクト。

例外:

- XMSEException
- IllegalStateException (セッションは終了しています)

CreateConsumer - コンシューマーの作成

インターフェース:

```
IMessageConsumer CreateConsumer(IDestination dest) ;
```

指定された宛先のメッセージ・コンシューマーを作成します。

パラメーター:

dest (入力)

Destination オブジェクト。

戻り値:

MessageConsumer オブジェクト。

例外:

- XMSEException
- InvalidDestinationException

CreateConsumer - コンシューマーの作成 (メッセージ・セレクターを使用)

インターフェース:

```
IMessageConsumer CreateConsumer(IDestination dest,  
                                String selector) ;
```

メッセージ・セレクターを使用して、指定された宛先のメッセージ・コンシューマーを作成します。

パラメーター:

dest (入力)

Destination オブジェクト。

selector (入力)

メッセージ・セレクター式をカプセル化している String オブジェクト。メッセージ・セレクター式に一致するプロパティを持つメッセージのみが、メッセージ・コンシューマーに配信されます。

String オブジェクトがヌルであるとは、メッセージ・コンシューマー用のメッセージ・セレクターが存在しないという意味です。

戻り値:

MessageConsumer オブジェクト。

例外:

- XMSEException
- InvalidDestinationException
- InvalidSelectorException

CreateConsumer - コンシューマーの作成 (メッセージ・セレクターおよびローカル・メッセージ・フラグを使用)

インターフェース:

```
IMessageConsumer CreateConsumer(IDestination dest,  
                                String selector,  
                                Boolean noLocal) ;
```

メッセージ・セレクターを使用し、宛先がトピックの場合はメッセージ・コンシューマーが自身の接続により公開されたメッセージを受信するかどうかを指定して、指定された宛先のメッセージ・コンシューマーを作成します。

パラメーター:

dest (入力)

Destination オブジェクト。

selector (入力)

メッセージ・セレクター式をカプセル化している String オブジェクト。メッセージ・セレクター式に一致するプロパティを持つメッセージのみが、メッセージ・コンシューマーに配信されます。

String オブジェクトがヌルであるとは、メッセージ・コンシューマー用のメッセージ・セレクターが存在しないという意味です。

noLocal (入力)

値 True は、メッセージ・コンシューマーが、自身の接続により公開されたメッセージを受信しないことを意味します。値 False は、メッセージ・コンシューマーが、自身の接続により公開されたメッセージを受信することを意味します。デフォルト値は False です。

戻り値:

MessageConsumer オブジェクト。

例外:

- XMSEException
- InvalidDestinationException

- InvalidSelectorException

CreateDurableSubscriber - 永続サブスクライバーの作成

インターフェース:

```
IMessageConsumer CreateDurableSubscriber(IDestination dest,  
                                           String subscription) ;
```

指定されたトピックの永続サブスクライバーを作成します。

ブローカーへのリアルタイム接続の場合、このメソッドは無効です。

永続サブスクライバーについては、[永続サブスクライバー](#)を参照してください。

パラメーター:

dest (入力)

トピックを表す Destination オブジェクト。 トピックは、一時トピックであってはなりません。

subscription (入力)

永続サブスクリプションを識別する名前をカプセル化している String オブジェクト。 名前は、接続のクライアント ID 内で固有である必要があります。

戻り値:

永続サブスクライバーを表す MessageConsumer オブジェクト。

例外:

- XMSEException
- InvalidDestinationException

CreateDurableSubscriber - 永続サブスクライバーの作成 (メッセージ・セ렉ターおよびローカル・メッセージ・フラグを使用)

インターフェース:

```
IMessageConsumer CreateDurableSubscriber(IDestination dest,  
                                           String subscription,  
                                           String selector,  
                                           Boolean noLocal) ;
```

メッセージ・セ렉ターを使用し、永続サブスクライバーが自身の接続により公開されたメッセージを受信するかどうかを指定して、指定されたトピックの永続サブスクライバーを作成します。

ブローカーへのリアルタイム接続の場合、このメソッドは無効です。

永続サブスクライバーについては、[永続サブスクライバー](#)を参照してください。

パラメーター:

dest (入力)

トピックを表す Destination オブジェクト。 トピックは、一時トピックであってはなりません。

subscription (入力)

永続サブスクリプションを識別する名前をカプセル化している String オブジェクト。 名前は、接続のクライアント ID 内で固有である必要があります。

selector (入力)

メッセージ・セ렉ター式をカプセル化している String オブジェクト。 メッセージ・セ렉ター式に一致するプロパティーを持つメッセージのみが、永続サブスクライバーに配信されます。

String オブジェクトがヌルであるとは、永続サブスクライバー用のメッセージ・セ렉ターが存在しないという意味です。

noLocal (入力)

値 True は、永続サブスクライバーが、自身の接続により公開されたメッセージを受信しないことを意味します。 値 False は、永続サブスクライバーが、自身の接続により公開されたメッセージを受信することを意味します。 デフォルト値は False です。

戻り値:

永続サブスクライバーを表す `MessageConsumer` オブジェクト。

例外:

- `XMSEException`
- `InvalidDestinationException`
- `InvalidSelectorException`

CreateMapMessage - マップ・メッセージの作成

インターフェース:

```
IMapMessage CreateMapMessage();
```

マップ・メッセージを作成します。

パラメーター:

なし

戻り値:

`MapMessage` オブジェクト。

例外:

- `XMSEException`
- `IllegalStateException` (セッションは終了しています)

CreateMessage - メッセージの作成

インターフェース:

```
IMessage CreateMessage();
```

本体を持たないメッセージを作成します。

パラメーター:

なし

戻り値:

`Message` オブジェクト。

例外:

- `XMSEException`
- `IllegalStateException` (セッションは終了しています)

CreateObjectMessage - オブジェクト・メッセージの作成

インターフェース:

```
IObjectMessage CreateObjectMessage();
```

オブジェクト・メッセージを作成します。

パラメーター:

なし

戻り値:

`ObjectMessage` オブジェクト。

例外:

- `XMSEException`
- `IllegalStateException` (セッションは終了しています)

CreateProducer - プロデューサーの作成

インターフェース:

```
IMessageProducer CreateProducer(IDestination dest) ;
```

メッセージを指定された宛先へ送信するメッセージ・プロデューサーを作成します。

パラメーター:

dest (入力)

Destination オブジェクト。

ヌルの Destination オブジェクトを指定すると、宛先のないメッセージ・プロデューサーが作成されます。この場合アプリケーションは、メッセージを送信するためにメッセージ・プロデューサーを使用するたびに、宛先を指定する必要があります。

戻り値:

MessageProducer オブジェクト。

例外:

- XMSEException
- InvalidDestinationException

CreateQueue - キューの作成

インターフェース:

```
IDestination CreateQueue(String queue) ;
```

メッセージング・サーバー内のキューを表すための Destination オブジェクトを作成します。

このメソッドは、メッセージング・サーバー内にキューを作成しません。アプリケーションがこのメソッドを呼び出すためには、その前にキューを作成する必要があります。

パラメーター:

queue (入力)

キューの名前をカプセル化している String オブジェクト、またはキューを識別する Uniform Resource Identifier (URI) をカプセル化している String オブジェクト。

戻り値:

キューを表す Destination オブジェクト。

例外:

- XMSEException

CreateStreamMessage - ストリーム・メッセージの作成

インターフェース:

```
IStreamMessage CreateStreamMessage();
```

ストリーム・メッセージを作成します。

パラメーター:

なし

戻り値:

StreamMessage オブジェクト。

例外:

- XMSEException
- XMS_ILLEGAL_STATE_EXCEPTION

CreateTemporaryQueue - 一時キューの作成

インターフェース:

```
IDestination CreateTemporaryQueue();
```

一時キューを作成します。

一時キューのスコープは接続です。 接続によって作成されたセッションのみが、一時キューを使用できます。

一時キューは、明示的に削除されるまで、あるいは接続が終了するまで存続します。

一時キューについて詳しくは、[一時宛先](#)を参照してください。

パラメーター:

なし

戻り値:

一時キューを表す Destination オブジェクト。

例外:

- XMSEException

CreateTemporaryTopic - 一時トピックの作成

インターフェース:

```
IDestination CreateTemporaryTopic();
```

一時トピックを作成します。

一時トピックのスコープは接続です。 接続によって作成されたセッションのみが、一時トピックを使用できます。

一時トピックは、明示的な削除または接続終了のいずれかが発生するまで存在します。

一時トピックについて詳しくは、[一時宛先](#)を参照してください。

パラメーター:

なし

戻り値:

一時トピックを表す Destination オブジェクト。

例外:

- XMSEException

CreateTextMessage - テキスト・メッセージの作成

インターフェース:

```
ITextMessage CreateTextMessage();
```

本体が空であるテキスト・メッセージを作成します。

パラメーター:

なし

戻り値:

TextMessage オブジェクト。

例外:

- XMSEException

CreateTextMessage - テキスト・メッセージの作成 (初期化済み)

インターフェース:

```
ITextMessage CreateTextMessage(String initialValue);
```

本体が指定されたテキストで初期化されているテキスト・メッセージを作成します。

パラメーター:

initialValue (入力)

テキスト・メッセージの本体を初期化するためのテキストをカプセル化している String オブジェクト。

なし

戻り値:

TextMessage オブジェクト。

例外:

- XMSEException

CreateTopic - トピックの作成

インターフェース:

```
IDestination CreateTopic(String topic) ;
```

トピックを表すための Destination オブジェクトを作成します。

パラメーター:

topic (入力)

トピックの名前をカプセル化している String オブジェクト、またはトピックを識別する Uniform Resource Identifier (URI) をカプセル化している String オブジェクト。

戻り値:

トピックを表す Destination オブジェクト。

例外:

- XMSEException

Recover - 回復

インターフェース:

```
void Recover();
```

セッションを回復します。メッセージ配信が一旦停止され、その後応答されていない最も古いメッセージを使用して再開されます。

セッションは、トランザクション化セッションであってはなりません。

セッションの回復について詳しくは、[メッセージの肯定応答](#)を参照してください。

パラメーター:

なし

戻り値:

Void

例外:

- XMSEException
- IllegalStateException

Rollback - ロールバック

インターフェース:

```
void Rollback();
```

現在のトランザクションで処理されたすべてのメッセージをロールバックします。

セッションは、トランザクション化セッションでなければなりません。

パラメーター:

なし

戻り値:

Void

例外:

- XMSEException
- IllegalStateException

Unsubscribe - アンサブスクライブ

インターフェース:

```
void Unsubscribe(String subscription);
```

永続サブスクリプションを削除します。メッセージング・サーバーは保守している永続サブスクリプションのレコードを削除し、永続サブスクライバーにこれ以降メッセージを送信しなくなります。

アプリケーションは、以下のいずれの状況でも、永続サブスクリプションを削除することはできません。

- 永続サブスクリプションのアクティブなメッセージ・コンシューマーがあるとき
- コンシュームされたメッセージが保留中のトランザクションの一部であるとき
- コンシュームされたメッセージの確認応答がなかったとき

ブローカーへのリアルタイム接続の場合、このメソッドは無効です。

パラメーター:

subscription (入力)

永続サブスクリプションを識別する名前をカプセル化している String オブジェクト。

戻り値:

Void

例外:

- XMSEException
- InvalidDestinationException
- IllegalStateException

継承されたプロパティおよびメソッド

以下のメソッドは、[IPropertyContext](#) インターフェースから継承されています。

[GetBooleanProperty](#)、[GetByteProperty](#)、[GetBytesProperty](#)、[GetCharProperty](#)、[GetDoubleProperty](#)、[GetFloatProperty](#)、[GetIntProperty](#)、[GetLongProperty](#)、[GetObjectProperty](#)、[GetShortProperty](#)、[GetStringProperty](#)、[SetBooleanProperty](#)、[SetByteProperty](#)、[SetBytesProperty](#)、[SetCharProperty](#)、[SetDoubleProperty](#)、[SetFloatProperty](#)、[SetIntProperty](#)、[SetLongProperty](#)、[SetObjectProperty](#)、[SetShortProperty](#)、[SetStringProperty](#)

IStreamMessage

ストリーム・メッセージとは、本体が値のストリームで構成されるメッセージです。それぞれの値に、関連付けられたデータ・タイプがあります。本体の内容は、順番に読み書きされます。

継承の階層:

```
IBM.XMS.IPropertyContext
|
+----IBM.XMS.IMessage
|
+----IBM.XMS.IStreamMessage
```

アプリケーションがメッセージ・ストリームから値を読み取る場合、その値は XMS によって別のデータ・タイプに変換されることがあります。この形式の暗黙の型変換について詳しくは、[XMS メッセージの本体](#)を参照してください。

方法

ReadBoolean - ブール値の読み取り

インターフェース:

```
Boolean ReadBoolean();
```

メッセージ・ストリームからブール値を読み取ります。

パラメーター:

なし

戻り値:

読み取られるブール値。

例外:

- XMSEException
- MessageNotReadableException
- MessageEOFException

ReadByte - バイトの読み取り

インターフェース:

```
Int16 ReadSignedByte();
Byte ReadByte();
```

メッセージ・ストリームから符号付き 8 ビット整数を読み取ります。

パラメーター:

なし

戻り値:

読み取られるバイト。

例外:

- XMSEException
- MessageNotReadableException
- MessageEOFException

ReadBytes - バイトの読み取り

インターフェース:

```
Int32 ReadBytes(Byte[] array);
```

メッセージ・ストリームからバイトの配列を読み取ります。

パラメーター:

array (入力)

読み取られるバイトの配列が収容されているバッファーとそのバッファーの長さ (単位: バイト)。

配列のバイト数が、バッファーの長さ以下である場合は、配列全体がバッファーに読み取られます。配列のバイト数がバッファーの長さを超える場合、配列の一部でバッファーはいっぱいになり、内部カーソルは次に読み取られるバイトの位置をマークします。次に `readBytes()` を呼び出すと、カーソルの現在位置から始まる配列から、バイトが読み取られます。

入力に `NULL` ポインターを指定すると、呼び出しではそのバイトの配列は読み取られずにスキップオーバーされます。

戻り値:

バッファーに読み取るバイト数。バッファーが部分的に埋まっている場合は、値はバッファーの長さよりも小さく、配列内に読み取るバイトが残っていないことを示します。呼び出し前の配列に読み取り可能なバイトが残っていない場合、値は `XMSC_END_OF_BYTEARRAY` です。

入力に `NULL` ポインターを指定すると、メソッドは値を返しません。

例外:

- `XMSEException`
- `MessageNotReadableException`
- `MessageEOFException`

ReadChar - 文字の読み取り

インターフェース:

```
Char    ReadChar();
```

メッセージ・ストリームから 2 バイト文字を読み取ります。

パラメーター:

なし

戻り値:

読み取られる文字。

例外:

- `XMSEException`
- `MessageNotReadableException`
- `MessageEOFException`

ReadDouble - 倍精度浮動小数点数の読み取り

インターフェース:

```
Double  ReadDouble();
```

メッセージ・ストリームから 8 バイトの倍精度浮動小数点数を読み取ります。

パラメーター:

なし

戻り値:

読み取られる倍精度浮動小数点数。

例外:

- `XMSEException`
- `MessageNotReadableException`
- `MessageEOFException`

ReadFloat - 浮動小数点数の読み取り

インターフェース:

```
Single ReadFloat();
```

メッセージ・ストリームから 4 バイトの浮動小数点数を読み取ります。

パラメーター:

なし

戻り値:

読み取られる浮動小数点数。

例外:

- XMSEException
- MessageNotReadableException
- MessageEOFException

ReadInt - 整数の読み取り

インターフェース:

```
Int32 ReadInt();
```

メッセージ・ストリームから符号付き 32 ビット整数を読み取ります。

パラメーター:

なし

戻り値:

読み取られる整数。

例外:

- XMSEException
- MessageNotReadableException
- MessageEOFException

ReadLong - 長整数の読み取り

インターフェース:

```
Int64 ReadLong();
```

メッセージ・ストリームから符号付き 64 ビット整数を読み取ります。

パラメーター:

なし

戻り値:

読み取られる長整数。

例外:

- XMSEException
- MessageNotReadableException
- MessageEOFException

ReadObject - オブジェクトの読み取り

インターフェース:

```
Object ReadObject();
```

メッセージ・ストリームから値を読み取り、そのデータ・タイプを戻します。

パラメーター:

なし

戻り値:

値。以下のオブジェクト・タイプのいずれかです。

Boolean
Byte
Byte[]
Char
Double
Single
Int32
Int64
Int16
String

例外:

XMSEException

ReadShort - 短整数の読み取り

インターフェース:

```
Int16 ReadShort();
```

メッセージ・ストリームから符号付き 16 ビット整数を読み取ります。

パラメーター:

なし

戻り値:

読み取られる短整数。

例外:

- XMSEException
- MessageNotReadableException
- MessageEOFException

ReadString - スtringの読み取り

インターフェース:

```
String ReadString();
```

メッセージ・ストリームからStringを読み取ります。必要な場合、XMS がString内の文字をローカル・コード・ページに変換します。

パラメーター:

なし

戻り値:

読み取られるStringをカプセル化している String オブジェクト。データ変換が必要な場合、これは変換後のStringです。

例外:

- XMSEException
- MessageNotReadableException
- MessageEOFException

Reset - リセット

インターフェース:

```
void Reset();
```

メッセージの本体を読み取り専用モードにして、カーソルをメッセージ・ストリームの先頭に位置変更します。

パラメーター:

なし

戻り値:

Void

例外:

- XMSEException
- MessageNotReadableException
- MessageEOFException

WriteBoolean - ブール値の書き込み

インターフェース:

```
void WriteBoolean(Boolean value);
```

メッセージ・ストリームへブール値を書き込みます。

パラメーター:

value (入力)

書き込まれるブール値。

戻り値:

Void

例外:

- XMSEException
- MessageNotWritableException

WriteByte - バイトの書き込み

インターフェース:

```
void WriteByte(Byte value);  
void WriteSignedByte(Int16 value);
```

メッセージ・ストリームへバイトを書き込みます。

パラメーター:

value (入力)

書き込まれるバイト。

戻り値:

Void

例外:

- XMSEException
- MessageNotWritableException

WriteBytes - 複数バイトの書き込み

インターフェース:

```
void WriteBytes(Byte[] value);
```

メッセージ・ストリームへバイトの配列を書き込みます。

パラメーター:

value (入力)

書き込まれるバイト配列。

length (入力)

配列のバイト数。

戻り値:

Void

例外:

- XMSEException
- MessageNotWritableException

WriteChar - 文字の書き込み

インターフェース:

```
void WriteChar(Char value);
```

文字を、上位バイトを先にして、2 バイトでメッセージ・ストリームに書き込みます。

パラメーター:

value (入力)

書き込まれる文字。

戻り値:

Void

例外:

- XMSEException
- MessageNotWritableException

WriteDouble - 倍精度浮動小数点数の書き込み

インターフェース:

```
void WriteDouble(Double value);
```

倍精度浮動小数点数を長整数に変換し、その長整数を、上位バイトを先にして、8 バイトでメッセージ・ストリームに書き込みます。

パラメーター:

value (入力)

書き込まれる倍精度浮動小数点数。

戻り値:

Void

例外:

- XMSEException
- MessageNotWritableException

WriteFloat - 浮動小数点数の書き込み

インターフェース:

```
void WriteFloat(Single value);
```

浮動小数点数を整数に変換し、その整数を、上位バイトを先にして、4 バイトでメッセージ・ストリームに書き込みます。

パラメーター:

value (入力)

書き込まれる浮動小数点数。

戻り値:

Void

例外:

- XMSEException
- MessageNotWritableException

WriteInt - 整数の書き込み

インターフェース:

```
void WriteInt(Int32 value);
```

整数を、上位バイトを先にして、4 バイトでメッセージ・ストリームに書き込みます。

パラメーター:

value (入力)

書き込まれる整数。

戻り値:

Void

例外:

- XMSEException
- MessageNotWritableException

WriteLong - 長整数の書き込み

インターフェース:

```
void WriteLong(Int64 value);
```

長整数を、上位バイトを先にして、8 バイトでメッセージ・ストリームに書き込みます。

パラメーター:

value (入力)

書き込まれる長整数。

戻り値:

Void

例外:

- XMSEException
- MessageNotWritableException

WriteObject - オブジェクトの書き込み

インターフェース:

```
void WriteObject(Object value);
```

メッセージ・ストリームに、指定されたデータ・タイプの値を書き込みます。

パラメーター:

objectType (入力)

値。以下のオブジェクト・タイプのいずれかでなければなりません。

Boolean
Byte
Byte[]
Char
Double
Single
Int32
Int64
Int16
String

value (入力)

書き込まれる値を含むバイトの配列。

length (入力)

配列のバイト数。

戻り値:

Void

例外:

- XMSEException

WriteShort - 短整数の書き込み

インターフェース:

```
void WriteShort(Int16 value);
```

短整数を、上位バイトを先にして、2 バイトでメッセージ・ストリームに書き込みます。

パラメーター:

value (入力)

書き込まれる短整数。

戻り値:

Void

例外:

- XMSEException
- MessageNotWritableException

WriteString - スtringの書き込み

インターフェース:

```
void WriteString(String value);
```

メッセージ・ストリームへStringを書き込みます。

パラメーター:

value (入力)

書き込まれるStringをカプセル化している String オブジェクト。

戻り値:

Void

例外:

- XMSException
- MessageNotWritableException

継承されたプロパティおよびメソッド

以下のプロパティは、[IMessage](#) インターフェースから継承されています。

[JMSCorrelationID](#)、[JMSDeliveryMode](#)、[JMSDestination](#)、[JMSExpiration](#)、[JMSMessageID](#)、[JMSPriority](#)、[JMSRedelivered](#)、[JMSReplyTo](#)、[JMSTimestamp](#)、[JMSType](#)、[Properties](#)

以下のメソッドは、[IMessage](#) インターフェースから継承されています。

[clearBody](#)、[clearProperties](#)、[PropertyExists](#)

以下のメソッドは、[IPropertyContext](#) インターフェースから継承されています。

[GetBooleanProperty](#)、[GetByteProperty](#)、[GetBytesProperty](#)、[GetCharProperty](#)、[GetDoubleProperty](#)、[GetFloatProperty](#)、[GetIntProperty](#)、[GetLongProperty](#)、[GetObjectProperty](#)、[GetShortProperty](#)、[GetStringProperty](#)、[SetBooleanProperty](#)、[SetByteProperty](#)、[SetBytesProperty](#)、[SetCharProperty](#)、[SetDoubleProperty](#)、[SetFloatProperty](#)、[SetIntProperty](#)、[SetLongProperty](#)、[SetObjectProperty](#)、[SetShortProperty](#)、[SetStringProperty](#)

ITextMessage

テキスト・メッセージとは、本体がストリングからなるメッセージです。

継承の階層:

```
IBM.XMS.IPropertyContext
|
+----IBM.XMS.IMessage
|
+----IBM.XMS.ITextMessage
```

.NET プロパティ

Text - テキストの取得および設定

インターフェース:

```
String Text
{
    get;
    set;
}
```

テキスト・メッセージの本文を形成するストリングを取得および設定します。

必要な場合、XMS がストリング内の文字をローカル・コード・ページに変換します。

例外:

- XMSException
- MessageNotReadableException
- MessageNotWritableException
- MessageEOFException

継承されたプロパティおよびメソッド

以下のプロパティは、[IMessage](#) インターフェースから継承されています。

[JMSCorrelationID](#)、[JMSDeliveryMode](#)、[JMSDestination](#)、[JMSExpiration](#)、[JMSMessageID](#)、[JMSPriority](#)、[JMSRedelivered](#)、[JMSReplyTo](#)、[JMSTimestamp](#)、[JMSType](#)、[Properties](#)

以下のメソッドは、[IMessage](#) インターフェースから継承されています。

[clearBody](#)、[clearProperties](#)、[PropertyExists](#)

以下のメソッドは、[IPropertyContext](#) インターフェースから継承されています。

[GetBooleanProperty](#)、[GetByteProperty](#)、[GetBytesProperty](#)、[GetCharProperty](#)、[GetDoubleProperty](#)、[GetFloatProperty](#)、[GetIntProperty](#)、[GetLongProperty](#)、[GetObjectProperty](#)、[GetShortProperty](#)、[GetStringProperty](#)、[SetBooleanProperty](#)、[SetByteProperty](#)、[SetBytesProperty](#)、[SetCharProperty](#)、[SetDoubleProperty](#)、[SetFloatProperty](#)、[SetIntProperty](#)、[SetLongProperty](#)、[SetObjectProperty](#)、[SetShortProperty](#)、[SetStringProperty](#)

TransactionInProgressException

XMS がこの例外をスローするのは、トランザクションが進行中であるために無効になっている操作をアプリケーションが要求した場合です。

継承の階層:

```
IBM.XMS.XMSEException
|
+----IBM.XMS.XMSEException
|
+----IBM.XMS.TransactionInProgressException
```

継承されたプロパティおよびメソッド

以下のメソッドは、[XMSEException](#) インターフェースから継承されています。

[GetErrorCode](#)、[GetLinkedException](#)

TransactionRolledBackException

XMS がこの例外をスローするのは、アプリケーションが現行のトランザクションをコミットするために `Session.commit()` を呼び出したにもかかわらず、その後このトランザクションがロールバックされた場合です。

継承の階層:

```
IBM.XMS.XMSEException
|
+----IBM.XMS.XMSEException
|
+----IBM.XMS.TransactionRolledBackException
```

継承されたプロパティおよびメソッド

以下のメソッドは、[XMSEException](#) インターフェースから継承されています。

[GetErrorCode](#)、[GetLinkedException](#)

XMSEException

XMS が .NET メソッドの呼び出しを処理しているときにエラーを検出すると、XMS は例外をスローします。例外とは、エラーに関する情報をカプセル化するオブジェクトのことです。

継承の階層:

```
System.Exception
|
+----IBM.XMS.XMSEException
```

XMS 例外にはさまざまなタイプがあり、XMSEException オブジェクトは例外の 1 つのタイプにすぎません。ただし、XMSEException クラスは、その他の XMS 例外クラスのスーパークラスです。XMS は、XMSEException オブジェクト以外のタイプの例外では適切でない状態では、XMSEException オブジェクトをスローします。

.NET プロパティ

ErrorCode - エラー・コードの取得

インターフェース:

```
public String ErrorCode
{
    get {return errorCode_;}
}
```

エラー・コードを取得します。

例外:

- *XMSEException*

LinkedException - リンク例外の取得

インターフェース:

```
public Exception LinkedException
{
    get { return linkedException_;}
    set { linkedException_ = value;}
}
```

例外のチェーン内の次の例外を取得します。

チェーン内に次の例外がない場合、このメソッドは **NULL** を戻します。

例外:

- *XMSEException*

XMSFactoryFactory

アプリケーションが管理対象オブジェクトを使用していない場合は、このクラスを使用して接続ファクトリー、キュー、およびトピックを作成します。

継承の階層:

なし

.NET プロパティ

Metadata - メタデータの検索

インターフェース:

```
IConnectionMetaData Metadata
```

XMSFactoryFactory オブジェクトの接続タイプに該当するメタデータを取得します。

例外:

なし

方法

CreateConnectionFactory - 接続ファクトリーの作成

インターフェース:

```
IConnectionFactory CreateConnectionFactory();
```

宣言したタイプの *ConnectionFactory* オブジェクトを作成します。

パラメーター:

なし

戻り値:

ConnectionFactory オブジェクト。

例外:

- XMSEException

CreateQueue - キューの作成

インターフェース:

```
IDestination CreateQueue(String name);
```

メッセージング・サーバー内のキューを表すための Destination オブジェクトを作成します。

このメソッドは、メッセージング・サーバー内にキューを作成しません。アプリケーションがこのメソッドを呼び出すためには、その前にキューを作成する必要があります。

パラメーター:

name (入力)

キューの名前をカプセル化している String オブジェクト、またはキューを識別する Uniform Resource Identifier (URI) をカプセル化している String オブジェクト。

戻り値:

キューを表す Destination オブジェクト。

例外:

- XMSEException

CreateTopic - トピックの作成

インターフェース:

```
IDestination CreateTopic(String name);
```

トピックを表すための Destination オブジェクトを作成します。

パラメーター:

name (入力)

トピックの名前をカプセル化している String オブジェクト、またはトピックを識別する Uniform Resource Identifier (URI) をカプセル化している String オブジェクト。

戻り値:

トピックを表す Destination オブジェクト。

例外:

- XMSEException

GetInstance - XMSFactoryFactory のインスタンスの取得

インターフェース:

```
static XMSFactoryFactory GetInstance(int connectionType);
```

XMSFactoryFactory のインスタンスを作成します。XMS アプリケーションは、XMSFactoryFactory オブジェクトを使用して、必要なタイプのプロトコルに適した ConnectionFactory オブジェクトへの参照を取得します。この結果、この ConnectionFactory オブジェクトは、対象のプロトコル・タイプに対してのみ接続経路を作成できます。

パラメーター:

connectionType (入力)

ConnectionFactory オブジェクトによって接続経路が作成される接続のタイプ

- XMSC.CT_WPM
- XMSC.CT_RTT
- XMSC.CT_WMQ

戻り値:

宣言した接続タイプ専用の XMSFactoryFactory オブジェクト。

例外:

- NotSupportedException

XMS オブジェクトのプロパティ

このセクションでは、XMS で定義したオブジェクトのプロパティについて説明します。

このセクションでは、以下のタイプのオブジェクトに関する情報を取り上げます。

- [2038 ページの『Connection のプロパティ』](#)
- [2039 ページの『ConnectionFactory のプロパティ』](#)
- [2043 ページの『ConnectionMetaData のプロパティ』](#)
- [2044 ページの『Destination のプロパティ』](#)
- [2045 ページの『InitialContext のプロパティ』](#)
- [2046 ページの『Message のプロパティ』](#)
- [2051 ページの『MessageConsumer のプロパティ』](#)
- [2051 ページの『MessageProducer のプロパティ』](#)
- [2051 ページの『Session のプロパティ』](#)

各オブジェクト・タイプの説明には、指定された型のオブジェクトのプロパティがリストで表示され、各プロパティの簡略説明が記載されています。

このセクションでは、各プロパティの定義も提供します ([2051 ページの『プロパティ定義』](#)を参照してください)。

このセクションで取り上げているオブジェクトに関する独自のプロパティをアプリケーションで定義すると、エラーは発生しませんが、予測できない結果が発生する可能性があります。

注: このセクションのプロパティ名および値は、XMSC.NAME という形式で表示されます。これは、C および C++ に使用される形式です。ただし、.NET では、プロパティ名の形式は、XMSC.NAME または XMSC_NAME のいずれかにすることができます。使用方法によっては、以下のようになります。

- プロパティを指定している場合、以下の例のように、プロパティ名は XMSC.NAME の形式でなければなりません。

```
cf.SetStringProperty(XMSC.WMQ_CHANNEL, "DOTNET.SVRCONN");
```

- スtringを指定している場合、以下の例のように、プロパティ名は XMSC_NAME の形式でなければなりません。

```
cf.SetStringProperty("XMSC_WMQ_CHANNEL", "DOTNET.SVRCONN");
```

.NET では、プロパティ名と値は XMSC クラスの定数として提供されます。これらの定数によって String が識別され、任意の XMS .NET アプリケーションで使用されます。これらの事前定義定数を使用する場合、プロパティ名と値の形式は XMSC.NAME となるため、例えば XMSC_USERID ではなく、XMSC.USERID を使用します。

データ・タイプは C/C++ に使用される形式です。.NET に対応する値は [.NET のデータ・タイプ](#)にあります。

Connection のプロパティ

Connection オブジェクトのプロパティの概要と、詳細な参照情報へのリンクを示します。

表 908. Connection のプロパティ	
プロパティ名	説明
2088 ページの『XMSC_WMQ_RESOLVED_QUEUE_MANAGER』	このプロパティは、接続先のキュー・マネージャーの名前を取得するために使用します。
2088 ページの『XMSC_WMQ_RESOLVED_QUEUE_MANAGER_ID』	接続後に、このプロパティにはキュー・マネージャーの ID が設定されます。
XMSC_WPM_CONNECTION_PROTOCOL	メッセージング・エンジンへの接続に使用される通信プロトコルです。このプロパティは読み取り専用です。
XMSC_WPM_HOST_NAME	アプリケーションの接続先となるメッセージング・エンジンが収容されているシステムのホスト名または IP アドレスです。このプロパティは読み取り専用です。
XMSC_WPM_ME_NAME	アプリケーションの接続先となるメッセージング・エンジンの名前です。このプロパティは読み取り専用です。
XMSC_WPM_PORT	アプリケーションの接続先となるメッセージング・エンジンによって listen されるポートの数です。このプロパティは読み取り専用です。

Connection オブジェクトには、接続経路を作成するときに使用した接続ファクトリーのプロパティから導出した読み取り専用プロパティもあります。これらのプロパティの導出元は、接続経路作成時に設定された接続ファクトリー・プロパティだけでなく、未設定だったプロパティのデフォルト値の場合もあります。これらのプロパティは、アプリケーションの接続先になっているメッセージング・サーバーのタイプに該当するプロパティに限定されます。プロパティの名前は、接続ファクトリー・プロパティの名前と同じです。

ConnectionFactory のプロパティ

ConnectionFactory オブジェクトのプロパティの概要と、詳細な参照情報へのリンクを示します。

表 909. ConnectionFactory のプロパティ	
プロパティ名	説明
2063 ページの『XMSC_ASYNC_EXCEPTIONS』	このプロパティは、XMS が ExceptionListener への通知を、接続が切断された時のみ行うか、または XMS API 呼び出しに対して非同期に例外が発生したときに行うかを決定します。このプロパティは、ExceptionListener が登録されているこの ConnectionFactory から作成されたすべての接続に適用されます。
2071 ページの『XMSC_WMQ_BALANCING_APPLICATION_TYPE』	バランシング・オプションのタイプ
2072 ページの『XMSC_WMQ_BALANCING_OPTIONS』	呼び出しアプリケーションによって設定されるバランシング・オプション
2072 ページの『XMSC_WMQ_BALANCING_TIMEOUT』	リバランシングがアプリケーション・アクティビティに割り込んだ後のタイムアウト。
XMSC_CLIENT_ID	接続のクライアント ID です。
XMSC_CONNECTION_TYPE	アプリケーションの接続先となるメッセージング・サーバーのタイプです。
XMSC_PASSWORD	アプリケーションがメッセージング・サーバーに接続しようとしているときに、このアプリケーションを認証するために使用するパスワードです。

表 909. <i>ConnectionFactory</i> のプロパティ (続き)	
プロパティ名	説明
2068 ページの 『XMSC_RTT_BROKER_PING_INTERVAL』	XMS.NET がリアルタイム・メッセージング・サーバーへの接続を検査することでアクティビティの検出を行うまでの時間間隔 (ミリ秒単位)。
XMSC_RTT_CONNECTION_PROTOCOL	ブローカーへのリアルタイム接続に使用される通信プロトコルです。
XMSC_RTT_HOST_NAME	ブローカーを実行するシステムのホスト名または IP アドレスです。
XMSC_RTT_LOCAL_ADDRESS	ブローカーへのリアルタイム接続に使用するローカル・ネットワーク・インターフェースのホスト名または IP アドレスです。
XMSC_RTT_MULTICAST	接続ファクトリーまたは宛先のマルチキャスト設定です。
XMSC_RTT_PORT	ブローカーが着信要求を <code>listen</code> するポートの数です。
XMSC_USERID	アプリケーションがメッセージング・サーバーに接続しようとしているときに、このアプリケーションを認証するために使用するユーザー ID です。
XMSC_WMQ_BROKER_CONTROLQ	ブローカーが使用する制御キューの名前。
XMSC_WMQ_BROKER_PUBQ	ブローカーによってモニターされているキューの名前で、このキューでは、アプリケーションが発行したメッセージをアプリケーション自体が送信します。
XMSC_WMQ_BROKER_QMGR	ブローカーの接続先となるキュー・マネージャーの名前です。
XMSC_WMQ_BROKER_SUBQ	非永続メッセージ・コンシューマー用のサブスクライバー・キューの名前です。
XMSC_WMQ_BROKER_VERSION	接続または宛先に合わせてアプリケーションが使用するブローカーのタイプです。
2074 ページの 『XMSC_WMQ_CCDTURL』	クライアント・チャネル定義テーブルを含むファイルの名前および場所を識別すると同時に、そのファイルへのアクセス方法も指定する、URL (Uniform Resource Locator)。
XMSC_WMQ_CHANNEL	接続に使用するチャネルの名前です。
2075 ページの 『XMSC_WMQ_CLIENT_RECONNECT_OPTIONS』	このプロパティでは、このファクトリーで作成する新しい接続のクライアント再接続オプションを指定します。
2076 ページの 『XMSC_WMQ_CLIENT_RECONNECT_TIMEOUT』	このプロパティでは、クライアント接続が再接続を試みる期間を秒単位で指定します。
XMSC_WMQ_CONNECTION_MODE	アプリケーションがキュー・マネージャーに接続する場合のモードです。
2076 ページの 『XMSC_WMQ_CONNECTION_NAME_LIST』	このプロパティでは、接続で障害が発生した後にクライアントが再接続を試みる対象のホストを指定します。
XMSC_WMQ_FAIL_IF QUIESCE	アプリケーションの接続先のキュー・マネージャーが静止状態である場合、特定のメソッドの呼び出しが失敗するかどうかを表します。
XMSC_WMQ_HOST_NAME	キュー・マネージャーを実行するシステムのホスト名または IP アドレスです。

表 909. <i>ConnectionFactory</i> のプロパティ (続き)	
プロパティ名	説明
XMSC_WMQ_LOCAL_ADDRESS	キュー・マネージャーへの接続の場合、このプロパティは、使用するローカル・ネットワーク・インターフェース、または使用するローカル・ポート (1 つまたは一定範囲)、あるいはその両方を指定します。
XMSC_WMQ_MESSAGE_SELECTION	メッセージの選択をクライアント XMS が行うか、ブローカーが行うかを決定します。
XMSC_WMQ_MSG_BATCH_SIZE	非同期のメッセージ配信機能を使用する場合に 1 バッチ内のキューから取り出すメッセージの最大数を表します。
XMSC_WMQ_POLLING_INTERVAL	セッション内の各メッセージ・リスナーのキューに適切なメッセージがない場合、この値は、各メッセージ・リスナーがそのキューから再度メッセージを読み取ろうとするまでに経過する最大の時間間隔 (ミリ秒) になります。
2085 ページの 『XMSC_WMQ_PROVIDER_VERSION』	アプリケーションの接続先のキュー・マネージャーのバージョン、リリース、モディフィケーション・レベル、およびフィックスパック。
XMSC_WMQ_PORT	キュー・マネージャーが着信要求を listen するポートの数です。
XMSC_WMQ_PUB_ACK_INTERVAL	クライアントが XMS ブローカーに確認応答を要求する前に、パブリッシャーが発行するメッセージの数。
2081 ページの 『XMSC_WMQ_PUT_ASYNC_ALLOWED』	このプロパティは、メッセージ・プロデューサーが、非同期書き込みを使用して、この宛先にメッセージを送信できるかどうかを決定します。
XMSC_WMQ_QMGR_CCSID	メッセージ・キュー・インターフェース (MQI) で定義された文字データフィールドが、クライアント IBM MQXMS 間で交換される際に使用される、符号化文字セットまたはコードページの識別子 (CCSID)。
XMSC_WMQ_QUEUE_MANAGER	接続先となるキュー・マネージャーの名前です。
XMSC_WMQ_RECEIVE_EXIT	実行するチャネル受信出口を示します。
XMSC_WMQ_RECEIVE_EXIT_INIT	チャネル受信出口が呼び出されたときにこの出口に渡されるユーザー・データです。
XMSC_WMQ_SECURITY_EXIT	チャネル・セキュリティ出口を示します。
XMSC_WMQ_SECURITY_EXIT_INIT	チャネル・セキュリティ出口を呼び出した場合にこの出口に渡されるユーザー・データです。
2089 ページの 『XMSC_WMQ_SEND_CHECK_COUNT』	単一の未処理の XMS セッション内での、非同期書き込みエラーの検査の間に許可する Send 呼び出しの数。
XMSC_WMQ_SEND_EXIT	チャネル送信出口を示します。
XMSC_WMQ_SEND_EXIT_INIT	複数のチャネル送信出口を呼び出した場合にこれらの出口に渡されるユーザー・データです。
2089 ページの 『XMSC_WMQ_SHARE_CONV_ALLOWED』	チャネル定義が一致する場合、クライアント接続が、同じプロセスから同じキューマネージャーへの他のトップ XMS レベル接続とソケットを共有できるかどうか。このプロパティは、アプリケーション開発、保守、または操作に必要な場合に、別のソケットの接続を完全に分離できるようにするために提供されています。

表 909. <i>ConnectionFactory</i> のプロパティ (続き)	
プロパティ名	説明
XMSC_WMQ_SSL_CERT_STORES	キュー・マネージャーとの SSL 接続で使用する証明書取り消しリスト (CRL) を保持するサーバーの位置。
XMSC_WMQ_SSL_CIPHER_SPEC	キュー・マネージャーとのセキュア接続で使用する CipherSpec の名前。
XMSC_WMQ_SSL_CIPHER_SUITE	キュー・マネージャーとの TLS 接続で使用する CipherSuite の名前。セキュア接続のネゴシエーションで使用するプロトコルは、指定されている CipherSuite によって異なります。
XMSC_WMQ_SSL_CRYPTO_HW	クライアント・システムに接続されている暗号ハードウェアに関する構成詳細情報。
XMSC_WMQ_SSL_FIPS_REQUIRED	このプロパティの値によって、アプリケーションが非準拠の FIPS 暗号スイートを使用できるかどうかが決まります。このプロパティが true に設定されている場合、クライアントとサーバー間の接続にはアルゴリズムのみ FIPS が使用されます。
v 10.0.0 XMSC_WMQ_SSL_キーリポジトリ 1	鍵および証明書が保存されている鍵データベース・ファイルの位置。
XMSC_WMQ_SSL_KEY_RESETCOUNT	KeyResetCount は、秘密鍵の再ネゴシエーションが実行されるまで、1 つの SSL 会話の中で送受信される暗号化されていないデータの合計バイト数を表します。
XMSC_WMQ_SSL_PEER_NAME	キュー・マネージャーとの SSL 接続で使用するピア名。
XMSC_WMQ_SYNCPOINT_ALL_GETS	すべてのメッセージを同期点制御の対象範囲内のキューから取り出す必要があるかどうかを示します。
2096 ページの『XMSC_WMQ_TARGET_CLIENT』	
XMSC_WMQ_TEMP_Q_PREFIX	アプリケーションが一時キュー XMS を作成する際に生成される動的キュー IBM MQ の名前を構成するために使用される接頭辞。
XMSC_WMQ_TEMP_TOPIC_PREFIX	一時トピックを作成する際、XMS "TEMP/TEMPTOPICPREFIX/unique_id" という形式のトピック文字列が生成されます。ただし、このプロパティにデフォルト値が設定されている場合は、その文字列が TEMP/unique_id 生成されます。空以外の値を指定すると、この接続で作成された一時トピックへのサブスクライバーの管理対象キューを作成するために、特定のモデル・キューを定義できます。
XMSC_WMQ_TEMPORARY_MODEL	アプリケーションが一時キュー XMS を作成する際に、その一時キューの基となるモデルキュー IBM MQ の名前。
XMSC_WPM_BUS_NAME	接続ファクトリーの場合は、アプリケーションの接続先となるサービス統合バスの名前であり、宛先の場合は、その宛先が存在するサービス統合バスの名前です。
XMSC_WPM_CONNECTION_PROXIMITY	接続に対する接続接近性の設定です。
XMSC_WPM_DUR_SUB_HOME	ある接続または宛先に対するすべての永続サブスクリプションが管理対象になっているメッセージング・エンジンの名前です。

表 909. <i>ConnectionFactory</i> のプロパティ (続き)	
プロパティ名	説明
XMSC_WPM_LOCAL_ADDRESS	サービス統合バスへの接続の場合、このプロパティは、使用するローカル・ネットワーク・インターフェース、または使用するローカル・ポート (1 つまたは一定範囲)、あるいはその両方を指定します。
XMSC_WPM_NON_PERSISTENT_MAP	接続を使用して送信される非永続メッセージの信頼性レベルです。
XMSC_WPM_PERSISTENT_MAP	接続を使用して送信される永続メッセージの信頼性レベルです。
XMSC_WPM_PROVIDER_ENDPOINTS	ブートストラップ・サーバーの 1 つ以上のエンドポイント・アドレスの列です。
XMSC_WPM_TARGET_GROUP	メッセージング・エンジンのターゲット・グループの名前です。
XMSC_WPM_TARGET_SIGNIFICANCE	メッセージング・エンジンのターゲット・グループの重要度です。
XMSC_WPM_TARGET_TRANSPORT_CHAIN	アプリケーションがメッセージング・エンジンに接続するために使用する必要があるインバウンド・トランスポート・チェーンの名前です。
XMSC_WPM_TARGET_TYPE	メッセージング・エンジンのターゲット・グループのタイプです。
XMSC_WPM_TEMP_Q_PREFIX	アプリケーションが一時キュー XMS を作成する際に、サービス統合バス内に作成される一時キューの名前を構成するために使用される接頭辞。
XMSC_WPM_TEMP_TOPIC_PREFIX	アプリケーションによって作成される一時トピックの名前を構成する場合に使用されるプレフィックスです。
V 10.0.0 X MSC_WMQ_SSL_キーリポジトリパスワード 1	ハンドシェイクの前に TLS 証明書を開くために必要なパスワードをクライアントが設定するためのプロパティ名。
V 10.0.0 X MSC_WMQ_暗号化キー 1	クライアントが暗号化キーファイルの保存場所を設定するためのプロパティ名。

注: V 10.0.0 から IBM MQ 9.4.4、XMS.NET アプリケーションはカスタムパスを指定することで、TLS 接続用に個別の証明書パスを使用できます。Dotnet Password Encrypt ユーティリティを使用して暗号化されたパスワードを提供することも、プレーンテキストのパスワードを提供することもできます。詳細については、XMS の「[証明書のカスタムパス](#)」を参照してください。.NETTLS 接続におけるプレーンテキストまたは暗号化されたパスワードの使用について。

ConnectionMetaData のプロパティ

ConnectionMetaData オブジェクトのプロパティの概要と、詳細な参照情報へのリンクを示します。

表 910. <i>ConnectionMetaData</i> のプロパティ	
プロパティ名	説明
XMSC_JMS_MAJOR_VERSION	が XMS 基づいている仕 JMS 様のメジャーバージョン番号。このプロパティは読み取り専用です。
XMSC_JMS_MINOR_VERSION	が XMS 基づいている仕 JMS 様のマイナーバージョン番号。このプロパティは読み取り専用です。

表 910. *ConnectionMetaData* のプロパティ (続き)

プロパティ名	説明
XMSC_JMS_VERSION	が XMS 基づく仕 JMS 様のバージョン識別子。このプロパティは読み取り専用です。
XMSC_MAJOR_VERSION	XMS クライアントのバージョン番号。このプロパティは読み取り専用です。
XMSC_MINOR_VERSION	XMS クライアントのリリース番号。このプロパティは読み取り専用です。
XMSC_PROVIDER_NAME	XMS クライアントの提供元。このプロパティは読み取り専用です。
XMSC_VERSION	XMS クライアントのバージョン識別子。このプロパティは読み取り専用です。

Destination のプロパティ

Destination オブジェクトのプロパティの概要と、詳細な参照情報へのリンクを示します。

表 911. *Destination* のプロパティ

プロパティ名	説明
XMSC_DELIVERY_MODE	宛先に送信されたメッセージの送達モードです。
XMSC_PRIORITY	宛先に送信されたメッセージの優先順位です。
XMSC_RTT_MULTICAST	接続ファクトリーまたは宛先のマルチキャスト設定です。
XMSC_TIME_TO_LIVE	宛先に送信されたメッセージの存続時間です。
XMSC_WMQ_BROKER_VERSION	接続または宛先に合わせてアプリケーションが使用するブローカーのタイプです。
XMSC_WMQ_CCSID	クライアントが XMS メッセージを宛先に転送する際、メッセージ本文内の文字データ文字列が属する符号化文字セット (CCSID) またはコードページ。
XMSC_WMQ_DUR_SUBQ	宛先からメッセージを受信している永続サブスクライバー用のサブスクライバー・キューの名前です。 注: このプロパティは、IBM Message Service Client for .NET バージョン 2.0 で使用できますが、接続ファクトリーの XMSC_WMQ_PROVIDER_VERSION プロパティが 7 より前のバージョン番号に設定されている場合を除き、IBM WebSphere MQ 7.0 キュー・マネージャーに接続されているアプリケーションでは無効になります。
XMSC_WMQ_ENCODING	クライアントが XMS メッセージを宛先に転送する際、メッセージ本文内の数値データがどのように表現されるか。
XMSC_WMQ_FAIL_IF_QUIESCE	アプリケーションの接続先のキュー・マネージャーが静止状態である場合、特定のメソッドの呼び出しが失敗するかどうかを表します。
2079 ページの『XMSC_WMQ_MESSAGE_BODY』	このプロパティは、アプリケーションが XMS メッセージ IBM MQ の MQRFH2 をメッセージのペイロードの一部 (つまり、メッセージ本文の一部) として処理するかどうかを決定します。

表 911. Destination のプロパティ (続き)	
プロパティ名	説明
2079 ページの 『XMSC_WMQ_MQMD_MESSAGE_CONTEXT』	アプリケーション XMS によって設定されるメッセージコンテキストのレベルを決定します。このプロパティが有効となるためには、アプリケーションが適切なコンテキスト権限を持って実行されていなければなりません。
2080 ページの 『XMSC_WMQ_MQMD_READ_ENABLED』	このプロパティは、アプリケーションが XMS MQMD フィールドの値を抽出できるかどうかを決定します。
2080 ページの 『XMSC_WMQ_MQMD_WRITE_ENABLED』	このプロパティは、アプリケーションが XMS MQMD フィールドの値を設定できるかどうかを決定します。
2081 ページの 『XMSC_WMQ_READ_AHEAD_ALLOWED』	このプロパティは、メッセージ・コンシューマーとキュー・ブラウザーが、先行読み取りを使用して、非永続の非トランザクション・メッセージを受信する前に、この宛先から内部バッファにこれらのメッセージを取得できるかどうかを決定します。
2082 ページの 『XMSC_WMQ_READ_AHEAD_CLOSE_POLICY』	このプロパティは、非同期メッセージ・リスナーに配信されているメッセージについて、メッセージ・コンシューマーがクローズされたときに、内部先行読み取りバッファ内のメッセージがどうなるかを決定します。
2087 ページの 『XMSC_WMQ_RECEIVE_CCSID』	キュー・マネージャー・メッセージ変換のターゲット CCSID を設定する宛先プロパティ。 XMSC_WMQ_RECEIVE_CONVERSION が WMQ_RECEIVE_CONVERSION_QMGR に設定されていなければ、この値は無視されます。
2087 ページの 『XMSC_WMQ_RECEIVE_CONVERSION』	キュー・マネージャーによりデータ変換を実行するかどうかを決定する宛先プロパティ。
XMSC_WMQ_TARGET_CLIENT	宛先に送信されるメッセージに MQRFH2 ヘッダーを付けるかどうかを示します。
XMSC_WMQ_TEMP_TOPIC_PREFIX	一時トピックを作成する際、XMS "TEMP/ TEMPTOPICPREFIX/unique_id" という形式のトピック文字列が生成されます。ただし、このプロパティにデフォルト値が設定されている場合は、その文字列が TEMP/ unique_id 生成されます。空以外の値を指定すると、この接続で作成された一時トピックへのサブスクライバーの管理対象キューを作成するために、特定のモデル・キューを定義できます。
XMSC_WPM_BUS_NAME	接続ファクトリーの場合は、アプリケーションの接続先となるサービス統合バスの名前であり、宛先の場合は、その宛先が存在するサービス統合バスの名前です。
XMSC_WPM_TOPIC_SPACE	トピックが収容されているトピック・スペースの名前です。

InitialContext のプロパティ

InitialContext オブジェクトのプロパティの概要と、詳細な参照情報へのリンクを示します。

表 912. <i>InitialContext</i> のプロパティ	
プロパティ名	説明
<u>XMSC_IC_PROVIDER_URL</u>	JNDI ネーミング・ディレクトリーの位置を指定するために使用します。その位置を指定すれば、COS ネーミング・サービスが Web サービスと同じサーバーに存在する必要はなくなります。
<u>XMSC_IC_SECURITY_AUTHENTICATION</u>	Java Context インターフェース <code>SECURITY_AUTHENTICATION</code> に基づいています。このプロパティは COS ネーミング・コンテキストにのみ適用されます。
<u>XMSC_IC_SECURITY_CREDENTIALS</u>	Context インターフェース <code>Java SECURITY_CREDENTIALS</code> に基づいています。このプロパティは COS ネーミング・コンテキストにのみ適用されます。
<u>XMSC_IC_SECURITY_PRINCIPAL</u>	Java Context インターフェース <code>SECURITY_PRINCIPAL</code> に基づいています。このプロパティは COS ネーミング・コンテキストにのみ適用されます。
<u>XMSC_IC_SECURITY_PROTOCOL</u>	Java Context インターフェースに基づく <code>SECURITY_PROTOCOL</code> このプロパティは COS ネーミング・コンテキストにのみ適用されます。
<u>XMSC_IC_URL</u>	LDAP コンテキストや <code>FileSystem</code> コンテキストの場合は、管理対象オブジェクトを収容しているリポジトリのアドレスです。COS ネーミング・コンテキストの場合は、ディレクトリー内のオブジェクトを検索する Web サービスのアドレスです。

Message のプロパティ

Message オブジェクトのプロパティの概要と、詳細な参照情報へのリンクを示します。

表 913. <i>Message</i> のプロパティ	
プロパティ名	説明
<u>JMS_IBM_CHARACTER_SET</u>	クライアントが XMS メッセージを宛先へ転送する際、メッセージ本文内の文字データ列が属する符号化文字セット（またはコードページ）の識別子（CCSID）。XMS では、このプロパティには数値が指定され、CCSID にマップされます。ただし、このプロパティは JMS プロパティに基づいているため、ストリング型の値を持ち、この数値 CCSID を表す Java 文字セットにマップされます。
<u>JMS_IBM_ENCODING</u>	クライアントが XMS メッセージを宛先へ転送する際、メッセージ本文内の数値データがどのように表現されるか。
<u>JMS_IBM_EXCEPTIONMESSAGE</u>	メッセージが例外の宛先に送信された理由を説明するテキストです。このプロパティは読み取り専用です。
<u>JMS_IBM_EXCEPTIONPROBLEMDESTINATION</u>	メッセージが例外の宛先に送信される前に、そのメッセージが存在した宛先の名前です。
<u>JMS_IBM_EXCEPTIONREASON</u>	メッセージが例外の宛先に送信された理由を示す理由コードです。
<u>JMS_IBM_EXCEPTIONTIMESTAMP</u>	メッセージが例外の宛先に送信された時刻です。
<u>JMS_IBM_FEEDBACK</u>	レポート・メッセージの種類を示すコードです。

表 913. Message のプロパティ (続き)	
プロパティ名	説明
JMS_IBM_FORMAT	メッセージ内にあるアプリケーション・データの種類です。
JMS_IBM_LAST_MSG_IN_GROUP	メッセージがメッセージ・グループ内の最後のメッセージであるかどうかを示します。
JMS_IBM_MSGTYPE	メッセージのタイプ。
JMS_IBM_PUTAPPLTYPE	メッセージを送信したアプリケーションのタイプです。
JMS_IBM_PUTDATE	メッセージが送信された日付です。
JMS_IBM_PUTTIME	メッセージが送信された時刻です。
JMS_IBM_REPORT_COA	「到着時の確認」レポート・メッセージを要求し、元のメッセージからのアプリケーション・データをレポート・メッセージにどの程度組み込む必要があるかを指定します。
JMS_IBM_REPORT_COD	「配信時の確認」レポート・メッセージを要求し、元のメッセージからのアプリケーション・データをレポート・メッセージにどの程度組み込む必要があるかを指定します。
JMS_IBM_REPORT_DISCARD_MSG	メッセージを目的の宛先に配信できない場合に、そのメッセージを廃棄することを要求します。
JMS_IBM_REPORT_EXCEPTION	例外レポート・メッセージを要求し、元のメッセージからのアプリケーション・データをレポート・メッセージにどの程度組み込む必要があるかを指定します。
JMS_IBM_REPORT_EXPIRATION	期限満了レポート・メッセージを要求し、元のメッセージからのアプリケーション・データをレポート・メッセージにどの程度組み込む必要があるかを指定します。
JMS_IBM_REPORT_NAN	否定アクション通知レポート・メッセージを要求します。
JMS_IBM_REPORT_PAN	肯定アクション通知レポート・メッセージを要求します。
JMS_IBM_REPORT_PASS_CORREL_ID	任意のレポート・メッセージまたは応答メッセージの相関 ID を元のメッセージの相関 ID と同じにするという要求です。
JMS_IBM_REPORT_PASS_MSG_ID	任意のレポート・メッセージまたは応答メッセージのメッセージ ID を元のメッセージのメッセージ ID と同じにするという要求です。
JMS_IBM_RETAIN	このプロパティを設定すると、キュー・マネージャーは、メッセージを保存パブリケーションとして扱うように指定されます。
JMS_IBM_SYSTEM_MESSAGEID	サービス統合バスの内部でメッセージを一意的に識別する ID です。このプロパティは読み取り専用です。
JMSX_APPID	メッセージを送信したアプリケーションの名前です。
JMSX_DELIVERY_COUNT	メッセージ配信の試行回数です。
JMSX_GROUPID	メッセージが属するメッセージ・グループの ID です。
JMSX_GROUPSEQ	メッセージ・グループ内にあるメッセージのシーケンス番号です。
JMSX_USERID	メッセージを送信したアプリケーションに関連付けられているユーザー ID です。

JMS_IBM_MQMD* プロパティ

IBM Message Service Client for .NET では、API を使用して、クライアント・アプリケーションによる MQMD フィールドの読み取り/書き込みが可能です。また、MQ メッセージ・データにアクセスすることもできます。デフォルトでは、MQMD へのアクセスは無効になっており、Destination のプロパティ `XMSC_WMQ_MQMD_WRITE_ENABLED` と `XMSC_WMQ_MQMD_READ_ENABLED` を使用してアプリケーションで明示的に有効にする必要があります。これらの 2 つのプロパティは互いに独立しています。

StrucId と Version を除く MQMD フィールドはすべて、追加の Message オブジェクト・プロパティとして公開され、JMS_IBM_MQMD というプレフィックスが付けられます。

JMS_IBM_MQMD* プロパティは、上の表で取り上げられている他のプロパティ (JMS_IBM* など) より優先されます。

メッセージの送信

StrucId と Version を除くすべての MQMD フィールドが表されます。これらのプロパティは MQMD フィールドのみを参照しています。このフィールドでは、MQMD ヘッダーと MQRFH2 ヘッダーの両方でプロパティが発生し、MQRFH2 のバージョンは設定も抽出もされません。JMS_IBM_MQMD_BackoutCount を除き、これらのすべてのプロパティを設定できます。JMS_IBM_MQMD_BackoutCount に設定された値はすべて無視されます。

プロパティが最大長を持っていて、長過ぎる値が提供された場合、その値は切り捨てられます。

特定のプロパティでは、Destination オブジェクトで `XMSC_WMQ_MQMD_MESSAGE_CONTEXT` プロパティも設定する必要があります。このプロパティが有効となるためには、アプリケーションが適切なコンテキスト権限を持って実行されていなければなりません。`XMSC_WMQ_MQMD_MESSAGE_CONTEXT` を適切な値に設定しないと、プロパティ値は無視されます。`XMSC_WMQ_MQMD_MESSAGE_CONTEXT` を適切な値に設定しても、キュー・マネージャーに対して十分なコンテキスト権限がない場合は、例外が発行されます。`XMSC_WMQ_MQMD_MESSAGE_CONTEXT` の特定の値が必要なプロパティは以下のとおりです。

以下のプロパティでは、`XMSC_WMQ_MQMD_MESSAGE_CONTEXT` を `XMSC_WMQ_MDCTX_SET_IDENTITY_CONTEXT` または `XMSC_WMQ_MDCTX_SET_ALL_CONTEXT` に設定する必要があります。

- JMS_IBM_MQMD_UserIdentifier
- JMS_IBM_MQMD_AccountingToken
- JMS_IBM_MQMD_ApplIdentityData

以下のプロパティでは、`XMSC_WMQ_MQMD_MESSAGE_CONTEXT` を `XMSC_WMQ_MDCTX_SET_ALL_CONTEXT` に設定する必要があります。

- JMS_IBM_MQMD_PutApplType
- JMS_IBM_MQMD_PutApplName
- JMS_IBM_MQMD_PutDate
- JMS_IBM_MQMD_PutTime
- JMS_IBM_MQMD_ApplOriginData

メッセージの受信

`XMSC_WMQ_MQMD_READ_ENABLED` プロパティが `true` に設定されていれば、メッセージを作成するアプリケーションで設定されている実際のプロパティに関係なく、受信するメッセージでこれらのプロパティがすべて有効になります。JMS 仕様によれば、最初にプロパティをすべてクリアしない限り、アプリケーションでは、受信したメッセージのプロパティを変更できません。プロパティを変更せずに、受信メッセージを転送することができます。

注：アプリケーションが、`XMSC_WMQ_MQMD_READ_ENABLED` プロパティが `true` に設定された宛先からメッセージを受信して、`XMSC_WMQ_MQMD_WRITE_ENABLED` が `true` に設定された宛先にそのメッセ

ージを転送すると、受信したメッセージの MQMD フィールド値はすべて、転送メッセージにコピーされます。プロパティの表

表 914. MQMD フィールドを表す Message オブジェクトのプロパティ		
Property	説明	タイプ
JMS_IBM_MQMD_REPORT	レポート・メッセージのオプション	System.Int32
JMS_IBM_MQMD_MSGTYPE	メッセージ・タイプ	System.Int32
JMS_IBM_MQMD_EXPIRY	メッセージの存続時間	System.Int32
JMS_IBM_MQMD_FEEDBACK	フィードバックまたは理由コード	System.Int32
JMS_IBM_MQMD_ENCODING	メッセージ・データの数値エンコード	System.Int32
JMS_IBM_MQMD_CODEDCHARSETID	メッセージ・データの文字セット ID	System.Int32
JMS_IBM_MQMD_FORMAT	メッセージ・データの形式名。	System.String
JMS_IBM_MQMD_PRIORITY 注: 0 から 9 までの範囲にない値を JMS_IBM_MQMD_PRIORITY に割り当てると、JMS 仕様違反になります。	メッセージ優先順位	System.Int32
JMS_IBM_MQMD_PERSISTENCE	メッセージの持続性	System.Int32
JMS_IBM_MQMD_MSGID 注: JMS 仕様には、メッセージ ID は JMS プロバイダーによって設定されている必要があり、かつ固有であるかヌルである必要があると記されています。 JMS_IBM_MQMD_MSGID に値を割り当てると、この値は JMSMessageID にコピーされます。つまり、値が JMS プロバイダーによって設定されず、固有の値でなくなる可能性があります。その場合は、JMS 仕様違反になります。	メッセージ ID	バイト配列 注: メッセージでバイト配列プロパティを使用すると、JMS 仕様に違反します。
JMS_IBM_MQMD_CORRELID 注: スtring「ID:」で始まる JMS_IBM_MQMD_CORRELID に割り当てると、JMS 仕様違反になります。	相関 ID	バイト配列 注: メッセージでバイト配列プロパティを使用すると、JMS 仕様に違反します。
JMS_IBM_MQMD_BACKOUTCOUNT	バックアウトのカウンター	System.Int32
JMS_IBM_MQMD_REPLYTOQ	応答キューの名前	System.String
JMS_IBM_MQMD_REPLYTOQMGR	応答キュー・マネージャーの名前	System.String
JMS_IBM_MQMD_USERIDENTIFIER	ユーザー ID	System.String
JMS_IBM_MQMD_ACCOUNTINGTOKEN	アカウントिंग・トークン	バイト配列 注: メッセージでバイト配列プロパティを使用すると、JMS 仕様に違反します。
JMS_IBM_MQMD_APPLIDENTITYDATA	ID に関連するアプリケーション・データ	System.String

表 914. MQMD フィールドを表す Message オブジェクトのプロパティ (続き)		
Property	説明	タイプ
JMS_IBM_MQMD_PUTAPPLTYPE	メッセージを書き込んだアプリケーションのタイプ	System.Int32
JMS_IBM_MQMD_PUTAPPLNAME	メッセージを書き込むアプリケーションの名前	System.String
JMS_IBM_MQMD_PUTDATE	メッセージを書き込んだ日付	System.String
JMS_IBM_MQMD_PUTTIME	メッセージを書き込んだ時刻	System.String
JMS_IBM_MQMD_APPLORIGINDATA	発生元に関係するアプリケーション・データ	System.String
JMS_IBM_MQMD_GROUPID	グループ ID	バイト配列 注: メッセージでバイト配列プロパティを使用すると、JMS 仕様に違反します。
JMS_IBM_MQMD_MSGSEQNUMBER	グループ内のローカル・メッセージのシーケンス番号	System.Int32
JMS_IBM_MQMD_OFFSET	論理メッセージの先頭を起点とする、物理メッセージ中のデータのオフセット	System.Int32
JMS_IBM_MQMD_MSGFLAGS	メッセージ・フラグ	System.Int32
JMS_IBM_MQMD_ORIGINALLENGTH	元のメッセージの長さ	System.Int32

詳しくは、[MQMD](#) を参照してください。

例

この例では、MQMD.UserIdentifer が「JoeBloggs」に設定されたキューまたはトピックにメッセージが書き込まれます。

```
// Create a ConnectionFactory, connection, session, producer, message
// ...

// Create a destination
// ...

// Enable MQMD write
dest.setBooleanProperty(XMSC_WMQ_MQMD_WRITE_ENABLED,
    XMSC_WMQ_MQMD_WRITE_ENABLED_YES);

// Optionally, set a message context if applicable for this MD field
dest.setIntProperty(XMSC_WMQ_MQMD_MESSAGE_CONTEXT,
    XMSC_WMQ_MDCTX_SET_IDENTITY_CONTEXT);

// On the message, set property to provide custom UserId
msg.setStringProperty(JMS_IBM_MQMD_USERIDENTIFIER, "JoeBloggs");

// Send the message
// ...
```

JMS_IBM_MQMD_USERIDENTIFIER を設定する前に、XMSC_WMQ_MQMD_MESSAGE_CONTEXT を設定する必要があります。XMSC_WMQ_MQMD_MESSAGE_CONTEXT の使用の詳細については、Message オブジェクト・プロパティを参照してください。

同様に、メッセージを受信する前に XMSC_WMQ_MQMD_READ_ENABLED を true に設定してから、メッセージの get メソッド (getStringProperty など) を使用することによって、MQMD フィールドの内容を抽出できます。受信するプロパティはすべて読み取り専用です。

この例では、メッセージの MQMD.ApplIdentityData フィールドの値を保持する値フィールドがキューまたはトピックから取得されます。

```
// Create a ConnectionFactory, connection, session, consumer
// ...

// Create a destination
// ...

// Enable MQMD read
dest.setBooleanProperty(XMSC_WMQ_MQMD_READ_ENABLED, XMSC_WMQ_MQMD_READ_ENABLED_YES);

// Receive a message
// ...

// Get required MQMD field value using a property
System.String value = rcvMsg.getStringProperty(JMS_IBM_MQMD_APPLIDENTITYDATA);
```

MessageConsumer のプロパティ

MessageConsumer オブジェクトのプロパティの概要と、詳細な参照情報へのリンクを示します。

表 915. MessageConsumer のプロパティ	
プロパティ名	説明
XMSC_IS_SUBSCRIPTION_MULTICAST	WebSphere MQ Multicast Transport メッセージがメッセージコンシューマーに配信されているかどうかを示します。このプロパティは読み取り専用です。
XMSC_IS_SUBSCRIPTION_RELIABLE_MULTICAST	メッセージが、信頼性の高いサービス品質 (QoS) を用いて WebSphere MQ Multicast Transport メッセージコンシューマーに配信されているかどうかを示します。このプロパティは読み取り専用です。

詳しくは、[『IMessageConsumer』の『.NET プロパティ』](#)を参照してください。

MessageProducer のプロパティ

MessageProducer オブジェクトのプロパティの概要と、詳細な参照情報へのリンクを示します。

詳しくは、[IMessageProducer の .NET プロパティ](#) を参照してください。

Session のプロパティ

Session オブジェクトのプロパティの概要と、詳細な参照情報へのリンクを示します。

詳しくは、[ISession の .NET プロパティ](#) を参照してください。

プロパティ定義

このセクションでは、各オブジェクト・プロパティの定義について説明します。

各プロパティ定義には、以下の情報が含まれます。

- プロパティのデータ・タイプ
- プロパティを持つオブジェクトの型
- Destination のプロパティの場合は、Uniform Resource Identifier (URI) で使用できる名前
- プロパティの詳細な説明
- プロパティの有効値
- プロパティのデフォルト値

名前が以下のいずれかのプレフィックスで始まるプロパティは、指定されたタイプの接続にのみ関連します。

XMSC_RTT

このプロパティは、ブローカーへのリアルタイム接続にのみ関連します。プロパティの名前は、`xmssc_rtt.h` ヘッダー・ファイル内で名前付き定数として定義されます。

XMSC_WMQ

このプロパティは、アプリケーションが IBM MQ キュー・マネージャーに接続している場合にのみ関連します。プロパティの名前は、`xmssc_wmq.h` ヘッダー・ファイル内で名前付き定数として定義されます。

XMSC_WPM

このプロパティを使用するのは、アプリケーションが WebSphere サービス統合バスに接続する場合に限られます。プロパティの名前は、`xmssc_wpm.h` ヘッダー・ファイル内で名前付き定数として定義されます。

プロパティの定義に特に明記されていない限り、残りのプロパティは、すべてのタイプの接続に関連します。プロパティの名前は、`xmssc.h` ヘッダー・ファイル内で名前付き定数として定義されます。名前が `JMSX` というプレフィックスで始まるプロパティは、メッセージの JMS 定義のプロパティであり、名前が `JMS_IBM` というプレフィックスで始まるプロパティは、メッセージの IBM 定義のプロパティです。メッセージのプロパティの詳細については、[XMS メッセージのプロパティ](#)を参照してください。

プロパティの定義に特に明記されていない限り、各プロパティは Point-to-Point とパブリッシュ・サブスクライブの2つのドメインに関連します。

プロパティが読み取り専用と指定されていない限り、アプリケーションは任意のプロパティの値を取得して設定できます。

以下のプロパティが定義されています。

[2054 ページの『JMS IBM CHARACTER_SET』](#)
[2054 ページの『JMS IBM ENCODING』](#)
[2055 ページの『JMS IBM EXCEPTIONMESSAGE』](#)
[2055 ページの『JMS IBM EXCEPTIONPROBLEMDESTINATION』](#)
[2055 ページの『JMS IBM EXCEPTIONREASON』](#)
[2056 ページの『JMS IBM EXCEPTIONTIMESTAMP』](#)
[2056 ページの『JMS IBM FEEDBACK』](#)
[2056 ページの『JMS IBM FORMAT』](#)
[2056 ページの『JMS IBM LAST_MSG_IN_GROUP』](#)
[2057 ページの『JMS IBM MSGTYPE』](#)
[2057 ページの『JMS IBM PUTAPPLTYPE』](#)
[2057 ページの『JMS IBM PUTDATE』](#)
[2057 ページの『JMS IBM PUTTIME』](#)
[2058 ページの『JMS IBM REPORT_COA』](#)
[2058 ページの『JMS IBM REPORT_COD』](#)
[2059 ページの『JMS IBM REPORT DISCARD MSG』](#)
[2059 ページの『JMS IBM REPORT EXCEPTION』](#)
[2059 ページの『JMS IBM REPORT EXPIRATION』](#)
[2060 ページの『JMS IBM REPORT_NAN』](#)
[2060 ページの『JMS IBM REPORT PAN』](#)
[2060 ページの『JMS IBM REPORT_PASS_CORREL_ID』](#)
[2061 ページの『JMS IBM REPORT_PASS_MSG_ID』](#)
[2061 ページの『JMS IBM SYSTEM_MESSAGEID』](#)
[2062 ページの『JMSX APPID』](#)
[2062 ページの『JMSX DELIVERY_COUNT』](#)
[2062 ページの『JMSX GROUPID』](#)
[2062 ページの『JMSX GROUPSEQ』](#)
[2063 ページの『JMSX USERID』](#)
[2063 ページの『XMSC_CLIENT_ID』](#)

[2064 ページの『XMSC_CONNECTION_TYPE』](#)
[2064 ページの『XMSC_DELIVERY_MODE』](#)
[2065 ページの『XMSC_IC_PROVIDER_URL』](#)
[2065 ページの『XMSC_IC_SECURITY_AUTHENTICATION』](#)
[2065 ページの『XMSC_IC_SECURITY_CREDENTIALS』](#)
[2065 ページの『XMSC_IC_SECURITY_PRINCIPAL』](#)
[2066 ページの『XMSC_IC_SECURITY_PROTOCOL』](#)
[2066 ページの『XMSC_IC_URL』](#)
[2066 ページの『XMSC_IS_SUBSCRIPTION_MULTICAST』](#)
[2066 ページの『XMSC_IS_SUBSCRIPTION_RELIABLE_MULTICAST』](#)
[2066 ページの『XMSC_JMS_MAJOR_VERSION』](#)
[2067 ページの『XMSC_JMS_MINOR_VERSION』](#)
[2067 ページの『XMSC_JMS_VERSION』](#)
[2067 ページの『XMSC_MAJOR_VERSION』](#)
[2067 ページの『XMSC_MINOR_VERSION』](#)
[2067 ページの『XMSC_PASSWORD』](#)
[2068 ページの『XMSC_PRIORITY』](#)
[2068 ページの『XMSC_PROVIDER_NAME』](#)
[2068 ページの『XMSC_RTT_BROKER_PING_INTERVAL』](#)
[2069 ページの『XMSC_RTT_CONNECTION_PROTOCOL』](#)
[2069 ページの『XMSC_RTT_HOST_NAME』](#)
[2069 ページの『XMSC_RTT_LOCAL_ADDRESS』](#)
[2069 ページの『XMSC_RTT_MULTICAST』](#)
[2070 ページの『XMSC_RTT_PORT』](#)
[2070 ページの『XMSC_TIME_TO_LIVE』](#)
[2071 ページの『XMSC_USERID』](#)
[2071 ページの『XMSC_VERSION』](#)
[2073 ページの『XMSC_WMQ_BROKER_CONTROLQ』](#)
[2073 ページの『XMSC_WMQ_BROKER_PUBQ』](#)
[2073 ページの『XMSC_WMQ_BROKER_QMGR』](#)
[2073 ページの『XMSC_WMQ_BROKER_SUBQ』](#)
[2074 ページの『XMSC_WMQ_BROKER_VERSION』](#)
[2074 ページの『XMSC_WMQ_CCDTURL』](#)
[2075 ページの『XMSC_WMQ_CCSID』](#)
[2075 ページの『XMSC_WMQ_CHANNEL』](#)
[2076 ページの『XMSC_WMQ_CONNECTION_MODE』](#)
[2077 ページの『XMSC_WMQ_DUR_SUBQ』](#)
[2077 ページの『XMSC_WMQ_ENCODING』](#)
[2078 ページの『XMSC_WMQ_FAIL_IF_QUIESCE』](#)
[2083 ページの『XMSC_WMQ_HOST_NAME』](#)
[2083 ページの『XMSC_WMQ_LOCAL_ADDRESS』](#)
[2084 ページの『XMSC_WMQ_MESSAGE_SELECTION』](#)
[2084 ページの『XMSC_WMQ_MSG_BATCH_SIZE』](#)
[2085 ページの『XMSC_WMQ_POLLING_INTERVAL』](#)
[2085 ページの『XMSC_WMQ_PORT』](#)
[2086 ページの『XMSC_WMQ_PUB_ACK_INTERVAL』](#)
[2086 ページの『XMSC_WMQ_QMGR_CCSID』](#)
[2086 ページの『XMSC_WMQ_QUEUE_MANAGER』](#)
[2087 ページの『XMSC_WMQ_RECEIVE_EXIT』](#)
[2087 ページの『XMSC_WMQ_RECEIVE_EXIT_INIT』](#)
[2088 ページの『XMSC_WMQ_SECURITY_EXIT』](#)

[2088 ページの『XMSC WMQ SECURITY_EXIT_INIT』](#)
[2089 ページの『XMSC WMQ SEND_EXIT』](#)
[2089 ページの『XMSC WMQ SEND_EXIT_INIT』](#)
[2096 ページの『XMSC WMQ SYNCPOINT_ALL_GETS』](#)
[2096 ページの『XMSC WMQ TARGET_CLIENT』](#)
[2097 ページの『XMSC WMQ TEMP_Q_PREFIX』](#)
[2097 ページの『XMSC WMQ TEMPORARY_MODEL』](#)
[2099 ページの『XMSC WPM_BUS_NAME』](#)
[2099 ページの『XMSC WPM_CONNECTION_PROTOCOL』](#)
[2099 ページの『XMSC WPM_CONNECTION_PROXIMITY』](#)
[2100 ページの『XMSC WPM_DUR_SUB_HOME』](#)
[2100 ページの『XMSC WPM_HOST_NAME』](#)
[2100 ページの『XMSC WPM_LOCAL_ADDRESS』](#)
[2101 ページの『XMSC WPM_ME_NAME』](#)
[2101 ページの『XMSC WPM_NON_PERSISTENT_MAP』](#)
[2102 ページの『XMSC WPM_PERSISTENT_MAP』](#)
[2102 ページの『XMSC WPM_PORT』](#)
[2102 ページの『XMSC WPM_PROVIDER_ENDPOINTS』](#)
[2105 ページの『XMSC WPM_TARGET_GROUP』](#)
[2105 ページの『XMSC WPM_TARGET_SIGNIFICANCE』](#)
[2106 ページの『XMSC WPM_TARGET_TRANSPORT_CHAIN』](#)
[2106 ページの『XMSC WPM_TARGET_TYPE』](#)
[2107 ページの『XMSC WPM_TEMP_Q_PREFIX』](#)
[2107 ページの『XMSC WPM_TEMP_TOPIC_PREFIX』](#)
[2107 ページの『XMSC WPM_TOPIC_SPACE』](#)

JMS_IBM_CHARACTER_SET

データ型:

System.Int32

プロパティ:

メッセージ

クライアントが XMS メッセージを宛先へ転送する際、メッセージ本文内の文字データ列が属する符号化文字セット（またはコードページ）の識別子（CCSID）。XMS では、このプロパティには数値が指定され、CCSID にマップされます。ただし、このプロパティは JMS プロパティに基づいているため、ストリング型の値を持ち、この数値 CCSID を表す Java 文字セットにマップされます。このプロパティは、[XMSC WMQ CCSID](#) プロパティによって宛先に指定されたすべての CCSID に優先します。

デフォルトでは、このプロパティは設定されていません。

アプリケーションがサービス統合バスに接続している場合、このプロパティは関係ありません。

JMS_IBM_ENCODING

データ型:

System.Int32

プロパティ:

メッセージ

クライアントが XMS メッセージを宛先へ転送する際、メッセージ本文内の数値データがどのように表現されるか。このプロパティは、[XMSC WMQ_ENCODING](#) プロパティによって宛先に指定されたすべてのエンコード方式に優先します。このプロパティは、2 進整数、パック 10 進数、および浮動小数点数の表記を指定します。

このプロパティの有効値は、メッセージ記述子の **Encoding** フィールドに指定できる値と同じです。

アプリケーションは、以下の名前付き定数を使用してプロパティを設定できます。

名前付き定数	意味
MQENC_INTEGER_NORMAL	標準の整数エンコード方式
MQENC_INTEGER_REVERSED	逆方向の整数エンコード方式
MQENC_DECIMAL_NORMAL	標準のパック 10 進エンコード方式
MQENC_DECIMAL_REVERSED	逆方向のパック 10 進エンコード方式
MQENC_FLOAT_IEEE_NORMAL	標準の IEEE 浮動小数点エンコード方式
MQENC_FLOAT_IEEE_REVERSED	逆方向の IEEE 浮動小数点エンコード方式
 MQENC_FLOAT_S390	z/OS アーキテクチャの浮動小数点エンコード方式
MQENC_NATIVE	ネイティブのマシン・エンコード方式

プロパティの値を設定するために、アプリケーションは、以下に示す 3 つの定数を加算できます。

- 2 進整数の表記を指定するための、名前が MQENC_INTEGER で始まる定数
- パック 10 進整数の表記を指定するための、名前が MQENC_DECIMAL で始まる定数
- 浮動小数点数の表記を指定するための、名前が MQENC_FLOAT で始まる定数

あるいは、アプリケーションは、その値が環境に依存する MQENC_NATIVE にプロパティを設定できます。

デフォルトでは、このプロパティは設定されていません。

アプリケーションがサービス統合バスに接続している場合、このプロパティは関係ありません。

JMS_IBM_EXCEPTIONMESSAGE

データ型:

ストリング

プロパティ:

メッセージ

メッセージが例外の宛先に送信された理由を説明するテキストです。このプロパティは読み取り専用です。

このプロパティが関連するのは、アプリケーションがサービス統合バスに接続し、例外の宛先からメッセージを受信した場合に限られます。

JMS_IBM_EXCEPTIONPROBLEMDESTINATION

データ型:

ストリング

プロパティ:

メッセージ

メッセージが例外の宛先に送信される前に、そのメッセージが存在した宛先の名前です。

このプロパティが関連するのは、アプリケーションがサービス統合バスに接続し、例外の宛先からメッセージを受信した場合に限られます。

JMS_IBM_EXCEPTIONREASON

データ型:

System.Int32

プロパティ:

メッセージ

メッセージが例外の宛先に送信された理由を示す理由コードです。

このプロパティが関連するのは、アプリケーションがサービス統合バスに接続し、例外の宛先からメッセージを受信した場合に限られます。

JMS_IBM_EXCEPTIONTIMESTAMP

データ型:
System.Int64

プロパティ:
メッセージ

メッセージが例外の宛先に送信された時刻です。

この時刻は、1970 年 1 月 1 日 00:00:00 GMT からの経過時間をミリ秒単位で表現したものです。

このプロパティが関連するのは、アプリケーションがサービス統合バスに接続し、例外の宛先からメッセージを受信した場合に限られます。

JMS_IBM_FEEDBACK

データ型:
System.Int32

プロパティ:
メッセージ

レポート・メッセージの種類を示すコードです。

このプロパティの有効値は、メッセージ記述子の **Feedback** フィールドに指定できるフィードバック・コードおよび理由コードです。

デフォルトでは、このプロパティは設定されていません。

JMS_IBM_FORMAT

データ型:
ストリング

プロパティ:
メッセージ

メッセージ内にあるアプリケーション・データの種類の種類です。

このプロパティの有効値は、メッセージ記述子の **Format** フィールドに指定できる値と同じです。

デフォルトでは、このプロパティは設定されていません。

アプリケーションがサービス統合バスに接続している場合、このプロパティは関係ありません。

JMS_IBM_LAST_MSG_IN_GROUP

データ型:
System.Boolean

プロパティ:
メッセージ

メッセージがメッセージ・グループ内の最後のメッセージであるかどうかを示します。

メッセージがメッセージ・グループ内の最後のメッセージである場合は、このプロパティを **true** に設定します。それ以外の場合は、このプロパティを **false** に設定するか、このプロパティを設定しないようにします。デフォルトでは、このプロパティは設定されていません。

true の値は、状況フラグ **MQMF_LAST_MSG_IN_GROUP** に対応しています。これは、メッセージ記述子の **MsgFlags** フィールドで指定できます。

このプロパティはパブリッシュ/サブスクライブ・ドメイン内では無視され、アプリケーションがサービス統合バスに接続している場合は関係ありません。

JMS_IBM_MSGTYPE

データ型:

System.Int32

プロパティ:

メッセージ

メッセージのタイプ。

プロパティの有効値は以下のとおりです。

有効値

意味

MQMT_DATAGRAM

メッセージは、応答が不要なメッセージです。

MQMT_REQUEST

メッセージは、応答が必要なメッセージです。

MQMT_REPLY

このメッセージは、応答メッセージです。

MQMT_REPORT

このメッセージは、レポート・メッセージです。

これらの値は、メッセージ記述子の **MsgType** フィールドに指定できるメッセージのタイプに対応します。

デフォルトでは、このプロパティは設定されていません。

アプリケーションがサービス統合バスに接続している場合、このプロパティは関係ありません。

JMS_IBM_PUTAPPLTYPE

データ型:

System.Int32

プロパティ:

メッセージ

メッセージを送信したアプリケーションのタイプです。

このプロパティの有効値は、メッセージ記述子の **PutApplType** フィールドに指定できるアプリケーション・タイプです。

デフォルトでは、このプロパティは設定されていません。

アプリケーションがサービス統合バスに接続している場合、このプロパティは関係ありません。

JMS_IBM_PUTDATE

データ型:

ストリング

プロパティ:

メッセージ

メッセージが送信された日付です。

このプロパティの有効値は、メッセージ記述子の **PutDate** フィールドに指定できる値と同じです。

デフォルトでは、このプロパティは設定されていません。

アプリケーションがサービス統合バスに接続している場合、このプロパティは関係ありません。

JMS_IBM_PUTTIME

データ型:

ストリング

プロパティ:

メッセージ

メッセージが送信された時刻です。

このプロパティの有効値は、メッセージ記述子の **PutTime** フィールドに指定できる値と同じです。

デフォルトでは、このプロパティは設定されていません。

アプリケーションがサービス統合バスに接続している場合、このプロパティは関係ありません。

JMS_IBM_REPORT_COA

データ型:

System.Int32

プロパティ:

メッセージ

「到着時の確認」レポート・メッセージを要求し、元のメッセージからのアプリケーション・データをレポート・メッセージにどの程度組み込む必要があるかを指定します。

プロパティの有効値は以下のとおりです。

有効値

意味

MQRO_COA

「到着時の確認」レポート・メッセージを要求します。元のメッセージからのアプリケーション・データはレポート・メッセージに組み込みません。

MQRO_COA_WITH_DATA

「到着時の確認」レポート・メッセージを要求し、元のメッセージからのアプリケーション・データのうち先頭の 100 バイトをレポート・メッセージに組み込みます。

MQRO_COA_WITH_FULL_DATA

「到着時の確認」レポート・メッセージを要求し、元のメッセージからのすべてのアプリケーション・データをレポート・メッセージに組み込みます。

これらの値は、メッセージ記述子の **Report** フィールドに指定できるレポート・オプションに対応しています。これらのオプションの詳細については、[Report \(MQLONG\)](#)を参照してください。

デフォルトでは、このプロパティは設定されていません。

JMS_IBM_REPORT_COD

データ型:

System.Int32

プロパティ:

メッセージ

「配信時の確認」レポート・メッセージを要求し、元のメッセージからのアプリケーション・データをレポート・メッセージにどの程度組み込む必要があるかを指定します。

プロパティの有効値は以下のとおりです。

有効値

意味

MQRO_COD

「配信時の確認」レポート・メッセージを要求します。元のメッセージからのアプリケーション・データはレポート・メッセージに組み込みません。

MQRO_COD_WITH_DATA

「配信時の確認」レポート・メッセージを要求し、元のメッセージからのアプリケーション・データのうち先頭の 100 バイトをレポート・メッセージに組み込みます。

MQRO_COD_WITH_FULL_DATA

「配信時の確認」レポート・メッセージを要求し、元のメッセージからのすべてのアプリケーション・データをレポート・メッセージに組み込みます。

これらの値は、メッセージ記述子の **Report** フィールドに指定できるレポート・オプションに対応しています。

デフォルトでは、このプロパティは設定されていません。

JMS_IBM_REPORT_DISCARD_MSG

データ型:
System.Int32

プロパティ:
メッセージ

メッセージを目的の宛先に配信できない場合に、そのメッセージを廃棄することを要求します。

メッセージを目的の宛先に配信できない場合に、そのメッセージを廃棄することを要求するには、このプロパティを MQRO_DISCARD_MSG に設定します。そうではなく、メッセージを送達不能キューに書き込むか、例外の宛先に送信することを要求する場合は、このプロパティを設定しないでください。デフォルトでは、このプロパティは設定されていません。

MQRO_DISCARD_MSG という値は、メッセージ記述子の **Report** フィールドに指定できるレポート・オプションに対応します。

JMS_IBM_REPORT_EXCEPTION

データ型:
System.Int32

プロパティ:
メッセージ

例外レポート・メッセージを要求し、元のメッセージからのアプリケーション・データをレポート・メッセージにどの程度組み込む必要があるかを指定します。

プロパティの有効値は以下のとおりです。

有効値	意味
MQRO_EXCEPTION	例外レポート・メッセージを要求します。元のメッセージからのアプリケーション・データはレポート・メッセージに組み込みません。
MQRO_EXCEPTION_WITH_DATA	例外レポート・メッセージを要求し、元のメッセージからのアプリケーション・データのうち先頭の 100 バイトをレポート・メッセージに組み込みます。
MQRO_EXCEPTION_WITH_FULL_DATA	例外レポート・メッセージを要求し、元のメッセージからのすべてのアプリケーション・データをレポート・メッセージに組み込みます。

これらの値は、メッセージ記述子の **Report** フィールドに指定できるレポート・オプションに対応しています。

デフォルトでは、このプロパティは設定されていません。

JMS_IBM_REPORT_EXPIRATION

データ型:
System.Int32

プロパティ:
メッセージ

期限満了レポート・メッセージを要求し、元のメッセージからのアプリケーション・データをレポート・メッセージにどの程度組み込む必要があるかを指定します。

プロパティの有効値は以下のとおりです。

有効値

MQRO_EXPIRATION

意味

期限満了レポート・メッセージを要求します。元のメッセージからのアプリケーション・データはレポート・メッセージに組み込みません。

MQRO_EXPIRATION_WITH_DATA

期限満了レポート・メッセージを要求し、元のメッセージからのアプリケーション・データのうち先頭の 100 バイトをレポート・メッセージに組み込みます。

MQRO_EXPIRATION_WITH_FULL_DATA

期限満了レポート・メッセージを要求し、元のメッセージからのすべてのアプリケーション・データをレポート・メッセージに組み込みます。

これらの値は、メッセージ記述子の **Report** フィールドに指定できるレポート・オプションに対応しています。

デフォルトでは、このプロパティは設定されていません。

JMS_IBM_REPORT_NAN

データ型:

System.Int32

プロパティ:

メッセージ

否定アクション通知レポート・メッセージを要求します。

否定アクション通知レポート・メッセージを要求するには、このプロパティを MQRO_NAN に設定します。否定アクション通知レポート・メッセージが必要でない場合は、このプロパティを設定しないでください。デフォルトでは、このプロパティは設定されていません。

MQRO_NAN という値は、メッセージ記述子の **Report** フィールドに指定できるレポート・オプションに対応します。

JMS_IBM_REPORT_PAN

データ型:

System.Int32

プロパティ:

メッセージ

肯定アクション通知レポート・メッセージを要求します。

肯定アクション通知レポート・メッセージを要求するには、このプロパティを MQRO_PAN に設定します。肯定アクション通知レポート・メッセージが必要でない場合は、このプロパティを設定しないでください。デフォルトでは、このプロパティは設定されていません。

MQRO_PAN という値は、メッセージ記述子の **Report** フィールドに指定できるレポート・オプションに対応します。

JMS_IBM_REPORT_PASS_CORREL_ID

データ型:

System.Int32

プロパティ:

メッセージ

任意のレポート・メッセージまたは応答メッセージの相関 ID を元のメッセージの相関 ID と同じにするという要求です。

プロパティの有効値は以下のとおりです。

有効値	意味
MQRO_PASS_CORREL_ID	任意のレポート・メッセージまたは応答メッセージの相関 ID を元のメッセージの相関 ID と同じにするという要求です。
MQRO_COPY_MSG_ID_TO_CORREL_ID	任意のレポート・メッセージまたは応答メッセージの相関 ID を元のメッセージのメッセージ ID と同じにするという要求です。

これらの値は、メッセージ記述子の **Report** フィールドで指定可能なレポートオプションに対応します。
このプロパティのデフォルト値は MQRO_COPY_MSG_ID_TO_CORREL_ID です。

JMS_IBM_REPORT_PASS_MSG_ID

データ型:
System.Int32
プロパティ:
メッセージ

任意のレポート・メッセージまたは応答メッセージのメッセージ ID を元のメッセージのメッセージ ID と同じにするという要求です。

プロパティの有効値は以下のとおりです。

有効値	意味
MQRO_PASS_MSG_ID	任意のレポート・メッセージまたは応答メッセージのメッセージ ID を元のメッセージのメッセージ ID と同じにするという要求です。
MQRO_NEW_MSG_ID	レポート・メッセージまたは応答メッセージごとに新しいメッセージ ID を生成するという要求です。

これらの値は、メッセージ記述子の **Report** フィールドに指定できるレポート・オプションに対応しています。

このプロパティのデフォルト値は MQRO_NEW_MSG_ID です。

JMS_IBM_RETAIN

データ型:
System.Int32
プロパティ:
メッセージ

このプロパティを設定すると、キュー・マネージャーは、メッセージを保存パブリケーションとして扱うように指定されます。サブスクライバーは、トピックからメッセージを受け取る時に、前のリリースで受け取っていたメッセージのほかに、サブスクライブ直後に追加のメッセージを受け取ることがあります。それらのメッセージは、サブスクライブしたトピックのオプションの保存パブリケーションです。サブスクリプションに合致するトピックごとに、保存パブリケーションがある場合は、そのパブリケーションも、サブスクライブしているメッセージ・コンシューマーに配信できるようになります。

RETAIN_PUBLICATION は、このプロパティで唯一有効な値です。デフォルトでは、このプロパティは設定されていません。

注: このプロパティが関連するのは、パブリッシュ/サブスクライブ・ドメインの場合に限られます。

JMS_IBM_SYSTEM_MESSAGEID

データ型:
ストリング

プロパティ:
メッセージ

サービス統合バスの内部でメッセージを一意的に識別する ID です。このプロパティは読み取り専用です。

このプロパティが関連するのは、アプリケーションがサービス統合バスに接続している場合に限られます。

JMSX_APPID

データ型:
ストリング

プロパティ:
メッセージ

メッセージを送信したアプリケーションの名前です。

このプロパティは JMS 定義のプロパティで、JMSXAppID という JMS 名が付いています。プロパティの詳細については、メッセージサービス仕様書（バージョン 1.1Java）を参照してください。

デフォルトでは、このプロパティは設定されていません。

ブローカーへのリアルタイム接続の場合、このプロパティは無効です。

JMSX_DELIVERY_COUNT

データ型:
System.Int32

プロパティ:
メッセージ

メッセージ配信の試行回数です。

このプロパティは JMS 定義のプロパティで、JMSXDeliveryCount という JMS 名が付いています。物件の詳細については、仕様 *Java Message Service* 書（バージョン 1.1）をご覧ください。

デフォルトでは、このプロパティは設定されていません。

ブローカーへのリアルタイム接続の場合、このプロパティは無効です。

JMSX_GROUPID

データ型:
ストリング

プロパティ:
メッセージ

メッセージが属するメッセージ・グループの ID です。

このプロパティは JMS 定義のプロパティで、JMSXGroupID という JMS 名が付いています。物件の詳細については、仕様 *Java Message Service* 書（バージョン 1.1）をご覧ください。

デフォルトでは、このプロパティは設定されていません。

ブローカーへのリアルタイム接続の場合、このプロパティは無効です。

JMSX_GROUPSEQ

データ型:
System.Int32

プロパティ:
メッセージ

メッセージ・グループ内にあるメッセージのシーケンス番号です。

このプロパティは JMS 定義のプロパティで、JMSXGroupSeq という JMS 名が付いています。物件の詳細については、仕様 *Java Message Service* 書（バージョン 1.1）をご覧ください。

デフォルトでは、このプロパティは設定されていません。

ブローカーへのリアルタイム接続の場合、このプロパティは無効です。

JMSX_USERID

データ型:

ストリング

プロパティ:

メッセージ

メッセージを送信したアプリケーションに関連付けられているユーザー ID です。

このプロパティは JMS 定義のプロパティで、JMSXUserID という JMS 名が付いています。物件の詳細については、仕様 *Java Message Service* 書（バージョン 1.1）をご覧ください。

デフォルトでは、このプロパティは設定されていません。

ブローカーへのリアルタイム接続の場合、このプロパティは無効です。

XMSC_ASYNC_EXCEPTIONS

データ型:

System.Int32

プロパティ:

ConnectionFactory

適用可能なオブジェクト:

JMS 管理ツールのロング・ネーム: ASYNCEXCEPTION

JMS 管理ツールのショート・ネーム: AEX

このプロパティは、XMS が ExceptionListener への通知を、接続が切断されたときのみ行うか、または XMS API 呼び出しに対して非同期に例外が発生したときに行うかを決定します。このプロパティは、ExceptionListener が登録されているこの ConnectionFactory から作成されたすべての接続に適用されます。

このプロパティの有効な値は以下のとおりです。

XMSC_ASYNC_EXCEPTIONS_ALL

同期 API 呼び出しの有効範囲外で非同期に検出されたすべての例外、および接続切断の例外はすべて ExceptionListener に送信されます。

XMSC_ASYNC_EXCEPTIONS_CONNECTIONBROKEN

切断された接続を示す例外のみが ExceptionListener に送信されます。非同期処理中に起きる他のすべての例外は ExceptionListener には報告されず、そのためアプリケーションにはそれらの例外は通知されません。

デフォルトでは、このプロパティは XMSC_ASYNC_EXCEPTIONS_ALL に設定されます。

XMSC_CLIENT_ID

データ型:

ストリング

プロパティ:

ConnectionFactory

適用可能なオブジェクト:

JMS 管理ツールのロング・ネーム: CLIENTID

JMS 管理ツールのショート・ネーム: CID

接続のクライアント ID です。

クライアント ID は、パブリッシュ/サブスクライブ・ドメイン内の永続サブスクリプションをサポートするためだけに使用され、Point-to-Point ドメインでは無視されます。クライアント ID の設定についての詳細は、[ConnectionFactory](#) および [Connection](#) オブジェクトを参照してください。

このプロパティは、ブローカーへのリアルタイム接続には関連していません。

XMSC_CONNECTION_TYPE

データ型:

System.Int32

プロパティ:

ConnectionFactory

アプリケーションの接続先となるメッセージング・サーバーのタイプです。

プロパティの有効値は以下のとおりです。

有効値	意味
XMSC_CT_RTT	ブローカーへのリアルタイム接続です。
XMSC_CT_WMQ	IBM MQ キュー・マネージャーへの接続です。
XMSC_CT_WPM	WebSphere Application Server service integration bus への接続。

デフォルトでは、このプロパティは設定されていません。

XMSC_DELIVERY_MODE

データ型:

System.Int32

プロパティ:

Destination

URI で使用される名前:

persistence (IBM MQ 宛先の場合)

deliveryMode (WebSphere デフォルト・メッセージング・プロバイダーの宛先の場合)

適用可能なオブジェクト:

JMS 管理ツールのロング・ネーム: PERSISTENCE

JMS 管理ツールのショート・ネーム: PER

宛先に送信されたメッセージの送達モードです。

プロパティの有効値は以下のとおりです。

有効値	意味
XMSC_DELIVERY_NOT_PERSISTENT	宛先に送信されたメッセージは、非永続です。メッセージ・プロデューサーのデフォルトの送達モード、または Send 呼び出しに指定されている任意の送達モードは無視されます。宛先が IBM MQ キューである場合は、キュー属性 <i>DefPersistence</i> の値も無視されます。
XMSC_DELIVERY_PERSISTENT	宛先に送信されたメッセージは、永続です。メッセージ・プロデューサーのデフォルトの送達モード、または Send 呼び出しに指定されている任意の送達モードは無視されます。宛先が IBM MQ キューである場合は、キュー属性 <i>DefPersistence</i> の値も無視されます。

有効値

XMSC_DELIVERY_AS_APP

意味

宛先に送信されたメッセージの送達モードは、Send 呼び出しに指定されている送達モードです。Send 呼び出しに送達モードが指定されていない場合は、代わりにメッセージ・プロデューサーのデフォルトの送達モードが使用されます。宛先が IBM MQ キューである場合は、キュー属性 *DefPersistence* の値も無視されます。

XMSC_DELIVERY_AS_DEST

宛先が IBM MQ キューである場合、キューに書き込まれたメッセージの送達モードは、キュー属性 *DefPersistence* の値で指定された送達モードになります。メッセージ・プロデューサーのデフォルトの送達モード、または Send 呼び出しに指定されている任意の送達モードは無視されます。

宛先が IBM MQ キューではない場合、この値の意味は XMSC_DELIVERY_AS_APP の意味と同じです。

デフォルト値は XMSC_DELIVERY_AS_APP です。

XMSC_IC_PROVIDER_URL

データ型:

ストリング

プロパティ:

InitialContext

JNDI ネーミング・ディレクトリーの位置を指定するために使用します。その位置を指定すれば、COS ネーミング・サービスが Web サービスと同じサーバーに存在する必要はなくなります。

XMSC_IC_SECURITY_AUTHENTICATION

データ型:

ストリング

プロパティ:

InitialContext

Java Context インターフェース SECURITY_AUTHENTICATION に基づいています。このプロパティは COS ネーミング・コンテキストにのみ適用されます。

XMSC_IC_SECURITY_CREDENTIALS

データ型:

ストリング

プロパティ:

InitialContext

Context インターフェース Java SECURITY_CREDENTIALS に基づいています。このプロパティは COS ネーミング・コンテキストにのみ適用されます。

XMSC_IC_SECURITY_PRINCIPAL

データ型:

ストリング

プロパティ:

InitialContext

Java Context インターフェース SECURITY_PRINCIPAL に基づいています。このプロパティは COS ネーミング・コンテキストにのみ適用されます。

XMSC_IC_SECURITY_PROTOCOL

データ型:
 ストリング

プロパティ:
 InitialContext

Java Context インターフェースに基づく SECURITY_PROTOCOL このプロパティは COS ネーミング・コンテキストにのみ適用されます。

XMSC_IC_URL

データ型:
 ストリング

プロパティ:
 InitialContext

LDAP コンテキストや FileSystem コンテキストの場合は、管理対象オブジェクトを収容しているリポジトリのアドレスです。

LDAP コンテキストや FileSystem コンテキストの場合は、管理対象オブジェクトを収容しているリポジトリのアドレスです。

XMSC_IS_SUBSCRIPTION_MULTICAST

データ型:
 System.Boolean

プロパティ:
 MessageConsumer

WebSphere MQ Multicast Transport メッセージがメッセージコンシューマーに配信されているかどうかを示します。このプロパティは読み取り専用です。

メッセージが WebSphere MQ Multicast Transport を使用してメッセージ・コンシューマーに配信される場合、このプロパティの値は true です。それ以外の場合、値は false です。

このプロパティは、ブローカーへのリアルタイム接続にのみ関連します。

XMSC_IS_SUBSCRIPTION_RELIABLE_MULTICAST

データ型:
 System.Boolean

プロパティ:
 MessageConsumer

メッセージが、信頼性の高いサービス品質（QoS）を用いて WebSphere MQ Multicast Transport メッセージコンシューマーに配信されているかどうかを示します。このプロパティは読み取り専用です。

メッセージが、信頼性の高いサービス品質を備えた WebSphere MQ Multicast Transport を使用してメッセージ・コンシューマーに配信される場合、このプロパティの値は true です。それ以外の場合、値は false です。

このプロパティは、ブローカーへのリアルタイム接続にのみ関連します。

XMSC_JMS_MAJOR_VERSION

データ型:
 System.Int32

プロパティ:
 ConnectionMetaData

が XMS に基づいている仕 JMS 様のメジャーバージョン番号。このプロパティは読み取り専用です。

XMSC_JMS_MINOR_VERSION

データ型:

System.Int32

プロパティ:

ConnectionMetaData

が XMS 基づいている仕 JMS 様のマイナーバージョン番号。このプロパティは読み取り専用です。

XMSC_JMS_VERSION

データ型:

ストリング

プロパティ:

ConnectionMetaData

が XMS 基づく仕 JMS 様のバージョン識別子。このプロパティは読み取り専用です。

XMSC_MAJOR_VERSION

データ型:

System.Int32

プロパティ:

ConnectionMetaData

XMS クライアントのバージョン番号。このプロパティは読み取り専用です。

XMSC_MINOR_VERSION

データ型:

System.Int32

プロパティ:

ConnectionMetaData

XMS クライアントのリリース番号。このプロパティは読み取り専用です。

XMSC_PASSWORD

データ型:

バイト配列

プロパティ:

ConnectionFactory

アプリケーションがメッセージング・サーバーに接続しようとしているときに、このアプリケーションを認証するために使用するパスワードです。このパスワードは、XMSC_USERID プロパティと一緒に使用します。

デフォルトでは、このプロパティは設定されていません。

Multi マルチプラットフォーム上の IBM MQ に接続する場合、接続ファクトリーの XMSC_USERID プロパティを設定するときは、ログオン・ユーザーの **userid** と一致する必要があります。これらのプロパティを設定しない場合、キュー・マネージャーはデフォルトでログオン・ユーザーの **userid** を使用します。個々のユーザーの接続レベル認証がさらに必要な場合には、IBM MQ で構成済みのクライアント認証出口を作成できます。クライアント認証出口の作成について詳しくは、クライアント・アプリケーションの認証の計画を参照してください。

z/OS IBM MQ for z/OS に接続するときにユーザーを認証するには、セキュリティ出口を使用する必要があります。

XMSC_PRIORITY

データ型:

System.Int32

プロパティ:

Destination

URI で使用される名前:

priority

宛先に送信されたメッセージの優先順位です。

プロパティの有効値は以下のとおりです。

有効値

0 (最低の優先順位) から 9 (最高の優先順位) までの範囲の整数

意味

宛先に送信されたメッセージには、指定された優先順位があります。メッセージ・プロデューサーのデフォルトの優先順位、および Send 呼び出しに指定されているすべての優先順位は無視されます。宛先が IBM MQ キューである場合は、キュー属性 **DefPriority** の値も無視されます。

XMSC_PRIORITY_AS_APP

宛先に送信されたメッセージの優先順位は、Send 呼び出しに指定されている優先順位です。Send 呼び出しに優先順位が指定されていない場合は、代わりにメッセージ・プロデューサーのデフォルトの優先順位が使用されます。宛先が IBM MQ キューである場合は、キュー属性 **DefPriority** の値は無視されます。

XMSC_PRIORITY_AS_DEST

宛先が IBM MQ キューである場合、キューに書き込まれたメッセージの優先順位は、キュー属性 **DefPriority** の値で指定された優先順位になります。メッセージ・プロデューサーのデフォルトの優先順位、および Send 呼び出しに指定されているすべての優先順位は無視されます。

宛先が IBM MQ キューではない場合、この値の意味は XMSC_PRIORITY_AS_APP の意味と同じです。

デフォルト値は XMSC_PRIORITY_AS_APP です。

WebSphere MQ Real-Time Transport および WebSphere MQ Multicast Transport には、メッセージの優先順位に基づく動作はありません。

XMSC_PROVIDER_NAME

データ型:

ストリング

プロパティ:

ConnectionMetaData

XMS クライアントの提供元。このプロパティは読み取り専用です。

XMSC_RTT_BROKER_PING_INTERVAL

データ型:

System.Int32

プロパティ:

ConnectionFactory

XMS .NET がリアルタイム・メッセージング・サーバーへの接続を検査することでアクティビティの検出を行うまでの時間間隔 (ミリ秒単位)。アクティビティが検出されない場合、クライアントは ping を開始します。ping に応答が検出されない場合、接続はクローズされます。

このプロパティのデフォルト値は 30000 です。

XMSC_RTT_CONNECTION_PROTOCOL

データ型:

System.Int32

プロパティ:

ConnectionFactory

ブローカーへのリアルタイム接続に使用される通信プロトコルです。

このプロパティの値は、TCP/IP を使用したブローカーへのリアルタイム接続という意味の XMSC_RTT_CP_TCP にする必要があります。デフォルト値は XMSC_RTT_CP_TCP です。

XMSC_RTT_HOST_NAME

データ型:

ストリング

プロパティ:

ConnectionFactory

ブローカーを実行するシステムのホスト名または IP アドレスです。

このプロパティは、ブローカーを識別するために、XMSC_RTT_PORT プロパティと一緒に使用します。

デフォルトでは、このプロパティは設定されていません。

XMSC_RTT_LOCAL_ADDRESS

データ型:

ストリング

プロパティ:

ConnectionFactory

ブローカーへのリアルタイム接続に使用するローカル・ネットワーク・インターフェースのホスト名または IP アドレスです。

このプロパティが有用なのは、アプリケーションを実行しているシステムに複数のネットワーク・インターフェースがあり、リアルタイム接続にどのインターフェースを使用する必要があるかを指定できることが必要な場合に限ります。システムが備えるネットワーク・インターフェースが1つのみである場合、使用できるのはそのインターフェースのみです。システムに複数のネットワーク・インターフェースがあり、このプロパティを設定していない場合、ネットワーク・インターフェースはランダムに選択されます。

デフォルトでは、このプロパティは設定されていません。

XMSC_RTT_MULTICAST

データ型:

System.Int32

プロパティ:

ConnectionFactory および Destination

URI で使用される名前:

multicast

このプロパティを持つことができるのは、トピックに含まれる宛先のみです 接続ファクトリーまたは宛先のマルチキャスト設定です。

アプリケーションがこのプロパティを使用する目的は、以下の2つです。1つは、ブローカーへのリアルタイム接続と関連してマルチキャストを使用可能にすることで、もう1つは、マルチキャストが使用可能になった場合に、ブローカーからメッセージ・コンシューマーにメッセージを配信するときにマルチキャストを使用する方法を正確に指定することです。このプロパティは、メッセージ・プロデューサーによるブローカーへのメッセージ送信方法については影響しません。

プロパティの有効値は以下のとおりです。

有効値

XMSC_RTT_MULTICAST_DISABLED

XMSC_RTT_MULTICAST_ASCF

XMSC_RTT_MULTICAST_ENABLED

XMSC_RTT_MULTICAST_RELIABLE

XMSC_RTT_MULTICAST_NOT_RELIABLE

意味

メッセージは WebSphere MQ Multicast Transport によってメッセージ・コンシューマーに配信されません。この値は、ConnectionFactory オブジェクトのデフォルト値です。

メッセージは、メッセージ・コンシューマーに関連付けられた接続ファクトリーのマルチキャスト設定に応じて、メッセージ・コンシューマーに送達されます。接続ファクトリーのマルチキャスト設定は、接続経路を作成するときに指定します。この値は Destination オブジェクトに対してのみ有効であり、Destination オブジェクトのデフォルト値になります。

ブローカーでのマルチキャストに合わせてトピックを構成した場合、メッセージは WebSphere MQ Multicast Transport によってメッセージ・コンシューマーに配信されます。信頼性の高いマルチキャストに合わせてトピックを構成した場合は、信頼性の高いサービス品質が使用されます。

ブローカーでの信頼性の高いマルチキャストに合わせてトピックを構成した場合、メッセージは、信頼性の高いサービス品質を備えた WebSphere MQ Multicast Transport によってメッセージ・コンシューマーに配信されます。信頼性の高いマルチキャストに合わせてトピックを構成しなかった場合は、このトピックに対してメッセージ・コンシューマーを作成することはできません。

ブローカーでのマルチキャストに合わせてトピックを構成した場合、メッセージは WebSphere MQ Multicast Transport によってメッセージ・コンシューマーに配信されます。信頼性の高いマルチキャストに合わせてトピックを構成した場合でも、信頼性の高いサービス品質は使用されません。

XMSC_RTT_PORT

データ型:

System.Int32

プロパティ:

ConnectionFactory

ブローカーが着信要求を listen するポートの数です。ブローカー上で、このポートで listen するために、Real-timeInput または Real-timeOptimizedFlow メッセージ処理ノードを構成する必要があります。

このプロパティは、ブローカーを識別するために、XMSC_RTT_HOST_NAME プロパティと一緒に使用します。

このプロパティのデフォルト値は XMSC_RTT_DEFAULT_PORT、つまり 1506 です。

XMSC_TIME_TO_LIVE

データ型:

System.Int32

プロパティ:

Destination

URI で使用される名前:

expiry (IBM MQ 宛先の場合)

timeToLive (WebSphere デフォルト・メッセージング・プロバイダーの宛先の場合)

宛先に送信されたメッセージの存続時間です。

プロパティの有効値は以下のとおりです。

有効値

0

正整数

意味

宛先に送信されたメッセージには有効期限がありません。

宛先に送信されたメッセージには、指定された存続時間 (ミリ秒単位) があります。メッセージ・プロデューサーのデフォルトの存続時間、または Send 呼び出しに指定されているすべての存続時間は無視されます。

XMSC_TIME_TO_LIVE_AS_APP

宛先に送信されたメッセージの存続時間は、Send 呼び出しに指定されている存続時間です。Send 呼び出しに存続時間が指定されていない場合は、代わりにメッセージ・プロデューサーのデフォルトの存続時間を使用されます。

デフォルト値は XMSC_TIME_TO_LIVE_AS_APP です。

XMSC_USERID

データ型:

ストリング

プロパティ:

ConnectionFactory

アプリケーションがメッセージング・サーバーに接続しようとしているときに、このアプリケーションを認証するために使用するユーザー ID です。このユーザー ID は、XMSC_PASSWORD プロパティと一緒に使用します。

デフォルトでは、このプロパティは設定されていません。

Multi IBM MQ for Multiplatforms に接続する場合、接続ファクトリーの XMSC_USERID プロパティを設定するときは、ログオン・ユーザーの **userid** と一致する必要があります。これらのプロパティを設定しない場合、キュー・マネージャーはデフォルトでログオン・ユーザーの **userid** を使用します。個々のユーザーの接続レベル認証がさらに必要な場合には、IBM MQ で構成済みのクライアント認証出口を作成できます。クライアント認証出口の作成について詳しくは、[クライアント・アプリケーションの認証の計画](#)を参照してください。

z/OS IBM MQ for z/OS に接続するときにユーザーを認証するには、セキュリティ出口を使用する必要があります。

XMSC_VERSION

データ型:

ストリング

プロパティ:

ConnectionMetaData

XMS クライアントのバージョン識別子。このプロパティは読み取り専用です。

XMSC_WMQ_BALANCING_APPLICATION_TYPE

データ型:

System.Int32

プロパティ:

ConnectionFactory

プロパティの有効値は以下のとおりです。

有効値

XMSC_WMQ_BALANCING_APPLICATION_TYPE_SIMPLE

意味

単純なバランシング。一様クラスターでのアプリケーションの再平衡化の影響で説明されているものに加えて、特定のルールは適用されません。これがデフォルト値です。

XMSC_WMQ_BALANCING_APPLICATION_TYPE_REQUEST_REPLY

要求/応答のバランシング。各 MQPUT 呼び出しの後で、対応する MQGET 呼び出しが応答メッセージに対して予期されます。そのようなメッセージが受信されるか、要求メッセージの EXPIRY が超過するまで、バランシングは遅延されます。

さらに、このプロパティは client.ini ファイルで設定できます。設定の順序は次のとおりです。

1. アプリケーションで設定されたプロパティ
2. mqclient.ini ファイル内の名前付き [Application stanza](#) に一致します。
3. mqclient.ini ファイル内の [アプリケーション・デフォルト・スタンザ](#)。

XMSC_WMQ_BALANCING_OPTIONS

データ型:

System.Int32

プロパティ:

ConnectionFactory

プロパティの有効値は以下のとおりです。

有効値

XMSC_WMQ_BALANCING_OPTIONS_NONE

対応する値

オプションは設定されていません。これはデフォルト値です

XMSC_WMQ_BALANCING_OPTIONS_IGNORE_TRANSACTION

このオプションを設定すると、トランザクションの途中であっても、アプリケーションをリバランスすることができます。

さらに、このプロパティは client.ini ファイルで設定できます。設定の順序は次のとおりです。

1. アプリケーションで設定されたプロパティ
2. mqclient.ini ファイル内の名前付き [Application stanza](#) に一致します。
3. mqclient.ini ファイル内の [アプリケーション・デフォルト・スタンザ](#)。

XMSC_WMQ_BALANCING_TIMEOUT

データ型:

System.Int32

プロパティ:

ConnectionFactory

プロパティの有効値は以下のとおりです。

有効値	意味
XMSC_WMQ_BALANCING_TIMEOUT_IMMEDIATE	即時タイムアウトが発生
XMSC_WMQ_BALANCING_TIMEOUT_AS_DEFAULT	設定されたデフォルトのタイムアウト値。これはデフォルト値です
XMSC_WMQ_BALANCING_TIMEOUT_NEVER	タイムアウト発生なし

注：定義された値から 1 つの値、または 0 から 999999999 秒の値を指定する必要があります。

さらに、このプロパティは `client.ini` ファイルで設定できます。設定の順序は次のとおりです。

1. アプリケーションで設定されたプロパティ
2. `mqclient.ini` ファイル内の名前付き [Application stanza](#) に一致します。
3. `mqclient.ini` ファイル内の [アプリケーション・デフォルト・スタンザ](#)。

XMSC_WMQ_BROKER_CONTROLQ

データ型:

ストリング

プロパティ:

ConnectionFactory

ブローカーが使用する制御キューの名前。

プロパティのデフォルト値は `SYSTEM.BROKER.CONTROL.QUEUE` です。

このプロパティが関連するのは、パブリッシュ/サブスクライブ・ドメインの場合に限られます。

XMSC_WMQ_BROKER_PUBQ

データ型:

ストリング

プロパティ:

ConnectionFactory

ブローカーによってモニターされているキューの名前で、このキューでは、アプリケーションが発行したメッセージをアプリケーション自体が送信します。

プロパティのデフォルト値は `SYSTEM.BROKER.DEFAULT.STREAM` です。

このプロパティが関連するのは、パブリッシュ/サブスクライブ・ドメインの場合に限られます。

XMSC_WMQ_BROKER_QMGR

データ型:

ストリング

プロパティ:

ConnectionFactory

ブローカーの接続先となるキュー・マネージャーの名前です。

デフォルトでは、このプロパティは設定されていません。

このプロパティが関連するのは、パブリッシュ/サブスクライブ・ドメインの場合に限られます。

XMSC_WMQ_BROKER_SUBQ

データ型:

ストリング

プロパティ:

ConnectionFactory

非永続メッセージ・コンシューマー用のサブスクライバー・キューの名前です。

以下の文字で始まる必要があるサブスクライバー・キューの名前です。

SYSTEM.JMS.ND.

すべての非永続メッセージ・コンシューマーがサブスクライバー・キューを共有するには、共有キューの完全な名前を指定します。アプリケーションが非永続メッセージ・コンシューマーを作成できるようにするには、指定した名前のキューが事前に存在する必要があります。

各非永続メッセージ・コンシューマーが自身の排他的サブスクライバー・キューからメッセージを検索するようにしたい場合は、アスタリスク (*)で終わるキュー名を指定します。その後、アプリケーションが非永続メッセージ・コンシューマーを作成すると、XMS クライアントは、メッセージ・コンシューマーが排他使用するための動的キューを作成します。XMS クライアントは、このプロパティの値を使用して、動的キューを作成するときに使用するオブジェクト記述子の **DynamicQName** フィールドの内容を設定します。

このプロパティのデフォルト値は SYSTEM.JMS.ND.SUBSCRIBER.QUEUE で、これは、XMS は、デフォルトでは共有キューの方法を使用するという意味です。

このプロパティが関連するのは、パブリッシュ/サブスクライブ・ドメインの場合に限られます。

XMSC_WMQ_BROKER_VERSION

データ型:

System.Int32

プロパティ:

ConnectionFactory および Destination

URI で使用される名前:

brokerVersion

このプロパティを持つことができるのは、トピックに含まれる宛先のみです。接続または宛先に合わせてアプリケーションが使用するブローカーのタイプです。

プロパティの有効値は以下のとおりです。

有効値

意味

XMSC_WMQ_BROKER_V1

アプリケーションは、IBM MQ パブリッシュ/サブスクライブのブローカーを使用しています。

IBM MQ パブリッシュ/サブスクライブから WebSphere Message Broker にマイグレーションしたが、アプリケーションを未変更だった場合にも、アプリケーションはこの値を使用できます。

XMSC_WMQ_BROKER_V2

アプリケーションは、IBM Integration Bus のブローカーを使用しています。

XMSC_WMQ_BROKER_UNSPECIFIED

ブローカーをマイグレーションした後で、RFH2 ヘッダーを使用できなくなるように、このプロパティを設定します。移行した後、このプロパティは関係なくなります。

接続ファクトリーのデフォルト値は XMSC_WMQ_BROKER_UNSPECIFIED ですが、デフォルトでは、このプロパティは宛先に設定されていません。このプロパティを宛先に設定すると、接続ファクトリーのプロパティによって指定された値は無効になります。

XMSC_WMQ_CCDTURL

データ型:

System.String

プロパティ:

ConnectionFactory

適用可能なオブジェクト:

JMS 管理ツールのロング・ネーム: CCDTURL

JMS 管理ツールのショート・ネーム: CCDT

クライアント・チャネル定義テーブルを含むファイルの名前および場所を識別すると同時に、そのファイルへのアクセス方法も指定する、URL (Uniform Resource Locator)。

デフォルトでは、このプロパティは設定されません。

XMSC_WMQ_CCSID**データ型:**

System.Int32

プロパティ:

Destination

URI で使用される名前:

CCSID

クライアントが XMS メッセージを宛先に転送する際、メッセージ本文内の文字データ文字列が属する符号化文字セット (CCSID) またはコードページ。個々のメッセージに対して設定した場合、このプロパティによって宛先に指定された CCSID は、JMS IBM CHARACTER SET プロパティによって無効になります。

このプロパティのデフォルト値は 1208 です。

このプロパティが関連するのは宛先に送信されたメッセージのみであり、宛先から受信したメッセージには関連しません。

XMSC_WMQ_CHANNEL**データ型:**

ストリング

プロパティ:

ConnectionFactory

適用可能なオブジェクト:

JMS 管理ツールのロング・ネーム: CHANNEL

JMS 管理ツールのショート・ネーム: CHAN

接続に使用するチャネルの名前です。

デフォルトでは、このプロパティは設定されていません。

アプリケーションがクライアント・モードでキュー・マネージャーに接続している場合のみ、このプロパティが関連します。

XMSC_WMQ_CLIENT_RECONNECT_OPTIONS**データ型:**

ストリング

プロパティ:

ConnectionFactory

適用可能なオブジェクト:

JMS 管理ツールのロング・ネーム: CLIENTRECONNECTOPTIONS

JMS 管理ツールのショート・ネーム: CROPT

このプロパティでは、このファクトリーで作成する新しい接続のクライアント再接続オプションを指定します。これは、XMSC 内にあり、以下のいずれかです。

- WMQ_CLIENT_RECONNECT_AS_DEF (デフォルト)。mqclient.ini ファイルに指定されている値を使用します。その値を設定するには、Channels スタンザにある **DefRecon** プロパティを使用します。以下のいずれかに設定できます。

1. はい。WMQ_CLIENT_RECONNECT オプションの動作になります。
 2. No. デフォルト。再接続オプションを指定しません。
 3. QMGR。WMQ_CLIENT_RECONNECT_Q_MGR オプションの動作になります。
 4. DISABLED。WMQ_CLIENT_RECONNECT_DISABLED オプションの動作になります。
- WMQ_CLIENT_RECONNECT。接続名リストで指定されているキュー・マネージャーのいずれかに再接続します。
 - WMQ_CLIENT_RECONNECT_Q_MGR。元の接続先と同じキュー・マネージャーに再接続します。接続しようとする対象のキュー・マネージャー (接続名リストで指定されているキュー・マネージャー) の QMID が元の接続先のキュー・マネージャーの QMID とは異なる場合は、MQRC_RECONNECT_QMID_MISMATCH を返します。
 - WMQ_CLIENT_RECONNECT_DISABLED。再接続が無効になります。

XMSC_WMQ_CLIENT_RECONNECT_TIMEOUT

データ型:

ストリング

プロパティ:

ConnectionFactory

適用可能なオブジェクト:

JMS 管理ツールのロング・ネーム: CLIENTRECONNECTTIMEOUT

JMS 管理ツールのショート・ネーム: CRT

XMSC_WMQ_CLIENT_RECONNECT_TIMEOUT プロパティが有効なのは、管理対象 XMS .NET クライアントに関してのみです。

このプロパティでは、クライアント接続が再接続を試みる期間を秒単位で指定します。

クライアントは、この期間、再接続を試みた後に、MQRC_RECONNECT_FAILED で失敗します。このプロパティのデフォルト設定は XMSC.WMQ_CLIENT_RECONNECT_TIMEOUT_DEFAULT です。

このプロパティのデフォルト値は 1800 です。

XMSC_WMQ_CONNECTION_MODE

データ型:

System.Int32

プロパティ:

ConnectionFactory

アプリケーションがキュー・マネージャーに接続する場合のモードです。

プロパティの有効値は以下のとおりです。

有効値	意味
XMSC_WMQ_CM_BINDINGS	パフォーマンス最適化のための、キュー・マネージャーへのバインディング・モードでの接続。この値は、C/C++ の場合のデフォルト値です。
XMSC_WMQ_CM_CLIENT	完全管理スタックを実現するための、キュー・マネージャーへのクライアント・モードでの接続。この値は、.NET の場合のデフォルト値です。
XMSC_WMQ_CM_CLIENT_UNMANAGED (.NET のみ)	強制的に非管理クライアント・スタックとなるキュー・マネージャーへの接続。

XMSC_WMQ_CONNECTION_NAME_LIST

データ型:

ストリング

プロパティ:

ConnectionFactory

適用可能なオブジェクト:

JMS 管理ツールのロング・ネーム: CONNECTIONNAMESLIST

JMS 管理ツールのショート・ネーム: CNLIST

このプロパティでは、接続で障害が発生した後にクライアントが再接続を試みる対象のホストを指定します。

接続名リストは、ホスト/IP ポートのペアのコンマ区切りリストです。このプロパティのデフォルト設定は WMQ_CONNECTION_NAME_LIST_DEFAULT です。

例えば、127.0.0.1(1414),host2.example.com(1400) のようになります。

このプロパティのデフォルト設定は localhost(1414) です。

XMSC_WMQ_DUR_SUBQ

データ型:

ストリング

プロパティ:

Destination

このプロパティを持つことができるのは、トピックに含まれる宛先のみです。宛先からメッセージを受信している永続サブスクライバー用のサブスクライバー・キューの名前です。

以下の文字で始まる必要があるサブスクライバー・キューの名前です。

SYSTEM.JMS.D.

すべての永続サブスクライバーがサブスクライバー・キューを共用するようにするには、共用キューの完全な名前を指定します。アプリケーションが永続サブスクライバーを作成できるようにするには、指定した名前のキューが事前に存在する必要があります。

各永続サブスクライバーが自身の排他的サブスクライバー・キューからメッセージを検索するようにするには、末尾にアスタリスク (*) を付けるキュー名を指定します。その後、アプリケーションが永続サブスクライバーを作成すると、XMS クライアントは、永続サブスクライバーが排他使用するための動的キューを作成します。XMS クライアントは、このプロパティの値を使用して、動的キューを作成するときに使用するオブジェクト記述子の **DynamicQName** フィールドの内容を設定します。

このプロパティのデフォルト値は SYSTEM.JMS.D.SUBSCRIBER.QUEUE で、これは、XMS は、デフォルトでは共用キューの方法を使用するという意味です。

このプロパティが関連するのは、パブリッシュ/サブスクライブ・ドメインの場合に限られます。

XMSC_WMQ_ENCODING

データ型:

System.Int32

プロパティ:

Destination

クライアントが XMS メッセージを宛先に転送する際、メッセージ本文内の数値データがどのように表現されるか。個々のメッセージに対して設定した場合、このプロパティによって宛先に指定されたエンコード方式は、JMS IBM_ENCODING プロパティによって無効になります。このプロパティは、2 進整数、パック 10 進数、および浮動小数点数の表記を指定します。

このプロパティの有効な値は、メッセージ記述子のフィールド **Encoding** で指定可能な値と同じです。

アプリケーションは、以下の名前付き定数を使用してプロパティを設定できます。

名前付き定数

MQENC_INTEGER_NORMAL

意味

標準の整数エンコード方式

名前付き定数	意味
MQENC_INTEGER_REVERSED	逆方向の整数エンコード方式
MQENC_DECIMAL_NORMAL	標準のパック 10 進エンコード方式
MQENC_DECIMAL_REVERSED	逆方向のパック 10 進エンコード方式
MQENC_FLOAT_IEEE_NORMAL	標準の IEEE 浮動小数点エンコード方式
MQENC_FLOAT_IEEE_REVERSED	逆方向の IEEE 浮動小数点エンコード方式
 MQENC_FLOAT_S390	z/OS アーキテクチャーの浮動小数点エンコード方式
MQENC_NATIVE	ネイティブのマシン・エンコード方式

プロパティの値を設定するために、アプリケーションは、以下に示す 3 つの定数を加算できます。

- 2 進整数の表記を指定するための、名前が MQENC_INTEGER で始まる定数
- パック 10 進整数の表記を指定するための、名前が MQENC_DECIMAL で始まる定数
- 浮動小数点数の表記を指定するための、名前が MQENC_FLOAT で始まる定数

あるいは、アプリケーションは、その値が環境に依存する MQENC_NATIVE にプロパティを設定できます。

このプロパティのデフォルト値は MQENC_NATIVE です。

このプロパティが関連するのは宛先に送信されたメッセージのみであり、宛先から受信したメッセージには関連しません。

XMSC_WMQ_FAIL_IF_QUIESCE

データ型:

System.Int32

プロパティ:

ConnectionFactory および Destination

URI で使用される名前:

failIfQuiesce

適用可能なオブジェクト:

JMS 管理ツールのロング・ネーム: FAILIFQUIESCE

JMS 管理ツールのショート・ネーム: FIQ

アプリケーションの接続先のキュー・マネージャーが静止状態である場合、特定のメソッドの呼び出しが失敗するかどうかを表します。

プロパティの有効値は以下のとおりです。

有効値	意味
XMSC_WMQ_FIQ_YES	キュー・マネージャーが静止状態である場合、特定のメソッドの呼び出しは失敗します。キュー・マネージャーが静止していることをアプリケーションが検出した場合、アプリケーションはその即時タスクを完了して接続を終了し、キュー・マネージャーを停止できます。
XMSC_WMQ_FIQ_NO	キュー・マネージャーは静止状態であるため、メソッドの呼び出しは失敗しません。この値を指定すると、アプリケーションはキュー・マネージャーが静止していることを検出できません。アプリケーションはキュー・マネージャーに対する操作を続行する可能性があるため、キュー・マネージャーの停止を防止する場合があります。

接続ファクトリーのデフォルト値は XMSC_WMQ_FIQ_YES ですが、デフォルトでは、このプロパティは宛先に設定されていません。このプロパティを宛先に設定すると、接続ファクトリーのプロパティによって指定された値は無効になります。

XMSC_WMQ_MESSAGE_BODY

データ型:

System.Int32

プロパティ:

Destination

このプロパティは、アプリケーションが XMS メッセージ IBM MQ の MQRFH2 をメッセージのペイロードの一部（つまり、メッセージ本文の一部）として処理するかどうかを決定します。

注: メッセージを宛先に送信すると、XMSC_WMQ_MESSAGE_BODY プロパティは、既存の XMS Destination プロパティ XMSC_WMQ_TARGET_CLIENT を置き換えます。

このプロパティの有効な値は以下のとおりです。

XMSC_WMQ_MESSAGE_BODY_JMS

Receive: インバウンド XMS メッセージのタイプと本文は、受信した IBM MQ メッセージ内の MQRFH2 (存在する場合) または MQMD (MQRFH2 がいない場合) の内容によって決定されます。

Send: アウトバウンド XMS メッセージの本文には、XMS メッセージのプロパティとヘッダー・フィールドに基づいて、前に付加され、自動生成された MQRFH2 ヘッダーが含まれています。

XMSC_WMQ_MESSAGE_BODY_MQ

受信する: 受信: 受信した XMS メッセージのタイプは、受信した IBM MQ メッセージの内容や受信した MQMD のフォーマットフィールドに関係なく、常に ByteMessage, である。XMS メッセージの本文は、下位層のメッセージング・プロバイダー API 呼び出しによって戻される、未変更のメッセージ・データです。メッセージ本文内のデータの文字セットとエンコードは、MQMD の CodedCharSetId フィールドと Encoding フィールドによって決定されます。メッセージ本文内のデータの形式は、MQMD の Format フィールドによって決定されます。

Send: アウトバウンド XMS メッセージの本文には、アプリケーション・ペイロードがそのまま含まれています。自動生成された IBM MQ ヘッダーは本文に追加されません。

XMSC_WMQ_MESSAGE_BODY_UNSPECIFIED

Receive: XMS クライアントは、このプロパティに適した値を決定します。受信パスの場合、この値は、WMQ_MESSAGE_BODY_JMS プロパティの値になります。

Send: XMS クライアントは、このプロパティに適した値を決定します。送信パスの場合、この値は、XMSC_WMQ_TARGET_CLIENT プロパティの値になります。

デフォルトでは、このプロパティは XMSC_WMQ_MESSAGE_BODY_UNSPECIFIED に設定されます。

XMSC_WMQ_MQMD_MESSAGE_CONTEXT

データ型:

System.Int32

プロパティ:

Destination

アプリケーション XMS によって設定されるメッセージコンテキストのレベルを決定します。このプロパティが有効となるためには、アプリケーションが適切なコンテキスト権限を持って実行されていなければなりません。

このプロパティの有効な値は以下のとおりです。

XMSC_WMQ_MDCTX_DEFAULT

アウトバウンド・メッセージでは、MQOPEN API 呼び出しと MQPMO 構造は、明示的なメッセージ・コンテキスト・オプションを指定しません。

XMSC_WMQ_MDCTX_SET_IDENTITY_CONTEXT

MQOPEN API 呼び出しは、メッセージ・コンテキスト・オプション MQOO_SET_IDENTITY_CONTEXT を指定し、MQPMO 構造は MQPMO_SET_IDENTITY_CONTEXT を指定します。

XMSC_WMQ_MDCTX_SET_ALL_CONTEXT

MQOPEN API 呼び出しはメッセージ・コンテキスト・オプション MQOO_SET_ALL_CONTEXT を指定し、MQPMO 構造は MQPMO_SET_ALL_CONTEXT を指定します。

デフォルトでは、このプロパティは XMSC_WMQ_MDCTX_DEFAULT に設定されます。

注: アプリケーションが WebSphere Application Server service integration bus に接続している場合、このプロパティは関係ありません。

望ましい効果を得るには、以下のプロパティでは、メッセージの送信時に、
XMSC_WMQ_MQMD_MESSAGE_CONTEXT プロパティが
XMSC_WMQ_MDCTX_SET_IDENTITY_CONTEXT プロパティ値または
XMSC_WMQ_MDCTX_SET_ALL_CONTEXT プロパティ値に設定されている必要があります。

- JMS_IBM_MQMD_USERIDENTIFIER
- JMS_IBM_MQMD_ACCOUNTINGTOKEN
- JMS_IBM_MQMD_APPLIDENTITYDATA

望ましい効果を得るには、以下のプロパティでは、メッセージの送信時に、
XMSC_WMQ_MQMD_MESSAGE_CONTEXT プロパティが XMSC_WMQ_MDCTX_SET_ALL_CONTEXT プロパティ値に設定されている必要があります。

- JMS_IBM_MQMD_PUTAPPLTYPE
- JMS_IBM_MQMD_PUTAPPLNAME
- JMS_IBM_MQMD_PUTDATE
- JMS_IBM_MQMD_PUTTIME
- JMS_IBM_MQMD_APPLORIGINDATA

XMSC_WMQ_MQMD_READ_ENABLED

データ型:

System.Int32

プロパティ:

Destination

このプロパティは、アプリケーションが XMS MQMD フィールドの値を抽出できるかどうかを決定します。

このプロパティの有効な値は以下のとおりです。

XMSC_WMQ_READ_ENABLED_NO

メッセージの送信時に、送信されたメッセージの JMS_IBM_MQMD* プロパティは、MQMD での更新済みのフィールド値を反映するように更新されません。

送信側が JMS_IBM_MQMD* プロパティの一部またはすべてを設定しておいた場合でも、メッセージの受信時には、受信したメッセージでこれらのプロパティはいずれも有効になりません。

XMSC_WMQ_READ_ENABLED_YES

メッセージの送信時に、送信したメッセージの JMS_IBM_MQMD* プロパティはすべて、MQMD の更新後のフィールド値に合わせて更新されます (送信側が明示的に設定しなかったプロパティも含めてそのようになります)。

メッセージの受信時には、受信したメッセージですべての JMS_IBM_MQMD* プロパティが有効になります (送信側が明示的に設定しなかったプロパティも含めてそのようになります)。

デフォルトでは、このプロパティは XMSC_WMQ_READ_ENABLED_NO に設定されます。

XMSC_WMQ_MQMD_WRITE_ENABLED

データ型:

System.Int32

プロパティ:

Destination

このプロパティは、アプリケーションがXMS MQMD フィールドの値を設定できるかどうかを決定します。
このプロパティの有効な値は以下のとおりです。

XMSC_WMQ_WRITE_ENABLED_NO

JMS_IBM_MQMD* プロパティはすべて無視され、その値は基礎となる MQMD 構造にコピーされません。

XMSC_WMQ_WRITE_ENABLED_YES

JMS_IBM_MQMD* プロパティは処理されます。それらの値は基礎となる MQMD 構造にコピーされます。

デフォルトでは、このプロパティはXMSC_WMQ_WRITE_ENABLED_NOに設定されます。

XMSC_WMQ_PUT_ASYNC_ALLOWED

データ型:

System.Int32

プロパティ:

Destination

このプロパティは、メッセージ・プロデューサーが、非同期書き込みを使用して、この宛先にメッセージを送信できるかどうかを決定します。

このプロパティの有効な値は以下のとおりです。

XMSC_WMQ_PUT_ASYNC_ALLOWED_AS_DEST

キュー定義またはトピック定義を参照することによって、非同期書き込みが許可されるかどうかを決定します。

XMSC_WMQ_PUT_ASYNC_ALLOWED_AS_Q_DEF

キュー定義を参照することによって非同期書き込みが許可されるかどうかを判別します。

XMSC_WMQ_PUT_ASYNC_ALLOWED_AS_TOPIC_DEF

トピック定義を参照することによって非同期書き込みが許可されるかどうかを判別します。

XMSC_WMQ_PUT_ASYNC_ALLOWED_DISABLED

非同期書き込みは許可されない。

XMSC_WMQ_PUT_ASYNC_ALLOWED_ENABLED

非同期書き込みは許可される。

デフォルトでは、このプロパティはXMSC_WMQ_PUT_ASYNC_ALLOWED_AS_DESTに設定されます。

注: アプリケーションが WebSphere Application Server service integration bus に接続している場合、このプロパティは関係ありません。

XMSC_WMQ_READ_AHEAD_ALLOWED

データ型:

System.Int32

プロパティ:

Destination

このプロパティは、メッセージ・コンシューマーとキュー・ブラウザーが、先行読み取りを使用して、非永続の非トランザクション・メッセージを受信する前に、この宛先から内部バッファにこれらのメッセージを取得できるかどうかを決定します。

このプロパティの有効な値は以下のとおりです。

XMSC_WMQ_READ_AHEAD_ALLOWED_AS_Q_DEF

キュー定義を参照することによって、先行読み取りが許可されるかどうかを決定します。

XMSC_WMQ_READ_AHEAD_ALLOWED_AS_TOPIC_DEF

トピック定義を参照することによって、先行読み取りが許可されるかどうかを決定します。

XMSC_WMQ_READ_AHEAD_ALLOWED_AS_DEST

キュー定義またはトピック定義を参照することによって、先行読み取りが許可されるかどうかを決定します。

XMSC_WMQ_READ_AHEAD_ALLOWED_DISABLED

メッセージのコンシュームまたは参照時には先行読み取りは許可されません。

XMSC_WMQ_READ_AHEAD_ALLOWED_ENABLED

先読みは許可される。

デフォルトでは、このプロパティは `XMSC_WMQ_READ_AHEAD_ALLOWED_AS_DEST` に設定されます。

XMSC_WMQ_READ_AHEAD_CLOSE_POLICY

データ型:

`System.Int32`

プロパティ:

`Destination`

このプロパティは、非同期メッセージ・リスナーに配信されているメッセージについて、メッセージ・コンシューマーがクローズされたときに、内部先行読み取りバッファ内のメッセージがどうなるかを決定します。

このプロパティは、宛先からメッセージをコンシュームするときにキューのクローズ・オプションを指定する際に適用され、宛先にメッセージを送信する際は適用されません。

参照中にメッセージはまだキューで使用できるため、このプロパティはキュー・ブラウザーでは無視されます。

このプロパティの有効な値は以下のとおりです。

XMSC_WMQ_READ_AHEAD_CLOSE_POLICY_DELIVER_CURRENT

現行のメッセージ・リスナーの呼び出しのみが完了して戻ります。この場合、内部先行読み取りバッファにメッセージが残る可能性があり、それらは廃棄されます。

XMSC_WMQ_READ_AHEAD_CLOSE_POLICY_DELIVER_ALL

内部先行読み取りバッファ内のメッセージがすべてアプリケーションのメッセージ・リスナーに配信されてから戻ります。

デフォルトでは、このプロパティは `XMSC_WMQ_READ_AHEAD_CLOSE_POLICY_DELIVER_CURRENT` に設定されます。

注:

アプリケーションの異常終了

XMS アプリケーションが不意に終了すると、先行読み取りバッファ内のメッセージはすべて失われます。

トランザクションの影響

アプリケーションでトランザクションが使用される場合は、先行読み取りは使用不可にされます。そのため、アプリケーションでは、トランザクション化セッションを使用する場合は、動作の相違点は認識されません。

セッション肯定応答モードの影響

肯定応答モードが `XMSC_AUTO_ACKNOWLEDGE` または `XMSC_DUPS_OK_ACKNOWLEDGE` になっていると、非トランザクション化セッションで先行読み取りが有効になります。トランザクション化セッションか未トランザクション化セッションかに関係なく、セッション肯定応答モードが `XMSC_CLIENT_ACKNOWLEDGE` の場合は、先行読み取りは使用不可になります。

キュー・ブラウザーとキュー・ブラウザー・セレクトアーの影響

XMS アプリケーションで使用されるキュー・ブラウザーとキュー・ブラウザー・セレクトアーでは、先行読み取りによりパフォーマンスが向上します。キュー・ブラウザーを閉じて、パフォーマンスが低下することはありません。その後の操作でも、引き続きキューのメッセージを使用できるからです。先行読み取りによってパフォーマンスが向上するという以外に、キュー・ブラウザーとキュー・ブラウザー・セレクトアーに対する影響はありません。

XMSC_WMQ_HOST_NAME

データ型:

ストリング

プロパティ:

ConnectionFactory

適用可能なオブジェクト:

JMS 管理ツールのロング・ネーム: HOSTNAME

JMS 管理ツールのショート・ネーム: HOST

キュー・マネージャーを実行するシステムのホスト名または IP アドレスです。

アプリケーションがクライアント・モードでキュー・マネージャーに接続している場合のみ、このプロパティが使用されます。このプロパティは、キュー・マネージャーを識別するために、XMSC_WMQ_PORT プロパティと一緒に使用します。

このプロパティのデフォルト値は localhost です。

XMSC_WMQ_LOCAL_ADDRESS

データ型:

ストリング

プロパティ:

ConnectionFactory

適用可能なオブジェクト:

JMS 管理ツールのロング・ネーム: LOCALADDRESS

JMS 管理ツールのショート・ネーム: LA

キュー・マネージャーへの接続の場合、このプロパティは、使用するローカル・ネットワーク・インターフェース、または使用するローカル・ポート (1 つまたは一定範囲)、あるいはその両方を指定します。

このプロパティの値は、次の形式のストリングです。

[host_name][(low_port)[,high_port]]

変数の意味は以下のとおりです。

host_name

接続に使用するローカル・ネットワーク・インターフェースのホスト名または IP アドレスです。

この情報の入力が必要なのは、アプリケーションを実行しているシステムに複数のネットワーク・インターフェースがあり、接続にどのインターフェースを使用する必要があるかを指定することが必要な場合に限ります。システムが備えるネットワーク・インターフェースが 1 つのみである場合、使用できるのはそのインターフェースのみです。システムに複数のネットワーク・インターフェースがあり、どのインターフェースを使用するかを指定していない場合、インターフェースはランダムに選択されます。

low_port

接続に使用するローカル・ポートの数です。

high_port も指定した場合、low_port は一連のポート番号のうち最小のポート番号と解釈されます。

high_port

一連のポート番号のうち最大のポート番号を表します。指定した範囲内のいずれかのポートを接続に使用する必要があります。

このストリングの最大長は 48 文字です。

プロパティの有効値の例の一部を以下に示します。

```
JUPITER
9.20.4.98
JUPITER(1000)
9.20.4.98(1000,2000)
(1000)
(1000,2000)
```

デフォルトでは、このプロパティは設定されていません。

アプリケーションがクライアント・モードでキュー・マネージャーに接続している場合のみ、このプロパティが関連します。

XMSC_WMQ_MESSAGE_SELECTION

データ型:

System.Int32

プロパティ:

ConnectionFactory

メッセージの選択をクライアント XMS が行うか、ブローカーが行うかを決定します。

プロパティの有効値は以下のとおりです。

有効値	意味
XMSC_WMQ_MSEL_CLIENT	メッセージの選択は XMS クライアントによって行われます。
XMSC_WMQ_MSEL_BROKER	ブローカーがメッセージ選択を行う。

デフォルト値は XMSC_WMQ_MSEL_CLIENT です。

このプロパティが関連するのは、パブリッシュ/サブスクライブ・ドメインの場合に限られます。

XMSC_WMQ_BROKER_VERSION プロパティを XMSC_BROKER_V1 に設定している場合、ブローカーによるメッセージの選択はサポートされません。

XMSC_WMQ_MSG_BATCH_SIZE

データ型:

System.Int32

プロパティ:

ConnectionFactory

非同期のメッセージ配信機能を使用する場合に 1 バッチ内のキューから取り出すメッセージの最大数を表します。

アプリケーションが特定の条件下で非同期のメッセージ配信機能を使用している場合、XMS クライアントは、1 バッチのメッセージをキューから取り出してから、各メッセージをアプリケーションに個別に転送します。このプロパティでは、バッチ内に収容できるメッセージの最大数が指定されます。

このプロパティの値は正の整数で、デフォルト値は 10 です。このプロパティを他の値に設定することを検討するのは、対応しなければならないパフォーマンスに関する具体的な問題がある場合に限るようにしてください。

アプリケーションがネットワークを介してキュー・マネージャーに接続されている場合、このプロパティの値を大きくすると、ネットワークのオーバーヘッドを削減して応答時間を短縮できますが、クライアントのシステムでメッセージを保管するのに必要なメモリーの量は増加します。逆に、このプロパティの値を小さくすると、ネットワークのオーバーヘッドは増大し、応答時間は長くなりますが、メッセージを保管するのに必要なメモリーの量を削減できます。

XMSC_WMQ_POLLING_INTERVAL

データ型:

System.Int32

プロパティ:

ConnectionFactory

セッション内の各メッセージ・リスナーのキューに適切なメッセージがない場合、この値は、各メッセージ・リスナーがそのキューから再度メッセージを読み取ろうとするまでに経過する最大の時間間隔(ミリ秒)になります。

セッション内のいずれのメッセージ・リスナーでも適切なメッセージがない状態が頻繁に発生する場合は、このプロパティの値を大きくすることを考えてください。

このプロパティの値は正の整数です。デフォルト値は 5000 です。

XMSC_WMQ_PORT

データ型:

System.Int32

プロパティ:

ConnectionFactory

適用可能なオブジェクト:

JMS 管理ツールのロング・ネーム: PORT

JMS 管理ツールのショート・ネーム: PORT

キュー・マネージャーが着信要求を listen するポートの数です。

アプリケーションがクライアント・モードでキュー・マネージャーに接続している場合のみ、このプロパティが使用されます。このプロパティは、キュー・マネージャーを識別するために、[XMSC_WMQ_HOST_NAME](#) プロパティと一緒に使用します。

プロパティのデフォルト値は XMSC_WMQ_DEFAULT_CLIENT_PORT、または 1414 です。

XMSC_WMQ_PROVIDER_VERSION

データ型:

ストリング

プロパティ:

ConnectionFactory

アプリケーションの接続先のキュー・マネージャーのバージョン、リリース、モディフィケーション・レベル、およびフィックスパック。このプロパティの有効な値は以下のとおりです。

- 指定なし

または、以下のいずれかの形式のストリング

- V.R.M.F
- V.R.M
- V.R
- V

ここで、V、R、M、および F は、ゼロ以上の整数値です。

デフォルトでは、このプロパティは「UNSPECIFIED」に設定されています。

IBM MQ クライアントのバージョンは、XMS クライアント・アプリケーションで IBM MQ の固有の機能を使用できるかどうかに関して重要な役割を果たします。

注: システム・プロパティ `XMSC_WMQ_OVERRIDEPROVIDERVERSION` は、`XMSC_WMQ_PROVIDER_VERSION` プロパティより優先されます。このプロパティは、接続ファクトリー設定を変更できない場合に使用できます。

XMSC_WMQ_PUB_ACK_INTERVAL

データ型:

System.Int32

プロパティ:

ConnectionFactory

クライアントが XMS ブローカーに確認応答を要求する前に、パブリッシャーが発行するメッセージの数。

このプロパティの値を小さくすると、クライアントによる肯定応答の要求頻度が高くなるため、パブリッシャーのパフォーマンスは低下します。この値を大きくすると、ブローカーに障害が発生した場合にクライアントが例外をスローするのに要する時間が長くなります。

このプロパティの値は正の整数です。デフォルト値は 25 です。

XMSC_WMQ_QMGR_CCSID

データ型:

System.Int32

プロパティ:

ConnectionFactory

メッセージ・キュー・インターフェース (MQI) で定義された文字データフィールドが、クライアント IBM MQXMS 間で交換される際に使用される、符号化文字セットまたはコードページの識別子 (CCSID)。このプロパティは、メッセージの本文にある文字データのストリングには適用されません。

XMS アプリケーションがキュー・マネージャーにクライアント・モードで接続すると、XMS クライアントは IBM MQ クライアントにリンクします。この 2 つのクライアント間で交換される情報には、MQI で定義された文字データのフィールドが含まれます。通常の場合は、IBM MQ クライアントは、そのクライアントが動作しているシステムのコード・ページでこれらのフィールドが記述されていることを前提にしています。XMS クライアントが異なるコード・ページでこれらのフィールドを提供する場合や、これらのフィールドを異なるコード・ページで受信することが予想される場合は、このプロパティを設定して IBM MQ クライアントに通知する必要があります。

IBM MQ クライアントがこれらの文字データ・フィールドをキュー・マネージャーに転送した場合は、必要に応じて、これらのフィールド内のデータを、キュー・マネージャーが使用するコード・ページに変換する必要があります。同様に、IBM MQ クライアントがこれらのフィールドをキュー・マネージャーから受信した場合は、必要に応じて、これらのフィールド内のデータを、XMS クライアントがデータを受信する場合に想定しているコード・ページに変換する必要があります。IBM MQ クライアントは、このプロパティを使用してこれらのデータ変換を実行します。

デフォルトでは、このプロパティは設定されていません。

このプロパティを設定することは、ネイティブの IBM MQ クライアント・アプリケーションをサポートしている IBM MQ クライアントに対して MQCCSID 環境変数を設定することと同等です。この環境変数に関して詳しくは、[MQCCSID](#) を参照してください。

XMSC_WMQ_QUEUE_MANAGER

データ型:

ストリング

プロパティ:

ConnectionFactory

適用可能なオブジェクト:

JMS 管理ツールのロング・ネーム: QMANAGER

JMS 管理ツールのショート・ネーム: QMGR

接続先となるキュー・マネージャーの名前です。

デフォルトでは、このプロパティは設定されていません。

XMSC_WMQ_RECEIVE_CCSID

キュー・マネージャー・メッセージ変換のターゲット CCSID を設定する宛先プロパティ。XMSC_WMQ_RECEIVE_CONVERSION が WMQ_RECEIVE_CONVERSION_QMGR に設定されていなければ、この値は無視されます。

データ型:

整数

値:

任意の正整数。

デフォルト値は 1208 です。

メッセージで GMO_CONVERT の値を指定するかどうかは任意です。GMO_CONVERT の値を指定する場合は、その値に基づいて変換が実行されます。

XMSC_WMQ_RECEIVE_CONVERSION

キュー・マネージャーによりデータ変換を実行するかどうかを決定する宛先プロパティ。

データ型:

整数

値:

XMSC_WMQ_RECEIVE_CONVERSION_CLIENT_MSG (デフォルト): XMS クライアントでのみデータ変換を実行します。変換では常にコード・ページ 1208 を使用します。

XMSC_WMQ_RECEIVE_CONVERSION_QMGR: XMS クライアントにメッセージを送信する前にキュー・マネージャーでデータ変換を実行します。

XMSC_WMQ_RECEIVE_EXIT

データ型:

ストリング

プロパティ:

ConnectionFactory

実行するチャンネル受信出口を示します。

このプロパティの値は、チャンネル受信出口を示すストリングで、その形式は次のとおりです。

libraryName(entryPointName)

ここで、

- **libraryName** は、管理出口 .dll の絶対パスです。
- **entryPointName** は、名前空間で修飾されるクラス名です。

例えば、C:\MyReceiveExit.dll(MyReceiveExitNameSpace.MyReceiveExitClassName)

デフォルトでは、このプロパティは設定されていません。

このプロパティが関連するのは、アプリケーションが管理クライアント・モードでキュー・マネージャーに接続している場合のみです。また、管理出口のみがサポートされます。

XMSC_WMQ_RECEIVE_EXIT_INIT

データ型:

ストリング

プロパティ:

ConnectionFactory

チャンネル受信出口が呼び出されたときにこの出口に渡されるユーザー・データです。

プロパティの値はストリングです。デフォルトでは、このプロパティは設定されていません。

このプロパティが関連するのは、アプリケーションが管理クライアント・モードでキュー・マネージャーに接続し、[2087 ページの『XMSC_WMQ_RECEIVE_EXIT』](#) プロパティが設定されている場合のみです。

XMSC_WMQ_RESOLVED_QUEUE_MANAGER

データ型:

ストリング

プロパティ:

ConnectionFactory

このプロパティは、接続先のキュー・マネージャーの名前を取得するために使用します。

CCDT (クライアント・チャネル定義テーブル) と一緒に使用する場合、この名前は、接続ファクトリーで指定されているキュー・マネージャー名とは異なる可能性があります。

XMSC_WMQ_RESOLVED_QUEUE_MANAGER_ID

データ型:

ストリング

プロパティ:

ConnectionFactory

接続後に、このプロパティにはキュー・マネージャーの ID が設定されます。

XMSC_WMQ_SECURITY_EXIT

データ型:

ストリング

プロパティ:

ConnectionFactory

チャネル・セキュリティ出口を示します。

このプロパティの値はチャネル・セキュリティ出口を示すストリングで、その形式は次のとおりです。

libraryName(entryPointName)

ここで、

- **libraryName** は、管理出口 .dll の絶対パスです。
- **entryPointName** は、名前空間で修飾されるクラス名です。

例えば、C:\MySecurityExit.dll(MySecurityExitNamespace.MySecurityExitClassName) となります。

ストリングの最大長は 128 文字です。

デフォルトでは、このプロパティは設定されていません。

このプロパティが関連するのは、アプリケーションが管理クライアント・モードでキュー・マネージャーに接続している場合のみです。また、管理出口のみがサポートされます。

XMSC_WMQ_SECURITY_EXIT_INIT

データ型:

ストリング

プロパティ:

ConnectionFactory

チャネル・セキュリティ出口を呼び出した場合にこの出口に渡されるユーザー・データです。

ユーザー・データのストリングの最大長は 32 文字です。

デフォルトでは、このプロパティは設定されていません。

このプロパティが関連するのは、アプリケーションが管理クライアント・モードでキュー・マネージャーに接続し、[2088 ページの『XMSC_WMQ_SECURITY_EXIT』](#) プロパティが設定されている場合のみです。

XMSC_WMQ_SEND_EXIT

データ型:

ストリング

プロパティ:

ConnectionFactory

チャンネル送信出口を示します。

プロパティの値はストリングです。チャンネル送信出口の形式は次のとおりです。

libraryName(entryPointName)

ここで、

- **libraryName** は、管理出口 .dll の絶対パスです。
- **entryPointName** は、名前空間で修飾されるクラス名です。

例えば、 C:\MySendExit.dll(MySendExitNameSpace.MySendExitClassName)

デフォルトでは、このプロパティは設定されていません。

このプロパティが関連するのは、アプリケーションが管理クライアント・モードでキュー・マネージャーに接続している場合のみです。また、管理出口のみがサポートされます。

XMSC_WMQ_SEND_EXIT_INIT

データ型:

ストリング

プロパティ:

ConnectionFactory

複数のチャンネル送信出口を呼び出した場合にこれらの出口に渡されるユーザー・データです。

このプロパティの値は、コンマで区切られた 1 つ以上のユーザー・データ項目から成るストリングです。デフォルトでは、このプロパティは設定されていません。

チャンネル送信出口のシーケンスに渡されるユーザー・データを指定するための規則は、チャンネル受信出口のシーケンスに渡されるユーザー・データを指定するための規則と同じです。したがって、この規則については、[2087 ページの『XMSC_WMQ_RECEIVE_EXIT_INIT』](#)を参照してください。

このプロパティが関連するのは、アプリケーションが管理クライアント・モードでキュー・マネージャーに接続し、[2089 ページの『XMSC_WMQ_SEND_EXIT』](#) プロパティが設定されている場合のみです。

XMSC_WMQ_SEND_CHECK_COUNT

データ型:

System.Int32

プロパティ:

ConnectionFactory

単一の未処理の XMS セッション内での、非同期書き込みエラーの検査の間に許可する Send 呼び出しの数。

デフォルトでは、このプロパティは 0 に設定されます。

XMSC_WMQ_SHARE_CONV_ALLOWED

データ型:

System.Int32

プロパティ:

ConnectionFactory

適用可能なオブジェクト:

JMS 管理ツールのロング・ネーム: SHARECONVALLOWED

JMS 管理ツールのショート・ネーム: SCALD

チャンネル定義が一致する場合、クライアント接続が、同じプロセスから同じキューマネージャーへの他のトップ XMS レベル接続とソケットを共有できるかどうか。このプロパティは、アプリケーション開発、保守、または操作に必要となる場合に、別のソケットの接続を完全に分離できるようにするために提供されています。このプロパティを設定すると、基礎となるソケットを共用するように XMS に指示のみをします。単一のソケットを共用する接続の数は示されません。ソケットを共用している接続の数は、IBM MQ クライアントと IBM MQ サーバー間でネゴシエーションされる SHARECNV 値によって決まります。

アプリケーションは、以下の名前付き定数を設定して、プロパティを設定できます。

- XMSC_WMQ_SHARE_CONV_ALLOWED_FALSE - 複数の接続間でソケットは共用されません。
- XMSC_WMQ_SHARE_CONV_ALLOWED_TRUE - 複数の接続間でソケットは共用されます。

デフォルトでは、このプロパティは XMSC_WMQ_SHARE_CONV_ALLOWED_ENABLED に設定されます。

アプリケーションがクライアント・モードでキュー・マネージャーに接続している場合のみ、このプロパティが関連します。

XMSC_WMQ_SSL_CERT_STORES

データ型:

ストリング

プロパティ:

ConnectionFactory

キュー・マネージャーとの SSL 接続で使用される証明書取り消しリスト (CRL) を保持するサーバーの位置。このプロパティの値は、1 つ以上の URL をコンマで区切ったリストです。各 URL の形式は次のとおりです。

```
[user[/password]@]ldap://[serveraddress][:portnum][,...]
```

この形式は、基本的な MQJMS 形式と互換性がありますが、一部拡張されています。

serveraddress の部分を空にしても有効です。その場合、XMS は、その値が「localhost」というストリングであるとみなします。

リストの例を以下に示します。

```
myuser/mypassword@ldap://server1.mycom.com:389
ldap://server1.mycom.com
ldap://
ldap://:389
```

以下の .NET 接続のみ：管理対象接続（IBM MQ WMQ_CM_CLIENT）と非管理対象接続（IBM MQ WMQ_CM_CLIENT_UNMANAGED）の両方が、TLS / SSL 接続をサポートします。

デフォルトでは、このプロパティは設定されていません。

関連概念

[非管理 .NET クライアントの SSL および TLS サポート](#)

[管理 .NET クライアントの SSL および TLS サポート](#)

XMSC_WMQ_SSL_CIPHER_SPEC

データ型:

ストリング

プロパティ:

ConnectionFactory

キュー・マネージャーとのセキュア接続で使用する CipherSpec の名前。

次の表に、IBM MQ TLS サポートとともに使用できる暗号仕様をリストします。個人用証明書を要求するときに、公開鍵と秘密鍵のペアの鍵サイズを指定します。SSL ハンドシェイク時に使用される鍵のサイズは、表の注記のとおり、CipherSpec によって決定されている場合を除き、証明書に保管されているサイズです。デフォルトでは、このプロパティは設定されません。

CipherSpec 名	使用される プロトコル	ハッシュ・アル ゴリズム	暗号化 アルゴ リズム	暗号化 ビット 数	FIPS ¹	Suite B 128 ビ ット	Suite B 192 ビ ット
TLS_RSA_WITH_AES_128_CBC_SHA	TLS 1.0	SHA-1	AES	128	はい	いいえ	いいえ
TLS_RSA_WITH_AES_256_CBC_SHA ²	TLS 1.0	SHA-1	AES	256	はい	いいえ	いいえ
TLS_RSA_WITH_DES_CBC_SHA	TLS 1.0	SHA-1	DES	56	いいえ	いいえ	いいえ
TLS_RSA_WITH_3DES_EDE_CBC_SHA ⁴	TLS 1.0	SHA-1	3DES	168	はい	いいえ	いいえ
TLS_RSA_WITH_AES_128_GCM_SHA256	TLS 1.2	SHA-256	AES	128	はい	いいえ	いいえ
TLS_RSA_WITH_AES_256_GCM_SHA384	TLS 1.2	SHA-384	AES	256	はい	いいえ	いいえ
TLS_RSA_WITH_AES_128_CBC_SHA256	TLS 1.2	SHA-256	AES	128	はい	いいえ	いいえ
TLS_RSA_WITH_AES_256_CBC_SHA256	TLS 1.2	SHA-256	AES	256	はい	いいえ	いいえ
ECDHE_ECDSA_RC4_128_SHA256	TLS 1.2	SHA-256	RC4	128	いいえ	いいえ	いいえ
ECDHE_ECDSA_3DES_EDE_CBC_SHA256	TLS 1.2	SHA-256	3DES	168	はい	いいえ	いいえ
ECDHE_RSA_RC4_128_SHA256	TLS 1.2	SHA-256	RC4	128	いいえ	いいえ	いいえ
ECDHE_RSA_3DES_EDE_CBC_SHA256	TLS 1.2	SHA-256	3DES	168	はい	いいえ	いいえ
ECDHE_ECDSA_AES_128_CBC_SHA256	TLS 1.2	SHA-256	AES	128	はい	いいえ	いいえ
ECDHE_ECDSA_AES_256_CBC_SHA384	TLS 1.2	SHA-384	AES	256	はい	いいえ	いいえ
ECDHE_RSA_AES_128_CBC_SHA256	TLS 1.2	SHA-256	AES	128	はい	いいえ	いいえ
ECDHE_RSA_AES_256_CBC_SHA384	TLS 1.2	SHA-384	AES	256	はい	いいえ	いいえ
ECDHE_ECDSA_AES_128_GCM_SHA256	TLS 1.2	SHA-256	AES	128	はい	はい	いいえ
ECDHE_ECDSA_AES_256_GCM_SHA384	TLS 1.2	SHA-384	AES	256	はい	いいえ	はい

CipherSpec 名	使用されるプロトコル	ハッシュ・アルゴリズム	暗号化アルゴリズム	暗号化ビット数	FIPS ¹	Suite B 128 ビット	Suite B 192 ビット
ECDHE_RSA_AES_128_GCM_SHA256	TLS 1.2	SHA-256	AES	128	はい	いいえ	いいえ
ECDHE_RSA_AES_256_GCM_SHA384	TLS 1.2	SHA-384	AES	256	はい	いいえ	いいえ
TLS_RSA_WITH_NULL_SHA256	TLS 1.2	SHA-256	なし	0	いいえ	いいえ	いいえ
ECDHE_RSA_NULL_SHA256	TLS 1.2	SHA-256	なし	0	いいえ	いいえ	いいえ
ECDHE_ECDSA_NULL_SHA256	TLS 1.2	SHA-256	なし	0	いいえ	いいえ	いいえ
TLS_RSA_WITH_NULL_NULL	TLS 1.2	なし	なし	0	いいえ	いいえ	いいえ
TLS_RSA_WITH_RC4_128_SHA256	TLS 1.2	SHA-256	RC4	128	いいえ	いいえ	いいえ

注:

1. CipherSpec が(FIPS) Federal Information Processing Standards 140-2 に準拠するかどうかを指定する。および FIPS の動作に関する説明と、準拠した FIPS 140-2 動作のための IBM MQ 設定方法についてはを参照してください [Federal Information Processing Standards \(FIPS\)](#)。
2. この CipherSpec は、適切な無制限ポリシーファイルが JRE によって使用されるに適用されていない限り、から IBM MQ Explorer キューマネージャ IBM MQ Explorer への接続を保護するために使用できません。
3. この CipherSpec は、2007 年 5 月 19 日より前に認証 FIPS 140-2 されました。
4. CipherSpecIBM MQFIPS 140-2 が準拠動作用に設定されている場合、接続がエラー AMQ9288 で終了する前に最大 32GB のデータを転送できます。このエラーを回避するには、トリプル DES（非推奨）の使用を避けるか、設定 FIPS 140-2 でこの CipherSpec を使用する際には秘密鍵リセットを有効にしてください。

関連概念

[メッセージのデータ保全性](#)

関連タスク

[セキュリティ](#)

[CipherSpec の指定](#)

XMSC_WMQ_SSL_CIPHER_SUITE

データ型:

ストリング

プロパティ:

ConnectionFactory

キュー・マネージャーとの TLS 接続で使用される CipherSuite の名前。セキュア接続のネゴシエーションで使用されるプロトコルは、指定されている CipherSuite によって異なります。

このプロパティの標準値は以下のとおりです。

- SSL_RSA_WITH_DES_CBC_SHA
- SSL_RSA_EXPORT1024_WITH_DES_CBC_SHA
- SSL_RSA_EXPORT1024_WITH_RC4_56_SHA
- SSL_RSA_EXPORT_WITH_RC4_40_MD5

- `SSL_RSA_WITH_RC4_128_MD5`
- `SSL_RSA_WITH_RC4_128_SHA`
- `SSL_RSA_WITH_3DES_EDE_CBC_SHA`
- `SSL_RSA_WITH_AES_128_CBC_SHA`
- `SSL_RSA_WITH_AES_256_CBC_SHA`
- `SSL_RSA_WITH_DES_CBC_SHA`
- `SSL_RSA_WITH_3DES_EDE_CBC_SHA`

この値は、`XMSC_WMQ_SSL_CIPHER_SPEC` に代わるものとして指定できます。

`XMSC_WMQ_SSL_CIPHER_SPEC` に空でない値が指定されている場合、`XMSC_WMQ_SSL_CIPHER_SUITE` の設定値はこの値によってオーバーライドされます。`XMSC_WMQ_SSL_CIPHER_SPEC` に値が設定されていない場合、`XMSC_WMQ_SSL_CIPHER_SUITE` の値が暗号スイートとして渡されます。IBM Global Security Kit (GSKit) の場合、値はキュー IBM MQ ・マネージャへの接続 XMS における名前マッピング (`CipherSuite` および `CipherSpec` を参照) で説明されているように、対応する `CipherSpec` 値にマッピングされます。

`XMSC_WMQ_SSL_CIPHER_SPEC` と `XMSC_WMQ_SSL_CIPHER_SUITE` のどちらも空の場合、フィールド `pChDef->SSLCipherSpec` にはスペースが入ります。

以下の .NET 接続のみ：管理対象接続 (IBM MQ `WMQ_CM_CLIENT`) と非管理対象接続 (IBM MQ `WMQ_CM_CLIENT_UNMANAGED`) の両方が、TLS / SSL 接続をサポートします。

デフォルトでは、このプロパティは設定されていません。

関連概念

[非管理 .NET クライアントの SSL および TLS サポート](#)

[管理 .NET クライアントの SSL および TLS サポート](#)

`XMSC_WMQ_SSL_CRYPTO_HW`

データ型:

ストリング

プロパティ:

`ConnectionFactory`

クライアント・システムに接続されている暗号ハードウェアに関する構成詳細情報。

このプロパティの標準値は以下のとおりです。

- `GSK_ACCELERATOR_RAINBOW_CS_OFF`
- `GSK_ACCELERATOR_RAINBOW_CS_ON`
- `GSK_ACCELERATOR_NCIPHER_NF_OFF`
- `GSK_ACCELERATOR_NCIPHER_NF_ON`

PKCS11 暗号ハードウェアについては特殊な形式があります (`DriverPath`、`TokenLabel`、`TokenPassword` はユーザー指定ストリング)。

```
GSK_PKCS11=PKCS#11 DriverPath; PKCS#11 TokenLabel;PKCS#11 TokenPassword
```

XMS は、ストリングの内容を解釈したり変更したりしません。指定された値のうち最大 256 個の 1 バイト文字に相当する部分を `MQSCO.CryptoHardware` フィールドにコピーします。

以下の .NET 接続のみ：管理対象接続 (IBM MQ `WMQ_CM_CLIENT`) と非管理対象接続 (IBM MQ `WMQ_CM_CLIENT_UNMANAGED`) の両方が、TLS / SSL 接続をサポートします。

デフォルトでは、このプロパティは設定されていません。

関連概念

[非管理 .NET クライアントの SSL および TLS サポート](#)

[管理 .NET クライアントの SSL および TLS サポート](#)

XMSC_WMQ_SSL_FIPS_REQUIRED

データ型:

Boolean

プロパティ:

ConnectionFactory

このプロパティの値によって、アプリケーションが非標準の FIPS 暗号スイートを使用できるかどうかが決まります。このプロパティが true に設定されている場合、クライアントとサーバー間の接続にはアルゴリズムのみ FIPS が使用されます。

このプロパティの値は次のとおりです。それらは、MQSCO.FipsRequired の 2 個の標準値に変換されます。

表 916. MQSCO.FlipsRequired プロパティの値の表		
値	説明	MQSCO.FipsRequired で対応する値
false	任意の CipherSpec を使用できます。	MQSSL_FIPS_NO (デフォルト)
true	このクライアント接続に適用される CipherSpec では、認証済み暗号 FIPS アルゴリズムのみを使用できます。	MQSSL_FIPS_YES

XMS は、MQCONN を呼び出す前に、関係する値を MQSCO.FipsRequired にコピーします。

以下の .NET 接続のみ：管理対象接続（IBM MQ WMQ_CM_CLIENT）と非管理対象接続（IBM MQ WMQ_CM_CLIENT_UNMANAGED）の両方が、TLS / SSL 接続をサポートします。

関連概念

[非管理 .NET クライアントの SSL および TLS サポート](#)

[管理 .NET クライアントの SSL および TLS サポート](#)

XMSC_WMQ_SSL_KEY_REPOSITORY

データ型:

ストリング

プロパティ:

ConnectionFactory

鍵および証明書が保存されている鍵データベース・ファイルの位置。

XMS は、ストリングのうち 256 個までの 1 バイト文字に相当する部分を MQSCO.KeyRepository フィールドにコピーします。IBM MQ はこのストリングを絶対パスを含むファイル名と解釈します。

以下の .NET 接続のみ：管理対象接続（IBM MQ WMQ_CM_CLIENT）と非管理対象接続（IBM MQ WMQ_CM_CLIENT_UNMANAGED）の両方が、TLS / SSL 接続をサポートします。

デフォルトでは、このプロパティは設定されていません。

[Video](#) から IBM MQ 10.0.0、TLS 証明書のカスタムパスを設定できます。これにより、柔軟な証明書ストレージ管理が可能となり、アプリケーション .NETXMS 向けにシステムキーストア外のパスを指定できます。これを行うには、パスワードの安全な暗号化と暗号化されたパスワードデータの管理をサポートする Dotnet Password Encryptor ([rundotnetcred](#))を使用できます。詳細については、[XMS の「証明書のカスタムパス」](#)を参照してください。 .NETTLS 接続におけるプレーンテキストまたは暗号化されたパスワードの使用について。

関連概念

[非管理 .NET クライアントの SSL および TLS サポート](#)

[管理 .NET クライアントの SSL および TLS サポート](#)

XMSC_WMQ_SSL_KEY_RESETCOUNT

データ型:

System.Int32

プロパティ:

ConnectionFactory

KeyResetCount は、秘密鍵の再ネゴシエーションが実行されるまで、1 つの SSL 会話の中で送受信される暗号化されていないデータの合計バイト数を表します。このバイト数には、MCA によって送信される制御情報が含まれます。

XMS は、MQCONN を呼び出す前に、このプロパティに指定された値を MQSCO.KeyResetCount にコピーします。

パラメーター MQSCO.KeyResetCount は、IBM MQ バージョン 6 からのみ使用可能です。IBM MQ バージョン 5.3 の場合、このプロパティが設定されていると、XMS はキュー・マネージャーへの接続を試行せず、代わりに適切な例外をスローします。

以下の .NET 接続のみ：管理対象接続（IBM MQ WMQ_CM_CLIENT）と非管理対象接続（IBM MQ WMQ_CM_CLIENT_UNMANAGED）の両方が、TLS / SSL 接続をサポートします。

このプロパティのデフォルト値はゼロです。これは、秘密鍵が再ネゴシエーションされることがないことを意味します。

関連概念

[非管理 .NET クライアントの SSL および TLS サポート](#)

[管理 .NET クライアントの SSL および TLS サポート](#)

XMSC_WMQ_SSL_PEER_NAME

データ型:

ストリング

プロパティ:

ConnectionFactory

キュー・マネージャーとの SSL 接続で使用されるピア名。

このプロパティの標準値のリストはありません。その代わりに、SSLPEER の規則に従って、このストリングを作成する必要があります。

ピア名の例を以下に示します。

```
"CN=John Smith, O=IBM ,OU=Test , C=GB"
```

XMS は、MQCONN を呼び出す前に、このストリングを正しい 1 バイト・コード・ページにコピーし、MQCD.SSLPeerNamePtr および MQCD.SSLPeerNameLength に正しい値を入れます。

アプリケーションがクライアント・モードでキュー・マネージャーに接続している場合のみ、このプロパティが関連します。

以下の .NET 接続のみ：管理対象接続（IBM MQ WMQ_CM_CLIENT）と非管理対象接続（IBM MQ WMQ_CM_CLIENT_UNMANAGED）の両方が、TLS / SSL 接続をサポートします。

デフォルトでは、このプロパティは設定されていません。

関連概念

[非管理 .NET クライアントの SSL および TLS サポート](#)

[管理 .NET クライアントの SSL および TLS サポート](#)

関連資料

[SSLPEERNAME](#)

XMSC_WMQ_SSL_キーリポジトリパスワード

データ型:
ストリング

プロパティ:
TLS 証明書

ハンドシェイクの前に TLS 証明書を開くために必要なパスワードをクライアントが設定するためのプロパティ名。

以下の例は、キーリポジトリのパスワードプロパティの使用例です：

```
Windows properties.Add(XMSC.WMQ_SSL_KEY_REPOSITORY_PASSWORD, "123456")
```

```
Linux properties.Add(XMSC.WMQ_SSL_KEY_REPOSITORY_PASSWORD, "123456")
```

アプリケーションがクライアント・モードでキュー・マネージャーに接続している場合のみ、このプロパティが関連します。

デフォルトでは、このプロパティは設定されていません。

XMSC_WMQ_SYNCPOINT_ALL_GETS

データ型:
System.Boolean

プロパティ:
ConnectionFactory

すべてのメッセージを同期点制御の対象範囲内のキューから取り出す必要があるかどうかを示します。

プロパティの有効値は以下のとおりです。

有効値	意味
false	環境が適している場合、XMS クライアントは同期点制御の対象範囲外のキューからメッセージを取り出すことができます。
true	XMS クライアントは、すべてのメッセージを同期点制御の対象範囲内のキューから取り出す必要があります。

デフォルト値は false です。

XMSC_WMQ_TARGET_CLIENT

データ型:
System.Int32

プロパティ:
Destination

URI で使用される名前:
targetClient

宛先に送信されるメッセージに MQRFH2 ヘッダーを付けるかどうかを示します。

アプリケーションが MQRFH2 ヘッダーのあるメッセージを送信する場合は、受信側のアプリケーションがこのヘッダーを処理する必要があります。

プロパティの有効値は以下のとおりです。

有効値

XMSC_WMQ_TARGET_DEST_JMS

意味

宛先に送信されるメッセージには MQRFH2 ヘッダーがあります。アプリケーションが、MQRFH2 ヘッダーを処理する目的で設計されている別の XMS アプリケーション、IBM MQ classes for JMS アプリケーション、ネイティブの IBM MQ アプリケーションのいずれかにメッセージを送信する場合は、この値を指定します。

XMSC_WMQ_TARGET_DEST_MQ

宛先に送信されるメッセージには MQRFH2 ヘッダーがありません。アプリケーションが、MQRFH2 ヘッダーを処理する目的で設計されているわけではないネイティブの IBM MQ アプリケーションにメッセージを送信する場合は、この値を指定します。

デフォルト値は XMSC_WMQ_TARGET_DEST_JMS です。

XMSC_WMQ_TEMP_Q_PREFIX

データ型:

ストリング

プロパティ:

ConnectionFactory

アプリケーションが一時キュー XMS を作成する際に生成される動的キュー IBM MQ の名前を構成するために使用される接頭辞。

プレフィックスの形成規則は、オブジェクト **DynamicQName** 記述子内のフィールド内容を形成する規則と同じであるが、最後の空白でない文字はアスタリスク(*)でなければならない。デフォルトでは、このプロパティは設定されていない。設定されていない場合、以下の値が使用されます:

-  AMQ.* マルチプラットフォーム上で
-  CSQ.* on z/OS

このプロパティが関連するのは、Point-to-Point ドメインの場合に限られます。

XMSC_WMQ_TEMP_TOPIC_PREFIX

データ型:

ストリング

プロパティ:

ConnectionFactory、Destination

一時トピックを作成する際、XMS "TEMP/TEMPTOPICPREFIX/unique_id" という形式のトピック文字列が生成されます。ただし、このプロパティにデフォルト値が設定されている場合は、その文字列が TEMP/unique_id 生成されます。空以外の値を指定すると、この接続で作成された一時トピックへのサブスクライバーの管理対象キューを作成するために、特定のモデル・キューを定義できます。

IBM MQ トピックで有効な文字のみで構成されるヌル以外のストリングはすべて、このプロパティの有効な値です。

デフォルトでは、このプロパティは「」(空ストリング)に設定されています。

注: このプロパティが関連するのは、パブリッシュ/サブスクライブ・ドメインの場合に限られます。

XMSC_WMQ_TEMPORARY_MODEL

データ型:

ストリング

プロパティ:

ConnectionFactory

アプリケーションが一時キュー XMS を作成する際に、その一時キューの基となるモデルキュー IBM MQ の名前。

プロパティのデフォルト値は SYSTEM.DEFAULT.MODEL.QUEUE です。

このプロパティが関連するのは、Point-to-Point ドメインの場合に限られます。

XMSC_WMQ_WILDCARD_FORMAT

データ型:

System.Int32

プロパティ:

ConnectionFactory、Destination

このプロパティは、使用されるワイルドカード構文のバージョンを決定します。

IBM MQ '*' および '?' を指定してパブリッシュ/サブスクライブを使用する場合 ワイルドカードとして扱われます。一方、IBM Integration Bus でパブリッシュ/サブスクライブを使用する場合は、「#」と「+」がワイルドカードとして扱われます。このプロパティは XMSC_WMQ_BROKER_VERSION プロパティを置き換えます。

このプロパティの有効な値は以下のとおりです。

XMSC_WMQ_WILDCARD_TOPIC_ONLY

トピック・レベルのワイルドカードのみを認識します。例:「#」 および「+」はワイルドカードとして扱われます。この値は XMSC_WMQ_BROKER_V2 と同じです。

XMSC_WMQ_WILDCARD_CHAR_ONLY

文字ワイルドカードのみを認識します。例: '*' および '?' ワイルドカードとして扱われます。この値は XMSC_WMQ_BROKER_V1 と同じです。

デフォルトでは、このプロパティは XMSC_WMQ_WILDCARD_TOPIC_ONLY に設定されます。

XMSC_WMQ_暗号化キー

データ型:

ストリング

プロパティ:

TLS 証明書

クライアントが暗号化キーファイルの保存場所を設定するためのプロパティ名。

以下の例は、暗号化キープロパティの使用例です：

```
Windows properties.Add(XMSC.WMQ_ENCRYPTION_KEY, "C:\\CustomPath_Certificates\\EncryptionKey\\mykey.txt")
```

```
Linux properties.Add(XMSC.WMQ_ENCRYPTION_KEY, "C:\\CustomPath_Certificates\\EncryptionKey\\mykey.txt")
```

アプリケーションがクライアント・モードでキュー・マネージャーに接続している場合のみ、このプロパティが関連します。

デフォルトでは、このプロパティは設定されていません。

注: クライアントアプリケーション .NET で暗号化キープロパティが設定されていない場合、ライブラリはパスワードプロパティに設定された値がプレーンテキスト形式か、Base64 でエンコードされた形式かをチェックします。

- 値がプレーンテキストの場合、ライブラリはそれをプレーンテキストのパスワードとして扱います。
- 値が Base64 でエンコードされている場合、デフォルトキーを使用して復号化し、TLS 接続を構築します。

確認後、アプリケーションは TLS 接続を続行します。

XMSC_WPM_BUS_NAME

データ型:

ストリング

プロパティ:

ConnectionFactory および Destination

URI で使用される名前:

busName

接続ファクトリーの場合は、アプリケーションの接続先となるサービス統合バスの名前であり、宛先の場合は、その宛先が存在するサービス統合バスの名前です。

内容がトピックである宛先の場合、このプロパティは、関連するトピック・スペースが存在するサービス統合バスの名前になります。このトピック・スペースは、XMSC_WPM_TOPIC_SPACE プロパティで指定します。

このプロパティを宛先に設定しなかった場合は、アプリケーションの接続先となるサービス統合バスにキューまたは関連するトピック・スペースが存在することが前提になります。

デフォルトでは、このプロパティは設定されていません。

XMSC_WPM_CONNECTION_PROTOCOL

データ型:

System.Int32

プロパティ:

接続

メッセージング・エンジンへの接続に使用される通信プロトコルです。このプロパティは読み取り専用です。

このプロパティの可能な値は以下のとおりです。

値	意味
XMSC_WPM_CP_HTTP	接続には HTTP over TCP/IP を使用します。
XMSC_WPM_CP_TCP	接続には TCP/IP を使用します。

XMSC_WPM_CONNECTION_PROXIMITY

データ型:

System.Int32

プロパティ:

ConnectionFactory

接続に対する接続接近性の設定です。このプロパティは、アプリケーションが接続するメッセージング・エンジンが、ブートストラップ・サーバーにどれだけ近ければよいかを決定します。

プロパティの有効値は以下のとおりです。

有効値	接続接近性の設定
XMSC_WPM_CONNECTION_PROXIMITY_BUS	バス
XMSC_WPM_CONNECTION_PROXIMITY_CLUSTER	クラスター
XMSC_WPM_CONNECTION_PROXIMITY_HOST	ホスト
XMSC_WPM_CONNECTION_PROXIMITY_SERVER	サーバー

デフォルト値は XMSC_WPM_CONNECTION_PROXIMITY_BUS です。

XMSC_WPM_DUR_SUB_HOME

データ型:

ストリング

プロパティ:

ConnectionFactory

URI で使用される名前:

durableSubscriptionHome

永続サブスクライバーに配信される ある接続または宛先に対するすべての永続サブスクリプションが管理対象になっているメッセージング・エンジンの名前です。メッセージは、同じメッセージング・エンジンの公開ポイントに保管されます。

接続を使用する永続サブスクライバーをアプリケーションが作成する前に、その接続の永続サブスクリプション・ホームを指定する必要があります。宛先に対して指定されている値は、接続に対して指定されている値に優先します。

デフォルトでは、このプロパティは設定されていません。

このプロパティが関連するのは、パブリッシュ/サブスクライブ・ドメインの場合に限られます。

XMSC_WPM_HOST_NAME

データ型:

ストリング

プロパティ:

接続

アプリケーションの接続先となるメッセージング・エンジンが収容されているシステムのホスト名または IP アドレスです。このプロパティは読み取り専用です。

XMSC_WPM_LOCAL_ADDRESS

データ型:

ストリング

プロパティ:

ConnectionFactory

サービス統合バスへの接続の場合、このプロパティは、使用するローカル・ネットワーク・インターフェース、または使用するローカル・ポート (1 つまたは一定範囲)、あるいはその両方を指定します。

このプロパティの値は、次の形式のストリングです。

[*host_name*][(*low_port*),*high_port*]]

変数の意味は以下のとおりです。

host_name

接続に使用するローカル・ネットワーク・インターフェースのホスト名または IP アドレスです。

この情報の入力が必要なのは、アプリケーションを実行しているシステムに複数のネットワーク・インターフェースがあり、接続にどのインターフェースを使用する必要があるかを指定できることが必要な場合に限ります。システムが備えるネットワーク・インターフェースが 1 つのみである場合、使用できるのはそのインターフェースのみです。システムに複数のネットワーク・インターフェースがあり、どのインターフェースを使用するかを指定していない場合、インターフェースはランダムに選択されます。

low_port

接続に使用するローカル・ポートの数です。

high_port も指定した場合、*low_port* は一連のポート番号のうち最小のポート番号と解釈されます。

high_port

一連のポート番号のうち最大のポート番号を表します。指定した範囲内のいずれかのポートを接続に使用する必要があります。

プロパティの有効値の例の一部を以下に示します。

JUPITER
9.20.4.98
JUPITER(1000)
9.20.4.98(1000,2000)
(1000)
(1000,2000)

デフォルトでは、このプロパティは設定されていません。

XMSC_WPM_ME_NAME

データ型:

ストリング

プロパティ:

接続

アプリケーションの接続先となるメッセージング・エンジンの名前です。このプロパティは読み取り専用です。

XMSC_WPM_NON_PERSISTENT_MAP

データ型:

System.Int32

プロパティ:

ConnectionFactory

接続を使用して送信される非永続メッセージの信頼性レベルです。

プロパティの有効値は以下のとおりです。

有効値

XMSC_WPM_MAPPING_AS_DESTINATION

XMSC_WPM_MAPPING_BEST_EFFORT_NON_PERSISTENT

XMSC_WPM_MAPPING_EXPRESS_NON_PERSISTENT

XMSC_WPM_MAPPING_RELIABLE_NON_PERSISTENT

XMSC_WPM_MAPPING_RELIABLE_PERSISTENT

XMSC_WPM_MAPPING_ASSURED_PERSISTENT

信頼性レベル

サービス統合バスでキューまたはトピック・スペースに指定されているデフォルトの信頼性レベルにより決定

ベスト・エフォート型の非永続

高速の非永続

高信頼性の非永続

高信頼性の永続

確実な永続

デフォルト値は XMSC_WPM_MAPPING_EXPRESS_NON_PERSISTENT です。

XMSC_WPM_PERSISTENT_MAP

データ型:

System.Int32

プロパティ:

ConnectionFactory

接続を使用して送信される永続メッセージの信頼性レベルです。

プロパティの有効値は以下のとおりです。

有効値

信頼性レベル

XMSC_WPM_MAPPING_AS_DESTINATION

サービス統合バスでキューまたはトピック・スペースに指定されているデフォルトの信頼性レベルにより決定

XMSC_WPM_MAPPING_BEST_EFFORT_NON_PERSISTENT

ベスト・エフォート型の非永続

XMSC_WPM_MAPPING_EXPRESS_NON_PERSISTENT

高速の非永続

XMSC_WPM_MAPPING_RELIABLE_NON_PERSISTENT

高信頼性の非永続

XMSC_WPM_MAPPING_RELIABLE_PERSISTENT

高信頼性の永続

XMSC_WPM_MAPPING_ASSURED_PERSISTENT

確実な永続

デフォルト値は XMSC_WPM_MAPPING_RELIABLE_PERSISTENT です。

XMSC_WPM_PORT

データ型:

System.Int32

プロパティ:

接続

アプリケーションの接続先となるメッセージング・エンジンによって listen されるポートの数です。このプロパティは読み取り専用です。

XMSC_WPM_PROVIDER_ENDPOINTS

データ型:

ストリング

プロパティ:

ConnectionFactory

ブートストラップ・サーバーの 1 つ以上のエンドポイント・アドレスの列です。エンドポイント・アドレスはコンマで区切られます。

ブートストラップ・サーバーとは、アプリケーションの接続先となるメッセージング・エンジンを選択する役割を果たすアプリケーション・サーバーのことです。ブートストラップ・サーバーのエンドポイント・アドレスの形式は次のとおりです。

host_name:port_number:chain_name

エンドポイント・アドレスの構成要素の意味は、以下のとおりです。

host_name

ブートストラップ・サーバーが存在するシステムのホスト名または IP アドレスです。ホスト名または IP アドレスを指定していない場合、デフォルトは localhost になります。

port_number

ブートストラップ・サーバーが着信要求を listen するポートの数です。ポート番号を指定していない場合、デフォルトは 7276 になります。

chain_name

ブートストラップ・サーバーが使用するブートストラップ・トランスポート・チェーンの名前です。有効な値は以下のとおりです。

有効値	ブートストラップ・トランスポート・チェーンの名前
XMSC_WPM_BOOTSTRAP_HTTP	BootstrapTunneledMessaging
XMSC_WPM_BOOTSTRAP_HTTPS	BootstrapTunneledSecureMessaging
XMSC_WPM_BOOTSTRAP_SSL	BootstrapSecureMessaging
XMSC_WPM_BOOTSTRAP_TCP	BootstrapBasicMessaging

名前が指定されていない場合、デフォルト値は XMSC_WPM_BOOTSTRAP_TCP です。

エンドポイント・アドレスを指定していない場合、デフォルトは localhost:7276:BootstrapBasicMessaging になります。

XMSC_WPM_SSL_CIPHER_SUITE

データ型:

ストリング

プロパティ:

ConnectionFactory

メッセージングエンジン WebSphere Application Server service integration bus への TLS 接続で使用する CipherSuite の名前。セキュア接続のネゴシエーションで使用するプロトコルは、指定されている CipherSuite によって異なります。

表 917. WebSphere Application Server service integration bus メッセージング・エンジンとの接続の CipherSuite オプション	
暗号スイート	使用されるプロトコル
TLS_RSA_WITH_DES_CBC_SHA	TLSv1
TLS_RSA_WITH_3DES_EDE_CBC_SHA	TLSv1
TLS_RSA_WITH_AES_128_CBC_SHA	TLSv1
TLS_RSA_WITH_AES_256_CBC_SHA	TLSv1

注:

-  TLS_RSA_WITH_AES_128_CBC_SHA TLS_RSA_WITH_AES_256_CBC_SHA と CipherSuites はでのみ Windows サポートされています。(これはによって指示されています GSKit。)
-  TLS_RSA_WITH_3DES_EDE_CBC_SHA は推奨されません。ただし、32 GB 以下のデータの転送にはまだ使用できますが、これを超えるとエラー AMQ9288 を出して接続が終了します。このエラーを回避するために、Triple-DES を使用しないか、またはこの CipherSpec を使用する際に秘密鍵リセットを有効にする必要があります。

このプロパティにはデフォルトがありません。SSL または TLS を使用する場合には、このプロパティに値を指定する必要があります。そうしない場合、アプリケーションは正常にサーバーに接続することができません。

XMSC_WPM_SSL_FIPS_REQUIRED

注:

-  Windows AIX、Linux、およびにおいて、IBM MQ は GSKit 9 IBM Crypto for C (ICC) 暗号モジュールを通じてへの準拠を実現 FIPS 140-3 しています。この FIPS 140-3 モジュールに関連する NIST 認証は、<https://csrc.nist.gov/projects/cryptographic-module-validation-program/certificate/4755> で確認できます。
-  Linux s390x FIPS 現在、への対応は行っていません。IBM MQ このプラットフォームをご利用のお客様でサポートが必要な FIPS 場合は、以前のバージョンのままご利用ください。このプラットフォーム上で暗号モジュールが IBM Crypto for C (ICC) 認証を取得次第、今後の FixPack でサポートが FIPS 140-3 有効化されます。
-  このソフトウェアに含まれる暗号 FIPS 140-3 モジュールは、2024 IBM Semeru Runtime 年 8 月に NIST の承認を受けた。IBM Semeru Runtime 11+IBM MQ 10.0.0 TLS を使用したおよび IBM MQ classes for Java のクライアント接続の IBM MQ classes for JMS 処理機能を、FIPS 140-3 およびに Java 8 追加しました。<https://csrc.nist.gov/projects/cryptographic-module-validation-program/certificate/4755> この FIPS 140-3 モジュールに関連する NIST 認証は、以下で確認できます。
- Containers では IBM MQ、3.2.0 およびキューマネージャ IBM MQ Operator 一のコンテナイメージ 9.4.0.0 以降は、UBI 9 をベースとしています。FIPS 140-3 Containers における IBM MQ コンプライアンスは、現在審査中です。
 が有効になっている FIPS 場合、コンテナ制御 IBM MQ プロセスでは、認定 FIPS 140-3 OpenSSL モジュールが使用されます。NIST 認証の詳細については、以下の URL をご覧ください：<https://access.redhat.com/compliance/fips>。IBM MQ コンテナイメージ内で実行されるキューマネージャは、ベースイメージのプラットフォームバージョンと同じ FIPS IBM MQ 認証レベルを持っています。

データ型:

Boolean

プロパティ:

ConnectionFactory

このプロパティの値は、アプリケーションが非準拠 FIPS の暗号スイートを使用できるかどうかを決定します。このプロパティが true に設定されている場合、クライアント-サーバー接続には暗号スイート FIPS のみを使用されます。このプロパティの値を TRUE に設定すると、アプリケーションが準拠 FIPS していない暗号スイートを使用するのを防ぎます。

デフォルトでは、プロパティは FALSE (つまりモードオフ FIPS) に設定されています。

XMSC_WPM_SSL_KEY_REPOSITORY

データ型:

ストリング

プロパティ:

ConnectionFactory

セキュア接続で使用される公開鍵または秘密鍵が含まれている鍵リング・ファイルであるファイルに至るパスです。

鍵リング・ファイル・プロパティを特殊値 XMSC_WPM_SSL_MS_CERTIFICATE_STORE に設定すると、Microsoft Windows 鍵データベースを使用することが指定されることになります。Microsoft Windows 鍵データベースは、「コントロールパネル」>「インターネットオプション」>「コンテンツ」>「証明書」にあります。この鍵データベースを使用すれば、別個の鍵ファイル・データベースは必要ありません。Windows x64 などのプラットフォームでこの定数を使用することはできません。

デフォルトでは、このプロパティは設定されていません。

XMSC_WPM_SSL_KEYRING_LABEL

データ型:

ストリング

プロパティ:

ConnectionFactory

サーバーによる認証で使用する証明書。値を指定しない場合は、デフォルトの証明書が使用されます。
デフォルトでは、このプロパティは設定されていません。

XMSC_WPM_SSL_KEYRING_PW

データ型:

ストリング

プロパティ:

ConnectionFactory

鍵リング・ファイルのパスワード。

XMSC WPM SSL KEYRING STASH FILE を使用する代わりにこのプロパティを使用することによって、
鍵リング・ファイルのパスワードを構成することができます。

デフォルトでは、このプロパティは設定されていません。

XMSC_WPM_SSL_KEYRING_STASH_FILE

データ型:

ストリング

プロパティ:

ConnectionFactory

鍵リポジトリ・ファイルのパスワードが含まれているバイナリー・ファイルの名前です。

XMSC WPM SSL KEYRING PW を使用する代わりにこのプロパティを使用することによって、鍵リン
グ・ファイルのパスワードを構成することができます。

デフォルトでは、このプロパティは設定されていません。

XMSC_WPM_TARGET_GROUP

データ型:

ストリング

プロパティ:

ConnectionFactory

メッセージング・エンジンのターゲット・グループの名前です。ターゲット・グループの種類は、
XMSC_WPM_TARGET_TYPE プロパティで決まります。

メッセージング・エンジンの検索対象を、サービス統合バス内のメッセージング・エンジンのサブグルー
プに絞り込む場合には、このプロパティを設定してください。アプリケーションがサービス統合バス内
の任意のメッセージング・エンジンに接続できるようにする場合は、このプロパティを設定しないでく
ださい。

デフォルトでは、このプロパティは設定されていません。

XMSC_WPM_TARGET_SIGNIFICANCE

データ型:

System.Int32

プロパティ:

ConnectionFactory

メッセージング・エンジンのターゲット・グループの重要度です。

プロパティの有効値は以下のとおりです。

有効値

XMSC_WPM_TARGET_SIGNIFICANCE_PREFERRED

XMSC_WPM_TARGET_SIGNIFICANCE_REQUIRED

意味

ターゲット・グループ内にメッセージング・エンジンが存在する場合は、そのメッセージング・エンジンが選択されます。そうでない場合は、ターゲット・グループの外側に存在するメッセージング・エンジンが選択されます。ただし、そのメッセージング・エンジンが同じサービス統合バス内に存在することが前提です。

選択されたメッセージング・エンジンは、ターゲット・グループ内に存在する必要があります。ターゲット・グループ内にあるメッセージング・エンジンが使用不可の場合、該当する接続処理は失敗します。

プロパティのデフォルト値は XMSC_WPM_TARGET_SIGNIFICANCE_PREFERRED です。

XMSC_WPM_TARGET_TRANSPORT_CHAIN

データ型:

ストリング

プロパティ:

ConnectionFactory

アプリケーションがメッセージング・エンジンに接続するために使用する必要があるインバウンド・トランスポート・チェーンの名前です。

このプロパティの値は、メッセージング・エンジンのホストとして動作するアプリケーション・サーバー内で使用できるいずれかのインバウンド・トランスポート・チェーンの名前です。事前に定義されているインバウンド・トランスポート・チェーンの1つには、次の名前付き定数が用意されています。

名前付き定数

XMSC_WPM_TARGET_TRANSPORT_CHAIN_BASIC

トランスポート・チェーンの名前

InboundBasicMessaging

プロパティのデフォルト値は XMSC_WPM_TARGET_TRANSPORT_CHAIN_BASIC です。

XMSC_WPM_TARGET_TYPE

データ型:

System.Int32

プロパティ:

ConnectionFactory

メッセージング・エンジンのターゲット・グループのタイプです。このプロパティは、XMSC_WPM_TARGET_GROUP プロパティで指定されるターゲット・グループの種類を判別するプロパティです。

プロパティの有効値は以下のとおりです。

有効値

XMSC_WPM_TARGET_TYPE_BUSMEMBER

意味

ターゲット・グループの名前は、バス・メンバーの名前になります。このターゲット・グループの構成要素は、バス・メンバー内のすべてのメッセージング・エンジンです。

有効値

XMSC_WPM_TARGET_TYPE_CUSTOM

XMSC_WPM_TARGET_TYPE_ME

意味

ターゲット・グループの名前は、メッセージング・エンジンのユーザー定義グループの名前になります。このターゲット・グループの構成要素は、ユーザー定義グループに登録されているすべてのメッセージング・エンジンです。

ターゲット・グループの名前は、メッセージング・エンジンの名前になります。ターゲット・グループは、指定されているメッセージング・エンジンです。

デフォルトでは、このプロパティは設定されていません。

XMSC_WPM_TEMP_Q_PREFIX

データ型:

ストリング

プロパティ:

ConnectionFactory

アプリケーションが一時キュー XMS を作成する際に、サービス統合バス内に作成される一時キューの名前を構成するために使用される接頭辞。接頭部には、最大 12 文字を含めることができます。

一時キューの名前は、「_Q」という文字で始まり、その後に接頭部が付きます。名前の残りの部分は、システム生成文字で構成されます。

デフォルトでは、このプロパティは設定されていません。つまり、一時キューの名前にプレフィックスはありません。

このプロパティが関連するのは、Point-to-Point ドメインの場合に限られます。

XMSC_WPM_TEMP_TOPIC_PREFIX

データ型:

ストリング

プロパティ:

ConnectionFactory

アプリケーションによって作成される一時トピックの名前を構成する場合に使用されるプレフィックスです。接頭部には、最大 12 文字を含めることができます。

一時トピックの名前は、先頭に文字「_T」が付き、その後に接頭部が付きます。名前の残りの部分は、システム生成文字で構成されます。

デフォルトでは、このプロパティは設定されていません。つまり、一時トピックの名前にプレフィックスはありません。

このプロパティが関連するのは、パブリッシュ/サブスクライブ・ドメインの場合に限られます。

XMSC_WPM_TOPIC_SPACE

データ型:

ストリング

プロパティ:

Destination

URI で使用される名前:

topicSpace

このプロパティを持つことができるのは、トピックに含まれる宛先のみです トピックが収容されているトピック・スペースの名前です。

デフォルトでは、このプロパティは設定されていません。つまり、デフォルトのトピック・スペースが前提になります。

このプロパティが関連するのは、パブリッシュ/サブスクライブ・ドメインの場合に限られます。

Managed File Transfer アプリケーション開発リファレンス

Managed File Transfer のアプリケーションの開発に役立つ参照情報。

fteCreateTransfer を使用してプログラムを開始する例

fteCreateTransfer コマンドを使用して、転送前または転送後に実行するプログラムを指定することができます。

fteCreateTransfer を使用する以外にも、転送前または転送後にプログラムを起動する方法があります。詳しくは、[MFT で実行するプログラムの指定](#)を参照してください。

これらの例はすべて、以下の構文を使用してプログラムを指定します。

```
[type:]commandspec[, [retrycount][, [retrywait][, successsrc]]]
```

この構文について詳しくは、[fteCreateTransfer: 新規ファイル転送の開始](#)を参照してください。

実行可能プログラムの実行

以下の例は、mycommand という実行可能プログラムを指定し、そのプログラムに 2 つの引数 a および b を渡します。

```
mycommand(a,b)
```

このプログラムを転送の開始前にソース・エージェント AGENT1 で実行するには、次のコマンドを使用します。

```
fteCreateTransfer -sa AGENT1 -da AGENT2 -presrc mycommand(a,b)
destinationSpecification sourceSpecification
```

実行可能プログラムの実行と再試行

以下の例は、simple という実行可能プログラムを指定しています。このプログラムは引数を取りません。retrycount には値 1 が指定され、retrywait には値 5 が指定されています。これらの値は、そのプログラムが成功を表す戻りコードを返さない場合には、5 秒後に 1 回再試行されることを意味しています。successsrc には値が指定されていないため、成功を表す戻りコードはデフォルト値の 0 のみです。

```
executable:simple,1,5
```

このプログラムを転送の完了後にソース・エージェント AGENT1 で実行するには、次のコマンドを使用します。

```
fteCreateTransfer -sa AGENT1 -da AGENT2 -postsrc executable:simple,1,5
destinationSpecification sourceSpecification
```

Ant スクリプトの実行と成功を表す戻りコードの指定

次の例は、`script` という Ant 名前のスクリプトを指定し、そのスクリプト myscript に 2 つのプロパティを渡します。このスクリプトは、**fteAnt** コマンドを使用して実行されます。successsrc の値は >2&<7&!5|0|14 として指定されます。この値は、戻りコード 0、3、4、6、および 14 が成功を示すことを指定します。

```
antscript:myscript(prop1=fred,prop2=bob),,,>2&<7&!5|0|14
```

このプログラムを転送の開始前に宛先エージェント AGENT2 で実行するには、次のコマンドを使用します。


```
fteCreateTransfer -sa AGENT1 -da AGENT2 -predst  
"antscript:myscript(prop1=fred,prop2=bob),,,>2&<7&!5|0|14"destinationSpecification sourceSpecification
```

Ant スクリプトの実行と呼び出すターゲットの指定

次の例は、`Ant script` と呼ばれる `script2` スクリプトと、呼び出す 2 つのターゲット `target1` `` および `target2` `` を指定しています。プロパティ `prop1` も値 `recmf(F,B)` を指定して渡されます。括弧、コンマ (,)、およびバックスラッシュ (\) は、MFT コマンドにおいて特殊文字となるため、バックスラッシュ (\) でエスケープする必要があります。  Windows では、ファイルパスを指定する際、区切り文字としてバックスラッシュ (\) を 2 つ並べる方法と、スラッシュ (/) を 1 つ使う方法のどちらでも指定できます。以下の例では、コンマ (,) と括弧はバックスラッシュ文字 (\) を使用してエスケープされています。

```
antscript:script2(target1,target2,prop1=recmf\F\B\),,,>2&<7&!5|0|14
```

このプログラムを転送の完了後に宛先エージェント AGENT2 で実行するには、次のコマンドを使用します。

```
fteCreateTransfer -sa AGENT1 -da AGENT2  
-postdst "antscript:script2(target1,target2,prop1=recmf\F\B\),,,>2&<7&!5|0|14"  
destinationSpecification sourceSpecification
```

Ant スクリプトでのメタデータの使用

転送の以下の呼び出しに Ant タスクを指定できます。

- ソースの前
- ソースの後
- 宛先の前
- 宛先の後

Ant タスクの実行時には、環境変数を使用して転送のユーザー・メタデータを提供できます。このデータを使用してアクセスできます。以下にコードの例を示します。

```
<property environment="environment" />  
<echo>${environment.mymetadata}</echo>
```

`mymetadata` は、転送に挿入するメタデータの名前です。

JCL スクリプトの実行

以下の例では ZOSBATCH という JCL スクリプトを指定しています。`retrycount` には値 3 が指定され、`retrywait` には値 30 が指定され、`successrc` には値 0 が指定されています。これらの値は、そのスクリプトが成功を表す戻りコード 0 を返さない場合には、30 秒おきに 3 回再試行されることを意味しています。

```
jcl:ZOSBATCH,3,30,0
```

ZOSBATCH は、MYSYS.JCL という PDS メンバーであり、`agent.properties` ファイルには `commandPath=...:/ 'MYSYS.JCL':...` という行が含まれています。

このプログラムを転送の完了後にソース・エージェント AGENT1 で実行するには、次のコマンドを使用します。

```
fteCreateTransfer -sa AGENT1 -da AGENT2 -postsrc jcl:ZOSBATCH,3,30,0  
destinationSpecification sourceSpecification
```

関連タスク

[MFT で実行するプログラムの指定](#)

関連資料

[fteCreateTransfer: 新規ファイル転送の開始](#)

fteAnt: タスク Ant を実行する MFT

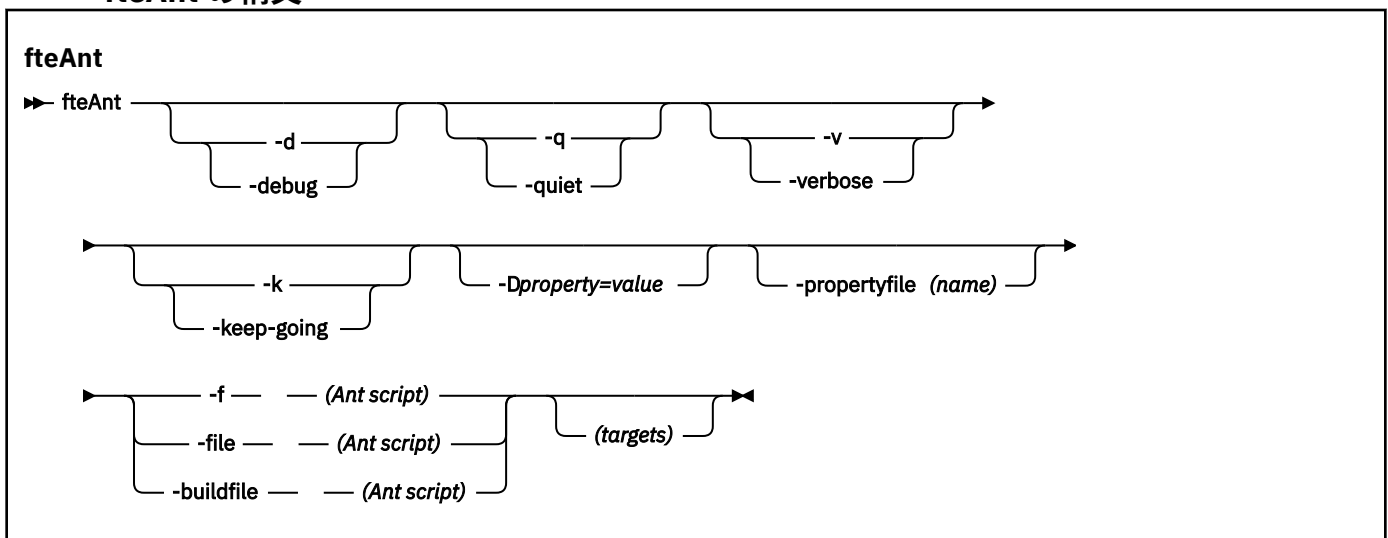
コマンド **fteAnt** は、利用可能なタスク AntManaged File Transfer が存在する環境でスクリプト Ant を実行します。標準 **ant** コマンドとは異なり、**fteAnt** の場合は、スクリプト・ファイルを定義する必要があります。

MFT Ant のタスクおよびネスト・パラメーター

Managed File Transfer では、数多くの Ant タスクが用意されており、これらのタスクを使用して、ファイル転送機能にアクセスできます。ネストされたパラメーターのセットもあります。これらのパラメーターは、用意されている Ant タスクのいくつかに共通するネスト・エレメントのセットを表します。

このトピックの残りの部分では、**fteAnt** コマンドの構文、パラメーター、使用例、および戻りコードについて説明します。MFT によって提供される Ant タスクおよびネストされたパラメーターの詳細については、以下を参照してください。サブトピックを参照してください。

fteAnt の構文



パラメーター

-debug または -d

オプション。デバッグ出力を生成します。

-quiet または -q

オプション。最小出力を生成します。

-verbose または -v

オプション。詳細出力を生成します。

-keep-going または -k

オプション。失敗したターゲットに依存しないすべてのターゲットを実行します。

-D property=value

オプション。一定のプロパティの値を使用します。-D で設定されたプロパティは、プロパティ・ファイルで設定されたプロパティよりも優先されます。

プロパティ **com.ibm.wmqfte.propertyset** を使用して、Ant タスクに使用される構成オプションのセットを指定します。このプロパティの値には、デフォルトでない調整キュー・マネージャーの名前を使用します。そうすることで、Ant タスクは、このデフォルトではない調整キュー・マネージャーに関連付けられた構成オプションのセットを使用します。このプロパティを指定しない場合、デフォルトの調整キュー・マネージャーに基づいたデフォルトの構成オプションのセットが使用されます。タスク **Antcmdqm** に対して属性を指定した場合、この属性はコマンド **fteAnt** に対して指定された設定オプションのセットよりも優先されます。この動作は、デフォルトの構成オプション・セット

を使用するか、**com.ibm.wmqfte.propertyset** プロパティを使用してセットを指定するかに関係なく適用されます。

-propertyfile (name)

オプション。 **-D** プロパティが優先されるファイルからすべてのプロパティをロードします。

-f (Ant スクリプト)、-file (Ant スクリプト)、または -buildfile (Ant スクリプト)

必須。 実行する Ant スクリプトの名前を指定します。

targets

オプション。 Ant スクリプトの実行元の 1 つ以上のターゲットの名前。 このパラメーターに値を指定しない場合は、スクリプトのデフォルト・ターゲットが実行されます。

-version

オプション。 Managed File Transfer コマンドおよび Ant のバージョンを表示します。

-? または -h

オプション。 コマンド構文を表示します。

例

この例では、Ant スクリプト `fte_script.xml` 内のターゲット **copy** が実行され、コマンドはデバッグ出力を標準出力に書き込みます。

```
fteAnt -d -f fte_script.xml copy
```

戻りコード

0

コマンドは正常に完了しました。

1

コマンドは失敗しました。

他の状況戻りコードは、Ant スクリプトからも、例えば、Ant fail タスクを使用するなどして指定できます。詳しくは、[失敗](#)を参照してください。

関連概念

[MFT とともに Ant スクリプトの使用を開始する](#)

関連タスク

[Apache Ant と MFT の併用](#)

関連資料

[サンプル Ant タスク MFT](#)

fte:awaitoutcome の Ant タスク

fte:filecopy、**fte:filemove**、または **fte:call** のいずれかの操作が完了するのを待機します。

属性

ID

必須。 結果を待機する対象の転送を指定します。 通常、これは、[fte:filecopy](#)、[fte:filemove](#)、または [fte:call](#) タスクの `idProperty` 属性によって設定されるプロパティです。

rcproperty

必須。 **fte:awaitoutcome** タスクの戻りコードを保管するためのプロパティの名前を指定します。

timeout

オプション。 操作が完了するまで待機する最大時間 (秒単位)。 タイムアウトの最小値は 1 秒です。 タイムアウト値を指定しなかった場合には、**fte:awaitoutcome** タスクは操作の結果が決定するまで無期限で待機します。

例

この例では、ファイル・コピーが開始され、その ID が `copy.id` プロパティに保管されます。コピーの進行中には、他の処理を行うことができます。 **`fte:awaitoutcome`** ステートメントを使用して、`copy` 操作が完了するまで待機しています。この **`fte:awaitoutcome`** ステートメントでは、`copy.id` プロパティに保管されている ID を使用して、待機対象の操作を指定しています。この **`fte:awaitoutcome`** は、`copy` 操作の結果を示す戻りコードを、`copy.result` というプロパティに保管します。

```
<-- issue a file copy request -->
<fte:filecopy
src="AGENT1@QM1"
dst="AGENT2@QM2"
idproperty="copy.id"
outcome="defer">

<fte:filespec
srcfilespec="/home/fteuser1/file.bin"
dstdir="/home/fteuser2"/>

</fte:filecopy>

<fte:awaitoutcome id="${copy.id}" rcProperty="copy.rc"/>

<echo>Copy id=${copy.id} rc=${copy.rc}</echo>
```

関連タスク

[Apache Ant と MFT の併用](#)

fte:call の Ant タスク

`fte:call` タスクを使用して、スクリプトおよびプログラムをリモートで呼び出すことができます。

このタスクによって、**`fte:call`** 要求をエージェントに送信できます。エージェントは、スクリプトまたはプログラムを実行してその結果を返すことで、この要求を処理します。呼び出すコマンドは、エージェントにアクセス可能である必要があります。 `agent.properties` ファイル内の `commandPath` プロパティ値に、呼び出すコマンドの場所が含まれていることを確認してください。コマンドがネストされているエレメントによって指定されたパス情報は、`commandPath` プロパティで指定された場所と相対的な位置になければなりません。デフォルトでは、`commandPath` は空であるため、エージェントはどのコマンドも呼び出すことができません。このプロパティについて詳しくは、[commandPath MFT プロパティ](#) を参照してください。

`agent.properties` ファイルについて詳しくは、[MFT agent.properties ファイル](#) を参照してください。

属性

エージェント

必須。 **`fte:call`** 要求の実行依頼先のエージェントを指定します。エージェント情報を `agentname@qmgrname` の形式で指定します。ここで、`agentname` はエージェントの名前、`qmgrname` はこのエージェントが直接接続されているキュー・マネージャーの名前です。

cmdqmq

オプション。要求の実行依頼先のコマンド・キュー・マネージャー。この情報は、`qmgrname@host@port@channel` の形式で指定します。ここで、

- `qmgrname` はキュー・マネージャーの名前です
- `host` は、キュー・マネージャーが実行されているシステムのオプションのホスト名です。
- `port` は、キュー・マネージャーが `listen` するオプションのポート番号です。
- `channel` は、使用するオプションの `SVRCONN` チャンネルです。

コマンド・キュー・マネージャーの `host`、`port`、または `channel` 情報を省略すると、`command.properties` ファイルに指定されている接続情報が使用されます。



重要: 値が指定されていない場合は、以下のようになります。

- ホスト変数、バインディング・モードが使用される
- ポート 変数。値 1414 が使用されます。
- チャンネル変数の場合は SYSTEM.DEF.SVRCONN が使用されます。

詳しくは、[MFT command.properties ファイル](#)を参照してください。

ただし、中央の属性 (qmgrname@host@@channel など) をスキップすることはできません。例えば、qmgrname@host、qmgrname@host@port、qmgrname@hostport@@channel のいずれかを指定できます。

MFT は、@区切り文字を使用して、指定された属性を分割します。検出されたトークンの数に応じて、最初のトークンは *qmgrname*、2 番目のトークンは *host*、3 番目のトークンは *port*、そして最後に *channel* となります。

詳しくは、[MFT command.properties ファイル](#)を参照してください。

com.ibm.wmqfte.propertySet プロパティを使用して、使用する `command.properties` ファイルを指定できます。詳細については、[com.ibm.wmqfte.propertySet](#) を参照してください。

cmdqm 属性を使用しない場合、タスクはデフォルトで

`com.ibm.wmqfte.ant.commandQueueManager` プロパティを使用します (このプロパティが設定されている場合)。`com.ibm.wmqfte.ant.commandQueueManager` プロパティが設定されていない場合、`command.properties` ファイルに定義されているデフォルト・キュー・マネージャーへの接続が試行されます。`com.ibm.wmqfte.ant.commandQueueManager` プロパティの形式は、`cmdqm` 属性と同じです。つまり、`qmgrname@host@port@channel` です。

idproperty

`defer` の `outcome` を指定していない場合はオプションです。転送 ID を割り当てるプロパティの名前を指定します。転送 ID は、転送要求が実行依頼された時点で生成されます。この転送 ID を使用して、転送の進行の追跡、転送で生じた問題の診断、および転送の取り消しを行うことができます。

`ignore` の `outcome` プロパティも指定した場合は、このプロパティを指定できません。ただし、`defer` の `outcome` プロパティも指定した場合は、`idproperty` を指定する必要があります。

jobname

オプション。ジョブ名を **fte:call** 要求に割り当てます。ジョブ名を使用して、論理転送グループを作成できます。[2125 ページの『fte:uuid の Ant タスク』](#) タスクを使用して、疑似固有ジョブ名を生成します。`jobname` 属性を使用しない場合、タスクはデフォルトで `com.ibm.wmqfte.ant.jobName` プロパティ値を使用します (このプロパティが設定されている場合)。このプロパティが設定されていない場合には、**fte:call** 要求に関連付けられるジョブ名はありません。

origuser

オプション。**fte:call** 要求に関連付ける発信ユーザー ID を指定します。`origuser` 属性を使用しなかった場合には、タスクはデフォルトで Ant スクリプトを実行するために使用されるユーザー ID を使用します。

outcome

オプション。Ant スクリプトに制御を返す前に、タスクが **fte:call** 操作の完了を待機するかどうかを決定します。以下のいずれかのオプションを指定します。

await

タスクは、戻る前に **fte:call** 操作が完了するまで待機します。`await` の `outcome` が指定されている場合、`idproperty` 属性はオプションです。

defer

タスクは、**fte:call** 要求がサブミットされるとすぐに戻り、[awaitoutcome](#) タスクまたは `ignoreoutcome` タスクのいずれかを使用して、呼び出し操作の結果が後で処理されることを想定します。`defer` の `outcome` が指定されている場合、`idproperty` 属性は必須です。

ignore

fte:call 操作の結果が重要ではない場合には、値 `ignore` を指定できます。この値を指定した場合、タスクは、コマンドの結果を追跡するためのリソースを割り当てずに、**fte:call** 要求が実行

依頼されるとすぐに戻ります。 **ignore** の **outcome** が指定されている場合、**idproperty** 属性を指定することはできません。

outcome 属性を指定しない場合、タスクはデフォルトで値 **await** を使用します。

rcproperty

オプション。 **fte:call** 要求の結果コードを割り当てるプロパティの名前を指定します。結果コードには、**fte:call** 要求の全体的結果が反映されます。

ignore または **defer** の **outcome** プロパティも指定した場合は、このプロパティを指定できません。ただし、**await** の結果を指定した場合は、**rcproperty** を指定する必要があります。

ネスト・エレメントとして指定するパラメーター

fte:command

エージェントで呼び出すコマンドを指定します。特定の **fte:call** 操作に関連付けることができる **fte:command** エレメントは1つのみです。呼び出されるコマンドは、エージェントの **agent.properties** ファイル内の **commandPath** プロパティによって指定されたパス上になければなりません。

fte:metadata

call 操作に関連付けるメタデータを指定できます。このメタデータは、**call** 操作で生成されたログ・メッセージに記録されます。特定の転送エレメントには、単一のメタデータ・ブロックのみを関連付けることができます。ただし、このブロックには、多くのメタデータを含めることができます。

例

次の例では、キュー・マネージャー QM1 で実行されている AGENT1 でコマンドを呼び出す方法を示します。呼び出すコマンドはスクリプト **command.sh** であり、このスクリプトは単一の引数 **xyz** を指定して呼び出されます。コマンド **command.sh** は、エージェントの **agent.properties** ファイル内のコマンド・パス・プロパティで指定されたパスにあります。

```
<fte:call cmdqm="QM0@localhost@1414@SYSTEM.DEF.SVRCONN"
  agent="AGENT1@QM1"
  rcproperty="call.rc"
  origuser="bob"
  jobname="${ijob.id}">

  <fte:command command="command.sh" successrc="1" retrycount="5" retrywait="30">
    <fte:arg value="xyz"/>
  </fte:command>

  <fte:metadata>
    <fte:entry name="org.foo.accountName" value="BDG3R"/>
  </fte:metadata>

</fte:call>
```

関連タスク

[Apache Ant と MFT の併用](#)

fte:cancel の Ant タスク

Managed File Transfer 管理対象転送または管理対象呼び出しを取り消します。管理対象転送は、**fte:filecopy** タスクまたは **fte:filemove** タスクを使用して作成された可能性があります。管理対象呼び出しは、**fte:call** タスクを使用して作成された可能性があります。

属性

agent

必須。 **fte:cancel** 要求の実行依頼先のエージェントを指定します。値の形式は **agentname@qmgrname** です。ここで、**agentname** はエージェントの名前、**qmgrname** はこのエージェントが直接接続されているキュー・マネージャーの名前です。

cmdqm

オプション。要求の実行依頼先のコマンド・キュー・マネージャー。この情報は、`qmgrname@host@port@channel` の形式で指定します。ここで、

- `qmgrname` はキュー・マネージャーの名前です
- `host` は、キュー・マネージャーが実行されているシステムのオプションのホスト名です。
- `port` は、キュー・マネージャーが `listen` するオプションのポート番号です。
- `channel` は、使用するオプションの SVRCONN チャンネルです。

コマンド・キュー・マネージャーの `host`、`port`、または `channel` 情報を省略すると、`command.properties` ファイルに指定されている接続情報が使用されます。



重要: 値が指定されていない場合は、以下のようになります。

- ホスト変数、バインディング・モードが使用される
- ポート 変数。値 1414 が使用されます。
- チャンネル変数の場合は `SYSTEM.DEF.SVRCONN` が使用されます。

詳しくは、[MFT `command.properties` ファイル](#)を参照してください。

ただし、中央の属性 (`qmgrname@host@@channel` など) をスキップすることはできません。例えば、`qmgrname@host`、`qmgrname@host@port`、`qmgrname@hostport@@channel` のいずれかを指定できます。

MFT は、@区切り文字を使用して、指定された属性を分割します。検出されたトークンの数に応じて、最初のトークンは `qmgrname`、2 番目のトークンは `host`、3 番目のトークンは `port`、そして最後に `channel` となります。

詳しくは、[MFT `command.properties` ファイル](#)を参照してください。

com.ibm.wmqfte.propertySet プロパティを使用して、使用する `command.properties` ファイルを指定できます。詳細については、[com.ibm.wmqfte.propertySet](#) を参照してください。

`cmdqm` 属性を使用しない場合、タスクはデフォルトで

`com.ibm.wmqfte.ant.commandQueueManager` プロパティを使用します (このプロパティが設定されている場合)。`com.ibm.wmqfte.ant.commandQueueManager` プロパティが設定されていない場合、`command.properties` ファイルに定義されているデフォルト・キュー・マネージャーへの接続が試行されます。`com.ibm.wmqfte.ant.commandQueueManager` プロパティの形式は、`cmdqm` 属性と同じです。つまり、`qmgrname@host@port@channel` です。

ID

必須。取り消す転送の転送 ID を指定します。転送 ID は、転送要求が `fte:filecopy` タスクと `fte:filemove` タスクの両方によって実行依頼される時点で生成されます。

origuser

オプション。**cancel** 要求に関連付ける発信ユーザー ID を指定します。`origuser` 属性を使用しない場合、タスクはデフォルトで Ant スクリプトを実行するために使用されるユーザー ID を使用します。

例

次の例では、**fte:cancel** 要求をコマンド・キュー・マネージャー `qm0` に送信します。**fte:cancel** 要求のターゲットは、キュー・マネージャー `qm1` 上の `agent1` で、転送 ID は `transfer.id` 変数によって設定されています。この要求は、"bob" ユーザー ID を使用して実行されます。

```
<fte:cancel cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
  agent="agent1@qm1"
  id="${transfer.id}"
  origuser="bob"/>
```

関連タスク

[Apache Ant と MFT の併用](#)

fte:filecopy の Ant タスク

fte:filecopy タスクは、Managed File Transfer エージェント間でファイルをコピーします。ファイルはソース・エージェントから削除されません。

属性

cmdqm

オプション。要求の実行依頼先のコマンド・キュー・マネージャー。この情報は、`qmgrname@host@port@channel` の形式で指定します。ここで、

- `qmgrname` はキュー・マネージャーの名前です
- `host` は、キュー・マネージャーが実行されているシステムのオプションのホスト名です。
- `port` は、キュー・マネージャーが `listen` するオプションのポート番号です。
- `channel` は、使用するオプションの SVRCONN チャンネルです。

コマンド・キュー・マネージャーの `host`、`port`、または `channel` 情報を省略すると、`command.properties` ファイルに指定されている接続情報が使用されます。



重要: 値が指定されていない場合は、以下のようになります。

- ホスト変数、バインディング・モードが使用される
- ポート 変数。値 1414 が使用されます。
- チャンネル変数の場合は `SYSTEM.DEF.SVRCONN` が使用されます。

詳しくは、[MFT command.properties ファイル](#)を参照してください。

ただし、中央の属性 (`qmgrname@host@@channel` など) をスキップすることはできません。例えば、`qmgrname@host`、`qmgrname@host@port`、`qmgrname@hostport@@channel` のいずれかを指定できます。

MFT は、@区切り文字を使用して、指定された属性を分割します。検出されたトークンの数に応じて、最初のトークンは `qmgrname`、2 番目のトークンは `host`、3 番目のトークンは `port`、そして最後に `channel` となります。

詳しくは、[MFT command.properties ファイル](#)を参照してください。

com.ibm.wmqfte.propertySet プロパティを使用して、使用する `command.properties` ファイルを指定できます。詳細については、[com.ibm.wmqfte.propertySet](#) を参照してください。

`cmdqm` 属性を使用しない場合、タスクはデフォルトで

`com.ibm.wmqfte.ant.commandQueueManager` プロパティを使用します (このプロパティが設定されている場合)。`com.ibm.wmqfte.ant.commandQueueManager` プロパティが設定されていない場合、`command.properties` ファイルに定義されているデフォルト・キュー・マネージャーへの接続が試行されます。`com.ibm.wmqfte.ant.commandQueueManager` プロパティの形式は、`cmdqm` 属性と同じです。つまり、`qmgrname@host@port@channel` です。

dst

必須。copy 操作の宛先エージェントを指定します。`agentname@qmgrname` の形式で情報を指定します。ここで、`agentname` は宛先エージェントの名前、`qmgrname` はこのエージェントが直接接続されているキュー・マネージャーの名前です。

idproperty

`defer` の `outcome` を指定していない場合はオプションです。転送 ID を割り当てるプロパティの名前を指定します。転送 ID は、転送要求が実行依頼された時点で生成されます。この転送 ID を使用して、転送の進行の追跡、転送で生じた問題の診断、および転送の取り消しを行うことができます。

`ignore` の `outcome` プロパティも指定した場合は、このプロパティを指定できません。ただし、`defer` の `outcome` プロパティも指定した場合は、`idproperty` を指定する必要があります。

jobname

オプション。ジョブ名を copy 要求に割り当てます。ジョブ名を使用して、論理転送グループを作成できます。[2125 ページの『fte:uuid の Ant タスク』](#) タスクを使用して、疑似固有ジョブ名を生成します。jobname 属性を使用しない場合、タスクはデフォルトで com.ibm.wmqfte.ant.jobName プロパティー値を使用します (このプロパティーが設定されている場合)。このプロパティーが設定されていない場合には、copy 要求に関連付けられるジョブ名はありません。

origuser

オプション。copy 要求に関連付ける発信ユーザー ID を指定します。origuser 属性を使用しなかった場合には、タスクはデフォルトで Ant スクリプトを実行するために使用されるユーザー ID を使用します。

outcome

オプション。タスクが、Ant スクリプトに制御を返す前に、copy 操作が完了するまで待機するかどうかを決定します。以下のいずれかのオプションを指定します。

await

タスクは、戻る前に copy 操作が完了するまで待機します。await の outcome が指定されている場合、idproperty 属性はオプションです。

defer

タスクは、copy 要求が実行依頼されるとすぐに戻り、後から [2111 ページの『fte:awaitoutcome の Ant タスク』](#) タスクまたは [2123 ページの『fte:ignoreoutcome の Ant タスク』](#) タスクを使用して copy 操作の結果を処理することを想定します。defer の outcome が指定されている場合、idproperty 属性は必須です。

ignore

コピー操作の結果が重要でない場合は、ignore の値を指定できます。この値を指定した場合、タスクは、転送結果を追跡するためのリソースを割り当てずに、copy 要求が実行依頼されるとすぐに戻ります。ignore の outcome が指定されている場合、idproperty 属性を指定することはできません。

outcome 属性を指定しない場合、タスクはデフォルトで値 await を使用します。

priority

オプション。copy 要求に関連付ける優先順位を指定します。一般に、優先順位が高い転送要求が、優先順位が低い要求より優先されます。優先順位の値は、0 以上 9 以下の範囲で指定する必要があります。優先順位値 0 は最低の優先順位であり、値 9 は最高の優先順位です。priority 属性を指定しない場合、転送の優先順位はデフォルトの 0 になります。

rcproperty

オプション。copy 要求の結果コードを割り当てるプロパティーの名前を指定します。結果コードには、copy 要求の全体的結果が反映されます。

ignore または defer の outcome プロパティーも指定した場合は、このプロパティーを指定できません。ただし、await の結果を指定する場合は、rcproperty を指定する必要があります。

transferRecoveryTimeout

オプション。停止したファイル転送のリカバリーをソース・エージェントが試行し続ける時間 (秒単位) を設定します。以下のいずれかのオプションを指定します。

-1

エージェントは、停止した転送のリカバリーを、転送が完了するまで試行し続けます。このオプションを使用すると、このプロパティーを設定しない場合のエージェントのデフォルトの動作と同じになります。

0

エージェントは、リカバリーに入るとすぐにファイル転送を停止します。

>0

エージェントは、指定された正整数値で設定された時間 (秒単位) だけ、停止した転送のリカバリーを試行し続けます。例:

```
<fte:filecopy cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
  src="agent1@qm1" dst="agent2@qm2"
  rcproperty="copy.result" transferRecoveryTimeout="21600">
```

```
<fte:filespec srcfilespec="/home/fteuser1/file.bin" dstfile="/home/fteuser2/
file.bin"/>
</fte:filecopy>
```

これは、エージェントがリカバリーに入ってから 6 時間にわたって転送のリカバリーを試行し続けることを示しています。この属性の最大値は 999999999 です。

このように指定した場合、転送のリカバリー・タイムアウト値は転送単位で設定されます。Managed File Transfer ネットワーク内のすべての転送が対象になるグローバルな値を設定するには、プロパティを 転送リカバリー・タイムアウト・プロパティ に追加します。詳しくは、転送のリカバリーのタイムアウト・オプション を参照してください。

src

必須。copy 操作のソース・エージェントを指定します。この情報は、*agentname@qmgrname* という形式で指定します (ここで、*agentname* はソース・エージェントの名前、*qmgrname* は、当該エージェントが直接接続されている先のキュー・マネージャーの名前)。

ネスト・エレメントとして指定するパラメーター

fte:filespec

必須。コピーするファイルを識別するファイル指定を少なくとも 1 つ指定する必要があります。必要に応じて複数のファイル指定を指定できます。詳しくは、[2125 ページの『fte:filespec Ant のネストされたエレメント』](#) を参照してください。

fte:metadata

copy 操作に関連付けるメタデータを指定できます。このメタデータは転送とともに渡され、転送によって生成されたログ・メッセージに記録されます。特定の転送エレメントには、単一のメタデータ・ブロックのみを関連付けることができます。ただし、このブロックには、多くのメタデータを含めることができます。詳しくは、『[fte:metadata](#)』のトピックを参照してください。

fte:presrc

転送開始前にソース・エージェントで行うプログラム呼び出しを指定します。特定の転送に関連付けることができる **fte:presrc** エレメントは 1 つのみです。詳しくは、『[プログラム呼び出し](#)』のトピックを参照してください。

fte:predst

転送開始前に宛先エージェントで行うプログラム呼び出しを指定します。特定の転送に関連付けることができる **fte:predst** エレメントは 1 つのみです。詳しくは、『[プログラム呼び出し](#)』のトピックを参照してください。

fte:postsrc

転送完了後にソース・エージェントで行うプログラム呼び出しを指定します。特定の転送に関連付けることができる **fte:postsrc** エレメントは 1 つのみです。詳しくは、『[プログラム呼び出し](#)』のトピックを参照してください。

fte:postdst

転送完了後に宛先エージェントで行うプログラム呼び出しを指定します。特定の転送に関連付けることができる **fte:postdst** エレメントは 1 つのみです。詳しくは、『[プログラム呼び出し](#)』のトピックを参照してください。

fte:presrc、**fte:predst**、**fte:postsrc**、**fte:postdst**、および出口が成功状態に戻さない場合、規則では以下で指定された順序になります。

1. ソース開始出口を実行します。ソース開始出口が失敗すると、転送は失敗し、それ以降何も実行されません。
2. 事前ソース呼び出しを実行します (存在する場合)。事前ソース呼び出しが失敗すると、転送は失敗し、それ以降何も実行されません。
3. 宛先開始出口を実行します。宛先開始出口が失敗すると、転送は失敗し、それ以降何も実行されません。
4. 事前宛先呼び出しを実行します (存在する場合)。事前宛先呼び出しが失敗すると、転送は失敗し、それ以降何も実行されません。

5. ファイル転送を実行します。
6. 宛先終了出口を実行します。これらの出口に失敗状況はありません。
7. 正常に転送された場合 (一部のファイルが正常に転送され、転送が成功したと判断される場合) は、事後宛先呼び出しがあれば、それを実行します。事後宛先呼び出しが失敗すると、転送は失敗します。
8. ソース終了出口を実行します。これらの出口に失敗状況はありません。
9. 正常に転送された場合は、事後ソース呼び出しがあれば、それを実行します。事後ソース呼び出しが失敗すると、転送は失敗します。

例

この例は、agent1 と agent2 の間の基本的なファイル転送を示しています。ファイル転送を開始するコマンドは、クライアント・トランスポート・モード接続を使用して `qm0`、というキュー・マネージャーに送信されます。ファイル転送操作の結果は、`copy.result` というプロパティに割り当てられます。

```
<fte:filecopy cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
  src="agent1@qm1" dst="agent2@qm2"
  rcproperty="copy.result">

  <fte:filespec srcfilespec="/home/fteuser1/file.bin" dstfile="/home/fteuser2/file.bin"/>
</fte:filecopy>
```

この例は、同じファイル転送を示していますが、転送の完了後にソース・エージェントでメタデータとプログラムの追加が行われるようになります。

```
<fte:filecopy cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
  src="agent1@qm1" dst="agent2@qm2"
  rcproperty="copy.result">

  <fte:metadata>
    <fte:entry name="org.example.departId" value="ACCOUNTS"/>
    <fte:entry name="org.example.batchGroup" value="A1"/>
  </fte:metadata>

  <fte:filespec srcfilespec="/home/fteuser1/file.bin" dstfile="/home/fteuser2/file.bin"/>

  <fte:postsrc command="/home/fteuser2/scripts/post.sh" successsrc="1" >
    <fte:arg value="/home/fteuser2/file.bin"/>
  </fte:postsrc>
</fte:filecopy>
```

関連概念

[ファイル転送のリカバリーのタイムアウト・オプション](#)

関連タスク

[Apache Ant と MFT の併用](#)

fte:filemove の Ant タスク

fte:filemove タスクは、Managed File Transfer エージェント間でファイルを移動します。ファイルがソース・エージェントから宛先エージェントに正常に転送されると、そのファイルはソース・エージェントから削除されます。

属性

cmdqm

オプション。要求の実行依頼先のコマンド・キュー・マネージャー。この情報は、`qmgrname@host@port@channel` の形式で指定します。ここで、

- `qmgrname` はキュー・マネージャーの名前です
- `host` は、キュー・マネージャーが実行されているシステムのオプションのホスト名です。
- `port` は、キュー・マネージャーが `listen` するオプションのポート番号です。

- `channel` は、使用するオプションの SVRCONN チャンネルです。

コマンド・キュー・マネージャーの `host`、`port`、または `channel` 情報を省略すると、`command.properties` ファイルに指定されている接続情報が使用されます。



重要: 値が指定されていない場合は、以下のようになります。

- ホスト変数、バインディング・モードが使用される
- ポート 変数。値 1414 が使用されます。
- チャンネル変数の場合は `SYSTEM.DEF.SVRCONN` が使用されます。

詳しくは、[MFT command.properties ファイル](#)を参照してください。

ただし、中央の属性 (`qmgrname@host@@channel` など) をスキップすることはできません。例えば、`qmgrname@host`、`qmgrname@host@port`、`qmgrname@hostport@@channel` のいずれかを指定できます。

MFT は、@区切り文字を使用して、指定された属性を分割します。検出されたトークンの数に応じて、最初のトークンは `qmgrname`、2 番目のトークンは `host`、3 番目のトークンは `port`、そして最後に `channel` となります。

詳しくは、[MFT command.properties ファイル](#)を参照してください。

com.ibm.wmqfte.propertySet プロパティを使用して、使用する `command.properties` ファイルを指定できます。詳細については、[com.ibm.wmqfte.propertySet](#) を参照してください。

`cmdqm` 属性を使用しない場合、タスクはデフォルトで

`com.ibm.wmqfte.ant.commandQueueManager` プロパティを使用します (このプロパティが設定されている場合)。`com.ibm.wmqfte.ant.commandQueueManager` プロパティが設定されていない場合、`command.properties` ファイルに定義されているデフォルト・キュー・マネージャーへの接続が試行されます。`com.ibm.wmqfte.ant.commandQueueManager` プロパティの形式は、`cmdqm` 属性と同じです。つまり、`qmgrname@host@port@channel` です。

dst

必須。copy 操作の宛先エージェントを指定します。`agentname@qmgrname` の形式で情報を指定します。ここで、`agentname` は宛先エージェントの名前、`qmgrname` はこのエージェントが直接接続されているキュー・マネージャーの名前です。

idproperty

`defer` の `outcome` を指定していない場合はオプションです。転送 ID を割り当てるプロパティの名前を指定します。転送 ID は、転送要求が実行依頼された時点で生成されます。この転送 ID を使用して、転送の進行の追跡、転送で生じた問題の診断、および転送の取り消しを行うことができます。

`ignore` の `outcome` プロパティも指定した場合は、このプロパティを指定できません。ただし、`defer` の `outcome` プロパティも指定した場合は、`idproperty` を指定する必要があります。

jobname

オプション。ジョブ名を `move` 要求に割り当てます。ジョブ名を使用して、論理転送グループを作成できます。`fte:uuid` タスクを使用して、疑似固有ジョブ名を生成します。`jobname` 属性を使用しない場合、タスクはデフォルトで `com.ibm.wmqfte.ant.jobName` プロパティ値を使用します (このプロパティが設定されている場合)。このプロパティが設定されていない場合には、`move` 要求に関連付けられるジョブ名はありません。

origuser

オプション。`move` 要求に関連付ける発信ユーザー ID を指定します。`origuser` 属性を使用しなかった場合には、タスクはデフォルトで Ant スクリプトを実行するために使用されるユーザー ID を使用します。

outcome

オプション。タスクが、Ant スクリプトに制御を返す前に、`move` 操作が完了するまで待機するかどうかを決定します。以下のいずれかのオプションを指定します。

await

タスクは、戻る前に `move` 操作が完了するまで待機します。`await` の `outcome` が指定されている場合、`idproperty` 属性はオプションです。

defer

タスクは、move 要求が実行依頼されるとすぐに戻り、後から 2111 ページの『[fte:awaitoutcome の Ant タスク](#)』タスクまたは 2123 ページの『[fte:ignoreoutcome の Ant タスク](#)』タスクを使用して move 操作の結果を処理することを想定します。defer の outcome が指定されている場合、idproperty 属性は必須です。

ignore

移動操作の結果が重要でない場合は、ignore の値を指定できます。この値を指定した場合、タスクは、転送結果を追跡するためのリソースを割り当てずに、move 要求が実行依頼されるとすぐに戻ります。ignore の outcome が指定されている場合、idproperty 属性を指定することはできません。

outcome 属性を指定しない場合、タスクはデフォルトで値 await を使用します。

priority

オプション。move 要求に関連付ける優先順位を指定します。一般に、優先順位が高い転送要求が、優先順位が低い要求より優先されます。優先順位の値は、0 以上 9 以下の範囲で指定する必要があります。優先順位値 0 は最低の優先順位であり、値 9 は最高の優先順位です。priority 属性を指定しない場合、転送の優先順位はデフォルトの 0 になります。

rcproperty

オプション。move 要求の結果コードを割り当てるプロパティの名前を指定します。結果コードには、move 要求の全体的結果が反映されます。

ignore または defer の outcome プロパティも指定した場合は、このプロパティを指定できません。ただし、await の結果を指定した場合は、rcproperty を指定する必要があります。

transferRecoveryTimeout

オプション。停止したファイル転送のリカバリーをソース・エージェントが試行し続ける時間 (秒単位) を設定します。以下のいずれかのオプションを指定します。

-1

エージェントは、停止した転送のリカバリーを、転送が完了するまで試行し続けます。このオプションを使用すると、このプロパティを設定しない場合のエージェントのデフォルトの動作と同じになります。

0

エージェントは、リカバリーに入るとすぐにファイル転送を停止します。

>0

エージェントは、指定された正整数値で設定された時間 (秒単位) だけ、停止した転送のリカバリーを試行し続けます。例:

```
<fte:filemove cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
src=agent1@qm1 dst="agent2@qm2"
rcproperty="move.result" transferRecoveryTimeout="21600">

  <fte:filespec srcfilespec="/home/fteuser1/file.bin" dstfile="/home/fteuser2/
file.bin"/>

</fte:filemove
```

これは、エージェントがリカバリーに入ってから 6 時間にわたって転送のリカバリーを試行し続けることを示しています。この属性の最大値は 999999999 です。

このように指定した場合、転送のリカバリー・タイムアウト値は転送単位で設定されます。Managed File Transfer ネットワーク内のすべての転送が対象になるグローバルな値を設定するには、プロパティを [転送リカバリー・タイムアウト・プロパティ](#) に追加します。詳しくは、[転送のリカバリーのタイムアウト・オプション](#)を参照してください。

src

必須。move 操作のソース・エージェントを指定します。この情報は agentname@qmgrname の形式で指定します。ここで、agentname はソース・エージェントの名前、qmgrname はこのエージェントが直接接続されているキュー・マネージャーの名前です。

ネスト・エレメントとして指定するパラメーター

fte:filespec

必須。移動対象ファイルを特定する少なくとも1つのファイル指定を指定する必要があります。必要に応じて複数のファイル指定を指定できます。詳しくは、[2125 ページの『fte:filespec Ant のネストされたエレメント』](#)を参照してください。

fte:metadata

オプション。ファイルの move 操作に関連付けるメタデータを指定できます。このメタデータは転送とともに渡され、転送によって生成されたログ・メッセージに記録されます。特定の転送エレメントには、単一のメタデータ・ブロックのみを関連付けることができます。ただし、このブロックには、多くのメタデータを含めることができます。詳しくは、『[fte:metadata](#)』のトピックを参照してください。

fte:presrc

オプション。転送開始前にソース・エージェントで行うプログラム呼び出しを指定します。特定の転送に関連付けることができる **fte:presrc** エレメントは1つのみです。詳しくは、『[プログラム呼び出し](#)』のトピックを参照してください。

fte:predst

オプション。転送開始前に宛先エージェントで行うプログラム呼び出しを指定します。特定の転送に関連付けることができる **fte:predst** エレメントは1つのみです。詳しくは、『[プログラム呼び出し](#)』のトピックを参照してください。

fte:postsrc

オプション。転送完了後にソース・エージェントで行うプログラム呼び出しを指定します。特定の転送に関連付けることができる **fte:postsrc** エレメントは1つのみです。詳しくは、『[プログラム呼び出し](#)』のトピックを参照してください。

fte:postdst

オプション。転送完了後に宛先エージェントで行うプログラム呼び出しを指定します。特定の転送に関連付けることができる **fte:postdst** エレメントは1つのみです。詳しくは、『[プログラム呼び出し](#)』のトピックを参照してください。

fte:presrc、**fte:predst**、**fte:postsrc**、**fte:postdst**、および出口が成功状態に戻さない場合、規則では以下で指定された順序になります。

1. ソース開始出口を実行します。ソース開始出口が失敗すると、転送は失敗し、それ以降何も実行されません。
2. 事前ソース呼び出しを実行します (存在する場合)。事前ソース呼び出しが失敗すると、転送は失敗し、それ以降何も実行されません。
3. 宛先開始出口を実行します。宛先開始出口が失敗すると、転送は失敗し、それ以降何も実行されません。
4. 事前宛先呼び出しを実行します (存在する場合)。事前宛先呼び出しが失敗すると、転送は失敗し、それ以降何も実行されません。
5. ファイル転送を実行します。
6. 宛先終了出口を実行します。これらの出口に失敗状況はありません。
7. 正常に転送された場合 (一部のファイルが正常に転送され、転送が成功したと判断される場合) は、事後宛先呼び出しがあれば、それを実行します。事後宛先呼び出しが失敗すると、転送は失敗します。
8. ソース終了出口を実行します。これらの出口に失敗状況はありません。
9. 正常に転送された場合は、事後ソース呼び出しがあれば、それを実行します。事後ソース呼び出しが失敗すると、転送は失敗します。

例

この例は、**agent1** と **agent2** の間の基本的なファイル移動を示しています。ファイル移動を開始するコマンドは、クライアント・トランスポート・モード接続を使用して **qm0**、というキュー・マネージャーに送信されます。ファイル転送操作の結果は、**move.result** というプロパティに割り当てられます。

```
<fte:filemove cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
src="agent1@qm1" dst="agent2@qm2"
```

```
rcproperty="move.result">

<fte:filespec srcfilespec="/home/fteuser1/file.bin" dstfile="/home/fteuser2/file.bin"/>

</fte:filemove>
```

関連概念

[ファイル転送のリカバリーのタイムアウト・オプション](#)

関連タスク

[Apache Ant と MFT の併用](#)

fte:ignoreoutcome の Ant タスク

fte:filecopy、**fte:filemove**、または **fte:call** コマンドの結果を無視します。**fte:filecopy**、**fte:filemove**、または **fte:call** タスクで `defer` の結果を得るように指定すると、Ant タスクでこの結果を追跡するためのリソースが割り振られます。この結果がもはや必要ない場合は、**fte:ignoreoutcome** タスクでこうしたリソースを解放することができます。

属性

ID

必須。もはや必要ない出力を識別します。通常、この ID は、[2116 ページの『fte:filecopy の Ant タスク』](#)、[2119 ページの『fte:filemove の Ant タスク』](#)、または [2112 ページの『fte:call の Ant タスク』](#) タスクの `idproperty` 属性を使用して設定するプロパティを使用して指定します。

例

この例では、`fte:ignoreoutcome` タスクを使用して、以前の [2116 ページの『fte:filecopy の Ant タスク』](#) タスクの出力を追跡するために割り振られていたリソースをどのように解放できるかを示します。

```
<!-- issue a file copy request -->
<fte:filecopy cmdqm="qm1@localhost@1414@SYSTEM.DEF.SVRCONN"
  src="agent1@qm1" dst="agent1@qm1"
  idproperty="copy.id"
  outcome="defer"/>

<!-- do some other things -->

<!-- decide that the result of the copy is not interesting -->
<fte:ignoreoutcome id="${copy.id}"/>
```

関連タスク

[Apache Ant と MFT の併用](#)

fte:ping の Ant タスク

この IBM MQ Managed File Transfer Ant タスクでは、エージェントに ping を送信して応答を引き出し、そのエージェントが転送を処理できるかどうかを確認します。

注：IBM WebSphere MQ File Transfer Edition (FTE) は、サポート対象外となりました。IBM MQ で FTE から Managed File Transfer コンポーネントにマイグレーションする場合は、[Managed File Transfer のマイグレーション](#)を参照してください。

属性

agent

必須。**fte:ping** 要求の送信先のエージェントを指定します。値の形式は `agentname@qmgrname` です。ここで、`agentname` はエージェントの名前、`qmgrname` はこのエージェントが直接接続されているキュー・マネージャーの名前です。

cmdqm

オプション。要求の実行依頼先のコマンド・キュー・マネージャー。この情報は、`qmgrname@host@port@channel` の形式で指定します。ここで、

- `qmgrname` はキュー・マネージャーの名前です
- `host` は、キュー・マネージャーが実行されているシステムのオプションのホスト名です。
- `port` は、キュー・マネージャーが `listen` するオプションのポート番号です。
- `channel` は、使用するオプションの SVRCONN チャンネルです。

コマンド・キュー・マネージャーの `host`、`port`、または `channel` 情報を省略すると、`command.properties` ファイルに指定されている接続情報が使用されます。



重要: 値が指定されていない場合は、以下のようになります。

- ホスト変数、バインディング・モードが使用される
- ポート 変数。値 1414 が使用されます。
- チャンネル変数の場合は `SYSTEM.DEF.SVRCONN` が使用されます。

詳しくは、[MFT command.properties ファイル](#)を参照してください。

ただし、中央の属性 (`qmgrname@host@@channel` など) をスキップすることはできません。例えば、`qmgrname@host`、`qmgrname@host@port`、`qmgrname@hostport@@channel` のいずれかを指定できます。

MFT は、@区切り文字を使用して、指定された属性を分割します。検出されたトークンの数に応じて、最初のトークンは `qmgrname`、2 番目のトークンは `host`、3 番目のトークンは `port`、そして最後に `channel` となります。

詳しくは、[MFT command.properties ファイル](#)を参照してください。

com.ibm.wmqfte.propertySet プロパティを使用して、使用する `command.properties` ファイルを指定できます。詳細については、[com.ibm.wmqfte.propertySet](#) を参照してください。

`cmdqm` 属性を使用しない場合、タスクはデフォルトで

`com.ibm.wmqfte.ant.commandQueueManager` プロパティを使用します (このプロパティが設定されている場合)。`com.ibm.wmqfte.ant.commandQueueManager` プロパティが設定されていない場合、`command.properties` ファイルに定義されているデフォルト・キュー・マネージャーへの接続が試行されます。`com.ibm.wmqfte.ant.commandQueueManager` プロパティの形式は、`cmdqm` 属性と同じです。つまり、`qmgrname@host@port@channel` です。

rcproperty

必須。ping 操作の戻りコードを保管するためのプロパティの名前を指定します。

timeout

オプション。タスクがエージェントの応答を待つ最大時間 (秒単位)。最小タイムアウトは 0 秒ですが、エージェントが応答するまでコマンドは永久に待つようにする場合はタイムアウトにマイナス 1 を指定することもできます。timeout に値が指定されていない場合、デフォルトでは、エージェントが応答するまで最大 5 秒待機します。

例

この例では、**fte:ping** 要求を `qm1` によってホスティングされる `agent1` に送信します。**fte:ping** 要求では、エージェントの応答を 15 秒間待機します。**fte:ping** 要求の結果は、`ping.rc` という名前のプロパティに保管されます。

```
<fte:ping agent="agent1@qm1" rcproperty="ping.rc" timeout="15"/>
```

戻りコード

0

コマンドは正常に完了しました。

2

コマンドはタイムアウトになりました。

関連タスク

[Apache Ant と MFT の併用](#)

fte:uuid の Ant タスク

疑似乱数固有 ID を生成して、所定のプロパティに割り当てます。例えば、この ID を使用して、その他のファイル転送操作のジョブ名を生成できます。

属性

長さ (length)

必須。生成する UUID の長さを示す数値。この長さの値には、**prefix** パラメーターで指定された接頭部の長さは含まれません。

プロパティ

必須。生成された UUID を割り当てるプロパティの名前。

PREFIX

オプション。生成された UUID に付加する接頭部。この接頭部は、**length** パラメーターで指定された UUID の長さの一部としてはカウントされません。

例

この例では、文字 ABC で始まり、16 個の疑似乱数 16 進文字が続く UUID を定義します。UUID は、`uuid.property` という名前のプロパティに割り当てられます。

```
<fte:uuid length="16" property="uuid.property" prefix="ABC"/>
```

関連タスク

[Apache Ant と MFT の併用](#)

fte:filespec Ant のネストされたエレメント

fte:filespec パラメーターは、他のタスクでネストされたエレメントとして使用されます。

fte:filespec を使用して、1 つ以上のソース・ファイル、ディレクトリー  **z/os** またはデータ・セット、および宛先の間のマッピングを記述します。一般的には、このエレメントは、移動またはコピーする一連のファイル、ディレクトリー  **z/os**、またはデータ・セットを表す場合に使用されます。

ネスト対象タスク:

- [fte:filecopy](#) タスク
- [fte:filemove](#) タスク

ソース指定属性

`srcfilespec` または `srcqueue` のいずれかを指定する必要があります。

srcfilespec

ファイル操作のソースを指定します。この属性の値は、ワイルドカードを含むことができます。

srcqueue

転送のソースがキューであることを指定します。転送すると、この属性によって指定されたキューに保管されているメッセージからデータが移動します。**fte:filespec** タスクが **fte:filecopy** タスク内にネストされている場合は、この属性を指定できません。

`srcqueue` 属性は、ソース・エージェントがプロトコル・ブリッジ・エージェントである場合はサポートされません。

宛先指定属性

dstdir、dstds、dstfilespace、dstfile、dstqueue、または dstpds のいずれかを指定する必要があります。

dstdir

ファイル操作の宛先としてディレクトリーを指定します。

dstds

ファイル操作の宛先としてデータ・セットを指定します。

この属性がサポートされるのは、宛先エージェントが z/OS プラットフォームで実行中の場合のみです。

dstfile

ファイル操作の宛先としてファイルを指定します。

dstfilespace

ファイル操作の宛先としてファイル・スペースを指定します。

この属性が適用されるのは、宛先エージェントが、Web ゲートウェイ・ファイル・スペースに対するアクセス権限を持つ IBM MQ 8.0 Web エージェントである場合のみです。

dstpds

ファイル操作の宛先として区分データ・セットを指定します。

この属性がサポートされるのは、宛先エージェントが z/OS プラットフォームで実行中の場合のみです。

dstqueue

ファイルからメッセージ操作の宛先としてキューを指定します。この指定に QUEUE@QUEUEMANAGER というフォーマットでキュー・マネージャー名を含めることもできます。キュー・マネージャーを指定しない場合は、enableClusterQueueInputOutput エージェント・プロパティを true に設定しない限り、宛先エージェントのキュー・マネージャーが使用されます。enableClusterQueueInputOutput プロパティが true に設定されている場合、宛先エージェントは、標準的な IBM MQ 手順を使用して、キューが配置されている場所を判別します。対象のキュー・マネージャーに存在する有効なキュー名を指定する必要があります。

dstqueue 属性を指定する場合、srcqueue 属性は指定できません。これらの属性は相互に排他的です。

dstqueue 属性は、宛先エージェントがプロトコル・ブリッジ・エージェントである場合はサポートされません。

ソース・オプション属性

srcencoding

オプション。転送するファイルで使用する文字セット・エンコード方式。

この属性を指定できるのは、変換属性の値が text. に設定されている場合のみです。

srcencoding 属性を指定しない場合、テキスト転送にはソース・システムの文字セットが使用されます。

srceol

オプション。転送されるファイルで使用する行の終わり区切り文字。有効な値は以下のとおりです。

- CRLF - 復帰文字とそれに続く改行文字を行の終わり区切り文字として使用します。この規則は、Windows システムの場合の標準です。
- LF - 行末区切り文字として改行文字を使用します。この規則は、UNIX システムの場合の標準です。

この属性を指定できるのは、変換属性の値が text に設定されている場合のみです。srceol 属性を指定しない場合、ソース・エージェントのオペレーティング・システムに基づいて、テキスト転送で正しい値が自動的に決定されます。

srckeeptrailingspaces

オプション。テキスト・モード転送の一部として固定長形式のデータ・セットから読み取られるソース・レコードの末尾スペースを、保持するかどうかを決定します。有効な値は以下のとおりです。

- true - 末尾のスペースは保持されます。

- **false** - 末尾のスペースは除去されます。

srckeeptrailingspaces 属性を指定しない場合、デフォルト値 **false** が指定されます。

この属性を指定できるのは、**srcfilespec** 属性も指定し、変換属性の値を **text.** に設定した場合のみです。

srcmsgdelimbytes

オプション。1つのバイナリー・ファイルに複数のメッセージを追加するときに区切り文字として挿入する1つ以上のバイト値を指定します。それぞれの値は、**x** という接頭部を付けた **00** から **FF** の範囲の2桁の16進数字として指定する必要があります。複数バイトの場合はコンマで区切る必要があります。例えば、**srcmsgdelimbytes="x08,xA4"** などです。**srcmsgdelimbytes** 属性を指定できるのは、**srcqueue** 属性も指定した場合に限られます。変換属性に値 **text** も指定した場合は、**srcmsgdelimbytes** 属性を指定できません。

srcmsgdelimtext

オプション。1つのテキスト・ファイルに複数のメッセージを追加するときに区切り文字として挿入するテキストのシーケンスを指定します。ストリング・リテラルの Java エスケープ・シーケンスを区切り文字に含めることもできます。例えば、**srcmsgdelimtext="\u007d\n"** などです。ソース・エージェントによって各メッセージの後にテキスト区切り文字が挿入されます。テキスト区切り文字は、転送のソース・エンコード方式に基づいてバイナリー・フォーマットにエンコードされます。各メッセージがバイナリー・フォーマットで読み取られ、エンコードされた区切り文字がバイナリー・フォーマットでメッセージに追加され、結果がバイナリー・フォーマットで宛先エージェントに転送されます。ソース・エージェントのコード・ページにシフトイン状態とシフトアウト状態が含まれていれば、エージェントは、各メッセージがメッセージの末尾でシフトアウト状態になると想定します。宛先エージェントでは、ファイルからファイルへのテキスト転送の場合と同じ要領でバイナリー・データが変換されます。**srcmsgdelimtext** 属性を指定できるのは、変換属性に **srcqueue** 属性と **text** の値も指定した場合のみです。

srcmsgdelimposition

オプション。テキストまたはバイナリー区切り文字が挿入される位置を指定します。有効な値は以下のとおりです。

- **prefix** - 区切り文字は、宛先ファイルの各メッセージからのデータの前に挿入されます。
- **postfix** - 区切り文字は、宛先ファイルの各メッセージのデータの後に挿入されます。

srcmsgdelimposition 属性を指定できるのは、**srcmsgdelimbytes** または **srcmsgdelimtext** 属性のいずれかをも指定した場合に限られます。

srcmsggroups

オプション。メッセージを IBM MQ グループ ID によってグループ化する動作を指定します。完全に揃った最初のグループが宛先ファイルに書き込まれます。この属性を指定しない場合は、ソース・キューに存在するすべてのメッセージが宛先ファイルに書き込まれます。**srcmsggroups** 属性を指定できるのは、**srcqueue** 属性も指定した場合に限られます。

srcqueuetimeout

オプション。以下のいずれかの条件が満たされるのを待つ時間を秒単位で指定します。

- 新しいメッセージがキューに書き込まれるという条件。
- **srcmsggroups** 属性を指定した場合は、完全に揃ったグループがキューに書き込まれるという条件。

srcqueuetimeout の値で指定した時間内にどちらの条件も満たされなければ、ソース・エージェントは、キューからの読み取りを停止して、転送を完了します。**srcqueuetimeout** 属性を指定しなければ、ソース・エージェントは、ソース・キューが空の場合にソース・キューからの読み取りをただちに停止します。あるいは、**srcmsggroups** 属性が指定されているのであれば、完全に揃ったグループがキューに存在しない場合にソース・キューからの読み取りをただちに停止します。**srcqueuetimeout** 属性を指定できるのは、**srcqueue** 属性も指定した場合に限られます。

srcqueuetimeout 値の設定については、[メッセージからファイルへの転送の待機時間を指定するためのガイダンス](#)を参照してください。

srcrcdelimbytes

オプション。1つのバイナリー・ファイルにレコード単位のソース・ファイルから複数のレコードを追加するときに区切り文字として挿入する、1つ以上のバイト値を指定します。それぞれの値は、接頭部 `x` を付けた 00 から FF の範囲の 2 桁の 16 進数字として指定する必要があります。複数バイトの場合はコンマで区切る必要があります。以下に例を示します。

```
srcrcdelimbytes="x08,xA4"
```

`srcrcdelimbytes` 属性は、転送のソース・ファイルがレコード単位 (例えば `z/OS` データ・セットなど) で、かつ、宛先ファイルがレコード単位ではない通常のファイルの場合にのみ指定できます。変換属性に値 `text` も指定した場合は、`srcrcdelimbytes` 属性を指定できません。

srcrcdelimpos

オプション。バイナリー区切り文字が挿入される位置を指定します。有効な値は以下のとおりです。

- `prefix` - 区切り文字は、宛先ファイルの、ソースのレコード単位ファイルの各レコードからのデータの前に挿入されます。
- `postfix` - 区切り文字は、宛先ファイルの、ソースのレコード単位ファイルの各レコードからのデータの後に挿入されます。

`srcrcdelimpos` 属性を指定できるのは、`srcrcdelimbytes` 属性も指定した場合に限られます。

宛先オプション属性

dstencoding

オプション。転送されるファイルに使用する文字セット・エンコード方式。

この属性を指定できるのは、変換属性の値が `text`. に設定されている場合のみです。

`dstencoding` 属性を指定しない場合、テキスト転送には宛先システムの文字セットが使用されます。

dsteol

オプション。転送されるファイルに使用する行の終わり区切り文字。有効な値は以下のとおりです。

- `CRLF` - 復帰文字とそれに続く改行文字を行の終わり区切り文字として使用します。この規則は、Windows システムの場合の標準です。
- `LF` - 行末区切り文字として改行文字を使用します。この規則は、UNIX システムの場合の標準です。

この属性を指定できるのは、変換属性の値が `text`. に設定されている場合のみです。

`dsteol` 属性を指定しない場合、宛先エージェントのオペレーティング・システムに基づいて、テキスト転送で正しい値が自動的に決定されます。

dstmsgdelimbytes

オプション。バイナリー・ファイルを複数のメッセージに分割するときに使用する 16 進数区切り文字を指定します。メッセージは、すべて同じ IBM MQ グループ ID を持ちます。グループの最後のメッセージは IBM MQ `LAST_MSG_IN_GROUP` フラグ・セットを持ちます。区切り文字として 16 進バイトを指定するための形式は、`xNN` です。ここで、`N` は 0 から 9 または `a` から `f` の範囲の文字です。16 進バイトのコンマ区切りリストを指定することにより、16 進バイトのシーケンスを区切り文字として指定できます (例: `x3e,x20,x20,xbf`)。

`dstmsgdelimbytes` 属性を指定できるのは、バイナリー・モードの転送で `dstqueue` 属性も指定した場合に限られます。`dstmsgsize`、`dstmsgdelimbytes`、および `dstmsgdelimpattern` 属性のいずれか 1 つのみを指定できます。

dstmsgdelimpattern

オプション。テキスト・ファイルを複数のメッセージに分割するときに使用する Java 正規表現を指定します。メッセージは、すべて同じ IBM MQ グループ ID を持ちます。グループの最後のメッセージは IBM MQ `LAST_MSG_IN_GROUP` フラグ・セットを持ちます。正規表現を区切り文字として指定するための形式は、括弧で囲んだ正規表現、(`regular_expression`)、または二重引用符で囲んだ正規表現、`"regular_expression"` です。詳しくは、[MFT が使用する正規表現](#)を参照してください。

デフォルトでは、正規表現にマッチング可能なストリングの長さは、宛先エージェントによって 5 文字に制限されています。この動作は、**maxDelimiterMatchLength** エージェント・プロパティを使用して変更できます。詳しくは、[MFT 拡張エージェント・プロパティ](#)を参照してください。

dstmsgdelimpattern 属性を指定できるのは、テキスト・モードの転送で dstqueue 属性も指定した場合に限られます。dstmsgsize、dstmsgdelimbytes、および dstmsgdelimpattern 属性のいずれか 1 つのみを指定できます。

dstmsgdelimposition

オプション。テキストまたはバイナリー区切り文字が入ると想定される位置を指定します。有効な値は以下のとおりです。

- **prefix** - 各行の先頭に区切り文字が必要です。
- **postfix** - 各行の末尾に区切り文字が必要です。

dstmsgdelimposition 属性を指定できるのは、dstmsgdelimpattern 属性も指定した場合に限られます。

dstmsgincludedelim

オプション。ファイルを複数のメッセージに分割するために使用する区切り文字をそれらのメッセージに組み込むかどうかを指定します。dstmsgincludedelim 属性を指定すると、区切り文字の前にあるファイル・データが含まれているメッセージの末尾に区切り文字が組み込まれます。デフォルトでは、メッセージに区切り文字は組み込まれません。dstmsgincludedelim 属性を指定できるのは、dstmsgdelimpattern および dstmsgdelimbytes 属性のいずれかをも指定した場合に限られます。

dstmsgpersist

オプション。宛先キューに書き込むメッセージを永続メッセージにするかどうかを指定します。有効な値は以下のとおりです。

- **true** - 永続メッセージを宛先キューに書き込みます。これがデフォルト値です。
- **false** - 非持続メッセージを宛先キューに書き込みます。
- **qdef** - パーシスタンス値は、宛先キューの DefPersistence 属性から取得されます。

この属性を指定できるのは、dstqueue 属性も指定されている場合のみです。

dstmsgprops

オプション。転送で宛先キューに書き込む最初のメッセージで IBM MQ メッセージ・プロパティを設定するかどうかを指定します。指定可能な値は以下のとおりです。

- **true** - 転送によって作成される最初のメッセージにメッセージ・プロパティを設定します。
- **false** - 転送によって作成される最初のメッセージにメッセージ・プロパティを設定しません。これがデフォルト値です。

詳しくは、[MFT が宛先キューに書き込むメッセージで設定する MQ メッセージ・プロパティ](#)を参照してください。

この属性を指定できるのは、dstqueue 属性も指定されている場合のみです。

dstmsgsize

オプション。ファイルを複数の固定長メッセージに分割するかどうかを指定します。すべてのメッセージは、同じ IBM MQ グループ ID を持ちます。グループの最後のメッセージには IBM MQ LAST_MSG_IN_GROUP フラグが設定されます。メッセージのサイズは、dstmsgsize の値で指定します。dstmsgsize のフォーマットは、*lengthunits* です (*length* は正整数値、*units* は以下のいずれかの値です)。

- **B** - バイト。指定できる最小値は、宛先メッセージのコード・ページの 1 文字あたりの最大バイト数の値の 2 倍です。
- **K** - キビバイト。1024 バイトに相当します。
- **M** - メビバイト。1024 キビバイトに相当します。

ファイルをテキスト・モードで転送していて、ファイルが 2 バイト文字セットまたはマルチバイト文字セットのファイルである場合、そのファイルは、指定のメッセージ・サイズに最も近い文字境界で複数のメッセージに分割されます。

dstmsgsize 属性を指定できるのは、dstqueue 属性も指定した場合に限られます。dstmsgsize、dstmsgdelimbytes、および dstmsgdelimpattern 属性のいずれか 1 つのみを指定できます。

dstunsupportedcodepage

オプション。dstqueue 属性で指定された宛先キュー・マネージャーが、テキスト転送としてファイル・データをキューに転送する際に使用されるコード・ページをサポートしていない場合に実行するアクションを指定します。この属性で有効な値は以下のとおりです。

- **binary** - 転送を続行しますが、転送されるデータにコード・ページ変換を適用しません。この値を指定することは、変換属性を **text** に設定しないことと同等です。
- **fail** - 転送操作を続行しません。ファイルは転送に失敗したものとして記録されます。これはデフォルトです。

dstunsupportedcodepage 属性を指定できるのは、dstqueue 属性も指定し、conversion 属性の値として **text** を指定した場合に限られます。

dsttruncaterecords

オプション。LRECL データ・セット属性よりも長い宛先レコードが切り捨てられることを指定します。**true** に設定すると、それらのレコードは切り捨てられます。**false** に設定すると、それらのレコードは折り返されます。デフォルト設定は **false** です。このパラメーターは、宛先がデータ・セットであるテキスト・モードの転送のみに有効です。

その他の属性

checksum

オプション。転送されたファイルのチェックサムを計算するために使用されるアルゴリズムを決定します。

- **MD5** - MD5 ハッシュ・アルゴリズムを使用します。
- **NONE** - チェックサム・アルゴリズムを使用しません。

チェックサム属性を指定しない場合は、デフォルト値の **MD5** が使用されます。

conversion

オプション。ファイル転送の際にファイルに適用される変換のタイプを指定します。指定可能な値は以下のとおりです。

- **binary** - 変換を適用しません。
- **text** - ソース・システムと宛先システムの間でコード・ページ変換を適用します。行区切り文字も変換されます。srcencoding、dstencoding、srceol、および dsteol の各属性は、適用される変換に影響します。

変換属性を指定しない場合、デフォルト値 **binary** が指定されます。

overwrite

オプション。操作で既存の宛先ファイル  またはデータ・セットを上書きできるかどうかを決定します。値 **true** を指定すると、既存の宛先ファイル  またはデータ・セットが上書きされます。値 **false** を指定すると、宛先に重複ファイル  またはデータ・セットが存在するため、操作が失敗します。上書き属性が指定されていない場合、デフォルト値 **false** が指定されます。

recurse

オプション。サブディレクトリーでファイル転送を繰り返すかどうかを決定します。**true** の値を指定すると、サブディレクトリーへの転送が繰り返されます。**false** の値を指定すると、サブディレクトリーへの転送は繰り返されません。再帰属性が指定されていない場合、デフォルト値 **false** が指定されます。

例

この例では、file1.bin のソース・ファイルと file2.bin. の宛先ファイルを指定して fte:filespec を指定します。

```
<fte:filespec srcfilespec="/home/fteuser/file1.bin" dstfile="/home/fteuser/file2.bin"/>
```

関連タスク

[Apache Ant と MFT の併用](#)

fte:metadata Ant のネストされたエレメント

メタデータは追加のユーザー定義情報を保持します。特に **fte:call** のようなタスクでは、ファイル転送を伴わない場合もあります。

詳細については、MFT ユーザー・エグジット 2135 ページの『[MFT ユーザー出口のメタデータ](#)』でのメタデータの使用 Managed File Transfer 方法に関する情報をご覧ください。

ネスト対象タスク:

- [fte:filecopy](#) タスク
- [fte:filemove](#) タスク
- [fte:call](#) タスク

ネスト・エレメントとして指定するパラメーター

fte:entry

fte:metadata ネストされた要素内に少なくとも 1 つの項目を指定する必要があります。複数の entry を指定することもできます。entry はキー名を値に関連付けます。キーは fte:metadata のブロック内で固有でなければなりません。

entry の属性

name

必須。当該 entry に所属するキーの名前。この名前は、fte:metadata エレメント内にネストされているすべての **entry** パラメーターで固有でなければなりません。

値

必須。当該 entry に割り当てる値。

例

この例は、2 つの項目を含む fte:metadata 定義を示しています

```
<fte:metadata>
  <fte:entry name="org.foo.partColor" value="red"/>
  <fte:entry name="org.foo.partSize" value="medium"/>
</fte:metadata>
```

例

この例は、. fte:call task 内のメタデータの使用を示しています。このタスクはファイル転送を実行しませんが、メタデータを保持することは可能です。

```
<fte:call
  agent="1@MFTQM"
  idproperty="doCall.id"
  rcproperty="doCall.rc">

  <fte:command command="foo" />

  <fte:metadata>
    <fte:entry name="org.foo.item.color" value="red"/>
    <fte:entry name="org.foo.item.size" value="medium"/>
  </fte:metadata>
</fte:call>
```

```
</fte:metadata>
```

```
</fte:call>
```

foo.cmd ファイル

```
@echo off
echo Program: %~dpnx0
echo Parameters: %*
echo Environment:
set org.foo
exit /b 0
```

foo.cmd によって生成された標準出力、以下を使用して IBM MQ Explorer

```
Program: C:\IBM\MQdata\mqft\Commands\foo.cmd
Parameters:
Environment:
org.foo.item.color=red
org.foo.item.size=medium
```

関連タスク

[Apache Ant と MFT の併用](#)

プログラム呼び出しのネスト・エレメント

プログラムは、`fte:presrc`、`fte:predst`、`fte:postdst`、`fte:postsrc`、および `fte:command` の 5 つのネストされたエレメントのいずれかを使用して開始できます。これらのネストされたエレメントは、エージェントにその処理の一部で外部プログラムを呼び出すように指示します。プログラムを開始するには、その前に、コマンドを実行するエージェントの `agent.properties` ファイル内の `commandPath` プロパティーで指定されている場所にコマンドがあることを確認する必要があります。

プログラム呼び出しの各エレメントは異なる名前を持っても、同じ属性セットおよび同じネストされたエレメントのセットを共有します。プログラムは、**`fte:filecopy`**、**`fte:filemove`**、および **`fte:command`** の Ant タスクによって開始することができます。

Connect:Direct® ブリッジ・エージェントからプログラムを呼び出すことはできません。

プログラムを呼び出すことのできる Ant タスクは、以下のとおりです。

- `fte:filecopy` タスクは、ネストされた `fte:predst`、`fte:postdst`、`fte:presrc`、および `fte:postsrc` エレメントを使用して、プログラム呼び出しパラメーターをネストします。
- `fte:filemove` タスクは、ネストされた `fte:predst`、`fte:postdst`、`fte:presrc`、および `fte:postsrc` エレメントを使用して、プログラム呼び出しパラメーターをネストします。
- `fte:call` タスクは、`fte:command` ネスト・エレメントを使用してプログラム呼び出しパラメーターをネストします。

属性

コマンド

必須。呼び出すプログラムの名前を示します。エージェントがコマンドを実行できるようにするには、コマンドは、エージェントの `agent.properties` ファイル内の `commandPath` プロパティーで指定されたロケーションになければなりません。詳しくは、[commandPath MFT プロパティー](#) を参照してください。 `command` 属性で指定されたパス情報は、`commandPath` プロパティーで指定されたロケーションに対する相対パスと見なされます。 `type` が `executable` の場合、実行可能プログラムが予想されます。そうでない場合は、呼び出しタイプに適したスクリプトが予想されます。

retrycount

オプション。プログラムが成功を示す戻りコードを戻さなかった場合に、プログラムの呼び出しを再試行する回数。 `command` 属性によって指定されたプログラムは、この回数まで呼び出されます。この属性に指定する値は、負数以外でなければなりません。 `retrycount` 属性を指定しない場合は、デフォルト値のゼロが使用されます。

retrywait

オプション。プログラム呼び出しを再度試行するまでの待機時間 (秒数)。 **command** 属性で指定されたプログラムが成功を示す戻りコードを返さず、**retrycount** 属性にゼロ以外の値が指定されている場合、このパラメータは再試行間の待機時間を決定します。この属性に指定する値は、負数以外でなければなりません。**retrywait** 属性を指定しない場合は、デフォルト値のゼロが使用されます。

successrc

オプション。この属性の値は、プログラム呼び出しが正常に実行される条件を決定するために使用されます。コマンドの処理戻りコードは、この式を使用して評価されます。この値は、1 つ以上の式を組み合わせ、ブールの OR を意味する垂直バー文字 (|)、またはブールの AND を意味するアンパサンド文字 (&) で構成することができます。各式は、以下のいずれかのタイプの式とすることができます。

- 処理戻りコードとの等価テストを示す数値。
- 処理戻りコードとの大なりテストを示す、接頭部に ">" 文字が付いた数値。
- 数値とプロセスの戻りコードとの間のより小さいテストを示すために、先頭に "<" 文字が付いた数値です。
- "!" の接頭部が付いた数値 数値とプロセスの戻りコードとの間の不等号テストを示す文字。

例えば、>2&<7&!5|0|14 は、0、3、4、6、14 の戻りコードが成功したと解釈されます。これ以外の戻りコードは、すべて失敗と解釈されます。**successrc** 属性を指定しない場合は、デフォルト値のゼロが使用されます。これは、ゼロの戻りコードを戻した場合にのみ、コマンドは正常に実行されたと判断されるという意味です。

タイプ

オプション。この属性の値は、呼び出されているプログラムのタイプを指定します。以下のいずれかのオプションを指定します。

executable

タスクは、実行可能プログラムを呼び出します。**arg** ネスト・エレメントを使用して追加の引数を指定することができます。プログラムは、**commandPath** 上、および実行許可が設定されている適切な場所においてアクセス可能であると想定されます。UNIX スクリプトは、シェル・プログラムを指定する限り呼び出すことができます (例えば、シェル・スクリプト・ファイルの最初の行は `#!/bin/sh` です)。**stderr** または **stdout** に書き込まれるコマンド出力は、その呼び出しの Managed File Transfer ログに送信されます。ただし、そのデータ出力量は、エージェントの構成によって制限されます。デフォルトでは、10K バイトのデータまでですが、エージェントのプロパティ **maxCommandOutput** を使用してこのデフォルトを指定変更することができます。

antscript

このタスクは、指定された Ant スクリプトをコマンド **fteAnt** を使用して実行します。プロパティは、**property** ネスト・エレメントを使用して指定できます。Ant ターゲットは **target** ネスト・エレメントを使用して指定できます。Ant スクリプトは、**commandPath** 上でアクセス可能と想定されます。**stderr** または **stdout** に書き込まれる Ant 出力は、その呼び出しに対する Managed File Transfer ログに送信されます。ただし、そのデータ出力量は、エージェントの構成によって制限されます。デフォルトでは、10K バイトのデータまでですが、エージェントのプロパティ **maxCommandOutput** を使用してこのデフォルトを指定変更することができます。

z/OS jcl

値 **jcl** は z/OS でのみサポートされ、指定された z/OS JCL スクリプトを実行します。JCL はジョブとして実行依頼され、ジョブ・カードがあることが必要です。ジョブが正常に実行依頼されると、Managed File Transfer ログに書き込まれる JCL コマンド出力に、**JOB job_name(job_id)** というテキストが入ります。ここで、

- **job_name** は、JCL のジョブ・カードによって特定されるジョブの名前です。
- **job_id** は、z/OS システムが生成したジョブ ID です。

ジョブを正常に実行依頼できないと JCL スクリプト・コマンドは失敗し、失敗の理由 (例えば、ジョブ・カードが無い、など) を示すメッセージをログに書き込みます。ジョブが実行されたか、あるいは正常に完了したかどうかを認識するには、SDSF などのシステム・サービスを使用します。Managed File Transfer はジョブの実行依頼のみを行うため、情報を提供しません。ジョブを実行するタイミングやジョブ出力の提示方法は、システムが決定します。JCL スクリプトはバッチ・ジョ

ブとして実行依頼されるため、**presrc** または **predst** ネスト・エレメントに **jcl** を指定することはお勧めしません。これは、ジョブが正常に実行依頼されたことのみが分かっており、転送の開始前に正常に実行されたかどうかは分からないためです。タイプが **jcl** の場合、有効なネストされたエレメントはありません。

以下の例は、JCL ジョブを示しています。

```
//MYJOB JOB
//*
//MYJOB EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=H
//SYSUT1 DD DSN=FRED.DEMO.TXT,DISP=SHR
//SYSUT2 DD DSN=BOB.DEMO.TXT,DISP=(NEW,CATLG),
// RECFM=VB,LRECL=133,BLKSIZE=2048,
// SPACE=(TRK,(30,5),RLSE)
//SYSIN DD DUMMY
```

ネスト・エレメントとして指定するパラメーター

fte:arg

type 属性の値が実行可能の場合にのみ有効です。ネストされた **fte:arg** エレメントを使用して、プログラム呼び出しの一部として呼び出されるプログラムへの引数を指定します。プログラム引数は、**fte:arg** エレメントが検出された順序で、**fte:arg** エレメントによって指定された値から作成されます。プログラム呼び出しのネストされたエレメントとして、ゼロ個以上の **fte:arg** エレメントを指定することができます。

fte:property

type 属性の値が **antscript** の場合にのみ有効です。ネストされた **fte:property** エレメントの **name** 属性と **value** 属性を使用して、名前と値のペアを Ant スクリプトに渡します。プログラム呼び出しのネストされたエレメントとして、ゼロ個以上の **property** エレメントを指定することができます。

fte:target

type 属性の値が **antscript** の場合にのみ有効です。呼び出す Ant スクリプトのターゲットを指定します。プログラム呼び出しのネストされたエレメントとして、ゼロ個以上の **fte:target** エレメントを指定することができます。

Arg 属性

値

必須。呼び出されるプログラムに渡される引数の値。

プロパティー属性

名前

必須。Ant スクリプトに渡すプロパティーの名前。

値

必須。Ant スクリプトに渡されるプロパティー名に関連付ける値。

例

この例は、**fte:filecopy** タスクの一部として指定されている **fte:postsrc** プログラム呼び出しを示しています。プログラム呼び出しは、**post.sh** というプログラムのためのもので、**/home/fteuser2/file.bin** の単一の引数を提供します。

```
<fte:filecopy
cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
src="agent1@qm1" dst="agent2@qm2"
rcproperty="copy.result">
<fte:filespec srcfilespec="/home/fteuser1/file.bin" dstfile="/home/fteuser2/file.bin"/>
<fte:postsrc command="post.sh" successsrc="1" >
  <fte:arg value="/home/fteuser2/file.bin"/>
</fte:postsrc>
</fte:filecopy>
```

この例は、`fte:call` タスクの一部として指定される `fte:command` プログラム呼び出しを示しています。このプログラム呼び出しは、コマンド行引数を渡さない `command.sh` という名前の実行可能プログラム用です。`command.sh` が成功の戻りコード 1 を戻さない場合は、30 秒後にコマンドが再試行されます。

```
<fte:call
cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
agent="agent1@qm1"
rcproperty="call.rc"
origuser="bob"
jobname="${job.id}">
<fte:command command="command.sh" successsrc="1" retrycount="5" retrywait="30"/>
</fte:call>
```

この例は、`fte:call` タスクの一部として指定される `fte:command` プログラム呼び出しを示しています。プログラム呼び出しは、`script.xml` という Ant スクリプト内のコピーおよび圧縮ターゲットに対して行われます。このスクリプトには 2 つのプロパティが渡されます。

```
<fte:call
cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
agent="agent1@qm1"
rcproperty="call.rc"
origuser="bob"
jobname="${job.id}">
<fte:command command="script.xml" type="antscript">
  <property name="src" value="AGENT5@QM5"/>
  <property name="dst" value="AGENT3@QM3"/>
  <target name="copy"/>
  <target name="compress"/>
</fte:command>
</fte:call>
```

関連タスク

[MFT で実行するプログラムの指定](#)

[Apache Ant と MFT の併用](#)

MFT ユーザー出口カスタマイズ・リファレンス

Managed File Transfer にユーザー出口を構成するために役立つ参照情報。

関連概念

[MFT のソースと宛先のユーザー出口](#)

MFT ユーザー出口のメタデータ

Managed File Transfer のユーザー出口ルーチンに提供できるメタデータには、環境、転送、およびファイル・メタデータの 3 つの異なるタイプがあります。このメタデータは、Java のキー/値ペアのマップとして示されます。

環境メタデータ

環境メタデータは、すべてのユーザー出口ルーチンに渡されます。また、ユーザー出口ルーチンの呼び出し元エージェント・ランタイム環境を記述します。このメタデータは読み取り専用で、ユーザー出口ルーチンでは更新できません。

表 918. 環境メタデータ	
キー	説明
AGENT_CONFIGURATION_DIRECTORY_KEY	エージェントの構成情報が入っているディレクトリーの名前。
AGENT_PRODUCT_DIRECTORY_KEY	エージェント・コードがインストールされているディレクトリーの名前。
AGENT_VERSION_KEY	出口ルーチンを呼び出すエージェント・ランタイムのバージョン番号。

表 1 で指定されているキー名および値名は、EnvironmentMetaDataConstants インターフェースで定義されている定数です。

転送メタデータ

転送メタデータは、すべてのユーザー出口ルーチンに渡されます。メタデータは、システム提供の値とユーザー提供の値で構成されます。システム提供値を変更しても、その変更は無視されます。ソース転送開始ユーザー出口のユーザー提供の初期値は、転送の定義時に提供する値に基づくものです。ソース・エージェントは、ソース転送開始ユーザー出口の処理の一部として、ユーザー提供の値を変更することができます。このユーザー出口は、ファイル転送全体が開始する前に呼び出されます。これらの変更は、その転送に関連した他の出口ルーチンへの以降の呼び出しで使用されます。転送メタデータは転送全体に適用されます。

すべてのユーザー出口で転送メタデータから値を読み取ることができますが、転送メタデータを変更できるのはソース転送開始ユーザー出口だけです。

転送メタデータを使用して異なるファイル転送の間で情報を伝搬することはできません。

システム提供の転送メタデータについては、表 2 で詳しく扱われています。

表 919. 転送メタデータ	
キー	説明
DESTINATION_AGENT_KEY	転送の宛先であるエージェントの名前。
JOB_NAME_KEY	転送要求に関連したジョブ名。
MQMD_USER_KEY	転送要求を実行依頼するために使用されるメッセージからの MQMD ユーザー・フィールド。
ORIGINATING_HOST_KEY	転送要求で発信ホスト名として指定されるホスト名。
ORIGINATING_USER_KEY	転送要求で発信ユーザー ID として指定されるユーザー名。
SOURCE_AGENT_KEY	転送のソースであるエージェントの名前。
TRANSFER_ID_KEY	転送の ID。

表 2 で指定されているキー名および値名は、TransferMetaDataConstants インターフェースで定義されている定数です。

ファイル・メタデータ

ファイル・メタデータは、ファイル仕様の一部として、ソース転送開始出口に渡されます。ソース・ファイルと宛先ファイル用に別個のファイル・メタデータがあります。

ファイル・メタデータを使用して異なるファイル転送の間で情報を伝搬することはできません。

表 920. ファイル・メタデータ		
キー	許可値	説明
CONVERT_LINE_SEPARATORS		テキスト転送で使用されるキー値。ソース・データに含まれる CRLF (復帰改行) または LF (改行) 行分離文字シーケンスが、宛先での行分離文字シーケンスに変換されるかどうかを示します。
DELIMITER_KEY		レコード単位のデータを通常のファイルに転送するときに使用されるキー値。レコード・データを分離するための区切り文字を定義します。 メッセージからファイルへの転送およびファイルからメッセージへの転送にも使用されます。

表 920. ファイル・メタデータ (続き)		
キー	許可値	説明
DELIMITER_POSITION_KEY	DELIMITER_POSITION_PREFIX_VALUE DELIMITER_POSITION_POSTFIX_VALUE	区切り文字の位置 (接頭部または接尾部) を定義するために、DELIMITER_KEY と一緒に使します。
DELIMITER_TYPE_KEY	DELIMITER_TYPE_BINARY_VALUE DELIMITER_TYPE_TEXT_VALUE DELIMITER_TYPE_SIZE_VALUE	区切り文字のタイプを定義するために、DELIMITER_KEY と一緒に使します。
DESTINATION_EXIST_KEY	DESTINATION_EXIST_KEY_ERROR_VALUE DESTINATION_EXIST_KEY_OVERWRITE_VALUE	宛先ファイルが存在する場合のファイル転送動作を決定します。
FILE_ALIAS_KEY		転送中のファイルの別名を定義するために使用されるキー値。
FILE_CHECKSUM_METHOD_KEY	FILE_CHECKSUM_METHOD_NONE_VALUE FILE_CHECKSUM_METHOD_MD5_VALUE	ファイルの転送時に使用するチェックサム方式を決定します。
FILE_CONVERSION_KEY	FILE_CONVERSION_TEXT_VALUE FILE_CONVERSION_BINARY_VALUE	ファイル内容に適用される変換のタイプを決定します。
FILE_ENCODING_KEY		テキスト・ファイルで使用されるエンコード方式を決定します。
FILE_END_OF_LINE_KEY	FILE_END_OF_LINE_LF_VALUE FILE_END_OF_LINE_CRLF_VALUE	行の終わりを示す文字シーケンスを決定します。 <LF> か <CR><LF> のいずれかです。
FILE_SPACE_ALIAS		ファイル・スペースに含まれているファイルの別名を指定します。 注: このメタデータを使用できるのは、FILE_TYPE_KEY が FILE_TYPE_FILE_SPACE_VALUE の場合に限られます。
FILE_SPACE_NAME		ファイル・スペースの名前を指定します。 注: このメタデータを使用できるのは、FILE_TYPE_KEY が FILE_TYPE_FILE_SPACE_VALUE の場合に限られます。
FILE_TYPE_KEY	FILE_TYPE_FILE_VALUE FILE_TYPE_DIRECTORY_VALUE FILE_TYPE_DATASET_VALUE FILE_TYPE_PDS_VALUE FILE_TYPE_QUEUE_VALUE FILE_TYPE_FILE_SPACE_VALUE	宛先ファイル、キュー、またはファイル・スペースを指定します。
GROUP_ID_KEY		メッセージからファイルへの転送で使用されるキー値。ソース・キューから読み取るメッセージ・グループを決定します。この属性が有効なのは、USE_GROUPS_KEY の値が USE_GROUPS_TRUE_VALUE の場合のみです。
INCLUDE_DELIMITER_IN_MESSAGE_KEY	INCLUDE_DELIMITER_IN_MESSAGE_TRUE_VALUE INCLUDE_DELIMITER_IN_MESSAGE_FALSE_VALUE	ファイルからメッセージへの転送で使用されるキー値。ファイルを複数のメッセージに分割するために使用した区切り文字を、それらのメッセージの末尾に組み込むかどうかを決定します。この属性が有効なのは、DELIMITER_TYPE_KEY の値が DELIMITER_TYPE_BINARY_VALUE または DELIMITER_TYPE_TEXT_VALUE の場合のみです。

表 920. ファイル・メタデータ (続き)		
キー	許可値	説明
INSERT_RECORD_LINE_SEPARATOR_KEY		レコード単位のファイルからテキストを転送するときに使用されるキー値。各レコードの後のデータに行分離文字を挿入するかどうかを指定します。
KEEP_TRAILING_SPACES_KEY	KEEP_TRAILING_SPACES_TRUE_VALUE KEEP_TRAILING_SPACES_FALSE_VALUE	固定長形式のデータ・セットから読み取られたレコードから末尾スペースを削除するかどうかを決定するために使用されるキー値。
NEW_RECORD_ON_LINE_SEPARATOR_KEY		レコード単位のファイルへのテキスト転送に使用されるキー値。データに含まれる行分離文字をレコード・データと一緒に組み込むか、または行分離文字で新規レコードの作成を指示するか(行分離文字は書き込まれません)を指定します。
PERSISTENT_KEY	PERSISTENT_TRUE_VALUE PERSISTENT_FALSE_VALUE PERSISTENT_QDEF_VALUE	ファイルからメッセージへの転送で使用されるキー値。メッセージを永続メッセージにするかどうかを決定します。
SET_MQ_PROPS_KEY	SET_MQ_PROPS_TRUE_VALUE SET_MQ_PROPS_FALSE_VALUE	ファイルからメッセージへの転送で使用されるキー値。ファイルの最初のメッセージで IBM MQ メッセージ・プロパティを設定するかどうか、およびエラーの発生時にメッセージをキューに書き込むかどうかを決定します。
UNRECOGNISED_CODE_PAGE_KEY	UNRECOGNISED_CODE_PAGE_FAIL_VALUE UNRECOGNISED_CODE_PAGE_BINARY_VALUE	ファイルからメッセージへの転送で使用されるキー値。宛先キュー・マネージャーがデータのコード・ページを認識できない場合に、テキスト・モードの転送が失敗するか、それとも変換が実行されるかを決定します。
USE_GROUPS_KEY	USE_GROUPS_TRUE_VALUE USE_GROUPS_FALSE_VALUE	メッセージからファイルへの転送で使用されるキー値。完全に揃ったメッセージ・グループだけをソース・キューから転送するかどうかを決定します。
WAIT_TIME_KEY		メッセージからファイルへの転送で使用されるキー値。ソース・エージェントが以下のいずれかの状態になるまで待機する時間を秒単位で決定します。 • USE_GROUPS_KEY の値が FALSE の場合、キューがもともと空であるか、後で空になった場合に、メッセージがソース・キューに出現するまで。 • USE_GROUPS_KEY の値が TRUE の場合、完全に揃ったグループがソース・キューに出現するまで。

表 3 で指定されているキー名および値名は、FileMetaDataConstants インターフェースで定義されている定数です。

関連概念

2145 ページの『Java ユーザー・エグジット MFT 用インターフェース』

ユーザー出口ルーチンの Java インターフェースに関する参照情報については、このセクションのトピックを参照してください。

関連タスク

[ユーザー出口での MFT のカスタマイズ](#)

関連資料

[2139 ページの『MFT リソース・モニター・ユーザー出口』](#)

リソース・モニターのユーザー出口を使用して、関連タスクが開始される前に、モニターのトリガー条件が満たされた場合に実行するようカスタム・コードを構成できます。

[2143 ページの『ユーザー出口用の MFT エージェント・プロパティ』](#)

`agent.properties` ファイル内の標準プロパティに加えて、ユーザー出口ルーチン専用の拡張プロパティがいくつかあります。これらのプロパティは、デフォルトでは含まれていないため、いずれかのプロパティを使用する場合は、`agent.properties` ファイルを手動で編集する必要があります。エージェントの実行中に `agent.properties` ファイルに変更を加える場合は、エージェントを停止してから再始動して、変更を反映させます。

MFT リソース・モニター・ユーザー出口

リソース・モニターのユーザー出口を使用して、関連タスクが開始される前に、モニターのトリガー条件が満たされた場合に実行するようカスタム・コードを構成できます。

ユーザー出口コードから直接新しい転送を呼び出すことは推奨されません。ユーザー出口はエージェントの再始動に対して回復力がないため、場合によってはファイルが複数回転送されることになります。

リソース・モニターのユーザー出口は、ユーザー出口の既存インフラストラクチャーを使用します。モニター・ユーザー出口は、モニターがトリガーしてから呼び出されますが、この呼び出しは、対応するタスクがモニターのタスクによって実行される前に行われます。これにより、ユーザー出口は実行されるタスクを変更して、タスクを処理するかどうかを決定できます。モニター・タスクは、モニター・メタデータを更新することで変更できます。更新されたモニター・メタデータは、元のモニターの作成によって作成されたタスク文書で変数置換に使用されます。別の方法として、モニター出口は、パラメーターとして渡されるタスク定義 XML ストリングを置換または更新できます。モニター出口は、タスクに対する結果コード（「`proceed`」または「`cancel`」のいずれか）を返すことができます。`cancel` が返された場合、タスクは開始されず、モニター対象リソースがトリガー条件と一致するまでモニターは再開されません。リソースが変更されなければ、トリガーは開始しません。他のユーザー出口と同様に、モニター出口はまとめてチェーニングできます。出口の 1 つが `cancel` の結果コードを返すと、結果全体が `cancel` となり、タスクは開始されません。

- 環境メタデータのマップ (他のユーザー出口と同じ)
- 不変システム・メタデータおよび可変ユーザー・メタデータを含むモニター・メタデータのマップ。不変システム・メタデータは、以下のとおりです。
 - `FILENAME` - トリガー条件を満たしたファイルの名前
 - `FILEPATH` - トリガー条件を満たしたファイルへのパス
 - `FILESIZE` (バイト単位 - このメタデータは存在しない場合がある) - トリガー条件を満たしたファイルのサイズ
 - `LASTMODIFIEDDATE` (地域別) - トリガー条件を満たしたファイルの最終変更日。エージェントを実行しているタイム・ゾーンの現地日付が ISO 8601 の日付形式で表示されます。
 - `LASTMODIFIEDTIME` (地域別) - トリガー条件を満たしたファイルの最終変更時刻 (地域別形式)。エージェントを実行しているタイム・ゾーンの現地時間が ISO 8601 の時間形式で表示されます。
 - `LASTMODIFIEDDATEUTC` - トリガー・ファイルの最終変更日 (世界共通形式)。この日付は、UTC タイム・ゾーンに変換された現地日付として表され、ISO 8601 日付として書式設定されます。
 - `LASTMODIFIEDTIMEUTC` - トリガー条件を満たしたファイルの最終変更時間 (世界共通形式)。この日付は、UTC タイム・ゾーンに変換された現地時間として表され、ISO 8601 時間として書式設定されます。
 - `AGENTNAME` - モニター・エージェント名
- モニター・トリガーの結果として実行されるタスクを表す XML ストリング。

モニター出口は、以下のデータを返します。

- さらに進行するかどうか (proceed または cancel) を示す標識
- トリガー条件を満たしたログ・メッセージに挿入するストリング

モニター出口コードを実行した結果、パラメーターとして最初に渡されたモニター・メタデータおよびタスク定義 XML ストリングも更新されている場合があります。

エージェント・プロパティ monitorExitClasses (agent.properties ファイル内) の値は、ロードするモニター出口クラスを指定します。各出口クラスはコンマで区切ります。以下に例を示します。

```
monitorExitClasses=testExits.TestExit1,testExits.testExit2
```

モニター・ユーザー出口のインターフェースは、以下のとおりです。

```
package com.ibm.wmqfte.exitroutine.api;

import java.util.Map;

/**
 * An interface that is implemented by classes that want to be invoked as part of
 * user exit routine processing. This interface defines a method that will be
 * invoked immediately prior to starting a task as the result of a monitor trigger
 */
public interface MonitorExit {

    /**
     * Invoked immediately prior to starting a task as the result of a monitor
     * trigger.
     *
     * @param environmentMetaData
     *      meta data about the environment in which the implementation
     *      of this method is running. This information can only be read,
     *      it cannot be updated by the implementation. The constant
     *      defined in <code>EnvironmentMetaDataConstants</code> class can
     *      be used to access the data held by this map.
     *
     * @param monitorMetaData
     *      meta data to associate with the monitor. The meta data passed
     *      to this method can be altered, and the changes will be
     *      reflected in subsequent exit routine invocations. This map
     *      also contains keys with IBM reserved names. These entries are
     *      defined in the <code>MonitorMetaDataConstants</code> class and
     *      have special semantics. The the values of the IBM reserved names
     *      cannot be modified by the exit
     *
     * @param taskDetails
     *      An XML String representing the task to be executed as a result of
     *      the monitor triggering. This XML string may be modified by the
     *      exit
     *
     * @return
     *      a monitor exit result object which is used to determine if the
     *      task should proceed, or be cancelled.
     */
    MonitorExitResult onMonitor(Map<String, String> environmentMetaData,
                               Map<String, String> monitorMetaData,
                               Reference<String> taskDetails);
}
```

モニター・メタデータ内の IBM 予約値の定数は、以下のとおりです。

```
package com.ibm.wmqfte.exitroutine.api;

/**
 * Constants for IBM reserved values placed into the monitor meta data
 * maps used by the monitor exit routines.
 */
public interface MonitorMetaDataConstants {

    /**
     * The value associated with this key is the name of the trigger
     */
}
```



```

* file associated with the monitor. Any modification performed
* to this property by user exit routines will be ignored.
*/
final String FILE_NAME_KEY = "FILENAME";

/**
* The value associated with this key is the path to the trigger
* file associated with the monitor. Any modification performed
* to this property by user exit routines will be ignored.
*/
final String FILE_PATH_KEY = "FILEPATH";

/**
* The value associated with this key is the size of the trigger
* file associated with the monitor. This will not be present in
* the cases where the size cannot be determined. Any modification
* performed to this property by user exit routines will be ignored.
*/
final String FILE_SIZE_KEY = "FILESIZE";

/**
* The value associated with this key is the local date on which
* the trigger file associated with the monitor was last modified.
* Any modification performed to this property by user exit routines
* will be ignored.
*/
final String LAST_MODIFIED_DATE_KEY = "LASTMODIFIEDDATE";

/**
* The value associated with this key is the local time at which
* the trigger file associated with the monitor was last modified.
* Any modification performed to this property by user exit routines
* will be ignored.
*/
final String LAST_MODIFIED_TIME_KEY = "LASTMODIFIEDTIME";

/**
* The value associated with this key is the UTC date on which
* the trigger file associated with the monitor was last modified.
* Any modification performed to this property by user exit routines
* will be ignored.
*/
final String LAST_MODIFIED_DATE_KEY_UTC = "LASTMODIFIEDDATEUTC";

/**
* The value associated with this key is the UTC time at which
* the trigger file associated with the monitor was last modified.
* Any modification performed to this property by user exit routines
* will be ignored.
*/
final String LAST_MODIFIED_TIME_KEY_UTC = "LASTMODIFIEDTIMEUTC";

/**
* The value associated with this key is the name of the agent on which
* the monitor is running. Any modification performed to this property by
* user exit routines will be ignored.
*/
final String MONITOR_AGENT_KEY = "AGENTNAME";
}

```

ユーザー出口の例

このクラスの例では、MonitorExit インターフェースを実装します。この例では、*REDIRECTEDAGENT* というモニター・メタデータにカスタム置換変数を追加します。この変数には、時刻が奇数の場合は値 *LONDON* が取り込まれ、時刻が偶数の場合は値 *PARIS* が取り込まれます。モニター出口の結果コードは、常に *proceed* を戻すように設定されます。

```

package com.ibm.wmqfte.monitor;

import java.util.Calendar;
import java.util.Map;

import com.ibm.wmqfte.exitroutine.api.MonitorExit;
import com.ibm.wmqfte.exitroutine.api.MonitorExitResult;
import com.ibm.wmqfte.exitroutine.api.Reference;

/**

```

```

* Example resource monitor user exit that changes the monitor mutable
* metadata value between 'LONDON' and 'PARIS' depending on the hour of the day.
*
*/
public class TestMonitorExit implements MonitorExit {

    // custom variable that will substitute destination agent
    final static String REDIRECTED_AGENT = "REDIRECTEDAGENT";

    public MonitorExitResult onMonitor(
        Map<String, String> environmentMetaData,
        Map<String, String> monitorMetaData,
        Reference<String> taskDetails) {

        // always succeed
        final MonitorExitResult result = MonitorExitResult.PROCEED_RESULT;

        final int hour = Calendar.getInstance().get(Calendar.HOUR_OF_DAY);

        if (hour%2 == 1) {
            monitorMetaData.put(REDIRECTED_AGENT, "LONDON");
        } else {
            monitorMetaData.put(REDIRECTED_AGENT, "PARIS");
        }

        return result;
    }
}

```

REDIRECTEDAGENT 置換変数を使用するモニターに対応するタスクは、以下のようなものになる可能性があります。

```

<?xml version="1.0" encoding="UTF-8"?>
<request version="4.00"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <managedTransfer>
    <originator>
      <hostName>reportserver.com</hostName>
      <userID>USER1</userID>
    </originator>
    <sourceAgent agent="AGENT1"
      QMgr="QM1"/>
    <destinationAgent agent="${REDIRECTEDAGENT}"
      QMgr="QM2"/>
    <transferSet>
      <item mode="binary" checksumMethod="MD5">
        <source recursive="false" disposition="delete">
          <file>c:\sourcefiles\reports.doc</file>
        </source>
        <destination type="file" exist="overwrite">
          <file>c:\destinationfiles\reports.doc</file>
        </destination>
      </item>
    </transferSet>
  </managedTransfer>
</request>

```

この転送が開始される前に、<destinationAgent>エレメントのエージェント属性の値が LONDON または PARIS のいずれかに置き換えられます。

モニター出口クラスで置換変数とタスク定義 XML (大文字) を指定する必要があります。

関連概念

[ユーザー出口での MFT のカスタマイズ](#)

[2135 ページの『MFT ユーザー出口のメタデータ』](#)

Managed File Transfer のユーザー出口ルーチンに提供できるメタデータには、環境、転送、およびファイル・メタデータの 3 つの異なるタイプがあります。このメタデータは、Java のキー/値ペアのマップとして示されます。

[2145 ページの『Java ユーザー・エグジット MFT 用インターフェース』](#)

ユーザー出口ルーチンの Java インターフェースに関する参照情報については、このセクションのトピックを参照してください。

[MFT のソースと宛先のユーザー出口](#)

関連資料

2143 ページの『ユーザー出口用の MFT エージェント・プロパティ』

`agent.properties` ファイル内の標準プロパティに加えて、ユーザー出口ルーチン専用の拡張プロパティがいくつかあります。これらのプロパティは、デフォルトでは含まれていないため、いずれかのプロパティを使用する場合は、`agent.properties` ファイルを手動で編集する必要があります。エージェントの実行中に `agent.properties` ファイルに変更を加える場合は、エージェントを停止してから再始動して、変更を反映させます。

ユーザー出口用の MFT エージェント・プロパティ

`agent.properties` ファイル内の標準プロパティに加えて、ユーザー出口ルーチン専用の拡張プロパティがいくつかあります。これらのプロパティは、デフォルトでは含まれていないため、いずれかのプロパティを使用する場合は、`agent.properties` ファイルを手動で編集する必要があります。エージェントの実行中に `agent.properties` ファイルに変更を加える場合は、エージェントを停止してから再始動して、変更を反映させます。

環境変数は、ファイルやディレクトリの場所を表す一部 Managed File Transfer のプロパティで使用できます。これにより、製品の一部の実行時に使用されるファイルまたはディレクトリの場所を、環境の変更（プロセスを実行しているユーザーなど）に合わせて変えることができます。詳しくは、[MFT プロパティの環境変数](#)を参照してください。

ユーザー出口ルーチン・プロパティ

ユーザー出口ルーチンは、以下の表にリストされている順序で呼び出されます。`agent.properties` ファイルについて詳しくは、[拡張エージェント・プロパティ: ユーザー出口ルーチン](#)を参照してください。

表 921. ユーザー出口用のエージェント・プロパティ	
プロパティ名	説明
<code>sourceTransferEndExitClasses</code>	ソース転送終了出口ルーチンを実装するクラスのコンマ区切りリストを指定します。
<code>sourceTransferStartExitClasses</code>	ソース転送開始出口ルーチンを実装するクラスのコンマ区切りリストを指定します。
<code>destinationTransferStartExitClasses</code>	宛先転送開始ユーザー出口ルーチンを実装するクラスのコンマ区切りリストを指定します。
<code>destinationTransferEndExitClasses</code>	宛先転送ユーザー出口ルーチンを実装するクラスのコンマ区切りリストを指定します。
<code>exitClassPath</code>	ユーザー出口ルーチンのクラスパスの役割を果たす、プラットフォーム固有のディレクトリーの文字区切りリストを指定します。 このクラスパス内のエントリよりも先に、エージェントの <code>exits</code> ディレクトリが検索されます。 Windows Windows では、ファイルパスを指定する際、区切り文字としてバックスラッシュ (\) を 2 つ並べる方法と、スラッシュ (/) を 1 つ使う方法のどちらでも指定できます。括弧、コンマ (,)、およびバックスラッシュ (\) は、MFT コマンドにおいて特殊文字となるため、バックスラッシュ (\) でエスケープする必要があります。例えば： <pre>exitClassPath=C:/mycomp/mqft/exits/encryptFileExit.jar; C:/mycomp/mqft/exits/fileFilter.jar.</pre> このプロパティの値には、環境変数を含めることができます。
<code>exitNativeLibraryPath</code>	ユーザー出口ルーチンのネイティブ・ライブラリー・パスの役割を果たす、プラットフォーム固有のディレクトリーの文字区切りリストを指定します。 このプロパティの値には、環境変数を含めることができます。
<code>monitorExitClasses</code>	モニター出口ルーチンを実装するクラスのコンマ区切りリストを指定します。詳しくは、 MFT リソース・モニター・ユーザー出口 を参照してください。

表 921. ユーザー出口用のエージェント・プロパティー (続き)	
プロパティー名	説明
protocolBridgeCredentialExitClasses	プロトコル・ブリッジ資格情報ユーザー出口ルーチンを実装するクラスのコンマ区切りリストを指定します。詳しくは、 出口クラスを使用したファイル・サーバーの資格情報のマップ を参照してください。
protocolBridgePropertiesExitClasses	プロトコル・ブリッジ・サーバー・プロパティー・ユーザー出口ルーチンを実装するクラスのコンマ区切りリストを指定します。詳しくは、 ProtocolBridgePropertiesExit2: プロトコル・ファイル・サーバー・プロパティーの検索 を参照してください。
IOExitClasses	入出力ユーザー出口ルーチンを実装するクラスのコンマ区切りリストを指定します。IOExit インターフェースを実装するクラスのみリストします。つまり、IOExitResourcePath や IOExitChannel などの他の入出力ユーザー出口インターフェースを実装するクラスはリストしないでください。詳しくは、 MFT 転送入出力ユーザー出口の使用 を参照してください。

出口の起動順序

ソース出口および宛先出口は、次の順序で起動されます。

1. SourceTransferStartExit
2. DestinationTransferStartExit
3. DestinationTransferEndExit
4. SourceTransferEndExit

ソース出口および宛先出口のチェーニング

複数の出口を指定する場合は、リスト中の最初の出口が最初に起動され、次に 2 番目の出口、という順序で起動されます。最初の出口で発生した変更内容は次に起動される出口の入力として渡されます。後続の出口についても同様です。例えば、ソース転送開始出口が 2 つあり、最初の出口による転送メタデータへの変更内容は、2 番目の出口に入力されます。それぞれの出口が独自の結果を返します。特定のタイプのすべての出口が転送結果コードとして PROCEED を返すと、全体的な結果は PROCEED になります。1 つ以上の出口が CANCEL_TRANSFER を返すと、全体的な結果は CANCEL_TRANSFER になります。出口から返された結果コードおよびストリングはすべて、転送ログに出力されます。

ソース転送開始出口からの全体的な結果が PROCEED であった場合は、出口が行った変更を使用して転送が開始します。全体的な結果が CANCEL_TRANSFER であった場合、ソース転送終了出口が起動された後、転送は取り消されます。転送ログにおける完了状況は、「取り消し済み」になります。

宛先転送開始出口からの全体的な結果が PROCEED であった場合は、出口が行った変更を使用して転送が開始します。全体的な結果が CANCEL_TRANSFER であった場合、宛先転送終了出口が起動された後、ソース転送終了出口が起動されます。最後に転送が取り消されます。転送ログにおける完了状況は、「取り消し済み」になります。

ソースまたは宛先出口が、チェーンで、または実行順に、情報を次の出口に渡す必要がある場合は、転送メタデータを更新することによって行う必要があります。転送メタデータの使用は、出口インプリメンテーションに固有です。例えば、出口が戻りの結果を CANCEL_TRANSFER に設定し、転送がキャンセルされたことを次の出口に伝達する必要がある場合は、他の出口が理解できるように転送メタデータの値を設定することによって行う必要があります。

例

```
sourceTransferStartExitClasses=com.ibm.wmqfte.test.MFTTestSourceTransferStartExit
sourceTransferEndExitClasses=com.ibm.wmqfte.test.MFTTestSourceTransferEndExit
destinationTransferStartExitClasses=com.ibm.wmqfte.test.MFTTestDestinationTransferStartExit
destinationTransferEndExitClasses=com.ibm.wmqfte.test.MFTTestDestinationTransferEndExit
exitClassPath=C:/mycomp/mqft/exits/encryptFileExit.jar;C:/mycomp/mqft/exits/fileFilter.jar
```

関連概念

[ユーザー出口での MFT のカスタマイズ](#)

[2135 ページの『MFT ユーザー出口のメタデータ』](#)

Managed File Transfer のユーザー出口ルーチンに提供できるメタデータには、環境、転送、およびファイル・メタデータの 3 つの異なるタイプがあります。このメタデータは、Java のキー/値ペアのマップとして示されます。

[2145 ページの『Java ユーザー・エグジット MFT 用インターフェース』](#)

ユーザー出口ルーチンの Java インターフェースに関する参照情報については、このセクションのトピックを参照してください。

関連資料

[2139 ページの『MFT リソース・モニター・ユーザー出口』](#)

リソース・モニターのユーザー出口を使用して、関連タスクが開始される前に、モニターのトリガー条件が満たされた場合に実行するようカスタム・コードを構成できます。

[MFT プロパティの環境変数](#)

[MFT agent.properties ファイル](#)

Java ユーザー・エグジット MFT 用インターフェース

ユーザー出口ルーチンの Java インターフェースに関する参照情報については、このセクションのトピックを参照してください。

関連タスク

[ユーザー出口での MFT のカスタマイズ](#)

関連資料

[2148 ページの『DestinationTransferStartExit.java インターフェース』](#)

[2147 ページの『DestinationTransferEndExit.java インターフェース』](#)

[2151 ページの『IOExit.java インターフェース』](#)

[2153 ページの『IOExitChannel.java インターフェース』](#)

[2154 ページの『IOExitLock.java インターフェース』](#)

[2155 ページの『IOExitPath.java インターフェース』](#)

[2156 ページの『IOExitProperties.java インターフェース』](#)

[2159 ページの『IOExitRecordChannel.java インターフェース』](#)

[2161 ページの『IOExitRecordResourcePath.java インターフェース』](#)

[2162 ページの『IOExitResourcePath.java インターフェース』](#)

[2166 ページの『IOExitWildcardPath.java インターフェース』](#)

[2167 ページの『MonitorExit.java インターフェース』](#)

[2168 ページの『ProtocolBridgeCredentialExit.java インターフェース』](#)

[2169 ページの『ProtocolBridgeCredentialExit2.java インターフェース』](#)

[2170 ページの『ProtocolBridgePropertiesExit2.java インターフェース』](#)

[2173 ページの『SourceTransferStartExit.java インターフェース』](#)

[2172 ページの『SourceTransferEndExit.java インターフェース』](#)

CDCCredentialExit.java インターフェース

CDCCredentialExit.java

```
/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * © Copyright IBM Corp. 2011, 2026. All Rights Reserved.
```

```

*   US Government Users Restricted Rights - Use, duplication or
*   disclosure restricted by GSA ADP Schedule Contract with
*   IBM Corp.
*/
package com.ibm.wmqfte.exitroutine.api;

import java.util.Map;

/**
 * An interface that is implemented by classes that are invoked as part of
 * user exit routine processing. This interface defines methods that are
 * invoked by a Connect:Direct bridge agent to map the IBM MQ user ID of the transfer to credentials
 * that are used to access the Connect:Direct node.
 * There will be one instance of each implementation class per Connect:Direct bridge agent. The methods
 * can be called from different threads so the methods must be synchronized.
 */
public interface CDCredentialExit {

    /**
     * Invoked once when a Connect:Direct bridge agent is started. It is intended to initialize
     * any resources that are required by the exit
     *
     * @param bridgeProperties
     *        The values of properties defined for the Connect:Direct bridge.
     *        These values can only be read, they cannot be updated by
     *        the implementation.
     *
     * @return
     *        true if the initialisation is successful and false if unsuccessful
     *        If false is returned from an exit the Connect:Direct bridge agent does not
     *        start.
     */
    public boolean initialize(final Map<String, String> bridgeProperties);

    /**
     * Invoked once per transfer to map the IBM MQ user ID in the transfer message to the
     * credentials to be used to access the Connect:Direct node.
     *
     * @param mqUserId
     *        The IBM MQ user ID from which to map to the credentials to be used
     *        to access the Connect:Direct node
     * @param snode
     *        The name of the Connect:Direct SNODE specified as the cdNode in the
     *        file path. This is used to map the correct user ID and password for the
     *        SNODE.
     *
     * @return
     *        A credential exit result object that contains the result of the map and
     *        the credentials to use to access the Connect:Direct node
     */
    public CDCredentialExitResult mapMQUserId(final String mqUserId, final String snode);

    /**
     * Invoked once when a Connect:Direct bridge agent is shutdown. This method releases
     * any resources that were allocated by the exit
     *
     * @param bridgeProperties
     *        The values of properties defined for the Connect:Direct bridge.
     *        These values can only be read, they cannot be updated by
     *        the implementation.
     *
     * @return
     */
    public void shutdown(final Map<String, String> bridgeProperties); }

```

CredentialExitResult.java インターフェース

CredentialExitResult.java

```

/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * © Copyright IBM Corp. 2008, 2026. All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with
 * IBM Corp.
 */

```

```

package com.ibm.wmqfte.exitroutine.api;

/**
 * The result of invoking a Credential mapMQUserId exit method. It is composed of a result
 * code, which determines whether the mapping of the user id was successful, and an optional
 * Credentials object if the mapping is successful.
 */
public class CredentialExitResult {

    private final CredentialExitResultCode resultCode;
    private final Credentials credentials;

    /**
     * Constructor. Creates a credential exit result object with a specified result
     * code and optionally credentials.
     *
     * @param resultCode
     *         The result code to associate with the exit result being created.
     *
     * @param credentials
     *         The credentials to associate with the exit result being created.
     *         A value of <code>null</code> can be specified to indicate no
     *         credentials. If the resultCode is USER_SUCCESSFULLY_MAPPED the
     *         credentials must be set to a non-null value,
     */
    public CredentialExitResult(CredentialExitResultCode resultCode, Credentials credentials) {
        this.resultCode = resultCode;
        this.credentials = credentials;
    }

    /**
     * Returns the result code associated with this credential exit result
     *
     * @return the result code associated with this exit result.
     */
    public CredentialExitResultCode getResultCode() {
        return resultCode;
    }

    /**
     * Returns the credentials associated with this credential exit result
     *
     * @return the explanation associated with this credential exit result.
     */
    public Credentials getCredentials() {
        return credentials;
    }
}

```

関連タスク

[ユーザー出口での MFT のカスタマイズ](#)

関連資料

[2173 ページの『SourceTransferStartExit.java インターフェース』](#)

[2148 ページの『DestinationTransferStartExit.java インターフェース』](#)

[2147 ページの『DestinationTransferEndExit.java インターフェース』](#)

[2167 ページの『MonitorExit.java インターフェース』](#)

[2168 ページの『ProtocolBridgeCredentialExit.java インターフェース』](#)

***DestinationTransferEndExit.java* インターフェース**

DestinationTransferEndExit.java

```

/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * © Copyright IBM Corp. 2008, 2026. All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or

```

```

* disclosure restricted by GSA ADP Schedule Contract with
* IBM Corp.
*/
package com.ibm.wmqfte.exitpoint.api;

/**
 * An interface that is implemented by classes that want to be invoked as part of
 * user exit routine processing. This interface defines a method that will be
 * invoked immediately after completing a transfer on the agent acting as the
 * destination of the transfer.
 */
public interface DestinationTransferEndExit {

    /**
     * Invoked immediately after the completion of a transfer on the agent acting as
     * the destination of the transfer.
     *
     * @param transferExitResult
     *        a result object reflecting whether or not the transfer completed
     *        successfully.
     *
     * @param sourceAgentName
     *        the name of the agent acting as the source of the transfer.
     *
     * @param destinationAgentName
     *        the name of the agent acting as the destination of the
     *        transfer. This is the name of the agent that the
     *        implementation of this method will be invoked from.
     *
     * @param environmentMetaData
     *        meta data about the environment in which the implementation
     *        of this method is running. This information can only be read,
     *        it cannot be updated by the implementation. The constants
     *        defined in <code>EnvironmentMetaDataConstants</code> class can
     *        be used to access the data held by this map.
     *
     * @param transferMetaData
     *        meta data to associate with the transfer. The information can
     *        only be read, it cannot be updated by the implementation. This
     *        map may also contain keys with IBM reserved names. These
     *        entries are defined in the <code>TransferMetaDataConstants</code>
     *        class and have special semantics.
     *
     * @param fileResults
     *        a list of file transfer result objects that describe the source
     *        file name, destination file name and result of each file transfer
     *        operation attempted.
     *
     * @return
     *        an optional description to enter into the log message describing
     *        transfer completion. A value of <code>null</code> can be used
     *        when no description is required.
     */
    String onDestinationTransferEnd(TransferExitResult transferExitResult,
                                   String sourceAgentName,
                                   String destinationAgentName,
                                   Map<String, String>environmentMetaData,
                                   Map<String, String>transferMetaData,
                                   List<FileTransferResult>fileResults);
}

```

関連タスク

[ユーザー出口での MFT のカスタマイズ](#)

関連資料

[2173 ページの『SourceTransferStartExit.java インターフェース』](#)

[2172 ページの『SourceTransferEndExit.java インターフェース』](#)

[2148 ページの『DestinationTransferStartExit.java インターフェース』](#)

[2167 ページの『MonitorExit.java インターフェース』](#)

[2168 ページの『ProtocolBridgeCredentialExit.java インターフェース』](#)

DestinationTransferStartExit.java インターフェース

DestinationTransferStartExit.java

```
/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * © Copyright IBM Corp. 2008, 2026. All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with
 * IBM Corp.
 */
package com.ibm.wmqfte.exitpoint.api;

/**
 * An interface that is implemented by classes that want to be invoked as part of
 * user exit routine processing. This interface defines a method that will be
 * invoked immediately prior to starting a transfer on the agent acting as the
 * destination of the transfer.
 */
public interface DestinationTransferStartExit {

    /**
     * Invoked immediately prior to starting a transfer on the agent acting as
     * the destination of the transfer.
     *
     * @param sourceAgentName
     *     the name of the agent acting as the source of the transfer.
     *
     * @param destinationAgentName
     *     the name of the agent acting as the destination of the
     *     transfer. This is the name of the agent that the
     *     implementation of this method will be invoked from.
     *
     * @param environmentMetaData
     *     meta data about the environment in which the implementation
     *     of this method is running. This information can only be read,
     *     it cannot be updated by the implementation. The constants
     *     defined in EnvironmentMetaDataConstants class can
     *     be used to access the data held by this map.
     *
     * @param transferMetaData
     *     meta data to associate with the transfer. The information can
     *     only be read, it cannot be updated by the implementation. This
     *     map may also contain keys with IBM reserved names. These
     *     entries are defined in the TransferMetaDataConstants
     *     class and have special semantics.
     *
     * @param fileSpecs
     *     a list of file specifications that govern the file data to
     *     transfer. The implementation of this method can modify the
     *     entries in this list and the changes will be reflected in the
     *     files transferred. However, new entries may not be added and
     *     existing entries may not be removed.
     *
     * @return
     *     a transfer exit result object which is used to determine if the
     *     transfer should proceed, or be cancelled.
     */
    TransferExitResult onDestinationTransferStart(String sourceAgentName,
                                                  String destinationAgentName,
                                                  Map<String, String> environmentMetaData,
                                                  Map<String, String> transferMetaData,
                                                  List<Reference<String>> fileSpecs);
}
```

関連タスク

[ユーザー出口での MFT のカスタマイズ](#)

関連資料

[2173 ページの『SourceTransferStartExit.java インターフェース』](#)

[2172 ページの『SourceTransferEndExit.java インターフェース』](#)

[2147 ページの『DestinationTransferEndExit.java インターフェース』](#)

[2167 ページの『MonitorExit.java インターフェース』](#)

FileTransferResult.java インターフェース**FileTransferResult.java**

```

/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * © Copyright IBM Corp. 2008, 2026. All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with
 * IBM Corp.
 */

package com.ibm.wmqfte.exitroutine.api;

/**
 * Result information about a file transfer.
 */
public interface FileTransferResult {

    /** An enumeration for the <code>getCorrelatorType()</code> method. */
    public enum CorrelationInformationType {
        /** No correlation information is available for this result */
        NONE,
        /**
         * The correlation information relates to work done in
         * IBM Sterling File Gateway.
         */
        SFG
    }

    /**
     * Returns the source file specification, from which the file was transferred.
     *
     * @return the source file specification, from which the file was
     * transferred.
     */
    String getSourceFileSpecification();

    /**
     * Returns the destination file specification, to which the file was transferred.
     *
     * @return the destination file specification, to which the file was
     * transferred. A value of <code>null</code> may be returned
     * if the transfer did not complete successfully.
     */
    String getDestinationFileSpecification();

    /**
     * Returns the result of the file transfer operation.
     *
     * @return the result of the file transfer operation.
     */
    FileExitResult getExitResult();

    /**
     * @return an enumerated value that identifies the product to which this correlating
     * information relates.
     */
    CorrelationInformationType getCorrelatorType();

    /**
     * @return the first string component of the correlating identifier that relates
     * this transfer result to work done in another product. A value of null
     * may be returned either because the other product does not utilize a
     * string based correlation information or because there is no correlation
     * information.
     */
    String getString1Correlator();

    /**
     * @return the first long component of the correlating identifier that relates

```

```

        *      this transfer result to work done in another product. A value of zero
        *      is returned when there is no correlation information or the other
        *      product does not utilize long based correlation information or because
        *      the value really is zero!
        */
    long getLong1Correlator();
}

```

関連タスク

[ユーザー出口での MFT のカスタマイズ](#)

関連資料

[2173 ページの『SourceTransferStartExit.java インターフェース』](#)

[2148 ページの『DestinationTransferStartExit.java インターフェース』](#)

[2147 ページの『DestinationTransferEndExit.java インターフェース』](#)

[2167 ページの『MonitorExit.java インターフェース』](#)

[2168 ページの『ProtocolBridgeCredentialExit.java インターフェース』](#)

IOExit.java インターフェース

IOExit.java

```

/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * © Copyright IBM Corp. 2011, 2026. All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with
 * IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.io.IOException;
import java.util.Map;

import com.ibm.wmqfte.exitroutine.api.IOExitRecordResourcePath.RecordFormat;

/**
 * An interface that is implemented by classes that you want to be invoked as
 * part of user exit routine processing. This interface defines methods that
 * will be invoked during transfers to perform the underlying file system I/O
 * work for WMQFTE transfers.
 *
 * <p>
 * The {@link #initialize(Map)} method will be called once when the exit is
 * first installed. The WMQFTE agent properties are passed to this method, thus
 * enabling the exit to understand its environment.
 *
 * <p>
 * The {@link #isSupported(String)} method will be invoked during WMQFTE
 * transfers to determine whether the user exit should be used. If the
 * {@link #isSupported(String)} method returns a value of {@code true}, the
 * {@link #newPath(String)} method will be invoked for the paths specified for
 * the transfer request. The returned {@link IOExitPath} instance from a
 * {@link #newPath(String)} method invocation will then be used by the WMQFTE
 * transfer to obtain information about the resource and to transfer data to or
 * from the resource.
 *
 * <p>
 * To obtain transfer context for an I/O exit, a {@link SourceTransferStartExit}
 * or {@link DestinationTransferStartExit} as appropriate, should be installed
 * to enable information to be seen by this exit. The
 * {@link SourceTransferStartExit} or {@link DestinationTransferStartExit} are
 * passed the transfer's environment, metadata, and a list of file
 * specifications for the transfer. The paths for the file specifications are
 * the paths passed to the I/O exit's {@link #newPath(String)} method.
 *
 * <p>
 * Note also that the {@link #isSupported(String)} and {@link #newPath(String)}
 * methods might be called at other times by a WMQFTE agent and not just during
 * transfers. For example, at transfer setup time the I/O system is queried to
 * resolve the full resource paths for transfer.

```

```

*/
public interface IOExit {

    /**
     * Invoked once when the I/O exit is first required for use. It is intended
     * to initialize any resources that are required by the exit.
     *
     * @param agentProperties
     *      The values of properties defined for the WMQFTE agent. These
     *      values can only be read, they cannot be updated by the
     *      implementation.
     * @return {@code true} if the initialization is successful and {@code
     *      false} if unsuccessful. If {@code false} is returned from an
     *      exit, the exit will not be used.
     */
    boolean initialize(final Map<String, String> agentProperties);

    /**
     * Indicates whether this I/O user exit supports the specified path.
     *
     * <p>
     * This method is used by WMQFTE to determine whether the I/O user exit
     * should be used within a transfer. If no I/O user exit returns true for
     * this method, the default WMQFTE file I/O function will be used.
     *
     * @param path
     *      The path to the required I/O resource.
     * @return {@code true} if the specified path is supported by the I/O exit,
     *      {@code false} otherwise
     */
    boolean isSupported(String path);

    /**
     * Obtains a new {@link IOExitPath} instance for the specified I/O resource
     * path.
     *
     * <p>
     * This method will be invoked by WMQFTE only if the
     * {@link #isSupported(String)} method has been called for the path and
     * returned {@code true}.
     *
     * @param path
     *      The path to the required I/O resource.
     * @return A {@link IOExitPath} instance for the specified path.
     * @throws IOException
     *      If the path cannot be created for any reason.
     */
    IOExitPath newPath(String path) throws IOException;

    /**
     * Obtains a new {@link IOExitPath} instance for the specified I/O resource
     * path and passes record format and length information required by the
     * WMQFTE transfer.
     *
     * <p>
     * Typically this method will be called for the following cases:
     *
     * <ul>
     * <li>A path where a call to {@link #newPath(String)} has previously
     * returned a {@link IOExitRecordResourcePath} instance and WMQFTE is
     * re-establishing a new {@link IOExitPath} instance for the path, from an
     * internally-serialized state. The passed recordFormat and recordLength
     * will be the same as those for the original
     * {@link IOExitRecordResourcePath} instance.</li>
     * <li>A transfer destination path where the source of the transfer is
     * record oriented. The passed recordFormat and recordLength will be the
     * same as those for the source.</li>
     * </ul>
     *
     * The implementation can act on the record format and length information as
     * deemed appropriate. For example, for a destination agent if the
     * destination does not already exist and the source of the transfer is
     * record oriented, the passed recordFormat and recordLength information
     * could be used to create an appropriate record-oriented destination path.
     * If the destination path already exists, the passed recordFormat and
     * recordLength information could be used to perform a compatibility check
     * and throw an {@link IOException} if the path is not compatible. A
     * compatibility check could ensure that a record oriented path's record
     * format is the same as the passed record format or that the record length
     * is greater or equal to the passed record length.
     *
     * <p>
     * This method will be invoked by WMQFTE only if the
     * {@link #isSupported(String)} method has been called for the path and
     * returned {@code true}.
     *
     * @param path
     *      The path to the required I/O resource.
     */
}

```

```

* @param recordFormat
*     The advised record format.
* @param recordLength
*     The advised record length.
* @return A {@link IOExitPath} instance for the specified path.
* @throws IOException
*     If the path cannot be created for any reason. For example,
*     the passed record format or length is incompatible with the
*     path's actual record format or length.
*/
IOExitPath newPath(String path, RecordFormat recordFormat, int recordLength)
    throws IOException;

```

関連タスク

[MFT 転送入出力ユーザー出口の使用](#)

[ユーザー出口での MFT のカスタマイズ](#)

IOExitChannel.java インターフェース

IOExitChannel.java

```

/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * © Copyright IBM Corp. 2011, 2026. All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with
 * IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.io.IOException;
import java.nio.ByteBuffer;

/**
 * Represents a channel that enables data to be read from or written to an
 * {@link IOExitResourcePath} resource.
 */
public interface IOExitChannel {

    /**
     * Obtains the data size for the associated {@link IOExitResourcePath} in
     * bytes.
     *
     * @return The data size in bytes.
     * @throws IOException
     *     If a problem occurs while attempting obtain the size.
     */
    long size() throws IOException;

    /**
     * Closes the channel, flushing any buffered write data to the resource and
     * releasing any locks.
     *
     * @throws RecoverableIOException
     *     If a recoverable problem occurs while closing the resource.
     *     This means that WMQFTE can attempt to recover the transfer.
     * @throws IOException
     *     If some other I/O problem occurs. For example, the channel might
     *     already be closed.
     */
    void close() throws RecoverableIOException, IOException;

    /**
     * Reads data from this channel into the given buffer, starting at this
     * channel's current position, and updates the current position by the
     * amount of data read.
     *
     * <p>
     * Data is copied into the buffer starting at its current position and up to
     * its limit. On return, the buffer's position is updated to reflect the
     * number of bytes read.
     */

```

```

*
* @param buffer
*         The buffer that the data is to be copied into.
* @return The number of bytes read, which might be zero, or -1 if the end of
*         data has been reached.
* @throws RecoverableIOException
*         If a recoverable problem occurs while reading the data. For a
*         WMQFTE transfer this means that it will attempt to recover.
* @throws IOException
*         If some other I/O problem occurs. For a WMQFTE transfer this
*         means that it will be failed.
*/
int read(ByteBuffer buffer) throws RecoverableIOException, IOException;

/**
 * Writes data to this channel from the given buffer, starting at this
 * channel's current position, and updates the current position by the
 * amount of data written. The channel's resource is grown to accommodate
 * the data, if necessary.
 * <p>
 * Data is copied from the buffer starting at its current position and up to
 * its limit. On return, the buffer's position is updated to reflect the
 * number of bytes written.
 *
 * @param buffer
 *         The buffer containing the data to be written.
 * @return The number of bytes written, which might be zero.
 * @throws RecoverableIOException
 *         If a recoverable problem occurs while writing the data. For a
 *         WMQFTE transfer this means that it will attempt to recover.
 * @throws IOException
 *         If some other I/O problem occurs. For a WMQFTE transfer this
 *         means that it will be failed.
 */
int write(ByteBuffer buffer) throws RecoverableIOException, IOException;

/**
 * Forces any updates to this channel's resource to be written to its
 * storage device.
 * <p>
 * This method is required to force changes to both the resource's content
 * and any associated metadata to be written to storage.
 *
 * @throws RecoverableIOException
 *         If a recoverable problem occurs while performing the force.
 *         For a WMQFTE transfer this means that it will attempt to
 *         recover.
 * @throws IOException
 *         If some other I/O problem occurs. For a WMQFTE transfer this
 *         means that it will be failed.
 */
void force() throws RecoverableIOException, IOException;

/**
 * Attempts to lock the entire resource associated with the channel for
 * shared or exclusive access.
 * <p>
 * The intention is for this method not to block if the lock is currently
 * unavailable.
 *
 * @param shared
 *         {@code true} if a shared lock is required, {@code false} if an
 *         exclusive lock is required.
 * @return A {@link IOExitLock} instance representing the newly acquired
 *         lock or null if the lock cannot be obtained.
 * @throws IOException
 *         If a problem occurs while attempting to acquire the lock.
 */
IOExitLock tryLock(boolean shared) throws IOException;
}

```

関連タスク

[MFT 転送入出力ユーザー出口の使用](#)

[ユーザー出口での MFT のカスタマイズ](#)

IOExitLock.java インターフェース

IOExitLock.java

```
/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * © Copyright IBM Corp. 2011, 2026. All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with
 * IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.io.IOException;

/**
 * Represents a lock on a resource for either shared or exclusive access.
 * {@link IOExitLock} instances are returned from
 * {@link IOExitChannel#tryLock(boolean)} calls and WMQFTE will request the
 * release of the lock at the appropriate time during a transfer. Additionally, when
 * a {@link IOExitChannel#close()} method is called it will be the
 * responsibility of the channel to release any associated locks.
 */
public interface IOExitLock {

    /**
     * Releases the lock.
     * <p>
     * After this method has been successfully called the lock is to be deemed as invalid.
     *
     * @throws IOException
     *         If the channel associated with the lock is not open or
     *         another problem occurs while attempting to release the lock.
     */
    void release() throws IOException;

    /**
     * Indicates whether this lock is valid.
     * <p>
     * A lock is considered valid until its @ {@link #release()} method is
     * called or the associated {@link IOExitChannel} is closed.
     *
     * @return {@code true} if this lock is valid, {@code false} otherwise.
     */
    boolean isValid();

    /**
     * @return {@code true} if this lock is for shared access, {@code false} if
     *         this lock is for exclusive access.
     */
    boolean isShared();
}
```

関連タスク

[MFT 転送入出力ユーザー出口の使用](#)

[ユーザー出口での MFT のカスタマイズ](#)

IOExitPath.java インターフェース

IOExitPath.java

```
/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * © Copyright IBM Corp. 2011, 2026. All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or
```

```

* disclosure restricted by GSA ADP Schedule Contract with
* IBM Corp.
*/
package com.ibm.wmqfte.exitroutine.api;

/**
 * Represents an abstract path that can be inspected and queried by WMQFTE for
 * transfer purposes.
 * <p>
 * There are two types of path supported:
 * <ul>
 * <li>{@link IOExitResourcePath} - Represents a path that denotes a data
 * resource. For example, a file, directory, or group of database records.</li>
 * <li>{@link IOExitWildcardPath} - Represents a wildcard path that can be
 * expanded to multiple {@link IOExitResourcePath} instances.</li>
 * </ul>
 */
public abstract interface IOExitPath {

    /**
     * Obtains the abstract path as a {@link String}.
     *
     * @return The abstract path as a {@link String}.
     */
    String getPath();

    /**
     * Obtains the name portion of this abstract path as a {@link String}.
     * <p>
     * For example, a UNIX-style file system implementation evaluates the
     * path {@code /home/fteuser/file1.txt} as having a name of {@code
     * file1.txt}.
     *
     * @return the name portion of this abstract path as a {@link String}.
     */
    String getName();

    /**
     * Obtains the parent path for this abstract path as a {@link String}.
     * <p>
     * For example, a UNIX-style file system implementation evaluates the
     * path {@code /home/fteuser/file1.txt} as having a parent path of {@code
     * /home/fteuser}.
     *
     * @return The parent portion of the path as a {@link String}.
     */
    String getParent();

    /**
     * Obtains the abstract paths that match this abstract path.
     * <p>
     * If this abstract path denotes a directory resource, a list of paths
     * for all resources within the directory are returned.
     * <p>
     * If this abstract path denotes a wildcard, a list of all paths
     * matching the wildcard are returned.
     * <p>
     * Otherwise null is returned, because this abstract path probably denotes a
     * single file resource.
     *
     * @return An array of {@link IOExitResourcePath}s that
     * match this path, or null if this method is not applicable.
     */
    IOExitResourcePath[] listPaths();
}

```

関連タスク

[MFT 転送入出力ユーザー出口の使用](#)

[ユーザー出口での MFT のカスタマイズ](#)

IOExitProperties.java インターフェース

IOExitProperties.java

```

/*
 * Licensed Materials - Property of IBM

```



```

*
* "Restricted Materials of IBM"
*
* 5724-H72
*
* © Copyright IBM Corp. 2011, 2026. All Rights Reserved.
*
* US Government Users Restricted Rights - Use, duplication or
* disclosure restricted by GSA ADP Schedule Contract with
* IBM Corp.
*/
package com.ibm.wmqfte.exitroutine.api;

/**
 * Properties that determine how WMQFTE treats an {@link IOExitPath} for certain
 * aspects of I/O. For example, whether to use intermediate files.
 */
public class IOExitProperties {

    private boolean rereadSourceOnRestart = true;
    private boolean rechecksumSourceOnRestart = true;
    private boolean rechecksumDestinationOnRestart = true;
    private boolean useIntermediateFileAtDestination = true;
    private boolean requiresSingleThreadedChannelIO = false;

    /**
     * Determines whether the I/O exit implementation expects the resource to be
     * re-read from the start if a transfer is restarted.
     *
     * @return {@code true} if, on restart, the I/O exit expects the source
     * resource to be opened at the beginning and re-read from the
     * beginning (the {@link IOExitPath#openForRead(long)} method is
     * always invoked with 0L as an argument). {@code false} if, on
     * restart, the I/O exit expects the source to be opened at the
     * offset that the source agent intends to start reading from (the
     * {@link IOExitPath#openForRead(long)} method can be invoked with a
     * non-zero value as its argument).
     */
    public boolean getRereadSourceOnRestart() {
        return rereadSourceOnRestart;
    }

    /**
     * Sets the value to determine whether the I/O exit implementation expects
     * the resource to be re-read from the beginning if a transfer is restarted.
     * <p>
     * The default is {@code true}. The I/O exit should call this method when
     * required to change this value.
     *
     * @param rereadSourceOnRestart
     *      {@code true} if, on restart, the I/O exit expects the source
     *      resource to be opened at the beginning and re-read from the
     *      beginning (the {@link IOExitPath#openForRead(long)} method
     *      is always invoked with 0L as an argument). {@code false}
     *      if, on restart, the I/O exit expects the source to be opened
     *      at the offset that the source agent intends to start reading
     *      from (the {@link IOExitPath#openForRead(long)} method can be
     *      invoked with a non-zero value as its argument).
     */
    public void setRereadSourceOnRestart(boolean rereadSourceOnRestart) {
        this.rereadSourceOnRestart = rereadSourceOnRestart;
    }

    /**
     * Determines whether the I/O exit implementation requires the source
     * resource to be re-checksummed if the transfer is restarted.
     * Re-checksumming takes place only if the
     * {@link #getRereadSourceOnRestart()} method returns {@code true}.
     *
     * @return {@code true} if, on restart, the I/O exit expects the already-
     * transferred portion of the source to be re-checksummed for
     * inconsistencies. Use this option in environments
     * where the source could be changed during a restart. {@code
     * false} if, on restart, the I/O exit does not require the
     * already-transferred portion of the source to be re-checksummed.
     */
    public boolean getRechecksumSourceOnRestart() {
        return rechecksumSourceOnRestart;
    }

    /**
     * Sets the value to determine whether the I/O exit implementation requires

```

```

* the source resource to be re-checksummed if the transfer is restarted.
* Re-checksumming takes place only if the
* {@link #getRereadSourceOnRestart()} method returns {@code true}.
* <p>
* The default is {@code true}. The I/O exit should call this method when
* required to change this value.
*
* @param rechecksumSourceOnRestart
*     {@code true} if, on restart, the I/O exit expects the already
*     transferred portion of the source to be re-checksummed
*     for inconsistencies. Use this option in environments
*     where the source could be changed during a restart.
*     {@code false} if, on restart, the I/O exit does not
*     require the already-transferred portion of the source to be
*     re-checksummed.
*/
public void setRechecksumSourceOnRestart(boolean rechecksumSourceOnRestart) {
    this.rechecksumSourceOnRestart = rechecksumSourceOnRestart;
}

/**
* Determines whether the I/O exit implementation requires the destination
* resource to be re-checksummed if the transfer is restarted.
*
* @return {@code true} if, on restart, the I/O exit expects the already
*     transferred portion of the destination to be re-checksummed to
*     check for inconsistencies. This option should be used in
*     environments where the destination could have been changed while
*     a restart is occurring. {@code false} if, on restart, the I/O exit
*     does not require the already transferred portion of the
*     destination to be re-checksummed.
*/
public boolean getRechecksumDestinationOnRestart() {
    return rechecksumDestinationOnRestart;
}

/**
* Sets the value to determine whether the I/O exit implementation requires
* the destination resource to be re-checksummed if the transfer is
* restarted.
* <p>
* The default is {@code true}. The I/O exit should call this method when
* required to change this value.
*
* @param rechecksumDestinationOnRestart
*     {@code true} if, on restart, the I/O exit expects the already-
*     transferred portion of the destination to be re-checksummed
*     for inconsistencies. Use this option in environments
*     where the destination could have been changed during a
*     restart. {@code false} if, on restart, the I/O exit does not
*     require the already-transferred portion of the destination
*     to be re-checksummed.
*/
public void setRechecksumDestinationOnRestart(
    boolean rechecksumDestinationOnRestart) {
    this.rechecksumDestinationOnRestart = rechecksumDestinationOnRestart;
}

/**
* Determines whether the I/O exit implementation requires the use of an
* intermediate file when writing the data at the destination. The
* intermediate file mechanism is typically used to prevent an incomplete
* destination resource from being processed.
*
* @return {@code true} if data should be written to an intermediate file at
*     the destination and then renamed (to the requested destination
*     path name as specified in the transfer request) after the transfer is
*     complete. {@code false} if data should be written directly to the
*     requested destination path name without the use of an
*     intermediate file.
*/
public boolean getUseIntermediateFileAtDestination() {
    return useIntermediateFileAtDestination;
}

/**
* Sets the value to determine whether the I/O exit implementation requires
* the use of an intermediate file when writing the data at the destination.
* The intermediate file mechanism is typically used to prevent an
* incomplete destination resource from being processed.
*
* <p>

```

```

* The default is {@code true}. The I/O exit should call this method when
* required to change this value.
*
* @param useIntermediateFileAtDestination
*         {@code true} if data should be written to an intermediate file
*         at the destination and then renamed (to the requested
*         destination path name as specified in the transfer request) after
*         the transfer is complete. {@code false} if data should be written
*         directly to the requested destination path name without the
*         use of an intermediate file
*/
public void setUseIntermediateFileAtDestination(
    boolean useIntermediateFileAtDestination) {
    this.useIntermediateFileAtDestination = useIntermediateFileAtDestination;
}

/**
* Determines whether the I/O exit implementation requires
* {@link IOExitChannel} instances to be accessed by a single thread only.
*
* @return {@code true} if {@link IOExitChannel} instances are to be
*         accessed by a single thread only.
*/
public boolean requiresSingleThreadedChannelIO() {
    return requiresSingleThreadedChannelIO;
}

/**
* Sets the value to determine whether the I/O exit implementation requires
* channel operations for a particular instance to be accessed by a
* single thread only.
* <p>
* For certain I/O implementations it is necessary that resource path
* operations such as open, read, write, and close are invoked only from a
* single execution {@link Thread}. When set {@code true}, WMQFTE ensures
* that the following are invoked on a single thread:
* <ul>
* <li>{@link IOExitResourcePath#openForRead(long)} method and all methods of
* the returned {@link IOExitChannel} instance.</li>
* <li>{@link IOExitResourcePath#openForWrite(boolean)} method and all
* methods of the returned {@link IOExitChannel} instance.</li>
* </ul>
* <p>
* This has a slight performance impact, hence enable single-threaded channel
* I/O only when absolutely necessary.
* <p>
* The default is {@code false}. The I/O exit should call this method when
* required to change this value.
*
* @param requiresSingleThreadedChannelIO
*         {@code true} if {@link IOExitChannel} instances are to be
*         accessed by a single thread only.
*/
public void setRequiresSingleThreadedChannelIO(boolean requiresSingleThreadedChannelIO) {
    this.requiresSingleThreadedChannelIO = requiresSingleThreadedChannelIO;
}
}

```

関連タスク

[MFT 転送入出力ユーザー出口の使用](#)

[ユーザー出口での MFT のカスタマイズ](#)

IOExitRecordChannel.java インターフェース

IOExitRecordChannel.java

```

/*
* Licensed Materials - Property of IBM
*
* "Restricted Materials of IBM"
*
* 5724-H72
*
* © Copyright IBM Corp. 2011, 2026. All Rights Reserved.
*
* US Government Users Restricted Rights - Use, duplication or

```

```

* disclosure restricted by GSA ADP Schedule Contract with
* IBM Corp.
*/
package com.ibm.wmqfte.exitroutine.api;

import java.io.IOException;
import java.nio.ByteBuffer;

/**
 * Represents a channel that enables records of data to be read from or written
 * to an {@link IOExitRecordResourcePath} resource.
 * <p>
 * This is an extension of the {@link IOExitChannel} interface such that the
 * {@link #read(java.nio.ByteBuffer)} and {@link #write(java.nio.ByteBuffer)}
 * methods are expected to deal in whole records of data only. That is, the
 * {@link java.nio.ByteBuffer} returned from the read method and passed to the
 * write method is assumed to contain one or more complete records.
 */
public interface IOExitRecordChannel extends IOExitChannel {

    /**
     * Reads records from this channel into the given buffer, starting at this
     * channel's current position, and updates the current position by the
     * amount of data read.
     * <p>
     * Record data is copied into the buffer starting at its current position
     * and up to its limit. On return, the buffer's position is updated to
     * reflect the number of bytes read.
     * <p>
     * Only whole records are copied into the buffer.
     * <p>
     * For a fixed-record-format resource, this might be multiple records. The
     * amount of data in the return buffer does not necessarily need to be a
     * multiple of the record length, but the last record is still to be treated
     * as a complete record and padded as required by the caller.
     * <p>
     * For a variable-format resource, this is a single whole record of a size
     * corresponding to the amount of return data or multiple whole records with
     * all except the last being treated as records of maximum size.
     *
     * @param buffer
     *         The buffer that the record data is to be copied into.
     * @return The number of bytes read, which might be zero, or -1 if the end of
     *         data has been reached.
     * @throws RecoverableIOException
     *         If a recoverable problem occurs while reading the data. For a
     *         WMQFTE transfer this means that it will attempt to recover.
     * @throws IOException
     *         If some other I/O problem occurs, for example, if the passed
     *         buffer is insufficient to contain at least one complete
     *         record). For a WMQFTE transfer this means that it will be
     *         failed.
     */
    int read(ByteBuffer buffer) throws RecoverableIOException, IOException;

    /**
     * Writes records to this channel from the given buffer, starting at this
     * channel's current position, and updates the current position by the
     * amount of data written. The channel's resource is grown to accommodate
     * the data, if necessary.
     * <p>
     * Record data is copied from the buffer starting at its current position
     * and up to its limit. On return, the buffer's position is updated to
     * reflect the number of bytes written.
     * <p>
     * The buffer is expected to contain only whole records.
     * <p>
     * For a fixed-record-format resource, this might be multiple records and if
     * there is insufficient data in the buffer for a complete record, the
     * record is to be padded as required to complete the record.
     * <p>
     * For a variable-record format resource the buffer is normally expected to
     * contain a single record of length corresponding to the amount of data
     * within the buffer. However, if the amount of data within the buffer
     * exceeds the maximum record length, the implementation can either:
     * <ol>
     * <li>throw an {@link IOException} indicating that it cannot handle the
     * situation.</li>
     * <li>Consume a record's worth of data from the buffer, leaving the remaining
     * data within the buffer.</li>
     * <li>Consume all the buffer data and just write what it can to the current
     * record. This effectively truncates the data.</li>
     */

```

```

* <li>Consume all the buffer data and write to multiple records.</li>
* </ol>
*
* @param buffer
*         The buffer containing the data to be written.
* @return The number of bytes written, which might be zero.
* @throws RecoverableIOException
*         If a recoverable problem occurs while writing the data. For a
*         WMQFTE transfer this means that it will attempt to recover.
* @throws IOException
*         If some other I/O problem occurs. For a WMQFTE transfer this
*         means that it will be failed.
*/
int write(ByteBuffer buffer) throws RecoverableIOException, IOException;
}

```

関連タスク

[MFT 転送入出力ユーザー出口の使用](#)

[ユーザー出口での MFT のカスタマイズ](#)

IOExitRecordResourcePath.java インターフェース

IOExitRecordResourcePath.java

```

/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * © Copyright IBM Corp. 2011, 2026. All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with
 * IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.io.IOException;

/**
 * Represents a path that denotes a record-oriented data resource (for example,
 * a z/OS data set). It allows the data to be located, the record format to be
 * understood, and {@link IOExitRecordChannel} instances to be created for read
 * or write operations.
 */
public interface IOExitRecordResourcePath extends IOExitResourcePath {

    /**
     * Record formats for record-oriented resources.
     */
    public enum RecordFormat {
        FIXED, VARIABLE
    }

    /**
     * Obtains the record length for records that are maintained by the resource
     * denoted by this abstract path.
     * <p>
     * For a resource with fixed-length records, the data for each record read
     * and written is assumed to be this length.
     * <p>
     * For a resource with variable-length records, this is the maximum length
     * for a record's data.
     * <p>
     * This method should return a value greater than zero, otherwise it can
     * result in the failure of a WMQFTE transfer that involves this abstract
     * path.
     *
     * @return The record length, in bytes, for records maintained by the
     *         resource.
     */
    int getRecordLength();
}

```

```

/**
 * Obtains record format, as a {@link RecordFormat} instance, for records
 * that are maintained by the resource denoted by this abstract path.
 *
 * @return A {@link RecordFormat} instance for the record format for records
 *         that are maintained by the resource denoted by this abstract
 *         path.
 */
RecordFormat getRecordFormat();

/**
 * Opens a {@link IOExitRecordChannel} instance for reading data from the
 * resource denoted by this abstract path. The current data byte position
 * for the resource is expected to be the passed position value, such that
 * when {@link IOExitRecordChannel#read(java.nio.ByteBuffer)} is called,
 * data starting from that position is read.
 * <p>
 * Note that the data byte read position will be on a record boundary.
 *
 * @param position
 *       The required data byte read position.
 * @return A new {@link IOExitRecordChannel} instance allowing data to be
 *         read from the resource denoted by this abstract path.
 * @throws RecoverableIOException
 *       If a recoverable problem occurs while attempting to open the
 *       resource for reading. This means that WMQFTE can attempt to
 *       recover the transfer.
 * @throws IOException
 *       If some other I/O problem occurs.
 */
IOExitRecordChannel openForRead(long position)
    throws RecoverableIOException, IOException;

/**
 * Opens a {@link IOExitRecordChannel} instance for writing data to the
 * resource denoted by this abstract path. Writing of data, using the
 * {@link IOExitRecordChannel#write(java.nio.ByteBuffer)} method, starts at
 * either the beginning of the resource or end of the current data for the
 * resource, depending on the specified append parameter.
 *
 * @param append
 *       When {@code true} indicates that data written to the resource
 *       should be appended to the end of the current data. When
 *       {@code false} indicates that writing of data is to start at
 *       the beginning of the resource; any existing data is lost.
 * @return A new {@link IOExitRecordChannel} instance allowing data to be
 *         written to the resource denoted by this abstract path.
 * @throws RecoverableIOException
 *       If a recoverable problem occurs while attempting to open the
 *       resource for writing. This means that WMQFTE can attempt to
 *       recover the transfer.
 * @throws IOException
 *       If some other I/O problem occurs.
 */
IOExitRecordChannel openForWrite(boolean append)
    throws RecoverableIOException, IOException;
}

```

関連タスク

[MFT 転送入出力ユーザー出口の使用](#)

[ユーザー出口での MFT のカスタマイズ](#)

IOExitResourcePath.java インターフェース

IOExitResourcePath.java

```

/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * © Copyright IBM Corp. 2011, 2026. All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or

```

```

* disclosure restricted by GSA ADP Schedule Contract with
* IBM Corp.
*/
package com.ibm.wmqfte.exitroutine.api;

import java.io.IOException;

/**
 * Represents a path that denotes a data resource (for example, a file,
 * directory, or group of database records). It allows the data to be located
 * and {@link IOExitChannel} instances to be created for read or write
 * operations.
 * <p>
 * There are two types of data resources as follows:
 * <ul>
 * <li>Directory - a container for other data resources. The
 * {@link #isDirectory()} method returns {@code true} for these.</li>
 * <li>File - a data container. This allows data to be read from or written to
 * it. The {@link #isFile()} method returns {@code true} for these.</li>
 * </ul>
 */
public interface IOExitResourcePath extends IOExitPath {

    /**
     * Creates a new {@link IOExitResourcePath} instance for a child path of the
     * resource denoted by this abstract path.
     * <p>
     * For example, with a UNIX-style path, {@code
     * IOExitResourcePath("/home/fteuser/test").newPath("subtest")} could be
     * equivalent to: {@code IOExitResourcePath("/home/fteuser/test/subtest")}
     *
     * @param child
     *         The child path name.
     * @return A new {@link IOExitResourcePath} instance that represents a child
     *         of this path.
     */
    IOExitResourcePath newPath(final String child);

    /**
     * Creates the directory path for the resource denoted by this abstract
     * path, including any necessary but nonexistent parent directories. If the
     * directory path already exists, this method has no effect.
     * <p>
     * If this operation fails, it might have succeeded in creating some of the
     * necessary parent directories.
     *
     * @throws IOException
     *         If the directory path cannot be fully created, when it does
     *         not already exist.
     */
    void makePath() throws IOException;

    /**
     * Obtains the canonical path of the abstract path as a {@link String}.
     * <p>
     * A canonical path is defined as being absolute and unique. For example,
     * the path can be represented as UNIX-style relative path: {@code
     * test/file.txt} but the absolute and unique canonical path representation
     * is: {@code /home/fteuser/test/file.txt}
     *
     * @return The canonical path as a {@link String}.
     * @throws IOException
     *         If the canonical path cannot be determined for any reason.
     */
    String getCanonicalPath() throws IOException;

    /**
     * Tests if this abstract path is an absolute path.
     * <p>
     * For example, a UNIX-style path, {@code /home/fteuser/test} is an absolute
     * path, whereas {@code fteuser/test} is not.
     *
     * @return {@code true} if this abstract path is an absolute path, {@code
     *         false} otherwise.
     */
    boolean isAbsolute();

    /**
     * Tests if the resource denoted by this abstract path exists.
     *
     * @return {@code true} if the resource denoted by this abstract path
     *         exists, {@code false} otherwise.
     */

```

```

    * @throws IOException
    *     If the existence of the resource cannot be determined for any
    *     reason.
    */
    boolean exists() throws IOException;

    /**
     * Tests whether the calling application can read the resource denoted by
     * this abstract path.
     *
     * @return {@code true} if the resource for this path exists and can be
     *         read, {@code false} otherwise.
     * @throws IOException
     *     If a problem occurs while attempting to determine if the
     *     resource can be read.
     */
    boolean canRead() throws IOException;

    /**
     * Tests whether the calling application can modify the resource denoted by
     * this abstract path.
     *
     * @return {@code true} if the resource for this path exists and can be
     *         modified, {@code false} otherwise.
     * @throws IOException
     *     If a problem occurs while attempting to determine if the
     *     resource can be modified.
     */
    boolean canWrite() throws IOException;

    /**
     * Tests whether the specified user is permitted to read the resource
     * denoted by this abstract path.
     * <p>
     * When WMQFTE invokes this method, the user identifier is the MQMD user
     * identifier for the requesting transfer.
     *
     * @param userId
     *     User identifier to test for access.
     * @return {@code true} if the resource for this abstract path exists and is
     *         permitted to be read by the specified user, {@code false}
     *         otherwise.
     * @throws IOException
     *     If a problem occurs while attempting to determine if the user
     *     is permitted to read the resource.
     */
    boolean readPermitted(String userId) throws IOException;

    /**
     * Tests whether the specified user is permitted to modify the resource
     * denoted by this abstract path.
     * <p>
     * When WMQFTE invokes this method, the user identifier is the MQMD user
     * identifier for the requesting transfer.
     *
     * @param userId
     *     User identifier to test for access.
     * @return {@code true} if the resource for this abstract path exists and is
     *         permitted to be modified by the specified user, {@code false}
     *         otherwise.
     * @throws IOException
     *     If a problem occurs while attempting to determine if the user
     *     is permitted to modify the resource.
     */
    boolean writePermitted(String userId) throws IOException;

    /**
     * Tests if the resource denoted by this abstract path is a directory-type
     * resource.
     *
     * @return {@code true} if the resource denoted by this abstract path is a
     *         directory type resource, {@code false} otherwise.
     */
    boolean isDirectory();

    /**
     * Creates the resource denoted by this abstract path, if it does not
     * already exist.
     *
     * @return {@code true} if the resource does not exist and was successfully
     *         created, {@code false} if the resource already existed.
     * @throws RecoverableIOException

```



```

*           If a recoverable problem occurs while attempting to create
*           the resource. This means that WMQFTE can attempt to recover
*           the transfer.
* @throws IOException
*           If some other I/O problem occurs.
*/
boolean createNewPath() throws RecoverableIOException, IOException;

/**
* Tests if the resource denoted by this abstract path is a file-type
* resource.
*
* @return {@code true} if the resource denoted by this abstract path is a
*         file type resource, {@code false} otherwise.
*/
boolean isFile();

/**
* Obtains the last modified time for the resource denoted by this abstract
* path.
* <p>
* This time is measured in milliseconds since the epoch (00:00:00 GMT,
* January 1, 1970).
*
* @return The last modified time for the resource denoted by this abstract
*         path, or a value of 0L if the resource does not exist or a
*         problem occurs.
*/
long lastModified();

/**
* Deletes the resource denoted by this abstract path.
* <p>
* If the resource is a directory, it must be empty for the delete to work.
*
* @throws IOException
*         If the delete of the resource fails for any reason.
*/
void delete() throws IOException;

/**
* Renames the resource denoted by this abstract path to the specified
* destination abstract path.
* <p>
* The rename should still be successful if the resource for the specified
* destination abstract path already exists and it is possible to replace
* it.
*
* @param destination
*         The new abstract path for the resource denoted by this
*         abstract path.
* @throws IOException
*         If the rename of the resource fails for any reason.
*/
void renameTo(IOExceptionResourcePath destination) throws IOException;

/**
* Creates a new path to use for writing to a temporary resource that did
* not previously exist.
* <p>
* The implementation can choose the abstract path name for the temporary
* resource. However, for clarity and problem diagnosis, the abstract path
* name for the temporary resource should be based on this abstract path
* name with the specified suffix appended and additional characters to make
* the path unique (for example, sequence numbers), as required.
* <p>
* When WMQFTE transfers data to a destination it normally attempts to first
* write to a temporary resource then on transfer completion renames the
* temporary resource to the required destination. This method is called by
* WMQFTE to create a new temporary resource path. The returned path should
* be new and the resource should not previously exist.
*
* @param suffix
*         Recommended suffix to use for the generated temporary path.
*
* @return A new {@link IOExceptionResourcePath} instance for the temporary
*         resource path, that did not previously exist.
* @throws RecoverableIOException
*         If a recoverable problem occurs whilst attempting to create
*         the temporary resource. This means that WMQFTE can attempt to
*         recover the transfer.
* @throws IOException

```

```

        *           If some other I/O problem occurs.
        */
IOExitResourcePath createTempPath(String suffix)
    throws RecoverableIOException, IOException;

/**
 * Opens a {@link IOExitChannel} instance for reading data from the resource
 * denoted by this abstract path. The current data byte position for the
 * resource is expected to be the passed position value, such that when
 * {@link IOExitChannel#read(java.nio.ByteBuffer)} is called, data starting
 * from that position is read.
 *
 * @param position
 *     The required data byte read position.
 * @return A new {@link IOExitChannel} instance allowing data to be read
 *     from the resource denoted by this abstract path.
 * @throws RecoverableIOException
 *     If a recoverable problem occurs while attempting to open the
 *     resource for reading. This means that WMQFTE can attempt to
 *     recover the transfer.
 * @throws IOException
 *     If some other I/O problem occurs.
 */
IOExitChannel openForRead(long position) throws RecoverableIOException,
    IOException;

/**
 * Opens a {@link IOExitChannel} instance for writing data to the resource
 * denoted by this abstract path. Writing of data, using the
 * {@link IOExitChannel#write(java.nio.ByteBuffer)} method, starts at either
 * the beginning of the resource or end of the current data for the
 * resource, depending on the specified append parameter.
 *
 * @param append
 *     When {@code true} indicates that data written to the resource
 *     should be appended to the end of the current data. When
 *     {@code false} indicates that writing of data is to start at
 *     the beginning of the resource; any existing data is lost.
 * @return A new {@link IOExitChannel} instance allowing data to be written
 *     to the resource denoted by this abstract path.
 * @throws RecoverableIOException
 *     If a recoverable problem occurs whilst attempting to open the
 *     resource for writing. This means that WMQFTE can attempt to
 *     recover the transfer.
 * @throws IOException
 *     If some other I/O problem occurs.
 */
IOExitChannel openForWrite(boolean append) throws RecoverableIOException,
    IOException;

/**
 * Tests if the resource denoted by this abstract path is in use by another
 * application. Typically, this is because another application has a lock on
 * the resource either for shared or exclusive access.
 *
 * @return {@code true} if resource denoted by this abstract path is in use
 *     by another application, {@code false} otherwise.
 */
boolean inUse();

/**
 * Obtains a {@link IOExitProperties} instance for properties associated
 * with the resource denoted by this abstract path.
 *
 * <p>
 * WMQFTE will read these properties to govern how a transfer behaves when
 * interacting with the resource.
 *
 * @return A {@link IOExitProperties} instance for properties associated
 *     with the resource denoted by this abstract path.
 */
IOExitProperties getProperties();
}

```

関連タスク

[MFT 転送入出力ユーザー出口の使用](#)

[ユーザー出口での MFT のカスタマイズ](#)

IOExitWildcardPath.java インターフェース

IOExitWildcardPath.java

```
/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * Copyright IBM Corp. 2011, 2026. All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with
 * IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

/**
 * Represents a path that denotes a wildcard. This can be used to match multiple
 * resource paths.
 */
public interface IOExitWildcardPath extends IOExitPath {
```

関連タスク

[MFT 転送入出力ユーザー出口の使用](#)

[ユーザー出口での MFT のカスタマイズ](#)

MonitorExit.java インターフェース

MonitorExit.java

```
/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * Copyright IBM Corp. 2009, 2026. All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with
 * IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.util.Map;

/**
 * An interface that is implemented by classes that want to be invoked as part of
 * user exit routine processing. This interface defines a method that will be
 * invoked immediately prior to starting a task as the result of a monitor trigger
 */
public interface MonitorExit {

    /**
     * Invoked immediately prior to starting a task as the result of a monitor
     * trigger.
     *
     * @param environmentMetaData
     *      meta data about the environment in which the implementation
     *      of this method is running. This information can only be read,
     *      it cannot be updated by the implementation. The constant
     *      defined in EnvironmentMetaDataConstants class can
     *      be used to access the data held by this map.
     *
     * @param monitorMetaData
     *      meta data to associate with the monitor. The meta data passed
     *      to this method can be altered, and the changes will be
     *      reflected in subsequent exit routine invocations. This map
     *      also contains keys with IBM reserved names. These entries are
     *      defined in the MonitorMetaDataConstants class and
     *      have special semantics. The the values of the IBM reserved names
     *      cannot be modified by the exit
     *
     * @param taskDetails
     */
}
```

```

    *           An XML String representing the task to be executed as a result of
    *           the monitor triggering. This XML string may be modified by the
    *           exit
    *
    * @return    a monitor exit result object which is used to determine if the
    *           task should proceed, or be cancelled.
    */
    MonitorExitResult onMonitor(Map<String, String> environmentMetaData,
                               Map<String, String> monitorMetaData,
                               Reference<String> taskDetails);
}

```

関連タスク

[MFT リソースのモニター](#)

[ユーザー出口での MFT のカスタマイズ](#)

関連資料

[2173 ページの『SourceTransferStartExit.java インターフェース』](#)

[2172 ページの『SourceTransferEndExit.java インターフェース』](#)

[2148 ページの『DestinationTransferStartExit.java インターフェース』](#)

[2147 ページの『DestinationTransferEndExit.java インターフェース』](#)

[2168 ページの『ProtocolBridgeCredentialExit.java インターフェース』](#)

ProtocolBridgeCredentialExit.java インターフェース

ProtocolBridgeCredentialExit.java

```

/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * © Copyright IBM Corp. 2008, 2026. All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with
 * IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.util.Map;

/**
 * An interface that is implemented by classes that are to be invoked as part of
 * user exit routine processing. This interface defines methods that will
 * be invoked by a protocol bridge agent to map the MQ user ID of the transfer to credentials
 * that are to be used to access the protocol server.
 * There will be one instance of each implementation class per protocol bridge agent. The methods
 * can be called from different threads so the methods must be synchronized.
 */
public interface ProtocolBridgeCredentialExit {

    /**
     * Invoked once when a protocol bridge agent is started. It is intended to initialize
     * any resources that are required by the exit
     *
     * @param bridgeProperties
     *           The values of properties defined for the protocol bridge.
     *           These values can only be read, they cannot be updated by
     *           the implementation.
     *
     * @return    true if the initialization is successful and false if unsuccessful
     *           If false is returned from an exit the protocol bridge agent will not
     *           start
     */
    public boolean initialize(final Map<String> bridgeProperties);

}

```

```

    * Invoked once for each transfer to map the MQ user ID in the transfer message to the
    * credentials to be used to access the protocol server
    *
    * @param mqUserId The MQ user ID from which to map to the credentials to be used
    *                access the protocol server
    * @return        A credential exit result object that contains the result of the map and
    *                the credentials to use to access the protocol server
    */

public CredentialExitResult mapMQUserId(final String mqUserId);

/**
 * Invoked once when a protocol bridge agent is shutdown. It is intended to release
 * any resources that were allocated by the exit
 *
 * @param bridgeProperties
 *        The values of properties defined for the protocol bridge.
 *        These values can only be read, they cannot be updated by
 *        the implementation.
 *
 * @return
 */
public void shutdown(final Map<String> bridgeProperties);
}

```

関連タスク

[ユーザー出口での MFT のカスタマイズ](#)

[出口クラスを使用したファイル・サーバーの資格情報のマップ](#)

ProtocolBridgeCredentialExit2.java インターフェース

ProtocolBridgeCredentialExit2.java

```

/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * © Copyright IBM Corp. 2011, 2026. All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with
 * IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

/**
 * An interface that is implemented by classes that are invoked as part of user
 * exit routine processing. This interface defines methods that are invoked by a
 * protocol bridge agent to map the MQ user ID of the transfer to credentials
 * used to access a specified protocol bridge server. There will be one instance
 * of each implementation class for each protocol bridge agent. The methods can
 * be called from different threads so the methods must be synchronized.
 */
public interface ProtocolBridgeCredentialExit2 extends
    ProtocolBridgeCredentialExit {

    /**
     * Invoked once for each transfer to map the MQ user ID in the transfer
     * message to the credentials used to access a specified protocol server.
     *
     * @param endPoint
     *        Information that describes the protocol server to be accessed.
     * @param mqUserId
     *        The MQ user ID from which to map the credentials used to
     *        access the protocol server.
     * @return A {@link CredentialExitResult} instance that contains the result
     *        of the map and the credentials to use to access the protocol
     *        server.
     */
    public CredentialExitResult mapMQUserId(
        final ProtocolServerEndPoint endPoint, final String mqUserId);
}

```

関連タスク

[ユーザー出口での MFT のカスタマイズ](#)

[出口クラスを使用したファイル・サーバーの資格情報のマップ](#)

ProtocolBridgePropertiesExit2.java インターフェース

ProtocolBridgePropertiesExit2.java

```
/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * © Copyright IBM Corp. 2011, 2026. All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with
 * IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.util.Map;
import java.util.Properties;

/**
 * An interface that is implemented by classes that are to be invoked as part of
 * user exit routine processing. This interface defines methods that will be
 * invoked by a protocol bridge agent to look up properties for protocol servers
 * that are referenced in transfers.
 * <p>
 * There will be one instance of each implementation class for each protocol
 * bridge agent. The methods can be called from different threads so the methods
 * must be synchronised.
 */
public interface ProtocolBridgePropertiesExit2 {

    /**
     * Invoked once when a protocol bridge agent is started. It is intended to
     * initialize any resources that are required by the exit.
     *
     * @param bridgeProperties
     *      The values of properties defined for the protocol bridge.
     *      These values can only be read, they cannot be updated by the
     *      implementation.
     * @return {@code true} if the initialization is successful and {@code
     *      false} if unsuccessful. If {@code false} is returned from an exit
     *      the protocol bridge agent will not start.
     */
    public boolean initialize(final Map<String, String> bridgeProperties);

    /**
     * Invoked when the Protocol Bridge needs to access the protocol bridge credentials XML file.
     *
     * @return a {@link String} object giving the location of the ProtocolBridgeCredentials.xml
     */
    public String getCredentialLocation ();

    /**
     * Obtains a set of properties for the specified protocol server name.
     * <p>
     * The returned {@link Properties} must contain entries with key names
     * corresponding to the constants defined in
     * {@link ProtocolServerPropertyConstants} and in particular must include an
     * entry for all appropriate constants described as required.
     *
     * @param protocolServerName
     *      The name of the protocol server whose properties are to be
     *      returned. If a null or a blank value is specified, properties
     *      for the default protocol server are to be returned.
     * @return The {@link Properties} for the specified protocol server, or null
     *      if the server cannot be found.
     */
    public Properties getProtocolServerProperties(
        final String protocolServerName);
}
```

```

/**
 * Invoked once when a protocol bridge agent is shut down. It is intended to
 * release any resources that were allocated by the exit.
 *
 * @param bridgeProperties
 *         The values of properties defined for the protocol bridge.
 *         These values can only be read, they cannot be updated by the
 *         implementation.
 */
public void shutdown(final Map<String, String> bridgeProperties);
}

```

関連タスク

[ProtocolBridgePropertiesExit: プロトコル・ファイル・サーバー・プロパティの検索](#)

[ユーザー出口での MFT のカスタマイズ](#)

[出口クラスを使用したファイル・サーバーの資格情報のマップ](#)

SourceFileExitFileSpecification.java クラス

SourceFileExitFileSpecification.java

```

/**
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * © Copyright IBM Corp. 2012, 2026. All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with
 * IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.util.Map;

/**
 * A specification of the file names to use for a file transfer, as evaluated by the
 * agent acting as the source of the transfer.
 */
public final class SourceFileExitFileSpecification {

    private final String sourceFileSpecification;
    private final String destinationFileSpecification;
    private final Map<String, String> sourceFileMetaData;
    private final Map<String, String> destinationFileMetaData;

    /**
     * Constructor. Creates a source file exit file specification.
     *
     * @param sourceFileSpecification
     *        the source file specification to associate with the source file
     *        exit file specification.
     *
     * @param destinationFileSpecification
     *        the destination file specification to associate with the
     *        source file exit file specification.
     *
     * @param sourceFileMetaData
     *        the source file meta data.
     *
     * @param destinationFileMetaData
     *        the destination file meta data
     */
    public SourceFileExitFileSpecification(final String sourceFileSpecification,
                                           final String destinationFileSpecification,
                                           final Map<String, String> sourceFileMetaData,
                                           final Map<String, String> destinationFileMetaData) {

        this.sourceFileSpecification = sourceFileSpecification;
        this.destinationFileSpecification = destinationFileSpecification;
        this.sourceFileMetaData = sourceFileMetaData;
        this.destinationFileMetaData = destinationFileMetaData;
    }
}

```

```

/**
 * Returns the destination file specification.
 *
 * @return the destination file specification. This represents the location,
 *         on the agent acting as the destination for the transfer, where the
 *         file should be written. Exit routines installed into the agent
 *         acting as the destination for the transfer may override this value.
 */
public String getDestination() {
    return destinationFileSpecification;
}

/**
 * Returns the source file specification.
 *
 * @return the source file specification. This represents the location where
 *         the file data will be read from.
 */
public String getSource() {
    return sourceFileSpecification;
}

/**
 * Returns the file meta data that relates to the source file specification.
 *
 * @return the file meta data that relates to the source file specification.
 */
public Map<String, String> getSourceFileMetaData() {
    return sourceFileMetaData;
}

/**
 * Returns the file meta data that relates to the destination file specification.
 *
 * @return the file meta data that relates to the destination file specification.
 */
public Map<String, String> getDestinationFileMetaData() {
    return destinationFileMetaData;
}
}

```

関連概念

[2135 ページの『MFT ユーザー出口のメタデータ』](#)

Managed File Transfer のユーザー出口ルーチンに提供できるメタデータには、環境、転送、およびファイル・メタデータの 3 つの異なるタイプがあります。このメタデータは、Java のキー/値ペアのマップとして示されます。

SourceTransferEndExit.java インターフェース

SourceTransferEndExit.java

```

/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * © Copyright IBM Corp. 2008, 2026. All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with
 * IBM Corp.
 */
package com.ibm.wmqfte.exitpoint.api;

/**
 * An interface that is implemented by classes that want to be invoked as part of
 * user exit routine processing. This interface defines a method that will be
 * invoked immediately after completing a transfer on the agent acting as the
 * source of the transfer.
 */
public interface SourceTransferEndExit {

    /**
     * Invoked immediately after the completion of a transfer on the agent acting as
     * the source of the transfer.
     */
}

```



```

*
* @param transferExitResult
*       a result object reflecting whether or not the transfer completed
*       successfully.
*
* @param sourceAgentName
*       the name of the agent acting as the source of the transfer.
*       This is the name of the agent that the implementation of this
*       method will be invoked from.
*
* @param destinationAgentName
*       the name of the agent acting as the destination of the
*       transfer.
*
* @param environmentMetaData
*       meta data about the environment in which the implementation
*       of this method is running. This information can only be read,
*       it cannot be updated by the implementation. The constants
*       defined in <code>EnvironmentMetaDataConstants</code> class can
*       be used to access the data held by this map.
*
* @param transferMetaData
*       meta data to associate with the transfer. The information can
*       only be read, it cannot be updated by the implementation. This
*       map may also contain keys with IBM reserved names. These
*       entries are defined in the <code>TransferMetaDataConstants</code>
*       class and have special semantics.
*
* @param fileResults
*       a list of file transfer result objects that describe the source
*       file name, destination file name and result of each file transfer
*       operation attempted.
*
* @return an optional description to enter into the log message describing
*         transfer completion. A value of <code>null</code> can be used
*         when no description is required.
*/
String onSourceTransferEnd(TransferExitResult transferExitResult,
                          String sourceAgentName,
                          String destinationAgentName,
                          Map<String, String>environmentMetaData,
                          Map<String, String>transferMetaData,
                          List<FileTransferResult>fileResults);
}

```

関連タスク

[ユーザー出口での MFT のカスタマイズ](#)

関連資料

[2173 ページの『SourceTransferStartExit.java インターフェース』](#)

[2148 ページの『DestinationTransferStartExit.java インターフェース』](#)

[2147 ページの『DestinationTransferEndExit.java インターフェース』](#)

[2167 ページの『MonitorExit.java インターフェース』](#)

[2168 ページの『ProtocolBridgeCredentialExit.java インターフェース』](#)

SourceTransferStartExit.java インターフェース

SourceTransferStartExit.java

```

/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * □ Copyright IBM Corp. 2008, 2026. All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with
 * IBM Corp.
 */

```

```

package com.ibm.wmqfte.exitpoint.api;

import java.util.List;
import java.util.Map;

/**
 * An interface that is implemented by classes that want to be invoked as part of
 * user exit routine processing. This interface defines a method that will be
 * invoked immediately prior to starting a transfer on the agent acting as the
 * source of the transfer.
 */
public interface SourceTransferStartExit {

    /**
     * Invoked immediately prior to starting a transfer on the agent acting as
     * the source of the transfer.
     *
     * @param sourceAgentName
     *     the name of the agent acting as the source of the transfer.
     *     This is the name of the agent that the implementation of this
     *     method will be invoked from.
     *
     * @param destinationAgentName
     *     the name of the agent acting as the destination of the
     *     transfer.
     *
     * @param environmentMetaData
     *     meta data about the environment in which the implementation
     *     of this method is running. This information can only be read,
     *     it cannot be updated by the implementation. The constants
     *     defined in <code>EnvironmentMetaDataConstants</code> class can
     *     be used to access the data held by this map.
     *
     * @param transferMetaData
     *     meta data to associate with the transfer. The meta data passed
     *     to this method can be altered, and the changes to will be
     *     reflected in subsequent exit routine invocations. This map may
     *     also contain keys with IBM reserved names. These entries are
     *     defined in the <code>TransferMetaDataConstants</code> class and
     *     have special semantics.
     *
     * @param fileSpecs
     *     a list of file specifications that govern the file data to
     *     transfer. The implementation of this method can add entries,
     *     remove entries, or modify entries in this list and the changes
     *     will be reflected in the files transferred.
     *
     * @return
     *     a transfer exit result object which is used to determine if the
     *     transfer should proceed, or be cancelled.
     */
    TransferExitResult onSourceTransferStart(String sourceAgentName,
        String destinationAgentName,
        Map<String, String> environmentMetaData,
        Map<String, String> transferMetaData,
        List<SourceFileExitFileSpecification> fileSpecs);
}

```

関連タスク

[ユーザー出口での MFT のカスタマイズ](#)

関連資料

[2171 ページの『SourceFileExitFileSpecification.java クラス』](#)

[2172 ページの『SourceTransferEndExit.java インターフェース』](#)

[2148 ページの『DestinationTransferStartExit.java インターフェース』](#)

[2147 ページの『DestinationTransferEndExit.java インターフェース』](#)

[2167 ページの『MonitorExit.java インターフェース』](#)

[2168 ページの『ProtocolBridgeCredentialExit.java インターフェース』](#)

TransferExitResult.java インターフェース

TransferExitResult.java

TransferExitResult.java は次のような内容になっています：

```
/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * © Copyright IBM Corp. 2008, 2026. All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with
 * IBM Corp.
 */

package com.ibm.wmqfte.exitroutine.api;

/**
 * The result of invoking a transfer exit routine. It is composed of a result
 * code, which determines if the transfer should proceed, and an optional explanatory
 * message. The explanation, if present, is entered into the log message.
 */
public class TransferExitResult {

    private final TransferExitResultCode resultCode;
    private final String explanation;

    /**
     * For convenience, a static "proceed" result with no associated explanation
     * message.
     */
    public static final TransferExitResult PROCEED_RESULT =
        new TransferExitResult(TransferExitResultCode.PROCEED, null);

    /**
     * Constructor. Creates a transfer exit result object with a specified result
     * code and explanation.
     *
     * @param resultCode
     *     The result code to associate with the exit result being created.
     *
     * @param explanation
     *     The explanation to associate with the exit result being created.
     *     A value of <code>null</code> can be specified to indicate no
     *     explanation.
     */
    public TransferExitResult(TransferExitResultCode resultCode, String explanation) {
        this.resultCode = resultCode;
        this.explanation = explanation;
    }

    /**
     * Returns the explanation associated with this transfer exit result.
     *
     * @return
     *     the explanation associated with this exit result.
     */
    public String getExplanation() {
        return explanation;
    }

    /**
     * Returns the result code associated with this transfer exit result.
     *
     * @return
     *     the result code associated with this exit result.
     */
    public TransferExitResultCode getResultCode() {
        return resultCode;
    }

    /**
     * Returns a string representation of this TransferExitResult.
     * @return a string representation of this TransferExitResult.
     */
    @Override
    public String toString() {
        return "[" + getResultCode() + ", " + getExplanation() + "]";
    }
}
```

```
}  
}
```

関連タスク

[ユーザー出口での MFT のカスタマイズ](#)

関連資料

[2173 ページの『SourceTransferStartExit.java インターフェース』](#)

[2148 ページの『DestinationTransferStartExit.java インターフェース』](#)

[2147 ページの『DestinationTransferEndExit.java インターフェース』](#)

[2167 ページの『MonitorExit.java インターフェース』](#)

[2168 ページの『ProtocolBridgeCredentialExit.java インターフェース』](#)

MFT エージェントのコマンド・キューに PUT できるメッセージ形式

これらの XML スキーマは、エージェントによる操作の実行を要求するために、エージェントのコマンド・キューに PUT できるメッセージの形式を定義します。XML メッセージは、コマンド行のコマンドまたはアプリケーションを使用して、エージェントのコマンド・キューに置くことができます。

- [ファイル転送要求メッセージ・フォーマット](#)
- [MFT モニター要求メッセージ・フォーマット](#)
- [Ping MFT エージェント要求メッセージ・フォーマット](#)
- [MFT エージェント応答メッセージ・フォーマット](#)

メッセージング REST API に関する参照情報

[messaging REST API に関する参照情報](#)。

messaging REST API の使用方法について詳しくは、[REST API を使用したメッセージング](#)を参照してください。

REST API リソース

このトピック集では、それぞれの messaging REST API リソースについての参照情報を提供します。

messaging REST API の使用方法について詳しくは、[REST API を使用したメッセージング](#)を参照してください。

/messaging/qmgr/{qmgrName}/queue/{queueName}/message

この /messaging/qmgr/{qmgrName}/queue/{queueName}/message リソースを使用すると、メッセージをキューに追加したり、キューからメッセージを参照したり、キューからメッセージを破壊的に取得したり messaging REST API できます。

注：[メッセージ分割機能](#)はではサポートされて IBM MQmessaging REST API いません。

POST /messaging/qmgr/{qmgrName}/queue/{queueName}/message

/messaging/qmgr/{qmgrName}/queue/{queueName}/message リソースを指定した HTTP POST メソッドを使用すると、指定したキュー・マネージャー上の指定したキューにメッセージを書き込むことができます。

HTTP 要求本体を格納する IBM MQ メッセージを指定されたキュー・マネージャーおよびキューに入れます。キュー・マネージャーは mqweb サーバーと同じマシン上になければなりません。このメソッドは、テキスト・ベースの HTTP 要求本体だけをサポートします。メッセージはフォーマットされたメッセージ MQSTR または JMS TextMessage として送信され、現在のユーザーコンテキストを使用して配置されます。

REST API V3 では、ユーザー定義のメッセージ・プロパティを指定し、メッセージ優先度を含める機能を追加します。ibm-mq-md 優先順位および ibm-mq-usr 要求ヘッダーは、REST API V3 でのみ使用可能

です。ibm-mq-md-correlationId 要求ヘッダーは、REST API V3 では異なるフォーマットになっています。ヘッダーは、アプリケーション固有の ID にすることも、エンコードされたストリングの場合は ID:接頭部を必要とすることもできます。POST 要求にユーザー定義メッセージまたはアプリケーション固有の相関 ID が含まれている場合、メッセージは JMS TextMessage としてフォーマット設定されます。

- [2177 ページの『リソース URL』](#)
- [2178 ページの『要求ヘッダー』](#)
- [2180 ページの『要求本体の形式』](#)
- [2180 ページの『セキュリティ要件』](#)
- [2180 ページの『応答状況コード』](#)
- [2181 ページの『応答ヘッダー』](#)
- [2182 ページの『応答本体の形式』](#)
- [2182 ページの『例』](#)

リソース URL

`https://host:port/ibmmq/rest/v1/messaging/qmgr/{qmgrName}/queue/{queueName}/message`

`https://host:port/ibmmq/rest/v2/messaging/qmgr/{qmgrName}/queue/{queueName}/message`

`https://host:port/ibmmq/rest/v3/messaging/qmgr/{qmgrName}/queue/{queueName}/message`

qmgrName

メッセージングのための接続先のキュー・マネージャーの名前を指定します。キュー・マネージャーは mqweb サーバーと同じマシン上になければなりません。

から IBM MQ 9.3.3、ローカルのキューマネージャまたはリモートのキューマネージャのいずれかに接続できます。指定する名前は、mqweb サーバーの設定 **qmgrName** 方法によって異なります：

- mqweb サーバーがローカルのキューマネージャのみに接続するよう設定されている場合は、キューマネージャの名前を使用してください。
- mqweb サーバーがリモートキューマネージャへの接続用に構成されている場合、リモートキューマネージャ接続情報で指定されているキューマネージャの一意の名前を使用してください。

mqweb サーバーがローカルキューマネージャに接続するように設定されているか、リモートキューマネージャに接続するように設定されているかは、コマンド **dspmweb properties** を使用して **mqRestMessagingConnectionMode** プロパティを確認することで判断できます。

キュー・マネージャーの名前には、大/小文字の区別があります。

キュー・マネージャー名にスラッシュ、ピリオド、または % 記号が含まれている場合は、その文字を URL エンコードする必要があります。

- スラッシュは、%2F としてエンコードする必要があります。
- ピリオドは、%2E としてエンコードする必要があります。
- パーセント記号は %25 としてエンコードする必要があります。

queueName

メッセージを入れるキューの名前を指定します。

キューは、指定されたキュー・マネージャーに対してローカル、リモート、または別名のいずれかで定義する必要があります。クラスター・キューを参照することもできます。

キューの名前には、大/小文字の区別があります。

キュー名にスラッシュまたは % 記号が含まれている場合は、その文字を URL エンコードする必要があります。

- スラッシュ / は、%2F としてエンコードする必要があります。

- % 記号は、%25 としてエンコードする必要があります。

HTTP 接続を使用可能にすれば、HTTPS ではなく HTTP を使用できます。HTTP の使用可能化について詳しくは、[HTTP および HTTPS ポートの構成](#)を参照してください。

要求ヘッダー

要求で以下のヘッダーを送信する必要があります。

認証

基本認証を使用している場合、このヘッダーを送信する必要があります。詳しくは、[REST API での HTTP 基本認証の使用](#)を参照してください。

Content-Type

このヘッダーは、以下のいずれかの値で送信する必要があります。

- text/plain;charset=utf-8
- text/html;charset=utf-8
- text/xml;charset=utf-8
- application/json;charset=utf-8
- application/xml;charset=utf-8

ibm-mq-rest-csrf-token

このヘッダーを設定する必要がありますが、その値はブランクを含む任意のものにすることができます。

要求で以下のヘッダーをオプションで送信できます。

Accept-Language

このヘッダーは、応答メッセージ本体で返される例外メッセージやエラー・メッセージに必要な言語を指定します。

REST API V1 REST API V2 ibm-mq-md-correlationId

このヘッダーは、作成されるメッセージの相関 ID を設定します。ヘッダーは 48 文字の 16 進数エンコード文字列として指定され、24 バイトを表す。値の前に "ID:" を付けしないでください。REST API はそのストリングを自動的に追加します。

例: ibm-mq-md-correlationId: 414d5120514d4144455620202020202067d8bf5923582e02

REST API V3 ibm-mq-md-correlationId

このヘッダーは、作成されるメッセージの相関 ID を設定します。相関 ID は、以下のいずれかの形式をとります。

- ストリング "ID:" の接頭部が付いた、24 バイトを表す 48 文字の 16 進エンコード・ストリング。例:
ibm-mq-md-correlationId:
ID:414d5120514d4144455620202020202067d8bf5923582e02
- アプリケーション固有の値。この値は、アプリケーション固有のストリングです。ibm-mq-md-correlationId: *My-Custom-CorrelId*

この形式の相関 ID を指定すると、メッセージ宛先は WMQ_CLIENT_JMS_COMPLIANT としてターゲットにされるため、MQRFH2 ヘッダーが組み込まれます。

ibm-mq-md-expiry

このヘッダーは、作成されるメッセージの有効期限を設定します。メッセージの有効期限は、メッセージがキューに到着した時点から始まります。このため、ネットワーク待ち時間は無視されます。ヘッダーは、以下のいずれかの値として指定する必要があります。

unlimited

メッセージは満了しません。

この値がデフォルト値です。

整数値

メッセージが期限切れになるまでのミリ秒数。

0 から 999999999900 の範囲に制限されます。

ibm-mq-md-persistence

このヘッダーは、作成されるメッセージの持続性を設定します。ヘッダーは次のいずれかの値として指定されます：

nonPersistent

システム障害後またはキュー・マネージャー再始動後に、メッセージは存続しません。

この値がデフォルト値です。

persistent

メッセージはシステムの障害後、またはキュー・マネージャーの再始動後も存続します。

REST API V3 ibm-mq-md-priority

REST API V3 において、このヘッダーは作成されたメッセージの優先度を設定します。ヘッダーは次のいずれかの値として指定されます：

asDestination

このメッセージは、基礎となる IBM MQ キュー・オブジェクトの DEFPRTY 属性に指定されている優先順位を使用します。

この値がデフォルト値です。

整数値

実際の優先順位を 0 から 9 の範囲の整数として指定します。

例: `ibm-mq-md-priority: asDestination`

REST API V1 および REST API V2 において、POST のメッセージ優先度は常に 4 です。

ibm-mq-md-replyTo

このヘッダーは、作成されるメッセージの返信先宛先を設定します。ヘッダーの形式は、応答先キューとオプションのキュー・マネージャーを提供する標準表記 (`replyQueue[@replyQmgr]`) を使用します。

例: `ibm-mq-md-replyTo: myReplyQueue@myReplyQMGr`

REST API V3 ibm-mq-usr

このヘッダーはリクエストメッセージのユーザー定義プロパティを設定します。

プロパティの構文は次のとおりです。

`ibm-mq-usr: property_name; user_value; user_type`

property_name

指定されているユーザー・プロパティの名前。これは有効な JMS プロパティ名でなければなりません。

user_value

プロパティの値。

user_type

プロパティのタイプは以下のとおりです。

- `boolean` (true/false、MQBOOL)
- `byte` (8 ビット整数、MQINT8)
- `short` (16 ビット整数、MQINT16)
- `integer` (32 ビット整数、MQINT32)
- `long` (64 ビット整数、MQINT64)
- `float` (32 ビット実数、MQFLOAT32)
- `double` (64 ビット実数、MQFLOAT64)
- `string` (引用符付きストリング)

メッセージに対して複数のプロパティを設定できます。単一の `ibm-mq-usr` リクエストヘッダー内で複数のプロパティをカンマ区切りで指定するか、または複数の独立した `ibm-mq-usr` リクエストヘッダーを使用することができます。

たとえば、単一のヘッダーに複数のユーザー定義プロパティを設定できます: `ibm-mq-usr: userPropertyA;5;byte,userPropertyB;-10;integer`

あるいは、ヘッダーの別々のインスタンスに複数のユーザー定義プロパティを設定することもできます: `ibm-mq-usr: userPropertyA;5;byte ibm-mq-usr: userPropertyB;-10;integer`

要求本体の形式

要求本体は、テキスト形式で UTF-8 エンコードを使用する必要があります。特定のテキスト構造にする必要はありません。要求本文テキストを含む MQSTR 形式のメッセージが作成され、指定されたキューに書き込まれます。

REST API V3 REST API V3 のユーザー定義プロパティ、またはアプリケーション固有の相関 ID 機能が使用される場合、リクエスト本文テキストを含むフォーマット済み JMS `TextMessage` メッセージが生成され、指定されたキューに格納されます。

詳しくは、[例](#)を参照してください。

セキュリティ要件

呼び出し元は mqweb サーバーで認証する必要があります。MQWebAdmin 役割および MQWebAdminRO 役割は、messaging REST API には適用されません。REST API のセキュリティについて詳しくは、[IBM MQ Console](#) および [REST API のセキュリティ](#)を参照してください。

ユーザーは、mqweb サーバーで認証された後に、messaging REST API および administrative REST API の両方を使用できるようになります。

呼び出し元のセキュリティ・プリンシパルには、指定されたキューにメッセージを入れるための権限が付与されていなければなりません。

- リソース URL の `{queueName}` 部分で指定されるキューは、PUT に対応している必要があります。
-  **MQ Appliance** リソース URL の `{queueName}` 部分で指定されるキューについては、呼び出し元のセキュリティ・プリンシパルに +PUT 権限が付与されている必要があります。
-  **z/OS** リソース URL の `{queueName}` 部分で指定されるキューの場合、呼び出し元のセキュリティ・プリンシパルに UPDATE アクセス権限が付与されている必要があります。

このセキュリティプリンシパルは、mqweb サーバーを起動したユーザー、または mqweb サーバーにログインしているユーザーである可能性があります。接続先のキューマネージャがリモートキューマネージャの場合、セキュリティプリンシパルは代わりに、リモートキューマネージャ接続情報で指定された認証情報によって提供されるユーザー、またはチャンネルセキュリティルールによって決定される別のユーザーとなる可能性があります。詳細については、[「.NET Framework アプリケーションで使用されるセキュリティ主体の messaging REST API 決定」](#)を参照してください。

 **ALW** では AIX, Linux, and Windows、コマンド `setmqaut` を使用して、セキュリティプリンシパルにリソース IBM MQ を使用する権限を付与できます。詳しくは、[setmqaut \(権限の付与または取り消し\)](#)を参照してください。

z/OS では、[z/OS でのセキュリティのセットアップ](#)を参照してください。

messaging REST API で Advanced Message Security (AMS) を使用する場合は、メッセージを投稿するユーザーのコンテキストではなく、mqweb サーバーのコンテキストを使用してすべてのメッセージが暗号化されることに注意してください。

応答状況コード

201

メッセージは正常に作成されて送信されました。

400

無効なデータが指定されました。

例えば、無効な要求ヘッダーの値が指定されました。

401

認証されませんでした。

呼び出し元は mqweb サーバーに対して認証されている必要があり、1 つ以上の MQWebAdmin、MQWebAdminRO、または MQWebUser ロールのメンバーでなければなりません。ibm-mq-rest-csrf-token ヘッダーも指定する必要があります。詳しくは、[2180 ページの『セキュリティ要件』](#)を参照してください。

403

許可がありません。

呼び出し元は mqweb サーバーで認証を受け、有効なプリンシパルと関連付けられました。しかし、プリンシパルが、必要な IBM MQ リソースのすべてまたはサブセットに対するアクセス権限を持っていないか、または MQWebUser 役割に含まれていません。必要なアクセス権について詳しくは、[2180 ページの『セキュリティ要件』](#)を参照してください。

404

キューが存在しません。

405

キューでは PUT が禁止されています。

415

メッセージ・ヘッダーまたはメッセージ本体のメディア・タイプがサポートされていません。

例えば、Content-Type ヘッダーが、サポートされないメディア・タイプに設定されています。

500

サーバーの問題または IBM MQ からのエラー・コード。

502

メッセージング・プロバイダーが必要な関数をサポートしていないので、現在のセキュリティ・プリンシパルはメッセージを送信できません。例えば、mqweb サーバーのクラスパスが無効です。

503

キュー・マネージャーが実行されていません。

応答ヘッダー

応答では以下のヘッダーが返されます。

Content-Language

エラーや例外が発生した場合の応答メッセージの言語 ID を指定します。エラーまたは例外条件に必要な言語を示すために、Accept-Language 要求ヘッダーと一緒に使用されます。要求された言語がサポートされていない場合は、mqweb サーバーのデフォルトが使用されます。

Content-Length

内容が存在しない場合にも適用される、HTTP 応答本体の長さを指定します。正常に実行された場合、値はゼロになります。

Content-Type

応答本体のタイプを指定します。成功すると、値は text/plain;charset=utf-8 になります。エラーまたは例外が発生した場合、値は application/json;charset=utf-8 になります。

ibm-mq-md-messageId

IBM MQ によってこのメッセージに割り振られるメッセージ ID を指定します。ibm-mq-md-correlationId 要求ヘッダーと同様に、24 バイトを表す 48 文字の 16 進エンコード・ストリングとして表されます。

以下に例を示します。

```
ibm-mq-md-messageId: 414d5120514d4144455620202020202067d8ce5923582f07
```

REST API V3 **ibm-mq-md-messageId**

IBM MQ によってこのメッセージに割り振られるメッセージ ID を指定します。 **ibm-mq-md-correlationId** 要求ヘッダーと同様に、これは、24 バイトを表す 48 文字の 16 進エンコード・ストリングとして表され、接頭部にストリング ID: が付きます。

以下に例を示します。

```
ibm-mq-md-messageId: ID:414d5120514d4144455620202020202067d8ce5923582f07
```

ibm-mq-解決済み-qmgr

リクエストに使用されたキューマネージャの名前を指定します。 リソース URL で一意の名前が使用された場合、このヘッダーは、その一意の名前に関連付けられたキューマネージャの名前を識別します。 リソース URL で使用される一意の名前がキューマネージャグループを指す場合、このヘッダーはグループ内で使用されたキューマネージャを識別します。

応答本体の形式

メッセージが正常に送信された場合、応答本体は空になります。 エラーが発生した場合、応答本体にエラー・メッセージが入ります。 詳しくは、[REST API エラー処理](#)を参照してください。

例

以下の例では、v2 のリソース URL を使用しています。 IBM MQ の IBM MQ 9.1.5 よりも前のバージョンを使用している場合、v1 リソースを代わりに使用する必要があります。 つまり、リソース URL では、URL の例で v2 が使用されている v1 が置き換わります。

以下の例では、パスワード **mquser** を使用して **mquser** というユーザーにログインします。 cURL では、ログイン要求は、次の Windows の例のようになります。 LTPA トークンは、**-c** フラグを使用して **cookiejar.txt** ファイルに保管されます。

```
curl -k "https://localhost:9443/ibmmq/rest/v2/login" -X POST
-H "Content-Type: application/json" --data "{\"username\":\"mquser\", \"password\":\"mquser\"}"
-c c:\cookiejar.txt
```

ユーザーがログインすると、さらに要求を認証するために LTPA トークンと **ibm-mq-rest-csrf-token** HTTP ヘッダーが使用されます。 **ibm-mq-rest-csrf-token** は、ブランクを含む任意の値にすることができます。

- 以下の Windows cURL の例では、デフォルト・オプションを使用して、キュー・マネージャー QM1 上のキュー Q1 にメッセージを送信します。 メッセージには、テキスト **"Hello World!"** が入ります。

```
curl -k "https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/queue/Q1/message"
-X POST -b c:\cookiejar.txt -H "ibm-mq-rest-csrf-token: token_value"
-H "Content-Type: text/plain; charset=utf-8" --data "Hello World!"
```

- 以下の Windows cURL の例では、有効期限 2 分を指定して、永続メッセージをキュー・マネージャー QM1 のキュー Q1 に送信します。 メッセージには、テキスト **"Hello World!"** が入ります。

```
curl -k "https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/queue/Q1/message"
-X POST -b c:\cookiejar.txt -H "ibm-mq-rest-csrf-token: token_value"
-H "Content-Type: text/plain; charset=utf-8" -H "ibm-mq-md-persistence: persistent"
-H "ibm-mq-md-expiry: 120000" --data "Hello World!"
```

- 以下の Windows cURL の例では、有効期限および相関 ID を定義せずに、非持続メッセージをキュー・マネージャー QM1 上のキュー Q1 に送信します。 メッセージには、テキスト **"Hello World!"** が入ります。

```
curl -k "https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/queue/Q1/message"
-X POST -b c:\cookiejar.txt -H "ibm-mq-rest-csrf-token: token_value"
-H "Content-Type: text/plain; charset=utf-8" -H "ibm-mq-md-persistence: nonPersistent"
-H "ibm-mq-md-expiry: unlimited" -H "ibm-mq-md-correlationId:
414d5120514d41444556202020202067d8b
f5923582e02" --data "Hello World!"
```

GET /messaging/qmgr/{qmgrName}/queue/{queueName}/message

/messaging/qmgr/{qmgrName}/queue/{queueName}/message リソースを指定した HTTP GET メソッドを使用して、関連付けられたキュー・マネージャーおよびキューからメッセージを参照できます。

指定されたキュー・マネージャーおよびキューから最初に使用可能なメッセージを参照します。キュー・マネージャーは mqweb サーバーと同じマシン上になければなりません。メッセージ本文は HTTP 応答本体に返されます。メッセージは または MQSTR の形式でなければ JMS TextMessage ならず、現在のユーザーコンテキストを使用して受信されます。

すべてのメッセージはキューに残されて、不適切なメッセージについては、メッセージ呼び出し元に該当する状況コードが返されます。例えば、MQSTR 形式または JMS TextMessage 形式ではないメッセージなどです。

REST API V3 は、ユーザー定義のメッセージ・プロパティを指定する機能を追加し、メッセージとともにメッセージ優先順位を組み込みます。ibm-mq-md-priority および ibm-mq-usr 応答ヘッダーは、REST API V3 でのみ使用可能です。ibm-mq-md-correlationId 要求ヘッダーは、REST API V3 では異なるフォーマットになっています。ヘッダーは、アプリケーション固有の ID にすることも、エンコードされたストリングの場合は ID:接頭部を保持することもできます。REST API V3 では ibm-mq-md-messageId 応答ヘッダーと照会パラメーターの形式が異なり、ID:接頭部が保持されます。

- [2183 ページの『リソース URL』](#)
- [2184 ページの『オプションの照会パラメーター』](#)
- [2185 ページの『要求ヘッダー』](#)
- [2185 ページの『要求本体の形式』](#)
- [2185 ページの『セキュリティ要件』](#)
- [2186 ページの『応答状況コード』](#)
- [2186 ページの『応答ヘッダー』](#)
- [2188 ページの『応答本体の形式』](#)
- [2189 ページの『例』](#)

リソース URL

`https://host:port/ibmmq/rest/v1/messaging/qmgr/{qmgrName}/queue/{queueName}/message`

`https://host:port/ibmmq/rest/v2/messaging/qmgr/{qmgrName}/queue/{queueName}/message`

`https://host:port/ibmmq/rest/v3/messaging/qmgr/{qmgrName}/queue/{queueName}/message`

qmgrName

メッセージングのための接続先のキュー・マネージャーの名前を指定します。キュー・マネージャーは mqweb サーバーと同じマシン上になければなりません。

から IBM MQ 9.3.3、ローカルのキューマネージャまたはリモートのキューマネージャのいずれかに接続できます。指定する名前は、mqweb サーバーの設定 **qmgrName** 方法によって異なります：

- mqweb サーバーがローカルのキューマネージャのみに接続するよう設定されている場合は、キューマネージャの名前を使用してください。
- mqweb サーバーがリモートキューマネージャへの接続用に構成されている場合、リモートキューマネージャ接続情報で指定されているキューマネージャの一意の名前を使用してください。

mqweb サーバーがローカルキューマネージャに接続するように設定されているか、リモートキューマネージャに接続するように設定されているかは、コマンド **dspmweb properties** を使用して **mqRestMessagingConnectionMode** プロパティを確認することで判断できます。

キュー・マネージャーの名前には、大/小文字の区別があります。

キュー・マネージャー名にスラッシュ、ピリオド、または % 記号が含まれている場合は、その文字を URL エンコードする必要があります。

- スラッシュ (/) は、%2F としてエンコードする必要があります。
- パーセント記号 (%) は、%25 とエンコードする必要があります。

queueName

メッセージを参照するキューの名前を指定します。

キューは、ローカルであるか、またはローカル・キューを指し示す別名として定義することが必要です。

キューの名前には、大/小文字の区別があります。

キュー名にスラッシュまたは % 記号が含まれている場合は、その文字を URL エンコードする必要があります。

- スラッシュ / は、%2F としてエンコードする必要があります。
- % 記号は、%25 とエンコードする必要があります。

HTTP 接続を使用可能にすれば、HTTPS ではなく HTTP を使用できます。HTTP の使用可能化について詳しくは、[HTTP および HTTPS ポートの構成](#)を参照してください。

オプションの照会パラメーター

REST API V1 → REST API V2 **correlationId=hexValue**

HTTP メソッドが次のメッセージに対応する相関 ID と共に返すように指定します。

hexValue

照会パラメーターは、24 バイトを表す、48 文字の 16 進エンコード・ストリングとして指定する必要があります。

以下に例を示します。

```
../message?correlationId=414d5120514d4144455620202020202067d8bf5923582e02
```

REST API V3 **correlationId=ID:hexValue** または **correlationId=application_specific_value**

HTTP メソッドが次のメッセージに対応する相関 ID と共に返すように指定します。

hexValue

照会パラメーターは、24 バイトを表す 48 文字の 16 進エンコード・ストリングとして指定する必要があり、その前にストリング "ID:" を付ける必要があります。

以下に例を示します。

```
../message?correlationId=ID:414d5120514d4144455620202020202067d8bf5923582e02
```

application_specific_value

照会パラメーターは、アプリケーション固有のストリングとして指定できます。

以下に例を示します。

```
../message?correlationId=My-Custom-CorrelId
```

REST API V1 → REST API V2 **messageId=hexValue**

HTTP メソッドが次のメッセージに対応するメッセージ ID と共に返すように指定します。

hexValue

照会パラメーターは、24 バイトを表す、48 文字の 16 進エンコード・ストリングとして指定する必要があります。

以下に例を示します。

```
../message?messageId=414d5120514d4144455620202020202067d8ce5923582f07
```

REST API V3 `messageId=ID:hexValue`

HTTP メソッドが次のメッセージに対応するメッセージ ID と共に返すように指定します。

hexValue

照会パラメーターは、24 バイトを表す 48 文字の 16 進エンコード・ストリングとして指定する必要があります、その前にストリング "ID:" を付ける必要があります。

以下に例を示します。

```
../message?messageId=ID:414d5120514d4144455620202020202067d8ce5923582f07
```

要求ヘッダー

要求で以下のヘッダーを送信する必要があります。

認証

基本認証を使用している場合、このヘッダーを送信する必要があります。詳しくは、[REST API での HTTP 基本認証の使用](#) を参照してください。

ibm-mq-rest-csrf-token

このヘッダーを設定する必要がありますが、その値はブランクを含む任意のものにすることができます。

要求で以下のヘッダーをオプションで送信できます。

Accept-Charset

このヘッダーを使用して、応答のために許容される文字セットを指定できます。指定する場合、このヘッダーは UTF-8 として設定する必要があります。

Accept-Language

このヘッダーは、応答メッセージ本体で返される例外メッセージやエラー・メッセージに必要な言語を指定します。

要求本体の形式

なし。

セキュリティ要件

呼び出し元は mqweb サーバーで認証する必要があります。MQWebAdmin 役割および MQWebAdminRO 役割は、messaging REST API には適用されません。REST API のセキュリティについて詳しくは、[IBM MQ Console および REST API のセキュリティ](#) を参照してください。

ユーザーは、mqweb サーバーで認証された後に、messaging REST API および administrative REST API の両方を使用できるようになります。

呼び出し元のセキュリティ・プリンシパルには、指定されたキューからメッセージを参照するための権限が付与されていなければなりません。

- リソース URL の `{queueName}` 部分で指定されるキューは、BROWSE に対応している必要があります。
-   リソース URL の `{queueName}` 部分で指定されたキューについて、+GET、+INQ、および+BROWSE 権限が呼び出し元のセキュリティ・プリンシパルに付与されている必要があります。
-  リソース URL の `{queueName}` 部分で指定されるキュー UPDATE では、アクセス権限が呼び出し元のセキュリティ・プリンシパルに付与されている必要があります。

このセキュリティプリンシパルは、mqweb サーバーを起動したユーザー、または mqweb サーバーにログインしているユーザーである可能性があります。接続先のキューマネージャがリモートキューマネージャの場合、セキュリティプリンシパルは代わりに、リモートキューマネージャ接続情報で指定された認証情報によって提供されるユーザー、またはチャネルセキュリティルールによって決定される別のユーザーと

なる可能性があります。詳細については、[「.NET Framework アプリケーションで使用されるセキュリティ主体の messaging REST API 決定」](#)を参照してください。

ALW では AIX, Linux, and Windows、コマンド **setmqaut** を使用して、セキュリティプリンシパルにリソース IBM MQ を使用する権限を付与できます。詳しくは、[setmqaut \(権限の付与または取り消し\)](#)を参照してください。

z/OS では、[z/OS でのセキュリティのセットアップ](#)を参照してください。.

応答状況コード

200

メッセージは正常に受信されました。

204

メッセージが使用できません。

400

無効なデータが指定されました。

例えば、無効な照会パラメーターの値が指定されました。

401

認証されませんでした。

呼び出し元は mqweb サーバーに対して認証されている必要があり、1 つ以上の MQWebAdmin、MQWebAdminRO、または MQWebUser ロールのメンバーでなければなりません。ibm-mq-rest-csrf-token ヘッダーも指定する必要があります。詳しくは、[2185 ページの『セキュリティ要件』](#)を参照してください。

403

許可がありません。

呼び出し元は mqweb サーバーで認証を受け、有効なプリンシパルと関連付けられました。しかし、プリンシパルが、必要な IBM MQ リソースのすべてまたはサブセットに対するアクセス権限を持っていないか、または MQWebUser 役割に含まれていません。必要なアクセス権について詳しくは、[2185 ページの『セキュリティ要件』](#)を参照してください。

404

キューが存在しません。

500

サーバーの問題または IBM MQ からのエラー・コード。

501

HTTP 応答を作成できませんでした。

例えば、受信したメッセージのタイプが正しくないか、タイプは正しくても本体を処理できない場合があります。

502

メッセージング・プロバイダーが必要な関数をサポートしていないので、現在のセキュリティ・プリンシパルはメッセージを受信できません。例えば、mqweb サーバーのクラスパスが無効です。

503

キュー・マネージャーが実行されていません。

応答ヘッダー

応答では以下のヘッダーが返されます。

Content-Language

エラーや例外が発生した場合の応答メッセージの言語 ID を指定します。エラーまたは例外条件に必要な言語を示すために、Accept-Language 要求ヘッダーと一緒に使用されます。要求された言語がサポートされていない場合は、mqweb サーバーのデフォルトが使用されます。

Content-Length

内容が存在しない場合にも適用される、HTTP 応答本体の長さを指定します。値には、メッセージ・データの長さ (バイト数) が含まれます。

Content-Type

受信したメッセージの応答本体で返されるコンテンツのタイプを指定します。成功すると、値は `text/plain; charset=utf-8` になります。エラーまたは例外が発生した場合、値は `application/json; charset=utf-8` になります。

REST API V1 ▶ REST API V2 **ibm-mq-md-correlationId**

受信メッセージの相関 ID を指定します。受信したメッセージに有効な相関 ID が含まれている場合、このヘッダーが返されます。これは 24 バイトを表す、48 文字の 16 進エンコード・ストリングとして表記されます。

以下に例を示します。

```
ibm-mq-md-correlationId: 414d5120514d4144455620202020202067d8bf5923582e02
```

REST API V3 **ibm-mq-md-correlationId**

受信メッセージの相関 ID を指定します。受信したメッセージに有効な相関 ID が含まれている場合、このヘッダーが返されます。相関 ID は、以下のいずれかの形式をとります。

- ストリング "ID:" の接頭部が付いた、24 バイトを表す 48 文字の 16 進エンコード・ストリング。以下に例を示します。

```
ibm-mq-md-correlationId: ID:414d5120514d4144455620202020202067d8bf5923582e02
```

- アプリケーション固有の値。この値は、アプリケーション固有のストリングです。

```
ibm-mq-md-correlationId: My-Custom-CorrelId
```

ibm-mq-md-expiry

受信メッセージの有効期限までの残り時間を示します。ヘッダーは、次の値のうちのいずれかです。

unlimited

メッセージは満了しません。

整数値

メッセージの有効期限までの残り時間 (ミリ秒単位)。

REST API V1 ▶ REST API V2 **ibm-mq-md-messageId**

IBM MQ によってこのメッセージに割り振られるメッセージ ID を指定します。ibm-mq-md-correlationId ヘッダーと同様に、24 バイトを表す 48 文字の 16 進エンコード・ストリングとして表されます。

以下に例を示します。

```
ibm-mq-md-messageId: 414d5120514d4144455620202020202067d8ce5923582f07
```

REST API V3 **ibm-mq-md-messageId**

IBM MQ によってこのメッセージに割り振られるメッセージ ID を指定します。ibm-mq-md-correlationId ヘッダーと同様に、これは 24 バイトを表す 48 文字の 16 進エンコード・ストリングとして表され、接頭部にストリング "ID:" が付きます。

以下に例を示します。

```
ibm-mq-md-messageId: ID:414d5120514d4144455620202020202067d8ce5923582f07
```

ibm-mq-md-persistence

受信メッセージの持続性を指定します。ヘッダーは、次の値のうちのいずれかです。

nonPersistent

システム障害後またはキュー・マネージャー再始動後に、メッセージは存続しません。

persistent

メッセージはシステムの障害後、またはキュー・マネージャーの再始動後も存続します。

REST API V3 **ibm-mq-md-priority**

メッセージ優先順位の設定を戻します。以下に例を示します。

```
ibm-mq-md-priority: 3
```

ibm-mq-md-replyTo

受信メッセージの返信先宛先を指定します。ヘッダーの形式は、応答先キューおよびキュー・マネージャー `replyQueue@replyQmgr` の標準表記を使用します。

以下に例を示します。

```
ibm-mq-md-replyTo: myReplyQueue@myReplyQMgr
```

ibm-mq-解決済み-qmgr

リクエストに使用されたキューマネージャの名前を指定します。リソース URL で一意の名前が使用された場合、このヘッダーは、その一意の名前に関連付けられたキューマネージャの名前を識別します。リソース URL で使用される一意の名前がキューマネージャグループを指す場合、このヘッダーはグループ内で使用されたキューマネージャを識別します。

REST API V3 **ibm-mq-usr**

メッセージ・ユーザー定義プロパティを戻します。メッセージに複数のプロパティを設定できます。この場合、`ibm-mq-usr` 応答ヘッダーの2つ以上の個別のインスタンスが存在します。

以下に例を示します。

```
ibm-mq-usr: myIPProp;5;short
ibm-mq-usr: mySProp;"hi";string
ibm-mq-usr: myBProp>true;boolean
```

プロパティの構文は次のとおりです。

```
ibm-mq-usr: property_name; user_value; user_type
```

property_name

指定されているユーザー・プロパティの名前。これは有効な JMS プロパティ名でなければなりません。

user_value

プロパティの値。

user_type

プロパティのタイプは以下のとおりです。

- `boolean` (`true/false`、MQBOOL)
- `byte` (8 ビット整数、MQINT8)
- `short` (16 ビット整数、MQINT16)
- `integer` (32 ビット整数、MQINT32)
- `long` (64 ビット整数、MQINT64)
- `float` (32 ビット実数、MQFLOAT32)
- `double` (64 ビット実数、MQFLOAT64)
- `string` (引用符付きストリング)

応答本体の形式

常に実行された場合、応答本体には受信メッセージからのメッセージ本体が入ります。エラーが発生した場合、応答本体に JSON 形式のエラー・メッセージが入ります。どちらの応答も UTF-8 エンコードされます。詳しくは、[REST API エラー処理](#)を参照してください。

メッセージを受信する際は、または JMS TextMessage 形式のメッセージ IBM MQMQSTR のみがサポートされていることに注意してください。

取得禁止とマークされているキューを参照しても、内容は返されません。

参照されているキューにメッセージ ID が重複したメッセージが含まれている場合、メッセージ ID でフィルタリングすると、最初のメッセージが返されます。

例

以下の例では、v2 のリソース URL を使用しています。IBM MQ の IBM MQ 9.1.5 よりも前のバージョンを使用している場合、v1 リソースを代わりに使用する必要があります。つまり、リソース URL では、URL の例で v2 が使用されている v1 が置き換わります。

以下の例では、パスワード `mquser` を使用して `mquser` というユーザーにログインします。cURL では、ログイン要求は、次の Windows の例のようになります。LTPA トークンは、`-c` フラグを使用して `cookiejar.txt` ファイルに保管されます。

```
curl -k "https://localhost:9443/ibmmq/rest/v2/login" -X POST
-H "Content-Type: application/json" --data '{"username": "mquser", "password": "mquser"}'
-c c:\cookiejar.txt
```

ユーザーがログインすると、さらに要求を認証するために LTPA トークンと `ibm-mq-rest-csrf-token` HTTP ヘッダーが使用されます。`ibm-mq-rest-csrf-token` は、空白を含む任意の値にすることができます。

- 以下の Windows cURL の例は、デフォルトのオプションを使用して、次に使用可能なメッセージをキュー・マネージャー 0M1 上のキュー 01 から参照します。

```
curl -k "https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/queue/Q1/message"  
-X GET -b c:\cookiejar.txt -H "ibm-mq-rest-csrf-token: token-value"  
-H "Accept: text/plain"
```

- **REST API V1**、**REST API V2**以下の Windows cURL の例は、特定の相関 ID (00000000000000000000000000000000000000abcdabcd) を持つメッセージを、キュー・マネージャー QM1 上のキュー Q1 から参照します。

```
curl -k "https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/queue/Q1/message?correlationId=000000000000000000000000000000000000abcdabcd"  
-X GET -b c:\cookiejar.txt -H "ibm-mq-rest-csrf-token: token-value"  
-H "Accept: text/plain"
```

- RESTAPI V3

以下の Windows cURL の例は、前の例と同じですが、REST API V3 を使用しています。この例では、キューマネージャー QM10000000000000000000000000000000000000000abcdabcd: の Q1 キューから、対応する相関 ID を持つメッセージのみをリストします：

```
curl -k "https://localhost:9443/ibmmq/rest/v3/messaging/qmgr/QM1/queue/Q1/message?correlationId=ID:0000000000000000000000000000000000000000abcdacbcd"  
-X GET -b c:\cookiejar.txt -H "ibm-mq-rest-csrf-token: token-value"  
-H "Accept: text/plain"
```

`/messaging/qmqr/{qmqrName}/queue/{queueName}/message` の削除

HTTP/messaging/qmgr/{qmgrName}/queue/{queueName}/message の DELETE メソッドをリソースとともに使用すると、関連付けられたキューマネージャおよびキューからメッセージを破壊的に取得できます。

使用可能な次のメッセージを指定されたキュー・マネージャーおよびキューから破壊的に取得し、HTTP 応答本体でメッセージ本体を返します。キュー・マネージャーは mqweb サーバーと同じマシン上になければなりません。メッセージは MQSTR または JMS TextMessage の形式でなければならず、現在のユーザー・コンテキストを使用して受信されます。

非互換のメッセージはキューに残されて、呼び出し元には該当する状況コードが返されます。例えば、MQSTR 形式または JMS TextMessage 形式ではないメッセージなどです。

REST API V3 は、ユーザー定義のメッセージ・プロパティを指定する機能を追加し、メッセージとともにメッセージ優先順位を組み込みます。ibm-mq-md-priority および ibm-mq-usr 応答ヘッダーは、REST API V3 でのみ使用可能です。ibm-mq-md-correlationId 要求ヘッダーは、REST API V3 では異なるフォーマットになっています。ヘッダーは、アプリケーション固有の ID にすることも、エンコードされたストリングの場合は ID:接頭部を保持することもできます。REST API V3 では ibm-mq-md-messageId 応答ヘッダーと照会パラメーターの形式が異なり、ID:接頭部が保持されます。

- [2190 ページの『リソース URL』](#)
- [2191 ページの『オプションの照会パラメーター』](#)
- [2192 ページの『要求ヘッダー』](#)
- [2192 ページの『要求本体の形式』](#)
- [2192 ページの『セキュリティ要件』](#)
- [2193 ページの『応答状況コード』](#)
- [2193 ページの『応答ヘッダー』](#)
- [2195 ページの『応答本体の形式』](#)
- [2196 ページの『例』](#)

リソース URL

`https://host:port/ibmmq/rest/v1/messaging/qmgr/{qmgrName}/queue/{queueName}/message`

`https://host:port/ibmmq/rest/v2/messaging/qmgr/{qmgrName}/queue/{queueName}/message`

`https://host:port/ibmmq/rest/v3/messaging/qmgr/{qmgrName}/queue/{queueName}/message`

qmgrName

メッセージングのための接続先のキュー・マネージャーの名前を指定します。キュー・マネージャーは mqweb サーバーと同じマシン上になければなりません。

から IBM MQ 9.3.3、ローカルのキューマネージャまたはリモートのキューマネージャのいずれかに接続できます。指定する名前は、mqweb サーバーの設定 **qmgrName** 方法によって異なります：

- mqweb サーバーがローカルのキューマネージャのみに接続するよう設定されている場合は、キューマネージャの名前を使用してください。
- mqweb サーバーがリモートキューマネージャへの接続用に構成されている場合、リモートキューマネージャ接続情報で指定されているキューマネージャの一意の名前を使用してください。

mqweb サーバーがローカルキューマネージャに接続するように設定されているか、リモートキューマネージャに接続するように設定されているかは、コマンド **dspmweb properties** を使用して **mqRestMessagingConnectionMode** プロパティを確認することで判断できます。

キュー・マネージャーの名前には、大/小文字の区別があります。

キュー・マネージャー名にスラッシュ、ピリオド、または % 記号が含まれている場合は、その文字を URL エンコードする必要があります。

- スラッシュ (/) は、%2F としてエンコードする必要があります。
- パーセント記号 (%) は、%25 とエンコードする必要があります。

queueName

次のメッセージを取得するキューの名前を指定します。

キューは、ローカルであるか、またはローカル・キューを指し示す別名として定義することが必要です。

キューの名前には、大/小文字の区別があります。

キュー名にスラッシュまたは % 記号が含まれている場合は、その文字を URL エンコードする必要があります。

- スラッシュ / は、%2F としてエンコードする必要があります。
- % 記号は、%25 としてエンコードする必要があります。

HTTP 接続を使用可能にすれば、HTTPS ではなく HTTP を使用できます。HTTP の使用可能化について詳しくは、[HTTP および HTTPS ポートの構成](#)を参照してください。

オプションの照会パラメーター

REST API V1 ▶ REST API V2 **correlationId=hexValue**

HTTP メソッドが次のメッセージに対応する相関 ID と共に返すように指定します。

hexValue

照会パラメーターは、24 バイトを表す、48 文字の 16 進エンコード・ストリングとして指定する必要があります。

以下に例を示します。

```
../message?correlationId=414d5120514d4144455620202020202067d8bf5923582e02
```

REST API V3 **correlationId=ID:hexValue** または **correlationId=application_specific_value**

HTTP メソッドが次のメッセージに対応する相関 ID と共に返すように指定します。

hexValue

照会パラメーターは、24 バイトを表す 48 文字の 16 進エンコード・ストリングとして指定する必要があり、その前にストリング "ID:" を付ける必要があります。

以下に例を示します。

```
../message?correlationId=ID:414d5120514d4144455620202020202067d8bf5923582e02
```

application_specific_value

照会パラメーターは、アプリケーション固有のストリングとして指定できます。

以下に例を示します。

```
../message?correlationId=My-Custom-CorrelId
```

REST API V1 ▶ REST API V2 **messageId=hexValue**

HTTP メソッドが次のメッセージに対応するメッセージ ID と共に返すように指定します。

hexValue

照会パラメーターは、24 バイトを表す、48 文字の 16 進エンコード・ストリングとして指定する必要があります。

以下に例を示します。

```
../message?messageId=414d5120514d4144455620202020202067d8ce5923582f07
```

REST API V3 **messageId=ID:hexValue**

HTTP メソッドが次のメッセージに対応するメッセージ ID と共に返すように指定します。

hexValue

照会パラメーターは、24 バイトを表す 48 文字の 16 進エンコード・ストリングとして指定する必要があり、その前にストリング "ID:" を付ける必要があります。

以下に例を示します。

```
../message?messageId=ID:414d5120514d4144455620202020202067d8ce5923582f07
```

wait=integerValue

次のメッセージが使用可能になるまで HTTP メソッドが *integerValue* ミリ秒の間待機するように指定します。

integerValue

照会パラメーターは、ミリ秒の期間を表す整数値として指定する必要があります。最大値は 2147483647 です。

以下に例を示します。

```
../message?wait=120000
```

要求ヘッダー

要求で以下のヘッダーを送信する必要があります。

認証

基本認証を使用している場合、このヘッダーを送信する必要があります。詳しくは、[REST API での HTTP 基本認証の使用](#) を参照してください。

ibm-mq-rest-csrf-token

このヘッダーを設定する必要がありますが、その値はブランクを含む任意のものにすることができます。

要求で以下のヘッダーをオプションで送信できます。

Accept-Charset

このヘッダーを使用して、応答のために許容される文字セットを指定できます。指定する場合、このヘッダーは UTF-8 として設定する必要があります。

Accept-Language

このヘッダーは、応答メッセージ本体で返される例外メッセージやエラー・メッセージに必要な言語を指定します。

要求本体の形式

なし。

セキュリティ要件

呼び出し元は mqweb サーバーで認証する必要があります。MQWebAdmin 役割および MQWebAdminRO 役割は、messaging REST API には適用されません。REST API のセキュリティについて詳しくは、[IBM MQ Console および REST API のセキュリティ](#) を参照してください。

ユーザーは、mqweb サーバーで認証された後に、messaging REST API および administrative REST API の両方を使用できるようになります。

呼び出し元のセキュリティ・プリンシパルには、指定されたキューからメッセージを取得するための権限が付与されていないと見なされます。

- リソース URL の {queueName} 部分で指定されるキューは、GET に対応している必要があります。
-   リソース URL の {queueName} 部分で指定されたキューについて、+GET、+INQ、および+BROWSE 権限が呼び出し元のセキュリティ・プリンシパルに付与されている必要があります。
-  リソース URL の {queueName} 部分で指定されるキュー UPDATE では、アクセス権限が呼び出し元のセキュリティ・プリンシパルに付与されている必要があります。

このセキュリティプリンシパルは、mqweb サーバーを起動したユーザー、または mqweb サーバーにログインしているユーザーである可能性があります。接続先のキューマネージャがリモートキューマネージャの場合、セキュリティプリンシパルは代わりに、リモートキューマネージャ接続情報で指定された認証情報によって提供されるユーザー、またはチャンネルセキュリティルールによって決定される別のユーザーと

なる可能性があります。詳細については、[「.NET Framework アプリケーションで使用するセキュリティ主体の messaging REST API 決定」](#)を参照してください。

ALW では AIX, Linux, and Windows、コマンド **setmqaut** を使用して、セキュリティプリンシパルにリソース IBM MQ を使用する権限を付与できます。詳しくは、[setmqaut \(権限の付与または取り消し\)](#)を参照してください。

z/OS では、[z/OS でのセキュリティのセットアップ](#)を参照してください。.

応答状況コード

200

メッセージは正常に受信されました。

204

メッセージが使用できません。

400

無効なデータが指定されました。

例えば、無効な照会パラメーターの値が指定されました。

401

認証されませんでした。

呼び出し元は mqweb サーバーに対して認証されている必要があり、1 つ以上の MQWebAdmin、MQWebAdminRO、または MQWebUser ロールのメンバーでなければなりません。ibm-mq-rest-csrf-token ヘッダーも指定する必要があります。詳しくは、[2192 ページの『セキュリティ要件』](#)を参照してください。

403

許可がありません。

呼び出し元は mqweb サーバーで認証を受け、有効なプリンシパルと関連付けられました。しかし、プリンシパルが、必要な IBM MQ リソースのすべてまたはサブセットに対するアクセス権限を持っていないか、または MQWebUser 役割に含まれていません。必要なアクセス権について詳しくは、[2192 ページの『セキュリティ要件』](#)を参照してください。

404

キューが存在しません。

405

キューでは GET が禁止されています。

500

サーバーの問題または IBM MQ からのエラー・コード。

501

HTTP 応答を作成できませんでした。

例えば、受信したメッセージのタイプが正しくないか、タイプは正しくても本体を処理できない場合があります。

502

メッセージング・プロバイダーが必要な関数をサポートしていないので、現在のセキュリティ・プリンシパルはメッセージを受信できません。例えば、mqweb サーバーのクラスパスが無効です。

503

キュー・マネージャーが実行されていません。

応答ヘッダー

応答では以下のヘッダーが返されます。

Content-Language

エラーや例外が発生した場合の応答メッセージの言語 ID を指定します。エラーまたは例外条件に必要な言語を示すために、Accept-Language 要求ヘッダーと一緒に使用されます。要求された言語がサポートされていない場合は、mqweb サーバーのデフォルトが使用されます。

Content-Length

内容が存在しない場合にも適用される、HTTP 応答本体の長さを指定します。値には、メッセージ・データの長さ (バイト数) が含まれます。

Content-Type

受信したメッセージの応答本体で返されるコンテンツのタイプを指定します。成功すると、値は `text/plain; charset=utf-8` になります。エラーまたは例外が発生した場合、値は `application/json; charset=utf-8` になります。

REST API V1 ▶ REST API V2 **ibm-mq-md-correlationId**

受信メッセージの相関 ID を指定します。受信したメッセージに有効な相関 ID が含まれている場合、このヘッダーが返されます。これは 24 バイトを表す、48 文字の 16 進エンコード・ストリングとして表記されます。

以下に例を示します。

```
ibm-mq-md-correlationId: 414d5120514d4144455620202020202067d8bf5923582e02
```

REST API V3 **ibm-mq-md-correlationId**

受信メッセージの相関 ID を指定します。受信したメッセージに有効な相関 ID が含まれている場合、このヘッダーが返されます。相関 ID は、以下のいずれかの形式をとります。

- ストリング "ID:" の接頭部が付いた、24 バイトを表す 48 文字の 16 進エンコード・ストリング。以下に例を示します。

```
ibm-mq-md-correlationId: ID:414d5120514d4144455620202020202067d8bf5923582e02
```

- アプリケーション固有の値。この値は、アプリケーション固有のストリングです。

```
ibm-mq-md-correlationId: My-Custom-CorrelId
```

ibm-mq-md-expiry

受信メッセージの有効期限までの残り時間を示します。ヘッダーは、次の値のうちのいずれかです。

unlimited

メッセージは満了しません。

整数値

メッセージの有効期限までの残り時間 (ミリ秒単位)。

REST API V1 ▶ REST API V2 **ibm-mq-md-messageId**

IBM MQ によってこのメッセージに割り振られるメッセージ ID を指定します。ibm-mq-md-correlationId ヘッダーと同様に、24 バイトを表す 48 文字の 16 進エンコード・ストリングとして表されます。

以下に例を示します。

```
ibm-mq-md-messageId: 414d5120514d4144455620202020202067d8ce5923582f07
```

REST API V3 **ibm-mq-md-messageId**

IBM MQ によってこのメッセージに割り振られるメッセージ ID を指定します。ibm-mq-md-correlationId ヘッダーと同様に、これは 24 バイトを表す 48 文字の 16 進エンコード・ストリングとして表され、接頭部にストリング "ID:" が付きます。

以下に例を示します。

```
ibm-mq-md-messageId: ID:414d5120514d4144455620202020202067d8ce5923582f07
```

ibm-mq-md-persistence

受信メッセージの持続性を指定します。ヘッダーは、次の値のうちのいずれかです。

nonPersistent

システム障害後またはキュー・マネージャー再始動後に、メッセージは存続しません。

persistent

メッセージはシステムの障害後、またはキュー・マネージャーの再始動後も存続します。

REST API V3 **ibm-mq-md-priority**

メッセージ優先順位の設定を戻します。以下に例を示します。

```
ibm-mq-md-priority: 3
```

ibm-mq-md-replyTo

受信メッセージの返信先宛先を指定します。ヘッダーの形式は、応答先キューおよびキュー・マネージャー `replyQueue@replyQmgr` の標準表記を使用します。

以下に例を示します。

```
ibm-mq-md-replyTo: myReplyQueue@myReplyQMGr
```

ibm-mq-解決済み-qmgr

リクエストに使用されたキューマネージャの名前を指定します。リソース URL で一意の名前が使用された場合、このヘッダーは、その一意の名前に関連付けられたキューマネージャの名前を識別します。リソース URL で使用される一意の名前がキューマネージャグループを指す場合、このヘッダーはグループ内で使用されたキューマネージャを識別します。

REST API V3 **ibm-mq-usr**

メッセージ・ユーザー定義プロパティを戻します。メッセージに複数のプロパティを設定できます。この場合、`ibm-mq-usr` 応答ヘッダーの2つ以上の個別のインスタンスが存在します。

以下に例を示します。

```
ibm-mq-usr: myIPProp;5;short
ibm-mq-usr: mySProp;"hi";string
ibm-mq-usr: myBProp>true;boolean
```

プロパティの構文は次のとおりです。

```
ibm-mq-usr: property_name; user_value; user_type
```

property_name

指定されているユーザー・プロパティの名前。これは有効な JMS プロパティ名でなければなりません。

user_value

プロパティの値。

user_type

プロパティのタイプは以下のとおりです。

- `boolean` (`true/false`、MQBOOL)
- `byte` (8 ビット整数、MQINT8)
- `short` (16 ビット整数、MQINT16)
- `integer` (32 ビット整数、MQINT32)
- `long` (64 ビット整数、MQINT64)
- `float` (32 ビット実数、MQFLOAT32)
- `double` (64 ビット実数、MQFLOAT64)
- `string` (引用符付きストリング)

応答本体の形式

常に実行された場合、応答本体には受信メッセージからのメッセージ本体が入ります。エラーが発生した場合、応答本体に JSON 形式のエラー・メッセージが入ります。どちらの応答も UTF-8 エンコードされます。詳しくは、[REST API エラー処理](#)を参照してください。

例

以下の例では、パスワード `mquser` を使用して `mquser` というユーザーにログインします。cURL では、ログイン要求は、次の Windows の例のようになります。LTPA トークンは、`-c` フラグを使用して `cookiejar.txt` ファイルに保管されます。

指定されたキュー・マネージャーおよびキューからメッセージの要約リストを参照します。キュー・マネージャーはmqweb サーバーと同じマシン上になければなりません。要約データが HTTP 応答本体に JSON フォーマット済みアレイとして返されます。データにはメッセージのペイロードは含まれません。データは、現在のユーザー・コンテキストを使用して受け取ります。メッセージは関連キューから除去されません。

取得禁止になっているキューから、使用可能なメッセージのリストを取得する要求が行われた場合、空の JSON 配列が返されます。

- [2197 ページの『リソース URL』](#)
- [2198 ページの『オプションの照会パラメーター』](#)
- [2199 ページの『要求ヘッダー』](#)
- [2199 ページの『要求本体の形式』](#)
- [2199 ページの『セキュリティ要件』](#)
- [2200 ページの『応答状況コード』](#)
- [2200 ページの『応答ヘッダー』](#)
- [2201 ページの『応答本体の形式』](#)
- [2201 ページの『例』](#)

リソース URL

`https://host:port/ibmmq/rest/v1/messaging/qmgr/{qmgrName}/queue/{queueName}/messagelist`

`https://host:port/ibmmq/rest/v2/messaging/qmgr/{qmgrName}/queue/{queueName}/messagelist`

`https://host:port/ibmmq/rest/v3/messaging/qmgr/{qmgrName}/queue/{queueName}/messagelist`

qmgrName

メッセージングのための接続先のキュー・マネージャーの名前を指定します。キュー・マネージャーはmqweb サーバーと同じマシン上になければなりません。

から IBM MQ 9.3.3、ローカルのキューマネージャまたはリモートのキューマネージャのいずれかに接続できます。指定する名前は、mqweb サーバーの設定 **qmgrName** 方法によって異なります：

- mqweb サーバーがローカルのキューマネージャのみに接続するよう設定されている場合は、キューマネージャの名前を使用してください。
- mqweb サーバーがリモートキューマネージャへの接続用に構成されている場合、リモートキューマネージャ接続情報で指定されているキューマネージャの一意の名前を使用してください。

mqweb サーバーがローカルキューマネージャに接続するように設定されているか、リモートキューマネージャに接続するように設定されているかは、コマンド **dspmweb properties** を使用して **mqRestMessagingConnectionMode** プロパティを確認することで判断できます。

キュー・マネージャーの名前には、大/小文字の区別があります。

キュー・マネージャー名にスラッシュ、ピリオド、または % 記号が含まれている場合は、その文字を URL エンコードする必要があります。

- スラッシュ (/) は、%2F としてエンコードする必要があります。
- パーセント記号 (%) は、%25 とエンコードする必要があります。

queueName

メッセージを参照するキューの名前を指定します。

キューは、ローカルであるか、またはローカル・キューを指し示す別名として定義することが必要です。

キューの名前には、大/小文字の区別があります。

キュー名にスラッシュまたは % 記号が含まれている場合は、その文字を URL エンコードする必要があります。

- スラッシュ / は、%2F としてエンコードする必要があります。
- % 記号は、%25 としてエンコードする必要があります。

HTTP 接続を使用可能にすれば、HTTPS ではなく HTTP を使用できます。HTTP の使用可能化について詳しくは、[HTTP および HTTPS ポートの構成](#)を参照してください。

オプションの照会パラメーター

REST API V2 **correlationId=hexValue**

HTTP メソッドが次のメッセージを対応する相関 ID と共に返すように指定します。

hexValue

照会パラメーターは、24 バイトを表す、48 文字の 16 進エンコード・ストリングとして指定する必要があります。

以下に例を示します。

```
../messageList?correlationId=414d5120514d4144455620202020202067d8bf5923582e02
```

REST API V3 **correlationId=ID:hexValue** または **correlationId=application_specific_value**

HTTP メソッドが、対応する相関 ID を持つメッセージのリストを返すことを指定します。

hexValue

照会パラメーターは、24 バイトを表す 48 文字の 16 進エンコード・ストリングとして指定する必要があり、その前にストリング "ID:" を付ける必要があります。

以下に例を示します。

```
../message?correlationId=ID:414d5120514d4144455620202020202067d8bf5923582e02
```

application_specific_value

照会パラメーターは、アプリケーション固有のストリングとして指定できます。

以下に例を示します。

```
../message?correlationId=My-Custom-CorrelId
```

REST API V1 → REST API V2 **messageId=hexValue**

HTTP メソッドが次のメッセージを対応するメッセージ ID と共に返すように指定します。

hexValue

照会パラメーターは、24 バイトを表す、48 文字の 16 進エンコード・ストリングとして指定する必要があります。

以下に例を示します。

```
../message?messageId=414d5120514d4144455620202020202067d8ce5923582f07
```

REST API V3 **messageId=ID:hexValue**

HTTP メソッドが次のメッセージを対応するメッセージ ID と共に返すように指定します。

hexValue

照会パラメーターは、24 バイトを表す 48 文字の 16 進エンコード・ストリングとして指定する必要があり、その前にストリング "ID:" を付ける必要があります。

以下に例を示します。

```
../message?messageId=ID:414d5120514d4144455620202020202067d8ce5923582f07
```

limit=integerValue

HTTP メソッド応答本体を *integerValue* JSON 要素に制限することを指定します。

integerValue

照会パラメーターは、JSON 応答本体に含まれる要素の最大数を表す整数値として指定する必要があります。

デフォルト値は 10 で、最大値は 2147483647 です。

以下に例を示します。

```
../messagelist?limit=250
```

要求ヘッダー

要求で以下のヘッダーを送信する必要があります。

認証

基本認証を使用している場合、このヘッダーを送信する必要があります。詳しくは、[REST API での HTTP 基本認証の使用](#) を参照してください。

ibm-mq-rest-csrf-token

このヘッダーを設定する必要がありますが、その値はブランクを含む任意のものにすることができます。

要求で以下のヘッダーをオプションで送信できます。

Accept-Charset

このヘッダーを使用して、応答のために許容される文字セットを指定できます。指定する場合、このヘッダーは UTF-8 として設定する必要があります。

Accept-Language

このヘッダーは、応答メッセージ本体で返される例外メッセージやエラー・メッセージに必要な言語を指定します。

要求本体の形式

なし。

セキュリティ要件

呼び出し元は mqweb サーバーで認証する必要があります。MQWebAdmin 役割および MQWebAdminRO 役割は、messaging REST API には適用されません。REST API のセキュリティについて詳しくは、[IBM MQ Console および REST API のセキュリティ](#) を参照してください。

ユーザーは、mqweb サーバーで認証された後に、messaging REST API および administrative REST API の両方を使用できるようになります。

呼び出し元のセキュリティ・プリンシパルには、指定されたキューからメッセージを参照するための権限が付与されていなければなりません。

- リソース URL の {queueName} 部分で指定されるキューは、BROWSE に対応している必要があります。
-   リソース URL の {queueName} 部分で指定されたキューについて、+GET、+INQ、および+BROWSE 権限が呼び出し元のセキュリティ・プリンシパルに付与されている必要があります。
-  リソース URL の {queueName} 部分で指定されるキュー UPDATE では、アクセス権限が呼び出し元のセキュリティ・プリンシパルに付与されている必要があります。

このセキュリティプリンシパルは、mqweb サーバーを起動したユーザー、または mqweb サーバーにログインしているユーザーである可能性があります。接続先のキューマネージャがリモートキューマネージャの場合、セキュリティプリンシパルは代わりに、リモートキューマネージャ接続情報で指定された認証情報によって提供されるユーザー、またはチャネルセキュリティルールによって決定される別のユーザーとなる可能性があります。詳細については、「[.NET Framework アプリケーションで使用されるセキュリティ主体の messaging REST API 決定](#)」を参照してください。

ALW では AIX, Linux, and Windows、コマンド **setmqaut** を使用して、セキュリティプリンシパルにリソース IBM MQ を使用する権限を付与できます。詳しくは、[setmqaut \(権限の付与または取り消し\)](#) を参照してください。

z/OS では、[z/OS でのセキュリティのセットアップ](#)を参照してください。.

応答状況コード

200

メッセージ・リストは正常に受信されました。

400

無効なデータが指定されました。

例えば、無効な照会パラメーターの値が指定されました。

401

認証されませんでした。

呼び出し元は mqweb サーバーに対して認証されている必要があり、1 つ以上の MQWebAdmin、MQWebAdminRO、または MQWebUser ロールのメンバーでなければなりません。ibm-mq-rest-csrf-token ヘッダーも指定する必要があります。詳しくは、[2199 ページの『セキュリティ要件』](#)を参照してください。

403

許可がありません。

呼び出し元は mqweb サーバーで認証を受け、有効なプリンシパルと関連付けられました。しかし、プリンシパルが、必要な IBM MQ リソースのすべてまたはサブセットに対するアクセス権を持っていないか、または MQWebUser 役割に含まれていません。必要なアクセス権について詳しくは、[2199 ページの『セキュリティ要件』](#)を参照してください。

404

キューが存在しません。

500

サーバーの問題または IBM MQ からのエラー・コード。

501

HTTP 応答を作成できませんでした。

例えば、受信したメッセージのタイプが正しくないか、タイプは正しくても本体を処理できない場合があります。

502

メッセージング・プロバイダーが必要な関数をサポートしていないので、現在のセキュリティ・プリンシパルはメッセージを受信できません。例えば、mqweb サーバーのクラスパスが無効です。

503

キュー・マネージャーが実行されていません。

応答ヘッダー

Content-Language

エラーや例外が発生した場合の応答メッセージの言語 ID を指定します。エラーまたは例外条件に必要な言語を示すために、Accept-Language 要求ヘッダーと一緒に使用されます。要求された言語がサポートされていない場合は、mqweb サーバーのデフォルトが使用されます。

Content-Length

内容が存在しない場合にも適用される、HTTP 応答本体の長さを指定します。値には、メッセージ・データ長 (バイト単位) が含まれます。

Content-Type

応答本体のタイプを指定します。値は application/json;charset=utf-8 です。

ibm-mq-解決済み-qmgr

リクエストに使用されたキューマネージャの名前を指定します。リソース URL で一意の名前が使用された場合、このヘッダーは、その一意の名前に関連付けられたキューマネージャの名前を識別します。リソース URL で使用される一意の名前がキューマネージャグループを指す場合、このヘッダーはグループ内で使用されたキューマネージャを識別します。

応答本体の形式

正常に実行された場合、応答本体は UTF-8 エンコード応答です。応答で返される外部 JSON オブジェクトの内側には、`messages` という単一の JSON 配列が含まれています。配列の各エレメントは、キュー上のメッセージに関する情報を含む JSON オブジェクトです。各エレメントに以下の属性が含まれます。

REST API V1 REST API V2 `correlationId`

メッセージの相関 ID を指定します。メッセージに有効な相関 ID が含まれている場合、この値が返されます。

REST API V3 `correlationId`

メッセージの相関 ID を指定します。メッセージに有効な相関 ID が含まれている場合、この値が返されます。相関 ID には、「ID:」ストリングが接頭部として付けられます。また、アプリケーション固有の値にすることもできます。

REST API V1 REST API V2 `messageId`

IBM MQ によってこのメッセージに割り振られるメッセージ ID を指定します。これは 24 バイトを表す、48 文字の 16 進エンコード・ストリングとして表記されます。

REST API V3 `messageId`

IBM MQ によってこのメッセージに割り振られるメッセージ ID を指定します。これは 24 バイトを表す、48 文字の 16 進エンコード・ストリングとして表記されます。メッセージ ID の先頭に「ID:」ストリングが付いています。

形式

MQMD format フィールドを指定します。通常の場合では、テキスト・メッセージには IBM MQ MQSTR 値が入ります。

取得禁止になっているキュー上のメッセージのリストを取得する要求が行われた場合、空の JSON 配列が返されます。

エラーが発生した場合、応答本体に JSON 形式のエラー・メッセージが入ります。詳しくは、[REST API エラー処理](#)を参照してください。

例

以下の例では、v2 のリソース URL を使用しています。IBM MQ の IBM MQ 9.1.5 よりも前のバージョンを使用している場合、v1 リソースを代わりに使用する必要があります。つまり、リソース URL では、URL の例で v2 が使用されている v1 が置き換わります。

以下の例では、パスワード `muser` を使用して `muser` というユーザーにログインします。cURL では、ログイン要求は、次の Windows の例のようになります。LTPA トークンは、`-c` フラグを使用して `cookiejar.txt` ファイルに保管されます。

```
curl -k "https://localhost:9443/ibmmq/rest/v2/login" -X POST
-H "Content-Type: application/json" --data '{"username":"muser","password":"muser"}'
-c c:\cookiejar.txt
```

ユーザーがログインすると、さらに要求を認証するために LTPA トークンと `ibm-mq-rest-csrf-token` HTTP ヘッダーが使用されます。`ibm-mq-rest-csrf-token_value` は、ブランクを含む任意の値にすることができます。

- 以下の Windows cURL の例は、デフォルトのオプションを使用して、キュー・マネージャー QM1 上のキュー Q1 から次に使用可能な 10 のメッセージをリストします。

```
curl -k "https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/queue/Q1/messagelist"  
-X GET -b c:\cookiejar.txt -H "ibm-mq-rest-csrf-token: token-value"  
-H "Accept: application/json"
```

- 以下の Windows cURL の例は、デフォルトのオプションを使用して、キュー・マネージャー QM1 上のキュー Q1 から次に使用可能な 200 のメッセージをリストします。

```
curl -k "https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/queue/Q1/messagelist?limit=200"
-X GET -b c:\cookiejar.txt -H "ibm-mq-rest-csrf-token: token-value"
-H "Accept: application/json"
```

- **REST API V1**、**REST API V2**以下の Windows cURL の例は、対応する相関 ID (000000000000000000000000000000000000abcdabcd) を持つメッセージだけを、キュー・マネージャー QM1 上のキュー Q1 からリストします。

```
curl -k "https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/queue/Q1/messageList?correlationId=0000000000000000000000000000000000000000abcdcabcd"  
-X GET -b c:\cookiejar.txt -H "ibm-mq-rest-csrftoken: token-value"  
-H "Accept: application/json"
```

- REST API V3

以下の Windows cURL の例は、前の例と同じですが、REST API V3 を使用しています。この例では、キューマネージャー QM10000000000000000000000000000000000000000000abcdabcd: の Q1 キューから、対応する相関 ID を持つメッセージのみをリストします：

```
curl -k "https://localhost:9443/ibmmq/rest/v3/messaging/qmgr/QM1/queue/Q1/messageList?correlationId=ID:000000000000000000000000000000000000000000abcdabcd"  
-X GET -b c:\cookiejar.txt -H "ibm-mq-rest-csrftoken: token-value"  
-H "Accept: application/json"
```

/messaging/qmgr/{qmgrName}/topic/{topicString}/message

/messaging/qmgr/{qmgrName}/topic/{topicString}/message リソースを指定した HTTP POST メソッドを使用すると、指定したキュー・マネージャー上の指定したトピックにメッセージをパブリッシュできます。

POST /messaging/qmgr/{qmgrName}/topic/{topicString}/message

/messaging/qmgr/{qmgrName}/topic/{topicString}/message リソースを指定した HTTP POST メソッドを使用すると、指定したキュー・マネージャー上の指定したトピックにメッセージをパブリッシュできます。

HTTP 要求本体のテキスト・ベースのメッセージを、指定されたキュー・マネージャーおよびトピックにパブリッシュします。このキュー・マネージャーは、mqweb サーバーと同じマシンに配置する必要があります。テキスト・ベースのメッセージのみがサポートされます。メッセージは現在のユーザーコンテキストを使用して MQSTR または JMS TextMessage 形式で公開され、REST API ではデフォルトのメッセージ V2、および REST asDestination API 用 V3 優先度は 4 です。

REST API V3 では、ユーザー定義のメッセージ・プロパティを指定し、メッセージ優先度を含める機能を追加します。ibm-mq-md 優先順位および ibm-mq-usr 要求ヘッダーは、REST API V3 でのみ使用可能です。ibm-mq-md-correlationId 要求ヘッダーは、REST API V3 では異なるフォーマットになっています。ヘッダーは、アプリケーション固有の ID にすることも、エンコードされたストリングの場合は ID: 接頭部を必要とすることもできます。POST 要求にユーザー定義メッセージまたはアプリケーション固有の関連 ID が含まれている場合、メッセージは JMS TextMessage としてフォーマット設定されます。

- [2203 ページの『リソース URL』](#)
- [2204 ページの『要求ヘッダー』](#)

- [2205 ページの『要求本体の形式』](#)
- [2206 ページの『セキュリティ要件』](#)
- [2206 ページの『応答状況コード』](#)
- [2207 ページの『応答ヘッダー』](#)
- [2207 ページの『応答本体の形式』](#)
- [2207 ページの『例』](#)

リソース URL

`https://host:port/ibmmq/rest/v2/messaging/qmgr/{qmgrName}/topic/{topicString}/message`

`https://host:port/ibmmq/rest/v3/messaging/qmgr/{qmgrName}/topic/{topicString}/message`

qmgrName

メッセージングのために接続するキューマネージャの名前を指定します。

から IBM MQ 9.3.3、ローカルのキューマネージャまたはリモートのキューマネージャのいずれかに接続できます。指定する名前は、mqweb サーバーの設定 **qmgrName** 方法によって異なります：

- mqweb サーバーがローカルのキューマネージャのみに接続するよう設定されている場合は、キューマネージャの名前を使用してください。
- mqweb サーバーがリモートキューマネージャへの接続用に構成されている場合、リモートキューマネージャ接続情報で指定されているキューマネージャの一意の名前を使用してください。

mqweb サーバーがローカルキューマネージャに接続するように設定されているか、リモートキューマネージャに接続するように設定されているかは、コマンド **dspmweb properties** を使用して **mqRestMessagingConnectionMode** プロパティを確認することで判断できます。

名前は大文字小文字が区別されます。

名前がスラッシュ、ピリオド、またはパーセント記号を含む場合、これらの文字は URL エンコードする必要があります：

- スラッシュは、%2F としてエンコードする必要があります。
- ピリオドは、%2E としてエンコードする必要があります。
- パーセント記号は %25 としてエンコードする必要があります。

topicString

メッセージをパブリッシュするトピック・ストリングを指定します。

トピック・ストリングでは大/小文字を区別します。トピック・ストリングには、スラッシュ区切り文字で区切られた、複数のトピック・レベルを含めることができます。

トピック・ストリングに % 記号、ピリオド、または疑問符が含まれている場合、その文字を次のように URL エンコードする必要があります。

- パーセント記号は %25 としてエンコードする必要があります。
- ピリオドは、%2E としてエンコードする必要があります。
- 疑問符は %3F としてエンコードする必要があります。

トピック・ストリングの先頭または末尾がスラッシュの場合は、%2F でエンコードする必要があります。

例えば、以下のように、トピック・ストリングにパブリッシュします。

- キュー・マネージャー MY.QMGR 上の sport/football には、以下の URL を使用します。

```
https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/MY%2EQMGR/topic/sport/football/message
```

- キュー・マネージャー MY.QMGR 上の/sport/football には、以下の URL を使用します。

```
https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/MY%2EQMGR/topic/%2Fsport/football/  
message
```

HTTP 接続を使用可能にすれば、HTTPS ではなく HTTP を使用できます。HTTP の使用可能化について詳しくは、[HTTP および HTTPS ポートの構成](#)を参照してください。

要求ヘッダー

要求で以下のヘッダーを送信する必要があります。

認証

基本認証を使用している場合、このヘッダーを送信する必要があります。詳しくは、[REST API での HTTP 基本認証の使用](#)を参照してください。

Content-Type

このヘッダーは、以下のいずれかの値で送信する必要があります。

- text/plain;charset=utf-8
- text/html;charset=utf-8
- text/xml;charset=utf-8
- application/json;charset=utf-8
- application/xml;charset=utf-8

ibm-mq-rest-csrf-token

このヘッダーを設定する必要がありますが、その値はブランクを含む任意のものにすることができます。

要求で以下のヘッダーをオプションで送信できます。

Accept-Language

このヘッダーは、応答メッセージ本体で返される例外メッセージやエラー・メッセージに必要な言語を指定します。

ibm-mq-md-expiry

このヘッダーは、作成されるメッセージの有効期限を設定します。メッセージの有効期限は、メッセージがキュー・マネージャーに到着した時点から始まります。このため、ネットワーク待ち時間は無視されます。ヘッダーは、以下のいずれかの値として指定する必要があります。

unlimited

メッセージは満了しません。

この値がデフォルト値です。

整数値

メッセージが期限切れになるまでのミリ秒数。

0 から 99999999900 の範囲に制限されます。

ibm-mq-md-persistence

このヘッダーは、作成されるメッセージの持続性を設定します。ヘッダーは、以下のいずれかの値として指定する必要があります。

nonPersistent

システム障害後またはキュー・マネージャー再始動後に、メッセージは存続しません。

この値がデフォルト値です。

persistent

メッセージはシステムの障害後、またはキュー・マネージャーの再始動後も存続します。

REST API V3 ibm-mq-md-priority

このヘッダーは作成されたメッセージの優先順位を設定します。ヘッダーは、以下のいずれかの値として指定する必要があります。

asDestination

このメッセージは、基礎となる IBM MQ キュー・オブジェクトの DEFPRTY 属性に指定されている優先順位を使用します。

整数値

実際の優先順位を 0 から 9 の範囲の整数として指定します。

以下に例を示します。

```
ibm-mq-md-priority: asDestination
```

ibm-mq-md-replyTo

このヘッダーは、作成されるメッセージの返信先宛先を設定します。ヘッダーの形式は、応答先キューとオプションのキュー・マネージャーを提供する標準表記 (replyQueue[@replyQmgr]) を使用します。

以下に例を示します。

```
ibm-mq-md-replyTo: myReplyQueue@myReplyQMgr
```

REST API V3 ibm-mq-usr

要求メッセージのユーザー定義プロパティを設定します。メッセージには複数のプロパティを設定できます。1 つの ibm-mq-usr 要求ヘッダーに複数のコンマ区切りのプロパティを指定することも、ibm-mq-usr 要求ヘッダーの 2 つ以上の個別のインスタンスを使用することもできます。

以下に例を示します。

```
ibm-mq-usr: myIProp;5;short  
ibm-mq-usr: mySProp;"hi";string  
ibm-mq-usr: myBProp>true;boolean  
ibm-mq-usr: myA;5;byte,myB;-10;integer
```

プロパティの構文は次のとおりです。

```
ibm-mq-usr: property_name; user_value; user_type
```

property_name

指定されているユーザー・プロパティの名前。これは有効な JMS プロパティ名でなければなりません。

user_value

プロパティの値。

user_type

プロパティのタイプは以下のとおりです。

- boolean (true/false、MQBOOL)
- byte (8 ビット整数、MQINT8)
- short (16 ビット整数、MQINT16)
- integer (32 ビット整数、MQINT32)
- long (64 ビット整数、MQINT64)
- float (32 ビット実数、MQFLOAT32)
- double (64 ビット実数、MQFLOAT64)
- string (引用符付きストリング)

要求本体の形式

要求本体は、テキスト形式で UTF-8 エンコードを使用する必要があります。特定のテキスト構造にする必要はありません。要求本文テキストを含む MQSTR 形式のメッセージが作成され、指定されたトピックにパブリッシュされます。

REST API V3 REST API V3 ユーザー定義プロパティ、またはアプリケーション固有の関連 ID フィーチャーが使用されている場合は、要求本文テキストを含む JMS `TextMessage` 形式のメッセージが作成され、指定されたキューに書き込まれます。

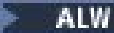
詳しくは、[例](#)を参照してください。

セキュリティ要件

呼び出し元は mqweb サーバーで認証する必要があります。MQWebAdmin 役割および MQWebAdminRO 役割は、messaging REST API には適用されません。REST API のセキュリティについて詳しくは、[IBM MQ Console](#) および [REST API のセキュリティ](#)を参照してください。

ユーザーは、mqweb サーバーで認証された後に、messaging REST API および administrative REST API の両方を使用できるようになります。

呼び出し元のセキュリティ・プリンシパルには、指定されたトピックにメッセージをパブリッシュするための権限が付与されていなければなりません。

- リソース URL の `{topicString}` 部分で指定されるトピックは、PUBLISH が有効になっている必要があります。
-   リソース URL の `{topicString}` 部分で指定されるトピックについては、呼び出し元のセキュリティ・プリンシパルに +PUB 権限が付与されている必要があります。
-  リソース URL の `{topicString}` 部分で指定されるトピックについては、呼び出し元のセキュリティ・プリンシパルに UPDATE アクセス権限が付与されている必要があります。

このセキュリティプリンシパルは、mqweb サーバーを起動したユーザー、または mqweb サーバーにログインしているユーザーである可能性があります。接続先のキューマネージャがリモートキューマネージャの場合、セキュリティプリンシパルは代わりに、リモートキューマネージャ接続情報で指定された認証情報によって提供されるユーザー、またはチャンネルセキュリティルールによって決定される別のユーザーとなる可能性があります。詳細については、「[.NET Framework アプリケーションで使用されるセキュリティ主体の messaging REST API 決定](#)」を参照してください。

 では AIX, Linux, and Windows、コマンド **setmqaut** を使用して、セキュリティプリンシパルにリソース IBM MQ を使用する権限を付与できます。詳しくは、[setmqaut](#) (権限の付与または取り消し) を参照してください。

z/OS では、[z/OS でのセキュリティのセットアップ](#)を参照してください。

messaging REST API で Advanced Message Security (AMS) を使用する場合は、メッセージを投稿するユーザーのコンテキストではなく、mqweb サーバーのコンテキストを使用してすべてのメッセージが暗号化されることに注意してください。

応答状況コード

201

メッセージは正常に作成されてパブリッシュされました。

400

無効なデータが指定されました。

例えば、無効な要求ヘッダーの値が指定されました。

401

認証されませんでした。

呼び出し元は mqweb サーバーに対して認証されている必要があります、1 つ以上の MQWebAdmin、MQWebAdminRO、または MQWebUser ロールのメンバーでなければなりません。ibm-mq-rest-csrf-token ヘッダーも指定する必要があります。詳しくは、[2206 ページの『セキュリティ要件』](#)を参照してください。

403

許可がありません。

呼び出し元は mqweb サーバーで認証を受け、有効なプリンシパルと関連付けられました。しかし、プリンシパルが、必要な IBM MQ リソースのすべてまたはサブセットに対するアクセス権を持っていないか、または MQWebUser 役割に含まれていません。必要なアクセス権について詳しくは、[2206 ページの『セキュリティ要件』](#)を参照してください。

404

キュー・マネージャーがありません。

405

トピックでは PUBLISH が禁止されています。

415

メッセージ・ヘッダーまたはメッセージ本体のメディア・タイプがサポートされていません。

例えば、Content-Type ヘッダーが、サポートされないメディア・タイプに設定されています。

500

サーバーの問題または IBM MQ からのエラー・コード。

502

メッセージング・プロバイダーが必要な関数をサポートしていないので、現在のセキュリティ・プリンシパルはメッセージをパブリッシュできません。例えば、mqweb サーバーのクラスパスが無効です。

503

キュー・マネージャーが実行されていません。

応答ヘッダー

応答では以下のヘッダーが返されます。

Content-Language

エラーや例外が発生した場合の応答メッセージの言語 ID を指定します。エラーまたは例外条件に必要な言語を示すために、Accept-Language 要求ヘッダーと一緒に使用されます。要求された言語がサポートされていない場合は、mqweb サーバーのデフォルトが使用されます。

Content-Length

内容が存在しない場合にも適用される、HTTP 応答本体の長さを指定します。正常に実行された場合、値はゼロになります。

Content-Type

応答本体のタイプを指定します。成功すると、値は text/plain;charset=utf-8 になります。エラーまたは例外が発生した場合、値は application/json;charset=utf-8 になります。

ibm-mq-解決済み-qmgr

リクエストに使用されたキューマネージャの名前を指定します。リソース URL で一意の名前が使用された場合、このヘッダーは、その一意の名前に関連付けられたキューマネージャの名前を識別します。リソース URL で使用される一意の名前がキューマネージャグループを指す場合、このヘッダーはグループ内で使用されたキューマネージャを識別します。

応答本体の形式

メッセージが正常にパブリッシュされた場合、応答本体は空になります。エラーが発生した場合、応答本体にエラー・メッセージが入ります。詳しくは、[REST API エラー処理](#)を参照してください。

例

以下の例では、パスワード mquser を使用して mquser というユーザーにログインします。cURL では、ログイン要求は、次の Windows の例のようになります。LTPA トークンは、-c フラグを使用して cookiejar.txt ファイルに保管されます。

```
curl -k "https://localhost:9443/ibmmq/rest/v1/login" -X POST
-H "Content-Type: application/json" --data '{"username":"'mquser'","password":"'mquser'"}'
-c c:\cookiejar.txt
```

ユーザーがログインすると、さらに要求を認証するために LTPA トークンと `ibm-mq-rest-csrf-token` HTTP ヘッダーが使用されます。 `ibm-mq-rest-csrf-token: token_value` は、ブランクを含む任意の値にすることができます。

- 以下の Windows cURL の例では、デフォルト・オプションを使用して、キュー・マネージャー QM1 上のトピック・ストリング `myTopic` にメッセージをパブリッシュします。メッセージには、テキスト `"Hello World!"` が入ります。

```
curl -k "https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/topic/myTopic/message"
-X POST -b c:\cookiejar.txt -H "ibm-mq-rest-csrf-token: token_value"
-H "Content-Type: text/plain; charset=utf-8" --data "Hello World!"
```

- 以下の Windows cURL の例では、有効期限が 2 分の永続メッセージをキュー・マネージャー QM1 のトピック・ストリング `myTopic/thisTopic` にパブリッシュします。メッセージには、テキスト `"Hello World!"` が入ります。

```
curl -k "https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/topic/myTopic%2FthisTopic/
message"
-X POST -b c:\cookiejar.txt -H "ibm-mq-rest-csrf-token: token_value"
-H "Content-Type: text/plain; charset=utf-8" -H "ibm-mq-md-persistence: persistent"
-H "ibm-mq-md-expiry: 120000" --data "Hello World!"
```

特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものです。

IBM 他の国では、本資料に記載されている製品、サービス、または機能を提供しない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒 103-8510

東京都中央区日本橋箱崎町 19 番 21 号

日本アイ・ビー・エム株式会社

日本アイ・ビー・エム株式会社

法務・知的財産

U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing

Legal and Intellectual Property Law

〒 103-8510

19-21, Nihonbashi-Hakozakicho, Chuo-ku

Tokyo 103-8510, Japan

以下の段落は、英国または当該規定が現地法と矛盾するその他の国には適用されません： インターナショナル・ビジネス・マシーンズ・コーポレーションは、本出版物を"現状有姿のまま"提供し、明示的または黙示的ないかなる種類の保証も一切行いません。これには、権利侵害の非存在、商品性、特定目的への適合性に関する黙示的保証が含まれますが、これらに限定されるものではありません。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

東京都中央区日本橋箱崎町 19 番 21 号

日本アイ・ビー・エム株式会社

Software Interoperability Coordinator, Department 49XA

3605 Highway 52 N

Rochester, MN 55901

U.S.A.

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性があります。その測定値が、一般に利用可能なシステムのものと同一である保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確認できません。IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があります、単に目標を示しているものです。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。

この情報をソフトコピーでご覧になっている場合は、写真やカラーの図表は表示されない場合があります。

プログラミング・インターフェース情報

プログラミングインターフェース情報は、提供されている場合、本プログラムで使用するアプリケーションソフトウェアの作成を支援することを目的としています。

本書には、顧客が MQIBM のサービスを利用するためのプログラムを作成できるようにする、意図されたプログラミングインターフェースに関する情報が含まれています。

ただし、この情報には、診断、修正、および調整情報が含まれている場合があります。診断、修正、調整情報は、お客様のアプリケーション・ソフトウェアのデバッグ支援のために提供されています。

重要: この診断、変更、および調整情報は変更される可能性があるため、プログラミングインターフェースとして使用しないでください。

商標

IBM ロゴ IBM は、ibm.com® 株式会社 IBM の商標であり、世界中の多くの管轄区域で登録されています。商標 IBM の最新のリストは、"著作権および商標情報" www.ibm.com/legal/copytrade.shtml でウェブ上で入手可能です。他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。

Microsoft および Windows は、Microsoft Corporation の米国およびその他の国における商標です。

UNIX は The Open Group の米国およびその他の国における登録商標です。

Linux は、Linus Torvalds 氏の米国およびその他の国における登録商標です。

本製品には、Eclipse プロジェクト（<https://www.eclipse.org/>）によって開発されたソフトウェアが含まれています。

Java およびすべての Java 関連の商標およびロゴは Oracle やその関連会社の米国およびその他の国における商標または登録商標です。



部品番号:

(1P) P/N: