

10.0

*IBM MQ Referencia para el desarrollo de
aplicaciones*



Nota

Antes de utilizar esta información y el producto al que da soporte, lea la información en [“Avisos” en la página 2243](#).

Esta edición se aplica a la versión 9, lanzamiento 4, de IBM® MQ y a todos los lanzamientos y modificaciones posteriores, salvo que se indique lo contrario en nuevas ediciones.

Al enviar información a IBM, usted concede IBM un derecho no exclusivo para utilizar o distribuir la información de la forma que considere oportuna sin incurrir en ninguna obligación hacia usted.

© Copyright International Business Machines Corporation 2007, 2026.

Contenido

Guía de consulta para el desarrollo de aplicaciones.....	7
MQI Referencia de aplicaciones.....	7
Constantes.....	7
Tipos de datos utilizados en el lenguaje de programación « MQI ».....	177
Llamadas de funciones.....	626
objeto, atributos de.....	800
Códigos de retorno.....	881
Normas para validar las opciones de « MQI ».....	882
Mensajes del mandato de publicación/suscripción en cola.....	885
Codificaciones de máquina.....	908
Opciones de informe y distintivos de mensaje.....	911
salida de conversión de datos.....	915
Propiedades especificadas como elementos MQRFH2.....	938
Conversión de páginas de códigos.....	947
Estándares de codificación en plataformas de 64 bits.....	1010
IBM i Referencia de programación de aplicaciones (ILE/RPG).....	1014
Descripciones de tipos de datos en IBM i.....	1016
Llamadas de función en IBM i.....	1278
Atributos de objetos en IBM i.....	1399
Códigos de retorno para IBM i (ILE RPG).....	1447
Normas para validar las opciones de « MQI » en IBM i (ILE RPG).....	1448
Codificaciones de máquina en IBM i.....	1451
Opciones de informe y distintivos de mensaje en IBM i.....	1454
Conversión de datos en IBM i.....	1457
Proceso de conversión en IBM i.....	1458
Convenios de proceso en IBM i.....	1459
Conversión de mensajes de informe en IBM i.....	1463
MQDXP (parámetro de salida de conversión de datos) en IBM i.....	1464
MQXCNCV (Convertir caracteres) en IBM i.....	1469
MQCONVX (Salida de conversión de datos) en IBM i.....	1474
Referencia de salidas de usuarios, salidas de API y servicios instalables.....	1478
Estructura MQIEP.....	1478
Referencia de salida de conversión de datos.....	1482
MQ_PUBLISH_EXIT - Salida de publicación.....	1486
Llamadas de salida de canal y estructuras de datos.....	1494
Llamada de salida de carga de trabajo del clúster y estructuras de datos.....	1561
Referencia a la salida de la API.....	1587
Información de consulta sobre la interfaz de servicios instalables.....	1646
Información de consulta sobre la interfaz de servicios instalables en IBM i.....	1710
Las clases e interfaces de IBM MQ .NET.....	1750
Clase MQAsyncStatus.NET.....	1750
Clase MQAuthenticationInformationRecord.NET.....	1752
Clase MQDestination.NET.....	1753
MQEnvironment.NET clase.....	1755
Clase MQException.NET.....	1758
Clase MQGetMessageOptions.NET.....	1758
Clase MQManagedObject.NET.....	1761
Clase MQMessage.NET.....	1764
Clase MQProcess.NET.....	1775
Clase MQPropertyDescriptor.NET.....	1778
Clase MQPutMessageOptions.NET.....	1779
MQQueue.NET clase.....	1782

MQQueueManager.NET clase.....	1790
Clase MQSubscription.NET.....	1802
Clase MQTopic.NET.....	1803
Interfaz de IMQObjectTrigger.NET.....	1809
Interfaz de MQC.NET.....	1810
Identificadores de juego de caracteres para aplicaciones .NET.....	1810
Clases C++ de IBM MQ.....	1813
Referencia cruzada de C++ y MQI.....	1814
ImqAuthenticationRecord Clase C.....	1831
Clase C++ ImqBinary.....	1833
Clase C++ ImqCache.....	1835
Clase C++ ImqChannel.....	1838
ImqCICSBridgeHeader Clase C++.....	1844
ImqDeadLetterHeader Clase C.....	1850
ImqDistributionList Clase C.....	1853
Clase C++ ImqError.....	1854
ImqGetMessageOptions Clase C.....	1855
Clase C++ ImqHeader.....	1859
ImqIMSBridgeHeader Clase C++.....	1860
Clase C++ ImqItem.....	1863
ImqMessage Clase C++.....	1865
ImqMessageTracker Clase C++.....	1872
Clase C++ ImqNamelist.....	1875
ImqObject Clase C++.....	1876
Clase C++ ImqProcess.....	1882
ImqPutMessageOptions Clase C.....	1884
Clase C++ ImqQueue.....	1886
ImqQueueManager Clase C++.....	1897
ImqReferenceHeader Clase C.....	1913
Clase C++ ImqString.....	1916
Clase C++ ImqTrigger.....	1921
ImqWorkHeader Clase C.....	1924
Propiedades de IBM MQ classes for JMS los objetos.....	1926
Dependencias entre propiedades de objetos IBM MQ classes for JMS.....	1931
APPLICATIONNAME.....	1932
ASYNCEXCEPTION.....	1933
BALOPTIONS.....	1934
BALTYPE.....	1934
BALTIMEOUT.....	1935
BROKERCCDURSUBQ.....	1936
BROKERCCSUBQ.....	1936
BROKERCONQ.....	1937
BROKERDURSUBQ.....	1937
BROKERPUBQ.....	1937
BROKERPUBQMGR.....	1938
BROKERQMGR.....	1938
BROKERSUBQ.....	1939
BROKERVER.....	1939
CCDTURL.....	1940
CCSID.....	1940
CERTVALPO.....	1941
CHANNEL.....	1942
CLEANUP.....	1942
CLEANUPINT.....	1943
ConnectionNameList.....	1943
CLIENTRECONNECTOPTIONS.....	1943
CLIENTRECONNECTTIMEOUT.....	1944
CLIENTID.....	1945

CLONESUPP.....	1945
COMPHDR.....	1946
COMPMSG.....	1946
CONNOPT.....	1947
CONNTAG.....	1948
DESCRIPCIÓN.....	1948
DIRECTAUTH.....	1949
ENCODING.....	1949
EXPIRY.....	1950
FAILIFQUIESCE.....	1950
HOSTNAME.....	1951
LOCALADDRESS.....	1952
ESTILO de nombre de mapa.....	1952
MAXBUFFSIZE.....	1953
MDREAD.....	1953
MDWRITE.....	1954
MDMSGCTX.....	1955
MSGBATCHSZ.....	1955
MSGBODY.....	1956
MSGRETENTION.....	1956
MSGSELECTION.....	1957
MULTICAST.....	1957
OPTIMISTICPUBLICATION.....	1958
OUTCOMENOTIFICATION.....	1959
PERSISTENCE.....	1959
POLLINGINT.....	1960
Puerto.....	1960
PRIORITY.....	1961
PROCESSDURATION.....	1961
PROVIDERVERSION.....	1962
PROXYHOSTNAME.....	1964
PROXYPORT.....	1964
PUBACKINT.....	1965
PUTASYNCALLOWED.....	1965
QMANAGER.....	1966
COLA.....	1966
READAHEADALLOWED.....	1967
READAHEADCLOSEPOLICY.....	1968
RECEIVECCSID.....	1968
RECEIVECONVERSION.....	1969
RECEIVEISOLATION.....	1969
RECEXIT.....	1970
RECEXITINIT.....	1970
REPLYTOSTYLE.....	1971
RESCANINT.....	1971
SECEXIT.....	1972
SECEXITINIT.....	1972
SENDCHECKCOUNT.....	1973
SENDEXIT.....	1973
SENDEXITINIT.....	1974
SHARECONVALLOWED.....	1974
SPARSESUBS.....	1975
SSLCIPHERSUITE.....	1975
SSLCRL.....	1976
SSLFIPSREQUIRED.....	1976
SSLPEERNAME.....	1978
SSLRESETCOUNT.....	1978
STATREFRESHINT.....	1979

SUBSTORE.....	1979
SYNCPOINTALLGETS.....	1980
TARGCLIENT.....	1980
TARGCLIENTMATCHING.....	1981
TEMPMODEL.....	1981
TEMPQPREFIX.....	1982
TEMPTOPICPREFIX.....	1982
ID_CLIENTE_TOKEN.....	1982
SECRETO_DEL_CLIENTE_TOKEN.....	1983
PUNTO FINAL DEL TOKEN.....	1983
POLÍTICA_DE_VALIDACIÓN_DE_CERTIFICADOS_HTTPS_TOKEN.....	1984
TOPIC.....	1985
TRANSPORT.....	1985
WILDCARDFORMAT.....	1986
Política de validación de certificados HTTPS de WMQ_CCDT.....	1986
La propiedad ENCODING.....	1987
Referencia de IBM MQ Message Service Client (XMS) for .NET.....	1988
.NET interfaces.....	1988
Propiedades de objetos XMS.....	2068
Referencia de aplicaciones en desarrollo de Managed File Transfer.....	2139
Ejemplos de uso de fteCreateTransfer para iniciar programas.....	2139
fteAnt : ejecutar tareas Ant en MFT.....	2141
MFT Salidas de usuario para personalización de referencia.....	2167
Formatos de mensaje para los mensajes que puede colocar en la cola de mandatos del agente de MFT.....	2209
Referencia de la REST API de mensajería.....	2209
Recursos de REST API.....	2209
Avisos.....	2243
Información acerca de las interfaces de programación.....	2244
Marcas registradas.....	2245

Guía de consulta para el desarrollo de aplicaciones

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

- [“MQI Referencia de aplicaciones” en la página 7](#)
-  [“IBM i Referencia de programación de aplicaciones \(ILE/RPG\)” en la página 1014](#)
-  [“Conversión de datos en IBM i” en la página 1457](#)
- [“Referencia de salidas de usuarios, salidas de API y servicios instalables” en la página 1478](#)
- [“Las clases e interfaces de IBM MQ .NET” en la página 1750](#)
- [“Clases C++ de IBM MQ” en la página 1813](#)
- [“Propiedades de IBM MQ classes for JMS los objetos” en la página 1926](#)
- [“Referencia de la REST API de mensajería” en la página 2209](#)

Tareas relacionadas

[Desarrollo de aplicaciones](#)

Referencia relacionada

[IBM MQ clases para Java bibliotecas](#)

[Clases IBM MQ para JMS](#)

MQI Referencia de aplicaciones

Utiliza los enlaces que se incluyen en esta sección para ayudarte a desarrollar tus aplicaciones de la interfaz de colas de mensajes (MQI).

Tareas relacionadas

[Desarrollo de aplicaciones MQI con IBM MQ](#)

Referencia relacionada

[“Estándares de codificación en plataformas de 64 bits” en la página 1010](#)

Utilice esta información para obtener información sobre los estándares de codificación en plataformas de 64 bits y los tipos de datos preferidos.

[IBM MQ Clases para Java bibliotecas](#)

[IBM MQ Clases para JMS 2.0](#)

[IBM MQ Clases para Jakarta Messaging](#)

Constantes

Información sobre las constantes que se pueden utilizar en la interfaz de colas de mensajes (MQI).

MQ_* (Longitudes de cadena)

Tabla 1. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQ_ABEND_LONGITUD_DEL_CÓDIGO	4	X'00000004'
MQ_ACCOUNTING_TOKEN_LENGTH	32	X'00000020'
MQ_APPL_FUNCTION_NAME_LENGTH	10	X'0000000A'
MQ_APPL_IDENTITY_DATA_LENGTH	32	X'00000020'
MQ_APPL_NAME_LENGTH	28	X'0000001C'

Tabla 1. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQ_APPL_ORIGIN_DATA_LENGTH	4	X'00000004'
MQ_APPL_TAG_LONGITUD	28	X'0000001C'
MQ_ARM_SUFFIX_LENGTH	2	X'00000002'
MQ_ATTENTION_ID_LENGTH	4	X'00000004'
MQ_AUTH_INFO_CONN_NAME_LENGTH	264	X'00000108'
MQ_AUTH_INFO_DESC_LENGTH	64	X'00000040'
MQ_AUTH_INFO_NAME_LENGTH	48	X'00000030'
MQ_AUTH_INFO_OCSP_URL_LENGTH	256	X'00000100'
MQ_AUTHENTICATOR_LENGTH	8	X'00000008'
MQ_AUTO_REORG_CATÁLOGO_LONGITUD	44	X'0000002C'
Duración de la reorganización automática de MQ	4	X'00000004'
MQ_BATCH_INTERFACE_ID_LENGTH	8	X'00000008'
MQ_BRIDGE_NAME_LENGTH	24	X'00000018'
MQ_CANCEL_CODE_LENGTH	4	X'00000004'
MQ_CF_STRUC_DESC_LONGITUD	64	X'00000040'
MQ_CF_STRUC_NAME_LENGTH	12	X'0000000C'
MQ_CHANNEL_DATE_LENGTH	12	X'0000000C'
MQ_CHANNEL_DESC_LENGTH	64	X'00000040'
MQ_CHANNEL_NAME_LENGTH	20	X'00000014'
MQ_CHANNEL_TIME_LENGTH	8	X'00000008'
MQ_CHINIT_SERVICE_PARM_LENGTH	32	X'00000020'
MQ_CICS_FILE_NAME_LENGTH	8	X'00000008'
MQ_CLIENT_ID_LENGTH	23	X'00000017'
MQ_CLUSTER_NAME_LENGTH	48	X'00000030'
MQ_CONN_NAME_LENGTH	264	X'00000108'
MQ_CONN_TAG_LENGTH	128	X'00000080'
MQ_CONNECTION_ID_LENGTH	24	X'00000018'
MQ_CORREL_ID_LENGTH	24	X'00000018'
MQ_CREATION_DATE_LENGTH	12	X'0000000C'
MQ_CREATION_TIME_LENGTH	8	X'00000008'
MQ_DATE_LENGTH	12	X'0000000C'
MQ_LONGITUD_DEL_NOMBRE_DISTINTIVO	1024	X'00000400'
MQ_DNS_GROUP_NAME_LENGTH	18	X'00000012'
MQ_EXIT_DATA_LENGTH	32	X'00000020'
MQ_EXIT_INFO_NAME_LENGTH	48	X'00000030'
MQ_EXIT_NAME_LENGTH	(value differs by platform or version)	
MQ_EXIT_PD_AREA_LENGTH	48	X'00000030'
MQ_EXIT_USER_AREA_LENGTH	16	X'00000010'
MQ_FACILITY_LENGTH	8	X'00000008'

Tabla 1. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQ_FACILITY_LIKE_LENGTH	4	X'00000004'
MQ_FORMAT_LENGTH	8	X'00000008'
MQ_FUNCTION_LENGTH	4	X'00000004'
MQ_GROUP_ID_LENGTH	24	X'00000018'
MQ_LDAP_LONGITUD_CONTRASEÑA	32	X'00000020'
MQ_LISTENER_NAME_LENGTH	48	X'00000030'
MQ_LISTENER_DESC_LENGTH	64	X'00000040'
MQ_LOCAL_ADDRESS_LENGTH	48	X'00000030'
MQ_LTERM_OVERRIDE_LENGTH	8	X'00000008'
MQ_LU_NAME_LENGTH	8	X'00000008'
MQ_LUWID_LONGITUD	16	X'00000010'
MQ_MAX_EXIT_NAME_LENGTH	128	X'00000080'
MQ_MAX_MCA_ID_DE_USUARIO_LONGITUD	64	X'00000040'
MQ_MAX_PROPERTY_NAME_LENGTH	4095	X'0000FFFF'
MQ_MAX_ID_DE_USUARIO_LONGITUD	64	X'00000040'
MQ_MCA_LONGITUD_DEL_NOMBRE_DEL_TRABAJO	28	X'0000001C'
MQ_MCA_NOMBRE_LONGITUD	20	X'00000014'
MQ_MCA_LONGITUD_DE_LOS_DATOS_DEL_USUARIO	32	X'00000020'
MQ_MCA_ID_DE_USUARIO_LONGITUD	(value differs by platform or version)	(value differs by platform or version)
MQ_MFS_MAP_NAME_LENGTH	8	X'00000008'
MQ_MODE_NAME_LENGTH	8	X'00000008'
MQ_MSG_HEADER_LENGTH	4000	X'0000FA0'
MQ_MSG_ID_LENGTH	24	X'00000018'
MQ_MSG_TOKEN_LENGTH	16	X'00000010'
MQ_NAMELIST_DESC_LENGTH	64	X'00000040'
MQ_NAMELIST_NAME_LENGTH	48	X'00000030'
MQ_NHA_INSTANCE_NAME_LENGTH	48	X'00000030'
MQ_OBJECT_INSTANCE_ID_LENGTH	24	X'00000018'
MQ_OBJECT_NAME_LENGTH	48	X'00000030'
MQ_PASS_TICKET_APPL_LENGTH	8	X'00000008'
MQ_PASSWORD_LENGTH	12	X'0000000C'
MQ_PROCESS_APPL_ID_LENGTH	256	X'00000100'
MQ_PROCESS_DESC_LENGTH	64	X'00000040'
MQ_PROCESS_ENV_DATA_LENGTH	128	X'00000080'
MQ_PROCESS_NAME_LENGTH	48	X'00000030'
MQ_PROCESS_USER_DATA_LENGTH	128	X'00000080'
MQ_PROGRAM_NAME_LENGTH	20	X'00000014'
MQ_PUT_APPL_NAME_LENGTH	28	X'0000001C'
MQ_PUT_DATE_LENGTH	8	X'00000008'

Tabla 1. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQ_PUT_TIME_LENGTH	8	X'00000008'
MQ_Q_DESC_LONGITUD	64	X'00000040'
MQ_Q_MGR_DESC_LONGITUD	64	X'00000040'
MQ_Q_MGR_IDENTIFICADOR_LONGITUD	48	X'00000030'
MQ_Q_MGR_NAME_LENGTH	48	X'00000030'
MQ_Q_NAME_LENGTH	48	X'00000030'
MQ_QSG_NAME_LENGTH	4	X'00000004'
MQ_REMOTE_SYS_ID_LENGTH	4	X'00000004'
MQ_SECURITY_ID_LENGTH	40	X'00000028'
MQ_SELECTOR_LENGTH	10240	X'00002800'
MQ_SERVICE_ARGS_LENGTH	255	X'000000FF'
MQ_SERVICE_COMMAND_LENGTH	255	X'000000FF'
MQ_SERVICE_DESC_LENGTH	64	X'00000040'
MQ_SERVICE_NAME_LENGTH	32	X'00000020'
MQ_SERVICE_PATH_LENGTH	255	X'000000FF'
MQ_SERVICE_STEP_LENGTH	8	X'00000008'
MQ_SHORT_CONN_NAME_LENGTH	20	X'00000014'
MQ_SHORT_DNAME_LENGTH	256	X'00000100'
MQ_SSL_CIFRADO_ESPECÍFICO_LONGITUD	32	X'00000020'
MQ_SSL_CRYPTO_HARDWARE_LONGITUD	256	X'00000100'
MQ_SSL_HANDSHAKE_STAGE_LENGTH	32	X'00000020'
MQ_SSL_KEY_LIBRARY_LENGTH	44	X'0000002C'
MQ_SSL_KEY_MEMBER_LENGTH	8	X'00000008'
MQ_SSL_KEY_REPOSITORY_LENGTH	256	X'00000100'
MQ_SSL_PEER_NAME_LENGTH	1024	X'00000400'
MQ_SSL_LONGITUD_MÍNIMA_DEL_NOMBRE_DEL_PAR	256	X'00000100'
MQ_START_CODE_LENGTH	4	X'00000004'
MQ_STORAGE_CLASS_DESC_LENGTH	64	X'00000040'
MQ_STORAGE_CLASS_LENGTH	8	X'00000008'
MQ_SUB_IDENTITY_LENGTH	128	X'00000080'
MQ_SUB_POINT_LENGTH	128	X'00000080'
MQ_SUITE_B_128_BIT	2	X'00000002'
MQ_SUITE_B_192_BIT	4	X'00000004'
MQ_SUITE_B_NONE	1	X'00000001'
MQ_SUITE_B_NO_DISPONIBLE	0	X'00000000'
MQ_TCP_LONGITUD_DEL_NOMBRE	8	X'00000008'
MQ_TIME_LENGTH	8	X'00000008'
MQ_TOPIC_DESC_LENGTH	64	X'00000040'
MQ_TOPIC_NAME_LENGTH	48	X'00000030'

Tabla 1. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQ_TOPIC_STR_LENGTH	10240	X'00002800'
MQ_TOTAL_EXIT_DATA_LENGTH	999	X'000003E7'
MQ_TOTAL_EXIT_NAME_LENGTH	999	X'000003E7'
MQ_TP_NAME_LENGTH	64	X'00000040'
MQ_TPIPE_NAME_LENGTH	8	X'00000008'
MQ_TRAN_INSTANCE_ID_LENGTH	16	X'00000010'
MQ_TRANSACTION_ID_LENGTH	4	X'00000004'
MQ_TRIGGER_DATA_LENGTH	64	X'00000040'
MQ_TRIGGER_PROGRAM_NAME_LENGTH	8	X'00000008'
MQ_TRIGGER_TERM_ID_LENGTH	4	X'00000004'
MQ_TRIGGER_TRANS_ID_LENGTH	4	X'00000004'
MQ_USER_ID_LENGTH	12	X'0000000C'
MQ_VERSION_LENGTH	8	X'00000008'
MQ_XCF_GROUP_NAME_LENGTH	8	X'00000008'
MQ_XCF_LONGITUD_DEL_NOMBRE_DEL_MIEMBRO	16	X'00000010'

MQ_* (Longitudes de cadena del formato de comando)

Tabla 2. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQ_ARCHIVE_PFX_LENGTH	36	X'00000024'
MQ_ARCHIVE_UNIT_LENGTH	8	X'00000008'
MQ_ASID_LENGTH	4	X'00000004'
Longitud del nombre del perfil de autenticación MQ	48	X'00000030'
MQ_CF_LEID_LONGITUD	12	X'0000000C'
MQ_COMMAND_MQSC_LENGTH	32768	X'00008000'
MQ_DATA_SET_NAME_LENGTH	44	X'0000002C'
MQ_DB2_NAME_LENGTH	4	X'00000004'
MQ_DSG_NAME_LENGTH	8	X'00000008'
MQ_ENTITY_NAME_LENGTH	1024	X'00000400'
MQ_ENV_INFO_LENGTH	96	X'00000060'
MQ_IP_ADDRESS_LENGTH	48	X'00000030'
MQ_LOG_CORREL_ID_LENGTH	8	X'00000008'
MQ_LOG_EXTENT_NAME_LENGTH	24	X'00000018'
MQ_LOG_PATH_LENGTH	1024	X'00000400'
MQ_LRSN_LONGITUD	12	X'0000000C'
MQ_ORIGIN_NAME_LENGTH	8	X'00000008'
MQ_PSB_NOMBRE_LONGITUD	8	X'00000008'
MQ_PST_ID_LONGITUD	8	X'00000008'
MQ_Q_MGR_CPF_LONGITUD	4	X'00000004'
MQ_RESPUESTA_ID_LONGITUD	24	X'00000018'

Tabla 2. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQ_RBA_LONGITUD	16	X'00000010'
MQ_SECURITY_PROFILE_LENGTH	40	X'00000028'
MQ_SERVICE_COMPONENT_LENGTH	48	X'00000030'
MQ_SUB_NAME_LENGTH	10240	X'00002800'
MQ_SYSP_SERVICE_LENGTH	32	X'00000020'
MQ_SYSTEM_NAME_LENGTH	8	X'00000008'
MQ_TASK_NUMBER_LENGTH	8	X'00000008'
MQ_TPIPE_PFX_LONGITUD	4	X'00000004'
MQ_UOW_ID_LONGITUD	256	X'00000100'
MQ_USER_DATA_LENGTH	10240	X'00002800'
MQ_VOLSER_LENGTH	6	X'00000006'

MQACH_* (estructura de encabezado del área de cadena de salida de API)

Tabla 3. Estructuras de constantes	
Nombre	Estructura
MQACH_STRUC_ID	"ACH¬"
MQACH_STRUC_ID_ARRAY	'A', 'C', 'H', '¬'

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

Tabla 4. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQACH_VERSION_1	1	X'00000001'
MQACH_VERSIÓN_ACTUAL	1	X'00000001'
MQACH_LENGTH_1	(value differs by platform or version)	(value differs by platform or version)
MQACH_LONGITUD_ACTUAL	(value differs by platform or version)	(value differs by platform or version)

MQACT_* (Token contable)

Tabla 5. Nombres y valores constantes	
Nombre	Valor
MQACT_NONE	X'00...00' (32 nulos)
MQACT_NINGUNA_MATRIZ	'\0', '\0', ... (32 nulos)

MQACT_* (Opciones de acción del formato de comando)

Tabla 6. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQACT_FORCE_REMOVE	1	X'00000001'
MQACT_ADVANCE_LOG	2	X'00000002'
MQACT_RECOPIRAR_ESTADÍSTICAS	3	X'00000003'
MQACT_PUBSUB	4	X'00000004'

MQACTP_* (Acción)

Tabla 7. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQACTP_NEW	0	X'00000000'
MQACTP_FORWARD	1	X'00000001'
MQACTP_REPLY	2	X'00000002'
MQACTP_REPORT	3	X'00000003'

MQACTT_* (Tipos de tokens contables)

Tabla 8. Valores de las constantes	
Nombre	Valor hexadecimal
MQACTT_DESCONOCIDO	X'00'
MQACTT_CICS_LUOW_ID	X'01'
MQACTT_OS2_DEFAULT	X'04'
MQACTT_DOS_PREDEFINIDO	X'05'
MQACTT_UNIX_ID_NUMÉRICO	X'06'
MQACTT_OS400_ACCOUNT_TOKEN	X'08'
MQACTT_WINDOWS_DEFAULT	X'09'
MQACTT_NT_ID_DE_SEGURIDAD	X'0B'
MQACTT_USUARIO	X'19'

MQADOPT_* (Adoptar nuevas comprobaciones MCA y adoptar nuevos tipos MCA)

Adopte los nuevos controles MCA

Tabla 9. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQADOPT_CHECK_NONE	0	X'00000000'
MQADOPT_COMPROBAR_TODO	1	X'00000001'
MQADOPT_COMPROBAR_NOMBRE_GESTOR_COLA	2	X'00000002'
MQADOPT_COMPROBAR_DIRECCIÓN_RED	4	X'00000004'

Adoptar nuevos tipos de MCA

Tabla 10. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQADOPT_TYPE_NO	0	X'00000000'
MQADOPT_TYPE_ALL	1	X'00000001'
MQADOPT_TYPE_SVR	2	X'00000002'
MQADOPT_TYPE_SDR	4	X'00000004'
MQADOPT_TYPE_RCVR	8	X'00000008'
MQADOPT_TYPE_CLUSRCVR	16	X'00000010'

MQAIR_* (Estructura del registro de información de autenticación)

Tabla 11. Estructuras de constantes	
Nombre	Estructura
MQAIR_STRUC_ID	"AIR¬"
MQAIR_STRUC_ID_ARRAY	'A','I','R','¬'

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

Tabla 12. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQAIR_VERSION_1	1	X'00000001'
MQAIR_VERSION_2	2	X'00000002'
MQAIR_VERSION_ACTUAL	2	X'00000002'

MQAIT_* (Tipo de información de autenticación)

Tabla 13. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQAIT_TODOS	0	X'00000000'
MQAIT_CRL_LDAP	1	X'00000001'
MQAIT_OCSP	2	X'00000002'
MQAIT_IDPW_SISTEMA_OPERATIVO	3	X'00000003'
MQAIT_IDPW_LDAP	4	X'00000004'

MQAS_* (Valores de estado asíncronos en formato de comando)

Tabla 14. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQAS_NINGUNO	0	X'00000000'
MQAS_INICIADO	1	X'00000001'
MQAS_INICIO_ESPERA	2	X'00000002'
MQAS_DETENIDO	3	X'00000003'
MQAS_SUSPENDIDO	4	X'00000004'
MQAS_SUSPENDIDO_TEMPORALMENTE	5	X'00000005'
MQAS_ACTIVO	6	X'00000006'
MQAS_INACTIVO	7	X'00000007'

MQAT_* (Tipos de aplicación de puesta)

Tabla 15. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQAT_DESCONOCIDO	-1	X'FFFFFFFF'
MQAT_SIN_CONTEXTO	0	X'00000000'
MQAT_CICS	1	X'00000001'
MQAT_MVS	2	X'00000002'
MQAT_OS390	2	X'00000002'

Tabla 15. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQAT_ZOS	2	X'00000002'
MQAT_IMS	3	X'00000003'
MQAT_OS2	4	X'00000004'
MQAT_DOS	5	X'00000005'
MQAT_AIX	6	X'00000006'
MQAT_UNIX	6	X'00000006'
MQAT_QMGR	7	X'00000007'
MQAT_OS400	8	X'00000008'
MQAT_VENTANAS	9	X'00000009'
MQAT_CICS_VSE	10	X'0000000A'
MQAT_WINDOWS_NT	11	X'0000000B'
MQAT_VMS	12	X'0000000C'
MQAT_GUARDIÁN	13	X'0000000D'
MQAT_NSK	13	X'0000000D'
MQAT_VOS	14	X'0000000E'
MQAT_OPEN_TP1	15	X'0000000F'
MQAT_VM	18	X'00000012'
MQAT_IMS_PUENTE	19	X'00000013'
MQAT_XCF	20	X'00000014'
MQAT_CICS_PUENTE	21	X'00000015'
MQAT_NOTAS_AGENTE	22	X'00000016'
MQAT_TPF	23	X'00000017'
MQAT_USUARIO	25	X'00000019'
MQAT_CORREDOR	26	X'0000001A'
MQAT_QMGR_PUBLISH	26	X'0000001A'
MQAT_JAVA	28	X'0000001C'
MQAT_DQM	29	X'0000001D'
MQAT_INICIADOR_DEL_CANAL	30	X'0000001E'
MQAT_WLM	31	X'0000001F'
MQAT_LOTE	32	X'00000020'
MQAT_RRS_LOTE	33	X'00000021'
MQAT_SIB	34	X'00000022'
MQAT_DEFAULT	(value differs by platform or version)	(value differs by platform or version)
MQAT_USUARIO_PRIMERO	65536	X'00010000'
MQAT_USUARIO_ÚLTIMO	999999999	X'3B9AC9FF'



MQAUSC_* (Valores de ámbito de autoridad del formato de comando)

Tabla 16. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQAUSC_FALLOS	0	X'00000000'
MQAUSC_TODAS_LAS_CONEXIONES	1	X'00000001'
MQAUSC_TODAS_LAS_COMPROBACIONES	2	X'00000002'

MQAUTH_* (Valores de autoridad del formato de comando)

Tabla 17. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQAUTH_NONE	0	X'00000000'
MQAUTH_ALT_USER_AUTHORITY	1	X'00000001'
MQAUTH_EXPLORAR	2	X'00000002'
MQAUTH_CHANGE	3	X'00000003'
MQAUTH_CLEAR	4	X'00000004'
MQAUTH_CONNECT	5	X'00000005'
MQAUTH_CREATE	6	X'00000006'
MQAUTH_DELETE	7	X'00000007'
MQAUTH_DISPLAY	8	X'00000008'
MQAUTH_ENTRADA	9	X'00000009'
MQAUTH_CONSULTAR	10	X'0000000A'
MQAUTH_SALIDA	11	X'0000000B'
MQAUTH_PASS_ALL_CONTEXT	12	X'0000000C'
MQAUTH_PASS_IDENTITY_CONTEXT	13	X'0000000D'
MQAUTH_SET	14	X'0000000E'
MQAUTH_SET_ALL_CONTEXT	15	X'0000000F'
MQAUTH_SET_IDENTITY_CONTEXT	16	X'00000010'
MQAUTH_CONTROL	17	X'00000011'
MQAUTH_CONTROL_EXTENDIDO	18	X'00000012'
MQAUTH_PUBLISH	19	X'00000013'
MQAUTH_SUSCRIBIRSE	20	X'00000014'
MQAUTH_RESUME	21	X'00000015'
MQAUTH_SISTEMA	22	X'00000016'

MQAUTHOPT_* (Opciones de autoridad del formato de comando)

Tabla 18. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQAUTHOPT_ACUMULATIVO	256	X'00000100'
MQAUTHOPT_ENTIDAD_EXPLÍCITA	1	X'00000001'
MQAUTHOPT_ENTIDAD_CONJUNTO	2	X'00000002'

Tabla 18. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQAUTHOPT_NAME_ALL_MATCHING	32	X'00000020'
MQAUTHOPT_NAME_AS_WILDCARD	64	X'00000040'
MQAUTHOPT_NAME_EXPLÍCITO	16	X'00000010'

MQAXC_* (estructura de contexto de salida de API)

Tabla 19. Estructuras de constantes	
Nombre	Estructura
MQAXC_STRUC_ID	"AXC↵"
MQAXC_STRUC_ID_ARRAY	'A','X','C','↵'

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

Tabla 20. Valores de constantes		
Nombre	Valor decimal	Valor hexadecimal
MQAXC_VERSION_1	1	X'00000001'
MQAXC_VERSION_ACTUAL	1	X'00000001'

MQAXP_* (estructura de parámetros de salida de API)

Tabla 21. Estructuras de constantes	
Nombre	Estructura
MQAXP_ID_ESTRUCTURA	"AXP↵"
MQAXP_STRUC_ID_ARRAY	'A','X','P','↵'

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

Tabla 22. Valores de constantes		
Nombre	Valor decimal	Valor hexadecimal
MQAXP_VERSION_1	1	X'00000001'
MQAXP_VERSION_2	2	X'00000002'
VERSION_ACTUAL_MQAXP	2	X'00000002'

MQBA_* (Selektors de atributos de bytes)

Tabla 23. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQBA_PRIMERO	6001	X'00001771'
MQBA_ÚLTIMO	8000	X'00001F40'

MQBACF_* (Tipos de parámetros de bytes de formato de comando)

Tabla 24. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQBACF_PRIMERO	7001	X'00001B59'
MQBACF_EVENTO_CONTABILIDAD_TOKEN	7001	X'00001B59'

Tabla 24. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQBACF_EVENTO_ID_SEGURIDAD	7002	X'00001B5A'
MQBACF_RESPUESTA_CONJUNTO	7003	X'00001B5B'
MQBACF_RESPUESTA_ID	7004	X'00001B5C'
MQBACF_EXTERNO_UOW_ID	7005	X'00001B5D'
MQBACF_CONNECTION_ID	7006	X'00001B5E'
MQBACF_ID_CONEXIÓN_GENÉRICA	7007	X'00001B5F'
MQBACF_ORIGEN_UOW_ID	7008	X'00001B60'
MQBACF_Q_MGR_UOW_ID	7009	X'00001B61'
MQBACF_ACCOUNTING_TOKEN	7010	X'00001B62'
MQBACF_CORREL_ID	7011	X'00001B63'
MQBACF_GROUP_ID	7012	X'00001B64'
MQBACF_MSG_ID	7013	X'00001B65'
MQBACF_CF_LEID	7014	X'00001B66'
MQBACF_DESTINO_CORRELACIÓN_ID	7015	X'00001B67'
MQBACF_SUB_ID	7016	X'00001B68'
MQBACF_ÚLTIMO_USADO	7016	X'00001B68'

MQBL_* (Longitud del búfer para mqAddString y mqSetString)

Tabla 25. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQBL_TERMINADO_CON_NULL	-1	X'FFFFFFFF'

MQBMHO_* (Opciones y estructura del búfer al identificador de mensaje)

Estructura de opciones del búfer al identificador de mensaje

Tabla 26. Estructuras de constantes	
Nombre	Estructura
MQBMHO_STRUC_ID	"BMHO"
MQBMHO_STRUC_ID_ARRAY	'B', 'M', 'H', 'O'

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

Tabla 27. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQBMHO_VERSION_1	1	X'00000001'
MQBMHO_VERSIÓN_ACTUAL	1	X'00000001'

Opciones del controlador de mensajes del búfer

Tabla 28. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQBMHO_NINGUNO	0	X'00000000'

Tabla 28. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQBMHO_ELIMINAR_PROPIEDADES	1	X'00000001'

MQBND_* (Enlaces predeterminados)

Tabla 29. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQBND_BIND_ON_OPEN	0	X'00000000'
MQBND_BIND_NO_FIJO	1	X'00000001'
MQBND_BIND_ON_GROUP	2	X'00000002'

MQBO_* (Inicio de opciones y estructura)

Comience la estructura de opciones

Tabla 30. Estructuras de constantes	
Nombre	Estructura
MQBO_STRUC_ID	"B0¬¬"
MQBO_STRUC_ID_ARRAY	'B','0','¬','¬'

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

Tabla 31. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQBO_VERSION_1	1	X'00000001'
MQBO_VERSIÓN_ACTUAL	1	X'00000001'

Opciones de inicio

Tabla 32. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQBO_NONE	0	X'00000000'

MQBT_* (Formato de comando Tipos de puente)

Tabla 33. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQBT_OTMA	1	X'00000001'

MQCA_* (Selektors de atributos de caracteres)

Tabla 34. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCA_ADMIN_TOPIC_NAME	2105	X'00000839'
MQCA_ALTERATION_DATE	2027	X'000007EB'
MQCA_ALTERATION_TIME	2028	X'000007EC'
MQCA_APPL_ID	2001	X'000007D1'

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

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

Tabla 34. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQCA_STORAGE_CLASS_DESC	2042	X'000007FA'
MQCA_SYSTEM_LOG_Q_NAME	2065	X'00000811'
MQCA_TCP_NAME	2075	X'0000081B'
MQCA_TOPIC_DESC	2093	X'0000082D'
MQCA_TOPIC_NAME	2092	X'0000082C'
MQCA_TOPIC_STRING_FILTER	2108	X'0000083C'
MQCA_TOPIC_STRING	2094	X'0000082E'
MQCA_TPIPE_NAME	2085	X'00000825'
MQCA_TRIGGER_CHANNEL_NAME	2064	X'00000810'
MQCA_TRIGGER_DATA	2023	X'000007E7'
MQCA_TRIGGER_PROGRAM_NAME	2062	X'0000080E'
MQCA_TRIGGER_TERM_ID	2063	X'0000080F'
MQCA_TRIGGER_TRANS_ID	2061	X'0000080D'
MQCA_USER_DATA	2021	X'000007E5'
MQCA_USER_LIST	4000	X'00000FA0'
MQCA_VERSION	2120	X'00000848'
MQCA_XCF_GROUP_NAME	2043	X'000007FB'
MQCA_XCF_MEMBER_NAME	2044	X'000007FC'
MQCA_XMIT_Q_NAME	2024	X'000007E8'

MQCACF_* (Tipos de parámetros de caracteres de formato de comando)

Tabla 35. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCACF_PRIMERO	3001	X'00000BB9'
MQCACF_FROM_Q_NAME	3001	X'00000BB9'
MQCACF_TO_Q_NAME	3002	X'00000BBA'
MQCACF_DESDE_NOMBRE_PROCESO	3003	X'00000BBB'
MQCACF_TO_PROCESS_NAME	3004	X'00000BBC'
MQCACF_A_PARTIR_DEL_NOMBRE_DE_LA_LISTA_DE_NOMBRES	3005	X'00000BBD'
MQCACF_A_LISTA_DE_NOMBRES	3006	X'00000BBE'
MQCACF_FROM_CHANNEL_NAME	3007	X'00000BBF'
MQCACF_TO_NOMBRE_DEL_CANAL	3008	X'00000BC0'
MQCACF_FROM_AUTH_INFO_NAME	3009	X'00000BC1'
MQCACF_TO_AUTH_INFO_NAME	3010	X'00000BC2'
MQCACF_Q_NOMBRES	3011	X'00000BC3'
MQCACF_NOMBRES_DE_PROCESOS	3012	X'00000BC4'
MQCACF_LISTA_DE_NOMBRES	3013	X'00000BC5'
MQCACF_TEXTO_DE_ESCAPE	3014	X'00000BC6'
MQCACF_LOCAL_Q_NAMES	3015	X'00000BC7'

Tabla 35. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQCACF_MODELO_Q_NOMBRES	3016	X'00000BC8'
MQCACF_ALIAS_Q_NOMBRES	3017	X'00000BC9'
MQCACF_REMOTO_Q_NOMBRES	3018	X'00000BCA'
MQCACF_NOMBRES_DE_CANALES_DEL_REMITENTE	3019	X'00000BCB'
Nombres de canales del servidor MQCACF	3020	X'00000BCC'
MQCACF_SOLICITANTE_NOMBRES_DE_CANALES	3021	X'00000BCD'
MQCACF_RECEPTOR_NOMBRES_CANALES	3022	X'00000BCE'
MQCACF_OBJECT_Q_MGR_NAME	3023	X'00000BCF'
MQCACF_APPL_NAME	3024	X'00000BD0'
MQCACF_USER_IDENTIFIER	3025	X'00000BD1'
MQCACF_AUX_ERROR_DATA_STR_1	3026	X'00000BD2'
MQCACF_AUX_ERROR_DATA_STR_2	3027	X'00000BD3'
MQCACF_AUX_ERROR_DATA_STR_3	3028	X'00000BD4'
MQCACF_BRIDGE_NAME	3029	X'00000BD5'
MQCACF_STREAM_NAME	3030	X'00000BD6'
MQCACF_TEMA	3031	X'00000BD7'
MQCACF_PARENT_Q_MGR_NAME	3032	X'00000BD8'
MQCACF_CORREL_ID	3033	X'00000BD9'
MQCACF_FECHA_PUBLICACIÓN	3034	X'00000BDA'
MQCACF_STRING_DATA	3035	X'00000BDB'
MQCACF_NOMBRE_DE_FLUJO_COMPATIBLE	3036	X'00000BDC'
MQCACF_REG_TEMA	3037	X'00000BDD'
MQCACF_REG_TIME	3038	X'00000BDE'
MQCACF_REG_USER_ID	3039	X'00000BDF'
MQCACF_NIÑO_Q_MGR_NOMBRE	3040	X'00000BE0'
MQCACF_REG_STREAM_NAME	3041	X'00000BE1'
MQCACF_REG_Q_MGR_NAME	3042	X'00000BE2'
MQCACF_REG_Q_NOMBRE	3043	X'00000BE3'
MQCACF_REG_CORREL_ID	3044	X'00000BE4'
MQCACF_EVENTO_ID_USUARIO	3045	X'00000BE5'
MQCACF_OBJECT_NAME	3046	X'00000BE6'
MQCACF_EVENTO_Q_GESTOR	3047	X'00000BE7'
MQCACF_AUTH_INFO_NOMBRES	3048	X'00000BE8'
MQCACF_EVENTO_APLICACIÓN_IDENTIDAD	3049	X'00000BE9'
MQCACF_EVENTO_APLICACIÓN_NOMBRE	3050	X'00000BEA'
MQCACF_EVENTO_SOLICITUD_ORIGEN	3051	X'00000BEB'
MQCACF_NOMBRE_DE_LA_SUSCRIPCIÓN	3052	X'00000BEC'
MQCACF_REG_SUB_NAME	3053	X'00000BED'
MQCACF_SUSCRIPCIÓN_IDENTIDAD	3054	X'00000BEE'

Tabla 35. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQCACF_REG_SUB_IDENTIDAD	3055	X'00000BEF'
MQCACF_SUSCRIPCIÓN_DATOS_USUARIO	3056	X'00000BF0'
MQCACF_REG_SUB_USER_DATA	3057	X'00000BF1'
MQCACF_APPL_TAG	3058	X'00000BF2'
MQCACF_CONJUNTO_DE_DATOS_NOMBRE	3059	X'00000BF3'
MQCACF_UOW_FECHA_DE_INICIO	3060	X'00000BF4'
MQCACF_UOW_HORA_DE_INICIO	3061	X'00000BF5'
MQCACF_UOW_LOG_FECHA_INICIO	3062	X'00000BF6'
MQCACF_UOW_HORA_INICIO_REGISTRO	3063	X'00000BF7'
MQCACF_UOW_LOG_EXTENT_NAME	3064	X'00000BF8'
MQCACF_NOMBRES_DE_ENTIDADES_PRINCIPALES	3065	X'00000BF9'
MQCACF_GRUPO_NOMBRES_ENTIDADES	3066	X'00000BFA'
MQCACF_AUTH_PROFILE_NAME	3067	X'00000BFB'
MQCACF_NOMBRE_ENTIDAD	3068	X'00000BFC'
MQCACF_COMPONENTE_SERVICIO	3069	X'00000BFD'
MQCACF_RESPUESTA_Q_MGR_NOMBRE	3070	X'00000BFE'
MQCACF_CURRENT_LOG_EXTENT_NAME	3071	X'00000BFF'
MQCACF_REINICIAR_REGISTRO_EXTENSION_NOMBRE	3072	X'00000C00'
MQCACF_MEDIA_LOG_EXTENT_NAME	3073	X'00000C01'
MQCACF_LOG_PATH	3074	X'00000C02'
MQCACF_COMANDO_MQSC	3075	X'00000C03'
MQCACF_Q_MGR_CPF	3076	X'00000C04'
MQCACF_REGISTRO_DE_USO_RBA	3078	X'00000C06'
MQCACF_REGISTRO_DE_USO_LRSN	3079	X'00000C07'
MQCACF_COMANDO_ÁMBITO	3080	X'00000C08'
MQCACF_ASID	3081	X'00000C09'
MQCACF_PSB_NOMBRE	3082	X'00000C0A'
MQCACF_PST_ID	3083	X'00000C0B'
NÚMERO DE TAREA MQCACF	3084	X'00000C0C'
MQCACF_ID_TRANSACCIÓN	3085	X'00000C0D'
MQCACF_Q_MGR_UOW_ID	3086	X'00000C0E'
MQCACF_ORIGEN_NOMBRE	3088	X'00000C10'
MQCACF_INFO_ENTORNO	3089	X'00000C11'
MQCACF_PERFIL_DE_SEGURIDAD	3090	X'00000C12'
MQCACF_FECHA_DE_CONFIGURACIÓN	3091	X'00000C13'
MQCACF_CONFIGURACIÓN_TIEMPO	3092	X'00000C14'
MQCACF_A_PARTIR_DEL_NOMBRE_DE_LA_ESTRUCTURA_CF	3093	X'00000C15'
MQCACF_TO_CF_STRUC_NAME	3094	X'00000C16'
MQCACF_CF_STRUC_NAMES	3095	X'00000C17'

Tabla 35. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQCACF_FECHA_FALLO	3096	X'00000C18'
MQCACF_FAIL_TIME	3097	X'00000C19'
MQCACF_FECHA_DE_COPIA_DE_RESERVA	3098	X'00000C1A'
MQCACF_HORA_DE_COPIA_DE_RESERVA	3099	X'00000C1B'
MQCACF_NOMBRE_SISTEMA	3100	X'00000C1C'
MQCACF_CF_STRUC_BACKUP_START	3101	X'00000C1D'
MQCACF_CF_STRUC_BACKUP_FINAL	3102	X'00000C1E'
MQCACF_CF_ESTRUCTURA_REGISTRO_Q_MGRS	3103	X'00000C1F'
MQCACF_DESDE_CLASE_DE_ALMACENAMIENTO	3104	X'00000C20'
MQCACF_A_CLASE_DE_ALMACENAMIENTO	3105	X'00000C21'
MQCACF_CLASES_DE_ALMACENAMIENTO_NOMBRES	3106	X'00000C22'
MQCACF_DSG_NOMBRE	3108	X'00000C24'
MQCACF_DB2_NAME	3109	X'00000C25'
MQCACF_SYSP_CMD_ID_USUARIO	3110	X'00000C26'
MQCACF_SYSP_OTMA_GRUPO	3111	X'00000C27'
MQCACF_SYSP_OTMA_MIEMBRO	3112	X'00000C28'
MQCACF_SYSP_OTMA_DRU_SALIDA	3113	X'00000C29'
MQCACF_SYSP_OTMA_TPIPE_PFX	3114	X'00000C2A'
MQCACF_SYSP_ARCHIVE_PFX1	3115	X'00000C2B'
MQCACF_SYSP_ARCHIVE_UNIT1	3116	X'00000C2C'
MQCACF_SYSP_LOG_CORREL_ID	3117	X'00000C2D'
MQCACF_SYSP_UNIT_VOLSER	3118	X'00000C2E'
MQCACF_SYSP_Q_MGR_TIME	3119	X'00000C2F'
MQCACF_SYSP_Q_MGR_FECHA	3120	X'00000C30'
MQCACF_SYSP_Q_MGR_RBA	3121	X'00000C31'
MQCACF_SYSP_LOG_RBA	3122	X'00000C32'
MQCACF_SYSP_SERVICIO	3123	X'00000C33'
MQCACF_FROM_LISTENER_NAME	3124	X'00000C34'
MQCACF_A_NOMBRE_DEL_OYENTE	3125	X'00000C35'
MQCACF_A_PARTIR_DEL_NOMBRE_DEL_SERVICIO	3126	X'00000C36'
MQCACF_TO_SERVICE_NAME	3127	X'00000C37'
MQCACF_ÚLTIMA_FECHA_DE_ENTREGA	3128	X'00000C38'
MQCACF_ÚLTIMA_HORA_DE_ENTREGA	3129	X'00000C39'
MQCACF_ÚLTIMA_FECHA_DE_OBTENCIÓN	3130	X'00000C3A'
MQCACF_ÚLTIMA_OBTENCIÓN_DE_TIEMPO	3131	X'00000C3B'
MQCACF_FECHA_OPERACIÓN	3132	X'00000C3C'
MQCACF_TIEMPO_DE_OPERACIÓN	3133	X'00000C3D'
MQCACF_DESCRIPCIÓN_ACTIVIDAD	3134	X'00000C3E'
MQCACF_APPL_IDENTIDAD_DATOS	3135	X'00000C3F'

Tabla 35. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQCACF_APPL_ORIGEN_DATOS	3136	X'00000C40'
MQCACF_PUT_DATE	3137	X'00000C41'
MQCACF_PUT_TIME	3138	X'00000C42'
MQCACF_REPLY_TO_Q	3139	X'00000C43'
MQCACF_REPLY_TO_Q_MGR	3140	X'00000C44'
MQCACF_RESOLVED_Q_NAME	3141	X'00000C45'
MQCACF_STRUC_ID	3142	X'00000C46'
MQCACF_VALOR_NOMBRE	3143	X'00000C47'
MQCACF_FECHA_DE_INICIO_DEL_SERVICIO	3144	X'00000C48'
MQCACF_HORA_DE_INICIO_DEL_SERVICIO	3145	X'00000C49'
MQCACF_SYSP_OFFLINE_RBA	3146	X'00000C4A'
MQCACF_SYSP_ARCHIVE_PFX2	3147	X'00000C4B'
MQCACF_SYSP_ARCHIVE_UNIT2	3148	X'00000C4C'
MQCACF_A_NOMBRE_DEL_TEMA	3149	X'00000C4D'
MQCACF_DESDE_EL_NOMBRE_DEL_TEMA	3150	X'00000C4E'
MQCACF_NOMBRES_DE_TEMAS	3151	X'00000C4F'
MQCACF_SUB_NAME	3152	X'00000C50'
MQCACF_DESTINO_Q_GESTOR	3153	X'00000C51'
MQCACF_DESTINO	3154	X'00000C52'
MQCACF_SUB_ID_USUARIO	3156	X'00000C54'
MQCACF_SUB_USER_DATA	3159	X'00000C57'
MQCACF_SUB_SELECTOR	3160	X'00000C58'
MQCACF_ÚLTIMA_FECHA_PUBLICACIÓN	3161	X'00000C59'
MQCACF_ÚLTIMA_PUBLICACIÓN	3162	X'00000C5A'
MQCACF_FROM_SUB_NAME	3163	X'00000C5B'
MQCACF_TO_SUB_NAME	3164	X'00000C5C'
MQCACF_ÚLTIMA_HORA_DEL_MENSAJE	3167	X'00000C5F'
MQCACF_ÚLTIMA_FECHA_MENSAJE	3168	X'00000C60'
Punto de suscripción MQCACF	3169	X'00000C61'
MQCACF_FILTRO	3170	X'00000C62'
MQCACF_NINGUNO	3171	X'00000C63'
MQCACF_ADMIN_NOMBRES_DE_TEMAS	3172	X'00000C64'
MQCACF_ENRUTAMIENTO_HUELLA_DACTILAR	3173	X'00000C65'
MQCACF_APPL_DESC	3174	X'00000C66'
MQCACF_Q_MGR_FECHA_INICIO	3175	X'00000C67'
MQCACF_Q_MGR_HORA_DE_INICIO	3176	X'00000C68'
MQCACF_FROM_COMM_INFO_NAME	3177	X'00000C69'
MQCACF_TO_COMM_INFO_NAME	3178	X'00000C6A'
MQCACF_CF_OFFLOAD_SIZE1	3179	X'00000C6B'

Tabla 35. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQCACF_CF_OFFLOAD_SIZE2	3180	X'00000C6C'
MQCACF_CF_OFFLOAD_SIZE3	3181	X'00000C6D'
MQCACF_CF_SMDS_NOMBRE_GENÉRICO	3182	X'00000C6E'
MQCACF_CF_SMDS	3183	X'00000C6F'
MQCACF_FECHA_DE_RECUPERACIÓN	3184	X'00000C70'
MQCACF_TIEMPO_DE_RECUPERACIÓN	3185	X'00000C71'
MQCACF_CF_SMDSCONN	3186	X'00000C72'
MQCACF_CF_STRUC_NAME	3187	X'00000C73'
MQCACF_ALTERNATE_USERID	3188	X'00000C74'
MQCACF_CHAR_ATTRS	3189	X'00000C75'
MQCACF_DYNAMIC_Q_NAME	3190	X'00000C76'
MQCACF_HOST_NAME	3191	X'00000C77'
MQCACF_MQCB_NAME	3192	X'00000C78'
MQCACF_OBJECT_STRING	3193	X'00000C79'
MQCACF_RESOLVED_LOCAL_Q_MGR	3194	X'00000C7A'
MQCACF_RESOLVED_LOCAL_Q_NAME	3195	X'00000C7B'
MQCACF_RESOLVED_OBJECT_STRING	3196	X'00000C7C'
MQCACF_RESOLVED_Q_MGR	3197	X'00000C7D'
MQCACF_SELECTION_STRING	3198	X'00000C7E'
MQCACF_XA_INFO	3199	X'00000C7F'
MQCACF_APPL_FUNCTION	3200	X'00000C80'
MQCACF_XQH_REMOTE_Q_NAME	3201	X'00000C81'
MQCACF_XQH_REMOTE_Q_MGR	3202	X'00000C82'
MQCACF_XQH_PUT_TIME	3203	X'00000C83'
MQCACF_XQH_PUT_DATE	3204	X'00000C84'
MQCACF_EXCL_MENSAJES_OPERADOR	3205	X'00000C85'
MQCACF_CSP_IDENTIFICADOR_DE_USUARIO	3206	X'00000C86'
MQCACF_AMQP_ID_CLIENTE	3207	X'00000C87'
MQCACF_ARCHIVO_REGISTRO_EXTENSION_NOMBRE	3208	X'00000C88'
MQCACF_APPL_FECHA_INMOVILIZABLE	3209	X'00000C89'
MQCACF_APPL_INMUEBLE_TIEMPO	3210	X'00000C8A'
MQCACF_NHA_INSTANCE_NAME	3211	X'00000C8B'
MQCACF_EVENTO_DUPLICADO_DESDE	3232	X'00000CA0'
MQCACF_ÚLTIMO_USADO	3232	X'00000CA0'

MQCACH_* (Tipos de parámetros de canal de caracteres de formato de comando)

Tabla 36. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCACH_PRIMERO	3501	X'00000DAD'

Tabla 36. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQCACH_CHANNEL_NAME	3501	X'00000DAD'
MQCACH_DESC	3502	X'00000DAE'
MQCACH_MODO_NOMBRE	3503	X'00000DAF'
MQCACH_TP_NOMBRE	3504	X'00000DB0'
MQCACH_XMIT_Q_NAME	3505	X'00000DB1'
MQCACH_CONNECTION_NAME	3506	X'00000DB2'
MQCACH_MCA_NOMBRE	3507	X'00000DB3'
MQCACH_SEC_EXIT_NAME	3508	X'00000DB4'
MQCACH_MSG_EXIT_NAME	3509	X'00000DB5'
MQCACH_ENVIAR_NOMBRE_SALIDA	3510	X'00000DB6'
MQCACH_RCV_EXIT_NAME	3511	X'00000DB7'
NOMBRES_DE_CANALES_MQCACH	3512	X'00000DB8'
MQCACH_SEC_SALIDA_DATOS_USUARIO	3513	X'00000DB9'
MQCACH_MSG_SALIDA_DATOS_USUARIO	3514	X'00000DBA'
MQCACH_ENVIAR_SALIDA_DATOS_USUARIO	3515	X'00000DBB'
MQCACH_RCV_EXIT_DATOS_USUARIO	3516	X'00000DBC'
MQCACH_ID_USUARIO	3517	X'00000DBD'
MQCACH_CONTRASEÑA	3518	X'00000DBE'
MQCACH_DIRECCIÓN_LOCAL	3520	X'00000DC0'
MQCACH_NOMBRE_LOCAL	3521	X'00000DC1'
MQCACH_ÚLTIMA_HORA_MENSAJE	3524	X'00000DC4'
MQCACH_ÚLTIMA_FECHA_MENSAJE	3525	X'00000DC5'
MQCACH_MCA_ID_USUARIO_EE_UU	3527	X'00000DC7'
MQCACH_CANAL_HORA_INICIO	3528	X'00000DC8'
MQCACH_CANAL_FECHA_INICIO	3529	X'00000DC9'
MQCACH_MCA_NOMBRE_TAREA	3530	X'00000DCA'
MQCACH_ÚLTIMO_LUWID	3531	X'00000DCB'
MQCACH_CURRENT_LUWID	3532	X'00000DCC'
MQCACH_FORMAT_NAME	3533	X'00000DCD'
MQCACH_MR_EXIT_NAME	3534	X'00000DCE'
MQCACH_MR_SALIDA_DATOS_USUARIO	3535	X'00000DCF'
MQCACH_SSL_CIFRADO_ESPECÍFICO	3544	X'00000DD8'
MQCACH_SSL_NOMBRE_DEL_PAR	3545	X'00000DD9'
MQCACH_SSL_ETAPA_DE_APRETÓN_DE_MANOS	3546	X'00000DDA'
MQCACH_SSL_NOMBRE_CORT_DE_PAR	3547	X'00000ddb'
MQCACH_REMOTE_APPL_TAG	3548	X'00000DDC'
MQCACH_SSL_CERT_ID_USUARIO	3549	X'00000DDD'
MQCACH_SSL_CERT_ISSUER_NAME	3550	X'00000DDE'
MQCACH_LU_NOMBRE	3551	X'00000DDF'

Tabla 36. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQCACH_DIRECCIÓN_IP	3552	X'00000DE0'
MQCACH_TCP_NAME	3553	X'00000DE1'
MQCACH_LISTENER_NAME	3554	X'00000DE2'
MQCACH_LISTENER_DESC	3555	X'00000DE3'
MQCACH_LISTENER_START_DATE	3556	X'00000DE4'
MQCACH_LISTENER_START_TIME	3557	X'00000DE5'
MQCACH_SSL_KEY_RESET_DATE	3558	X'00000DE6'
MQCACH_SSL_KEY_RESET_TIME	3559	X'00000DE7'
MQCACH_ÚLTIMO_USADO	3559	X'00000DE7'

MQCADSD_* (CICS encabezado de información Descriptores ADS)

Tabla 37. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCADSD_NINGUNO	0	X'00000000'
MQCADSD_ENVIAR	1	X'00000001'
MQCADSD_RECEPCIÓN	16	X'00000010'
MQCADSD_FORMATO DE MENSAJE	256	X'00000100'

MQCAFTY_* (Valores de afinidad de conexión)

Tabla 38. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCAFTY_NINGUNO	0	X'00000000'
MQCAFTY_PREFERIDO	1	X'00000001'

MQCAMO_* (Tipos de parámetros de supervisión de caracteres de formato de comando)

Tabla 39. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCAMO_PRIMERO	2701	X'00000A8D'
MQCAMO_CLOSE_DATE	2701	X'00000A8D'
MQCAMO_CLOSE_TIME	2702	X'00000A8E'
MQCAMO_CONN_DATE	2703	X'00000A8F'
MQCAMO_CONN_TIME	2704	X'00000A90'
MQCAMO_DISC_DATE	2705	X'00000A91'
MQCAMO_DISC_TIME	2706	X'00000A92'
MQCAMO_END_DATE	2707	X'00000A93'
MQCAMO_END_TIME	2708	X'00000A94'
MQCAMO_OPEN_DATE	2709	X'00000A95'
MQCAMO_OPEN_TIME	2710	X'00000A96'
MQCAMO_START_DATE	2711	X'00000A97'

Tabla 39. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQCAMO_START_TIME	2712	X'00000A98'
MQCAMO_ÚLTIMO_USADO	2712	X'00000A98'

MQCBC_* (estructura de constantes MQCBC)

Tabla 40. Estructuras de constantes	
Nombre	Estructura
MQCBC_STRUC_ID	"CBC¬"
MQCBC_STRUC_ID_ARRAY	'C','B','C','¬'

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

Tabla 41. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCBC_VERSION_1	1	X'00000001'
MQCBC_VERSION_ACTUAL	1	X'00000001'

MQCBCF_* (Indicadores de constantes MQCBC)

Tabla 42. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCBCF_NINGUNO	0	X'00000000'
MQCBCF_READA_BUFFER_VACÍO	1	X'00000001'

MQCBCT_* (constantes MQCBC tipo Callback)

Tabla 43. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCBCT_INICIAR_LLAMADA	1	X'00000001'
MQCBCT_DETENER_LLAMADA	2	X'00000002'
MQCBCT_REGISTRAR_LLAMADA	3	X'00000003'
MQCBCT_DEREGISTRAR_LLAMADA	4	X'00000004'
MQCBCT_EVENTO_LLAMADA	5	X'00000005'
MQCBCT_MSG_ELIMINADO	6	X'00000006'
MQCBCT_MSG_NO_ELIMINADO	7	X'00000007'

MQCBD_* (estructura de constantes MQCBD)

Tabla 44. Estructuras de constantes	
Nombre	Estructura
MQCBD_ID_STRUCD	"CBD¬"
MQCBD_STRUC_ID_ARRAY	'C','B','D','¬'

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

Tabla 45. Valores de constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCBD_VERSION_1	1	X'00000001'
MQCBD_VERSION_ACTUAL	1	X'00000001'

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

Tabla 46. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCBDO_NINGUNO	0	X'00000000'
MQCBDO_INICIAR_LLAMADA	1	X'00000001'
MQCBDO_DETENER_LLAMADA	4	X'00000004'
MQCBDO_REGISTRAR_LLAMADA	256	X'00000100'
MQCBDO_DEREGISTRAR_LLAMADA	512	X'00000200'
MQCBDO_FAIL_IF QUIESCING	8192	X'00002000'

MQCBO_* (Opciones de creación de bolsas para mqCreateBag)

Tabla 47. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCBO_NINGUNO	0	X'00000000'
MQCBO_USUARIO_BOLSA	0	X'00000000'
MQCBO_ADMIN_BAG	1	X'00000001'
MQCBO_COMMAND_BAG	16	X'00000010'
MQCBO_SISTEMA_BOLSA	32	X'00000020'
MQCBO_GRUPO_BOLSA	64	X'00000040'
MQCBO_LISTA_FORMULARIO_PERMITIDO	2	X'00000002'
MQCBO_LIST_FORM_INHIBITED	0	X'00000000'
MQCBO_REORDER_AS_REQUIRED	4	X'00000004'
MQCBO_NO_REORDENAR	0	X'00000000'
MQCBO_COMPROBAR_SELECCIONADORES	8	X'00000008'
MQCBO_NO_COMPROBAR_SELECTORES	0	X'00000000'

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

Tabla 48. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCBT_MENSAJE_CONSUMIDOR	1	X'00000001'
MQCBT_EVENT_HANDLER	2	X'00000002'

MQCC_* (códigos de finalización)

Tabla 49. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCC_OK	0	X'00000000'

Tabla 49. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQCC_WARNING	1	X'00000001'
MQCC_FAILED	2	X'00000002'
MQCC_DESCONOCIDO	-1	X'FFFFFFFF'

MQCCSI_* (Identificadores de conjuntos de caracteres codificados)

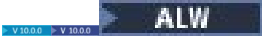
Tabla 50. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCCSI_INDETERMINADO	0	X'00000000'
MQCCSI_POR_DEFAULT	0	X'00000000'
MQCCSI_Q_MGR	0	X'00000000'
MQCCSI_HEREDAR	-2	X'FFFFFFFE'
MQCCSI_INTEGRADO	-1	X'FFFFFFFF'
MQCCSI_APPL	-3	X'FFFFFFFD'

MQCCT_* (cabecera CICS de información Opciones de tareas conversacionales)

Tabla 51. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCCT_SÍ	1	X'00000001'
MQCCT_NO	0	X'00000000'

MQCD_* (Estructura de definición de canal)

Tabla 52. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCD_VERSION_1	1	X'00000001'
MQCD_VERSION_2	2	X'00000002'
MQCD_VERSION_3	3	X'00000003'
MQCD_VERSION_4	4	X'00000004'
MQCD_VERSION_5	5	X'00000005'
MQCD_VERSION_6	6	X'00000006'
MQCD_VERSION_7	7	X'00000007'
MQCD_VERSION_8	8	X'00000008'
MQCD_VERSION_9	9	X'00000009'
MQCD_VERSION_10	10	X'0000000A'
MQCD_VERSION_11	11	X'0000000B'
MQCD_VERSION_12	12	X'0000000C'
 MQCD_VERSION_13	13	X'0000000D'
 MQCD_VERSIÓN_ACTUAL	12	X'0000000C'

Tabla 52. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
 MQCD_VERSION_ACTUAL	13	X'0000000D'
MQCD_LENGTH_4	(value differs by platform or version)	(value differs by platform or version)
MQCD_LENGTH_5	(value differs by platform or version)	(value differs by platform or version)
MQCD_LENGTH_6	(value differs by platform or version)	(value differs by platform or version)
MQCD_LENGTH_7	(value differs by platform or version)	(value differs by platform or version)
MQCD_LENGTH_8	(value differs by platform or version)	(value differs by platform or version)
MQCD_LENGTH_9	(value differs by platform or version)	(value differs by platform or version)
MQCD_LENGTH_10	(value differs by platform or version)	(value differs by platform or version)
MQCD_LENGTH_11	(value differs by platform or version)	(value differs by platform or version)
MQCD_LENGTH_12	(value differs by platform or version)	(value differs by platform or version)
 MQCD_LENGTH_13	(value differs by platform or version)	(value differs by platform or version)
MQCD_LONGITUD_ACTUAL	(value differs by platform or version)	(value differs by platform or version)

MQCDC_* (Conversión de datos de canal)

Tabla 53. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCDC_REMITENTE_CONVERSIÓN	1	X'00000001'
MQCDC_NO_CONVERSIÓN_DEL_REMITENTE	0	X'00000000'

MQCERT_* (Tipo de política de validación de certificados)

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

MQCF_* (Indicadores de capacidad)

Tabla 54. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCF_NINGUNO	0	X'00000000'
MQCF_LISTAS_DISTANCIAS	1	X'00000001'

MQCFAC_* (CICS cabecera de información de instalación)

Tabla 55. Nombres y valores constantes	
Nombre	Valor hexadecimal
MQCFAC_NINGUNO	X'00...00' (8 nulos)
MQCFAC_NINGUNA_MATRIZ	'\0', '\0', ... (8 nulos)

MQCFBF_* (Estructura del parámetro del filtro de cadena de bytes del formato de comando)

Tabla 56. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCFBF_ESTRUCTURA_LONGITUD_FIJA	20	X'00000014'

MQCFBS_* (Estructura de parámetros de cadena de bytes de formato de comando)

Tabla 57. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCFBS_ESTRUCTURA_LONGITUD_FIJA	16	X'00000010'

MQCFC_* (Opciones de control del encabezado del formato de comando)

Tabla 58. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCFC_ÚLTIMO	1	X'00000001'
MQCFC_NO_ÚLTIMO	0	X'00000000'

MQCFGR_* (Estructura de parámetros del grupo de formatos de comando)

Tabla 59. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCFGR_ESTRUCTURA_LONGITUD	16	X'00000010'

MQCFH_* (Estructura del encabezado del formato de comando)

Tabla 60. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCFH_STRUC_LENGTH	36	X'00000024'
MQCFH_VERSION_1	1	X'00000001'
MQCFH_VERSION_2	2	X'00000002'
MQCFH_VERSION_3	3	X'00000003'
MQCFH_VERSION_ACTUAL	3	X'00000003'

MQCFIF_* (Estructura del parámetro del filtro de enteros del formato de comando)

Tabla 61. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCFIF_ESTRUCTURA_LONGITUD	20	X'00000014'

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

Tabla 62. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCFIL_STRUC_LONGITUD_FIJA	16	X'00000010'

MQCFIL64_* (Estructura del parámetro de lista de enteros de 64 bits del formato de comando)

Tabla 63. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCFIL64_STRUC_LENGTH_FIXED	16	X'00000010'

MQCFIN_* (Estructura de parámetros enteros del formato de comando)

Tabla 64. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCFIN_ESTRUCTURA_LONGITUD	16	X'00000010'

MQCFIN64_* (Estructura del parámetro entero de 64 bits del formato de comando)

Tabla 65. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCFIN64_STRUC_LENGTH	24	X'00000018'

MQCFO_* (Formato de comando Actualizar opciones del repositorio y Formato de comando Eliminar opciones de colas)

Formato del comando Actualizar opciones del repositorio

Tabla 66. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCFO_ACTUALIZAR_REPOSITORIO_SÍ	1	X'00000001'
MQCFO_ACTUALIZAR_REPOSITORIO_NO	0	X'00000000'

Formato del comando Eliminar colas Opciones

Tabla 67. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCFO_ELIMINAR_COLAS_SÍ	1	X'00000001'

Tabla 67. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQCFO_REMOVE_QUEUES_NO	0	X'00000000'

MQCFOP_* (Operadores de filtro de formato de comando)

Tabla 68. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCFOP_MENOS	1	X'00000001'
MQCFOP_IGUAL	2	X'00000002'
MQCFOP_MAYOR	4	X'00000004'
MQCFOP_NO_MENOS	6	X'00000006'
MQCFOP_NO_IGUAL	5	X'00000005'
MQCFOP_NO_MAYOR	3	X'00000003'
MQCFOP_ME_GUSTA	18	X'00000012'
MQCFOP_NO_ES_IGUAL	21	X'00000015'
MQCFOP_CONTIENE	10	X'0000000A'
MQCFOP_EXCLUYE	13	X'0000000D'
MQCFOP_CONTIENE_GEN	26	X'0000001A'
MQCFOP_EXCLUYE_GEN	29	X'0000001D'

MQCFR_* (Recuperabilidad CF)

Tabla 69. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCFR_SÍ	1	X'00000001'
MQCFR_NO	0	X'00000000'

MQCFSF_* (Estructura de parámetros del filtro de cadena de formato de comando)

Tabla 70. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCFSF_ESTRUCTURA_LONGITUD_FIJA	24	X'00000018'

MQCFSL_* (Estructura de parámetros de la lista de cadenas de formato de comando)

Tabla 71. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCFSL_ESTRUCTURA_LONGITUD_FIJA	24	X'00000018'

MQCFST_* (Estructura de parámetros de cadena de formato de comando)

Tabla 72. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCFST_STRUC_LONGITUD_FIJA	20	X'00000014'

MQCFSTATUS_* (Estado del formato de comando CF)

Tabla 73. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCFSTATUS_NO_ENCONTRADO	0	X'00000000'
MQCFSTATUS_ACTIVO	1	X'00000001'
MQCFSTATUS_EN_RECUPERACIÓN	2	X'00000002'
MQCFSTATUS_EN_COPIA_DE_RESERVA	3	X'00000003'
MQCFSTATUS_FAILED	4	X'00000004'
MQCFSTATUS_NONE	5	X'00000005'
MQCFSTATUS_DESCONOCIDO	6	X'00000006'
MQCFSTATUS_ADMIN_INCOMPLETO	20	X'00000014'
MQCFSTATUS_NUNCA_UTILIZADO	21	X'00000015'
MQCFSTATUS_SIN_COPIA_DE_SEGURIDAD	22	X'00000016'
MQCFSTATUS_NO_FALLÓ	23	X'00000017'
MQCFSTATUS_NO_RECUPERABLE	24	X'00000018'
MQCFSTATUS_XES_ERROR	25	X'00000019'

MQCFT_* (Tipos de estructura de formato de comando)

Tabla 74. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCFT_NINGUNO	0	X'00000000'
MQCFT_COMMAND	1	X'00000001'
MQCFT_RESPONSE	2	X'00000002'
MQCFT_ENTERO	3	X'00000003'
MQCFT_CADEN	4	X'00000004'
MQCFT_LISTA_ENTEROS	5	X'00000005'
MQCFT_STRING_LIST	6	X'00000006'
MQCFT_EVENT	7	X'00000007'
MQCFT_USER	8	X'00000008'
MQCFT_CADENA_DE_BYTES	9	X'00000009'
MQCFT_TRACE_ROUTE	10	X'0000000A'
MQCFT_REPORT	12	X'0000000C'
MQCFT_FILTRO_ENTER	13	X'0000000D'
MQCFT_STRING_FILTER	14	X'0000000E'
MQCFT_BYTE_STRING_FILTRO	15	X'0000000F'
MQCFT_COMANDO_XR	16	X'00000010'
MQCFT_XR_MSG	17	X'00000011'
MQCFT_XR_ARTÍCULO	18	X'00000012'
MQCFT_XR_RESUMEN	19	X'00000013'
MQCFT_GROUP	20	X'00000014'
MQCFT_ESTADÍSTICAS	21	X'00000015'

Tabla 74. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQCFT_CONTABILIDAD	22	X'00000016'
MQCFT_INTEGER64	23	X'00000017'
MQCFT_INTEGER64_LIST	25	X'00000019'

MQCFTYPE_* (Tipos de formato de comando CF)

Tabla 75. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCFTYPE_APPL	0	X'00000000'
MQCFTYPE_ADMIN	1	X'00000001'

MQCFUNC_* (Funciones de CICS encabezado de información)

Tabla 76. Estructuras de constantes	
Nombre	Estructura
MQCFUNC_MQCONN	"CONN"
MQCFUNC_MQGET	"GET¬"
MQCFUNC_MQINQ	"INQ¬"
MQCFUNC_MQOPEN	"OPEN"
MQCFUNC_MQPUT	"PUT¬"
MQCFUNC_MQPUT1	"PUT1"
MQCFUNC_NONE	"¬¬¬"
MQCFUNC_MQCONN_ARRAY	'C','O','N','N'
MQCFUNC_MQGET_ARRAY	'G','E','T','¬'
MQCFUNC_MQINQ_ARRAY	'I','N','Q','¬'
MQCFUNC_MQOPEN_ARRAY	'O','P','E','N'
MQCFUNC_MQPUT_ARRAY	'P','U','T','¬'
MQCFUNC_MQPUT1_ARRAY	'P','U','T','1'
MQCFUNC_NONE_ARRAY	'¬','¬','¬','¬'

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

MQCGWI_* (cabecera CICS de información Obtener intervalo de espera)

Tabla 77. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCGWI_DEFAULT	-2	X'FFFFFFFE'

MQCHAD_* (Definición automática de canales)

Tabla 78. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCHAD_DISABLED	0	X'00000000'
MQCHAD_ENABLED	1	X'00000001'

MQCHIDS_* (Formato de comando en estado dudoso)

Tabla 79. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCHIDS_NO_DUDA	0	X'00000000'
MQCHIDS_INDOUBT	1	X'00000001'

MQCHLD_* (Disposiciones de canal en formato de comando)

Tabla 80. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCHLD_TODOS	-1	X'FFFFFFFF'
MQCHLD_DEFAULT	1	X'00000001'
MQCHLD_COMPARTIDO	2	X'00000002'
MQCHLD_PRIVADO	4	X'00000004'
MQCHLD_FIXSHARED	5	X'00000005'

MQCHS_* (Formato de mandato, Estado del canal)

Tabla 81. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCHS_INACTIVE	0	X'00000000'
MQCHS_BINDING	1	X'00000001'
MQCHS_STARTING	2	X'00000002'
MQCHS_RUNNING	3	X'00000003'
MQCHS_STOPPING	4	X'00000004'
MQCHS_RETRYING	5	X'00000005'
MQCHS_STOPPED	6	X'00000006'
MQCHS_REQUESTING	7	X'00000007'
MQCHS_PAUSED	8	X'00000008'
MQCHS_INITIALIZING	13	X'0000000D'
MQCHS_SWITCHING	14	X'0000000E'

MQCHSH_* (Opciones de reinicio compartido del canal en formato de comando)

Tabla 82. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCHSH_REINICIAR_NO	0	X'00000000'
MQCHSH_REINICIAR_SÍ	1	X'00000001'

MQCHSR_* (Opciones de parada de canal en formato de comando)

Tabla 83. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCHSR_STOP_NO_SOLICITADO	0	X'00000000'

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

MQCHSSTATE_* (Formato de comando Subestados de canal)

Tabla 84. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCHSSTATE_OTRO	0	X'00000000'
MQCHSSTATE_FIN_DEL_LOTE	100	X'00000064'
MQCHSSTATE_ENVIANDO	200	X'000000C8'
MQCHSSTATE_RECEIVING	300	X'0000012C'
MQCHSSTATE_SERIALIZING	400	X'00000190'
MQCHSSTATE_RESINCRONIZANDO	500	X'000001F4'
MQCHSSTATE_HEARTBEATING	600	X'00000258'
MQCHSSTATE_IN_SCYEXIT	700	X'000002BC'
MQCHSSTATE_IN_RCVEXIT	800	X'00000320'
MQCHSSTATE_IN_SENDEXIT	900	X'00000384'
MQCHSSTATE_IN_MSGEXIT	1000	X'000003E8'
MQCHSSTATE_IN_MREXIT	1100	X'0000044C'
MQCHSSTATE_EN_CHADEXIT	1200	X'000004B0'
MQCHSSTATE_NET_CONECTANDO	1250	X'000004E2'
MQCHSSTATE_SSL_HANDSHAKING	1300	X'00000514'
MQCHSSTATE_NAME_SERVER	1400	X'00000578'
MQCHSSTATE_EN_MQPUT	1500	X'000005DC'
MQCHSSTATE_IN_MQGET	1600	X'00000640'
MQCHSSTATE_IN_MQI_CALL	1700	X'000006A4'
MQCHSSTATE_COMPRIENDO	1800	X'00000708'

MQCHT_* (Tipos de canal)

Tabla 85. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCHT_SENDER	1	X'00000001'
MQCHT_SERVER	2	X'00000002'
MQCHT_RECEIVER	3	X'00000003'
MQCHT_REQUESTER	4	X'00000004'
MQCHT_TODOS	5	X'00000005'
MQCHT_CLNTCONN	6	X'00000006'
MQCHT_SVRCONN	7	X'00000007'
MQCHT_CLUSRCVR	8	X'00000008'
MQCHT_CLUSSDR	9	X'00000009'

MQCHTAB_* (Tipos de tabla de canales de formato de comando)

Tabla 86. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCHTAB_Q_MGR	1	X'00000001'
MQCHTAB_CLNTCONN	2	X'00000002'

MQCI_* (Identificador de correlación)

Tabla 87. Nombres y valores constantes	
Nombre	Valor
MQCI_NONE	X'00...00' (24 nulos)
MQCI_NINGUNA_MATRIZ	'\0','\0',... (24 nulos)
MQCI_NUEVA_SESIÓN	X'414D5121...'
MQCI_NUEVA_SESIÓN_MATRIZ	'\x41','\x4D','\51','\x21',...

MQCIH_* (estructura de CICS encabezado de información y banderas)

CICS Estructura del encabezado de información

Tabla 88. Estructuras de constantes	
Nombre	Estructura
MQCIH_STRUC_ID	"CIH¬"
MQCIH_STRUC_ID_ARRAY	'C','I','H','¬'

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

Tabla 89. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCIH_VERSION_1	1	X'00000001'
MQCIH_VERSION_2	2	X'00000002'
MQCIH_VERSION_ACTUAL	2	X'00000002'
MQCIH_LENGTH_1	164	X'000000A4'
MQCIH_LENGTH_2	180	X'000000B4'
MQCIH_LONGITUD_ACTUAL	180	X'000000B4'

CICS cabecera de información Indicadores

Tabla 90. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCIH_NINGUNO	0	X'00000000'
MQCIH_PLAZO_DE_VIGENCIA	1	X'00000001'
MQCIH_VENCIMIENTO_ILIMITADO	0	X'00000000'
MQCIH_RESPUESTA_SIN_NULOS	2	X'00000002'
MQCIH_RESPONDER_CON_NULLS	0	X'00000000'
MQCIH_SINCRONIZAR_AL_RETORNO	4	X'00000004'
MQCIH_NO_SINCRONIZACIÓN_AL_RETORNO	0	X'00000000'

MQCLCT_* (Tipos de caché de clúster)

Tabla 91. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCLCT_ESTÁTICO	0	X'00000000'
MQCLCT_DINÁMICO	1	X'00000001'

MQCLRS_* (Formato de comando Borrar cadena de tema Ámbito)

Tabla 92. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCLRS_LOCAL	1	X'00000001'
MQCLRS_GLOBAL	2	X'00000002'

MQCLRT_* (Formato de comando Borrar cadena de tipo de tema)

Tabla 93. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCLRT_RETENIDO	1	X'00000001'

MQCLT_* (tipos de CICS enlace de encabezado de información)

Tabla 94. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
PROGRAMA_MQCLT	1	X'00000001'
MQCLT_TRANSACCIÓN	2	X'00000002'

MQCLWL_* (Carga de trabajo del clúster)

Tabla 95. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCLWL_USEQ_LOCAL	0	X'00000000'
MQCLWL_USEQ_CUALQUIERA	1	X'00000001'
MQCLWL_USEQ_AS_Q_MGR	-3	X'FFFFFFFD'

MQCLXQ_* (Tipo de cola de transmisión de clúster)

MQCLXQ_* son los valores que se pueden establecer en el atributo del gestor de colas DEFCLXQ. El **DEFCLXQ** atributo controla qué cola de transmisión seleccionan por defecto los canales emisores del clúster para recibir mensajes y enviarlos a los canales receptores del clúster.

Tabla 96. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCLXQ_SCTQ	0	X'00000000'
MQCLXQ_CHANNEL	1	X'00000001'

Referencia relacionada

[“DefClusterXmitQueueType \(MQLONG\)” en la página 821](#)

El atributo « DefClusterXmitQueueType » controla qué cola de transmisión seleccionan por defecto los canales «cluster-sender» para recibir mensajes y enviarlos a los canales «cluster-receiver».

[Cambiar gestor de colas](#)

[Consultar gestor de colas](#)

[Gestor de colas de consultas \(respuesta\)](#)

“MQINQ - Consultar atributos de objeto” en la página 707

La MQINQ llamada devuelve una matriz de números enteros y un conjunto de cadenas de caracteres que contienen los atributos de un objeto.

MQCMD_* (Códigos de comando)

Tabla 97. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCMD_NINGUNO	0	X'00000000'
MQCMD_CAMBIO_Q_GESTOR	1	X'00000001'
MQCMD_CONSULTAR_Q_MGR	2	X'00000002'
MQCMD_CAMBIO_PROCESO	3	X'00000003'
MQCMD_COPY_PROCESS	4	X'00000004'
MQCMD_CREAR_PROCESO	5	X'00000005'
MQCMD_ELIMINAR_PROCESO	6	X'00000006'
MQCMD_CONSULTAR_PROCESO	7	X'00000007'
MQCMD_CAMBIO_Q	8	X'00000008'
MQCMD_CLEAR_Q	9	X'00000009'
MQCMD_COPY_Q	10	X'0000000A'
MQCMD_CREAR_Q	11	X'0000000B'
MQCMD_DELETE_Q	12	X'0000000C'
MQCMD_CONSULTAR_Q	13	X'0000000D'
MQCMD_ACTUALIZAR_GESTOR_DE_COLAS_DE_MESSAGES	16	X'00000010'
MQCMD_RESET_Q_STATS	17	X'00000011'
MQCMD_CONSULTAR_Q_NOMBRES	18	X'00000012'
MQCMD_CONSULTAR_NOMBRES_PROCESOS	19	X'00000013'
MQCMD_CONSULTAR_NOMBRES_DE_CANALES	20	X'00000014'
MQCMD_CAMBIAR_CANAL	21	X'00000015'
MQCMD_COPY_CHANNEL	22	X'00000016'
MQCMD_CREAR_CANAL	23	X'00000017'
MQCMD_ELIMINAR_CANAL	24	X'00000018'
MQCMD_CONSULTAR_CANAL	25	X'00000019'
MQCMD_PING_CANAL	26	X'0000001A'
MQCMD_RESET_CANAL	27	X'0000001B'
MQCMD_INICIAR_CANAL	28	X'0000001C'
MQCMD_DETENER_CANAL	29	X'0000001D'
MQCMD_INICIO_INICIALIZACIÓN_CANAL	30	X'0000001E'
MQCMD_INICIAR_ESCUCHADOR_DE_CANALES	31	X'0000001F'
MQCMD_CAMBIAR_LISTA_DE_NOMBRES	32	X'00000020'
MQCMD_COPY_LISTA_DE_NOMBRES	33	X'00000021'
MQCMD_CREAR_LISTA_DE_NOMBRES	34	X'00000022'

Tabla 97. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQCMD_ELIMINAR_LISTA_DE_NOMBRES	35	X'00000023'
MQCMD_CONSULTAR_LISTA_DE_NOMBRES	36	X'00000024'
MQCMD_CONSULTAR_LISTA_DE_NOMBRES	37	X'00000025'
MQCMD_ESCAPE	38	X'00000026'
MQCMD_RESOLVER_CANAL	39	X'00000027'
MQCMD_PING_Q_MGR	40	X'00000028'
MQCMD_CONSULTAR_ESTADO_DE_LA_CONSULTA	41	X'00000029'
MQCMD_CONSULTAR_ESTADO_CANAL	42	X'0000002A'
MQCMD_CONFIG_EVENT	43	X'0000002B'
MQCMD_Q_MGR_EVENTO	44	X'0000002C'
MQCMD_PERFM_EVENT	45	X'0000002D'
MQCMD_CANAL_EVENTO	46	X'0000002E'
MQCMD_ELIMINAR_PUBLICACIÓN	60	X'0000003C'
MQCMD_DEREGISTRAR_EDITOR	61	X'0000003D'
MQCMD_DEREGISTRAR_SUSCRIPTOR	62	X'0000003E'
MQCMD_PUBLICAR	63	X'0000003F'
MQCMD_REGISTRAR_EDITOR	64	X'00000040'
MQCMD_REGISTRAR_SUSCRIPTOR	65	X'00000041'
MQCMD_SOLICITAR_ACTUALIZACIÓN	66	X'00000042'
MQCMD_BROKER_INTERNO	67	X'00000043'
MQCMD_ACTIVITY_MSG	69	X'00000045'
MQCMD_CONSULTAR_CLÚSTER_Q_MGR	70	X'00000046'
MQCMD_RESUME_Q_MGR_CLUSTER	71	X'00000047'
MQCMD_SUSPENDER_Q_MGR_CLUSTER	72	X'00000048'
MQCMD_REFRESH_CLUSTER	73	X'00000049'
MQCMD_RESET_CLUSTER	74	X'0000004A'
MQCMD_TRACE_ROUTE	75	X'0000004B'
MQCMD_ACTUALIZAR_SEGURIDAD	78	X'0000004E'
MQCMD_CAMBIAR_INFORMACIÓN_DE_AUTENTICACIÓN	79	X'0000004F'
MQCMD_COPY_AUTH_INFO	80	X'00000050'
MQCMD_CREAR_INFORMACIÓN_DE_AUTENTICACIÓN	81	X'00000051'
MQCMD_ELIMINAR_INFORMACIÓN_DE_AUTORIZACIÓN	82	X'00000052'
MQCMD_SOLICITAR_INFORMACIÓN_AUTENTIFICACIÓN	83	X'00000053'
MQCMD_CONSULTAR_INFORMACIÓN_AUTENTICACIÓN_NOMBRES	84	X'00000054'
MQCMD_CONSULTAR_CONEXIÓN	85	X'00000055'
MQCMD_DETENER_CONEXIÓN	86	X'00000056'
MQCMD_CONSULTAR_REGISTROS_AUTENTIFICACIÓN	87	X'00000057'
MQCMD_CONSULTAR_ENTIDAD_AUTENTICACIÓN	88	X'00000058'
MQCMD_DELETE_AUTH_REC	89	X'00000059'

Tabla 97. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQCMD_SET_AUTH_REC	90	X'0000005A'
MQCMD_REGISTRADOR_EVENTO	91	X'0000005B'
MQCMD_RESET_Q_MGR	92	X'0000005C'
MQCMD_CAMBIAR_LISTENER	93	X'0000005D'
MQCMD_COPY_LISTENER	94	X'0000005E'
MQCMD_CREAR_RECEPTOR	95	X'0000005F'
MQCMD_DELETE_LISTENER	96	X'00000060'
MQCMD_CONSULTAR_RECEPTOR	97	X'00000061'
MQCMD_CONSULTAR_ESTADO_DEL_OYENTE	98	X'00000062'
MQCMD_COMANDO_EVENTO	99	X'00000063'
MQCMD_CAMBIO_SEGURIDAD	100	X'00000064'
MQCMD_CAMBIO_ESTRUCTURA_CF	101	X'00000065'
MQCMD_CAMBIO_CLASE_ETAPA	102	X'00000066'
MQCMD_CAMBIO TRASLADO	103	X'00000067'
MQCMD_ARCHIVO_REGISTRO	104	X'00000068'
MQCMD_BACKUP_CF_STRUC	105	X'00000069'
MQCMD_CREAR_POOL_DE_BÚFER	106	X'0000006A'
MQCMD_CREAR_CONJUNTO_DE_PÁGINAS	107	X'0000006B'
MQCMD_CREAR_ESTRUCTURA_CF	108	X'0000006C'
MQCMD_CREAR_CLASE_STG	109	X'0000006D'
MQCMD_COPY_CF_STRUC	110	X'0000006E'
MQCMD_COPY_STG_CLASE	111	X'0000006F'
MQCMD_DELETE_CF_STRUC	112	X'00000070'
MQCMD_DELETE_STG_CLASS	113	X'00000071'
MQCMD_CONSULTAR_ARCHIVO	114	X'00000072'
MQCMD_CONSULTAR_ESTRUCTURA_CF	115	X'00000073'
MQCMD_CONSULTAR_ESTADO_ESTRUCTURA_CF	116	X'00000074'
MQCMD_CONSULTAR_CMD_SERVIDOR	117	X'00000075'
MQCMD_CONSULTAR_INICIALIZACIÓN_CANAL	118	X'00000076'
MQCMD_CONSULTAR_QSG	119	X'00000077'
MQCMD_CONSULTAR_REGISTRO	120	X'00000078'
MQCMD_CONSULTAR_SEGURIDAD	121	X'00000079'
MQCMD_CONSULTAR_CLASE_STG	122	X'0000007A'
MQCMD_CONSULTAR_SISTEMA	123	X'0000007B'
MQCMD_CONSULTAR_HILO	124	X'0000007C'
MQCMD_CONSULTAR_RASTREO	125	X'0000007D'
MQCMD_CONSULTAR_USO	126	X'0000007E'
MQCMD_MOVER_Q	127	X'0000007F'
MQCMD_RECUPERAR_BSDS	128	X'00000080'

Tabla 97. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQCMD_RECUPERAR_ESTRUCTURA_CF	129	X'00000081'
MQCMD_RESET_TPIPE	130	X'00000082'
MQCMD_RESOLVER_DUDAS	131	X'00000083'
MQCMD_RESUME_Q_MGR	132	X'00000084'
MQCMD_VERIFICAR_SEGURIDAD	133	X'00000085'
MQCMD_ESTABLECER_ARCHIVO	134	X'00000086'
MQCMD_SET_LOG	136	X'00000088'
MQCMD_CONFIGURAR_SISTEMA	137	X'00000089'
MQCMD_INICIAR_CMD_SERVIDOR	138	X'0000008A'
MQCMD_INICIO_Q_MGR	139	X'0000008B'
MQCMD_INICIAR_RASTREO	140	X'0000008C'
MQCMD_DETENER_INICIALIZACIÓN_CANAL	141	X'0000008D'
MQCMD_DETENER_LISTENER_DE_CANAL	142	X'0000008E'
MQCMD_STOP_CMD_SERVER	143	X'0000008F'
MQCMD_DETENER_GESTOR_DE_COLAS_DE_TRABAJO	144	X'00000090'
MQCMD_DETENER_RASTREO	145	X'00000091'
MQCMD_SUSPENDER_Q_MGR	146	X'00000092'
MQCMD_CONSULTAR_NOMBRES_ESTRUCTURAS_CF	147	X'00000093'
MQCMD_CONSULTAR_CLASES_NOMBRES	148	X'00000094'
MQCMD_CAMBIAR_SERVICIO	149	X'00000095'
MQCMD_COPY_SERVICE	150	X'00000096'
MQCMD_CREAR_SERVICIO	151	X'00000097'
MQCMD_ELIMINAR_SERVICIO	152	X'00000098'
MQCMD_CONSULTAR_SERVICIO	153	X'00000099'
MQCMD_CONSULTAR_ESTADO_DEL_SERVICIO	154	X'0000009A'
MQCMD_INICIAR_SERVICIO	155	X'0000009B'
MQCMD_DETENER_SERVICIO	156	X'0000009C'
MQCMD_ELIMINAR_POOL_DE_BÚFER	157	X'0000009D'
MQCMD_ELIMINAR_CONJUNTO_DE_PÁGINAS	158	X'0000009E'
MQCMD_CAMBIAR_POOL_DE_BÚFER	159	X'0000009F'
MQCMD_CAMBIAR_PÁGINA_CONFIGURACIÓN	160	X'000000A0'
MQCMD_CONSULTAR_ESTADO_GERENTE_PEDIDOS	161	X'000000A1'
MQCMD_CREAR_REGISTRO	162	X'000000A2'
MQCMD_STATISTICS_MQI	164	X'000000A4'
MQCMD_STATISTICS_Q	165	X'000000A5'
MQCMD_STATISTICS_CHANNEL	166	X'000000A6'
MQCMD_ACCOUNTING_MQI	167	X'000000A7'
MQCMD_ACCOUNTING_Q	168	X'000000A8'
MQCMD_CONSULTAR_SERVICIO_AUTENTICACIÓN	169	X'000000A9'

Tabla 97. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQCMD_CAMBIAR_TEMA	170	X'000000AA'
MQCMD_COPY_TOPIC	171	X'000000AB'
MQCMD_CREAR_TEMA	172	X'000000AC'
MQCMD_ELIMINAR_TEMA	173	X'000000AD'
MQCMD_CONSULTAR_TEMA	174	X'000000AE'
MQCMD_CONSULTAR_NOMBRES_DE_TEMAS	175	X'000000AF'
MQCMD_CONSULTAR_SUSCRIPCIÓN	176	X'000000B0'
MQCMD_CREAR_SUSCRIPCIÓN	177	X'000000B1'
MQCMD_CAMBIAR_SUSCRIPCIÓN	178	X'000000B2'
MQCMD_ELIMINAR_SUSCRIPCIÓN	179	X'000000B3'
MQCMD_COPY_SUBSCRIPTION	181	X'000000B5'
MQCMD_CONSULTAR_ESTADO_SUB	182	X'000000B6'
MQCMD_CONSULTAR_ESTADO_DEL_TEMA	183	X'000000B7'
MQCMD_CLEAR_TOPIC_STRING	184	X'000000B8'
MQCMD_CONSULTAR_ESTADO_PUBSUB	185	X'000000B9'
MQCMD_PURGA_CANAL	195	X'000000C3'

MQCMDI_* (Valores de información de comando en formato de comando)

Tabla 98. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCMDI_CMDScope_ACEPTADO	1	X'00000001'
MQCMDI_CMDScope_GENERADO	2	X'00000002'
MQCMDI_CMDScope_COMPLETADO	3	X'00000003'
MQCMDI_QSG_DISP_COMPLETADO	4	X'00000004'
MQCMDI_COMANDO_ACEPTADO	5	X'00000005'
MQCMDI_SOLICITUD_DE_CLÚSTER_EN_COLA	6	X'00000006'
MQCMDI_CANAL_INICIALIZACIÓN_INICIADA	7	X'00000007'
MQCMDI_RECUPERACIÓN_INICIADA	11	X'0000000B'
MQCMDI_BACKUP_INICIADO	12	X'0000000C'
MQCMDI_RECUPERACIÓN_COMPLETADA	13	X'0000000D'
MQCMDI_SEC_TIMER_CERO	14	X'0000000E'
MQCMDI_ACTUALIZAR_CONFIGURACIÓN	16	X'00000010'
MQCMDI_SEC_ERROR_DE_VERIFICACIÓN	17	X'00000011'
MQCMDI_IMS_PUENTE_SUSPENDIDO	18	X'00000012'
MQCMDI_DB2_SUSPENDED	19	X'00000013'
MQCMDI_DB2_OBSOLETE_MSGS	20	X'00000014'
MQCMDI_SEC_MAYÚSCULAS	21	X'00000015'
MQCMDI_SEC_MAYÚSCULAS_y_minúsculas	22	X'00000016'

MQCMDL_* (Niveles de mandatos)

Tabla 99. Nombres y valores constantes	
Nombre	Valor
MQCMDL_LEVEL_800	800
MQCMDL_LEVEL_801	801
MQCMDL_LEVEL_802	802
MQCMDL_LEVEL_900	900
MQCMDL_LEVEL_901	901
MQCMDL_LEVEL_902	902
MQCMDL_LEVEL_903	903
MQCMDL_LEVEL_904	904
MQCMDL_LEVEL_905	905
MQCMDL_LEVEL_910	910
MQCMDL_LEVEL_912	912
MQCMDL_LEVEL_913	913
MQCMDL_LEVEL_914	914
MQCMDL_LEVEL_915	915
MQCMDL_LEVEL_920	920
MQCMDL_LEVEL_921	921
MQCMDL_LEVEL_922	922
MQCMDL_LEVEL_923	923
MQCMDL_LEVEL_924	924
MQCMDL_LEVEL_925	925
MQCMDL_LEVEL_930	930
MQCMDL_LEVEL_931	931
MQCMDL_LEVEL_932	932

MQCMHO_* (Crear opciones y estructura del identificador de mensaje)

Crear estructura de opciones de control de mensajes

Tabla 100. Estructuras de constantes	
Nombre	Estructura
MQCMHO_STRUC_ID	"CMHO"
MQCMHO_STRUC_ID_ARRAY	'C', 'M', 'H', 'O'

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

Tabla 101. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCMHO_VERSION_1	1	X'00000001'
MQCMHO_VERSION_ACTUAL	1	X'00000001'

Crear opciones de control de mensajes

Tabla 102. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCMHO_VALIDACIÓN_POR_DEFAULT	0	X'00000000'
MQCMHO_SIN_VALIDACIÓN	1	X'00000001'
MQCMHO_VALIDAR	2	X'00000002'
MQCMHO_NINGUNO	0	X'00000000'

MQCNO_* (Opciones de conexión y estructura)

Estructura de opciones de conexión

Tabla 103. Estructuras de constantes	
Nombre	Estructura
MQCNO_STRUC_ID	"CNO~"
MQCNO_STRUC_ID_ARRAY	'C','N','O','~'

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

Tabla 104. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCNO_VERSION_1	1	X'00000001'
MQCNO_VERSION_2	2	X'00000002'
MQCNO_VERSION_3	3	X'00000003'
MQCNO_VERSION_4	4	X'00000004'
MQCNO_VERSION_5	5	X'00000005'
MQCNO_VERSIÓN_ACTUAL	5	X'00000005'

Opciones de conexión

Tabla 105. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCNO_STANDARD_BINDING	0	X'00000000'
MQCNO_FASTPATH_BINDING	1	X'00000001'
MQCNO_SERIALIZE_CONN_TAG_Q_MGR	2	X'00000002'
MQCNO_SERIALIZE_CONN_TAG_QSG	4	X'00000004'
MQCNO_RESTRICT_CONN_TAG_Q_MGR	8	X'00000008'
MQCNO_RESTRICT_CONN_TAG_QSG	16	X'00000010'
MQCNO_HANDLE_SHARE_NONE	32	X'00000020'
MQCNO_MANEJAR_COMPARTIR_BLOQUE	64	X'00000040'
MQCNO_HANDLE_SHARE_NO_BLOCK	128	X'00000080'
MQCNO_SHARED_BINDING	256	X'00000100'
MQCNO_ISOLATED_BINDING	512	X'00000200'
MQCNO_LOCAL_BINDING	1024	X'00000400'
MQCNO_CLIENT_BINDING	2048	X'00000800'

Tabla 105. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQCNO_ACCOUNTING_MQI_ENABLED	4096	X'00001000'
MQCNO_ACCOUNTING_MQI_DISABLED	8192	X'00002000'
MQCNO_ACCOUNTING_Q_ENABLED	16384	X'00004000'
MQCNO_ACCOUNTING_Q_DISABLED	32768	X'00008000'
MQCNO_NO_CONV_COMPARTIR	65536	X'00010000'
MQCNO_TODAS_CONVERSACIONES_COMPARTIDAS	262144	X'00040000'
MQCNO_CD_SOLO_PARA_SALIDA	524288	X'00080000'
MQCNO_USAR_SELECCIÓN_DE_CD	1048576	X'00100000'
MQCNO_RECONECTAR	16777216	X'01000000'
MQCNO_RECONECTAR_COMO_DEF	0	X'00000000'
MQCNO_RECONECTAR_DESACTIVADO	33554432	X'02000000'
MQCNO_RECONECTAR_Q_MGR	67108864	X'04000000'
MQCNO_ACTIVITY_TRACE_ENABLED	134217728	X'08000000'
MQCNO_ACTIVITY_TRACE_DISABLED	268435456	X'10000000'
MQCNO_NONE	0	X'00000000'

MQCO_* (Opciones de cierre)

Tabla 106. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCO_INMEDIATO	0	X'00000000'
MQCO_NONE	0	X'00000000'
MQCO_DELETE	1	X'00000001'
MQCO_DELETE_PURGE	2	X'00000002'
MQCO_MANTENER_SUB	4	X'00000004'
MQCO_REMOVE_SUB	8	X'00000008'
MQCO_QUIESCE	32	X'00000020'

MQCODL_* (longitud de los datos de CICS salida del encabezado de información)

Tabla 107. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCODL_AS_ENTRADA	-1	X'FFFFFFFF'

MQCOMPRESS_* (Compresión de canal)

Tabla 108. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCOMPRESS_NO_DISPONIBLE	-1	X'FFFFFFFF'
MQCOMPRESS_NONE	0	X'00000000'
MQCOMPRESS_RLE	1	X'00000001'
MQCOMPRESS_ZLIBFAST	2	X'00000002'

Tabla 108. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQCOMPRESS_ZLIBHIGH	4	X'00000004'
MQCOMPRESS_SYSTEM	8	X'00000008'
 MQCOMPRESS_LZ4FAST	16	X'10000000'
 MQCOMPRESS_LZ4HIGH	32	X'20000000'
MQCOMPRESS_ANY	268435455	X'0FFFFFFF'

MQCONNID_* (Identificador de conexión)

Tabla 109. Nombres y valores constantes	
Nombre	Valor
MQCONNID_NONE	X'00...00' (24 nulos)
MQCONNID_NINGUNA_MATRIZ	'\0', '\0', ... (24 nulos)

MQCOPY_* (Opciones de copia de propiedades)

Tabla 110. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCOPY_NONE	0	X'00000000'
MQCOPY_ALL	1	X'00000001'
MQCOPY_FORWARD	2	X'00000002'
MQCOPY_PUBLISH	4	X'00000004'
MQCOPY_RESPUESTA	8	X'00000008'
MQCOPY_INFORME	16	X'00000010'
MQCOPY_DEFAULT	22	X'00000016'

MQCQT_* (Tipos de colas de clúster)

Tabla 111. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCQT_LOCAL_Q	1	X'00000001'
MQCQT_ALIAS_Q	2	X'00000002'
MQCQT_REMOTO_Q	3	X'00000003'
MQCQT_Q_MGR_ALIAS	4	X'00000004'

MQCRC_* (códigos de CICS retorno del encabezado de información)

Tabla 112. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCRC_OK	0	X'00000000'
MQCRC_CICS_EXEC_ERROR	1	X'00000001'
MQCRC_MQ_API_ERROR	2	X'00000002'
ERROR_PUENTE_MQCRC	3	X'00000003'
MQCRC_PUENTE_ABEND	4	X'00000004'

Tabla 112. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQCRC_APLICACIÓN_ABEND	5	X'00000005'
MQCRC_ERROR_DE_SEGURIDAD	6	X'00000006'
MQCRC_PROGRAMA_NO_DISPONIBLE	7	X'00000007'
MQCRC_BRIDGE_TIMEOUT	8	X'00000008'
MQCRC_TRANSID_NO_DISPONIBLE	9	X'00000009'

MQCS_* (constantes MQCBC estado del consumidor)

Tabla 113. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCS_NINGUNO	0	X'00000000'
MQCS_SUSPENDIDO_TEMPORAL	1	X'00000001'
MQCS_ACCIÓN_DE_USUARIO_SUSPENDIDA	2	X'00000002'
MQCS_SUSPENDIDO	3	X'00000003'
MQCS_DETENIDO	4	X'00000004'

MQCSC_* (códigos de CICS inicio de encabezado de información)

Tabla 114. Estructuras de constantes	
Nombre	Estructura
MQCSC_INICIO	"S¬¬"
MQCSC_INICIO_DATOS	"SD¬¬"
MQCSC_TERMINPUT	"TD¬¬"
MQCSC_NINGUNO	"¬¬¬"
MQCSC_INICIO_MATRIZ	'S','¬','¬','¬'
MQCSC_INICIO_MATRIZ_DE_DATOS	'S','D','¬','¬'
MQCSC_TERMINPUT_ARRAY	'T','D','¬','¬'
MQCSC_NINGUNA_MATRIZ	'¬','¬','¬','¬'

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

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

Estructura de los parámetros de seguridad de la conexión

Tabla 115. Estructuras de constantes	
Nombre	Estructura
MQCSP_STRUC_ID	"CSP¬"
MQCSP_STRUC_ID_ARRAY	'C','S','P','¬'

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

Tabla 116. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCSP_VERSION_1	1	X'00000001'
MQCSP_VERSION_2	2	X'00000002'
MQCSP_VERSION_3	3	X'00000003'
MQCSP_VERSIÓN_ACTUAL	3	X'00000003'

Parámetros de seguridad de la conexión Tipos de autenticación

Tabla 117. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCSP_AUTH_NONE	0	X'00000000'
MQCSP_AUTH_ID_DE_USUARIO_Y_CONTRASEÑA	1	X'00000001'
MQCSP_AUTH_ID_TOKEN	2	X'00000002'

MQCSRV_* (Opciones del servidor de comandos)

Tabla 118. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCSRV_CONVERTIR_NO	0	X'00000000'
MQCSRV_CONVERTIR_SÍ	1	X'00000001'
MQCSRV_DLQ_NO	0	X'00000000'
MQCSRV_DLQ_SÍ	1	X'00000001'

MQCT_* (Etiqueta de conexión del gestor de colas)

Tabla 119. Nombres y valores constantes	
Nombre	Valor
MQCT_NINGUNO	X'00...00' (128 nulos)
MQCT_NINGUNA_MATRIZ	'\0','\0',... (128 nulos)

MQCTES_* (cabecera CICS de información del estado final de la tarea)

Tabla 120. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCTES_NO SINCRONIZADO	0	X'00000000'
MQCTES_COMMIT	256	X'00000100'
MQCTES_RETIRADA	4352	X'00001100'
MQCTES_FINALIZAR_TAREA	65536	X'00010000'

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

Estructura de opciones MQCTL

Tabla 121. Estructuras de constantes	
Nombre	Estructura
MQCTLO_STRUC_ID	"CTLO"
MQCTLO_STRUC_ID_ARRAY	'C','T','L','O'

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

Tabla 122. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCTLO_VERSION_1	1	X'00000001'
MQCTLO_VERSIÓN_ACTUAL	1	X'00000001'

Opciones MQCTL Opciones de control del consumidor

Tabla 123. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCTLO_NINGUNO	0	X'00000000'
Afinidad de subprocesos MQCTLO	1	X'00000001'
MQCTLO_FAIL_IF QUIESCING	8192	X'00002000'

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

Tabla 124. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCUOWC_SOLO	273	X'00000111'
MQCUOWC_CONTINUAR	65536	X'00010000'
MQCUOWC_PRIMERO	17	X'00000011'
MQCUOWC_MEDIO	16	X'00000010'
MQCUOWC_ÚLTIMO	272	X'00000110'
MQCUOWC_COMMIT	256	X'00000100'
MQCUOWC_RETIRADA	4352	X'00001100'

MQCXP_* (Estructura de parámetros de salida del canal)

Tabla 125. Estructuras de constantes	
Nombre	Estructura
MQCXP_STRUC_ID	"CXP¬"
MQCXP_STRUC_ID_ARRAY	'C','X','P','¬'

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

Tabla 126. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQCXP_VERSION_1	1	X'00000001'
MQCXP_VERSION_2	2	X'00000002'
MQCXP_VERSION_3	3	X'00000003'
MQCXP_VERSION_4	4	X'00000004'
MQCXP_VERSION_5	5	X'00000005'
MQCXP_VERSION_6	6	X'00000006'
MQCXP_VERSION_7	7	X'00000007'
MQCXP_VERSION_8	8	X'00000008'
MQCXP_VERSION_9	9	X'00000009'
MQCXP_VERSION_ACTUAL	9	X'00000009'

MQDC_* (Clase de destino)

Tabla 127. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQDC_GESTIONADO	1	X'00000001'
MQDC_PROPORCIONADO	2	X'00000002'

MQDCC_* (Opciones de conversión, máscaras y factores)

Opciones de conversión

Tabla 128. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQDCC_CONVERSIÓN_POR_DEFAULT	1	X'00000001'
MQDCC_LLENAR_BÚFER_DESTINO	2	X'00000002'
MQDCC_INT_CONVERSIÓN_POR_DEFAULT	4	X'00000004'
MQDCC_FUENTE_ENC_NATIVA	(value differs by platform or version)	(value differs by platform or version)
MQDCC_FUENTE_ENC_NORMAL	16	X'00000010'
MQDCC_FUENTE_ENCODING_INVERTIDA	32	X'00000020'
MQDCC_FUENTE_ENC_INDETERMINADA	0	X'00000000'
MQDCC_TARGET_ENC_NATIVE	(value differs by platform or version)	(value differs by platform or version)
MQDCC_OBJETIVO_ENC_NORMAL	256	X'00000100'
MQDCC_OBJETIVO_ENC_INVERTIDO	512	X'00000200'
MQDCC_TARGET_ENC_INDETERMINADO	0	X'00000000'
MQDCC_NINGUNO	0	X'00000000'

Opciones de conversión Máscaras y factores

Tabla 129. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQDCC_FUENTE_ENC_MÁSCARA	240	X'000000F0'

Tabla 129. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQDCC_TARGET_ENC_MASK	3840	X'00000F00'
MQDCC_FUENTE_ENC_FACTOR	16	X'00000010'
MQDCC_OBJETIVO_ENC_FACTOR	256	X'00000100'

MQDELO_* (Opciones de eliminación de publicación/suscripción)

Tabla 130. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQDELO_NONE	0	X'00000000'
MQDELO_LOCAL	4	X'00000004'

MQDH_* (Estructura del encabezado de distribución)

Tabla 131. Estructuras de constantes	
Nombre	Estructura
MQDH_STRUC_ID	"DH¬¬"
MQDH_STRUC_ID_ARRAY	'D','H','¬','¬'

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

Tabla 132. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQDH_VERSION_1	1	X'00000001'
MQDH_VERSION_ACTUAL	1	X'00000001'

MQDHF_* (Indicadores de encabezado de distribución)

Tabla 133. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQDHF_NUEVO_MSG_IDS	1	X'00000001'
MQDHF_NINGUNO	0	X'00000000'

MQDISCONNECT_* (Formato de comando Tipos de desconexión)

Tabla 134. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQDISCONNECT_NORMAL	0	X'00000000'
MQDISCONNECT_IMPLICIT	1	X'00000001'
MQDISCONNECT_Q_MGR	2	X'00000002'

MQDL_* (Listas de distribución)

Tabla 135. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQDL_COMPATIBLE	1	X'00000001'
MQDL_NO_COMPATIBLE	0	X'00000000'

MQDLH_* (Estructura del encabezado de correo no entregado)

Tabla 136. Estructuras de constantes	
Nombre	Estructura
MQDLH_STRUC_ID	"DLH¬"
MQDLH_STRUC_ID_ARRAY	'D', 'L', 'H', '¬'

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

MQDLH_VERSION_1	1	X'00000001'
MQDLH_VERSION_1_ACTUAL	1	X'00000001'

MQDLV_* (Entrega de mensajes persistente/no persistente)

Tabla 137. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQDLV_AS_PARENT	0	X'00000000'
MQDLV_TODOS	1	X'00000001'
MQDLV_TODOS_DUR	2	X'00000002'
MQDLV_TODOS_DISPONIBLES	3	X'00000003'

MQDMHO_* (Eliminar opciones y estructura del identificador de mensaje)

Eliminar estructura de opciones de control de mensajes

Tabla 138. Estructuras de constantes	
Nombre	Estructura
MQDMHO_STRUC_ID	"DMHO"
MQDMHO_STRUC_ID_ARRAY	'D', 'M', 'H', 'O'

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

Tabla 139. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQDMHO_VERSION_1	1	X'00000001'
MQDMHO_VERSION_1_ACTUAL	1	X'00000001'

Opciones para eliminar el identificador de mensaje

Tabla 140. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQDMHO_NINGUNO	0	X'00000000'

MQDMPO_* (Eliminar opciones y estructura de propiedades del mensaje)

Eliminar estructura de opciones de propiedades del mensaje

Tabla 141. Estructuras de constantes	
Nombre	Estructura
MQDMPO_STRUC_ID	"DMPO"
MQDMPO_STRUC_ID_ARRAY	'D','M','P','O'

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

Tabla 142. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQDMPO_VERSION_1	1	X'00000001'
MQDMPO_VERSION_ACTUAL	1	X'00000001'

Opciones de propiedades de mensaje eliminado

Tabla 143. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQDMPO_DEL_FIRST	0	X'00000000'
MQDMPO_DEL_PROP_BAJO_EL_CURSOR	1	X'00000001'
MQDMPO_NINGUNO	0	X'00000000'

MQDNSWLM_* (DNS WLM)

Tabla 144. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQDNSWLM_NO	0	X'00000000'
MQDNSWLM_SÍ	1	X'00000001'

MQDT_* (Tipos de destino)

Tabla 145. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQDT_APPL	1	X'00000001'
MQDT_CORREDOR	2	X'00000002'

MQDXP_* (Estructura del parámetro de salida de conversión)

Tabla 146. Estructuras de constantes	
Nombre	Estructura
MQDXP_STRUC_ID	"DXP¬"
MQDXP_STRUC_ID_ARRAY	'D','X','P','¬'

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

Tabla 147. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQDXP_VERSION_1	1	X'00000001'
MQDXP_VERSION_2	2	X'00000002'
MQDXP_VERSIÓN_ACTUAL	2	X'00000002'

MQEC_* (Valores de señal)

Tabla 148. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQEC_MSG_LLEGADO	2	X'00000002'
MQEC_WAIT_INTERVAL_EXPIRED	3	X'00000003'
MQEC_WAIT_CANCELADO	4	X'00000004'
MQEC_Q_MGR_INACTIVO	5	X'00000005'
MQEC_CONEXIÓN QUIESCING	6	X'00000006'

MQEI_* (Caducidad)

Tabla 149. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQEI_UNLIMITED	-1	X'FFFFFFFF'

MQENC_* (Codificación)

MQENC_* (Codificación)

Tabla 150. Valores de las constantes por plataforma			
Nombre	Plataforma	Valor decimal	Valor hexadecimal
MQENC_NATIVE	IBM i	273	«X'00000111»
	Linux®	546	«X'00000222»
	Linux en SPARC	273	«X'00000111»
	Linux en x86	546	«X'00000222»
	AIX and Linux	273	«X'00000111»
	Windows	546	«X'00000222»
	COBOL, de Micro Focus, en Windows	17	«X'00000011»
	z/OS	785	«X'00000311»

MQENC_* (Máscaras de codificación)

Tabla 151. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQENC_INTEGER_MASK	15	X'0000000F'
MQENC_MASCARADA_DECIMAL	240	X'000000F0'
MQENC_FLOAT_MASK	3840	X'00000F00'
MQENC_RESERVADO_MÁSCARA	-4096	X'FFFFFF000'

MQENC_* (Codificaciones para enteros binarios)

Tabla 152. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQENC_INTEGER_UNDEFINED	0	X'00000000'
MQENC_INTEGER_NORMAL	1	X'00000001'
MQENC_INTEGER_REVERSED	2	X'00000002'

MQENC_* (Codificaciones para enteros decimales empaquetados)

Tabla 153. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQENC_DECIMAL_UNDEFINED	0	X'00000000'
MQENC_DECIMAL_NORMAL	16	X'00000010'
MQENC_DECIMAL_REVERSED	32	X'00000020'

MQENC_* (Codificaciones para números de coma flotante)

Tabla 154. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQENC_FLOAT_UNDEFINED	0	X'00000000'
MQENC_FLOAT_IEEE_NORMAL	256	X'00000100'
MQENC_FLOAT_IEEE_REVERSED	512	X'00000200'
MQENC_FLOAT_S390	768	X'00000300'
MQENC_FLOAT_TNS	1024	X'00000400'

MQEPH_* (Estructura de encabezado de formato de comando integrado y indicadores)

Estructura del encabezado del formato de comando integrado

Tabla 155. Estructuras de constantes	
Nombre	Estructura
MQEPH_STRUC_ID	"EPH¬"
MQEPH_STRUC_ID_ARRAY	'E', 'P', 'H', '¬'

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

Tabla 156. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQEPH_ESTRUCTURA_LONGITUD_FIJA	68	X'00000044'
MQEPH_VERSION_1	1	X'00000001'
MQEPH_VERSION_ACTUAL	1	X'00000001'

Formato de comando integrado Encabezado Indicadores

Tabla 157. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQEPH_NONE	0	X'00000000'
MQEPH_CCSID_EMBEDDED	1	X'00000001'

MQET_* (Tipos de escape de formato de comando)

Tabla 158. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQET_MQSC	1	X'00000001'

MQEVO_* (Formato de comando Orígenes de eventos)

Tabla 159. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQEVO_OTROS	0	X'00000000'
MQEVO_CONSOLE	1	X'00000001'
MQEVO_INIT	2	X'00000002'
MQEVO_MSG	3	X'00000003'
MQEVO_MQSET	4	X'00000004'
MQEVO_INTERNO	5	X'00000005'
MQEVO_MQSUB	6	X'00000006'
MQEVO_CTLMSG	7	X'00000007'
MQEVO_REST	8	X'00000008'

MQEVR_* (Registro de eventos en formato de comando)

Tabla 160. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQEVR_DISABLED	0	X'00000000'
MQEVR_ENABLED	1	X'00000001'
MQEVR_EXCEPCIÓN	2	X'00000002'
MQEVR_NO_VISUALIZAR	3	X'00000003'

MQEXPI_* (Intervalo de exploración de caducidad)

Tabla 161. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQEXPI_OFF	0	X'00000000'

MQFB_* (Valores de retroalimentación)

Tabla 162. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQFB_NONE	0	X'00000000'

Tabla 162. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQFB_SISTEMA_PRIMERO	1	X'00000001'
MQFB_SALIR	256	X'00000100'
MQFB_CADUCIDAD	258	X'00000102'
MQFB_COA	259	X'00000103'
MQFB_COD	260	X'00000104'
MQFB_CANAL_COMPLETADO	262	X'00000106'
MQFB_CANAL_FALLO_REINTENTO	263	X'00000107'
MQFB_CANAL_FALLO	264	X'00000108'
MQFB_APPL_NO_SE_PUEDE_INICIAR	265	X'00000109'
MQFB_TM_ERROR	266	X'0000010A'
MQFB_APPL_TYPE_ERROR	267	X'0000010B'
MQFB_DETENIDO_POR_MENSAJE_SALIDA	268	X'0000010C'
MQFB_ACTIVITY	269	X'0000010D'
MQFB_XMIT_Q_MSG_ERROR	271	X'0000010F'
MQFB_PAN	275	X'00000113'
MQFB_NAN	276	X'00000114'
MQFB_DETENIDO_POR_SALIDA_DE_CHAD	277	X'00000115'
MQFB_DETENIDO_POR_PUBSUB_EXIT	279	X'00000117'
MQFB_NO_ES_UN_REPOSITORIO_MSG	280	X'00000118'
MQFB_BIND_OPEN_CLUSRCVR_DEL	281	X'00000119'
MQFB_MÁXIMO_ACTIVIDADES	282	X'0000011A'
MQFB_NO_REENVIADO	283	X'0000011B'
MQFB_NO_ENTREGADO	284	X'0000011C'
MQFB_REENVÍO NO ADMITIDO	285	X'0000011D'
MQFB_ENTREGA NO ADMISIBLE	286	X'0000011E'
MQFB_LONGITUD_DE_DATOS_CERO	291	X'00000123'
MQFB_LONGITUD_DE_DATOS_NEGATIVA	292	X'00000124'
MQFB_DATA_LENGTH_TOO_BIG	293	X'00000125'
MQFB_DESBORDAMIENTO_DE_BUFFER	294	X'00000126'
MQFB_LONGITUD_DESVIADA_EN_UNO	295	X'00000127'
MQFB_IIH_ERROR	296	X'00000128'
MQFB_NO_AUTORIZADO_PARA_IMS	298	X'0000012A'
MQFB_IMS_ERROR	300	X'0000012C'
MQFB_IMS_PRIMERO	301	X'0000012D'
MQFB_IMS_ÚLTIMO	399	X'0000018F'
MQFB_CICS_ERROR_INTERNO	401	X'00000191'
MQFB_CICS_NO_AUTORIZADO	402	X'00000192'
Fallo del puente MQFB_CICS	403	X'00000193'
MQFB_CICS_CORREL_ID_ERROR	404	X'00000194'

Tabla 162. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQFB_CICS_CCSID_ERROR	405	X'00000195'
MQFB_CICS_ERROR_DE_CODIFICACIÓN	406	X'00000196'
MQFB_CICS_CIH_ERROR	407	X'00000197'
MQFB_CICS_UOW_ERROR	408	X'00000198'
MQFB_CICS_COMMAREA_ERROR	409	X'00000199'
MQFB_CICS_APPL_NO_INICIADO	410	X'0000019A'
MQFB_CICS_APPL_ABENDED	411	X'0000019B'
MQFB_CICS_DLQ_ERROR	412	X'0000019C'
MQFB_CICS_UOW_RETIRADO	413	X'0000019D'
MQFB_PUBLICACIONES_A_PEDIDO	501	X'000001F5'
MQFB_SUSCRIPTOR_ES_EDITOR	502	X'000001F6'
MQFB_MSG_SCOPE_MISMATCH	503	X'000001F7'
MQFB_SELECTOR_MISMATCH	504	X'000001F8'
MQFB_IMS_NACK_1A_REASON_FIRST	600	X'00000258'
MQFB_IMS_NACK_1A_REASON_LAST	855	X'00000357'
MQFB_SISTEMA_ÚLTIMO	65535	X'0000FFFF'
MQFB_APPL_FIRST	65536	X'00010000'
MQFB_APPL_ÚLTIMO	99999999	X'3B9AC9FF'

MQFC_* (Opciones de formato de comando)

Tabla 163. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQFC_SÍ	1	X'00000001'
MQFC_NO	0	X'00000000'

MQFMT_* (Formatos)

Tabla 164. Nombres y valores constantes	
Nombre	Valor
MQFMT_NONE	"~~~~~"
MQFMT_ADMIN	"MQADMIN~"
MQFMT_CHANNEL_COMPLETED	"MQCHCOM~"
MQFMT_CICS	"MQCICS~"
MQFMT_COMMAND_1	"MQCMD1~"
MQFMT_COMMAND_2	"MQCMD2~"
MQFMT_DEAD_LETTER_HEADER	"MQDEAD~"
MQFMT_DIST_HEADER	"MQHDIST~"
MQFMT_EMBEDDED_PCF	"MQHEPCF~"
MQFMT_EVENT	"MQEVENT~"
MQFMT_IMS	"MQIMS~"

Tabla 164. Nombres y valores constantes (continuación)	
Nombre	Valor
MQFMT_IMS_VAR_STRING	"MQIMSVS¬"
MQFMT_MD_EXTENSION	"MQHMDE¬¬"
MQFMT_PCF	"MQPCF¬¬¬"
MQFMT_REF_MSG_HEADER	"MQHREF¬¬"
MQFMT_RF_HEADER	"MQHRF¬¬¬"
MQFMT_RF_HEADER_1	"MQHRF¬¬¬"
MQFMT_RF_HEADER_2	"MQHRF2¬¬"
MQFMT_STRING	"MQSTR¬¬¬"
MQFMT_TRIGGER	"MQTRIG¬¬"
MQFMT_WORK_INFO_HEADER	"MQHWIH¬¬"
MQFMT_XMIT_Q_HEADER	"MQXMIT¬¬"
MQFMT_NONE_ARRAY	'¬','¬','¬','¬','¬','¬','¬','¬'
MQFMT_ADMIN_MATRIZ	'M','Q','A','D','M','I','N','¬'
MQFMT_CANAL_COMPLETADO_MATRIZ	'M','Q','C','H','C','O','M','¬'
MQFMT_CICS_ARRAY	'M','Q','C','I','C','S','¬','¬'
MQFMT_COMMAND_1_ARRAY	'M','Q','C','M','D','1','¬','¬'
MQFMT_COMMAND_2_ARRAY	'M','Q','C','M','D','2','¬','¬'
MQFMT_CARTA_MUERTA_ENCABEZADO_MATRIZ	'M','Q','D','E','A','D','¬','¬'
MQFMT_DIST_HEADER_ARRAY	'M','Q','H','D','I','S','T','¬'
MQFMT_EMBEDDED_PCF_ARRAY	'M','Q','H','E','P','C','F','¬'
MQFMT_EVENTO_MATRIZ	'M','Q','E','V','E','N','T','¬'
MQFMT_IMS_MATRIZ	'M','Q','I','M','S','¬','¬','¬'
MQFMT_IMS_VAR_STRING_ARRAY	'M','Q','I','M','S','V','S','¬'
MQFMT_MD_EXTENSION_ARRAY	'M','Q','H','M','D','E','¬','¬'
MQFMT_PCF_MATRIZ	'M','Q','P','C','F','¬','¬','¬'
MQFMT_REF_MSG_HEADER_ARRAY	'M','Q','H','R','E','F','¬','¬'
MQFMT_RF_ENCABEZADO_MATRIZ	'M','Q','H','R','F','¬','¬','¬'
MQFMT_RF_HEADER_1_ARRAY	'M','Q','H','R','F','¬','¬','¬'
MQFMT_RF_HEADER_2_ARRAY	'M','Q','H','R','F','2','¬','¬'
MQFMT_STRING_ARRAY	'M','Q','S','T','R','¬','¬','¬'
MQFMT_TRIGGER_ARRAY	'M','Q','T','R','I','G','¬','¬'
MQFMT_WORK_INFO_HEADER_ARRAY	'M','Q','H','W','I','H','¬','¬'
MQFMT_XMIT_Q_HEADER_ARRAY	'M','Q','X','M','I','T','¬','¬'

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

MQFUN_* (Tipos de funciones de aplicación)

Tabla 165. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQFUN_TIPO_DESCONOCIDO	0	X'00000000'

Tabla 165. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQFUN_TIPO_JVM	1	X'00000001'
MQFUN_TIPO_PROGRAMA	2	X'00000002'
MQFUN_TIPO_PROCEDIMIENTO	3	X'00000003'
MQFUN_TIPO_USUARIO_DEFINIDO	4	X'00000004'
MQFUN_TIPO_COMANDO	5	X'00000005'

MQGA_* (Selektors de atributos de grupo)

Tabla 166. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQGA_PRIMERO	8001	X'00001F41'
MQGA_ÚLTIMO	9000	X'00002328'

MQGACF_* (Tipos de parámetros de grupo de formato de comando)

Tabla 167. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQGACF_PRIMERO	8001	X'00001F41'
MQGACF_CONTEXTO_DE_COMANDA	8001	X'00001F41'
MQGACF_DATOS_DE_COMANDA	8002	X'00001F42'
MQGACF TRASLADO_DE_RUTA	8003	X'00001F43'
MQGACF_OPERACIÓN	8004	X'00001F44'
MQGACF_ACTIVIDAD	8005	X'00001F45'
MQGACF_MQMD_INTEGRADO	8006	X'00001F46'
MQGACF_MENSAJE	8007	X'00001F47'
MQGACF_MQMD	8008	X'00001F48'
MQGACF_NOMBRAMIENTO_DE_VALORES	8009	X'00001F49'
MQGACF_Q_ACCOUNTING_DATA	8010	X'00001F4A'
MQGACF_Q_STATISTICS_DATA	8011	X'00001F4B'
MQGACF_CHL_ESTADÍSTICAS_DATOS	8012	X'00001F4C'
MQGACF_ÚLTIMO_USADO	8012	X'00001F4C'

MQGI_* (Identificador de grupo)

Tabla 168. Nombres y valores constantes	
Nombre	Valor
MQGI_NONE	X'00...00' (24 nulos)
MQGI_NINGUNA_MATRIZ	'\0','\0',... (24 nulos)

MQGMO_* (Obtener opciones y estructura del mensaje)

Obtener estructura de opciones de mensajes

Tabla 169. Estructuras de constantes	
Nombre	Estructura
MQGMO_STRUC_ID	"GMO¬"
MQGMO_STRUC_ID_ARRAY	'G','M','O','¬'

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

Tabla 170. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQGMO_VERSION_1	1	X'00000001'
MQGMO_VERSION_2	2	X'00000002'
MQGMO_VERSION_3	3	X'00000003'
MQGMO_VERSION_4	4	X'00000004'
MQGMO_VERSIÓN_ACTUAL	4	X'00000004'

Obtener opciones de mensaje

Tabla 171. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQGMO_WAIT	1	X'00000001'
MQGMO_NO_WAIT	0	X'00000000'
MQGMO_SET_SIGNAL	8	X'00000008'
MQGMO_FAIL_IF QUIESCING	8192	X'00002000'
MQGMO_SYNCPOINT	2	X'00000002'
MQGMO_SYNCPOINT_IF_PERSISTENT	4096	X'00001000'
MQGMO_NO_SYNCPOINT	4	X'00000004'
MQGMO_MARK_SKIP_BACKOUT	128	X'00000080'
MQGMO_BROWSE_FIRST	16	X'00000010'
MQGMO_BROWSE_NEXT	32	X'00000020'
MQGMO_BROWSE_MSG_UNDER_CURSOR	2048	X'00008000'
MQGMO_BROWSE_HANDLE	17825808	X'01100010'
MQGMO_EXPLORAR_COOPERATIVA	18874384	X'01200010'
MQGMO_MSG_UNDER_CURSOR	256	X'00000100'
MQGMO_LOCK	512	X'00000200'
MQGMO_UNLOCK	1024	X'00000400'
MQGMO_ACCEPT_TRUNCATED_MSG	64	X'00000040'
MQGMO_CONVERT	16384	X'00004000'
MQGMO_LOGICAL_ORDER	32768	X'00008000'
MQGMO_COMPLETE_MSG	65536	X'00010000'
MQGMO_ALL_MSGS_AVAILABLE	131072	X'00020000'
MQGMO_ALL_SEGMENTS_AVAILABLE	262144	X'00040000'

Tabla 171. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQGMO_MARK_BROWSE_HANDLE	1048576	X'00100000'
MQGMO_MARK_BROWSE_CO_OP	2097152	X'00200000'
MQGMO_UNMARK_BROWSE_CO_OP	4194304	X'00400000'
MQGMO_UNMARK_BROWSE_HANDLE	8388608	X'00800000'
MQGMO_UNMARKED_BROWSE_MSG	16777216	X'01000000'
MQGMO_PROPERTIES_FORCE_MQRFH2	33554432	X'02000000'
MQGMO_NO_PROPERTIES	67108864	X'04000000'
MQGMO_PROPERTIES_IN_HANDLE	134217728	X'08000000'
MQGMO_PROPERTIES_COMPATIBILITY	268435456	X'10000000'
MQGMO_PROPERTIES_AS_Q_DEF	0	X'00000000'
MQGMO_NONE	0	X'00000000'

MQGS_* (Estado del grupo)

Tabla 172. Nombres y valores constantes	
Nombre	Valor
MQGS_NO_EN_GRUPO	'¬'
MQGS_MSG_EN_GRUPO	'G'
Último mensaje en el grupo	'L'

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

MQHA_* (Selektoren für Handles)

Tabla 173. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQHA_PRIMERO	4001	X'00000FA1'
MQHA_ASA_DE_BOLSO	4001	X'00000FA1'
MQHA_ÚLTIMO_USADO	4001	X'00000FA1'
MQHA_ÚLTIMO	6000	X'00001770'

MQHB_* (Asas para bolsas)

Tabla 174. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQHB_INUTILIZABLE_HBAG	-1	X'FFFFFFFF'
MQHB_NINGUNO	-2	X'FFFFFFFE'

MQHC_* (Identificadores de conexión)

Tabla 175. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQHC_DEF_HCONN	0	X'00000000'
MQHC_CONEXIÓN_INUTILIZABLE	-1	X'FFFFFFFF'

Tabla 175. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQHC_NO_ASOCIADO_HCONN	-3	X'FFFFFFFFD'

MQHM_* (Identificador de mensaje)

Tabla 176. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQHM_INUTILIZABLE_HMSG	-1	X'FFFFFFFF'
MQHM_NINGUNO	0	X'00000000'

MQHO_* (Identificador de objeto)

Tabla 177. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQHO_OBJETO_INUTILIZABLE	-1	X'FFFFFFFF'
MQHO_NONE	0	X'00000000'

MQHSTATE_* (Formato de comando Estados de control)

Tabla 178. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQHSTATE_INACTIVO	0	X'00000000'
MQHSTATE_ACTIVE	1	X'00000001'

MQIA_* (Selektors de atributos enteros)

Tabla 179. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQIA_ACCOUNTING_CONN_OVERRIDE	136	X'00000088'
MQIA_ACCOUNTING_INTERVAL	135	X'00000087'
MQIA_ACCOUNTING_MQI	133	X'00000085'
MQIA_ACCOUNTING_Q	134	X'00000086'
MQIA_ACTIVE_CHANNELS	100	X'00000064'
MQIA_ACTIVITY_CONN_OVERRIDE	239	X'000000EF'
MQIA_ACTIVITY_RECORDING	138	X'0000008A'
MQIA_ACTIVITY_TRACE	240	X'000000F0'
MQIA_ADOPTNEWMCA_CHECK	102	X'00000066'
MQIA_ADOPTNEWMCA_INTERVAL	104	X'00000068'
MQIA_ADOPTNEWMCA_TYPE	103	X'00000067'
MQIA_ADOPT_CONTEXT	260	X'00000104'
 MQIA_ADVANCED_CAPABILITY	273	X'00000111'
MQIA_AMQP_CAPABILITY	265	X'00000109'
MQIA_APPL_TYPE	1	X'00000001'
MQIA_ARCHIVE	60	X'0000003C'

Tabla 179. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQIA_AUTHENTICATION_FAIL_DELAY	259	X'00000103'
MQIA_AUTHENTICATION_METHOD	266	X'0000010A'
MQIA_AUTH_INFO_TYPE	66	X'00000042'
MQIA_AUTHOREV_SCOPE	277	X'00000115'
MQIA_AUTHORITY_EVENT	47	X'0000002F'
MQIA_AUTO_REORG_INTERVAL	174	X'000000AE'
MQIA_AUTO_REORGANIZATION	173	X'000000AD'
MQIA_BACKOUT_THRESHOLD	22	X'00000016'
MQIA_BASE_TYPE	193	X'000000C1'
MQIA_BATCH_INTERFACE_AUTO	86	X'00000056'
MQIA_BRIDGE_EVENT	74	X'0000004A'
MQIA_CF_LEVEL	70	X'00000046'
MQIA_CF_RECOVER	71	X'00000047'
MQIA_CHANNEL_AUTO_DEF	55	X'00000037'
MQIA_CHANNEL_AUTO_DEF_EVENT	56	X'00000038'
MQIA_CHANNEL_EVENT	73	X'00000049'
MQIA_CHECK_CLIENT_BINDING	258	X'00000102'
MQIA_CHECK_LOCAL_BINDING	257	X'00000101'
MQIA_CHINIT_ADAPTERS	101	X'00000065'
MQIA_CHINIT_CONTROL	119	X'00000077'
MQIA_CHINIT_DISPATCHERS	105	X'00000069'
MQIA_CHINIT_TRACE_AUTO_START	117	X'00000075'
MQIA_CHINIT_TRACE_TABLE_SIZE	118	X'00000076'
MQIA_CLUSTER_OBJECT_STATE	256	X'00000100'
MQIA_CLUSTER_PUB_ROUTE	255	X'000000FF'
MQIA_CLUSTER_Q_TYPE	59	X'0000003B'
MQIA_CLUSTER_WORKLOAD_LENGTH	58	X'0000003A'
MQIA_CLWL_MRU_CHANNELS	97	X'00000061'
MQIA_CLWL_Q_RANK	95	X'0000005F'
MQIA_CLWL_Q_PRIORITY	96	X'00000060'
MQIA_CLWL_USEQ	98	X'00000062'
MQIA_CMD_SERVER_AUTO	87	X'00000057'
MQIA_CMD_SERVER_CONTROL	120	X'00000078'
MQIA_CMD_SERVER_CONVERT_MSG	88	X'00000058'
MQIA_CMD_SERVER_DLQ_MSG	89	X'00000059'
MQIA_CODED_CHAR_SET_ID	2	X'00000002'
MQIA_COMM_EVENT	232	X'000000E8'
MQIA_COMMAND_EVENT	99	X'00000063'
MQIA_COMMAND_LEVEL	31	X'0000001F'

Tabla 179. Valores de las constantes (continuación)

Nombre	Valor decimal	Valor hexadecimal
MQIA_CONFIGURATION_EVENT	51	X'00000033'
MQIA_CPI_LEVEL	27	X'0000001B'
MQIA_CURRENT_Q_DEPTH	3	X'00000003'
MQIA_DEF_BIND	61	X'0000003D'
MQIA_DEF_CLUSTER_XMIT_Q_TYPE	250	X'000000FA'
MQIA_DEF_INPUT_OPEN_OPTION	4	X'00000004'
MQIA_DEF_PERSISTENCE	5	X'00000005'
MQIA_DEF_PRIORITY	6	X'00000006'
MQIA_DEF_PUT_RESPONSE_TYPE	184	X'000000B8'
MQIA_DEF_READ_AHEAD	188	X'000000BC'
MQIA_DEFINITION_TYPE	7	X'00000007'
MQIA_DISPLAY_TYPE	262	X'00000106'
MQIA_DIST_LISTS	34	X'00000022'
MQIA_DNS_WLM	106	X'0000006A'
MQIA_DURABLE_SUB	175	X'000000AF'
MQIA_EXPIRY_INTERVAL	39	X'00000027'
MQIA_FIRST	1	X'00000001'
 MQIA_GROUP_UR	221	X'000000DD'
MQIA_HARDEN_GET_BACKOUT	8	X'00000008'
MQIA_HIGH_Q_DEPTH	36	X'00000024'
MQIA_IGQ_PUT_AUTHORITY	65	X'00000041'
MQIA_INDEX_TYPE	57	X'00000039'
MQIA_INHIBIT_EVENT	48	X'00000030'
MQIA_INHIBIT_GET	9	X'00000009'
MQIA_INHIBIT_PUB	181	X'000000B5'
MQIA_INHIBIT_PUT	10	X'0000000A'
MQIA_INHIBIT_SUB	182	X'000000B6'
 MQIA_INTRA_GROUP_QUEUING	64	X'00000040'
MQIA_IP_ADDRESS_VERSION	93	X'0000005D'
MQIA_KEY_REUSE_COUNT	267	X'0000010B'
MQIA_LAST	2000	X'000007D0'
MQIA_LAST_USED	277	X'00000115'
MQIA_LDAP_AUTHORMD	263	X'00000107'
MQIA_LDAP_NESTGRP	264	X'00000108'
MQIA_LDAP_SECURE_COMM	261	X'00000105'
MQIA_LISTENER_PORT_NUMBER	85	X'00000055'
MQIA_LISTENER_TIMER	107	X'0000006B'
MQIA_LOGGER_EVENT	94	X'0000005E'
MQIA_LU62_CHANNELS	108	X'0000006C'

Tabla 179. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQIA_LOCAL_EVENT	49	X'00000031'
MQIA_MSG_MARK_BROWSE_INTERVAL	68	X'00000044'
MQIA_MAX_CHANNELS	109	X'0000006D'
MQIA_MAX_CLIENTS	172	X'000000AC'
MQIA_MAX_GLOBAL_LOCKS	83	X'00000053'
MQIA_MAX_HANDLES	11	X'0000000B'
MQIA_MAX_LOCAL_LOCKS	84	X'00000054'
MQIA_MAX_MSG_LENGTH	13	X'0000000D'
MQIA_MAX_OPEN_Q	80	X'00000050'
MQIA_MAX_PRIORITY	14	X'0000000E'
MQIA_MAX_PROPERTIES_LENGTH	192	X'000000C0'
MQIA_MAX_Q_DEPTH	15	X'0000000F'
MQIA_MAX_Q_TRIGGERS	90	X'0000005A'
MQIA_MAX_RECOVERY_TASKS	171	X'000000AB'
MQIA_MAX_UNCOMMITTED_MSGS	33	X'00000021'
MQIA_MCAST_BRIDGE	233	X'000000E9'
MQIA_MONITOR_INTERVAL	81	X'00000051'
MQIA_MONITORING_AUTO_CLUSSDR	124	X'0000007C'
MQIA_MONITORING_CHANNEL	122	X'0000007A'
MQIA_MONITORING_Q	123	X'0000007B'
MQIA_MSG_DELIVERY_SEQUENCE	16	X'00000010'
MQIA_MSG_DEQ_COUNT	38	X'00000026'
MQIA_MSG_ENQ_COUNT	37	X'00000025'
MQIA_NAME_COUNT	19	X'00000013'
MQIA_NAMELIST_TYPE	72	X'00000048'
MQIA_NPM_CLASS	78	X'0000004E'
MQIA_NPM_DELIVERY	196	X'000000C4'
MQIA_OPEN_INPUT_COUNT	17	X'00000011'
MQIA_OPEN_OUTPUT_COUNT	18	X'00000012'
MQIA_OUTBOUND_PORT_MAX	140	X'0000008C'
MQIA_OUTBOUND_PORT_MIN	110	X'0000006E'
MQIA_PAGESET_ID	62	X'0000003E'
MQIA_PERFORMANCE_EVENT	53	X'00000035'
MQIA_PLATFORM	32	X'00000020'
MQIA_PM_DELIVERY	195	X'000000C3'
MQIA_PROPERTY_CONTROL	190	X'000000BE'
MQIA_PROT_POLICY_CAPABILITY	251	X'000000FB'
MQIA_PROXY_SUB	199	X'000000C7'
MQIA_PUB_COUNT	215	X'000000D7'

Tabla 179. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQIA_PUB_SCOPE	219	X'000000DB'
MQIA_PUBSUB_CLUSTER	249	X'000000F9'
MQIA_PUBSUB_MAXMSG_RETRY_COUNT	206	X'000000CE'
MQIA_PUBSUB_MODE	187	X'000000BB'
MQIA_PUBSUB_NP_MSG	203	X'000000CB'
MQIA_PUBSUB_NP_RESP	205	X'000000CD'
MQIA_PUBSUB_SYNC_PT	207	X'000000CF'
MQIA_Q_DEPTH_HIGH_EVENT	43	X'0000002B'
MQIA_Q_DEPTH_HIGH_LIMIT	40	X'00000028'
MQIA_Q_DEPTH_LOW_EVENT	44	X'0000002C'
MQIA_Q_DEPTH_LOW_LIMIT	41	X'00000029'
MQIA_Q_DEPTH_MAX_EVENT	42	X'0000002A'
MQIA_Q_SERVICE_INTERVAL	54	X'00000036'
MQIA_Q_SERVICE_INTERVAL_EVENT	46	X'0000002E'
MQIA_Q_TYPE	20	X'00000014'
MQIA_Q_USERS	82	X'00000052'
MQIA_QMGR_CFCNLOS	245	X'000000F5'
MQIA_QMOPT_CONS_COMMS_MSGS	155	X'0000009B'
MQIA_QMOPT_CONS_CRITICAL_MSGS	154	X'0000009A'
MQIA_QMOPT_CONS_ERROR_MSGS	153	X'00000099'
MQIA_QMOPT_CONS_INFO_MSGS	151	X'00000097'
MQIA_QMOPT_CONS_REORG_MSGS	156	X'0000009C'
MQIA_QMOPT_CONS_SYSTEM_MSGS	157	X'0000009D'
MQIA_QMOPT_CONS_WARNING_MSGS	152	X'00000098'
MQIA_QMOPT_CSMT_ON_ERROR	150	X'00000096'
MQIA_QMOPT_INTERNAL_DUMP	170	X'000000AA'
MQIA_QMOPT_LOG_COMMS_MSGS	162	X'000000A2'
MQIA_QMOPT_LOG_CRITICAL_MSGS	161	X'000000A1'
MQIA_QMOPT_LOG_ERROR_MSGS	160	X'000000A0'
MQIA_QMOPT_LOG_INFO_MSGS	158	X'0000009E'
MQIA_QMOPT_LOG_REORG_MSGS	163	X'000000A3'
MQIA_QMOPT_LOG_SYSTEM_MSGS	164	X'000000A4'
MQIA_QMOPT_LOG_WARNING_MSGS	159	X'0000009F'
MQIA_QMOPT_TRACE_COMMS	166	X'000000A6'
MQIA_QMOPT_TRACE_CONVERSION	168	X'000000A8'
MQIA_QMOPT_TRACE_REORG	167	X'000000A7'
MQIA_QMOPT_TRACE_MQI_CALLS	165	X'000000A5'
MQIA_QMOPT_TRACE_SYSTEM	169	X'000000A9'
MQIA_QSG_DISP	63	X'0000003F'

Tabla 179. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQIA_READ_AHEAD	189	X'000000BD'
MQIA_RECEIVE_TIMEOUT	111	X'0000006F'
MQIA_RECEIVE_TIMEOUT_MIN	113	X'00000071'
MQIA_RECEIVE_TIMEOUT_TYPE	112	X'00000070'
MQIA_REMOTE_EVENT	50	X'00000032'
MQIA_RETENTION_INTERVAL	21	X'00000015'
MQIA_REVERSE_DNS_LOOKUP	254	X'000000FE'
MQIA_SCOPE	45	X'0000002D'
MQIA_SECURITY_CASE	141	X'0000008D'
MQIA_SERVICE_CONTROL	139	X'0000008B'
MQIA_SERVICE_TYPE	121	X'00000079'
MQIA_SHAREABILITY	23	X'00000017'
MQIA_SHARED_Q_Q_MGR_NAME	77	X'0000004D'
MQIA_SSL_EVENT	75	X'0000004B'
MQIA_SSL_FIPS_REQUIRED	92	X'0000005C'
MQIA_SSL_RESET_COUNT	76	X'0000004C'
MQIA_SSL_TASKS	69	X'00000045'
MQIA_START_STOP_EVENT	52	X'00000034'
MQIA_STATISTICS_CHANNEL	129	X'00000081'
MQIA_STATISTICS_AUTO_CLUSSDR	130	X'00000082'
MQIA_STATISTICS_INTERVAL	131	X'00000083'
MQIA_STATISTICS_MQI	127	X'0000007F'
MQIA_STATISTICS_Q	128	X'00000080'
MQIA_SUB_COUNT	204	X'000000CC'
MQIA_SUB_SCOPE	218	X'000000DA'
MQIA_SYNCPOINT	30	X'0000001E'
MQIA_TCP_CHANNELS	114	X'00000072'
MQIA_TCP_KEEP_ALIVE	115	X'00000073'
MQIA_TCP_STACK_TYPE	116	X'00000074'
MQIA_TIME_SINCE_RESET	35	X'00000023'
MQIA_TOPIC_DEF_PERSISTENCE	185	X'000000B9'
MQIA_TOPIC_NODE_COUNT	253	X'000000FD'
MQIA_TOPIC_TYPE	208	X'000000D0'
MQIA_TRACE_ROUTE_RECORDING	137	X'00000089'
MQIA_TREE_LIFE_TIME	183	X'000000B7'
MQIA_TRIGGER_CONTROL	24	X'00000018'
MQIA_TRIGGER_DEPTH	29	X'0000001D'
MQIA_TRIGGER_INTERVAL	25	X'00000019'
MQIA_TRIGGER_MSG_PRIORITY	26	X'0000001A'

Tabla 179. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQIA_TRIGGER_TYPE	28	X'0000001C'
MQIA_TRIGGER_RESTART	91	X'0000005B'
MQIA_USAGE	12	X'0000000C'
MQIA_USE_DEAD_LETTER_Q	234	X'000000EA'
MQIA_USER_LIST	2000	X'000007D0'
MQIA_WILDCARD_OPERATION	216	X'000000D8'
MQIA_XR_CAPABILITY	243	X'000000F3'

MQIACF_* (Tipos de parámetros enteros de formato de comando)

Tabla 180. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQIACF_PRIMERO	1001	X'000003E9'
MQIACF_Q_MGR_ATTRS	1001	X'000003E9'
MQIACF_Q_ATRIBUTOS	1002	X'000003EA'
MQIACF_PROCESO_ATRIBUTOS	1003	X'000003EB'
MQIACF_LISTA_DE_ATRIBUTOS_DE_NOMBRES	1004	X'000003EC'
MQIACF_FORCE	1005	X'000003ED'
MQIACF_REEMPLAZAR	1006	X'000003EE'
MQIACF_PURGA	1007	X'000003EF'
MQIACF QUIESCE	1008	X'000003F0'
MQIACF_MODO	1008	X'000003F0'
MQIACF_ALL	1009	X'000003F1'
MQIACF_EVENTO_TIPO_APLICACIÓN	1010	X'000003F2'
MQIACF_EVENTO_ORIGEN	1011	X'000003F3'
MQIACF_PARAMETER_ID	1012	X'000003F4'
MQIACF_ERROR_ID	1013	X'000003F5'
MQIACF_ERROR_IDENTIFICADOR	1013	X'000003F5'
MQIACF_SELECTOR	1014	X'000003F6'
MQIACF_ATRIBUTOS_CANAL	1015	X'000003F7'
MQIACF_OBJECT_TYPE	1016	X'000003F8'
MQIACF_TIPO_DE_ESCAPE	1017	X'000003F9'
MQIACF_ERROR_OFFSET	1018	X'000003FA'
MQIACF_AUTH_INFO_ATTRS	1019	X'000003FB'
MQIACF_MOTIVO_CALIFICADOR	1020	X'000003FC'
MQIACF_COMANDO	1021	X'000003FD'
MQIACF_OPEN_OPTIONS	1022	X'000003FE'
MQIACF_TIPO_ABIERTO	1023	X'000003FF'
MQIACF_PROCESS_ID	1024	X'00000400'
MQIACF_THREAD_ID	1025	X'00000401'

Tabla 180. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQIACF_Q_ATRIBUTOS_ESTADO	1026	X'00000402'
MQIACF_MENSAJES_NO_ASIGNADOS	1027	X'00000403'
MQIACF_HANDLE_STATE	1028	X'00000404'
MQIACF_AUX_ERROR_DATA_INT_1	1070	X'0000042E'
MQIACF_AUX_ERROR_DATA_INT_2	1071	X'0000042F'
MQIACF_CONV_MOTIVO_CÓDIGO	1072	X'00000430'
MQIACF_TIPO_DE_PUENTE	1073	X'00000431'
MQIACF_CONSULTA	1074	X'00000432'
MQIACF_WAIT_INTERVAL	1075	X'00000433'
MQIACF_OPCIONES	1076	X'00000434'
MQIACF_BROKER_OPCIONES	1077	X'00000435'
MQIACF_TIPO_ACTUALIZACIÓN	1078	X'00000436'
MQIACF_SEQUENCE_NUMBER	1079	X'00000437'
MQIACF_DATOS_ENTEROS	1080	X'00000438'
MQIACF_OPCIONES_DE_REGISTRO	1081	X'00000439'
MQIACF_OPCIONES_DE_PUBLICACIÓN	1082	X'0000043A'
MQIACF_INFORMACIÓN_DE_CLÚSTER	1083	X'0000043B'
MQIACF_Q_MGR_TIPO_DEFINICIÓN	1084	X'0000043C'
MQIACF_Q_MGR_TIPO	1085	X'0000043D'
MQIACF_ACCIÓN	1086	X'0000043E'
MQIACF_SUSPENDER	1087	X'0000043F'
MQIACF_NÚMERO_DE_CORREDORES	1088	X'00000440'
MQIACF_APPL_COUNT	1089	X'00000441'
MQIACF_CONTE ANÓNIMO	1090	X'00000442'
MQIACF_REG_REG_OPCIONES	1091	X'00000443'
MQIACF_ELIMINAR_OPCIONES	1092	X'00000444'
MQIACF_CLUSTER_Q_MGR_ATTRS	1093	X'00000445'
MQIACF_INTERVALO_DE_ACTUALIZACIÓN	1094	X'00000446'
MQIACF_ACTUALIZAR_REPOSITORIO	1095	X'00000447'
MQIACF_ELIMINAR_COLAS	1096	X'00000448'
MQIACF_TIPO_DE_ENTRADA_ABIERTA	1098	X'0000044A'
MQIACF_ABRIR_SALIDA	1099	X'0000044B'
MQIACF_ABRIR_CONJUNTO	1100	X'0000044C'
MQIACF_ABRIR_CONSULTA	1101	X'0000044D'
MQIACF_ABRIR_EXPLORAR	1102	X'0000044E'
MQIACF_Q_TIPO_ESTADO	1103	X'0000044F'
MQIACF_Q_HANDLE	1104	X'00000450'
MQIACF_Q_ESTADO	1105	X'00000451'
MQIACF_TIPO_DE_SEGURIDAD	1106	X'00000452'

Tabla 180. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQIACF_CONEXIÓN_ATRIBUTOS	1107	X'00000453'
MQIACF_CONNECT_OPTIONS	1108	X'00000454'
MQIACF_CONN_INFO_TYPE	1110	X'00000456'
MQIACF_CONN_INFO_CONN	1111	X'00000457'
MQIACF_CONN_INFO_HANDLE	1112	X'00000458'
MQIACF_CONN_INFO_ALL	1113	X'00000459'
MQIACF_AUTH_PERFIL_ATRIBUTOS	1114	X'0000045A'
Lista de autorizaciones MQIACF	1115	X'0000045B'
MQIACF_AUTH_ADD_AUTHS	1116	X'0000045C'
MQIACF_AUTH_REMOVE_AUTHS	1117	X'0000045D'
MQIACF_TIPO_ENTIDAD	1118	X'0000045E'
MQIACF_COMANDO_INFORMACIÓN	1120	X'00000460'
MQIACF_CMDSCOPE_Q_MGR_COUNT	1121	X'00000461'
MQIACF_Q_MGR_SISTEMA	1122	X'00000462'
MQIACF_Q_MGR_EVENTO	1123	X'00000463'
MQIACF_Q_MGR_DQM	1124	X'00000464'
MQIACF_Q_MGR_CLUSTER	1125	X'00000465'
MQIACF_QSG_DISPS	1126	X'00000466'
MQIACF_UOW_ESTADO	1128	X'00000468'
MQIACF_ELEMENTO_DE_SEGURIDAD	1129	X'00000469'
MQIACF_CF_ESTRUCTURA_ESTADO	1130	X'0000046A'
MQIACF_UOW_TIPO	1132	X'0000046C'
MQIACF_CF_ESTRUCTURA_ATRIBUTOS	1133	X'0000046D'
MQIACF_EXCLUIR_INTERVALO	1134	X'0000046E'
MQIACF_CF_TIPO_ESTADO	1135	X'0000046F'
Resumen del estado de MQIACF_CF	1136	X'00000470'
MQIACF_CF_ESTADO_CONEXIÓN	1137	X'00000471'
MQIACF_CF_ESTADO_COPIA_DE_RESERVA	1138	X'00000472'
MQIACF_CF_TIPO_ESTRUCTURA	1139	X'00000473'
MQIACF_CF_STRUC_TAMAÑO_MÁXIMO	1140	X'00000474'
MQIACF_CF_STRUC_TAMAÑO_UTILIZADO	1141	X'00000475'
MQIACF_CF_ESTRUCTURA_ENTRADAS_MÁXIMO	1142	X'00000476'
MQIACF_CF_ESTRUCTURA_ENTRADAS_UTILIZADAS	1143	X'00000477'
MQIACF_CF_STRUC_TAMAÑO_COPIA_DE_RESERVA	1144	X'00000478'
MQIACF_MOVIMIENTO_TIPO	1145	X'00000479'
MQIACF_MOVIMIENTO_TIPO_MOVIMIENTO	1146	X'0000047A'
MQIACF_MOVIMIENTO_TIPO_AÑADIR	1147	X'0000047B'
MQIACF_Q_MGR_NÚMERO	1148	X'0000047C'
MQIACF_Q_MGR_STATUS	1149	X'0000047D'

Tabla 180. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQIACF_DB2_CONN_STATUS	1150	X'0000047E'
MQIACF_ATRIBUTOS_DE_SEGURIDAD	1151	X'0000047F'
MQIACF_SECURITY_TIMEOUT	1152	X'00000480'
MQIACF_INTERVALO_DE_SEGURIDAD	1153	X'00000481'
MQIACF_INTERRUPTOR_DE_SEGURIDAD	1154	X'00000482'
MQIACF_CONFIGURACIÓN_DE_SEGURIDAD	1155	X'00000483'
MQIACF_CLASE_DE_ALMACENAMIENTO_ATRIBUTOS	1156	X'00000484'
MQIACF_TIPO_DE_USO	1157	X'00000485'
MQIACF_BUFFER_POOL_ID	1158	X'00000486'
MQIACF_USO_TOTAL_PÁGINAS	1159	X'00000487'
MQIACF_USO_PÁGINAS_NO_UTILIZADAS	1160	X'00000488'
MQIACF_USO_PÁGINAS_PERSISTENTES	1161	X'00000489'
MQIACF_USO_PÁGINAS_NO_PERSISTENTES	1162	X'0000048A'
MQIACF_USO_REINICIO_EXTENSIONES	1163	X'0000048B'
MQIACF_USAGE_EXPAND_COUNT	1164	X'0000048C'
MQIACF_PÁGINAS_ESTADO	1165	X'0000048D'
MQIACF_USO_TOTAL_AMORTIGUADORES	1166	X'0000048E'
MQIACF_TIPO_DE_CONJUNTO_DE_DATOS_DE_USO	1167	X'0000048F'
MQIACF_USAGE_PAGESET	1168	X'00000490'
MQIACF_USAGE_DATA_SET	1169	X'00000491'
MQIACF_USAGE_BUFFER_POOL	1170	X'00000492'
MQIACF_MOVIMIENTO_CONTADOR	1171	X'00000493'
MQIACF_EXPIRY_Q_COUNT	1172	X'00000494'
MQIACF_CONFIGURACIÓN_OBJETOS	1173	X'00000495'
MQIACF_CONFIGURACIÓN_EVENTOS	1174	X'00000496'
MQIACF_TIPO_SISTEMA	1175	X'00000497'
MQIACF_SYSP_DEALLOC_INTERVAL	1176	X'00000498'
MQIACF_SYSP_MAX_ARCHIVO	1177	X'00000499'
MQIACF_SYSP_MÁXIMO_Lectura_CINTAS	1178	X'0000049A'
MQIACF_SYSP_TAMAÑO_DEL_BUFFER_DE_ENTRADA	1179	X'0000049B'
MQIACF_SYSP_OUT_BUFFER_SIZE	1180	X'0000049C'
MQIACF_SYSP_OUT_BUFFER_COUNT	1181	X'0000049D'
MQIACF_SYSP_ARCHIVO	1182	X'0000049E'
MQIACF_SYSP_DUAL_ACTIVO	1183	X'0000049F'
MQIACF_SYSP_ARCHIVO_DUAL	1184	X'000004A0'
MQIACF_SYSP_DUAL_BSDS	1185	X'000004A1'
MQIACF_SYSP_MAX_CONNS	1186	X'000004A2'
MQIACF_SYSP_MAX_CONNS_FORE	1187	X'000004A3'
MQIACF_SYSP_MAX_CONNS_BACK	1188	X'000004A4'

Tabla 180. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQIACF_SYSP_EXIT_INTERVAL	1189	X'000004A5'
MQIACF_SYSP_SALIR_TAREAS	1190	X'000004A6'
MQIACF_SYSP_CHKPOINT_COUNT	1191	X'000004A7'
MQIACF_SYSP_OTMA_INTERVAL	1192	X'000004A8'
MQIACF_SYSP_Q_ÍNDICE_APLAZAR	1193	X'000004A9'
MQIACF_SYSP_DB2_TASKS	1194	X'000004AA'
MQIACF_SYSP_RESLEVEL_AUDIT	1195	X'000004AB'
MQIACF_SYSP_CÓDIGO_DE_ENRUTAMIENTO	1196	X'000004AC'
MQIACF_SYSP_SMF_CONTABILIDAD	1197	X'000004AD'
MQIACF_SYSP_SMF_ESTADÍSTICAS	1198	X'000004AE'
MQIACF_SYSP_SMF_INTERVAL	1199	X'000004AF'
MQIACF_SYSP_TRACE_CLASS	1200	X'000004B0'
MQIACF_SYSP_TRACE_SIZE	1201	X'000004B1'
MQIACF_SYSP_WLM_INTERVAL	1202	X'000004B2'
MQIACF_SYSP_ALLOC_UNIT	1203	X'000004B3'
MQIACF_SYSP_ARCHIVE_RETAIN	1204	X'000004B4'
MQIACF_SYSP_ARCHIVO_WTOR	1205	X'000004B5'
MQIACF_SYSP_TAMAÑO_BLOQUE	1206	X'000004B6'
MQIACF_SYSP_CATÁLOGO	1207	X'000004B7'
MQIACF_SYSP_COMPACT	1208	X'000004B8'
MQIACF_SYSP_ASIGNACIÓN_PRIMARIA	1209	X'000004B9'
MQIACF_SYSP_ASIGNACIÓN_SECUNDARIA	1210	X'000004BA'
MQIACF_SYSP_PROTECT	1211	X'000004BB'
MQIACF_SYSP_INTERVALO_DE_INACTIVIDAD	1212	X'000004BC'
MQIACF_SYSP_TIMESTAMP	1213	X'000004BD'
MQIACF_SYSP_DIRECCIÓN_DE_UNIDAD	1214	X'000004BE'
MQIACF_SYSP_ESTADO_UNIDAD	1215	X'000004BF'
MQIACF_SYSP_LOG_COPY	1216	X'000004C0'
MQIACF_SYSP_LOG_USED	1217	X'000004C1'
MQIACF_SYSP_LOG_SUSPEND	1218	X'000004C2'
MQIACF_SYSP_DESCARGA_ESTADO	1219	X'000004C3'
MQIACF_SYSP_TOTAL_LOGS	1220	X'000004C4'
MQIACF_SYSP_REGISTROS_COMPLETOS	1221	X'000004C5'
MQIACF_LISTENER_ATTRS	1222	X'000004C6'
MQIACF_LISTENER_STATUS_ATTRS	1223	X'000004C7'
MQIACF_SERVICIO_ATRIBUTOS	1224	X'000004C8'
MQIACF_ESTADO_DEL_SERVICIO_ATRIBUTOS	1225	X'000004C9'
MQIACF_Q_INDICADOR_DE_TIEMPO	1226	X'000004CA'
MQIACF_EDAD_MÁXIMA_MENSAJE	1227	X'000004CB'

Tabla 180. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQIACF_AUTH_OPTIONS	1228	X'000004CC'
MQIACF_Q_MGR_STATUS_ATTRS	1229	X'000004CD'
MQIACF_CONSTRUCCIÓN_CONEXIÓN	1230	X'000004CE'
MQIACF_Q_MGR_INSTALACIÓN	1231	X'000004CF'
MQIACF_CHINIT_STATUS	1232	X'000004D0'
MQIACF_CMD_SERVIDOR_ESTADO	1233	X'000004D1'
MQIACF_DETALLE_DE_LA_RUTA	1234	X'000004D2'
MQIACF_ACTIVIDADES_REGISTRADAS	1235	X'000004D3'
MQIACF_MÁXIMO_ACTIVIDADES	1236	X'000004D4'
MQIACF_CONTABILIZACIÓN_DE_DISCONTINUIDAD	1237	X'000004D5'
MQIACF_ACUMULACIÓN_DE_RUTAS	1238	X'000004D6'
MQIACF_RUTA_ENTREGA	1239	X'000004D7'
MQIACF_TIPO_DE_OPERACIÓN	1240	X'000004D8'
MQIACF_RECUENTO_DE_RETIRADAS	1241	X'000004D9'
MQIACF_COMP_CODE	1242	X'000004DA'
MQIACF_ENCODING	1243	X'000004DB'
MQIACF_EXPIRY	1244	X'000004DC'
MQIACF_FEEDBACK	1245	X'000004DD'
MQIACF_MSG_FLAGS	1247	X'000004DF'
MQIACF_MSG_LENGTH	1248	X'000004E0'
MQIACF_MSG_TYPE	1249	X'000004E1'
MQIACF_OFFSET	1250	X'000004E2'
MQIACF_LONGITUD_ORIGINAL	1251	X'000004E3'
MQIACF_PERSISTENCE	1252	X'000004E4'
MQIACF_PRIORITY	1253	X'000004E5'
MQIACF_REASON_CODE	1254	X'000004E6'
MQIACF_REPORT	1255	X'000004E7'
MQIACF VERSIÓN	1256	X'000004E8'
MQIACF_ACTIVIDADES_NO_REGISTRADAS	1257	X'000004E9'
MQIACF_SUPERVISIÓN	1258	X'000004EA'
MQIACF_REENVÍO_DE_RUTAS	1259	X'000004EB'
MQIACF_ESTADO_DEL_SERVICIO	1260	X'000004EC'
MQIACF_Q_TIPOS	1261	X'000004ED'
MQIACF_ID_DE_USUARIO_SOPORTE	1262	X'000004EE'
MQIACF VERSIÓN INTERFAZ	1263	X'000004EF'
MQIACF_AUTH_SERVICE_ATTRS	1264	X'000004F0'
MQIACF_USO_TIPO_EXPANDIR	1265	X'000004F1'
MQIACF_SYSP_CLUSTER_CACHE	1266	X'000004F2'
MQIACF_SYSP_DB2_BLOB_TASKS	1267	X'000004F3'

Tabla 180. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQIACF_SYSP_WLM_INT_UNITS	1268	X'000004F4'
MQIACF_ATRIBUTOS_DE_TEMA	1269	X'000004F5'
MQIACF_PUBSUB_PROPIEDADES	1271	X'000004F7'
MQIACF_CLASE_DESTINO	1273	X'000004F9'
MQIACF_SUSCRIPCIÓN_DURADERA	1274	X'000004FA'
MQIACF_ÁMBITO_DE_SUSCRIPCIÓN	1275	X'000004FB'
MQIACF_ID_DE_USUARIO_VARIABLE	1277	X'000004FD'
MQIACF_SOLO_SOLICITUD	1280	X'00000500'
MQIACF_PUB_PRIORIDAD	1283	X'00000503'
MQIACF_SUB_ATRIBUTOS	1287	X'00000507'
MQIACF_COMODÍN_ESQUEMA	1288	X'00000508'
MQIACF_SUBTIPO	1289	X'00000509'
MQIACF_NÚMERO_DE_MENSAJES	1290	X'0000050A'
MQIACF_Q_MGR_PUBSUB	1291	X'0000050B'
MQIACF_Q_MGR_VERSIÓN	1292	X'0000050C'
MQIACF_SUB_STATUS_ATTRS	1294	X'0000050E'
MQIACF_ESTADO_DEL_TEMA	1295	X'0000050F'
MQIACF_TEMA_SUB	1296	X'00000510'
MQIACF_TEMA_PUBLICACIÓN	1297	X'00000511'
MQIACF_RETAINED_PUBLICATION	1300	X'00000514'
MQIACF_ESTADO_ATRIBUTOS_TEMA	1301	X'00000515'
MQIACF_TEMA_ESTADO_TIPO	1302	X'00000516'
MQIACF_SUB_OPTIONS	1303	X'00000517'
MQIACF_PUBLISH_COUNT	1304	X'00000518'
MQIACF_CLEAR_TYPE	1305	X'00000519'
MQIACF_BORRAR_ÁMBITO	1306	X'0000051A'
MQIACF_SUBNIVEL	1307	X'0000051B'
MQIACF_ASYNC_STATE	1308	X'0000051C'
MQIACF_SUB_RESUMEN	1309	X'0000051D'
MQIACF_MENSAJES_OBSOLETOS	1310	X'0000051E'
MQIACF_PUBSUB_ESTADO	1311	X'0000051F'
MQIACF_PS_TIPO_ESTADO	1314	X'00000522'
MQIACF_PUBSUB_STATUS_ATTRS	1318	X'00000526'
MQIACF_TIPO_SELECTOR	1321	X'00000529'
MQIACF_MCAST_REL_INDICADOR	1351	X'00000547'
MQIACF_CHLAUTH_TIPO	1352	X'00000548'
MQXR_TIPO_DIAGNÓSTICO	1354	X'0000054A'
MQIACF_CHLAUTH_ATTRS	1355	X'0000054B'
MQIACF_ID_OPERACIÓN	1356	X'0000054C'

Tabla 180. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQIACF_API_CALLER_TYPE	1357	X'0000054D'
MQIACF_API_ENVIRONMENT	1358	X'0000054E'
MQIACF_TRACE_DETAIL	1359	X'0000054F'
MQIACF_HOBJ	1360	X'00000550'
MQIACF_CALL_TYPE	1361	X'00000551'
MQIACF_MQCB_OPERATION	1362	X'00000552'
MQIACF_MQCB_TYPE	1363	X'00000553'
MQIACF_MQCB_OPTIONS	1364	X'00000554'
MQIACF_CLOSE_OPTIONS	1365	X'00000555'
MQIACF_CTL_OPERATION	1366	X'00000556'
MQIACF_GET_OPTIONS	1367	X'00000557'
MQIACF_RECS_PRESENT	1368	X'00000558'
MQIACF_KNOWN_DEST_COUNT	1369	X'00000559'
MQIACF_UNKNOWN_DEST_COUNT	1370	X'0000055A'
MQIACF_INVALID_DEST_COUNT	1371	X'0000055B'
MQIACF_RESOLVED_TYPE	1372	X'0000055C'
MQIACF_PUT_OPTIONS	1373	X'0000055D'
MQIACF_BUFFER_LENGTH	1374	X'0000055E'
MQIACF_TRACE_DATA_LENGTH	1375	X'0000055F'
MQIACF_SMDS_EXPANDST	1376	X'00000560'
MQIACF_ESTRUCTURA_LONGITUD	1377	X'00000561'
MQIACF_NÚMERO_DE_ARTÍCULOS	1378	X'00000562'
MQIACF_FECHA_DE_CADUCIDAD	1379	X'00000563'
MQIACF_CONNECT_TIME	1380	X'00000564'
MQIACF_DISCONNECTAR_TIEMPO	1381	X'00000565'
MQIACF_HSUB	1382	X'00000566'
MQIACF_SUBRQ_OPTIONS	1383	X'00000567'
MQIACF_XA_RMID	1384	X'00000568'
MQIACF_XA_FLAGS	1385	X'00000569'
MQIACF_XA_RETCODE	1386	X'0000056A'
MQIACF_XA_HANDLE	1387	X'0000056B'
MQIACF_XA_RETVAL	1388	X'0000056C'
MQIACF_STATUS_TYPE	1389	X'0000056D'
MQIACF_XA_COUNT	1390	X'0000056E'
MQIACF_SELECTOR_COUNT	1391	X'0000056F'
MQIACF_SELECTORS	1392	X'00000570'
MQIACF_INTATTR_COUNT	1393	X'00000571'
MQIACF_INTATTRS	1394	X'00000572'
MQIACF_SUBRQ_ACTION	1395	X'00000573'

Tabla 180. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQIACF_NUM_PUBS	1396	X'00000574'
MQIACF_POINTER_SIZE	1397	X'00000575'
MQIACF_REMOVER_AUTOREC	1398	X'00000576'
MQIACF_XR_ATTRS	1399	X'00000577'
MQIACF_APPL_FUNCTION_TYPE	1400	X'00000578'
MQIACF_AMQP_ATTRS	1401	X'00000579'
MQIACF_TIPO_EXPORTACIÓN	1402	X'0000057A'
MQIACF_EXPORT_ATTRS	1403	X'0000057B'
MQIACF_SISTEMA_OBJETOS	1404	X'0000057C'
MQIACF_CONEXIÓN_INTERCAMBIO	1405	X'0000057D'
MQIACF_AMQP_DIAGNÓSTICO_TIPO	1406	X'0000057E'
Ubicación del búfer MQIACF	1408	X'00000580'
MQIACF_LDAP_CONEXIÓN_ESTADO	1409	X'00000581'
MQIACF_SYSP_MAX_ACE_POOL	1410	X'00000582'
MQIACF_PÁGINACLAS	1411	X'00000583'
MQIACF_AUTH_REC_TYPE	1412	X'00000584'
MQIACF_SYSP_MÁXIMO_NÚMERO_DE_DESCARGAS	1413	X'00000585'
MQIACF_SYSP_ZHYPERWRITE	1414	X'00000586'
MQIACF_Q_MGR_STATUS_LOG	1415	X'00000587'
Tamaño del registro de archivo MQIACF	1416	X'00000588'
MQIACF_MEDIA_LOG_SIZE	1417	X'00000589'
MQIACF_REINICIAR_TAMAÑO_REGISTRO	1418	X'0000058A'
MQIACF_TAMAÑO_DE_REGISTRO_REUTILIZABLE	1419	X'0000058B'
MQIACF_INICIAR_SESIÓN_USO	1420	X'0000058C'
MQIACF_LOG_UTILIZACIÓN	1421	X'0000058D'
MQIACF_IGNORAR_ESTADO	1423	X'0000058F'
MQIACF_NÚMERO_DE_APLICACIONES_MÓVILES	1424	X'00000590'
MQIACF_APPL_INFO_ATTRS	1425	X'00000591'
MQIACF_APPL_MOVIBLE	1426	X'00000592'
MQIACF_REMOTO_QMGR_ACTIVO	1427	X'00000593'
MQIACF_APPL_INFO_TYPE	1428	X'00000594'
MQIACF_APPL_INFO_APPL	1429	X'00000595'
MQIACF_APPL_INFO_QMGR	1430	X'00000596'
MQIACF_APPL_INFO_LOCAL	1431	X'00000597'
MQIACF_APPL_CONTABILIZACIÓN_INMUEBLE	1432	X'00000598'
MQIACF_EQUILIBRADO	1433	X'00000599'
MQIACF_BALSTATE	1434	X'0000059A'
MQIACF_APPL_MOTIVO_INMOVILIZABLE	1435	X'0000059B'
MQIACF_DS_ENCRIPTADO	1436	X'0000059C'

Tabla 180. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQIACF_CUR_Q_TAMAÑO_ARCHIVO	1437	X'0000059D'
MQIACF_CUR_MAX_TAMAÑO_ARCHIVO	1438	X'0000059E'
MQIACF_TIPO_DE_EQUILIBRIO	1439	X'0000059F'
MQIACF_OPCIONES_DE_EQUILIBRIO	1440	X'000005A0'
MQIACF_TIEMPO_DE_EQUILIBRIO_EXCESS	1441	X'000005A1'
MQIACF_SYSP_SMF_STAT_TIME_SECS	1442	X'000005A2'
MQIACF_SYSP_SMF_ACCT_TIEMPO_MINUTOS	1443	X'000005A3'
MQIACF_SYSP_SMF_ACCT_TIEMPO_SEGUNDOS	1444	X'000005A4'
MQIACF_EVENTO_CONTE_DUPLICADO	1478	X'000005C6'
MQIACF_ÚLTIMO_USADO	1478	X'000005C6'

MQIACH_* (Formato de comando Tipos de canales enteros)

Tabla 181. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQIACH_FIRST	1501	X'000005DD'
MQIACH_XMIT_PROTOCOL_TYPE	1501	X'000005DD'
MQIACH_BATCH_SIZE	1502	X'000005DE'
MQIACH_DISC_INTERVAL	1503	X'000005DF'
MQIACH_SHORT_TIMER	1504	X'000005E0'
MQIACH_SHORT_RETRY	1505	X'000005E1'
MQIACH_LONG_TIMER	1506	X'000005E2'
MQIACH_LONG_RETRY	1507	X'000005E3'
MQIACH_PUT_AUTHORITY	1508	X'000005E4'
MQIACH_SEQUENCE_NUMBER_WRAP	1509	X'000005E5'
MQIACH_MAX_MSG_LENGTH	1510	X'000005E6'
MQIACH_CHANNEL_TYPE	1511	X'000005E7'
MQIACH_DATA_COUNT	1512	X'000005E8'
MQIACH_NAME_COUNT	1513	X'000005E9'
MQIACH_MSG_SEQUENCE_NUMBER	1514	X'000005EA'
MQIACH_DATA_CONVERSION	1515	X'000005EB'
MQIACH_IN_DOUBT	1516	X'000005EC'
MQIACH_MCA_TYPE	1517	X'000005ED'
MQIACH_SESSION_COUNT	1518	X'000005EE'
MQIACH_ADAPTER	1519	X'000005EF'
MQIACH_COMMAND_COUNT	1520	X'000005F0'
MQIACH_SOCKET	1521	X'000005F1'
MQIACH_PORT	1522	X'000005F2'
MQIACH_CHANNEL_INSTANCE_TYPE	1523	X'000005F3'
MQIACH_CHANNEL_INSTANCE_ATTRS	1524	X'000005F4'

Tabla 181. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQIACH_CHANNEL_ERROR_DATA	1525	X'000005F5'
MQIACH_CHANNEL_TABLE	1526	X'000005F6'
MQIACH_CHANNEL_STATUS	1527	X'000005F7'
MQIACH_INDOUBT_STATUS	1528	X'000005F8'
MQIACH_LAST_SEQ_NUMBER	1529	X'000005F9'
MQIACH_LAST_SEQUENCE_NUMBER	1529	X'000005F9'
MQIACH_CURRENT_MSGS	1531	X'000005FB'
MQIACH_CURRENT_SEQ_NUMBER	1532	X'000005FC'
MQIACH_CURRENT_SEQUENCE_NUMBER	1532	X'000005FC'
MQIACH_SSL_RETURN_CODE	1533	X'000005FD'
MQIACH_MSGS	1534	X'000005FE'
MQIACH_BYTES_SENT	1535	X'000005FF'
MQIACH_BYTES_RCVD	1536	X'00000600'
MQIACH_BYTES_RECEIVED	1536	X'00000600'
MQIACH_BATCHES	1537	X'00000601'
MQIACH_BUFFERS_SENT	1538	X'00000602'
MQIACH_BUFFERS_RCVD	1539	X'00000603'
MQIACH_BUFFERS_RECEIVED	1539	X'00000603'
MQIACH_LONG_RETRIES_LEFT	1540	X'00000604'
MQIACH_SHORT_RETRIES_LEFT	1541	X'00000605'
MQIACH_MCA_STATUS	1542	X'00000606'
MQIACH_STOP_REQUESTED	1543	X'00000607'
MQIACH_MR_COUNT	1544	X'00000608'
MQIACH_MR_INTERVAL	1545	X'00000609'
MQIACH_NPM_SPEED	1562	X'0000061A'
MQIACH_HB_INTERVAL	1563	X'0000061B'
MQIACH_BATCH_INTERVAL	1564	X'0000061C'
MQIACH_NETWORK_PRIORITY	1565	X'0000061D'
MQIACH_KEEP_ALIVE_INTERVAL	1566	X'0000061E'
MQIACH_BATCH_HB	1567	X'0000061F'
MQIACH_SSL_CLIENT_AUTH	1568	X'00000620'
MQIACH_ALLOC_RETRY	1570	X'00000622'
MQIACH_ALLOC_FAST_TIMER	1571	X'00000623'
MQIACH_ALLOC_SLOW_TIMER	1572	X'00000624'
MQIACH_DISC_RETRY	1573	X'00000625'
MQIACH_PORT_NUMBER	1574	X'00000626'
MQIACH_HDR_COMPRESSION	1575	X'00000627'
MQIACH_MSG_COMPRESSION	1576	X'00000628'
MQIACH_CLWL_CHANNEL_RANK	1577	X'00000629'

Tabla 181. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQIACH_CLWL_CHANNEL_PRIORITY	1578	X'0000062A'
MQIACH_CLWL_CHANNEL_WEIGHT	1579	X'0000062B'
MQIACH_CHANNEL_DISP	1580	X'0000062C'
MQIACH_INBOUND_DISP	1581	X'0000062D'
MQIACH_CHANNEL_TYPES	1582	X'0000062E'
MQIACH_ADAPS_STARTED	1583	X'0000062F'
MQIACH_ADAPS_MAX	1584	X'00000630'
MQIACH_DISPS_STARTED	1585	X'00000631'
MQIACH_DISPS_MAX	1586	X'00000632'
MQIACH_SSLTASKS_STARTED	1587	X'00000633'
MQIACH_SSLTASKS_MAX	1588	X'00000634'
MQIACH_CURRENT_CHL	1589	X'00000635'
MQIACH_CURRENT_CHL_MAX	1590	X'00000636'
MQIACH_CURRENT_CHL_TCP	1591	X'00000637'
MQIACH_CURRENT_CHL_LU62	1592	X'00000638'
MQIACH_ACTIVE_CHL	1593	X'00000639'
MQIACH_ACTIVE_CHL_MAX	1594	X'0000063A'
MQIACH_ACTIVE_CHL_PAUSED	1595	X'0000063B'
MQIACH_ACTIVE_CHL_STARTED	1596	X'0000063C'
MQIACH_ACTIVE_CHL_STOPPED	1597	X'0000063D'
MQIACH_ACTIVE_CHL_RETRY	1598	X'0000063E'
MQIACH_LISTENER_STATUS	1599	X'0000063F'
MQIACH_SHARED_CHL_RESTART	1600	X'00000640'
MQIACH_LISTENER_CONTROL	1601	X'00000641'
MQIACH_BACKLOG	1602	X'00000642'
MQIACH_XMITQ_TIME_INDICATOR	1604	X'00000644'
MQIACH_NETWORK_TIME_INDICATOR	1605	X'00000645'
MQIACH_EXIT_TIME_INDICATOR	1606	X'00000646'
MQIACH_BATCH_SIZE_INDICATOR	1607	X'00000647'
MQIACH_XMITQ_MSGS_AVAILABLE	1608	X'00000648'
MQIACH_CHANNEL_SUBSTATE	1609	X'00000649'
MQIACH_SSL_KEY_RESETS	1610	X'0000064A'
MQIACH_COMPRESSION_RATE	1611	X'0000064B'
MQIACH_COMPRESSION_TIME	1612	X'0000064C'
MQIACH_MAX_XMIT_SIZE	1613	X'0000064D'
MQIACH_DEF_CHANNEL_DISP	1614	X'0000064E'
MQIACH_SHARING_CONVERSATIONS	1615	X'0000064F'
MQIACH_MAX_SHARING_CONVS	1616	X'00000650'
MQIACH_CURRENT_SHARING_CONVS	1617	X'00000651'

Tabla 181. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQIACH_MAX_INSTANCES	1618	X'00000652'
MQIACH_MAX_INSTS_PER_CLIENT	1619	X'00000653'
MQIACH_CLIENT_CHANNEL_WEIGHT	1620	X'00000654'
MQIACH_CONNECTION_AFFINITY	1621	X'00000655'
MQIACH_AUTH_INFO_TYPES	1622	X'00000656'
MQIACH_RESET_REQUESTED	1623	X'00000657'
MQIACH_BATCH_DATA_LIMIT	1624	X'00000658'
MQIACH_MSG_HISTORY	1625	X'00000659'
MQIACH_MULTICAST_PROPERTIES	1626	X'0000065A'
MQIACH_NEW_SUBSCRIBER_HISTORY	1627	X'0000065B'
MQIACH_MC_HB_INTERVAL	1628	X'0000065C'
MQIACH_USE_CLIENT_ID	1629	X'0000065D'
MQIACH_MQTT_KEEP_ALIVE	1630	X'0000065E'
MQIACH_IN_DOUBT_IN	1631	X'0000065F'
MQIACH_IN_DOUBT_OUT	1632	X'00000660'
MQIACH_MSGS_SENT<	1633	X'00000661'
MQIACH_MSGS_RECEIVED	1634	X'00000662'
MQIACH_MSGS_RCVD	1634	X'00000662'
MQIACH_PENDING_OUT	1635	X'00000663'
MQIACH_AVAILABLE_CIPHERSPECS	1636	X'00000664'
MQIACH_MATCH	1637	X'00000665'
MQIACH_USER_SOURCE	1638	X'00000666'
MQIACH_WARNING	1639	X'00000667'
MQIACH_DEF_RECONNECT	1640	X'00000668'
MQIACH_CHANNEL_SUMMARY_ATTRS	1642	X'0000066A'
MQIACH_PROTOCOL	1643	X'0000066B'
MQIACH_AMQPKEEPALIVE	1644	X'0000066C'
MQIACH_SECURITY_PROTOCOL	1645	X'0000066D'
 MQIACH_SPL_PROTECTION	1646	X'0000066E'
MQIACH_LAST_USED	1646	X'0000066E'

MQIAMO_* (Formato de comando Tipos de parámetros de monitorización de enteros)

Tabla 182. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQIAMO_PRIMERO	701	X'000002BD'
MQIAMO_AVG_TAMAÑO_LOTE	702	X'000002BE'
MQIAMO_AVG_Q_TIEMPO	703	X'000002BF'
MQIAMO_BACKOUTS	704	X'000002C0'

Tabla 182. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQIAMO_BROWSES	705	X'000002C1'
MQIAMO_BROWSE_MAX_BYTES	706	X'000002C2'
MQIAMO_BROWSE_MIN_BYTES	707	X'000002C3'
MQIAMO_BROWSES_FAILED	708	X'000002C4'
MQIAMO_CLOSES	709	X'000002C5'
MQIAMO_COMMITS	710	X'000002C6'
MQIAMO_COMMITS_FAILED	711	X'000002C7'
MQIAMO_CONEXIONES	712	X'000002C8'
MQIAMO_CONNS_MAX	713	X'000002C9'
MQIAMO_DISCOS	714	X'000002CA'
MQIAMO_DISCS_IMPLÍCITO	715	X'000002CB'
MQIAMO_DISC_TYPE	716	X'000002CC'
MQIAMO_TIEMPO_DE_SALIDA_MEDIO	717	X'000002CD'
MQIAMO_EXIT_TIME_MAX	718	X'000002CE'
MQIAMO_EXIT_TIME_MIN	719	X'000002CF'
MQIAMO_LOTES_COMPLETOS	720	X'000002D0'
MQIAMO_GENERATED_MSGS	721	X'000002D1'
MQIAMO_GETS	722	X'000002D2'
MQIAMO_GET_MAX_BYTES	723	X'000002D3'
MQIAMO_GET_MIN_BYTES	724	X'000002D4'
MQIAMO_GETS_FAILED	725	X'000002D5'
MQIAMO_LOTES_INCOMPLETOS	726	X'000002D6'
MQIAMO_INQS	727	X'000002D7'
MQIAMO_MSGS	728	X'000002D8'
MQIAMO_NET_TIME_AVG	729	X'000002D9'
MQIAMO_NET_TIME_MAX	730	X'000002DA'
MQIAMO_NET_TIME_MIN	731	X'000002DB'
MQIAMO_OBJECT_COUNT	732	X'000002DC'
MQIAMO_OPENS	733	X'000002DD'
MQIAMO_PUT1S	734	X'000002DE'
MQIAMO_PUTS	735	X'000002DF'
MQIAMO_PUT_MAX_BYTES	736	X'000002E0'
MQIAMO_PUT_MIN_BYTES	737	X'000002E1'
MQIAMO_PUT_RETRIES	738	X'000002E2'
MQIAMO_Q_MAX_DEPTH	739	X'000002E3'
MQIAMO_Q_MIN_DEPTH	740	X'000002E4'
MQIAMO_Q_TIEMPO_MEDIO	741	X'000002E5'
MQIAMO_Q_TIEMPO_MÁXIMO	742	X'000002E6'
MQIAMO_Q_TIEMPO_MÍNIMO	743	X'000002E7'

Tabla 182. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQIAMO_SETS	744	X'000002E8'
MQIAMO_CONEXIONES_FALLIDAS	749	X'000002ED'
MQIAMO_OPENS_FAILED	751	X'000002EF'
MQIAMO_INQS_FAILED	752	X'000002F0'
MQIAMO_SETS_FAILED	753	X'000002F1'
MQIAMO_PUTS_FAILED	754	X'000002F2'
MQIAMO_PUT1S_FAILED	755	X'000002F3'
MQIAMO_CLOSES_FAILED	757	X'000002F5'
MQIAMO_MSGS_EXPIRED	758	X'000002F6'
MQIAMO_MSGS_NOT_QUEUED	759	X'000002F7'
MQIAMO_MSGS_PURGED	760	X'000002F8'
MQIAMO_SUBS_DUR	764	X'000002FC'
MQIAMO_SUBS_NDUR	765	X'000002FD'
MQIAMO_SUBS_FAILED	766	X'000002FE'
MQIAMO_SUBRQS	767	X'000002FF'
MQIAMO_SUBRQS_FAILED	768	X'00000300'
MQIAMO_CBS	769	X'00000301'
MQIAMO_CBS_FAILED	770	X'00000302'
MQIAMO_CTL5	771	X'00000303'
MQIAMO_CTL5_FAILED	772	X'00000304'
MQIAMO_ESTADÍSTICAS	773	X'00000305'
MQIAMO_STATS_FAILED	774	X'00000306'
MQIAMO_SUB_DUR_HIGHWATER	775	X'00000307'
MQIAMO_SUB_DUR_LOWWATER	776	X'00000308'
MQIAMO_SUB_NDUR_HIGHWATER	777	X'00000309'
MQIAMO_SUB_NDUR_LOWWATER	778	X'0000030A'
MQIAMO_TOPIC_PUTS	779	X'0000030B'
MQIAMO_TOPIC_PUTS_FAILED	780	X'0000030C'
MQIAMO_TOPIC_PUT1S	781	X'0000030D'
MQIAMO_TOPIC_PUT1S_FAILED	782	X'0000030E'
MQIAMO_PUBLISH_MSG_COUNT	784	X'00000310'
MQIAMO_UNSUBS_DUR	786	X'00000312'
MQIAMO_UNSUBS_NDUR	787	X'00000313'
MQIAMO_UNSUBS_FAILED	788	X'00000314'
MQIAMO_INTERVAL	789	X'00000315'
MQIAMO_MENSAJES_ENVIADOS	790	X'00000316'
MQIAMO_BYTES_ENVIADOS	791	X'00000317'
MQIAMO_REPARACIÓN_BOCETOS	792	X'00000318'
MODO_DE_RESPUESTA_MQIAMO	793	X'00000319'

Tabla 182. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQIAMO_TIPO_FIABILIDAD	794	X'0000031A'
MQIAMO_ÚLTIMA_INCORPORACIÓN_MARCA	795	X'0000031B'
MQIAMO_NACKS_RCVD	796	X'0000031C'
MQIAMO_REPARACIÓN_PAQUETES	797	X'0000031D'
MQIAMO_HISTORIA_PKTS	798	X'0000031E'
MQIAMO_PAQUETES_PENDENTES	799	X'0000031F'
MQIAMO_PKT_RATE	800	X'00000320'
MQIAMO_MCAST_XMIT_RATE	801	X'00000321'
MQIAMO_MCAST_TIEMPO_LOTE	802	X'00000322'
MQIAMO_MCAST_LATIDO	803	X'00000323'
MQIAMO_DEST_DATA_PORT	804	X'00000324'
MQIAMO_DEST_REPARACIÓN_PUERTO	805	X'00000325'
MQIAMO_ACKS_RCVD	806	X'00000326'
MQIAMO_ACTIVO_ACKERS	807	X'00000327'
MQIAMO_PKTS_ENVIADOS	808	X'00000328'
MQIAMO_TOTAL_REPARACIÓN_PKTS	809	X'00000329'
MQIAMO_TOTAL_PUNTOS_ENVIADOS	810	X'0000032A'
MQIAMO_TOTAL_MENSAJES_ENVIADOS	811	X'0000032B'
MQIAMO_TOTAL_BYTES_ENVIADOS	812	X'0000032C'
MQIAMO_NÚMERO_DE_FLUJOS	813	X'0000032D'
MQIAMO_ACK_COMENTARIOS	814	X'0000032E'
MQIAMO_NACK_COMENTARIO	815	X'0000032F'
MQIAMO_PKTS_LOST	816	X'00000330'
MQIAMO_MSGS_RCVD	817	X'00000331'
MQIAMO_MSG_BYTES_RCVD	818	X'00000332'
MQIAMO_MENSAJES_ENTREGADOS	819	X'00000333'
MQIAMO_PKTS_PROCESADOS	820	X'00000334'
MQIAMO_PKTS_DLVD	821	X'00000335'
MQIAMO_PKTS_DROPPED	822	X'00000336'
MQIAMO_PKTS_DUPLICADOS	823	X'00000337'
MQIAMO_NACKS_CREADO	824	X'00000338'
MQIAMO_NACK_PKTS_SEND	825	X'00000339'
MQIAMO_REPARACIÓN_PAQUETES_SOLICITUD	826	X'0000033A'
MQIAMO_REPARACIÓN_PAQUETES_RECIBIDOS	827	X'0000033B'
MQIAMO_PKTS_REPARADOS	828	X'0000033C'
MQIAMO_TOTAL_MENSAJES_RECIBIDOS	829	X'0000033D'
MQIAMO_TOTAL_BYTES_DE_MENSAJES_RECEPCIONADOS	830	X'0000033E'
MQIAMO_TOTAL_REPARACIÓN_PKTS_RCVD	831	X'0000033F'
MQIAMO_TOTAL_REPARACIÓN_PKTS_RQSTD	832	X'00000340'

Tabla 182. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQIAMO_TOTAL_MENSAJES_PROCESADOS	833	X'00000341'
MQIAMO_TOTAL_MENSAJES_SELECCIONADOS	834	X'00000342'
MQIAMO_TOTAL_MENSAJES_CADUCADOS	835	X'00000343'
MQIAMO_TOTAL_MENSAJES_ENTREGADOS	836	X'00000344'
MQIAMO_TOTAL_MENSAJES_DEVUELTOS	837	X'00000345'
MQIAMO_ÚLTIMO_USADO	837	X'00000345'

MQIAMO64_* (Formato de comando: tipos de parámetros de supervisión de enteros de 64 bits)

Tabla 183. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQIAMO64_AVG_Q_TIME	703	X'000002BF'
MQIAMO64_Q_TIME_AVG	741	X'000002E5'
MQIAMO64_Q_TIME_MAX	742	X'000002E6'
MQIAMO64_Q_TIME_MIN	743	X'000002E7'
MQIAMO64_BROWSE_BYTES	745	X'000002E9'
MQIAMO64_BYTES	746	X'000002EA'
MQIAMO64_GET_BYTES	747	X'000002EB'
MQIAMO64_PUT_BYTES	748	X'000002EC'
MQIAMO64_TOPIC_PUT_BYTES	783	X'0000030F'
MQIAMO64_PUBLISH_MSG_BYTES	785	X'00000311'

MQIASY_* (Selektors de sistema enteros)

Tabla 184. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQIASY_PRIMERO	-1	X'FFFFFFFF'
MQIASY_CÓDIGO_CONJUNTO_CARACTERES_ID	-1	X'FFFFFFFF'
MQIASY_TIPO	-2	X'FFFFFFFE'
MQIASY_COMANDO	-3	X'FFFFFFFD'
NÚMERO DE SECUENCIA DE MENSAJE MQIASY	-4	X'FFFFFFFC'
MQIASY_CONTROL	-5	X'FFFFFFFB'
MQIASY_CÓDIGO_EMPRESA	-6	X'FFFFFFFA'
MQIASY_MOTIVO	-7	X'FFFFFFF9'
MQIASY_BAG_OPTIONS	-8	X'FFFFFFF8'
MQIASY_VERSIÓN	-9	X'FFFFFFF7'
MQIASY_ÚLTIMO_USADO	-9	X'FFFFFFF7'
MQIASY_ÚLTIMO	-2000	X'FFFFF830'

MQIAUT_* (autenticador de IMS encabezado de información)

Tabla 185. Nombres y valores constantes	
Nombre	Valor
MQIAUT_NINGUNO	"��������"
MQIAUT_NINGUNA_MATRIZ	'�', '�', '�', '�', '�', '�', '�', '�', '�', '�'

Nota: El s mbolo   representa un  nico car cter en blanco.

MQIAV_* (Valores de atributos enteros)

Tabla 186. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQIAV_NO_APLICABLE	-1	X'FFFFFFFF'
MQIAV_INDETERMINADO	-2	X'FFFFFFFE'

MQICM_* (modos de IMS confirmaci n de encabezado de informaci n)

Tabla 187. Nombres y valores constantes	
Nombre	Valor
MQICM_COMPROMETER_Y_ENVIAR	'0'
MQICM_ENVIAR_Y_CONFIRMAR	'1'

MQIDO_* (Opciones de formato de comando Indoubt)

Tabla 188. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQIDO_COMMIT	1	X'00000001'
MQIDO_RETIRADA	2	X'00000002'

MQIEP_* (Puntos de entrada de interfaz)

Estructura de los par metros de seguridad de la conexi n

Tabla 189. Estructuras de constantes	
Nombre	Estructura
MQIEP_STRUC_ID	"IEP�"
MQIEP_STRUC_ID_ARRAY	'I', 'E', 'P', '�'

Nota: El s mbolo   representa un  nico car cter en blanco.

Tabla 190. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQIEP_VERSION_1	1	X'00000001'
MQDXPversi�n_ACTUAL	1	X'00000001'

MQIGQ_* (Cola intragrupo)

Tabla 191. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQIGQ_DESACTIVADO	0	X'00000000'
MQIGQ_HABILITADO	1	X'00000001'

MQIGQPA_* (Autoridad de puesta en cola intragrupo)

Tabla 192. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQIGQPA_PREDEFINIDO	1	X'00000001'
MQIGQPA_CONTEXTO	2	X'00000002'
MQIGQPA_SOLO_IGQ	3	X'00000003'
MQIGQPA_ALTERNATIVO_O_IGQ	4	X'00000004'

MQIIH_* (estructura de IMS encabezado de información y indicadores)

IMS Estructura del encabezado de información

Tabla 193. Estructuras de constantes	
Nombre	Estructura
MQIIH_STRUC_ID	"IIH¬"
MQIIH_STRUC_ID_ARRAY	'I','I','H','¬'

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

Tabla 194. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQIIH_VERSION_1	1	X'00000001'
MQIIH_VERSIÓN_ACTUAL	1	X'00000001'
MQIIH_LENGTH_1	84	X'00000054'

IMS cabecera de información Indicadores

Tabla 195. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQIIH_NINGUNO	0	X'00000000'
MQIIH_PLAZO_DE_VIGENCIA	1	X'00000001'
MQIIH_VENCIMIENTO_ILIMITADO	0	X'00000000'
MQIIH_FORMATO_DE_RESPUESTA_NINGUNO	8	X'00000008'
MQIIH_IGNORAR_PURGA	16	X'00000010'
MQIIH_CM0_REQUEST_RESPONSE	32	X'00000020'

MQIMPO_* (Consultar opciones y estructura de las propiedades del mensaje)

Consultar la estructura de opciones de propiedad del mensaje

Tabla 196. Estructuras de constantes	
Nombre	Estructura
MQIMPO_STRUC_ID	"IMPO"
MQIMPO_STRUC_ID_ARRAY	'I','M','P','O'

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

Tabla 197. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQIMPO_VERSION_1	1	X'00000001'
MQIMPO_VERSION_ACTUAL	1	X'00000001'

Opciones de propiedad del mensaje de consulta

Tabla 198. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQIMPO_CONVERTIR_TIPO	2	X'00000002'
MQIMPO_LONGITUD_CONSULTA	4	X'00000004'
MQIMPO_INQ_PRIMERO	0	X'00000000'
MQIMPO_INQ_SIGUIENTE	8	X'00000008'
MQIMPO_INQ_PROP_BAJO_EL_CURSOR	16	X'00000010'
MQIMPO_CONVERTIR_VALOR	32	X'00000020'
MQIMPO_NINGUNO	0	X'00000000'

MQINBD_* (Formato de comando Disposiciones entrantes)

Tabla 199. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQINBD_Q_MGR	0	X'00000000'
MQINBD_GRUPO	3	X'00000003'

MQIND_* (Valores de índice especiales)

Tabla 200. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQIND_NONE	-1	X'FFFFFFFF'
MQIND_TODOS	-2	X'FFFFFFFE'

MQIPADDR_* (Versiones de direcciones IP)

Tabla 201. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQIPADDR_IPV4	0	X'00000000'
MQIPADDR_IPV6	1	X'00000001'

MQISS_* (IMS encabezado de información Ámbitos de seguridad)

Tabla 202. Nombres y valores constantes	
Nombre	Valor
MQISS_COMPROBAR	'C'
MQISS_COMPLETO	'F'

MQIT_* (Tipos de índice)

Tabla 203. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQIT_NINGUNO	0	X'00000000'
MQIT_MSG_ID	1	X'00000001'
MQIT_CORREL_ID	2	X'00000002'
MQIT_MSG_TOKEN	4	X'00000004'
MQIT_GROUP_ID	5	X'00000005'

MQITEM_* (Tipo de elemento para mqInquireItemInfo)

Tabla 204. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQITEM_ENTERO	1	X'00000001'
MQITEM_STRING	2	X'00000002'
MQITEM_BOLSA	3	X'00000003'
MQITEM_BYTE_STRING	4	X'00000004'
MQITEM_FILTRO_ENTEROS	5	X'00000005'
MQITEM_STRING_FILTER	6	X'00000006'
MQITEM_INTEGER64	7	X'00000007'
MQITEM_BYTE_STRING_FILTRO	8	X'00000008'

MQITII_* (identificador de instancia de IMS transacción del encabezado de información)

Tabla 205. Nombres y valores constantes	
Nombre	Valor
MQITII_NINGUNO	X'00...00' (16 nulos)
MQITII_NINGUNA_MATRIZ	'\0','\0',... (16 nulos)

MQITS_* (cabecera IMS de información Estados de transacción)

Tabla 206. Nombres y valores constantes	
Nombre	Valor
MQITS_EN_CONVERSACIÓN	'C'
MQITS_NO_EN_CONVERSACIÓN	'¬'
MQITS_ARCHITECTED	'A'

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

MQKAI_* (Intervalo KeepAlive)

Tabla 207. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQKAI_AUTO	-1	X'FFFFFFFF'

MQMASTER_* (Administración maestra)

Tabla 208. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQMASTER_NO	0	X'00000000'
MQMASTER_SÍ	1	X'00000001'

MQMCAS_* (Formato de comando: estado del agente del canal de mensajes)

Tabla 209. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQMCAS_DETENIDO	0	X'00000000'
MQMCAS_EN_MARCHA	3	X'00000003'

MQMCAT_* (Tipos MCA)

Tabla 210. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQMCAT_PROCESO	1	X'00000001'
MQMCAT_THREAD	2	X'00000002'

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

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

Tabla 211. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQMCD_CARPETA_VERSION	1	X'00000001'

Opciones de publicación/suscripción Etiqueta Nombres de etiquetas

Tabla 212. Nombres y valores constantes	
Nombre	Valor
MQMCD_MSG_DOMINIO	"Msd"
MQMCD_MSG_SET	"Set"
MQMCD_TIPO_MENSAJE	"Type"
MQMCD_FORMATO_DE_MENSAJE	"Fmt"

Opciones de publicación/suscripción Etiqueta Nombres de etiquetas XML

Tabla 213. Nombres y valores constantes	
Nombre	Valor
MQMCD_MSG_DOMINIO_B	"<Msd>"
MQMCD_MSG_DOMINIO_E	"</Msd>"
MQMCD_MSG_SET_B	"<Set>"
MQMCD_MSG_SET_E	"</Set>"
MQMCD_TIPO_MENSAJE_B	"<Type>"
MQMCD_TIPO_MENSAJE_E	"</Type>"
MQMCD_MSG_FORMATO_B	"<Fmt>"
MQMCD_MSG_FORMATO_E	"</Fmt>"

Opciones de publicación/suscripción Etiqueta Valores de etiqueta

Tabla 214. Nombres y valores constantes	
Nombre	Valor
MQMCD_DOMINIO_NINGUNO	"none"
MQMCD_DOMINIO_NEÓN	"neon"
MQMCD_DOMINIO_MRM	"mrm"
MQMCD_DOMINIO_JMS_NINGUNO	"jms_none"
MQMCD_DOMINIO_JMS_TEXTO	"jms_text"
MQMCD_DOMINIO_OBJETO_JMS	"jms_object"
MQMCD_DOMINIO_JMS_MAPA	"jms_map"
MQMCD_DOMINIO_JMS_FLUJO	"jms_stream"
MQMCD_DOMINIO_JMS_BYTES	"jms_bytes"

MQMD_* (Estructura descriptora de mensajes)

Tabla 215. Estructuras de constantes	
Nombre	Estructura
MQMD_STRUC_ID	"MD↵"
MQMD_STRUC_ID_ARRAY	'M', 'D', '↵', '↵'

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

Tabla 216. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQMD_VERSION_1	1	X'00000001'
MQMD_VERSION_2	2	X'00000002'
MQMD_VERSION_ACTUAL	2	X'00000002'

MQMDE_* (Estructura de extensión del descriptor de mensajes)

Tabla 217. Estructuras de constantes	
Nombre	Estructura
MQMDE_STRUC_ID	"MDE¬"
MQMDE_STRUC_ID_ARRAY	'M', 'D', 'E', '¬'

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

Tabla 218. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQMDE_VERSION_2	2	X'00000002'
MQMDE_VERSION_ACTUAL	2	X'00000002'
MQMDE_LENGTH_2	72	X'00000048'

MQMDEF_* (Indicadores de extensión del descriptor de mensajes)

Tabla 219. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQMDEF_NINGUNO	0	X'00000000'

MQMDS_* (Secuencia de entrega de mensajes)

Tabla 220. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQMDS_PRIORITY	0	X'00000000'
MQMDS_FIFO	1	X'00000001'

MQMF_* (Indicadores de mensajes)

Tabla 221. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQMF_SEGMENTACIÓN_INHIBIDA	0	X'00000000'
MQMF_SEGMENTATION_ALLOWED	1	X'00000001'
MQMF_MSG_EN_GRUPO	8	X'00000008'
Último mensaje en grupo	16	X'00000010'
MQMF_SEGMENTO	2	X'00000002'
MQMF_ÚLTIMO_SEGMENTO	4	X'00000004'
MQMF_NINGUNO	0	X'00000000'

MQMHBO_* (Manejador de mensajes para opciones y estructura del búfer)

Manejador de mensajes a estructura de opciones de búfer

Tabla 222. Estructuras de constantes	
Nombre	Estructura
MQMHBO_STRUC_ID	"MHBO"
MQMHBO_STRUC_ID_ARRAY	'M', 'H', 'B', 'O'

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

Tabla 223. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQMHBO_VERSION_1	1	X'00000001'
MQMHBO_VERSION_ACTUAL	1	X'00000001'

Opciones de manejo de mensajes al búfer

Tabla 224. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQMHBO_PROPERTIES_IN_MQRFH2	1	X'00000001'
MQMHBO_ELIMINAR_PROPIEDADES	2	X'00000002'
MQMHBO_NINGUNO	0	X'00000000'

MQMI_* (Identificador de mensaje)

Tabla 225. Nombres y valores constantes	
Nombre	Valor
MQMI_NONE	X'00...00' (24 nulos)
MQMI_NINGUNA_MATRIZ	'\0','\0',... (24 nulos)

MQMMBI_* (Intervalo de marcado y exploración de mensajes)

Tabla 226. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQMMBI_ILIMITADO	-1	X'FFFFFFFF'

MQMO_* (Opciones de coincidencia)

Tabla 227. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQMO_MATCH_MSG_ID	1	X'00000001'
MQMO_MATCH_CORREL_ID	2	X'00000002'
MQMO_MATCH_GROUP_ID	4	X'00000004'
MQMO_MATCH_MSG_SEQ_NUMBER	8	X'00000008'
MQMO_MATCH_OFFSET	16	X'00000010'
MQMO_MATCH_MSG_TOKEN	32	X'00000020'
MQMO_NONE	0	X'00000000'

MQMODE_* (Opciones de modo de formato de comando)

Tabla 228. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQMODE_FORCE	0	X'00000000'
MQMODE QUIESCE	1	X'00000001'
MQMODE_TERMINATE	2	X'00000002'

MQMON_* (Valores de supervisión)

Tabla 229. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQMON_NO_DISPONIBLE	-1	X'FFFFFFFF'
MQMON_NONE	-1	X'FFFFFFFF'
MQMON_Q_MGR	-3	X'FFFFFFFD'
MQMON_OFF	0	X'00000000'
MQMON_ON	1	X'00000001'
MQMON_DESACTIVADO	0	X'00000000'
MQMON_HABILITADO	1	X'00000001'
MQMON_BAJO	17	X'00000011'
MQMON_MEDIO	33	X'00000021'
MQMON_ALTO	65	X'00000041'

MQMT_* (Tipos de mensajes)

Tabla 230. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQMT_SISTEMA_PRIMERO	1	X'00000001'
MQMT_REQUEST	1	X'00000001'
MQMT_REPLY	2	X'00000002'
MQMT_DATAGRAM	8	X'00000008'
MQMT_REPORT	4	X'00000004'
MQMT_MQE_CAMPOS_DESDE_MQE	112	X'00000070'
MQMT_MQE_CAMPOS	113	X'00000071'
MQMT_SISTEMA_ÚLTIMO	65535	X'0000FFFF'
MQMT_APPL_FIRST	65536	X'00010000'
MQMT_APPL_ÚLTIMO	99999999	X'3B9AC9FF'

MQMTOK_* (Token de mensaje)

Tabla 231. Nombres y valores constantes	
Nombre	Valor
MQMTOK_NONE	X'00...00' (16 nulos)
MQMTOK_NONE_ARRAY	'\0','\0',... (16 nulos)

MQNC_* (Recuento de nombres)

Tabla 232. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQNC_MAX_NAMELIST_NAME_COUNT	256	X'00000100'

MQNPM_* (Clase de mensaje no persistente)

Tabla 233. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQNPM_CLASE_NORMAL	0	X'00000000'
MQNPM_CLASE_ALTA	10	X'0000000A'

MQNPMS_* (velocidades de NonPersistent-Message)

Tabla 234. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQNPMS_NORMAL	1	X'00000001'
MQNPMS_RÁPIDO	2	X'00000002'

MQNT_* (Tipos de listas de nombres)

Tabla 235. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQNT_NINGUNO	0	X'00000000'
MQNT_Q	1	X'00000001'
MQNT_CLUSTER	2	X'00000002'
MQNT_AUTH_INFO	4	X'00000004'
MQNT_TODOS	1001	X'000003E9'

MQNVS_* (Nombres para cadenas de nombre/valor)

Tabla 236. Nombres y valores constantes	
Nombre	Valor
MQNVS_TIPO_APLICACIÓN	"OPT_APP_GRP¬"
MQNVS_TIPO_MENSAJE	"OPT_MSG_TYPE¬"

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

MQOA_* (Límites para selectores de atributos de objetos)

Tabla 237. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQOA_PRIMERO	1	X'00000001'
MQOA_ÚLTIMO	9000	X'00002328'

MQOD_* (Estructura descriptora de objetos)

Tabla 238. Estructuras de constantes	
Nombre	Estructura
MQOD_STRUC_ID	"0D¬¬"
MQOD_STRUC_ID_ARRAY	'0','D','¬','¬'

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

Tabla 239. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQOD_VERSION_1	1	X'00000001'
MQOD_VERSION_2	2	X'00000002'
MQOD_VERSION_3	3	X'00000003'
MQOD_VERSION_4	4	X'00000004'
MQOD_VERSIÓN_ACTUAL	4	X'00000004'
MQOD_LONGITUD_ACTUAL	(value differs by platform or version)	(value differs by platform or version)

MQOII_* (Identificador de instancia de objeto)

Tabla 240. Nombres y valores constantes	
Nombre	Valor
MQOII_NINGUNO	X'00...00' (24 nulos)
MQOII_NONE_ARRAY	'\0','\0',... (24 nulos)

MQOL_* (Longitud original)

Tabla 241. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQOL_INDETERMINADO	-1	X'FFFFFFFF'

MQOM_* (Opciones de Db2 mensajes obsoletas en el grupo Inquire)

Tabla 242. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQOM_NO	0	X'00000000'
MQOM_SÍ	1	X'00000001'

MQOO_* (Opciones abiertas)

Tabla 243. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQOO_BIND_AS_Q_DEF	0	X'00000000'
MQOO_LEER_POR_ADELANTE_COMO_Q_DEF	0	X'00000000'
MQOO_INPUT_AS_Q_DEF	1	X'00000001'
MQOO_INPUT_SHARED	2	X'00000002'
MQOO_INPUT_EXCLUSIVE	4	X'00000004'
MQOO_BROWSE	8	X'00000008'
MQOO_OUTPUT	16	X'00000010'
MQOO_INQUIRE	32	X'00000020'
MQOO_SET	64	X'00000040'
MQOO_SAVE_ALL_CONTEXT	128	X'00000080'
MQOO_PASS_IDENTIDAD_CONTEXTO	256	X'00000100'
MQOO_PASS_ALL_CONTEXT	512	X'00000200'

Tabla 243. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQOO_SET_IDENTITY_CONTEXT	1024	X'00000400'
MQOO_SET_ALL_CONTEXT	2048	X'00000800'
MQOO_ALTERNATE_USER_AUTHORITY	4096	X'00001000'
MQOO_FAIL_IF QUIESCING	8192	X'00002000'
MQOO_BIND_ON_OPEN	16384	X'00004000'
MQOO_BIND_NO_CORREGIDO	32768	X'00008000'
MQOO_CO_OP	131072	X'00020000'
MQOO_RESOLVER_TEMA_LOCAL	262144	X'00040000'
MQOO_NO_LEER_POR_ADELANTE	524288	X'00080000'
MQOO_LEER_POR_ADELANTE	1048576	X'00100000'
MQOO_BIND_ON_GROUP	4194304	X'00400000'

***MQOO_** (Se utiliza únicamente en C++)**

Tabla 244. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQOO_RESOLVER_NOMBRES	65536	X'00010000'
MQOO_RESOLVER_LOCAL_Q	262144	X'00040000'

MQOP_* (Códigos de operación para MQCTL y MQCB)

Códigos de operación para MQCTL

Tabla 245. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQOP_INICIO	1	X'00000001'
MQOP_INICIO_ESPERA	2	X'00000002'
MQOP_STOP	4	X'00000004'

Códigos de operación para MQCB

Tabla 246. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQOP_REGISTRAR	256	X'00000100'
MQOP_DEREGISTRAR	512	X'00000200'

Códigos de operación para MQCTL y MQCB

Tabla 247. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQOP_SUSPENDER	65536	X'00010000'
MQOP_RESUMIR	131072	X'00020000'

MQOPEN_* (Valores relacionados con la estructura MQOPEN_PRIV)

Tabla 248. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQOPEN_PRIV_VERSION_1	1	X'00000001'
MQOPEN_PRIV_VERSIÓN_ACTUAL	1	X'00000001'

MQOPER_* (Operaciones de actividad)

Tabla 249. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQOPER_SISTEMA_PRIMERO	0	X'00000000'
MQOPER_DESCONOCIDO	0	X'00000000'
MQOPER_EXPLORAR	1	X'00000001'
MQOPER_DESCARTAR	2	X'00000002'
MQOPER_GET	3	X'00000003'
MQOPER_PUT	4	X'00000004'
MQOPER_PUT_REPLY	5	X'00000005'
MQOPER_PUT_REPORT	6	X'00000006'
MQOPER_RECEIVE	7	X'00000007'
MQOPER_ENVIAR	8	X'00000008'
MQOPER_TRANSFORM	9	X'00000009'
MQOPER_PUBLISH	10	X'0000000A'
MQOPER_EXCLUIDO_PUBLICADO	11	X'0000000B'
MQOPER_DESCARTADO_PUBLICADO	12	X'0000000C'
MQOPER_SISTEMA_ÚLTIMO	65535	X'0000FFFF'
MQOPER_APPL_FIRST	65536	X'00010000'
MQOPER_APPL_ÚLTIMO	99999999	X'3B9AC9FF'

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

Tipos de objeto

Tabla 250. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQOT_NINGUNO	0	X'00000000'
MQOT_Q	1	X'00000001'
LISTA_DE_NOMBRES_MQOT	2	X'00000002'
MQOT_PROCESO	3	X'00000003'
CLASE_DE_ALMACENAMIENTO_MQOT	4	X'00000004'
MQOT_Q_MGR	5	X'00000005'
MQOT_CANAL	6	X'00000006'
MQOT_AUTH_INFO	7	X'00000007'
MQOT_TEMA	8	X'00000008'

Tabla 250. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQOT_CF_ESTRUCTURA	10	X'0000000A'
MQOT_LISTENER	11	X'0000000B'
MQOT_SERVICIO	12	X'0000000C'
MQOT_RESERVED_1	999	X'000003E7'

Tipos de objetos ampliados

Tabla 251. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQOT_TODOS	1001	X'000003E9'
MQOT_ALIAS_Q	1002	X'000003EA'
MQOT_MODELO_Q	1003	X'000003EB'
MQOT_LOCAL_Q	1004	X'000003EC'
MQOT_REMOTO_Q	1005	X'000003ED'
CANAL_DE_ENVÍO_MQOT	1007	X'000003EF'
MQOT_SERVER_CANAL	1008	X'000003F0'
CANAL_SOLICITANTE_MQOT	1009	X'000003F1'
CANAL_RECEPTOR_MQOT	1010	X'000003F2'
MQOT_CANAL_ACTUAL	1011	X'000003F3'
MQOT_CANAL_GUARDADO	1012	X'000003F4'
MQOT_SVRCONN_CANAL	1013	X'000003F5'
MQOT_CLNTCONN_CANAL	1014	X'000003F6'
MQOT_CANAL_CORTO	1015	X'000003F7'
MQOT_CHLAUTH	1016	X'000003F8'
MQOT_REMOTO_GESTOR_COLA_NOMBRE	1017	X'000003F9'
MQOT_PROT_POLICY	1019	X'000003FB'
MQOT_TT_CANAL	1020	X'000003FC'
MQOT_CANAL_AMQP	1021	X'000003FD'
MQOT_AUTH_REC	1022	X'000003FE'

MQPA_* (Autoridad de colocación)

Tabla 252. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQPA_DEFAULT	1	X'00000001'
MQPA_CONTEXTO	2	X'00000002'
MQPA_SOLO_MCA	3	X'00000003'
MQPA_ALTERNATIVO_O_MCA	4	X'00000004'

MQPD_* (Descriptor de propiedad, soporte y contexto)

Estructura del descriptor de propiedad

Tabla 253. Estructuras de constantes	
Nombre	Estructura
MQPD_STRUC_ID	"PD¬¬"
MQPD_STRUC_ID_ARRAY	'P','D','¬','¬'

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

Tabla 254. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQPD_VERSION_1	1	X'00000001'
MQPD_VERSION_ACTUAL	1	X'00000001'

Opciones de descriptores de propiedades

Tabla 255. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQPD_NINGUNO	0	X'00000000'

Opciones de asistencia inmobiliaria

Tabla 256. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQPD_SOPORTE_OPCIONAL	1	X'00000001'
MQPD_SOPORTE_NECESARIO	1048576	X'00100000'
MQPD_SOPORTE_REQUERIDO_SI_LOCAL	1024	X'00000400'
MQPD_RECHAZAR_MÁSCARA_NO_SOPORTADA	-1048576	X'FFF00000'
MQPD_ACEPTAR_NO_SOPORTADO_SI_MÁSCARA_DE_TRANSMISIÓN	1047552	X'000FFC00'
MQPD_ACEPTAR_MÁSCARA_NO_SOPORTADA	1023	X'000003FF'

Contexto inmobiliario

Tabla 257. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQPD_NO_CONTEXT	0	X'00000000'
MQPD_USUARIO_CONTEXTO	1	X'00000001'

MQPER_* (Valores de persistencia)

Tabla 258. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQPER_PERSISTENCIA_COMO_PADRE	-1	X'FFFFFFFF'
MQPER_NOT_PERSISTENT	0	X'00000000'
MQPER_PERSISTENT	1	X'00000001'

Tabla 258. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQPER_PERSISTENCE_AS_Q_DEF	2	X'00000002'
MQPER_PERSISTENCIA_COMO_TEMA_DEF	2	X'00000002'

MQPL_* (Plataformas)

Tabla 259. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQPL_MVS	1	X'00000001'
MQPL_OS390	1	X'00000001'
MQPL_ZOS	1	X'00000001'
MQPL_OS2	2	X'00000002'
MQPL_AIX	3	X'00000003'
MQPL_UNIX	3	X'00000003'
MQPL_OS400	4	X'00000004'
MQPL_WINDOWS	5	X'00000005'
MQPL_WINDOWS_NT	11	X'0000000B'
MQPL_VMS	12	X'0000000C'
MQPL_NSK	13	X'0000000D'
MQPL_OPEN_TP1	15	X'0000000F'
MQPL_VM	18	X'00000012'
MQPL_TPF	23	X'00000017'
MQPL_VSE	27	X'0000001B'
MQPL_ELECTRODOMÉSTICO	28	X'0000001C'
MQPL_NATIVO	1	X'00000001'

MQPMO_* (Opciones de mensaje y estructura para la máscara de publicación)

Estructura de opciones de mensajes

Tabla 260. Estructuras de constantes	
Nombre	Estructura
MQPMO_STRUC_ID	"PMO¬"
MQPMO_STRUC_ID_ARRAY	'P','M','O','¬'

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

Tabla 261. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQPMO_VERSION_1	1	X'00000001'
MQPMO_VERSION_2	2	X'00000002'
MQPMO_VERSION_3	3	X'00000003'
MQPMO_VERSION_ACTUAL	3	X'00000003'

Tabla 261. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQPMO_LONGITUD_ACTUAL	(value differs by platform or version)	(value differs by platform or version)

Opciones de mensaje

Tabla 262. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQPMO_PUNTO_DE_SINCRONIZACIÓN	2	X'00000002'
MQPMO_NO_PUNTO_DE_SINCRONIZACIÓN	4	X'00000004'
MQPMO_DEFAULT_CONTEXT	32	X'00000020'
MQPMO_NUEVO_ID_MENSAJE	64	X'00000040'
MQPMO_NEW_CORREL_ID	128	X'00000080'
MQPMO_PASS_IDENTITY_CONTEXT	256	X'00000100'
MQPMO_PASS_ALL_CONTEXT	512	X'00000200'
MQPMO_SET_IDENTITY_CONTEXT	1024	X'00000400'
MQPMO_SET_ALL_CONTEXT	2048	X'00000800'
MQPMO_ALTERNATE_USER_AUTHORITY	4096	X'00001000'
MQPMO_FAIL_IF QUIESCING	8192	X'00002000'
MQPMO_NO_CONTEXT	16384	X'00004000'
MQPMO_LOGICAL_ORDER	32768	X'00008000'
MQPMO_RESPUESTA_ASÍNCRONA	65536	X'00010000'
MQPMO_SYNC_RESPUESTA	131072	X'00020000'
MQPMO_RESOLVE_LOCAL_Q	262144	X'00040000'
MQPMO_RETENER	2097152	X'00200000'
MQPMO_MD_SOLO_PARA_SALIDA	8388608	X'00800000'
MQPMO_ÁMBITO_QMGR	67108864	X'04000000'
MQPMO_SUPPRIMIR_RESPUESTA	134217728	X'08000000'
MQPMO_NOT_OWN_SUBS	268435456	X'10000000'
MQPMO_RESPUESTA_COMO_Q_DEF	0	X'00000000'
MQPMO_RESPUESTA_COMO_TEMA_DEF	0	X'00000000'
MQPMO_NINGUNO	0	X'00000000'

Opciones de mensaje para publicar máscara

Tabla 263. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQPMO_PUB_OPTIONS_MASK	2097152	X'00200000'

MQPMRF_* (Campos de registro de mensajes de envío)

Tabla 264. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQPMRF_MSG_ID	1	X'00000001'

Tabla 264. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQPMRF_CORREL_ID	2	X'00000002'
MQPMRF_ID_GRUPO	4	X'00000004'
MQPMRF_COMENTARIOS	8	X'00000008'
MQPMRF_CONTABILIDAD_TOKEN	16	X'00000010'
MQPMRF_NONE	0	X'00000000'

MQPO_* (Opciones de purga del formato de comando)

Tabla 265. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQPO_SÍ	1	X'00000001'
MQPO_NO	0	X'00000000'

MQPRI_* (Prioridad)

Tabla 266. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQPRI_PRIORITY_AS_Q_DEF	-1	X'FFFFFFFF'
MQPRI_PRIORIDAD_COMO_PADRE	-2	X'FFFFFFFE'
MQPRI_PRIORIDAD_COMO_PUBLICADO	-3	X'FFFFFFFD'
MQPRI_PRIORIDAD_COMO_TEMA_DEF	-1	X'FFFFFFFF'

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

Valores de control de propiedades de cola y canal

Tabla 267. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQPROP_COMPATIBILIDAD	0	X'00000000'
MQPROP_NONE	1	X'00000001'
MQPROP_TODOS	2	X'00000002'
MQPROP_FORCE_MQRFH2	3	X'00000003'

Longitud máxima propiedades

Tabla 268. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQPROP_LONGITUD_ILIMITADA	-1	X'FFFFFFFF'

MQPRT_* (Valores de respuesta de Put)

Tabla 269. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
RESPUESTA_MQPRT_COMO_PADRE	0	X'00000000'

Tabla 269. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQPRT_SYNC_RESPUESTA	1	X'00000001'
MQPRT_RESPUESTA_ASÍNCRONA	2	X'00000002'

MQPS_* (Publicar/Suscribirse)

Formato de comando Publicar/Suscribirse Estado

Tabla 270. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQPS_STATUS_INACTIVO	0	X'00000000'
MQPS_STATUS_INICIANDO	1	X'00000001'
MQPS_STATUS_DETENIÉNDOSE	2	X'00000002'
MQPS_STATUS_ACTIVE	3	X'00000003'
MQPS_STATUS_COMPATIBLE	4	X'00000004'
MQPS_STATUS_ERROR	5	X'00000005'
MQPS_STATUS_RECHAZADO	6	X'00000006'

Publicar/suscribirse Etiquetas como cadenas

MQPS_COMANDO	"MQPSCommand"
MQPS_CÓDIGO_COMP	"MQPSCompCode"
MQPS_CORREL_ID	"MQPSCorrelId"
MQPS_DELETE_OPTIONS	"MQPSDelOpts"
MQPS_ERROR_ID	"MQPSErrorId"
ERROR_POS_MQPS	"MQPSErrorPos"
MQPS_INTEGER_DATA	"MQPSIntData"
MQPS_PARÁMETRO_ID	"MQSParmId"
MQPS_OPCIONES_DE_PUBLICACIÓN	"MQSPubOpts"
MQPS_PUBLISH_TIMESTAMP	"MQSPubTime"
MQPS_Q_MGR_NAME	"MQPSQMgrName"
MQPS_Q_NAME	"MQPSQName"
MQPS_RAZÓN	"MQPSReason"
MQPS_RAZÓN_TEXTO	"MQPSReasonText"
MQPS_OPCIONES_DE_REGISTRO	"MQPSRegOpts"
NÚMERO DE SECUENCIA MQPS	"MQPSSeqNum"
MQPS_STREAM_NAME	"MQPSStreamName"
MQPS_STRING_DATA	"MQPSStringData"
MQPS_SUSCRIPCIÓN_IDENTIDAD	"MQPSSubIdentity"
MQPS_NOMBRE_DE_LA_SUSCRIPCIÓN	"MQPSSubName"

MQPS_SUSCRIPCIÓN_DATOS_DE_USUARIO	"MQPSSubUserData"
MQPS_TEMA	"MQPSTopic"
MQPS_ID_USUARIO	"MQPSUserId"

Publicar/suscribirse Etiquetas como cadenas entre comillas vacías

MQPS_COMANDO_B	"~MQPSCommand~"
MQPS_CÓDIGO_COMP_B	"~MQPSCompCode~"
MQPS_CORREL_ID_B	"~MQPSCorrelId~"
MQPS_DELETE_OPTIONS_B	"~MQPSDelOpts~"
MQPS_ERROR_ID_B	"~MQPSErrorId~"
ERROR_POSICIÓN_MQPS_B	"~MQPSErrorPos~"
MQPS_INTEGER_DATA_B	"~MQPSIntData~"
MQPS_PARÁMETRO_ID_B	"~MQSParmId~"
MQPS_OPCIONES_DE_PUBLICACIÓN_B	"~MQSPubOpts~"
MQPS_PUBLISH_TIMESTAMP_B	"~MQSPubTime~"
MQPS_Q_MGR_NAME_B	"~MQPSQMgrName~"
MQPS_Q_NOMBRE_B	"~MQPSQName~"
MQPS_RAZÓN_B	"~MQPSReason~"
MQPS_MOTIVO_TEXTO_B	"~MQPSReasonText~"
MQPS_OPCIONES_DE_REGISTRO_B	"~MQPSRegOpts~"
NÚMERO_DE_SECUENCIA_MQPS_B	"~MQPSSeqNum~"
MQPS_STREAM_NAME_B	"~MQPSStreamName~"
MQPS_STRING_DATA_B	"~MQPSStringData~"
MQPS_SUSCRIPCIÓN_IDENTIDAD_B	"~MQPSSubIdentity~"
MQPS_NOMBRE_DE_LA_SUSCRIPCIÓN_B	"~MQPSSubName~"
MQPS_SUSCRIPCIÓN_DATOS_USUARIO_B	"~MQPSSubUserData~"
MQPS_TEMA_B	"~MQPSTopic~"
MQPS_ID_USUARIO_B	"~MQPSUserId~"

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

Publicar/suscribirse a valores de etiquetas de comando como cadenas

MQPS_ELIMINAR_PUBLICACIÓN	"DeletePub"
MQPS_DEREGISTRAR_EDITOR	"DeregPub"
MQPS_DEREGISTRAR_SUSCRIPTOR	"DeregSub"
MQPS_PUBLICAR	"Publish"
MQPS_REGISTRAR_EDITOR	"RegPub"

MQPS_REGISTRAR_SUSCRIPTOR	"RegSub"
MQPS_SOLICITAR_ACTUALIZACIÓN	"ReqUpdate"

Publicar/suscribirse a los valores de las etiquetas de comando como cadenas entre comillas

MQPS_DELETE_PUBLICATION_B	"¬DeletePub¬"
MQPS_DEREGISTRAR_EDITOR_B	"¬DeregPub¬"
MQPS_DEREGISTRAR_SUSCRIPTOR_B	"¬DeregSub¬"
MQPS_PUBLISH_B	"¬Publish¬"
MQPS_REGISTRAR_EDITOR_B	"¬RegPub¬"
MQPS_REGISTRAR_SUSCRIPTOR_B	"¬RegSub¬"
MQPS_SOLICITUD_ACTUALIZACIÓN_B	"¬ReqUpdate¬"

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

Opciones de publicación/suscripción Valores de etiqueta como cadenas

MQPS_AÑADIR_NOMBRE	"AddName"
MQPS_ANÓNIMO	"Anon"
MQPS_CORREL_ID_AS_IDENTITY	"CorrelAsId"
MQPS_DEREGISTRAR_TODOS	"DeregAll"
MQPS_SOLICITUDES_DIRECTAS	"DirectReq"
MQPS_DUPLICADOS_OK	"DupsOK"
RESPUESTA_COMPLETA_MQPS	"FullResp"
MQPS_INCLUIR_NOMBRE_DE_FLUJO	"InclStreamName"
MQPS_INFORMAR_SI_RETENIDO	"InformIfRet"
MQPS_IS_RETAINED_PUBLICATION	"IsRetainedPub"
MQPS_JOIN_EXCLUSIVO	"JoinExcl"
MQPS_JOIN_SHARED	"JoinShared"
MQPS_DEJAR_SOLO	"LeaveOnly"
MQPS_LOCAL	"Local"
MQPS_BLOQUEADO	"Locked"
MQPS_NUEVAS_PUBLICACIONES_SOLO	"NewPubsOnly"
MQPS_SIN_MODIFICACIÓN	"NoAlter"
MQPS_SIN_REGISTRO	"NoReg"
MQPS_NO_PERSISTENTE	"NonPers"
MQPS_NONE	"None"
MQPS_OTROS_SUSCRIPTORES_SOLO	"OtherSubsOnly"
MQPS_PERSISTENTE	"Pers"

MQPS_PERSISTENTE_COMO_PUBLICACIÓN	"PersAsPub"
MQPS_PERSISTENTE_COMO_Q	"PersAsQueue"
MQPS_PUBLICAR_SOLO_A_PEDIDO	"PubOnReqOnly"
MQPS_RETENER_PUBLICACIÓN	"RetainPub"
MQPS_ID_DE_USUARIO_VARIABLE	"VariableUserId"

Opciones de publicación/suscripción Valores de etiqueta como cadenas entre comillas

MQPS_AÑADIR_NOMBRE_B	"¬AddName¬"
MQPS_ANÓNIMO_B	"¬Anon¬"
MQPS_CORREL_ID_AS_IDENTITY_B	"¬CorrelAsId¬"
MQPS_DEREGISTRAR_TODOS_B	"¬DeregAll¬"
MQPS_SOLICITUDES_DIRECTAS_B	"¬DirectReq¬"
MQPS_DUPLICADOS_ACEPTABLES_B	"¬DupsOK¬"
MQPS_RESPUESTA_COMPLETA_B	"¬FullResp¬"
MQPS_INCLUIR_NOMBRE_DE_FLUJO_B	"¬InclStreamName¬"
MQPS_INFORMAR_SI_RETENIDO_B	"¬InformIfRet¬"
MQPS_IS_RETAINED_PUBLICATION_B	"¬IsRetainedPub¬"
MQPS_JOIN_EXCLUSIVO_B	"¬JoinExcl¬"
MQPS_UNIRSE_COMPARTIDO_B	"¬JoinShared¬"
MQPS_DEJAR_SOLO_B	"¬LeaveOnly¬"
MQPS_LOCAL_B	"¬Local¬"
MQPS_LOCKED_B	"¬Locked¬"
MQPS_NUEVAS_PUBLICACIONES_SOLO_B	"¬NewPubsOnly¬"
MQPS_SIN_MODIFICACIÓN_B	"¬NoAlter¬"
MQPS_SIN_REGISTRO_B	"¬NoReg¬"
MQPS_NO_PERSISTENTE_B	"¬NonPers¬"
MQPS_NONE_B	"¬None¬"
MQPS_OTROS_SUSCRIPTORES_SOLO_B	"¬OtherSubsOnly¬"
MQPS_PERSISTENTE_B	"¬Pers¬"
MQPS_PERSISTENTE_COMO_PUBLICAR_B	"¬PersAsPub¬"
MQPS_PERSISTENTE_COMO_Q_B	"¬PersAsQueue¬"
MQPS_PUBLISH_ON_REQUEST_ONLY_B	"¬PubOnReqOnly¬"
MQPS_RETENER_PUBLICACIÓN_B	"¬RetainPub¬"
MQPS_VARIABLE_ID_USUARIO_B	"¬VariableUserId¬"

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

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

Tabla 271. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQPSC_CARPETA_VERSION	1	X'00000001'

MQPSC_* (Etiquetas de opciones de publicación/suscripción Nombres de etiquetas)

MQPSC_COMANDO	"Command"
MQPSC_OPCIÓN_DE_INSCRIPCIÓN	"RegOpt"
MQPSC_OPCIÓN_PUBLICACIÓN	"PubOpt"
MQPSC_DELETE_OPTION	"DelOpt"
MQPSC_TEMA	"Topic"
PUNTO DE SUSCRIPCIÓN MQPSC	"SubPoint"
FILTRO MQPSC	"Filter"
MQPSC_Q_MGR_NAME	"QMgrName"
MQPSC_Q_NAME	"QName"
MQPSC_PUBLISH_TIMESTAMP	"PubTime"
NÚMERO DE SECUENCIA MQPSC	"SeqNum"
MQPSC_NOMBRE_DE_LA_SUSCRIPCIÓN	"SubName"
MQPSC_SUSCRIPCIÓN_IDENTIDAD	"SubIdentity"
MQPSC_SUSCRIPCIÓN_DATOS_DE_USUARIO	"SubUserData"
MQPSC_CORREL_ID	"CorrelId"

MQPSC_* (Nombres de etiquetas XML de opciones de publicación/suscripción)

MQPSC_COMANDO_B	"<Command>"
MQPSC_COMANDO_E	"</Command>"
MQPSC_OPCIÓN_DE_INSCRIPCIÓN_B	"<RegOpt>"
MQPSC_OPCIÓN_DE_INSCRIPCIÓN_E	"</RegOpt>"
MQPSC_OPCIÓN_DE_PUBLICACIÓN_B	"<PubOpt>"
MQPSC_OPCIÓN_DE_PUBLICACIÓN_E	"</PubOpt>"
MQPSC_DELETE_OPTION_B	"<DelOpt>"
MQPSC_DELETE_OPTION_E	"</DelOpt>"
MQPSC_TEMA_B	"<Topic>"
MQPSC_TEMA_E	"</Topic>"
Punto de suscripción MQPSC	"<SubPoint>"
Punto de suscripción MQPSC	"</SubPoint>"
MQPSC_FILTRO_B	"<Filter>"
MQPSC_FILTRO_E	"</Filter>"

MQPSC_Q_MGR_NAME_B	"<QMgrName>"
MQPSC_Q_MGR_NAME_E	"</QMgrName>"
MQPSC_Q_NOMBRE_B	"<QName>"
MQPSC_Q_NAME_E	"</QName>"
MQPSC_PUBLISH_TIMESTAMP_B	"<PubTime>"
MQPSC_PUBLISH_TIMESTAMP_E	"</PubTime>"
NÚMERO DE SECUENCIA MQPSC	"<SeqNum>"
NÚMERO DE SECUENCIA MQPSC	"</SeqNum>"
MQPSC_NOMBRE_DE_LA_SUSCRIPCIÓN_B	"<SubName>"
MQPSC_NOMBRE_DE_LA_SUSCRIPCIÓN_E	"</SubName>"
MQPSC_SUSCRIPCIÓN_IDENTIDAD_B	"<SubIdentity>"
MQPSC_SUSCRIPCIÓN_IDENTIDAD_E	"</SubIdentity>"
MQPSC_SUSCRIPCIÓN_DATOS_DE_USUARIO_B	"<SubUserData>"
MQPSC_SUSCRIPCIÓN_DATOS_DE_USUARIO_E	"</SubUserData>"
MQPSC_CORREL_ID_B	"<CorrelId>"
MQPSC_CORREL_ID_E	"</CorrelId>"

MQPSC_* (Opciones de publicación/suscripción. Valores del editor de etiquetas como cadenas)

MQPSC_ELIMINAR_PUBLICACIÓN	"DeletePub"
MQPSC_DEREGISTRAR_SUSCRIPTOR	"DeregSub"
MQPSC_PUBLICAR	"Publish"
MQPSC_REGISTRAR_SUSCRIPTOR	"RegSub"
MQPSC_SOLICITAR_ACTUALIZACIÓN	"ReqUpdate"

MQPSC_* (Valores de nombre de etiqueta de opciones de publicación/suscripción como cadenas)

MQPSC_AÑADIR_NOMBRE	"AddName"
MQPSC_CORREL_ID_AS_IDENTITY	"CorrelAsId"
MQPSC_DEREGISTRAR_TODOS	"DeregAll"
MQPSC_DUPLICADOS_OK	"DupsOK"
MQPSC_RESPUESTA_COMPLETA	"FullResp"
MQPSC_INFORMAR_SI_RETENIDO	"InformIfRet"
MQPSC_IS_RETAINED_PUB	"IsRetainedPub"
MQPSC_UNIRSE_COMPARTIDO	"JoinShared"
MQPSC_JOIN_EXCLUSIVO	"JoinExcl"
MQPSC_DEJAR_SOLO	"LeaveOnly"
MQPSC_LOCAL	"Local"

MQPSC_BLOQUEADO	"Locked"
MQPSC_NUEVAS_PUBLICACIONES_SOLO	"NewPubsOnly"
MQPSC_SIN_MODIFICACIÓN	"NoAlter"
MQPSC_NO_PERSISTENTE	"NonPers"
MQPSC_OTROS_SUSCRIPCIONES_SOLO	"OtherSubsOnly"
MQPSC_PERSISTENTE	"Pers"
MQPSC_PERSISTENTE_COMO_PUBLICADO	"PersAsPub"
MQPSC_PERSISTENTE_COMO_Q	"PersAsQueue"
MQPSC_NINGUNO	"None"
MQPSC_PUB_SOLO_BAJO_SOLICITUD	"PubOnReqOnly"
MQPSC_RETENER_PUB	"RetainPub"
MQPSC_ID_DE_USUARIO_VARIABLE	"VariableUserId"

MQPSCR_* (Opciones de publicación/suscripción)

Opciones de publicación/suscripción Etiqueta de respuesta de carpeta de publicación/suscripción (pscr) Etiquetas

Tabla 272. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQPSCR_CARPETA_VERSION	1	X'00000001'

Opciones de publicación/suscripción Etiqueta Nombres de etiquetas

MQPSCR_FINALIZACIÓN	"Completion"
RESPUESTA_MQPSCR	"Response"
MQPSCR_MOTIVO	"Reason"

Opciones de publicación/suscripción Etiqueta Nombres de etiquetas XML

MQPSCR_COMPLETION_B	"<Completion>"
MQPSCR_COMPLETION_E	"</Completion>"
MQPSCR_RESPUESTA_B	"<Response>"
MQPSCR_RESPUESTA_E	"</Response>"
MQPSCR_RAZÓN_B	"<Reason>"
MQPSCR_RAZÓN_E	"</Reason>"

Opciones de publicación/suscripción Etiqueta Valores de etiqueta

MQPSCR_OK	"ok"
MQPSCR_ADVERTENCIA	"warning"
ERROR_MQPSCR	"error"

MQPSM_* (modo Pub/Sub)

Tabla 273. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQPSM_DESACTIVADO	0	X'00000000'
MQPSM_COMPATIBLE	1	X'00000001'
MQPSM_HABILITADO	2	X'00000002'

MQPSPROP_* (Propiedades de mensajes Pub/Sub)

Tabla 274. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQPSPROP_NONE	0	X'00000000'
MQPSPROP_COMPATIBLE	1	X'00000001'
MQPSPROP_RFH2	2	X'00000002'
MQPSPROP_MSGPROP	3	X'00000003'

MQPSST_* (Formato de comando Tipo de estado Pub/Sub)

Tabla 275. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQPSST_TODOS	0	X'00000000'
MQPSST_LOCAL	1	X'00000001'
MQPSST_PADRE	2	X'00000002'
MQPSST_NIÑO	3	X'00000003'

MQPUBO_* (Opciones de publicación/suscripción)

Tabla 276. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQPUBO_NONE	0	X'00000000'
MQPUBO_CORREL_ID_AS_IDENTITY	1	X'00000001'
MQPUBO_RETENER_PUBLICACIÓN	2	X'00000002'
MQPUBO_OTROS_SUSCRIPTORES_SOLO	4	X'00000004'
MQPUBO_SIN_REGISTRO	8	X'00000008'
MQPUBO_SE_CONSERVA_PUBLICACIÓN	16	X'00000010'

MQPXP_* (Estructura de parámetros de salida de enrutamiento de publicación/suscripción)

Tabla 277. Estructuras de constantes	
Nombre	Estructura
MQPXP_STRUC_ID	"PXP¬"
MQPXP_STRUC_ID_ARRAY	'P', 'X', 'P', '¬'

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

Tabla 278. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQPXP_VERSION_1	1	X'00000001'
MQPXP_VERSION_ACTUAL	1	X'00000001'

MQQA_* (Atributos de cola)

Inhibir obtener valores

Tabla 279. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQQA_GET_INHIBITED	1	X'00000001'
MQQA_GET_ALLOWED	0	X'00000000'

Inhibir valores de puesta

Tabla 280. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQQA_PUT_INHIBITED	1	X'00000001'
MQQA_PUT_ALLOWED	0	X'00000000'

Compartibilidad de colas

Tabla 281. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQQA_SHAREABLE	1	X'00000001'
MQQA_NOT_SHAREABLE	0	X'00000000'

Endurecimiento por retroceso

Tabla 282. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQQA_BACKOUT_HARDENED	1	X'00000001'
MQQA_RETROCESO_NO_ENDURECIDO	0	X'00000000'

MQQDT_* (Tipos de definición de cola)

Tabla 283. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQQDT_PREDEFINED	1	X'00000001'
MQQDT_PERMANENT_DYNAMIC	2	X'00000002'
MQQDT_TEMPORARY_DYNAMIC	3	X'00000003'
MQQDT_COMPARTIDO_DINÁMICO	4	X'00000004'

MQQF_* (Indicadores de cola)

Tabla 284. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQQF_LOCAL_Q	1	X'00000001'
MQQF_CLWL_USEQ_CUALQUIER	64	X'00000040'
MQQF_CLWL_USEQ_LOCAL	128	X'00000080'

MQQMDT_* (Tipos de definición del gestor de colas de formato de comando)

Tabla 285. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQQMDT_EXPLÍCITO_ENVIADOR_DE_CLÚSTER	1	X'00000001'
MQQMDT_AUTO_CLUSTER_SENDER	2	X'00000002'
MQQMDT_AUTO_EXP_CLUSTER_SENDER	4	X'00000004'
Receptor de clúster MQQMDT	3	X'00000003'

MQQMF_* (Indicadores del gestor de colas)

Tabla 286. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQQMF_REPOSITORIO_Q_MGR	2	X'00000002'
MQQMF_CLUSSDR_DEFINIDO_POR_EL_USUARIO	8	X'00000008'
MQQMF_CLUSSDR_AUTO_DEFINIDO	16	X'00000010'
MQQMF_DISPONIBLE	32	X'00000020'

MQQMFAC_* (Formato de comando del gestor de colas)

Tabla 287. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQQMFAC_IMS_PUENTE	1	X'00000001'
MQQMFAC_DB2	2	X'00000002'

MQQMSTA_* (Formato de comando Estado del gestor de colas)

Tabla 288. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQQMSTA_INICIO	1	X'00000001'
MQQMSTA_EN_MARCHA	2	X'00000002'
MQQMSTA_EN ESTADO DE REPOSO	3	X'00000003'

MQQMT_* (Tipos de administrador de colas de formato de comando)

Tabla 289. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQQMT_NORMAL	0	X'00000000'
MQQMT_REPOSITORIO	1	X'00000001'

MQQO_* (Opciones de comando para detener temporalmente)

Tabla 290. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQQO_SÍ	1	X'00000001'
MQQO_NO	0	X'00000000'

MQQS_* (Opciones de algoritmos a prueba de quantes)

Tabla 291. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQQS_DESCONOCIDO	-1	X'FFFFFFFF'
MQQS_SIN PROTECCIÓN	0	X'00000000'
MQQS_HYBMLKEM	1	X'00000001'
MQQS_MLKEM	2	X'00000002'

MQQSGD_* (Disposiciones de grupos de intercambio de colas)

Tabla 292. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQQSGD_TODOS	-1	X'FFFFFFFF'
MQQSGD_Q_MGR	0	X'00000000'
MQQSGD_COPY	1	X'00000001'
MQQSGD_COMPARTIDO	2	X'00000002'
MQQSGD_GRUPO	3	X'00000003'
MQQSGD_PRIVADO	4	X'00000004'
MQQSGD_EN VIVO	6	X'00000006'

MQQSGS_* (Estado del grupo de intercambio de colas de formato de comando)

Tabla 293. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQQSGS_DESCONOCIDO	0	X'00000000'
MQQSGS_CREADO	1	X'00000001'
MQQSGS_ACTIVO	2	X'00000002'
MQQSGS_INACTIVO	3	X'00000003'
MQQSGS_FALLÓ	4	X'00000004'
MQQSGS_PENDENTE	5	X'00000005'

MQQSIE_* (Formato de comando Servicio de cola-Eventos de intervalo)

Tabla 294. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQQSIE_NONE	0	X'00000000'
MQQSIE_HIGH	1	X'00000001'

Tabla 294. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQQSIE_OK	2	X'00000002'

MQQSO_* (Opciones de estado de cola de formato de comando abierto para SET, BROWSE, INPUT)

Tabla 295. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQQSO_NO	0	X'00000000'
MQQSO_SÍ	1	X'00000001'
MQQSO_COMPARTIDO	1	X'00000001'
MQQSO_EXCLUSIVO	2	X'00000002'

MQQSOT_* (Formato de comando Tipos de estado de cola abierta)

Tabla 296. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQQSOT TODOS	1	X'00000001'
MQQSOT_ENTRADA	2	X'00000002'
MQQSOT_SALIDA	3	X'00000003'

MQQSR_* (Opciones necesarias para la seguridad cuántica)

Tabla 297. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQQSR_OPCIONAL	0	X'00000000'
MQQSR_OBLIGATORIO	1	X'00000001'

MQQSUM_* (Formato de comando Estado de la cola Mensajes no confirmados)

Tabla 298. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQQSUM_SÍ	1	X'00000001'
MQQSUM_NO	0	X'00000000'

MQQT_* (Tipos de colas y tipos de colas extendidas)

Tipos de cola

Tabla 299. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQQT_LOCAL	1	X'00000001'
MQQT_MODEL	2	X'00000002'
MQQT_ALIAS	3	X'00000003'
MQQT_REMOTE	6	X'00000006'

Tabla 299. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQQT_CLUSTER	7	X'00000007'

Tipos de colas extendidas

Tabla 300. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQQT_TODOS	1001	X'000003E9'

MQRC_* (códigos de motivo)

Tabla 301. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQRC_NONE	0	X'00000000'
MQRC_APPL_FIRST	900	X'00000384'
MQRC_APPL_ÚLTIMO	999	X'000003E7'
MQRC_ALIAS_BASE_Q_TIPO_ERROR	2001	X'000007D1'
MQRC_YA_CONECTADO	2002	X'000007D2'
MQRC_BACKED_OUT	2003	X'000007D3'
MQRC_BUFFER_ERROR	2004	X'000007D4'
MQRC_BUFFER_LENGTH_ERROR	2005	X'000007D5'
MQRC_CHAR_ATTR_LENGTH_ERROR	2006	X'000007D6'
MQRC_CHAR_ATTRS_ERROR	2007	X'000007D7'
MQRC_CHAR_ATTRS_TOO_SHORT	2008	X'000007D8'
MQRC_CONNECTION_BROKEN	2009	X'000007D9'
MQRC_DATA_LENGTH_ERROR	2010	X'000007DA'
MQRC_DYNAMIC_Q_NAME_ERROR	2011	X'000007DB'
MQRC_ERROR_DE_ENTORNO	2012	X'000007DC'
ERROR_CADUCIDAD_MQRC	2013	X'000007DD'
MQRC_ERROR_DE_RESPUESTA	2014	X'000007DE'
MQRC_GET_INHIBITED	2016	X'000007E0'
MQRC_HANDLE_NO_DISPONIBLE	2017	X'000007E1'
MQRC_HCONN_ERROR	2018	X'000007E2'
MQRC_HOBJ_ERROR	2019	X'000007E3'
MQRC_INHIBIR_VALOR_ERROR	2020	X'000007E4'
MQRC_INT_ATTR_COUNT_ERROR	2021	X'000007E5'
MQRC_INT_ATTR_COUNT_TOO_SMALL	2022	X'000007E6'
MQRC_INT_ATTRS_ARRAY_ERROR	2023	X'000007E7'
MQRC_SYNCPOINT_LIMIT_REACHED	2024	X'000007E8'
MQRC_LÍMITE_MÁXIMO_DE_CONEXIONES_ALCANZADO	2025	X'000007E9'
MQRC_MD_ERROR	2026	X'000007EA'
MQRC_RESPUESTA_FALTANTE_A_P	2027	X'000007EB'

Tabla 301. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQRC_MSG_TYPE_ERROR	2029	X'000007ED'
MQRC_MSG_DEMASIADO_GRANDE_PARA_Q	2030	X'000007EE'
MQRC_MSG_DEMASIADO_GRANDE_PARA_Q_MGR	2031	X'000007EF'
MQRC_NO_MSG_AVAILABLE	2033	X'000007F1'
MQRC_NO_MSG_UNDER_CURSOR	2034	X'000007F2'
MQRC_NOT_AUTHORIZED	2035	X'000007F3'
MQRC_NOT_OPEN_FOR_BROWSE	2036	X'000007F4'
MQRC_NOT_OPEN_FOR_INPUT	2037	X'000007F5'
MQRC_NO_ABIERTO_PARA_CONSULTAS	2038	X'000007F6'
MQRC_NO_ABIERTO_PARA_SALIDA	2039	X'000007F7'
MQRC_NO_ABIERTO_PARA_CONFIGURAR	2040	X'000007F8'
MQRC_OBJECT_CHANGED	2041	X'000007F9'
MQRC_OBJECT_IN_USE	2042	X'000007FA'
MQRC_TIPO_OBJETO_ERROR	2043	X'000007FB'
MQRC_OD_ERROR	2044	X'000007FC'
MQRC_OPCIÓN_NO_VÁLIDA_PARA_EL_TIPO	2045	X'000007FD'
MQRC_OPTIONS_ERROR	2046	X'000007FE'
MQRC_PERSISTENCIA_ERROR	2047	X'000007FF'
MQRC_PERSISTENTE_NO_PERMITIDO	2048	X'00000800'
MQRC_PRIORIDAD_SUPERA_MÁXIMO	2049	X'00000801'
MQRC_ERROR_PRIORIDAD	2050	X'00000802'
MQRC_PUT_INHIBITED	2051	X'00000803'
MQRC_Q_DELETED	2052	X'00000804'
MQRC_Q_COMPLETO	2053	X'00000805'
MQRC_Q_NOT_EMPTY	2055	X'00000807'
MQRC_Q_ESPACIO_NO_DISPONIBLE	2056	X'00000808'
MQRC_Q_TIPO_ERROR	2057	X'00000809'
MQRC_Q_MGR_NAME_ERROR	2058	X'0000080A'
MQRC_Q_MGR_NOT_AVAILABLE	2059	X'0000080B'
MQRC_REPORT_OPTIONS_ERROR	2061	X'0000080D'
MQRC_SECOND_MARK_NOT_ALLOWED	2062	X'0000080E'
MQRC_ERROR_DE_SEGURIDAD	2063	X'0000080F'
ERROR_CON_EL_NÚMERO_DE_SELECCIONES_MQRC	2065	X'00000811'
MQRC_SELECTOR_LÍMITE_EXCEDIDO	2066	X'00000812'
MQRC_SELECTOR_ERROR	2067	X'00000813'
MQRC_SELECTOR_NO_APTO_PARA_TIPO	2068	X'00000814'
MQRC_SIGNAL_OUTSTANDING	2069	X'00000815'
MQRC_SIGNAL_REQUEST_ACCEPTED	2070	X'00000816'
MQRC_STORAGE_NOT_AVAILABLE	2071	X'00000817'

Tabla 301. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQRC_SYNCPOINT_NOT_AVAILABLE	2072	X'00000818'
ERROR_DE_CONTROL_DE_DISPARO_MQRC	2075	X'0000081B'
MQRC_TRIGGER_ERROR_DE_PROFUNDIDAD	2076	X'0000081C'
MQRC_TRIGGER_MSG_PRIORITY_ERR	2077	X'0000081D'
MQRC_TRIGGER_TYPE_ERROR	2078	X'0000081E'
MQRC_TRUNCATED_MSG_ACCEPTED	2079	X'0000081F'
MQRC_TRUNCATED_MSG_FAILED	2080	X'00000820'
MQRC_ALIAS_DESCONOCIDO_BASE_Q	2082	X'00000822'
MQRC_UNKNOWN_OBJECT_NAME	2085	X'00000825'
MQRC_OBJETO_DESCONOCIDO_Q_MGR	2086	X'00000826'
MQRC_DESCONOCIDO_REMOTO_Q_MGR	2087	X'00000827'
MQRC_WAIT_INTERVAL_ERROR	2090	X'0000082A'
MQRC_XMIT_Q_TYPE_ERROR	2091	X'0000082B'
MQRC_XMIT_Q_ERROR_DE_USO	2092	X'0000082C'
MQRC_NO_ABIERTO_PARA_PASAR_TODO	2093	X'0000082D'
MQRC_NO_ABIERTO_PARA_PASAR_IDENTIFICACIÓN	2094	X'0000082E'
MQRC_NO_ABIERTO_PARA_CONFIGURAR_TODO	2095	X'0000082F'
MQRC_NO_ABIERTO_PARA_ESTABLECER_IDENTIFICADOR	2096	X'00000830'
MQRC_CONTEXTO_ERROR_MANEJO	2097	X'00000831'
MQRC_CONTEXTO_NO_DISPONIBLE	2098	X'00000832'
MQRC_SIGNAL1_ERROR	2099	X'00000833'
MQRC_OBJETO_YA_EXISTE	2100	X'00000834'
MQRC_OBJECT_DAMAGED	2101	X'00000835'
MQRC_RESOURCE_PROBLEM	2102	X'00000836'
MQRC_OTRO_GESTOR_DE_COLAS_CONECTADO	2103	X'00000837'
MQRC_OPCIÓN_DE_INFORME_DESCONOCIDA	2104	X'00000838'
MQRC_CLASE_ALMACENAMIENTO_ERROR	2105	X'00000839'
MQRC_CÓDIGO_NO_VÁLIDO_PARA_XCF_Q	2106	X'0000083A'
MQRC_XWAIT_CANCELADO	2107	X'0000083B'
MQRC_XWAIT_ERROR	2108	X'0000083C'
MQRC_SUPPRESSED_BY_EXIT	2109	X'0000083D'
MQRC_FORMAT_ERROR	2110	X'0000083E'
MQRC_SOURCE_CCSDID_ERROR	2111	X'0000083F'
MQRC_SOURCE_INTEGER_ENC_ERROR	2112	X'00000840'
MQRC_SOURCE_DECIMAL_ENC_ERROR	2113	X'00000841'
MQRC_SOURCE_FLOAT_ENC_ERROR	2114	X'00000842'
MQRC_TARGET_CCSDID_ERROR	2115	X'00000843'
MQRC_TARGET_INTEGER_ENC_ERROR	2116	X'00000844'
MQRC_TARGET_DECIMAL_ENC_ERROR	2117	X'00000845'

Tabla 301. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQRC_TARGET_FLOAT_ENC_ERROR	2118	X'00000846'
MQRC_NOT_CONVERTED	2119	X'00000847'
MQRC_CONVERTED_MSG_TOO_BIG	2120	X'00000848'
MQRC_TRUNCADO	2120	X'00000848'
MQRC_SIN_PARTICIPANTES_EXTERNOS	2121	X'00000849'
MQRC_PARTICIPANTE_NO_DISPONIBLE	2122	X'0000084A'
MQRC_RESULTADO_MIXTO	2123	X'0000084B'
MQRC_RESULTADO_PENDIENTE	2124	X'0000084C'
MQRC_PUENTE_INICIADO	2125	X'0000084D'
MQRC_PUENTE_DETENIDO	2126	X'0000084E'
MQRC_ADAPTADOR_ALMACENAMIENTO_ESCASIDAD	2127	X'0000084F'
MQRC_UOW_EN_PROCESO	2128	X'00000850'
ERROR DE CARGA DE CONEXIÓN DEL ADAPTADOR MQRC	2129	X'00000851'
MQRC_ADAPTER_SERV_LOAD_ERROR	2130	X'00000852'
MQRC_ADAPTER_DEFS_ERROR	2131	X'00000853'
MQRC_ADAPTER_DEFS_ERROR_DE_CARGA	2132	X'00000854'
nMQRC_ADAPTER_CONV_LOAD_ERROR	2133	X'00000855'
MQRC_BO_ERROR	2134	X'00000856'
MQRC_DH_ERROR	2135	X'00000857'
MQRC_MÚLTIPLES_RAZONES	2136	X'00000858'
MQRC_ABIERTO_FALLIDO	2137	X'00000859'
ERROR DE CARGA DEL DISCO DEL ADAPTADOR MQRC	2138	X'0000085A'
MQRC_CNO_ERROR	2139	X'0000085B'
MQRC_CICS_WAIT_FAILED	2140	X'0000085C'
MQRC_DLH_ERROR	2141	X'0000085D'
MQRC_ERROR_ENCABEZADO	2142	X'0000085E'
MQRC_SOURCE_LENGTH_ERROR	2143	X'0000085F'
ERROR_DE_LONGITUD_DE_DESTINO_MQRC	2144	X'00000860'
MQRC_SOURCE_BUFFER_ERROR	2145	X'00000861'
MQRC_TARGET_BUFFER_ERROR	2146	X'00000862'
MQRC_IIH_ERROR	2148	X'00000864'
MQRC_PCF_ERROR	2149	X'00000865'
MQRC_DBCS_ERROR	2150	X'00000866'
MQRC_OBJETO_NOMBRE_ERROR	2152	X'00000868'
MQRC_OBJETO_Q_MGR_NOMBRE_ERROR	2153	X'00000869'
MQRC_RECS_PRESENT_ERROR	2154	X'0000086A'
ERROR_DE_REGISTROS_DE_OBJETOS_MQRC	2155	X'0000086B'
ERROR_EN_LOS_REGISTROS_DE_RESPUESTA_MQRC	2156	X'0000086C'
MQRC_ASID_MISMATCH	2157	X'0000086D'

Tabla 301. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQRC_PMO_REGISTRO_FLAGS_ERROR	2158	X'0000086E'
MQRC_PUT_MSG_RECORDS_ERROR	2159	X'0000086F'
MQRC_CONN_ID_EN_USO	2160	X'00000870'
MQRC_Q_MGR QUIESCING	2161	X'00000871'
MQRC_Q_MGR_STOPPING	2162	X'00000872'
MQRC_DUPLICADO_RECUPERACIÓN_COORDENADAS	2163	X'00000873'
MQRC_PMO_ERROR	2173	X'0000087D'
MQRC_API_SALIDA_NO_ENCONTRADA	2182	X'00000886'
MQRC_API_EXIT_LOAD_ERROR	2183	X'00000887'
MQRC_REMOTO_Q_NOMBRE_ERROR	2184	X'00000888'
MQRC_INCONSISTENT_PERSISTENCE	2185	X'00000889'
MQRC_GMO_ERROR	2186	X'0000088A'
MQRC_CICS_BRIDGE_RESTRICCIÓN	2187	X'0000088B'
MQRC_DETENIDO_POR_SALIDA_DEL_CLÚSTER	2188	X'0000088C'
MQRC_CLUSTER_RESOLUTION_ERROR	2189	X'0000088D'
MQRC_CONVERTED_STRING_TOO_BIG	2190	X'0000088E'
MQRC_TMC_ERROR	2191	X'0000088F'
MQRC_PÁGINAS_COMPLETAS	2192	X'00000890'
MQRC_STORAGE_MEDIUM_FULL	2192	X'00000890'
MQRC_PAGESET_ERROR	2193	X'00000891'
MQRC_NOMBRE_NO_VÁLIDO_PARA_TIPO	2194	X'00000892'
MQRC_UNEXPECTED_ERROR	2195	X'00000893'
MQRC_DESCONOCIDO_XMIT_Q	2196	X'00000894'
MQRC_DESCONOCIDO_DEF_XMIT_Q	2197	X'00000895'
MQRC_DEF_XMIT_Q_TYPE_ERROR	2198	X'00000896'
MQRC_DEF_XMIT_Q_USO_ERROR	2199	X'00000897'
MQRC_MSG_MARCADO_EXPLORAR_COOPERACIÓN	2200	X'00000898'
MQRC_NOMBRE_EN_USO	2201	X'00000899'
MQRC_CONNECTION QUIESCING	2202	X'0000089A'
MQRC_CONNECTION_STOPPING	2203	X'0000089B'
MQRC_ADAPTER_NOT_AVAILABLE	2204	X'0000089C'
MQRC_MSG_ID_ERROR	2206	X'0000089E'
MQRC_CORREL_ID_ERROR	2207	X'0000089F'
ERROR_DEL_SISTEMA_DE_ARCHIVOS_MQRC	2208	X'000008A0'
MQRC_NO_MSG_LOCKED	2209	X'000008A1'
MQRC_SOAP_DOTNET_ERROR	2210	X'000008A2'
MQRC_SOAP_AXIS_ERROR	2211	X'000008A3'
MQRC_SOAP_URL_ERROR	2212	X'000008A4'
MQRC_ARCHIVO_NO_AUDITADO	2216	X'000008A8'

Tabla 301. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQRC_CONNECTION_NOT_AUTHORIZED	2217	X'000008A9'
MQRC_MSG_DEMASIADO_GRANDE_PARA_EL_CANAL	2218	X'000008AA'
MQRC_CALL_IN_PROGRESS	2219	X'000008AB'
MQRC_RMH_ERROR	2220	X'000008AC'
MQRC_Q_MGR_ACTIVO	2222	X'000008AE'
MQRC_Q_MGR_NOT_ACTIVE	2223	X'000008AF'
MQRC_Q_PROFUNDIDAD_ALTA	2224	X'000008B0'
MQRC_Q_PROFUNDIDAD_BAJA	2225	X'000008B1'
MQRC_Q_INTERVALO_DE_SERVICIO_ALTO	2226	X'000008B2'
MQRC_Q_INTERVALO_DE_SERVICIO_OK	2227	X'000008B3'
MQRC_RFH_ENCABEZADO_CAMPO_ERROR	2228	X'000008B4'
MQRC_RAS_PROPERTY_ERROR	2229	X'000008B5'
MQRC_UNIDAD_DE_TRABAJO_NO_INICIADA	2232	X'000008B8'
MQRC_CANAL_AUTOMÁTICO_DEF_OK	2233	X'000008B9'
ERROR_DEFINICIÓN_AUTOMÁTICA_CANAL_MQRC	2234	X'000008BA'
MQRC_CFH_ERROR	2235	X'000008BB'
MQRC_CFIL_ERROR	2236	X'000008BC'
MQRC_CFIN_ERROR	2237	X'000008BD'
MQRC_CFSL_ERROR	2238	X'000008BE'
MQRC_CFST_ERROR	2239	X'000008BF'
MQRC_INCOMPLETE_GROUP	2241	X'000008C1'
MQRC_INCOMPLETE_MSG	2242	X'000008C2'
MQRC_INCONSISTENT_CCSDS	2243	X'000008C3'
MQRC_INCONSISTENT_ENCODINGS	2244	X'000008C4'
MQRC_INCONSISTENT_UOW	2245	X'000008C5'
MQRC_INVALID_MSG_UNDER_CURSOR	2246	X'000008C6'
MQRC_MATCH_OPTIONS_ERROR	2247	X'000008C7'
MQRC_MDE_ERROR	2248	X'000008C8'
MQRC_MSG_FLAGS_ERROR	2249	X'000008C9'
MQRC_MSG_SEQ_NUMBER_ERROR	2250	X'000008CA'
ERROR_DESVIACIÓN_MQRC	2251	X'000008CB'
ERROR_DE_LONGITUD_ORIGINAL_MQRC	2252	X'000008CC'
MQRC_LONGITUD_DEL_SEGMENTO_CERO	2253	X'000008CD'
MQRC_UOW_NOT_AVAILABLE	2255	X'000008CF'
MQRC_WRONG_GMO_VERSION	2256	X'000008D0'
MQRC_WRONG_MD_VERSION	2257	X'000008D1'
MQRC_GROUP_ID_ERROR	2258	X'000008D2'
MQRC_INCONSISTENT_BROWSE	2259	X'000008D3'
MQRC_XQH_ERROR	2260	X'000008D4'

Tabla 301. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQRC_SRC_ENV_ERROR	2261	X'000008D5'
ERROR_NOMBRE_MQRC_SRC	2262	X'000008D6'
MQRC_DEST_ENV_ERROR	2263	X'000008D7'
ERROR_NOMBRE_DESTINO_MQRC	2264	X'000008D8'
MQRC_TM_ERROR	2265	X'000008D9'
ERROR DE SALIDA DEL CLÚSTER MQRC	2266	X'000008DA'
MQRC_CLUSTER_SALIDA_ERROR_CARGA	2267	X'000008DB'
MQRC_CLUSTER_PUT_INHIBITED	2268	X'000008DC'
ERROR_RECURSO_CLÚSTER_MQRC	2269	X'000008DD'
MQRC_NO_DESTINOS_DISPONIBLES	2270	X'000008DE'
MQRC_CONN_TAG_EN_USO	2271	X'000008DF'
MQRC_PARTIALLY_CONVERTED	2272	X'000008E0'
ERROR_DE_CONEXIÓN_MQRC	2273	X'000008E1'
MQRC_OPTION_ERROR_ENTORNO	2274	X'000008E2'
MQRC_CD_ERROR	2277	X'000008E5'
ERROR_DE_CONEXIÓN_DEL_CLIENTE_MQRC	2278	X'000008E6'
MQRC_CHANNEL_STOPPED_BY_USER	2279	X'000008E7'
MQRC_HCONFIG_ERROR	2280	X'000008E8'
ERROR_DE_FUNCIÓN_MQRC	2281	X'000008E9'
MQRC_CHANNEL_STARTED	2282	X'000008EA'
MQRC_CHANNEL_STOPPED	2283	X'000008EB'
MQRC_CHANNEL_CONV_ERROR	2284	X'000008EC'
MQRC_SERVICIO_NO_DISPONIBLE	2285	X'000008ED'
MQRC_INICIALIZACIÓN_FALLIDA	2286	X'000008EE'
MQRC_TERMINACIÓN_FALLIDA	2287	X'000008EF'
MQRC_NOMBRE_DE_COLA_DESCONOCIDO	2288	X'000008F0'
ERROR_DE_SERVICIO_MQRC	2289	X'000008F1'
MQRC_Q_YA_EXISTE	2290	X'000008F2'
MQRC_ID_USUARIO_NO_DISPONIBLE	2291	X'000008F3'
MQRC_ENTIDAD_DESCONOCIDA	2292	X'000008F4'
MQRC_ENTIDAD_DESCONOCIDA	2293	X'000008F5'
MQRC_OBJETO_DE_REFERENCIA_DESCONOCIDO	2294	X'000008F6'
MQRC_CHANNEL_ACTIVATED	2295	X'000008F7'
MQRC_CHANNEL_NOT_ACTIVATED	2296	X'000008F8'
MQRC_UOW_CANCELADO	2297	X'000008F9'
MQRC_FUNCTION_NOT_SUPPORTED	2298	X'000008FA'
MQRC_TIPO_DE_SELECTOR_ERROR	2299	X'000008FB'
MQRC_COMANDO_TIPO_ERROR	2300	X'000008FC'
MQRC_ERROR_MÚLTIPLES_INSTANCIAS	2301	X'000008FD'

Tabla 301. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQRC_SISTEMA_ELEMENTO_NO_MODIFICABLE	2302	X'000008FE'
MQRC_BAG_ERROR_DE_CONVERSIÓN	2303	X'000008FF'
MQRC_SELECTOR_FUERA_DE_RANGO	2304	X'00000900'
MQRC_SELECTOR_NO_ÚNICO	2305	X'00000901'
MQRC_ÍNDICE_NO_PRESENTE	2306	X'00000902'
MQRC_STRING_ERROR	2307	X'00000903'
MQRC_CODIFICACIÓN_NO_ADMITIDA	2308	X'00000904'
MQRC_SELECTOR_NO_PRESENTE	2309	X'00000905'
MQRC_OUT_SELECTOR_ERROR	2310	X'00000906'
MQRC_STRING_TRUNCATED	2311	X'00000907'
MQRC_SELECTOR_TIPO_INCORRECTO	2312	X'00000908'
MQRC_TIPO_DE_ELEMENTO_INCONSISTENTE	2313	X'00000909'
MQRC_ERROR_ÍNDICE	2314	X'0000090A'
MQRC_SISTEMA_BOLSA_NO_MODIFICABLE	2315	X'0000090B'
ERROR_EN_EL_NÚMERO_DE_ARTÍCULOS_MQRC	2316	X'0000090C'
MQRC_FORMATO_NO_COMPATIBLE	2317	X'0000090D'
MQRC_SELECTOR_NO_COMPATIBLE	2318	X'0000090E'
MQRC_ERROR_DE_VALOR_DEL_ARTÍCULO	2319	X'0000090F'
MQRC_HBAG_ERROR	2320	X'00000910'
MQRC_PARÁMETRO_FALTANTE	2321	X'00000911'
MQRC_CMD_SERVIDOR_NO_DISPONIBLE	2322	X'00000912'
ERROR_DE_LONGITUD_DE_CADENA_MQRC	2323	X'00000913'
ERROR_DE_COMANDO_DE_CONSULTA_MQRC	2324	X'00000914'
MQRC_BOLSA_ANIDADA_NO_COMPATIBLE	2325	X'00000915'
MQRC_BAG_TIPO_INCORRECTO	2326	X'00000916'
MQRC_TIPO_DE_ERROR_DE_ELEMENTO	2327	X'00000917'
MQRC_SISTEMA_BOLSA_NO_ELIMINABLE	2328	X'00000918'
MQRC_SISTEMA_ELEMENTO_NO_ELIMINABLE	2329	X'00000919'
MQRC_CODIFICADO_CONJUNTO_DE_CARACTERES_ID_ERROR	2330	X'0000091A'
MQRC_MSG_TOKEN_ERROR	2331	X'0000091B'
MQRC_FALTA_WIH	2332	X'0000091C'
MQRC_WIH_ERROR	2333	X'0000091D'
MQRC_RFH_ERROR	2334	X'0000091E'
MQRC_RFH_STRING_ERROR	2335	X'0000091F'
ERROR_DE_COMANDO_MQRC_RFH	2336	X'00000920'
MQRC_RFH_PARM_ERROR	2337	X'00000921'
MQRC_RFH_PARÁMETRO_DUPLICADO	2338	X'00000922'
MQRC_RFH_PARM_FALTA	2339	X'00000923'
ERROR_DE_CONVERSIÓN_DE_CARACTERES_MQRC	2340	X'00000924'

Tabla 301. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQRC_UCS2_CONVERSION_ERROR	2341	X'00000925'
MQRC_DB2_NOT_AVAILABLE	2342	X'00000926'
MQRC_OBJETO_NO_ÚNICO	2343	X'00000927'
MQRC_CONN_TAG_NO_LIBERADO	2344	X'00000928'
MQRC_CF_NO_DISPONIBLE	2345	X'00000929'
MQRC_CF_STRUC_IN_USE	2346	X'0000092A'
MQRC_CF_STRUC_LIST_HDR_IN_USE	2347	X'0000092B'
MQRC_CF_STRUC_AUTH_FAILED	2348	X'0000092C'
MQRC_CF_STRUC_ERROR	2349	X'0000092D'
MQRC_CONN_TAG_NO_UTILIZABLE	2350	X'0000092E'
MQRC_GLOBAL_UOW_CONFLICT	2351	X'0000092F'
MQRC_LOCAL_UOW_CONFLICT	2352	X'00000930'
MQRC_HANDLE_IN_USE_FOR_UOW	2353	X'00000931'
MQRC_UOW_ENLISTMENT_ERROR	2354	X'00000932'
MQRC_UOW_MIX_NOT_SUPPORTED	2355	X'00000933'
MQRC_WXP_ERROR	2356	X'00000934'
MQRC_ERROR_REGISTRO_ACTUAL	2357	X'00000935'
MQRC_SIGUIENTE_DESPLAZAMIENTO_ERROR	2358	X'00000936'
MQRC_NO_HAY_REGISTROS_DISPONIBLES	2359	X'00000937'
MQRC_OBJETO_NIVEL_INCOMPATIBLE	2360	X'00000938'
MQRC_ERROR_SIGUIENTE_REGISTRO	2361	X'00000939'
MQRC_UMBRAL_DE_RETROCESO_ALCANZADO	2362	X'0000093A'
MQRC_MSG_NO_COINCIDE	2363	X'0000093B'
MQRC_JMS_FORMATO_ERROR	2364	X'0000093C'
MQRC_SEGMENTOS_NO_COMPATIBLES	2365	X'0000093D'
MQRC_NIVEL_CF_INCORRECTO	2366	X'0000093E'
MQRC_CONFIG_CREATE_OBJECT	2367	X'0000093F'
MQRC_CONFIG_CHANGE_OBJECT	2368	X'00000940'
MQRC_CONFIG_DELETE_OBJECT	2369	X'00000941'
MQRC_CONFIG_REFRESH_OBJECT	2370	X'00000942'
MQRC_CHANNEL_SSL_ERROR	2371	X'00000943'
MQRC_PARTICIPANTE_NO_DEFINIDO	2372	X'00000944'
MQRC_CF_STRUC_FAILED	2373	X'00000945'
MQRC_API_EXIT_ERROR	2374	X'00000946'
MQRC_API_SALIDA_ERROR_INICIAL	2375	X'00000947'
MQRC_API_EXIT_TERM_ERROR	2376	X'00000948'
MQRC_MOTIVO_DE_SALIDA_ERROR	2377	X'00000949'
MQRC_VALOR_RESERVADO_ERROR	2378	X'0000094A'
MQRC_NO_HAY_DATOS_DISPONIBLES	2379	X'0000094B'

Tabla 301. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQRC_SCO_ERROR	2380	X'0000094C'
ERROR_DEL_REPOSITORIO_DE_CLAVES_MQRC	2381	X'0000094D'
ERROR_DE_HARDWARE_CRIPTOGRÁFICO_MQRC	2382	X'0000094E'
ERROR_EN_EL_NÚMERO_DE_REGISTROS_DE_INFORMACIÓN_DE_AUTENTICACIÓN_MQRC	2383	X'0000094F'
MQRC_AUTH_INFO_REC_ERROR	2384	X'00000950'
MQRC_ERROR_AIR	2385	X'00000951'
MQRC_AUTH_INFO_TYPE_ERROR	2386	X'00000952'
MQRC_AUTH_INFO_CONN_NAME_ERROR	2387	X'00000953'
ERROR_DE_NOMBRE_DE_USUARIO_LDAP_MQRC	2388	X'00000954'
MQRC_LDAP_NOMBRE_USUARIO_LONGITUD_ERROR	2389	X'00000955'
ERROR_CON_LA_CONTRASEÑA_MQRC_LDAP	2390	X'00000956'
MQRC_SSL_YA_INICIALIZADO	2391	X'00000957'
ERROR_DE_CONFIGURACIÓN_MQRC_SSL	2392	X'00000958'
MQRC_SSL_INITIALIZATION_ERROR	2393	X'00000959'
MQRC_Q_INDEX_TYPE_ERROR	2394	X'0000095A'
MQRC_CFBS_ERROR	2395	X'0000095B'
MQRC_SSL_NOT_ALLOWED	2396	X'0000095C'
MQRC_JSSE_ERROR	2397	X'0000095D'
MQRC_SSL_PEER_NAME_MISMATCH	2398	X'0000095E'
MQRC_SSL_PEER_NAME_ERROR	2399	X'0000095F'
MQRC_UNSUPPORTED_CIPHER_SUITE	2400	X'00000960'
MQRC_SSL_CERTIFICATE_REVOKED	2401	X'00000961'
MQRC_SSL_CERT_STORE_ERROR	2402	X'00000962'
MQRC_CLIENT_EXIT_LOAD_ERROR	2406	X'00000966'
ERROR DE SALIDA DEL CLIENTE MQRC	2407	X'00000967'
MQRC_UOW_COMPROMETIDO	2408	X'00000968'
MQRC_SSL_KEY_RESET_ERROR	2409	X'00000969'
MQRC_NOMBRE_COMPONENTE_DESCONOCIDO	2410	X'0000096A'
MQRC_REGISTRADOR_ESTADO	2411	X'0000096B'
MQRC_COMANDO_MQSC	2412	X'0000096C'
MQRC_COMANDO_PCF	2413	X'0000096D'
MQRC_CFIF_ERROR	2414	X'0000096E'
MQRC_CFSF_ERROR	2415	X'0000096F'
MQRC_CFGR_ERROR	2416	X'00000970'
MQRC_MSG_NO_PERMITIDO_EN_GRUPO	2417	X'00000971'
ERROR_DEL_OPERADOR_DE_FILTRADO_MQRC	2418	X'00000972'
MQRC_ERROR_DE_SELECTOR_ANIDADO	2419	X'00000973'
MQRC_EPH_ERROR	2420	X'00000974'
MQRC_RFH_FORMATO_ERROR	2421	X'00000975'

Tabla 301. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQRC_CFBF_ERROR	2422	X'00000976'
MQRC_CONFLICTO_DE_CANAL_DEL_CLIENTE	2423	X'00000977'
MQRC_SD_ERROR	2424	X'00000978'
MQRC_TOPIC_STRING_ERROR	2425	X'00000979'
MQRC_STS_ERROR	2426	X'0000097A'
MQRC_NO_SUBSCRIPTION	2428	X'0000097C'
MQRC_SUSCRIPCIÓN_EN_USO	2429	X'0000097D'
MQRC_STAT_TYPE_ERROR	2430	X'0000097E'
MQRC_SUB_USER_DATA_ERROR	2431	X'0000097F'
MQRC_SUB_ALREADY_EXISTS	2432	X'00000980'
MQRC_IDENTITY_MISMATCH	2434	X'00000982'
MQRC_ALTER_SUB_ERROR	2435	X'00000983'
MQRC_DURABILITY_NOT_ALLOWED	2436	X'00000984'
MQRC_NO_MENSAJE_RETENIDO	2437	X'00000985'
MQRC_SRO_ERROR	2438	X'00000986'
MQRC_SUB_NAME_ERROR	2440	X'00000988'
MQRC_OBJECT_STRING_ERROR	2441	X'00000989'
ERROR_NOMBRE_PROPIEDAD_MQRC	2442	X'0000098A'
MQRC_SEGMENTACIÓN_NO_PERMITIDA	2443	X'0000098B'
MQRC_CBD_ERROR	2444	X'0000098C'
MQRC_CTLO_ERROR	2445	X'0000098D'
MQRC_NO_LLAMADAS_ACTIVAS	2446	X'0000098E'
MQRC_CALLBACK_NO_REGISTRADO	2448	X'00000990'
MQRC_OPTIONS_CHANGED	2457	X'00000999'
MQRC_LEER_POR_ADELANTE_MENSAJES	2458	X'0000099A'
MQRC_SELECTOR_SYNTAX_ERROR	2459	X'0000099B'
MQRC_HMSG_ERROR	2460	X'0000099C'
MQRC_CMHO_ERROR	2461	X'0000099D'
MQRC_DMHO_ERROR	2462	X'0000099E'
MQRC_SMPO_ERROR	2463	X'0000099F'
MQRC_IMPO_ERROR	2464	X'000009A0'
MQRC_NOMBRE_DE_LA_PROPIEDAD_DEMASIADO_LARGO	2465	X'000009A1'
MQRC_PROP_VALOR_NO_CONVERTIDO	2466	X'000009A2'
MQRC_PROP_TYPE_NOT_SUPPORTED	2467	X'000009A3'
MQRC_VALOR_DE_PROPIEDAD_DEMASIADO_GRANDE	2469	X'000009A5'
MQRC_PROP_CONV_NO_COMPATIBLE	2470	X'000009A6'
MQRC_PROPIEDAD_NO_DISPONIBLE	2471	X'000009A7'
ERROR_FORMATO_NÚMERO_PROPIEDAD_MQRC	2472	X'000009A8'
MQRC_TIPO_DE_PROPIEDAD_ERROR	2473	X'000009A9'

Tabla 301. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQRC_PROPIEDADES_DEMASIADO_GRADES	2478	X'000009AE'
MQRC_PUT_NO_RETENIDO	2479	X'000009AF'
MQRC_ALIAS_TARGET_TYPE_CHANGED	2480	X'000009B0'
MQRC_DMPO_ERROR	2481	X'000009B1'
MQRC_PD_ERROR	2482	X'000009B2'
MQRC_CALLBACK_TYPE_ERROR	2483	X'000009B3'
MQRC_CBD OPCIONES_ERROR	2484	X'000009B4'
MQRC_MAX_LONGITUD_DEL_MENSAJE_ERROR	2485	X'000009B5'
MQRC_CALLBACK_ROUTINE_ERROR	2486	X'000009B6'
MQRC_CALLBACK_LINK_ERROR	2487	X'000009B7'
ERROR_DE_OPERACIÓN_MQRC	2488	X'000009B8'
MQRC_BMHO_ERROR	2489	X'000009B9'
MQRC_PROPIEDAD_NO_COMPATIBLE	2490	X'000009BA'
MQRC_PROP_NAME_NOT_CONVERTED	2492	X'000009BC'
MQRC_OBTENER_HABILITADO	2494	X'000009BE'
MQRC_MÓDULO_NO_ENCONTRADO	2495	X'000009BF'
MQRC_MÓDULO_INVÁLIDO	2496	X'000009C0'
MQRC_MODULE_ENTRY_NOT_FOUND	2497	X'000009C1'
MQRC_CONTENIDO_MIXTO_NO_PERMITIDO	2498	X'000009C2'
MQRC_MSG_HANDLE_EN_USO	2499	X'000009C3'
MQRC_HCONN_ASYNC_ACTIVE	2500	X'000009C4'
MQRC_MHBO_ERROR	2501	X'000009C5'
MQRC_FALLO_PUBLICACIÓN	2502	X'000009C6'
MQRC_SUB_INHIBIDO	2503	X'000009C7'
MQRC_SELECTOR_SIEMPRE_FALSO	2504	X'000009C8'
MQRC_XEPO_ERROR	2507	X'000009CB'
MQRC_DURABILIDAD_NO_MODIFICABLE	2509	X'000009CD'
MQRC_TEMA_NO_MODIFICABLE	2510	X'000009CE'
MQRC_SUBNIVEL_NO_MODIFICABLE	2512	X'000009D0'
MQRC_PROPERTY_NAME_LENGTH_ERR	2513	X'000009D1'
MQRC_DUPLICADO_GRUPO_SUB	2514	X'000009D2'
MQRC_GRUPO_NO_MODIFICABLE	2515	X'000009D3'
MQRC_SELECTOR_INVÁLIDO_PARA_TIPO	2516	X'000009D4'
MQRC_HOBJ_EN_REPOSO	2517	X'000009D5'
MQRC_HOBJ QUIESCED SIN MENSAJES	2518	X'000009D6'
MQRC_SELECTION_STRING_ERROR	2519	X'000009D7'
ERROR_DE_CADENA_DE_OBJETO_MQRC	2520	X'000009D8'
MQRC_CONEXIÓN_SUSPENDIDA	2521	X'000009D9'
MQRC_DESTINO_INVÁLIDO	2522	X'000009DA'

Tabla 301. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQRC_SUSCRIPCIÓN_NO_VÁLIDA	2523	X'000009DB'
MQRC_SELECTOR_NO_MODIFICABLE	2524	X'000009DC'
MQRC_RETAINED_MSG_Q_ERROR	2525	X'000009DD'
MQRC_RETAINED_NOT_DELIVERED	2526	X'000009DE'
MQRC_RFH_FORMATO_RESTRINGIDO_ERROR	2527	X'000009DF'
MQRC_CONEXIÓN_DETENIDA	2528	X'000009E0'
MQRC_ASYNC_UOW_CONFLICT	2529	X'000009E1'
MQRC_ASYNC_XA_CONFLICT	2530	X'000009E2'
MQRC_PUBSUB_INHIBIDO	2531	X'000009E3'
MQRC_MSG_HANDLE_COPY_FAILURE	2532	X'000009E4'
MQRC_DEST_CLASS_NOT_ALTERABLE	2533	X'000009E5'
MQRC_OPERACIÓN_NO_PERMITIDA	2534	X'000009E6'
MQRC_ACTION_ERROR	2535	X'000009E7'
MQRC_CHANNEL_NOT_AVAILABLE	2537	X'000009E9'
MQRC_HOST_NOT_AVAILABLE	2538	X'000009EA'
ERROR_DE_CONFIGURACIÓN_DEL_CANAL_MQRC	2539	X'000009EB'
MQRC_NOMBRE_DE_CANAL_DESCONOCIDO	2540	X'000009EC'
MQRC_PUBLICACIÓN_EN_BUCLE	2541	X'000009ED'
MQRC_YA_UNIDO	2542	X'000009EE'
MQRC_ESPERA_C_GESTOR	2543	X'000009EF'
MQRC_RECONNECTING	2544	X'000009F0'
MQRC_RECONNECTED	2545	X'000009F1'
MQRC_RECONNECT_QMID_MISMATCH	2546	X'000009F2'
MQRC_RECONNECT_INCOMPATIBLE	2547	X'000009F3'
MQRC_RECONNECT_FAILED	2548	X'000009F4'
MQRC_CALL_INTERRUPTED	2549	X'000009F5'
MQRC_NO_SE_HAN_ENCONTRADO_SUSCRIPCIONES	2550	X'000009F6'
MQRC_SELECTION_NOT_AVAILABLE	2551	X'000009F7'
MQRC_CANAL_SSL_ADVERTENCIA	2552	X'000009F8'
MQRC_OCSP_URL_ERROR	2553	X'000009F9'
ERROR_DE CONTENIDO_MQRC	2554	X'000009FA'
MQRC_RECONNECT_Q_MGR_REQD	2555	X'000009FB'
MQRC_RECONEXIÓN_TIEMPO_LIMITE	2556	X'000009FC'
MQRC_PUBLISH_EXIT_ERROR	2557	X'000009FD'
MQRC_COMMINFO_ERROR	2558	X'000009FE'
MQRC_DEF_PUNTO_DE_SINCRONIZACIÓN_INHIBIDO	2559	X'000009FF'
MQRC_SOLO_MULTIDIFUSIÓN	2560	X'00000A00'
MQRC_CONJUNTO_DE_DATOS_NO_DISPONIBLE	2561	X'00000A01'
MQRC_AGRUPAZIONE_NON_CONSENTITA	2562	X'00000A02'

Tabla 301. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
ERROR_DIRECCIÓN_GRUPO_MQRC	2563	X'00000A03'
ERROR_DE_CONFIGURACIÓN_DE_MULTIDIFUSIÓN_MQRC	2564	X'00000A04'
ERROR_DE_INTERFAZ_MULTIDIFUSIÓN_MQRC	2565	X'00000A05'
MQRC_MULTICAST_SEND_ERROR	2566	X'00000A06'
MQRC_MULTICAST_ERROR_INTERNO	2567	X'00000A07'
MQRC_CONEXIÓN_NO_DISPONIBLE	2568	X'00000A08'
MQRC_SYNCPOINT_NO_PERMITIDO	2569	X'00000A09'
MQRC_SSL_ALT_PROVIDER_REQUIRED	2570	X'00000A0A'
MQRC_MCAST_PUB_STATUS	2571	X'00000A0B'
MQRC_MCAST_SUB_STATUS	2572	X'00000A0C'
MQRC_PRECONN_EXIT_LOAD_ERROR	2573	X'00000A0D'
MQRC_PRECONN_EXIT_NOT_FOUND	2574	X'00000A0E'
MQRC_PRECONN_EXIT_ERROR	2575	X'00000A0F'
MQRC_CD_ARRAY_ERROR	2576	X'00000A10'
MQRC_CHANNEL_BLOCKED	2577	X'00000A11'
MQRC_CANAL_BLOQUEADO_ADVERTENCIA	2578	X'00000A12'
MQRC_CREAR_SUSCRIPCIÓN	2579	X'00000A13'
MQRC_SUSCRIPCIÓN_ELIMINAR	2580	X'00000A14'
MQRC_CAMBIO_DE_SUSCRIPCIÓN	2581	X'00000A15'
MQRC_SUSCRIPCIÓN_ACTUALIZACIÓN	2582	X'00000A16'
MQRC_INSTALACIÓN_INCOMPATIBILIDAD	2583	X'00000A17'
MQRC_NO_PRIVILEGIADO	2584	X'00000A18'
MQRC_PROPIEDADES_DESACTIVADAS	2586	X'00000A1A'
MQRC_HMSG_NO_DISPONIBLE	2587	X'00000A1B'
MQRC_EXIT_PROPS_NO_COMPATIBLES	2588	X'00000A1C'
MQRC_INSTALACIÓN_FALTA	2589	X'00000A1D'
MQRC_FASTPATH_NO_DISPONIBLE	2590	X'00000A1E'
MQRC_CIPHER_SPEC_NOT_SUITE_B	2591	X'00000A1F'
MQRC_SUITE_B_ERROR	2592	X'00000A20'
ERROR DE POLÍTICA DE VALIDACIÓN DE CERTIFICADO MQRC	2593	X'00000A21'
ERROR_DE_PROTECCIÓN_CON_CONTRASEÑA_MQRC	2594	X'00000A22'
MQRC_CSP_ERROR	2595	X'00000A23'
MQRC_CERT_LABEL_NOT_ALLOWED	2596	X'00000A24'
MQRC_ADMIN_TOPIC_STRING_ERROR	2598	X'00000A26'
MQRC_AMQP_NO_DISPONIBLE	2599	X'00000A27'
MQRC_CCDT_URL_ERROR	2600	X'00000A28'
MQRC_Q_MGR_RECONEXIÓN_SOLICITADA	2601	X'00000A29'
MQRC_BNO_ERROR	2602	X'00000A2A'
MQRC_OUTBOUND_SNI_NO_VÁLIDO	2603	X'00000A2B'

Tabla 301. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
ERROR_AL_ACCEDER_AL_ALMACEN_DE_CLAVES_HTTPS_DE_MQRC	2604	X'00000A2C'
MQRC_REOPEN_EXCL_ERROR_DE_ENTRADA	6100	X'000017D4'
MQRC_REABRIR_CONSULTAR_ERROR	6101	X'000017D5'
MQRC_REABRIR_CONTEXTO_GUARDADO_ERROR	6102	X'000017D6'
MQRC_REABRIR_TEMPORALMENTE_COLONA_ERROR	6103	X'000017D7'
MQRC_ATTRIBUTO_BLOQUEADO	6104	X'000017D8'
MQRC_CURSOR_NO_VÁLIDO	6105	X'000017D9'
ERROR DE CODIFICACIÓN MQRC	6106	X'000017DA'
MQRC_STRUC_ID_ERROR	6107	X'000017DB'
MQRC_PUNTERO_NULO	6108	X'000017DC'
MQRC_SIN_REFERENCIA_DE_CONEXIÓN	6109	X'000017DD'
MQRC_SIN_BÚFER	6110	X'000017DE'
ERROR_DE_LONGITUD_DE_DATOS_BINARIOS_MQRC	6111	X'000017DF'
MQRC_BUFFER_NO_AUTOMÁTICO	6112	X'000017E0'
MQRC_BUFFER_INSUFICIENTE	6113	X'000017E1'
MQRC_DATOS_INSUFICIENTES	6114	X'000017E2'
MQRC_DATOS_TRUNCADOS	6115	X'000017E3'
MQRC_LONGITUD_CERO	6116	X'000017E4'
MQRC_LONGITUD_NEGATIVA	6117	X'000017E5'
MQRC_DESVIACIÓN_NEGATIVA	6118	X'000017E6'
MQRC_FORMATO_INCONSISTENTE	6119	X'000017E7'
MQRC_ESTADO_OBJETO_INCONSISTENTE	6120	X'000017E8'
MQRC_CONTEXTO_OBJETO_NO_VÁLIDO	6121	X'000017E9'
MQRC_CONTEXTO_ERROR_ABRIR	6122	X'000017EA'
ERROR_DE_LONGITUD_DE_ESTRUCTURA_MQRC	6123	X'000017EB'
MQRC_NO_CONECTADO	6124	X'000017EC'
MQRC_NO_ABIERTO	6125	X'000017ED'
LISTA_DE_DISTRIBUCIÓN_MQRC_VACÍA	6126	X'000017EE'
MQRC_INCONSISTENTE_OPCIONES_ABERTAS	6127	X'000017EF'
MQRC VERSIÓN_INCORRECTA	6128	X'000017F0'
ERROR DE REFERENCIA MQRC	6129	X'000017F1'
MQRC_XR_NO_DISPONIBLE	6130	X'000017F2'
MQRC_SUB_JOIN_NOT_ALTERABLE	29440	X'00007200'

MQRCCF_* (Códigos de motivo del encabezado del formato de comando)

Para obtener más información sobre la respuesta del programador, consulte [los códigos de motivo PCF](#).

Tabla 302. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQRCCF_CFH_TIPO_ERROR	3001	X'00000BB9'

Tabla 302. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQRCCF_CFH_LONGITUD_ERROR	3002	X'00000BBA'
MQRCCF_CFH VERSIÓN_ERROR	3003	X'00000BBB'
MQRCCF_CFH_MSG_NÚMERO_DE_SECUENCIA_ERROR	3004	X'00000BBC'
MQRCCF_CFH_ERROR_DE_CONTROL	3005	X'00000BBD'
MQRCCF_CFH_PARM_CONTRADO_ERROR	3006	X'00000BBE'
MQRCCF_CFH_ERROR_DE_COMANDO	3007	X'00000BBF'
MQRCCF_COMANDO_FALLIDO	3008	X'00000BC0'
MQRCCF_CFIN_LONGITUD_ERROR	3009	X'00000BC1'
MQRCCF_CFST_ERROR_DE_LONGITUD	3010	X'00000BC2'
MQRCCF_CFST_LONGITUD_CADENA_ERROR	3011	X'00000BC3'
MQRCCF_ERROR_DE_VALOR_FORZADO	3012	X'00000BC4'
MQRCCF_TIPO_DE_ERROR_DE_ESTRUCTURA	3013	X'00000BC5'
MQRCCF_CFIN_PARM_ID_ERROR	3014	X'00000BC6'
MQRCCF_CFST_PARM_ID_ERROR	3015	X'00000BC7'
MQRCCF_MSG_LONGITUD_ERROR	3016	X'00000BC8'
MQRCCF_CFIN_PARÁMETRO_DUPLICADO	3017	X'00000BC9'
MQRCCF_CFST_PARÁMETRO_DUPLICADO	3018	X'00000BCA'
MQRCCF_PARM_CONTE_DEMASIADO_PEQUEÑO	3019	X'00000BCB'
MQRCCF_PARM_CONTE_DEMASIADO_GRANDE	3020	X'00000BCC'
MQRCCF_Q_YA_EN_CELDA	3021	X'00000BCD'
MQRCCF_Q_TIPO_ERROR	3022	X'00000BCE'
MQRCCF_MD_FORMATO_ERROR	3023	X'00000BCF'
MQRCCF_CFSL_ERROR_DE_LONGITUD	3024	X'00000BD0'
MQRCCF_SUSTITUIR_VALOR_ERROR	3025	X'00000BD1'
MQRCCF_CFIL_VALOR_DUPLICADO	3026	X'00000BD2'
MQRCCF_CFIL_CONTROL_ERROR	3027	X'00000BD3'
MQRCCF_CFIL_LONGITUD_ERROR	3028	X'00000BD4'
MQRCCF_QUIESCE_VALOR_ERROR	3029	X'00000BD5'
MQRCCF_MODO_VALOR_ERROR	3029	X'00000BD5'
MQRCCF_MSG_NÚMERO_DE_SECUENCIA_ERROR	3030	X'00000BD6'
MQRCCF_PING_DATA_COUNT_ERROR	3031	X'00000BD7'
MQRCCF_PING_DATA_COMPARA_ERROR	3032	X'00000BD8'
MQRCCF_CFSL_PARM_ID_ERROR	3033	X'00000BD9'
MQRCCF_TIPO_DE_CANAL_ERROR	3034	X'00000BDA'
MQRCCF_PARM_ERROR_SECUENCIA	3035	X'00000BDB'
MQRCCF_XMIT_PROTOCOLO_TIPO_ERROR	3036	X'00000BDC'
MQRCCF_TAMAÑO_LOTE_ERROR	3037	X'00000BDD'
MQRCCF_ERROR_DE_INTERRUPCIÓN_DE_DISC	3038	X'00000BDE'
MQRCCF_ERROR_REINTENTO_CORT	3039	X'00000BDF'

Tabla 302. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQRCCF_ERROR_TEMPORIZADOR_CORT	3040	X'00000BE0'
MQRCCF_ERROR_DE_REINTENTO_LARGO	3041	X'00000BE1'
MQRCCF_ERROR_TEMPORIZADOR_LARGO	3042	X'00000BE2'
MQRCCF_SEQ_NUMBER_WRAP_ERROR	3043	X'00000BE3'
MQRCCF_ERROR_LONGITUD_MÁXIMA_DEL_MENSAJE	3044	X'00000BE4'
MQRCCF_ERROR_DE_AUTORIZACIÓN_DE_INGRESO	3045	X'00000BE5'
MQRCCF_PURGE_VALOR_ERROR	3046	X'00000BE6'
MQRCCF_CFIL_PARM_ID_ERROR	3047	X'00000BE7'
MQRCCF_MSG_TRUNCADO	3048	X'00000BE8'
MQRCCF_CCSID_ERROR	3049	X'00000BE9'
MQRCCF_ERROR_DE_CODIFICACIÓN	3050	X'00000BEA'
MQRCCF_COLAS_VALOR_ERROR	3051	X'00000BEB'
MQRCCF_DATA_CONV_VALOR_ERROR	3052	X'00000BEC'
MQRCCF_INDOUBT_VALUE_ERROR	3053	X'00000BED'
MQRCCF_TIPO_DE_ERROR_DE_ESCAPE	3054	X'00000BEE'
MQRCCF_REPOS_VALOR_ERROR	3055	X'00000BEF'
MQRCCF_TABLA_DE_CANALES_ERROR	3062	X'00000BF6'
MQRCCF_MCA_TIPO_ERROR	3063	X'00000BF7'
MQRCCF_CHL_INST_TYPE_ERROR	3064	X'00000BF8'
MQRCCF_CHL_ESTADO_NO_ENCONTRADO	3065	X'00000BF9'
MQRCCF_CFSL_PARÁMETRO_DUPLICADO	3066	X'00000BFA'
MQRCCF_CFSL_ERROR_LONGITUD_TOTAL	3067	X'00000BFB'
MQRCCF_CFSL_CONTROL_DE_ERRORES	3068	X'00000BFC'
MQRCCF_CFSL_LONGITUD_CADENA_ERROR	3069	X'00000BFD'
MQRCCF_CORREDOR_ELIMINADO	3070	X'00000BFE'
MQRCCF_ERROR_DE_TRANSMISIÓN	3071	X'00000BFF'
MQRCCF_TEMA_ERROR	3072	X'00000C00'
MQRCCF_NO_REGISTRADO	3073	X'00000C01'
MQRCCF_Q_MGR_NAME_ERROR	3074	X'00000C02'
MQRCCF_CORRECCIÓN_INCORRECTA	3075	X'00000C03'
MQRCCF_Q_NOMBRE_ERROR	3076	X'00000C04'
MQRCCF_NO_MENSAJE_RETENIDO	3077	X'00000C05'
MQRCCF_IDENTIDAD_DUPLICADA	3078	X'00000C06'
MQRCCF_INCORRECTO_Q	3079	X'00000C07'
MQRCCF_CORREL_ID_ERROR	3080	X'00000C08'
MQRCCF_NO_AUTORIZADO	3081	X'00000C09'
MQRCCF_FLUJO_DESCONOCIDO	3082	X'00000C0A'
MQRCCF_REG_OPTIONS_ERROR	3083	X'00000C0B'
MQRCCF_PUB_OPTIONS_ERROR	3084	X'00000C0C'

Tabla 302. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQRCCF_CORREDOR_DESCONOCIDO	3085	X'00000C0D'
MQRCCF_Q_MGR_CCSID_ERROR	3086	X'00000C0E'
MQRCCF_DEL_OPTIONS_ERROR	3087	X'00000C0F'
MQRCCF_NOMBRE_DEL_CLÚSTER_CONFLICTO	3088	X'00000C10'
MQRCCF_REPOS_NAME_CONFLICT	3089	X'00000C11'
MQRCCF_CLUSTER_Q_USAGE_ERROR	3090	X'00000C12'
MQRCCF_ERROR_DE_VALOR_DE_ACCIÓN	3091	X'00000C13'
MQRCCF_COMMS_LIBRARY_ERROR	3092	X'00000C14'
MQRCCF_ERROR_DE_NOMBRE_NETBIOS	3093	X'00000C15'
MQRCCF_BROKER_COMMAND_FAILED	3094	X'00000C16'
MQRCCF_CFST_PARÁMETRO_CONFLICTIVO	3095	X'00000C17'
MQRCCF_RUTA_NO_VÁLIDA	3096	X'00000C18'
MQRCCF_PARM_ERROR_SINTAXIS	3097	X'00000C19'
MQRCCF_PWD_LONGITUD_ERROR	3098	X'00000C1A'
ERROR_FILTRO_MQRCCF	3150	X'00000C4E'
MQRCCF_USUARIO_INCORRECTO	3151	X'00000C4F'
MQRCCF_SUSCRIPCIÓN_DUPLICADA	3152	X'00000C50'
MQRCCF_ERROR_EN_EL_NOMBRE_DEL_SUB	3153	X'00000C51'
MQRCCF_SUB_IDENTIDAD_ERROR	3154	X'00000C52'
MQRCCF_SUSCRIPCIÓN_EN_USO	3155	X'00000C53'
MQRCCF_SUSCRIPCIÓN_BLOQUEADA	3156	X'00000C54'
MQRCCF_YA_SE_HA_UNIDO	3157	X'00000C55'
MQRCCF_OBJETO_EN_USO	3160	X'00000C58'
MQRCCF_NOMBRE_DE_ARCHIVO_DESCONOCIDO	3161	X'00000C59'
MQRCCF_ARCHIVO_NO_DISPONIBLE	3162	X'00000C5A'
MQRCCF_DISC_RETRY_ERROR	3163	X'00000C5B'
MQRCCF_ASIGNAR_REINTENTO_ERROR	3164	X'00000C5C'
MQRCCF_ASIGNACIÓN_ERROR_TEMPORIZADOR_LENTO	3165	X'00000C5D'
MQRCCF_ASIGNACIÓN_ERROR_TEMPORIZADOR_RÁPIDO	3166	X'00000C5E'
MQRCCF_NÚMERO_DE_PUERTA_ERROR	3167	X'00000C5F'
MQRCCF_CHL_SISTEMA_NO_ACTIVO	3168	X'00000C60'
MQRCCF_NOMBRE_ENTIDAD_FALTA	3169	X'00000C61'
MQRCCF_PERFIL_NOMBRE_ERROR	3170	X'00000C62'
MQRCCF_AUTH_VALUE_ERROR	3171	X'00000C63'
MQRCCF_AUTH_VALOR_FALTANTE	3172	X'00000C64'
MQRCCF_TIPO_OBJETO_FALTANTE	3173	X'00000C65'
MQRCCF_ERROR_ID_CONEXIÓN	3174	X'00000C66'
MQRCCF_LOG_TYPE_ERROR	3175	X'00000C67'
MQRCCF_PROGRAMA_NO_DISPONIBLE	3176	X'00000C68'

Tabla 302. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQRCCF_PROGRAMA_AUTENTICACIÓN_FALLIDA	3177	X'00000C69'
MQRCCF_NINGUNO_ENCONTRADO	3200	X'00000C80'
MQRCCF_SEGURIDAD_INTERRUPTOR_APAGADO	3201	X'00000C81'
MQRCCF_SEGURIDAD_ACTUALIZACIÓN_FALLIDA	3202	X'00000C82'
MQRCCF_PARM_CONFLICTO	3203	X'00000C83'
MQRCCF_COMANDO_INHIBIDO	3204	X'00000C84'
MQRCCF_OBJETO_SIENDO_ELIMINADO	3205	X'00000C85'
MQRCCF_CLASE_DE_ALMACENAMIENTO_EN_USO	3207	X'00000C87'
MQRCCF_NOMBRE_OBJETO_RESTRINGIDO	3208	X'00000C88'
MQRCCF_LÍMITE_DE_OBJETOS_EXCEDIDO	3209	X'00000C89'
MQRCCF_OBJETO_ABRIR_FORZAR	3210	X'00000C8A'
MQRCCF_DISPOSICIÓN_CONFLICTO	3211	X'00000C8B'
MQRCCF_Q_MGR_NO_EN_QSG	3212	X'00000C8C'
MQRCCF_ATTR_VALUE_FIXED	3213	X'00000C8D'
MQRCCF_LISTA_DE_NOMBRES_ERROR	3215	X'00000C8F'
MQRCCF_NO_INICIADOR_DE_CANAL	3217	X'00000C91'
MQRCCF_CANAL_INICIADOR_ERROR	3218	X'00000C92'
MQRCCF_NIVEL_DE_COMANDO_CONFLICTO	3222	X'00000C96'
MQRCCF_Q_ATTR_CONFLICTO	3223	X'00000C97'
MQRCCF_EVENTOS_DESACTIVADOS	3224	X'00000C98'
MQRCCF_COMANDO_ESCOPO_ERROR	3225	X'00000C99'
ERROR_DE_RESPUESTA_DE_COMANDO_MQRCCF	3226	X'00000C9A'
MQRCCF_FUNCIÓN_RESTRINGIDA	3227	X'00000C9B'
MQRCCF_PARM_FALTA	3228	X'00000C9C'
MQRCCF_PARM_VALOR_ERROR	3229	X'00000C9D'
MQRCCF_ERROR_DE_LONGITUD_DEL_COMANDO	3230	X'00000C9E'
MQRCCF_COMANDO_ORIGEN_ERROR	3231	X'00000C9F'
MQRCCF_CONFLICTO_DE_LISTENER	3232	X'00000CA0'
MQRCCF_LISTENER_INICIADO	3233	X'00000CA1'
MQRCCF_LISTENER_DETENIDO	3234	X'00000CA2'
ERROR_CANAL_MQRCCF	3235	X'00000CA3'
MQRCCF_CF_STRUC_ERROR	3236	X'00000CA4'
MQRCCF_ID_DE_USUARIO_DESCONOCIDO	3237	X'00000CA5'
MQRCCF_ERROR_INESPERADO	3238	X'00000CA6'
MQRCCF_SIN_SOCIO_XCF	3239	X'00000CA7'
MQRCCF_CFGR_PARM_ID_ERROR	3240	X'00000CA8'
MQRCCF_CFIF_LONGITUD_ERROR	3241	X'00000CA9'
MQRCCF_CFIF_ERROR_OPERADOR	3242	X'00000CAA'
MQRCCF_CFIF_PARM_ID_ERROR	3243	X'00000CAB'

Tabla 302. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQRCCF_CFSF_FILTRO_VALOR_LONGITUD_ERROR	3244	X'00000CAC'
MQRCCF_CFSF_ERROR_DE_LONGITUD	3245	X'00000CAD'
MQRCCF_CFSF_ERROR_DEL_OPERADOR	3246	X'00000CAE'
MQRCCF_CFSF_PARM_ID_ERROR	3247	X'00000CAF'
MQRCCF_DEMASIADOS_FILTROS	3248	X'00000CB0'
MQRCCF_LISTENER_EN_MARCHA	3249	X'00000CB1'
MQRCCF_LSTR_ESTADO_NO_ENCONTRADO	3250	X'00000CB2'
MQRCCF_SERVICIO_EN_MARCHA	3251	X'00000CB3'
MQRCCF_SERV_STATUS_NO_ENCONTRADO	3252	X'00000CB4'
MQRCCF_SERVICIO_DETENIDO	3253	X'00000CB5'
MQRCCF_CFBS_PARÁMETRO_DUPLICADO	3254	X'00000CB6'
MQRCCF_CFBS_ERROR_DE_LONGITUD	3255	X'00000CB7'
MQRCCF_CFBS_PARM_ID_ERROR	3256	X'00000CB8'
MQRCCF_CFBS_LONGITUD_CADENA_ERROR	3257	X'00000CB9'
MQRCCF_CFGR_LONGITUD_ERROR	3258	X'00000CBA'
MQRCCF_CFGR_PARM_ERROR_DE_CONTADO	3259	X'00000CBB'
MQRCCF_CONN_NO_DETENIDA	3260	X'00000CBC'
MQRCCF_SOLICITUD_DE_SERVICIO_PENDIENTE	3261	X'00000CBD'
MQRCCF_NO_INICIO_CMD	3262	X'00000CBE'
MQRCCF_NO_STOP_CMD	3263	X'00000CBF'
MQRCCF_CFBF_LONGITUD_ERROR	3264	X'00000CC0'
MQRCCF_CFBF_PARM_ID_ERROR	3265	X'00000CC1'
MQRCCF_CFBF_ERROR_DEL_OPERADOR	3266	X'00000CC2'
MQRCCF_CFBF_FILTRO_VALOR_LONG_ERROR	3267	X'00000CC3'
MQRCCF_LISTENER_AÚN_ACTIVO	3268	X'00000CC4'
MQRCCF_DEF_XMIT_Q_CLUS_ERROR	3269	X'00000CC5'
MQRCCF_TOPICSTR_YA_EXISTE	3300	X'00000CE4'
MQRCCF_COMPARTIR_CONVERSACIONES_ERROR	3301	X'00000CE5'
MQRCCF_COMPARTIR_CONVERSACIONES_TIPO	3302	X'00000CE6'
MQRCCF_CASO_DE_SEGURIDAD_CONFLICTO	3303	X'00000CE7'
MQRCCF_TIPO_DE_ERROR_DE_TEMA	3305	X'00000CE9'
MQRCCF_MÁXIMO_DE_INSTANCIAS_ERROR	3306	X'00000CEA'
MQRCCF_MÁXIMO_NÚMERO_DE_INSTALACIONES_POR_CLIENTE_ERROR	3307	X'00000CEB'
MQRCCF_TOPIC_STRING_NOT_FOUND	3308	X'00000CEC'
MQRCCF_PUNTO_DE_SUSCRIPCIÓN_ERROR	3309	X'00000CED'
MQRCCF_SUB_YA_EXISTE	3311	X'00000CEF'
MQRCCF_NOMBRE_DE_OBJETO_DESCONOCIDO	3312	X'00000CF0'
MQRCCF_REMOTO_Q_NOMBRE_ERROR	3313	X'00000CF1'
MQRCCF_DURABILIDAD_NO_PERMITIDA	3314	X'00000CF2'

Tabla 302. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQRCCF_HOBJ_ERROR	3315	X'00000CF3'
MQRCCF_ERROR_NOMBRE_DESTINO	3316	X'00000CF4'
MQRCCF_DESTINO_INVÁLIDO	3317	X'00000CF5'
MQRCCF_PUBSUB_INHIBIDO	3318	X'00000CF6'
MQRCCF_CHLAUTH_TIPO_ERROR	3326	X'00000CFE'
MQRCCF_CHLAUTH_ACTION_ERROR	3327	X'00000CFF'
MQRCCF_CHLAUTH_USERSRC_ERROR	3335	X'00000D07'
MQRCCF_TIPO_DE_AUTENTICACIÓN_INCORRECTO	3336	X'00000D08'
MQRCCF_CHLAUTH_YA_EXISTE	3337	X'00000D09'
MQRCCF_CHLAUTH_NO_ENCONTRADO	3338	X'00000D0A'
MQRCCF_ERROR_EN_LA_ACCIÓN_DE_AUTENTICACIÓN_CHLAUTH	3339	X'00000D0B'
MQRCCF_ERROR_AUTENTICACIÓN_USUARIO	3340	X'00000D0C'
MQRCCF_CHLAUTH_ADVERTENCIA_ERROR	3341	X'00000D0D'
MQRCCF_ERROR_EN_LA_COMBINACIÓN_DE_AUTENTICACIÓN_CHLAUTH	3342	X'00000D0E'
MQRCCF_CONFLICTO_DE_RANGO_DE_DIRECCIONES_IP	3343	X'00000D0F'
MQRCCF_CHLAUTH_MÁXIMO_SUPERADO	3344	X'00000D10'
MQRCCF_IPADDR_ERROR	3345	X'00000D11'
MQRCCF_IPADDR_RANGE_ERROR	3346	X'00000D12'
MQRCCF_PERFIL_NOMBRE_FALTA	3347	X'00000D13'
MQRCCF_CHLAUTH_CLNTUSER_ERROR	3348	X'00000D14'
MQRCCF_CHLAUTH_NAME_ERROR	3349	X'00000D15'
MQRCCF_SUITE_B_ERROR	3353	X'00000D19'
MQRCCF_PSCLUS_DISABLED_TOPDEF	3359	X'00000D1F'
MQRCCF_PSCLUS_TOPIC_EXISTS	3360	X'00000D20'
MQRCCF_PROTOCOLO_NO_VÁLIDO	3365	X'00000D25'
MQRCCF_ACCESO_BLOQUEADO	3382	X'00000D36'
 MQRCCF_ATTRIBUTO_VALOR_ERROR_QSG_QM	3390	X'00000D3E'
MQRCCF_OBJETO_YA_EXISTE	4001	X'00000FA1'
MQRCCF_OBJETO_TIPO_INCORRECTO	4002	X'00000FA2'
MQRCCF_OBJETO_INCORRECTO_TIPO	4003	X'00000FA3'
MQRCCF_OBJETO_ABIERTO	4004	X'00000FA4'
MQRCCF_ATTR_VALUE_ERROR	4005	X'00000FA5'
MQRCCF_DESCONOCIDO_Q_MGR	4006	X'00000FA6'
MQRCCF_Q_TIPO_INCORRECTO	4007	X'00000FA7'
MQRCCF_NOMBRE_OBJETO_ERROR	4008	X'00000FA8'
MQRCCF_ASIGNACIÓN_FALLIDA	4009	X'00000FA9'
MQRCCF_HOST_NO_DISPONIBLE	4010	X'00000FAA'
MQRCCF_ERROR_DE_CONFIGURACIÓN	4011	X'00000FAB'
MQRCCF_CONEXIÓN_RECHAZADA	4012	X'00000FAC'

Tabla 302. Valores de las constantes (continuación)

Nombre	Valor decimal	Valor hexadecimal
MQRCCF_ERROR_DE_ENTRADA	4013	X'00000FAD'
MQRCCF_ENVÍO_FALLIDO	4014	X'00000FAE'
MQRCCF_ERROR_DE_DATOS_RECIBIDOS	4015	X'00000FAF'
MQRCCF_RECEPCIÓN_FALLIDA	4016	X'00000FB0'
MQRCCF_CONEXIÓN_CERRADA	4017	X'00000FB1'
MQRCCF_SIN_ALMACENAMIENTO	4018	X'00000FB2'
MQRCCF_SIN_GESTOR_DE_COMUNICACIONES	4019	X'00000FB3'
MQRCCF_LISTENER_NO_INICIADO	4020	X'00000FB4'
MQRCCF_BIND_FAILED	4024	X'00000FB8'
MQRCCF_CANAL_DUDOSO	4025	X'00000FB9'
MQRCCF_MQCONN_FALLÓ	4026	X'00000FBA'
MQRCCF_MQOPEN_FAILED	4027	X'00000FBB'
MQRCCF_MQGET_FAILED	4028	X'00000FBC'
MQRCCF_MQPUT_FAILED	4029	X'00000FBD'
MQRCCF_PING_ERROR	4030	X'00000FBE'
MQRCCF_CANAL_EN_USO	4031	X'00000FBF'
MQRCCF_CANAL_NO_ENCONTRADO	4032	X'00000FC0'
MQRCCF_CANAL_REMOTO_DESCONOCIDO	4033	X'00000FC1'
MQRCCF_REMOTO_QM_NO_DISPONIBLE	4034	X'00000FC2'
MQRCCF_REMOTO_QM_TERMINACIÓN	4035	X'00000FC3'
MQRCCF_MQINQ_FALLIDO	4036	X'00000FC4'
MQRCCF_NO_ENVIAR_Q	4037	X'00000FC5'
MQRCCF_CANAL_DESACTIVADO	4038	X'00000FC6'
MQRCCF_USUARIO_SALIDA_NO_DISPONIBLE	4039	X'00000FC7'
MQRCCF_COMMIT_FAILED	4040	X'00000FC8'
MQRCCF_TIPO_DE_CANAL_INCORRECTO	4041	X'00000FC9'
MQRCCF_CANAL_YA_EXISTE	4042	X'00000FCA'
MQRCCF_DATOS_DEMASIADO_GRANDES	4043	X'00000FCB'
MQRCCF_NOMBRE_CANAL_ERROR	4044	X'00000FCC'
MQRCCF_XMIT_Q_NAME_ERROR	4045	X'00000FCD'
MQRCCF_MCA_NOMBRE_ERROR	4047	X'00000FCF'
MQRCCF_ENVIAR_ERROR_NOMBRE_SALIDA	4048	X'00000FD0'
MQRCCF_SEC_EXIT_NAME_ERROR	4049	X'00000FD1'
MQRCCF_MSG_SALIDA_ERROR_NOMBRE	4050	X'00000FD2'
MQRCCF_RCV_EXIT_NAME_ERROR	4051	X'00000FD3'
MQRCCF_XMIT_Q_NAME_TIPO_INCORRECTO	4052	X'00000FD4'
MQRCCF_MCA_NOMBRE_TIPO_INCORRECTO	4053	X'00000FD5'
MQRCCF_DISC_INT_TIPO_INCORRECTO	4054	X'00000FD6'
MQRCCF_INTENTO_CORTOS_TIPO_INCORRECTO	4055	X'00000FD7'

Tabla 302. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQRCCF_TEMPORIZADOR_CORTOS_TIPO_INCORRECTO	4056	X'00000FD8'
MQRCCF_LONG_RETRY_TIPO_INCORRECTO	4057	X'00000FD9'
MQRCCF_LONG_TIMER_TIPO_INCORRECTO	4058	X'00000FDA'
MQRCCF_PUT_AUTH_TIPO_INCORRECTO	4059	X'00000FDB'
MQRCCF_MANTENER_ACTIVADO_ERROR_INTERNO	4060	X'00000FDC'
MQRCCF_NOMBRE_CONEXIÓN_FALTANTE	4061	X'00000FDD'
MQRCCF_CONN_NAME_ERROR	4062	X'00000FDE'
MQRCCF_MQSET_FALLÓ	4063	X'00000FDF'
MQRCCF_CANAL_NO_ACTIVADO	4064	X'00000FE0'
MQRCCF_TERMINADO_POR_SALIDA_SEC	4065	X'00000FE1'
MQRCCF_DYNAMIC_Q_SCOPE_ERROR	4067	X'00000FE3'
MQRCCF_CELL_DIR_NO_DISPONIBLE	4068	X'00000FE4'
MQRCCF_MR_CONTROL_ERROR	4069	X'00000FE5'
MQRCCF_MR_CONTAR_TIPO_INCORRECTO	4070	X'00000FE6'
MQRCCF_MR_SALIDA_ERROR_NOMBRE	4071	X'00000FE7'
MQRCCF_MR_EXIT_NAME_TIPO_INCORRECTO	4072	X'00000FE8'
MQRCCF_MR_INTERVALO_ERROR	4073	X'00000FE9'
MQRCCF_MR_INTERVALO_TIPO_INCORRECTO	4074	X'00000FEA'
MQRCCF_NPM_ERROR_VELOCIDAD	4075	X'00000FEB'
MQRCCF_NPM_VELOCIDAD_TIPO_INCORRECTO	4076	X'00000FEC'
MQRCCF_HB_INTERVALO_ERROR	4077	X'00000FED'
MQRCCF_HB_INTERVALO_TIPO_INCORRECTO	4078	X'00000FEE'
MQRCCF_CHAD_ERROR	4079	X'00000FEF'
MQRCCF_CHAD_TIPO_INCORRECTO	4080	X'00000FF0'
MQRCCF_CHAD_EVENTO_ERROR	4081	X'00000FF1'
MQRCCF_CHAD_EVENTO_TIPO_INCORRECTO	4082	X'00000FF2'
MQRCCF_CHAD_ERROR_DE_SALIDA	4083	X'00000FF3'
MQRCCF_CHAD_SALIDA_TIPO_INCORRECTO	4084	X'00000FF4'
MQRCCF_SUPRIMIDO_POR_SALIDA	4085	X'00000FF5'
MQRCCF_LOTE_ERROR_INT	4086	X'00000FF6'
MQRCCF_LOTE_INT_TIPO_INCORRECTO	4087	X'00000FF7'
MQRCCF_NET_PRIORITY_ERROR	4088	X'00000FF8'
MQRCCF_NET_PRIORIDAD_TIPO_INCORRECTO	4089	X'00000FF9'
MQRCCF_CANAL_CERRADO	4090	X'00000FFA'
MQRCCF_Q_ESTADO_NO_ENCONTRADO	4091	X'00000FFB'
MQRCCF_SSL_CIPHER_SPEC_ERROR	4092	X'00000FFC'
MQRCCF_SSL_NOMBRE_DEL_PAR_ERROR	4093	X'00000FFD'
MQRCCF_SSL_CLIENT_AUTH_ERROR	4094	X'00000FFE'
MQRCCF_RETENIDO_NO_COMPATIBLE	4095	X'00000FFF'

Tabla 302. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
 MQRCCE_KWD_VALOR_TIPO_INCORRECTO	4096	X'00001000'

MQRCN_* (Constantes de reconexión del cliente)

Tabla 303. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQRCN_NO	0	X'00000000'
MQRCN_YES	1	X'00000001'
MQRCN_Q_MGR	2	X'00000002'
MQRCN_DISABLED	3	X'00000003'

MQRCVTIME_* (Tipos de tiempo de espera de recepción)

Tabla 304. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQRCVTIME_MULTIPLICAR	0	X'00000000'
MQRCVTIME_ADD	1	X'00000001'
MQRCVTIME_EQUAL	2	X'00000002'

MQREADA_* (Valores de lectura anticipada)

Tabla 305. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQREADA_NO	0	X'00000000'
MQREADA_SÍ	1	X'00000001'
MQREADA_DESACTIVADO	2	X'00000002'
MQREADA_INHIBIDO	3	X'00000003'
MQREADA_ATASCO	4	X'00000004'

MQRECORDING_* (Opciones de grabación)

Tabla 306. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQRECORDING_DESACTIVADO	0	X'00000000'
MQRECORDING_Q	1	X'00000001'
MQRECORDING_MSG	2	X'00000002'

MQREGO_* (Opciones de registro de publicación/suscripción)

Tabla 307. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQREGO_NINGUNO	0	X'00000000'
MQREGO_CORREL_ID_COMO_IDENTIDAD	1	X'00000001'
MQREGO_ANÓNIMO	2	X'00000002'

Tabla 307. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQREGO_LOCAL	4	X'00000004'
MQREGO_SOLICITUDES_DIRECTAS	8	X'00000008'
MQREGO_NUEVAS_PUBLICACIONES_SOLO	16	X'00000010'
MQREGO_PUBLICAR_SOLO_BAJO_SOLICITUD	32	X'00000020'
MQREGO_DEREGISTRAR_TODOS	64	X'00000040'
MQREGO_INCLUIR_NOMBRE_DE_FLUJO	128	X'00000080'
MQREGO_INFORMAR_SI_RETENIDO	256	X'00000100'
MQREGO_DUPLICADOS_OK	512	X'00000200'
MQREGO_NO_PERSISTENTE	1024	X'00000400'
MQREGO_PERSISTENTE	2048	X'00000800'
MQREGO_PERSISTENTE_COMO_PUBLICACIÓN	4096	X'00001000'
MQREGO_PERSISTENTE_COMO_Q	8192	X'00002000'
MQREGO_AÑADIR_NOMBRE	16384	X'00004000'
MQREGO_SIN_MODIFICACIÓN	32768	X'00008000'
MQREGO_RESPUESTA_COMPLETA	65536	X'00010000'
MQREGO_UNIRSE_COMPARTIDO	131072	X'00020000'
MQREGO_JOIN_EXCLUSIVO	262144	X'00040000'
MQREGO_DEJAR_SOLO	524288	X'00080000'
MQREGO_ID_DE_USUARIO_VARIABLE	1048576	X'00100000'
MQREGO_BLOQUEADO	2097152	X'00200000'

MQRFH_* (Reglas y estructura del encabezado de formato y banderas)

Reglas y estructura del encabezado de formato

Tabla 308. Estructuras de constantes	
Nombre	Estructura
MQRFH_STRUC_ID	"RFH↵"
MQRFH_STRUC_ID_ARRAY	'R','F','H','↵'

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

Tabla 309. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQRFH_VERSION_1	1	X'00000001'
MQRFH_VERSION_2	2	X'00000002'
MQRFH_ESTRUCTURA_LONGITUD_FIJA	32	X'00000020'
MQRFH_STRUC_LENGTH_FIXED_2	36	X'00000024'

Reglas y formato del encabezado Banderas

Tabla 310. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQRFH_NINGUNO	0	X'00000000'
MQRFH_SIN_FLAGS	0	X'00000000'

MQRFH2_* (Etiquetas de opciones de publicación/suscripción RFH2 Etiquetas de carpetas de nivel superior)

Tabla 311. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQRFH2_NAME_VALUE_VERSION	1	X'00000001'

MQRFH2_* (Opciones de publicación/suscripción Etiquetas Nombres de etiquetas)

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

MQRFH2_* (Opciones de publicación/suscripción Nombres de etiquetas XML)

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

MQRL_* (Longitud devuelta)

Tabla 312. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQRL_INDETERMINADO	-1	X'FFFFFFFF'

MQRMH_* (Estructura del encabezado del mensaje de referencia)

Tabla 313. Estructuras de constantes	
Nombre	Estructura
MQRMH_STRUC_ID	"RMH↵"
MQRMH_STRUC_ID_ARRAY	'R','M','H','↵'

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

Tabla 314. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQRMH_VERSION_1	1	X'00000001'
MQRMH_VERSION_1	1	X'00000001'

MQRMHF_* (Indicadores del encabezado del mensaje de referencia)

Tabla 315. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQRMHF_ÚLTIMO	1	X'00000001'
MQRMHF_NO_ÚLTIMO	0	X'00000000'

MQRO_* (Opciones de informe)

Tabla 316. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQRO_EXCEPTION	16777216	X'01000000'
MQRO_EXCEPTION_WITH_DATA	50331648	X'03000000'
MQRO_EXCEPTION_WITH_FULL_DATA	117440512	X'07000000'
MQRO_EXPIRATION	2097152	X'00200000'
MQRO_EXPIRATION_WITH_DATA	6291456	X'00600000'
MQRO_EXPIRATION_WITH_FULL_DATA	14680064	X'00E00000'
MQRO_COA	256	X'00000100'
MQRO_COA_WITH_DATA	768	X'00000300'
MQRO_COA_WITH_FULL_DATA	1792	X'00000700'
MQRO_COD	2048	X'00000800'
MQRO_COD_WITH_DATA	6144	X'00001800'
MQRO_COD_WITH_FULL_DATA	14336	X'00003800'
MQRO_PAN	1	X'00000001'
MQRO_NAN	2	X'00000002'
MQRO_ACTIVITY	4	X'00000004'
MQRO_NEW_MSG_ID	0	X'00000000'
MQRO_PASS_MSG_ID	128	X'00000080'
MQRO_COPY_MSG_ID_TO_CORREL_ID	0	X'00000000'
MQRO_PASS_CORREL_ID	64	X'00000040'
MQRO_DEAD_LETTER_Q	0	X'00000000'
MQRO_DISCARD_MSG	134217728	X'08000000'
MQRO_PASS_DISCARD_AND_EXPIRY	16384	X'00004000'
MQRO_NONE	0	X'00000000'

MQRO_* (Máscaras de opciones de informe)

Tabla 317. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQRO_RECHAZAR_MÁSCARA_NO_SOPORTADA	270270464	X'101C0000'

Tabla 317. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQRO_ACCEPTAR_MÁSCARA_NO_SOPORTADA	-270532353	X'EFE000FF'
MQRO_ACCEPTAR_NO_SOPORTE_SI_MÁSCARA_DE_TRANSMISIÓN	261888	X'0003FF00'

MQROUTE_* (Ruta de seguimiento)

Actividades máximas de seguimiento de ruta (MQIACF_MAX_ACTIVITIES)

Tabla 318. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQROUTE_UNLIMITED_ACTIVITIES	0	X'00000000'

Detalle de la ruta de seguimiento (MQIACF_ROUTE_DETAIL)

Tabla 319. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQROUTE_DETAIL_LOW	2	X'00000002'
MQROUTE_DETAIL_MEDIUM	8	X'00000008'
MQROUTE_DETAIL_HIGH	32	X'00000020'

Reenvío de rutas de seguimiento (MQIACF_ROUTE_FORWARDING)

Tabla 320. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQROUTE_FORWARD_ALL	256	X'00000100'
MQROUTE_FORWARD_IF_SUPPORTED	512	X'00000200'
MQROUTE_REENVÍO_RECHAZADO_MÁSCARA_NO_SOPORTADA	-65536	X'FFFF0000'

Entrega con seguimiento de ruta (MQIACF_ROUTE_DELIVERY)

Tabla 321. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQROUTE_DELIVER_YES	4096	X'00001000'
MQROUTE_DELIVER_NO	8192	X'00002000'
MQROUTE_DELIVER_REJ_UNSUP_MASK	-65536	X'FFFF0000'

Acumulación de rutas de rastreo (MQIACF_ROUTE_ACCUMULATION)

Tabla 322. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQROUTE_ACCUMULATE_NONE	65539	X'00010003'
MQROUTE_ACCUMULATE_IN_MSG	65540	X'00010004'
MQROUTE_ACCUMULATE_AND_REPLY	65541	X'00010005'

MQRP_* (Opciones de sustitución del formato de comando)

Tabla 323. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQRP_SÍ	1	X'00000001'
MQRP_NO	0	X'00000000'

MQRQ_* (Calificadores de motivo del formato de comando)

Tabla 324. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQRQ_CONN_NO_AUTORIZADO	1	X'00000001'
MQRQ_OPEN_NO_AUTORIZADO	2	X'00000002'
MQRQ_CLOSE_NO_AUTORIZADO	3	X'00000003'
MQRQ_CMD_NO_AUTORIZADO	4	X'00000004'
MQRQ_Q_MGR_DETENCIÓN	5	X'00000005'
MQRQ_Q_MGR_EN ESTADO DE ESPERA	6	X'00000006'
MQRQ_CHANNEL_STOPPED_OK	7	X'00000007'
MQRQ_CHANNEL_STOPPED_ERROR	8	X'00000008'
MQRQ_CHANNEL_STOPPED_RETRY	9	X'00000009'
MQRQ_CHANNEL_STOPPED_DISABLED	10	X'0000000A'
MQRQ_PUENTE_DETENIDO_OK	11	X'0000000B'
MQRQ_PUENTE_DETENIDO_ERROR	12	X'0000000C'
MQRQ_SSL_HANDSHAKE_ERROR	13	X'0000000D'
MQRQ_SSL_CIFRADO_ESPECÍFICO_ERROR	14	X'0000000E'
MQRQ_SSL_CLIENT_AUTH_ERROR	15	X'0000000F'
MQRQ_SSL_PEER_NAME_ERROR	16	X'00000010'
MQRQ_SUB_NO_AUTORIZADO	17	X'00000011'
MQRQ_SUB_DEST_NO_AUTORIZADO	18	X'00000012'
MQRQ_SSL_REVOCACIÓN_DESCONOCIDA	19	X'00000013'
MQRQ_SYS_CONN_NO_AUTORIZADO	20	X'00000014'
MQRQ_CHANNEL_BLOCKED_ADDRESS	21	X'00000015'
MQRQ_CHANNEL_BLOCKED_USERID	22	X'00000016'
MQRQ_CHANNEL_BLOCKED_NOACCESS	23	X'00000017'
MQRQ_MAX_ACTIVE_CHANNELS	24	X'00000018'
MQRQ_MAX_CHANNELS	25	X'00000019'
MQRQ_SVRCONN_INST_LIMIT	26	X'0000001A'
¡LÍMITE DE INSTALACIÓN DEL CLIENTE MQRQ!	27	X'0000001B'
MQRQ_CAF_NO_INSTALADO	28	X'0000001C'
MQRQ_CSP_NO_AUTORIZADO	29	X'0000001D'
MQRQ_FAILOVER_PERMITIDO	30	X'0000001E'
MQRQ_FAILOVER_NO_PERMITIDO	31	X'0000001F'
MQRQ_EN ESPERA ACTIVADO	32	X'00000020'

Tabla 324. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQRQ_REPLICA_ACTIVADA	33	X'00000021'
 MQRQ_CONN_AUTORIZADO	65	X'00000041'
 MQRQ_ABIERTO_AUTORIZADO	66	X'00000042'
 MQRQ_SUB_AUTORIZADO	67	X'00000043'
 MQRQ_SUB_DEST_AUTORIZADO	68	X'00000044'

MQRT_* (Tipos de actualización del formato de comando)

Tabla 325. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQRT_CONFIGURACIÓN	1	X'00000001'
MQRT_EXPIRY	2	X'00000002'
MQRT_NSPROC	3	X'00000003'
MQRT_PROXYSUB	4	X'00000004'

MQRU_* (Solo solicitud)

Tabla 326. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQRU_PUBLICAR_A_PEDIDO	1	X'00000001'
MQRU_PUBLICAR_TODO	2	X'00000002'

MQSCA_* (Autenticación de cliente TLS)

Tabla 327. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQSCA_REQUISITO	0	X'00000000'
MQSCA_OPCIONAL	1	X'00000001'

MQSCO_* (opciones de configuración de TLS)

TLS Estructura de opciones de configuración

Tabla 328. Estructuras de constantes	
Nombre	Estructura
MQSCO_STRUC_ID	"SC0¬"
MQSCO_STRUC_ID_ARRAY	'S','C','0','¬'

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

Tabla 329. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQSCO_VERSION_1	1	X'00000001'
MQSCO_VERSION_2	2	X'00000002'
MQSCO_VERSION_3	3	X'00000003'

Tabla 329. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQSCO_VERSION_4	4	X'00000004'
MQSCO_VERSION_5	5	X'00000005'
MQSCO_VERSION_6	6	X'00000006'
MQSCO_VERSION_7	7	X'00000007'
MQSCO_VERSION_ACTUAL	7, LTS 6	X'00000007', LTS X'00000006'

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

TLS Opciones de configuración Restablecimiento de teclas Recuento

Tabla 330. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQSCO_RESET_COUNT_DEFAULT	0	X'00000000'

Formato de comando Definición de cola Ámbito

Tabla 331. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQSCO_Q_MGR	1	X'00000001'
MQSCO_CELL	2	X'00000002'

MQSCOPE_* (Ámbito de publicación)

Tabla 332. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQSCOPE_TODO	0	X'00000000'
MQSCOPE_COMO_PADRE	1	X'00000001'
MQSCOPE_QMGR	4	X'00000004'

MQSCYC_* (Caso de seguridad)

Tabla 333. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQSCYC_SUPERIOR	0	X'00000000'
MQSCYC_MIXED	1	X'00000001'

MQSD_* (Estructura descriptora de objetos)

Tabla 334. Nombres y estructuras constantes	
Nombre	Estructura
MQSD_STRUC_ID	"SD¬¬"
MQSD_STRUC_ID_ARRAY	'S','D','¬','¬'

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

Tabla 335. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQSD_VERSION_1	1	X'00000001'
MQSD_VERSION_ACTUAL	1	X'00000001'

MQSECITEM_* (Elementos de seguridad del formato de comando)

Tabla 336. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQSECITEM_TODOS	0	X'00000000'
MQSECITEM_MQADMIN	1	X'00000001'
MQSECITEM_MQNLIST	2	X'00000002'
MQSECITEM_MQPROC	3	X'00000003'
MQSECITEM_MQQUEUE	4	X'00000004'
MQSECITEM_MQCONN	5	X'00000005'
MQSECITEM_MQCMDS	6	X'00000006'
MQSECITEM_MXADMIN	7	X'00000007'
MQSECITEM_MXNLIST	8	X'00000008'
MQSECITEM_MXPROC	9	X'00000009'
MQSECITEM_MXQUEUE	10	X'0000000A'
MQSECITEM_MXTOPIC	11	X'0000000B'

MQSECPROT_* (Tipos de protocolo de seguridad)

Tabla 337. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQSECPROT_NONE	0	X'00000000'
MQSECPROT_SSLV30	1	X'00000001'
MQSECPROT_TLSV10	2	X'00000002'
MQSECPROT_TLSV12	4	X'00000004'

MQSECSW_* (Interruptores de seguridad del formato de comando y estados de los interruptores)

Formato de comando Interruptores de seguridad

Tabla 338. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQSECSW_PROCESO	1	X'00000001'
MQSECSW_LISTA_DE_NOMBRES	2	X'00000002'
MQSECSW_Q	3	X'00000003'
MQSECSW_TEMA	4	X'00000004'
MQSECSW_CONTEXTO	6	X'00000006'
MQSECSW_USUARIO_ALTERNATIVO	7	X'00000007'
MQSECSW_COMANDO	8	X'00000008'

Tabla 338. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQSECSW_CONEXIÓN	9	X'00000009'
MQSECSW_SUBSISTEMA	10	X'0000000A'
MQSECSW_COMMAND_RECursos	11	X'0000000B'
MQSECSW_Q_MGR	15	X'0000000F'
MQSECSW_QSG	16	X'00000010'

Formato de comando Estados del interruptor de seguridad

Tabla 339. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQSECSW_OFF_ENCONTRADO	21	X'00000015'
MQSECSW_ON_FOUND	22	X'00000016'
MQSECSW_OFF_NO_ENCONTRADO	23	X'00000017'
MQSECSW_ON_NOT_FOUND	24	X'00000018'
MQSECSW_DESACTIVADO_ERROR	25	X'00000019'
MQSECSW_EN SOBRESCRIPCIÓN	26	X'0000001A'

MQSECTYPE_* (Tipos de seguridad del formato de comando)

Tabla 340. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQSECTYPE_AUTHSERV	1	X'00000001'
MQSECTYPE_SSL	2	X'00000002'
MQSECTYPE_CLASES	3	X'00000003'

MQSEG_* (Segmentación)

Tabla 341. Nombres y valores constantes	
Nombre	Valor
MQSEG_INHIBIDO	'¬'
MQSEG_PERMITIDO	'A'

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

MQSEL_* (Valores selectores especiales)

Tabla 342. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQSEL_CUALQUIER_SELECTOR	-30001	X'FFFF8ACF'
MQSEL_CUALQUIER_USUARIO_SELECTOR	-30002	X'FFFF8ACE'
MQSEL_CUALQUIER_SELECTOR_DE_SISTEMA	-30003	X'FFFF8ACD'
MQSEL_ALL_SELECTORS	-30001	X'FFFF8ACF'
MQSEL_ALL_USER_SELECTORS	-30002	X'FFFF8ACE'
MQSEL_ALL_SYSTEM_SELECTORS	-30003	X'FFFF8ACD'

MQSELTYP* (Tipos de selector)

Tabla 343. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQSELTYP_NONE	0	X'00000000'
MQSELTYP_ESTÁNDAR	1	X'00000001'
MQSELTYP_EXTENDIDO	2	X'00000002'

MQSID* (Identificador de seguridad)

Tabla 344. Nombres y valores constantes	
Nombre	Valor
MQSID_NINGUNO	X'00...00' (40 nulos)
MQSID_NINGUNA_MATRIZ	'\0', '\0', ... (40 nulos)

MQSIDT* (Tipos de identificadores de seguridad)

Tabla 345. Nombres y valores constantes	
Nombre	Valor hexadecimal
MQSIDT_NINGUNO	X'00'
MQSIDT_NT_ID_DE_SEGURIDAD	X'01'
MQSIDT_WAS_ID_DE_SEGURIDAD	X'02'

MQSMPO* (Establecer opciones y estructura de propiedades de mensajes)

Establecer la estructura de opciones de propiedades del mensaje

Tabla 346. Estructuras de constantes	
Nombre	Estructura
MQSMPO_STRUC_ID	"SMPO"
MQSMPO_STRUC_ID_ARRAY	'S', 'M', 'P', 'O'

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

Tabla 347. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQSMPO_VERSION_1	1	X'00000001'
MQSMPO_VERSION_ACTUAL	1	X'00000001'

Configurar opciones de propiedades de mensajes

Tabla 348. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQSMPO_SET_FIRST	0	X'00000000'
MQSMPO_ESTABLECER_PROPIEDAD_BAJO_CURSOR	1	X'00000001'
MQSMPO_SET_PROP_AFTER_CURSOR	2	X'00000002'
MQSMPO_AÑADIR_PROPIEDAD	4	X'00000004'
MQSMPO_SET_PROP_BEFORE_CURSOR	8	X'00000008'

Tabla 348. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQSMPO_NINGUNO	0	X'00000000'

MQSO_* (Opciones de suscripción)

Tabla 349. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQSO_NINGUNO	0	X'00000000'
MQSO_NO_DURADERO	0	X'00000000'
MQSO_LEER_POR_ADELANTE_COMO_Q_DEF	0	X'00000000'
MQSO_ALTER	1	X'00000001'
MQSO_CREATE	2	X'00000002'
MQSO_RESUME	4	X'00000004'
MQSO_DURABLE	8	X'00000008'
MQSO_GRUPO_SUB	16	X'00000010'
MQSO_GESTIONADO	32	X'00000020'
MQSO_SET_IDENTITY_CONTEXT	64	X'00000040'
MQSO_FIXED_USERID	256	X'00000100'
MQSO_ANY_USERID	512	X'00000200'
MQSO_PUBLICACIONES_A_PEDIDO	2048	X'00000800'
MQSO_NUEVAS_PUBLICACIONES_SOLO	4096	X'00001000'
MQSO_FAIL_IF QUIESCING	8192	X'00002000'
MQSO_ALTERNATE_USER_AUTHORITY	262144	X'00040000'
MQSO_CARACTERE_COMODIN	1048576	X'00100000'
MQSO_WILDCARD_TEMA	2097152	X'00200000'
MQSO_SET_CORREL_ID	4194304	X'00400000'
MQSO_ÁMBITO_QMGR	67108864	X'04000000'
MQSO_NO_LEER_POR_ADELANTE	134217728	X'08000000'
MQSO_LEER_POR_ADELANTE	268435456	X'10000000'

MQSP_* (Disponibilidad del punto de sincronización)

Tabla 350. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQSP_AVAILABLE	1	X'00000001'
MQSP_NOT_AVAILABLE	0	X'00000000'

MQSPL_* (Opciones de protección de la política de seguridad)

Tabla 351. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQSPL_PASSTHRU	0	X'00000000'
MQSPL_REMOVE	1	X'00000001'
MQSPL_AS POLÍTICA	2	X'00000002'

MQSQM_* (Nombre del gestor de colas compartidas)

Tabla 352. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQSQM_USE	0	X'00000000'
MQSQM_IGNORAR	1	X'00000001'

MQSR_* (Acción)

Tabla 353. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQSR_ACCIÓN_PUBLICACIÓN	1	X'00000001'

MQSRO_* (Estructura de opciones de solicitud de suscripción)

Tabla 354. Estructuras de constantes	
Nombre	Estructura
MQSRO_STRUC_ID	"SRO¬"
MQSRO_STRUC_ID_ARRAY	'S', 'R', 'O', '¬'

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

Tabla 355. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQSRO_VERSION_1	1	X'00000001'
MQSRO_VERSIÓN_ACTUAL	1	X'00000001'
MQSRO_NINGUNO	0	X'00000000'
MQSRO_FALLO_SI_EN_REPOSO	8192	X'00002000'

MQSS_* (Estado del segmento)

Tabla 356. Nombres y estructuras constantes	
Nombre	Estructura
MQSS_NO_ES_UN_SEGMENTO	'¬'
MQSS_SEGMENTO	'S'
MQSS_ÚLTIMO_SEGMENTO	'L'

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

MQSSL_* (Requisitos FIPS de TLS)

Notas:

-  En AIX, Linux, y Windows, IBM MQ garantiza FIPS 140-3 el cumplimiento de la normativa a través del GSKit 9 módulo IBM Crypto for C (ICC) criptográfico. La certificación del NIST correspondiente al FIPS 140-3 módulo puede consultarse en <https://csrc.nist.gov/projects/cryptographic-module-validation-program/certificate/4755>.
-  FIPS Actualmente no hay soporte disponible para Linux s390x. Los clientes de esta plataforma que necesiten FIPS asistencia técnica deben seguir utilizando una versión anterior de IBM MQ. La compatibilidad con FIPS 140-3 se habilitará en una futura versión de FixPack una vez que el IBM Crypto for C (ICC) módulo criptográfico haya obtenido su certificación en esta plataforma.

- **V 10.0.0** El módulo FIPS 140-3 criptográfico integrado IBM Semeru Runtime fue aprobado por el NIST en agosto de 2024. IBM MQ 10.0.0 incorpora compatibilidad con la gestión de IBM MQ classes for JMS conexiones de IBM MQ classes for Java cliente mediante TLS para FIPS 140-3 en Java 8 y IBM Semeru Runtime 11+. La certificación del NIST correspondiente al FIPS 140-3 módulo se puede consultar en <https://csrc.nist.gov/projects/cryptographic-module-validation-program/certificate/4755>.
 - En IBM MQ el caso de Containers, las imágenes de IBM MQ Operator contenedor de 3.2.0 y del gestor de colas 9.4.0.0 en adelante se basan en UBI 9. FIPS 140-3 El cumplimiento de IBM MQ la normativa para productos envasados está actualmente pendiente.
- V 10.0.0** Si FIPS está habilitado, los procesos IBM MQ de control de contenedores utilizan un módulo FIPS 140-3 certificado de OpenSSL. Para obtener más información sobre la certificación del NIST, consulte: <https://access.redhat.com/compliance/fips>. IBM MQ Los gestores de colas que se ejecutan en imágenes de contenedor tienen el mismo nivel FIPS de certificación que la versión de la plataforma de IBM MQ la imagen base.

Tabla 357. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQSSL_FIPS_NO	0	X'00000000'
MQSSL_FIPS_YES	1	X'00000001'

MQSTAT_* (Opciones de estadísticas)

Tabla 358. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQSTAT_TYPE_ASYNC_ERROR	0	X'00000000'
MQSTAT_TYPE_RECONEXIÓN	0	X'00000000'
MQSTAT_TYPE_ERROR_RECONEXIÓN	0	X'00000000'

MQSTS_* (Estructura de informes de estado)

Tabla 359. Estructuras de constantes	
Nombre	Estructura
MQSTS_STRUC_ID	"STAT"
MQSTS_STRUC_ID_ARRAY	'S', 'T', 'A', 'T'

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

Tabla 360. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQSTS_VERSION_1	1	X'00000001'
MQSTS_VERSIÓN_ACTUAL	1	X'00000001'

MQSUB_* (Suscripciones duraderas)

Suscripciones duraderas permitidas

Tabla 361. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQSUB_DURABLE_COMO_PADRE	0	X'00000000'
MQSUB_DURABLE_PERMITIDO	1	X'00000001'
MQSUB_DURABLE_INHIBIDO	2	X'00000002'

Gama de suscripciones duraderas

Tabla 362. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQSUB_DURABLE_ALL	-1	X'FFFFFFFF'
MQSUB_DURABLE_SÍ	1	X'00000001'
MQSUB_DURABLE_NO	2	X'00000002'

MQSUBTYPE_* (Tipos de suscripción al formato de comando)

Tabla 363. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQSUBTYPE_API	1	X'00000001'
MQSUBTYPE_ADMIN	2	X'00000002'
MQSUBTYPE_PROXY	3	X'00000003'
MQSUBTYPE_TODOS	-1	X'FFFFFFFF'
MQSUBTYPE_USUARIO	-2	X'FFFFFFFE'

MQSUS_* (Formato de comando Estado de suspensión)

Tabla 364. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQSUS_SÍ	1	X'00000001'
MQSUS_NO	0	X'00000000'

MQSVC_* (Servicio)

Tipos de servicio

Tabla 365. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQSVC_TIPO_COMANDO	0	X'00000000'
MQSVC_TIPO_SERVIDOR	1	X'00000001'

Controles de servicio

Tabla 366. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQSVC_CONTROL_Q_MGR	0	X'00000000'
MQSVC_CONTROL_Q_MGR_INICIO	1	X'00000001'
MANUAL DE CONTROL DE MQSVC	2	X'00000002'

Estado del servicio

Tabla 367. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQSVC_STATUS_DETENIDO	0	X'00000000'

Tabla 367. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQSVC_STATUS_INICIANDO	1	X'00000001'
MQSVC_STATUS_EN_MARCHA	2	X'00000002'
MQSVC_STATUS_DETENIÉNDOSE	3	X'00000003'
MQSVC_STATUS_REINTENTANDO	4	X'00000004'

MQSYNCPPOINT_* (Valores de Syncpoint del formato de comando para la migración Pub/Sub)

Tabla 368. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQSYNCPPOINT_SÍ	0	X'00000000'
MQSYNCPPOINT_IFPER	1	X'00000001'

MQSYSP_* (Valores de los parámetros del sistema del formato de comando)

Tabla 369. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQSYSP_NO	0	X'00000000'
MQSYSP_SÍ	1	X'00000001'
MQSYSP_EXTENDIDO	2	X'00000002'
MQSYSP_TIPO_INICIAL	10	X'0000000A'
MQSYSP_TIPO_CONJUNTO	11	X'0000000B'
MQSYSP_TIPO_COPIA_REGISTRO	12	X'0000000C'
MQSYSP_TIPO_LOG_ESTADO	13	X'0000000D'
MQSYSP_TIPO_CINTA_ARCHIVO	14	X'0000000E'
MQSYSP_ASIGNAR_BLOQUE	20	X'00000014'
MQSYSP_ASIGNAR_TRK	21	X'00000015'
MQSYSP_ASIGNAR_CIL	22	X'00000016'
MQSYSP_ESTADO_OCUPADO	30	X'0000001E'
MQSYSP_STATUS_PREMOUNT	31	X'0000001F'
MQSYSP_ESTADO_DISPONIBLE	32	X'00000020'
MQSYSP_ESTADO_DESCONOCIDO	33	X'00000021'
MQSYSP_STATUS_ALLOC_ARCHIVE	34	X'00000022'
MQSYSP_ESTADO_COPIANDO_BSDS	35	X'00000023'
MQSYSP_ESTADO_COPIANDO_REGISTRO	36	X'00000024'

MQTA_* (Atributos del tema)

Comodines

Tabla 370. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQTA_BLOQUE	1	X'00000001'

Tabla 370. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQTA_PASSTHRU	2	X'00000002'

Suscripciones permitidas

Tabla 371. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQTA_SUB_COMO_PADRE	0	X'00000000'
MQTA_SUB_INHIBIDO	1	X'00000001'
MQTA_SUB_PERMITIDO	2	X'00000002'

Propagación de subproxies

Tabla 372. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQTA_PROXY_SUB_FORCE	1	X'00000001'
MQTA_PROXY_SUB_PRIMER_USO	2	X'00000002'

Publicaciones permitidas

Tabla 373. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQTA_PUB_COMO_PADRE	0	X'00000000'
MQTA_PUB_INHIBIDO	1	X'00000001'
MQTA_PUB_PERMITIDO	2	X'00000002'

MQTC_* (Controles de activación)

Tabla 374. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQTC_OFF	0	X'00000000'
MQTC_ON	1	X'00000001'

MQTCPKEEP_* (Mantenimiento de conexión TCP)

Tabla 375. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQTCPKEEP_NO	0	X'00000000'
MQTCPKEEP_SÍ	1	X'00000001'

MQTCPSTACK_* (tipos de pila TCP)

Tabla 376. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQTCPSTACK_SINGLE	0	X'00000000'
MQTCPSTACK_MÚLTIPLE	1	X'00000001'

MQTIME_* (Formato de comando Unidades de tiempo)

Tabla 377. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQTIME_UNIT_MINS	0	X'00000000'
MQTIME_UNIT_SECS	1	X'00000001'

MQTM_* (Estructura del mensaje de activación)

Tabla 378. Estructuras de constantes	
Nombre	Estructura
MQTM_STRUC_ID	"TM↵"
MQTM_STRUC_ID_ARRAY	'T','M','↵','↵'

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

Tabla 379. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQTM_VERSION_1	1	X'00000001'
MQTM_VERSION_2	1	X'00000001'

MQTMC_* (Estructura del formato de caracteres del mensaje de activación)

Tabla 380. Estructuras de constantes	
Nombre	Estructura
MQTMC_STRUC_ID	"TMC↵"
MQTMC_STRUC_ID_ARRAY	'T','M','C','↵'
MQTMC_VERSION_1	"↵↵1"
MQTMC_VERSION_2	"↵↵2"
MQTMC_VERSION_3	"↵↵2"
MQTMC_VERSION_1_ARRAY	'↵','↵','↵','1'
MQTMC_VERSION_2_ARRAY	'↵','↵','↵','2'
MQTMC_VERSION_3_ARRAY	'↵','↵','↵','2'

MQTOPT_* (Tipo de tema)

Tabla 381. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQTOPT_LOCAL	0	X'00000000'
MQTOPT_CLUSTER	1	X'00000001'
MQTOPT_TODOS	2	X'00000002'

MQTRAXSTR_* (Inicio automático del seguimiento del iniciador del canal)

Tabla 382. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQTRAXSTR_NO	0	X'00000000'
MQTRAXSTR_SÍ	1	X'00000001'

MQTSCOPE_* (Ámbito de suscripción)

Tabla 383. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQTSCOPE_QMGR	1	X'00000001'
MQTSCOPE_TODO	2	X'00000002'

MQTT_* (Tipos de activadores)

Tabla 384. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQTT_NONE	0	X'00000000'
MQTT_FIRST	1	X'00000001'
MQTT EVERY	2	X'00000002'
MQTT_DEPTH	3	X'00000003'

MQTYPE_* (Tipos de datos de propiedad)

Tabla 385. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQTYPE_AS_SET	0	X'00000000'
MQTYPE_NULL	2	X'00000002'
MQTYPE_BOOLEAN	4	X'00000004'
MQTYPE_BYTE_STRING	8	X'00000008'
MQTYPE_INT8	16	X'00000010'
MQTYPE_INT16	32	X'00000020'
MQTYPE_INT32	64	X'00000040'
MQTYPE_LONG	64	X'00000040'
MQTYPE_INT64	128	X'00000080'
MQTYPE_FLOAT32	256	X'00000100'
MQTYPE_FLOAT64	512	X'00000200'
MQTYPE_STRING	1024	X'00000400'

MQUA_* (Selektors de atributos de usuario para publicar/suscribirse)

Tabla 386. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQUA_PRIMERO	65536	X'00010000'
MQUA_ÚLTIMO	99999999	X'3B9AC9FF'

MQUIDSUPP_* (Compatibilidad con ID de usuario en formato de comando)

Tabla 387. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQUIDSUPP_NO	0	X'00000000'
MQUIDSUPP_SÍ	1	X'00000001'

MQUNDELIVERED_* (Valores no entregados del formato de comando para la migración Pub/Sub)

Tabla 388. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQUNDELIVERED_NORMAL	0	X'00000000'
ENTREGADO_SEGURO	1	X'00000001'
MQUNDELIVERED_DESCARTADO	2	X'00000002'
MQUNDELIVERED_KEEP	3	X'00000003'

MQUOWST_* (Formato de comando Estados UOW)

Tabla 389. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQUOWST_NONE	0	X'00000000'
MQUOWST_ACTIVO	1	X'00000001'
MQUOWST_PREPARADO	2	X'00000002'
MQUOWST_SIN Resolver	3	X'00000003'

MQUOWT_* (Tipos de UOW de formato de comando)

Tabla 390. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQUOWT_Q_MGR	0	X'00000000'
MQUOWT_CICS	1	X'00000001'
MQUOWT_RRS	2	X'00000002'
MQUOWT_IMS	3	X'00000003'
MQUOWT_XA	4	X'00000004'

MQUS_* (Usos de colas)

Tabla 391. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQUS_NORMAL	0	X'00000000'
MQUS_TRANSMISSION	1	X'00000001'

MQUSAGE_* (Valores de uso del conjunto de páginas y valores de uso del conjunto de datos en formato de comando)

Formato de comando Página Establecer valores de uso

Tabla 392. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQUSAGE_PS_DISPONIBLE	0	X'00000000'
MQUSAGE_PS_DEFINIDO	1	X'00000001'
MQUSAGE_PS_DESCONECTADO	2	X'00000002'
MQUSAGE_PS_NO_DEFINIDO	3	X'00000003'

Tabla 392. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQUSAGE_PS_SUSPENDIDO	4	X'00000004'
MQUSAGE_EXPAND_USER	1	X'00000001'
MQUSAGE_EXPAND_SYSTEM	2	X'00000002'
MQUSAGE_EXPAND_NONE	3	X'00000003'

Formato de comando Conjunto de datos Valores de uso

Tabla 393. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQUSAGE_DS_MÁS_ANTIGUO_ACTIVO_UOW	10	X'0000000A'
MENSAJE_DS_RECUPERACIÓN_PS_MÁS_ANTIGUA	11	X'0000000B'
MENSAJE_MQUS_DS_RECUPERACIÓN_CF_MÁS_ANTIGUA	12	X'0000000C'

MQVL_* (Longitud del valor)

Tabla 394. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQVL_TERMINADO_CON_NULL	-1	X'FFFFFFFF'
MQVL_CADENA_VACÍA	0	X'00000000'

MQVU_* (ID de usuario variable)

Tabla 395. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQVU_USUARIO_FIJO	1	X'00000001'
MQVU_CUALQUIER_USUARIO	2	X'00000002'

MQWDR_* (Estructura de registro de destino de salida de carga de trabajo del clúster)

Tabla 396. Estructuras de constantes	
Nombre	Estructura
MQWDR_STRUC_ID	"WDR¬"
MQWDR_STRUC_ID_ARRAY	'W','D','R','¬'

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

Tabla 397. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQWDR_VERSION_1	1	X'00000001'
MQWDR_VERSION_2	2	X'00000002'
MQWDR_VERSION ACTUAL	2	X'00000002'
MQWDR_LENGTH_1	124	X'0000007C'
MQWDR_LENGTH_2	136	X'00000088'
MQWDR_LONGITUD_ACTUAL	136	X'00000088'

MQWI_* (Intervalo de espera)

Tabla 398. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQWI_ILIMITADO	-1	X'FFFFFFFF'

MQWIH_* (Estructura del encabezado de información de carga de trabajo y indicadores)

Estructura del encabezado de la información sobre la carga de trabajo

Tabla 399. Estructuras de constantes	
Nombre	Estructura
MQWIH_STRUC_ID	"WIH¬"
MQWIH_STRUC_ID_ARRAY	'W','I','H','¬'

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

Tabla 400. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQWIH_VERSION_1	1	X'00000001'
MQWIH_VERSION_ACTUAL	1	X'00000001'
MQWIH_LENGTH_1	120	X'00000078'
MQWIH_LONGITUD_ACTUAL	120	X'00000078'

Cabecera de información sobre la carga de trabajo Indicadores

Tabla 401. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQWIH_NINGUNO	0	X'00000000'

MQWQR_* (Estructura de registro de cola de salida de carga de trabajo del clúster)

Tabla 402. Estructuras de constantes	
Nombre	Estructura
MQWQR_STRUC_ID	"WQR¬"
MQWQR_STRUC_ID_ARRAY	'W','Q','R','¬'

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

Tabla 403. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQWQR_VERSION_1	1	X'00000001'
MQWQR_VERSION_2	2	X'00000002'
MQWQR_VERSION_3	3	X'00000003'
MQWQR_VERSION_ACTUAL	3	X'00000003'
MQWQR_LENGTH_1	200	X'000000C8'

Tabla 403. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQWQR_LENGTH_2	208	X'000000D0'
MQWQR_LENGTH_3	212	X'000000D4'
MQWQR_LONGITUD_ACTUAL	212	X'000000D4'

MQWS_* (Esquema comodín)

Tabla 404. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQWS_DEFAULT	0	X'00000000'
MQWS_CHAR	1	X'00000001'
MQWS_TEMA	2	X'00000002'

MQWXP_* (Estructura de parámetros de salida de carga de trabajo del clúster)

MQWXP_* (Estructura de parámetros de salida de carga de trabajo del clúster)

Tabla 405. Estructuras de constantes	
Nombre	Estructura
MQWXP_STRUC_ID	"WXP¬"
MQWXP_STRUC_ID_ARRAY	'W','X','P','¬'

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

Tabla 406. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQWXP_VERSION_1	1	X'00000001'
MQWXP_VERSION_2	2	X'00000002'
MQWXP_VERSION_3	3	X'00000003'
MQWXP_VERSION_4	4	X'00000004'
MQWXP_VERSION_ACTUAL	4	X'00000004'

MQWXP_* (Indicadores de carga de trabajo del clúster)

Tabla 407. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQWXP_PUT_BY_CLUSTER_CHL	2	X'00000002'

Referencia relacionada

“Campos en MQWXP : estructura del parámetro de salida de la carga de trabajo del clúster” en la página 1569

Descripción de los campos en MQWXP : estructura del parámetro de salida de carga de trabajo del clúster

MQXACT_* (Tipos de llamadas a la API)

Tabla 408. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQXACT_EXTERNAL	1	X'00000001'
MQXACT_INTERNAL	2	X'00000002'

MQXC_* (Comandos de salida)

Tabla 409. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQXC_MQOPEN	1	X'00000001'
MQXC_MQCLOSE	2	X'00000002'
MQXC_MQGET	3	X'00000003'
MQXC_MQPUT	4	X'00000004'
MQXC_MQPUT1	5	X'00000005'
MQXC_MQINQ	6	X'00000006'
MQXC_MQSET	8	X'00000008'
MQXC_MQBACK	9	X'00000009'
MQXC_MQCMIT	10	X'0000000A'

MQXCC_* (Respuestas de salida)

Tabla 410. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQXCC_OK	0	X'00000000'
MQXCC_SUPPRIMIR_FUNCIÓN	-1	X'FFFFFFFF'
MQXCC_OMITIR_FUNCIÓN	-2	X'FFFFFFFE'
MQXCC_ENVIAR_Y_SOLICITAR_MENSAJE_SEC	-3	X'FFFFFFFD'
MQXCC_ENVIAR_MENSAJE_SEC	-4	X'FFFFFFFC'
MQXCC_SUPPRESS_EXIT	-5	X'FFFFFFFB'
MQXCC_CERRAR_CANAL	-6	X'FFFFFFFA'
MQXCC_SOLICITUD_ACUSE_DE_RECIBO	-7	X'FFFFFFF9'
MQXCC_FALLÓ	-8	X'FFFFFFF8'

MQXDR_* (Respuesta de salida)

Tabla 411. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQXDR_OK	0	X'00000000'
MQXDR_CONVERSIÓN_FALLIDA	1	X'00000001'

MQXE_* (Entornos)

Tabla 412. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQXE_OTHER	0	X'00000000'
MQXE_MCA	1	X'00000001'
MQXE_MCA_SVRCONN	2	X'00000002'
MQXE_COMMAND_SERVER	3	X'00000003'
MQXE_MQSC	4	X'00000004'

MQXEPO_* (Estructura de opciones de punto de entrada de registro y opciones de salida)

Estructura de opciones del punto de entrada del registro

Tabla 413. Estructuras de constantes	
Nombre	Estructura
MQXEPO_STRUC_ID	"XEPO"
MQXEPO_STRUC_ID_ARRAY	'X', 'E', 'P', 'O'

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

Tabla 414. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQXEPO_VERSION_1	1	X'00000001'
MQXEPO_VERSION_ACTUAL	1	X'00000001'

Opciones de salida

Tabla 415. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQXEPO_NONE	0	X'00000000'

MQXF_* (Identificadores de funciones API)

Tabla 416. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQXF_INIT	1	X'00000001'
MQXF_TERM	2	X'00000002'
MQXF_CONN	3	X'00000003'
MQXF_CONNX	4	X'00000004'
MQXF_DISCO	5	X'00000005'
MQXF_ABRIR	6	X'00000006'
MQXF_CERRAR	7	X'00000007'
MQXF_PUT1	8	X'00000008'
MQXF_PUT	9	X'00000009'
MQXF_OBTENER	10	X'0000000A'

Tabla 416. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQXF_DATA_CONV_ON_GET	11	X'0000000B'
MQXF_INQ	12	X'0000000C'
MQXF_SET	13	X'0000000D'
MQXF_BEGIN	14	X'0000000E'
MQXF_CMIT	15	X'0000000F'
MQXF_ATRÁS	16	X'00000010'
MQXF_STAT	18	X'00000012'
MQXF_CB	19	X'00000013'
MQXF_CTL	20	X'00000014'
MQXF_CALLBACK	21	X'00000015'
MQXF_SUB	22	X'00000016'
MQXF_SUBRQ	23	X'00000017'
MQXF_XACLOSE	24	X'00000018'
MQXF_XACOMMIT	25	X'00000019'
MQXF_XACOMplete	26	X'0000001A'
MQXF_XAEND	27	X'0000001B'
MQXF_XAFORGET	28	X'0000001C'
MQXF_XAOPEN	29	X'0000001D'
MQXF_XAPREPARE	30	X'0000001E'
MQXF_XARECOVER	31	X'0000001F'
MQXF_XAROLLBACK	32	X'00000020'
MQXF_XASTART	33	X'00000021'
MQXF_AXREG	34	X'00000022'
MQXF_AXUNREG	35	X'00000023'

MQXP_* (estructura de parámetros de salida de cruce de API)

Tabla 417. Estructuras de constantes	
Nombre	Estructura
MQXP_STRUC_ID	"XP¬¬"
MQXP_STRUC_ID_ARRAY	'X', 'P', '¬', '¬'

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

Tabla 418. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQXP_VERSION_1	1	X'00000001'

MQXPDA_* (Área de determinación de problemas)

Tabla 419. Nombres y valores constantes	
Nombre	Valor
MQXPDA_NINGUNO	X'00...00' (48 nulos)

Tabla 419. Nombres y valores constantes (continuación)	
Nombre	Valor
MQXPDA_NINGUNA_MATRIZ	'\0', '\0', ... (48 nulos)

MQXPT_* (Tipos de transporte)

Tabla 420. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQXPT_TODOS	-1	X'FFFFFFFF'
MQXPT_LOCAL	0	X'00000000'
MQXPT_LU62	1	X'00000001'
MQXPT_TCP	2	X'00000002'
MQXPT_NETBIOS	3	X'00000003'
MQXPT_SPX	4	X'00000004'
MQXPT_DECNET	5	X'00000005'
MQXPT_UDP	6	X'00000006'

MQXQH_* (Estructura de encabezado de cola de transmisión)

Tabla 421. Estructuras de constantes	
Nombre	Estructura
MQXQH_STRUC_ID	"XQH¬"
MQXQH_STRUC_ID_ARRAY	'X', 'Q', 'H', '¬'

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

Tabla 422. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQXQH_VERSION_1	1	X'00000001'
MQXQH_VERSION_ACTUAL	1	X'00000001'

MQXR_* (Motivos de salida)

Tabla 423. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQXR_ANTES	1	X'00000001'
MQXR_DESPUÉS	2	X'00000002'
MQXR_CONEXIÓN	3	X'00000003'
MQXR_INIT	11	X'0000000B'
MQXR_TERM	12	X'0000000C'
MQXR_MSG	13	X'0000000D'
MQXR_XMIT	14	X'0000000E'
MQXR_SEC_MSG	15	X'0000000F'
MQXR_INICIO_SEC	16	X'00000010'
MQXR_RETRY	17	X'00000011'
MQXR_AUTO_CLUSSDR	18	X'00000012'

Tabla 423. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQXR_RECEPTOR_AUTOMÁTICO	19	X'00000013'
MQXR_CLWL_ABIERTO	20	X'00000014'
MQXR_CLWL_PUT	21	X'00000015'
MQXR_CLWL_MOVE	22	X'00000016'
MQXR_CLWL_REPOS	23	X'00000017'
MQXR_CLWL_REPOS_MOVE	24	X'00000018'
MQXR_FIN_LOTE	25	X'00000019'
MQXR_ACK_RECEIVED	26	X'0000001A'
MQXR_AUTO_SVRCONN	27	X'0000001B'
MQXR_AUTO_RECEPTOR_CIRCULAR	28	X'0000001C'
MQXR_SEC_PARS	29	X'0000001D'

MQXR2_* (Respuesta de salida 2)

Tabla 424. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQXR2_PUT_WITH_DEF_ACTION	0	X'00000000'
MQXR2_PUT_WITH_DEF_USERID	1	X'00000001'
MQXR2_PUT_WITH_MSG_USERID	2	X'00000002'
MQXR2_USE_AGENT_BUFFER	0	X'00000000'
MQXR2_USE_EXIT_BUFFER	4	X'00000004'
MQXR2_DEFAULT_CONTINUATION	0	X'00000000'
MQXR2_CONTINUE_CHAIN	8	X'00000008'
MQXR2_SUPPRESS_CHAIN	16	X'00000010'
MQXR2_STATIC_CACHE	0	X'00000000'
MQXR2_DYNAMIC_CACHE	32	X'00000020'

MQXT_* (Identificadores de salida)

Tabla 425. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQXT_API_CROSSING_EXIT	1	X'00000001'
MQXT_API_SALIDA	2	X'00000002'
MQXT_CANAL_SEC_SALIDA	11	X'0000000B'
MQXT_CANAL_MENSAJE_SALIDA	12	X'0000000C'
MQXT_CANAL_ENVIAR_SALIDA	13	X'0000000D'
MQXT_CANAL_RCV_SALIDA	14	X'0000000E'
MQXT_CANAL_MENSAJE_REINTENTO_SALIDA	15	X'0000000F'
MQXT_CANAL_AUTOMÁTICO_SALIDA_DEF	16	X'00000010'
MQXT_CLUSTER_WORKLOAD_EXIT	20	X'00000014'
MQXT_PUBSUB_ENRUTAMIENTO_SALIDA	21	X'00000015'

MQXUA_* (Salir del valor del área de usuario)

Tabla 426. Nombres y valores constantes	
Nombre	Valor
MQXUA_NONE	X'00...00' (16 nulos)
MQXUA_NINGUNA_MATRIZ	'\0', '\0', ... (16 nulos)

MQXWD_* (Estructura descriptora de espera de salida)

Tabla 427. Estructuras de constantes	
Nombre	Estructura
MQXWD_STRUC_ID	"XWD¬"
MQXWD_STRUC_ID_ARRAY	'X', 'W', 'D', '¬'

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

Tabla 428. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQXWD_VERSION_1	1	X'00000001'

MQZAC_* (Estructura del contexto de la aplicación)

Tabla 429. Estructuras de constantes	
Nombre	Estructura
MQZAC_STRUC_ID	"ZAC¬"
MQZAC_STRUC_ID_ARRAY	'Z', 'A', 'C', '¬'

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

Tabla 430. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQZAC_VERSION_1	1	X'00000001'
MQZAC VERSIÓN_ACTUAL	1	X'00000001'

MQZAD_* (Estructura de datos de autoridad)

Tabla 431. Estructuras de constantes	
Nombre	Estructura
MQZAD_STRUC_ID	"ZAD¬"
MQZAD_STRUC_ID_ARRAY	'Z', 'A', 'D', '¬'

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

Tabla 432. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQZAD_VERSION_1	1	X'00000001'
MQZAD_VERSION_2	2	X'00000002'
MQZAD VERSIÓN_ACTUAL	2	X'00000002'

MQZAET_* (Tipos de entidad de servicios instalables)

Tabla 433. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQZAET_NINGUNO	0	X'00000000'
MQZAET_PRINCIPAL	1	X'00000001'
MQZAET_GRUPO	2	X'00000002'
MQZAET_DESCONOCIDO	3	X'00000003'

MQZAO_* (Autorizaciones de servicios instalables)

Tabla 434. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQZAO_CONNECT	1	X'00000001'
MQZAO_BROWSE	2	X'00000002'
MQZAO_INPUT	4	X'00000004'
MQZAO_OUTPUT	8	X'00000008'
MQZAO_INQUIRE	16	X'00000010'
MQZAO_SET	32	X'00000020'
MQZAO_PASS_IDENTITY_CONTEXT	64	X'00000040'
MQZAO_PASS_ALL_CONTEXT	128	X'00000080'
MQZAO_SET_IDENTITY_CONTEXT	256	X'00000100'
MQZAO_SET_ALL_CONTEXT	512	X'00000200'
MQZAO_ALTERNATE_USER_AUTHORITY	1024	X'00000400'
MQZAO_PUBLICAR	2048	X'00000800'
MQZAO_SUSCRIBIRSE	4096	X'00001000'
MQZAO_CURRÍCULUM	8192	X'00002000'
MQZAO_TODOS_MQI	16383	X'00003FFF'
MQZAO_CREAR	65536	X'00010000'
MQZAO_DELETE	131072	X'00020000'
MQZAO_DISPLAY	262144	X'00040000'
MQZAO_CHANGE	524288	X'00080000'
MQZAO_CLEAR	1048576	X'00100000'
MQZAO_CONTROL	2097152	X'00200000'
MQZAO_CONTROL_EXTENDED	4194304	X'00400000'
MQZAO_AUTORIZAR	8388608	X'00800000'
MQZAO_TODOS_ADMINISTRADORES	16646144	X'00FE0000'
MQZAO_TODOS	16662527	X'00FE3FFF'
MQZAO_REMOVE	16777216	X'01000000'
MQZAO_NINGUNO	0	X'00000000'

MQZAS_* (Versión de la interfaz de servicios instalables)

Tabla 435. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQZAS_VERSION_1	1	X'00000001'
MQZAS_VERSION_2	2	X'00000002'
MQZAS_VERSION_3	3	X'00000003'
MQZAS_VERSION_4	4	X'00000004'
MQZAS_VERSION_5	5	X'00000005'
MQZAS_VERSION_6	6	X'00000006'

MQZAT_* (Tipos de autenticación)

Tabla 436. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQZAT_CONTEXTO_INICIAL	0	X'00000000'
MQZAT_CAMBIAR_CONTEXTO	1	X'00000001'

MQZCI_* (Indicador de continuación de servicios instalables)

Tabla 437. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQZCI_DEFAULT	0	X'00000000'
MQZCI_CONTINUAR	0	X'00000000'
MQZCI_STOP	1	X'00000001'

MQZED_* (Estructura de datos de entidad)

Tabla 438. Estructuras de constantes	
Nombre	Estructura
MQZED_STRUC_ID	"ZED¬"
MQZED_STRUC_ID_ARRAY	'Z','E','D','¬'

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

Tabla 439. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQZED_VERSION_1	1	X'00000001'
MQZED_VERSION_2	2	X'00000002'
MQZED_VERSION_ACTUAL	2	X'00000002'

MQZFP_* (Estructura de parámetros libres)

Tabla 440. Estructuras de constantes	
Nombre	Estructura
MQZFP_STRUC_ID	"ZFP¬"
MQZFP_STRUC_ID_ARRAY	'Z','F','P','¬'

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

Tabla 441. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQZFP_VERSION_1	1	X'00000001'
MQZFP_VERSION_ACTUAL	1	X'00000001'

MQZIC_* (Estructura de contexto de identidad)

Tabla 442. Estructuras de constantes	
Nombre	Estructura
MQZIC_STRUC_ID	"ZIC¬"
MQZIC_STRUC_ID_ARRAY	'Z','I','C','¬'

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

Tabla 443. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQZIC_VERSION_1	1	X'00000001'
MQZIC_VERSION_ACTUAL	1	X'00000001'

MQZID_* (Identificadores de función para servicios)

Identificadores de función comunes a todos los servicios

Tabla 444. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQZID_INIT	0	X'00000000'
MQZID_TERM	1	X'00000001'

Identificadores de función para el servicio de autoridad

Tabla 445. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQZID_INIT_AUTHORITY	0	X'00000000'
MQZID_TERM_AUTHORITY	1	X'00000001'
MQZID_COMPROBAR_AUTORIDAD	2	X'00000002'
MQZID_COPY_ALL_AUTHORITY	3	X'00000003'
MQZID_DELETE_AUTHORITY	4	X'00000004'
MQZID_SET_AUTHORITY	5	X'00000005'
MQZID_OBTENER_AUTORIDAD	6	X'00000006'
MQZID_OBTENER_AUTORIDAD_EXPLÍCITA	7	X'00000007'
MQZID_REFRESH_CACHE	8	X'00000008'
MQZID_ENUMERAR_DATOS_DE_AUTORIDAD	9	X'00000009'
MQZID_AUTENTICAR_USUARIO	10	X'0000000A'
MQZID_USUARIO_GRATUITO	11	X'0000000B'
MQZID_CONSULTAR	12	X'0000000C'

Tabla 445. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQZID_COMPROBAR_PRIVILEGIADO	13	X'0000000D'

Identificadores de función para el servicio de nombres

Tabla 446. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQZID_INIT_NAME	0	X'00000000'
MQZID_TERM_NAME	1	X'00000001'
MQZID_BUSCAR_NOMBRE	2	X'00000002'
MQZID_INSERT_NAME	3	X'00000003'
MQZID_DELETE_NAME	4	X'00000004'

Identificadores de función para el servicio Userid

Tabla 447. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQZID_INIT_USERID	0	X'00000000'
MQZID_TERM_ID_USUARIO	1	X'00000001'
MQZID_BUSCAR_ID_USUARIO	2	X'00000002'

MQZIO_* (Opciones de inicialización de servicios instalables)

Tabla 448. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQZIO_PRIMARIO	0	X'00000000'
MQZIO_SECUNDARIO	1	X'00000001'

MQZNS_* (Versión de la interfaz del servicio de nombres)

Tabla 449. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQZNS_VERSION_1	1	X'00000001'

MQZSE_* (Indicador de inicio de enumeración de servicios instalables)

Tabla 450. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQZSE_INICIO	1	X'00000001'
MQZSE_CONTINUAR	0	X'00000000'

MQZSL_* (Indicador de selector de servicios instalables)

Tabla 451. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQZSL_NO_DEVUELTO	0	X'00000000'

Tabla 451. Valores de las constantes (continuación)		
Nombre	Valor decimal	Valor hexadecimal
MQZSL_DEVOLVIDO	1	X'00000001'

MQZTO_* (Opciones de terminación de servicios instalables)

Tabla 452. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQZTO_PRIMARIO	0	X'00000000'
MQZTO_SECUNDARIO	1	X'00000001'

MQZUS_* (Versión de la interfaz del servicio de identificación de usuario)

Tabla 453. Valores de las constantes		
Nombre	Valor decimal	Valor hexadecimal
MQZUS_VERSION_1	1	X'00000001'

Tipos de datos utilizados en el lenguaje de programación « MQI »

Información sobre los tipos de datos que se pueden utilizar en la interfaz de colas de mensajes (MQI).
Descripciones, campos y declaraciones de idioma para los idiomas pertinentes con cada tipo de datos.

Tipos de datos y programación para el lenguaje de programación « MQI »

Presentación de los tipos de datos elementales y estructurados, y cómo utilizar la biblioteca « MQI » mediante programación en C, programación en « COBOL » o programación en « High Level Assembler ».

Tipos de datos elementales

Información sobre los tipos de datos utilizados en el « MQI » o en las funciones de salida. Estos se describen detalladamente, seguidos de ejemplos que muestran cómo declarar los tipos de datos elementales en los lenguajes de programación compatibles.

Los tipos de datos utilizados en el « MQI » o en las funciones de salida son los siguientes:

- Tipos de datos elementales, o
- Agregados de tipos de datos elementales (matrices o estructuras)

En el « MQI » o en las funciones de salida se utilizan los siguientes tipos de datos elementales:

Tabla 454. Nombres, tipos y descripciones de los tipos de datos elementales		
Nombre del tipo de datos elemental	Tipo de datos	Descripción
MQBOOL	Boolean	El tipo de datos MQBOOL representa un valor booleano. El valor 0 representa falso. Cualquier otro valor representa verdadero. Un MQBOOL debe alinearse como el tipo de datos MQLONG.

Tabla 454. Nombres, tipos y descripciones de los tipos de datos elementales (continuación)

Nombre del tipo de datos elemental	Tipo de datos	Descripción
MQBYTE	Byte	El tipo de datos MQBYTE representa un solo byte de datos. No se le da ninguna interpretación particular al byte; se trata como una cadena de bits, y no como un número binario o un carácter. No se requiere ninguna alineación especial. Cuando se envían datos MQBYTE entre gestores de colas que utilizan diferentes conjuntos de caracteres o codificaciones, los datos MQBYTE no se convierten de ninguna manera. Los campos <i>MsgId CorrelId</i> y de la estructura MQMD son así. A veces se utiliza una matriz de MQBYTE para representar un área de la memoria principal que el gestor de colas desconoce. Por ejemplo, el área podría contener datos de mensajes de aplicación o una estructura. La alineación de los límites de esta zona debe ser compatible con la naturaleza de los datos que contiene. En el lenguaje de programación C, se puede utilizar cualquier tipo de datos para los parámetros de función que se muestran como matrices de MQBYTE. Esto se debe a que dichos parámetros siempre se pasan por dirección, y en C el parámetro de la función se declara como un puntero a void.

Tabla 454. Nombres, tipos y descripciones de los tipos de datos elementales (continuación)

Nombre del tipo de datos elemental	Tipo de datos	Descripción
MQBYTEn	Cadena de n bytes	Cada tipo de datos MQBYTEn representa una cadena de n bytes, donde n puede tomar cualquiera de los siguientes valores: 8, 16, 24, 32, 40 o 128. Cada byte se describe mediante el tipo de datos MQBYTE. No se requiere ninguna alineación especial. Si los datos de la cadena de bytes son más cortos que la longitud definida de la cadena, los datos deben rellenarse con valores nulos para completar la cadena. Cuando el gestor de colas devuelve cadenas de bytes a la aplicación (por ejemplo, en la llamada MQGET), el gestor de colas rellena con valores nulos hasta alcanzar la longitud definida de la cadena. Las constantes con nombre están disponibles para definir las longitudes de los campos de cadena de bytes. Estos se enumeran en “Constantes” en la página 7

Tabla 454. Nombres, tipos y descripciones de los tipos de datos elementales (continuación)

Nombre del tipo de datos elemental	Tipo de datos	Descripción
MQCHAR	Carácter	El tipo de datos MQCHAR representa un carácter de un solo byte, o un byte de un carácter de doble byte o multibyte. No se requiere ninguna alineación especial. Cuando se envían datos MQCHAR entre gestores de colas que utilizan diferentes conjuntos de caracteres o codificaciones, los datos MQCHAR suelen requerir una conversión para que puedan interpretarse correctamente. El gestor de colas realiza esta operación automáticamente para los datos MQCHAR de la estructura MQMD. La conversión de datos MQCHAR en los datos del mensaje de aplicación se controla mediante la opción MQGMO_CONVERT especificada en la llamada MQGET; consulte la descripción de esta opción en “MQGMO - Opciones para obtener mensajes” en la página 331 para obtener más detalles.

Tabla 454. Nombres, tipos y descripciones de los tipos de datos elementales (continuación)

Nombre del tipo de datos elemental	Tipo de datos	Descripción
MQCHARn	Cadena de n caracteres	Cada tipo de datos MQCHARn representa una cadena de n caracteres, donde n puede tomar cualquiera de los siguientes valores: 4, 8, 12, 20, 28, 32, 48, 64, 128 o 256. Cada carácter se describe mediante el tipo de datos MQCHAR. No se requiere ninguna alineación especial. Si los datos de la cadena son más cortos que la longitud definida de la cadena, los datos deben rellenarse con espacios en blanco para completar la cadena. En algunos casos, se puede utilizar un carácter nulo para terminar la cadena prematuramente, en lugar de rellenarla con espacios en blanco; el carácter nulo y los caracteres que le siguen se tratan como espacios en blanco, hasta la longitud definida de la cadena. Los lugares donde se puede utilizar un valor nulo se identifican en las descripciones de las llamadas y los tipos de datos. Cuando el gestor de colas devuelve cadenas de caracteres a la aplicación (por ejemplo, en la llamada MQGET), el gestor de colas siempre rellena con espacios en blanco hasta alcanzar la longitud definida de la cadena; el gestor de colas no utiliza el carácter nulo para delimitar la cadena. Hay constantes con nombre disponibles que definen las longitudes de los campos de cadena de caracteres y se enumeran en “Constantes” en la página 7.

Tabla 454. Nombres, tipos y descripciones de los tipos de datos elementales (continuación)

Nombre del tipo de datos elemental	Tipo de datos	Descripción
MQFLOAT32	Número de coma flotante de 32 bits	El tipo de datos ` MQFLOAT32 ` es un número de 32 bits en coma flotante representado utilizando el formato estándar IEEE en coma flotante. Un objeto de interfaz de programa de aplicación (MQFLOAT32) debe estar alineado en un límite de 4 bytes. El uso de MQFLOAT32 en C en z/OS requiere el uso de la bandera del compilador FLOAT(IEEE). El uso de MQFLOAT32 en COBOL está limitado a compiladores que admiten números de coma flotante en formato IEEE. Esto podría requerir el uso de la bandera del compilador FLOAT(NATIVE).
MQFLOAT64	Número de coma flotante de 64 bits	El tipo de datos ` MQFLOAT64 ` es un número de coma flotante de 64 bits representado utilizando el formato de coma flotante estándar IEEE. Un objeto de interfaz de programa de aplicación (MQFLOAT64) debe estar alineado en un límite de 8 bytes. El uso de MQFLOAT64 en C en z/OS requiere el uso de la bandera del compilador FLOAT(IEEE). El uso de MQFLOAT64 en COBOL está limitado a compiladores que admiten números de coma flotante en formato IEEE. Esto podría requerir el uso de la bandera del compilador FLOAT(NATIVE).

Tabla 454. Nombres, tipos y descripciones de los tipos de datos elementales (continuación)

Nombre del tipo de datos elemental	Tipo de datos	Descripción
MQHCONFIG	Manejo de la configuración	El tipo de datos MQHCONFIG representa un identificador de configuración, es decir, el componente que se está configurando para un servicio instalable concreto. Un identificador de configuración debe alinearse con su límite natural. Las aplicaciones no deben depender del formato de los datos almacenados dentro de este identificador. Si es válido, su valor está pensado para poder utilizarse en futuras llamadas a `MQI`, pero no tiene ningún otro significado más allá de ese propósito.
MQHCONN	Descriptor de contexto de conexión	El tipo de datos MQHCONN representa un identificador de conexión, es decir, la conexión a un gestor de colas concreto. Un identificador de conexión debe alinearse en un límite de 4 bytes. Las aplicaciones no deben depender del formato de los datos almacenados dentro de este identificador. Si es válido, su valor está pensado para poder utilizarse en futuras llamadas a `MQI`, pero no tiene ningún otro significado más allá de ese propósito.
MQHMSG	Descriptor de contexto de mensaje	El tipo de datos MQHMSG representa un identificador de mensaje que da acceso a un mensaje. Un identificador de mensaje debe estar alineado en un límite de 8 bytes. Las aplicaciones no deben depender del formato de los datos almacenados dentro de este identificador. Si es válido, su valor está pensado para poder utilizarse en futuras llamadas a `MQI`, pero no tiene ningún otro significado más allá de ese propósito.

Tabla 454. Nombres, tipos y descripciones de los tipos de datos elementales (continuación)

Nombre del tipo de datos elemental	Tipo de datos	Descripción
MQHOBj	Descriptor de contexto del objeto	El tipo de datos MQHOBj representa un identificador de objeto que da acceso a un objeto. Un identificador de objeto debe estar alineado en un límite de 4 bytes. Las aplicaciones no deben depender del formato de los datos almacenados dentro de este identificador. Si es válido, su valor está pensado para poder utilizarse en futuras llamadas a `MQI`, pero no tiene ningún otro significado más allá de ese propósito.
MQINT8	entero con signo de 8 bits	El tipo de datos `MQINT8` es un entero con signo de 8 bits que puede tomar cualquier valor en el rango de -128 a +127, a menos que el contexto lo restrinja de otra manera.
MQINT16	Entero con signo de 16 bits	El tipo de datos `MQINT16` es un entero con signo de 16 bits que puede tomar cualquier valor en el rango de -32 768 a +32 767, a menos que el contexto lo restrinja de otra manera. Un objeto de interfaz de programa de aplicación (MQINT16) debe estar alineado en un límite de 2 bytes.
MQINT32	Entero con signo de 32 bits	El tipo de datos `MQINT32` es un entero binario con signo de 32 bits que puede tomar cualquier valor en el rango de -2 147 483 648 a +2 147 483 647, a menos que el contexto lo restrinja. Véase la definición de MQLONG.

Tabla 454. Nombres, tipos y descripciones de los tipos de datos elementales (continuación)

Nombre del tipo de datos elemental	Tipo de datos	Descripción
MQINT64	Entero con signo de 64 bits	El tipo de datos `MQINT64` es un entero con signo de 64 bits que puede tomar cualquier valor en el rango comprendido entre `-9 223372036854775808` y `+9223372036854775807`, a menos que el contexto imponga restricciones. Para COBOL, el rango válido está limitado a -999 999 999 999 999 999 hasta +999 999 999 999 999 999. Un entero de 64 bits debe estar alineado en un límite de 8 bytes.
MQLONG	Entero con signo de 32 bits	El tipo de datos MQLONG es un entero binario con signo de 32 bits que puede tomar cualquier valor en el rango -2 147 483 648 a +2 147 483 647, a menos que el contexto lo restrinja de otro modo. Para COBOL, el rango válido está limitado entre -999 999 999 y +999 999 999. Un MQLONG debe estar alineado en un límite de 4 bytes.
MQPID	Identificador proceso	El IBM MQ identificador del proceso. Este es el mismo identificador que se utiliza en los volcados de seguimiento y FFST™ de MQ, pero puede ser diferente del identificador de proceso del sistema operativo.
MQPTR	Puntero	El tipo de datos MQPTR es la dirección de datos de cualquier tipo. Un puntero debe estar alineado en su límite natural; este es un límite de 16 bytes en IBM i, y un límite de 8 bytes en otras plataformas. Algunos lenguajes de programación admiten punteros tipados; el lenguaje de programación « MQI » también los utiliza en algunos casos (por ejemplo, PMQCHAR y PMQLONG en el lenguaje de programación C).

Tabla 454. Nombres, tipos y descripciones de los tipos de datos elementales (continuación)

Nombre del tipo de datos elemental	Tipo de datos	Descripción
MQTID	Identificador de hebra	El identificador de hebra de IBM MQ. Este es el mismo identificador utilizado en los volcados de seguimiento y FFST™ de MQ, pero puede ser diferente del identificador de subprocesos del sistema operativo.
MQUINT8	Entero sin signo de 8 bits	El tipo de datos ` MQUINT8 ` es un entero sin signo de 8 bits que puede tomar cualquier valor en el rango de 0 a +255, a menos que el contexto lo restrinja de otra manera.
MQUINT16	entero sin signo de 16 bits	El tipo de datos ` MQUINT16 ` es un entero sin signo de 16 bits que puede tomar cualquier valor en el rango de 0 a +65 535, a menos que el contexto lo restrinja. Un objeto de interfaz de programa de aplicación (MQUINT16) debe estar alineado en un límite de 2 bytes.
MQUINT32	Entero de 32 bits sin signo	El tipo de datos « MQUINT32 » es un entero binario sin signo de 32 bits. Véase la definición de MQULONG .
MQUINT64	Entero de 64 bits sin signo	El tipo de datos ` MQUINT64 ` es un entero sin signo de 64 bits que puede tomar cualquier valor en el rango de 0 a +18 446 744 073 709 551 615, a menos que el contexto lo restrinja de otra manera. Para COBOL, el rango válido está limitado a 0 a +999 999 999 999 999 999. Un entero de 64 bits debe estar alineado en un límite de 8 bytes.

Tabla 454. Nombres, tipos y descripciones de los tipos de datos elementales (continuación)

Nombre del tipo de datos elemental	Tipo de datos	Descripción
MQULONG	Entero de 32 bits sin signo	El tipo de datos MQULONG es un entero binario sin signo de 32 bits que puede tomar cualquier valor en el rango de 0 a +4 294 967 294, a menos que el contexto lo restrinja de otra manera. Para COBOL, el rango válido está limitado entre 0 y +999 999 999. Un MQULONG debe estar alineado en un límite de 4 bytes.
PMQACH	Puntero	Puntero a una estructura de datos de tipo MQACH
PMQAIR	Puntero	Puntero a una estructura de datos de tipo MQAIR
PMQAXC	Puntero	Puntero a una estructura de datos de tipo MQAXC
PMQAXP	Puntero	Puntero a una estructura de datos de tipo MQAXP
PMQBMHO	Puntero	Puntero a una estructura de datos de tipo MQBMHO
PMQBO	Puntero	Puntero a una estructura de datos de tipo MQBO
PMQBOOL	Puntero	Puntero a datos de tipo MQBOOL
PMQBYTE	Puntero	Puntero a datos de tipo MQBYTE
PMQBYTEn	Puntero	Puntero a datos de tipo MQBYTEn, donde n puede ser 8, 16, 24, 32, 40, 128
PMQCBC	Puntero	Puntero a una estructura de datos de tipo MQCBC
PMQCBD	Puntero	Puntero a una estructura de datos de tipo MQCBD
PMQCHAR	Puntero	Puntero a datos de tipo MQCHAR
PMQCHARN	Puntero	Puntero a un tipo de datos MQCHARN, donde n puede ser 4, 8, 12, 20, 28, 32, 48, 64, 128, 256, 264
PMQCHARV	Puntero	Puntero a una estructura de datos de tipo MQCHARV
PMQCIH	Puntero	Puntero a una estructura de datos de tipo MQCIH
PMQCMHO	Puntero	Puntero a una estructura de datos de tipo MQCMHO

Tabla 454. Nombres, tipos y descripciones de los tipos de datos elementales (continuación)		
Nombre del tipo de datos elemental	Tipo de datos	Descripción
PMQCNO	Puntero	Puntero a una estructura de datos de tipo MQCNO
PMQCSP	Puntero	Puntero a una estructura de datos de tipo MQCSP
PMQCTLO	Puntero	Puntero a una estructura de datos de tipo MQCTLO
PMQDH	Puntero	Puntero a una estructura de datos de tipo MQDH
PMQDHO	Puntero	Puntero a una estructura de datos de tipo MQDHO
PMQDLH	Puntero	Puntero a una estructura de datos de tipo MQDLH
PMQDMHO	Puntero	Puntero a una estructura de datos de tipo MQDMHO
PMQDMPO	Puntero	Puntero a una estructura de datos de tipo MQDMPO
PMQEPH	Puntero	Puntero a una estructura de datos de tipo MQEPH
PMQFLOAT32	Puntero	Puntero a una estructura de datos de tipo MQFLOAT32
PMQFLOAT64	Puntero	Puntero a una estructura de datos de tipo MQFLOAT64
PMQFUNC	Puntero	Puntero a una función
PMQGMO	Puntero	Puntero a una estructura de datos de tipo MQGMO
PMQHCONFIG	Puntero	Puntero a datos de tipo MQHCONFIG
PMQHCONN	Puntero	Puntero a datos de tipo MQHCONN
PMQHMSG	Puntero	Puntero a datos de tipo MQHMSG
PMQHOBJ	Puntero	Puntero a datos de tipo MQHOBJ
PMQIIH	Puntero	Puntero a una estructura de datos de tipo MQIIH
PMQIMPO	Puntero	Puntero a una estructura de datos de tipo MQIMPO
PMQINT8	Puntero	Puntero a datos de tipo MQINT8
PMQINT16	Puntero	Puntero a datos de tipo MQINT16
PMQINT32	Puntero	Puntero a datos de tipo MQINT32
PMQINT64	Puntero	Puntero a datos de tipo MQINT64
PMQLONG	Puntero	Puntero a datos de tipo MQLONG

Tabla 454. Nombres, tipos y descripciones de los tipos de datos elementales (continuación)		
Nombre del tipo de datos elemental	Tipo de datos	Descripción
PMQMD	Puntero	Puntero a estructura de tipo MQMD
PMQMDE	Puntero	Puntero a una estructura de datos de tipo MQMDE
PMQMD1	Puntero	Puntero a una estructura de datos de tipo MQMD1
PMQMD2	Puntero	Puntero a una estructura de datos de tipo MQMD2
PMQMHBO	Puntero	Puntero a una estructura de datos de tipo MQMHBO
PMQOD	Puntero	Puntero a una estructura de datos de tipo MQOD
PMQOR	Puntero	Puntero a una estructura de datos de tipo MQOR
PMQPD	Puntero	Puntero a una estructura de datos de tipo MQPD
PMQPID	Puntero	Puntero a un identificador de proceso
PMQMD	Puntero	Puntero a una estructura de datos de tipo MQMD
PMQPMO	Puntero	Puntero a una estructura de datos de tipo MQPMO
PMQPTR	Puntero	Puntero a datos de tipo MQPTR
PMQRFH	Puntero	Puntero a una estructura de datos de tipo MQRFH
PMQRFH2	Puntero	Puntero a una estructura de datos de tipo MQRFH2
PMQRMH	Puntero	Puntero a una estructura de datos de tipo MQRMH
PMQRR	Puntero	Puntero a una estructura de datos de tipo MQRR
PMQSCO	Puntero	Puntero a una estructura de datos de tipo MQSCO
PMQSD	Puntero	Puntero a una estructura de datos de tipo MQSD
PMQSMPO	Puntero	Puntero a una estructura de datos de tipo MQSMPO
PMQSRO	Puntero	Puntero a una estructura de datos de tipo MQSRO
PMSSTS	Puntero	Puntero a una estructura de datos de tipo MQSTS

Tabla 454. Nombres, tipos y descripciones de los tipos de datos elementales (continuación)		
Nombre del tipo de datos elemental	Tipo de datos	Descripción
PMQTID	Puntero	Puntero a un identificador de subproceso
PMQTM	Puntero	Puntero a una estructura de datos de tipo MQTM
PMQTMCM2	Puntero	Puntero a una estructura de datos de tipo MQTMCM2
PMQUINT8	Puntero	Puntero a un tipo de datos de MQUINT8
PMQUINT16	Puntero	Puntero a un tipo de datos de MQUINT16
PMQUINT32	Puntero	Puntero a un tipo de datos de MQUINT32
PMQUINT64	Puntero	Puntero a un tipo de datos de MQUINT64
PMQULONG	Puntero	Puntero a un tipo de datos MQULONG
PMQVOID	Puntero	
¿Me lo preguntas?	Puntero	Puntero a una estructura de datos de tipo MQWIH
PMQXQH	Puntero	Puntero a una estructura de datos de tipo MQXQH

Declaraciones de tipos de datos C

Tabla 455. Nombres y representaciones de tipos de datos C	
Tipo de datos	Representación
MQBOOL	<code>typedef MQLONG MQBOOL;</code>
MQBYTE	<code>typedef unsigned char MQBYTE;</code>
MQBYTE8	<code>typedef MQBYTE MQBYTE8[8];</code>
MQBYTE16	<code>typedef MQBYTE MQBYTE16[16];</code>
MQBYTE24	<code>typedef MQBYTE MQBYTE24[24];</code>
MQBYTE32	<code>typedef MQBYTE MQBYTE32[32];</code>
MQBYTE40	<code>typedef MQBYTE MQBYTE40[40];</code>
MQCHAR	<code>typedef char MQCHAR;</code>

Tabla 455. Nombres y representaciones de tipos de datos C (continuación)	
Tipo de datos	Representación
MQCHAR4	<code>typedef MQCHAR MQCHAR4[4];</code>
MQCHAR8	<code>typedef MQCHAR MQCHAR8[8];</code>
MQCHAR12	<code>typedef MQCHAR MQCHAR12[12];</code>
MQCHAR20	<code>typedef MQCHAR MQCHAR20[20];</code>
MQCHAR28	<code>typedef MQCHAR MQCHAR28[28];</code>
MQCHAR32	<code>typedef MQCHAR MQCHAR32[32];</code>
MQCHAR48	<code>typedef MQCHAR MQCHAR48[48];</code>
MQCHAR64	<code>typedef MQCHAR MQCHAR64[64];</code>
MQCHAR128	<code>typedef MQCHAR MQCHAR128[128];</code>
MQCHAR256	<code>typedef MQCHAR MQCHAR256[256];</code>
MQFLOAT32	<code>typedef float MQFLOAT32;</code>
MQFLOAT64	<code>typedef double MQFLOAT64;</code>
MQHCONFIG	<code>typedef void MQPOINTER MQHCONFIG;</code>
MQHCONN	<code>typedef MQLONG MQHCONN;</code>
MQHOBJ	<code>typedef MQLONG MQHOBJ;</code>
MQINT8	<code>typedef signed char MQINT8;</code>
MQINT16	<code>typedef short MQINT16;</code>

Tabla 455. Nombres y representaciones de tipos de datos C (continuación)

Tipo de datos	Representación
MQINT64	 En 64 bits UNIX: <pre>typedef long;</pre>  En 32 bits AIX: <pre>typedef int64_t;</pre>    El Linux, IBM i, y z/OS: <pre>typedef long long;</pre>  En Windows: <pre>typedef _int64;</pre>
MQLONG	 En IBM i: <pre>typedef long MQLONG;</pre>   En otras plataformas: <pre>if defined(MQ_64_BIT) typedef int MQLONG; else typedef long MQLONG;</pre>
MQPID	<pre>typedef MQLONG MQPID;</pre>
MQPTR	<pre>typedef void MQPOINTER MQPTR;</pre>
MQTID	<pre>typedef MQLONG MQTID;</pre>
MQUINT8	<pre>typedef unsigned char MQUINT8;</pre>
MQUINT16	<pre>typedef unsigned short MQUINT16;</pre>

Tabla 455. Nombres y representaciones de tipos de datos C (continuación)

Tipo de datos	Representación
MQUINT64	 En 64 bits UNIX: <pre>typedef unsigned long;</pre>  En 32 bits AIX: <pre>typedef uint64_t;</pre>    El Linux, IBM i, y z/OS: <pre>typedef unsigned long long;</pre>  En Windows: <pre>typedef unsigned _int64;</pre>
MQULONG	 En IBM i: <pre>typedef unsigned long MQULONG;</pre>   En otras plataformas: <pre>if defined(MQ_64_BIT) typedef unsigned int MQULONG; else typedef unsigned long MQULONG;</pre>
PMQBO	<pre>typedef MQBO MQPOINTER PMQBO;</pre>
PMQBOOL	<pre>typedef MQBOOL MQPOINTER PMQBOOL;</pre>
PMQBYTE	<pre>typedef MQBYTE MQPOINTER PMQBYTE;</pre>
PMQBYTE8	<pre>typedef MQBYTE8[8] MQPOINTER PMQBYTE8[8];</pre>
PMQBYTE16	<pre>typedef MQBYTE16[16] MQPOINTER PMQBYTE16[16];</pre>
PMQBYTE24	<pre>typedef MQBYTE24[24] MQPOINTER PMQBYTE24[24];</pre>
PMQBYTE32	<pre>typedef MQBYTE32[32] MQPOINTER PMQBYTE32[32];</pre>
PMQBYTE40	<pre>typedef MQBYTE40[40] MQPOINTER PMQBYTE40[40];</pre>
PMQBYTE128	<pre>typedef MQBYTE128[128] MQPOINTER PMQBYTE128[128];</pre>
PMQCHAR	<pre>typedef MQCHAR MQPOINTER PMQCHAR;</pre>

Tabla 455. Nombres y representaciones de tipos de datos C (continuación)

Tipo de datos	Representación
PMQCHAR4	<code>typedef MQCHAR4[4] MQPOINTER PMQCHAR4[4];</code>
PMQCHAR8	<code>typedef MQCHAR8[8] MQPOINTER PMQCHAR8[8];</code>
PMQCHAR12	<code>typedef MQCHAR12[12] MQPOINTER PMQCHAR12[12];</code>
PMQCHAR20	<code>typedef MQCHAR20[20] MQPOINTER PMQCHAR20[20];</code>
PMQCHAR28	<code>typedef MQCHAR28[28] MQPOINTER PMQCHAR28[28];</code>
PMQCHAR32	<code>typedef MQCHAR32[32] MQPOINTER PMQCHAR32[32];</code>
PMQCHAR48	<code>typedef MQCHAR48[48] MQPOINTER PMQCHAR48[48];</code>
PMQCHAR64	<code>typedef MQCHAR64[64] MQPOINTER PMQCHAR64[64];</code>
PMQCHAR128	<code>typedef MQCHAR128[128] MQPOINTER PMQCHAR128[128];</code>
PMQCHAR256	<code>typedef MQCHAR256[256] MQPOINTER PMQCHAR256[256];</code>
PMQCHAR264	<code>typedef MQCHAR264[264] MQPOINTER PMQCHAR264[264];</code>
PMQCIH	<code>typedef MQCIH MQPOINTER PMQCIH;</code>
PMQCNO	<code>typedef MQCNO MQPOINTER PMQCNO;</code>
PMQDLH	<code>typedef MQDLH MQPOINTER PMQDLH;</code>
PMQFUNC	<code>typedef void MQPOINTER PMQFUNC;</code>
PMQFLOAT32	<code>typedef MQFLOAT32 MQPOINTER PMQFLOAT32;</code>
PMQFLOAT64	<code>typedef MQFLOAT64 MQPOINTER PMQFLOAT64;</code>
PMQGM0	<code>typedef MQGM0 MQPOINTER PMQGM0;</code>
PMQHCONFIG	<code>typedef MQHCONFIG MQPOINTER PMQHCONFIG;</code>
PMQHCONN	<code>typedef MQHCONN MQPOINTER PMQHCONN;</code>
PMQHOBJ	<code>typedef MQHOBj MQPOINTER PMQHOBj;</code>

Tabla 455. Nombres y representaciones de tipos de datos C (continuación)	
Tipo de datos	Representación
PMQIIH	<code>typedef MQIIH MQPOINTER PMQIIH;</code>
PMQINT8	<code>typedef MQINT8 MQPOINTER PMQINT8;</code>
PMQINT16	<code>typedef MQINT16 MQPOINTER PMQINT16;</code>
PMQLONG	<code>typedef MQLONG MQPOINTER PMQLONG;</code>
PMQMD	<code>typedef MQMD MQPOINTER PMQMD;</code>
PMQMD1	<code>typedef MQMD1[1] MQPOINTER PMQMD1[1];</code>
PMQMDE	<code>typedef MQMDE MQPOINTER PMQMDE;</code>
PMQOD	<code>typedef MQOD MQPOINTER PMQOD;</code>
PMQPMO	<code>typedef MQPMO MQPOINTER PMQPMO;</code>
PMQPTR	<code>typedef MQPTR MQPOINTER PMQPTR;</code>
PMQRFH	<code>typedef MQRFH MQPOINTER PMQRFH;</code>
PMQRFH2	<code>typedef MQRFH2[2] MQPOINTER PMQRFH2[2];</code>
PMQRMH	<code>typedef MQRMH MQPOINTER PMQRMH;</code>
PMQTM	<code>typedef MQTM MQPOINTER PMQTM;</code>
PMQTM2	<code>typedef MQTM2[2] MQPOINTER PMQTM2[2];</code>
PMQUINT8	<code>typedef MQUINT8 MQPOINTER PMQUINT8;</code>
PMQUINT16	<code>typedef MQUINT16 MQPOINTER PMQUINT16;</code>
PMQULONG	<code>typedef MQULONG MQPOINTER PMQULONG;</code>
PMQVOID	<code>typedef void MQPOINTER PMQVOID;</code>
¿Me lo preguntas?	<code>typedef MQWIH MQPOINTER PMQWIH;</code>
PMQXQH	<code>typedef MQXQH MQPOINTER PMQXQH;</code>

Tabla 455. Nombres y representaciones de tipos de datos C (continuación)	
Tipo de datos	Representación
PPMQBO	<code>typedef PMQBO MQPOINTER PPMQBO;</code>
PPMQBYTE	<code>typedef PMQBYTE MQPOINTER PPMQBYTE;</code>
PPMQCHAR	<code>typedef PMQCHAR MQPOINTER PPMQCHAR;</code>
PPMQCNO	<code>typedef PMQCNO MQPOINTER PPMQCNO;</code>
PPMQGMO	<code>typedef PMQGMO MQPOINTER PPMQGMO;</code>
PPMQHCONN	<code>typedef PMQHCONN MQPOINTER PPMQHCONN;</code>
PPMQHOBJ	<code>typedef PMQHOBJ MQPOINTER PPMQHOBJ;</code>
PPMQLONG	<code>typedef PMLONG MQPOINTER PPMQLONG;</code>
PPMQMD	<code>typedef PMQMD MQPOINTER PPMQMD;</code>
PPMQOD	<code>typedef PMQOD MQPOINTER PPMQOD;</code>
PPMQPMO	<code>typedef PMQPMO MQPOINTER PPMQPMO;</code>
PPMQULONG	<code>typedef PMQULONG MQPOINTER PPMQULONG;</code>
PPMQVOID	<code>typedef PMQVOID MQPOINTER PPMQVOID;</code>
Donde defined(MQ_64_BIT) significa una plataforma de 64 bits.	

Consulte “Tipos de datos” en la [página 206](#) para obtener una descripción de la variable macro MQPOINTER.

COBOL declaraciones de tipos de datos

Tabla 456. COBOL Nombres y representaciones de tipos de datos	
Tipo de datos	Representación
MQBOOL	<code>PIC S9(9) BINARY</code>
MQBYTE	<code>PIC X</code>
MQBYTE8	<code>PIC X(8)</code>
MQBYTE16	<code>PIC X(16)</code>

Tabla 456. COBOL Nombres y representaciones de tipos de datos (continuación)	
Tipo de datos	Representación
MQBYTE24	PIC X(24)
MQBYTE32	PIC X(32)
MQBYTE40	PIC X(40)
MQCHAR	PIC X
MQCHAR4	PIC X(4)
MQCHAR8	PIC X(8)
MQCHAR12	PIC X(12)
MQCHAR20	PIC X(20)
MQCHAR28	PIC X(28)
MQCHAR32	PIC X(32)
MQCHAR48	PIC X(48)
MQCHAR64	PIC X(64)
MQCHAR128	PIC X(128)
MQCHAR256	PIC X(256)
MQFLOAT32	USAGE COMP-1
MQFLOAT64	USAGE COMP-2
MQHCONN	En z/OS PIC S9(9) COMP-5 En otras plataformas PIC S9(9) BINARY
MQHOBJ	PIC S9(9) BINARY
MQINT8	PIC S9(2) BINARY

Tabla 456. COBOL Nombres y representaciones de tipos de datos (continuación)

Tipo de datos	Representación
MQINT16	PIC S9(4) BINARY
MQINT64	PIC S9(18) BINARY
MQLONG	PIC S9(9) BINARY
MQPTR	POINTER
MQUINT8	PIC 9(2) BINARY
MQUINT16	PIC 9(4) BINARY
MQUINT64	PIC 9(18) BINARY
MQULONG	PIC 9(9) BINARY

PL/I declaraciones de tipos de datos

Tabla 457. PL/I Nombres y representaciones de tipos de datos

Tipo de datos	Representación
MQBOOL	fixed bin(31)
MQBYTE	char(1)
MQBYTE8	char(8)
MQBYTE16	char(16)
MQBYTE24	char(24)
MQBYTE32	char(32)
MQBYTE40	char(40)
MQCHAR	char(1)
MQCHAR4	char(4)
MQCHAR8	char(8)
MQCHAR12	char(12)

Tabla 457. PL/I Nombres y representaciones de tipos de datos (continuación)	
Tipo de datos	Representación
MQCHAR20	char(20)
MQCHAR28	char(28)
MQCHAR32	char(32)
MQCHAR48	char(48)
MQCHAR64	char(64)
MQCHAR128	char(128)
MQCHAR256	char(256)
MQFLOAT32	binary float(21) ieee
MQFLOAT64	binary float(52) ieee
MQHCONN	fixed bin(31)
MQHOBJ	fixed bin(31)
MQINT8	fixed bin(7)
MQINT16	fixed bin(15)
MQINT64	fixed bin(63)
MQLONG	fixed bin(31)
MQPTR	pointer
MQUINT8	fixed bin(8)
MQUINT16	fixed bin(16)
MQUINT64	fixed bin(64)
MQULONG	fixed bin(32)

High Level Assembler declaraciones de tipos de datos

Tabla 458. System/390 Nombres y representaciones de tipos de datos del ensamblador

Tipo de datos	Representación
MQBOOL	DS F
MQBYTE	DS XL1
MQBYTE8	DS XL8
MQBYTE16	DS XL16
MQBYTE24	DS XL24
MQBYTE32	DS XL32
MQBYTE40	DS XL40
MQCHAR	DS CL1
MQCHAR4	DS CL4
MQCHAR8	DS CL8
MQCHAR12	DS CL12
MQCHAR20	DS CL20
MQCHAR28	DS CL28
MQCHAR32	DS CL32
MQCHAR48	DS CL48
MQCHAR64	DS CL64
MQCHAR128	DS CL128
MQCHAR256	DS CL256
MQFLOAT32	DS EB
MQFLOAT64	DS DB
MQHCONN	DS F

Tabla 458. System/390 Nombres y representaciones de tipos de datos del ensamblador (continuación)	
Tipo de datos	Representación
MQHOBJ	DS F
MQINT8	DS XL1
MQINT16	DS H
MQINT64	DS D
MQLONG	DS F
MQPTR	DS F
MQUINT8	DS XL1
MQUINT16	DS H
MQUINT64	DS D
MQULONG	DS F

Tipos de datos de estructura

Un resumen de los tipos de datos de estructura, las reglas para mapear de forma coherente las estructuras de MQI y las convenciones utilizadas en la descripción de cada tipo de datos de estructura.

- “Resumen de los tipos de datos de estructura utilizados en las llamadas a « MQI » o en las funciones de salida” en la página 201
- “Resumen de los tipos de datos de estructura utilizados en los datos de mensaje” en la página 203
- “Normas para mapear las estructuras de MQI de forma coherente” en la página 203
- “Convenciones utilizadas en cada descripción de tipo de datos de estructura” en la página 204

Resumen de los tipos de datos de estructura utilizados en las llamadas a « MQI » o en las funciones de salida

Tabla 459. Tipos de datos de estructura utilizados en las llamadas a « MQI » o en las funciones de salida		
Estructura	Descripción	Llamadas donde se utilizan
MQACH	Cabecera de cadena de salida de API	
<u>MQAIR</u>	Registro de información de autenticación	<u>MQCONN</u>
MQAXC	Contexto de salida de API	
MQAXP	Parámetro de salida de API	

Tabla 459. Tipos de datos de estructura utilizados en las llamadas a « MQI » o en las funciones de salida (continuación)

Estructura	Descripción	Llamadas donde se utilizan
MQBMHO	Opciones de almacenamiento intermedio a manejador de mensajes	MQBUFMH
MQBO	Opciones de inicio	MQBEGIN
MQCBD	Descriptor de devolución de llamada	MQCB
MQCBO	Opciones de creación de paquete	mqCreateBag
MQCHARV	Serie de longitud variable	MQINQMP
MQCNO	Opciones de conexión	MQCONN
MQCSP	Parámetros de seguridad	MQCONN
MQCTLO	Opciones de devolución de llamada	MQCTL
MQDMPO	Suprimir opciones de propiedad de mensaje	MQDLTMP
MQGMO	Opciones para obtener mensaje	MQGet
MQIMPO	Consultar opciones de propiedad de mensaje	MQINQMP
MQMD	Descriptor de mensaje	MQBUFMH , MQMHBUF , MQCB , MQGET , MQPUT , MQPUT1
MQMHBO	Manejador de mensajes para opciones de almacenamiento intermedio	MQMHBUF
MQOD	Descriptor de objetos	MQOPEN , MQPUT1
MQOR	Registro de objeto	MQOPEN , MQPUT1
MQPD	Descriptor de propiedad	MQSETMP
MQPMO	Opciones para transferir mensaje	MQPUT , MQPUT1
MQPMR	Registro de transferencia de mensaje	MQPUT , MQPUT1
MQRR	Registro de respuestas	MQOPEN , MQPUT , MQPUT1
MQSCO	TLS opciones de configuración	MQCONN
MQSD	Descriptor de suscripción	MQSUB
MQSMPO	Establecer opción de propiedad de mensaje	MQSETMP
MQSRO	Opciones de solicitud de suscripción	MQSUBRQ
MQSTS	Estructura de informes de estado	MQSTAT

Resumen de los tipos de datos de estructura utilizados en los datos de mensaje

Tabla 460. Tipos de datos de estructura utilizados en datos de mensaje	
Estructura	Descripción
MQCIH	Cabecera de información de CICS
MQCFH	Cabecera PCF
MQEPH	Cabecera PCF incluida
MQDH	Cabecera de distribución
MQDLH	Cabecera de mensaje no entregado (mensaje no entregado)
MQIIH	Cabecera de información de IMS
MQMDE	Extensión del descriptor de mensaje
MQRFH	Cabecera de reglas y formato
MQRFH2	Cabecera de reglas y formato 2
MQRMH	Cabecera de mensaje de referencia
MQTM	Mensaje de desencadenante
MQTMC2	Mensaje desencadenante (formato de caracteres 2)
MQWIH	Cabecera de información de trabajo
MQXQH	Cabecera de cola de transmisión

Nota: La estructura MQDXP (parámetro de salida de conversión de datos) se describe en [“salida de conversión de datos”](#) en la [página 915](#), junto con las llamadas de conversión de datos asociadas.

Normas para mapear las estructuras de MQI de forma coherente

Los lenguajes de programación difieren en cuanto al nivel de compatibilidad que ofrecen con las estructuras, por lo que se adoptan ciertas reglas y convenciones para representar las estructuras MQI de forma coherente en cada lenguaje de programación:

- Las estructuras deben estar alineadas en sus límites naturales.
 - La mayoría de las estructuras de tipo « MQI » requieren una alineación de 4 bytes.
 - En IBM i, las estructuras que contienen punteros requieren una alineación de 16 bytes; estos son: MQCNO, MQOD, MQPMO.
- Cada campo de una estructura debe estar alineado en su límite natural.
 - Los campos con tipos de datos que equivalen a MQLONG deben estar alineados en límites de 4 bytes.
 - Los campos con tipos de datos que equivalen a MQPTR deben estar alineados en límites de 16 bytes en IBM i y en límites de 4 bytes en otros entornos.
 - Otros campos están alineados en límites de 1 byte.
- La longitud de una estructura debe ser un múltiplo de su alineación de límite.
 - La mayoría de las estructuras de tipo « MQI » tienen longitudes que son múltiplos de 4 bytes.
 - En IBM i, las estructuras que contienen punteros tienen longitudes que son múltiplos de 16 bytes.
- Cuando sea necesario, se deben añadir campos o bytes de relleno para garantizar el cumplimiento de las reglas anteriores.

Convenciones utilizadas en cada descripción de tipo de datos de estructura

La descripción de cada tipo de datos de estructura incluye:

- Una visión general de la finalidad y el uso de la estructura
- Descripciones de los campos de la estructura, en un formato independiente del lenguaje de programación
- Ejemplos de cómo se declara la estructura en cada uno de los lenguajes de programación soportados

La descripción de cada tipo de datos de estructura contiene las secciones siguientes:

Nombre de estructura

El nombre de la estructura, seguido de un resumen de los campos de la estructura.

Visión general

Una breve descripción del propósito y uso de la estructura.

Campos

Descripciones de los campos. Para cada campo, el nombre del campo va seguido de su tipo de datos elemental entre paréntesis (). En el texto, los nombres de campo se muestran utilizando un tipo de letra cursiva; por ejemplo, *Version*.

También hay una descripción de la finalidad del campo, junto con una lista de los valores que el campo puede tomar. Los nombres de constantes se muestran en mayúsculas; por ejemplo, MQGMO_STRUC_ID. Un conjunto de constantes que tienen el mismo prefijo se muestra utilizando el carácter *, por ejemplo: MQIA_ *.

En las descripciones de los campos, se utilizan los términos siguientes:

entrada

Proporcione información en el campo cuando realice una llamada.

salida

El gestor de colas devuelve información en el campo cuando la llamada se completa o falla.

entrada/salida

Proporcione información en el campo cuando realice una llamada y el gestor de colas cambie la información cuando la llamada se complete o falle.

Valores iniciales

Una tabla que muestra los valores iniciales de cada campo en los archivos de definición de datos que se incluyen con el programa « MQI ».

Declaración C

Declaración típica de la estructura en C.

COBOL declaración

Declaración típica de la estructura en COBOL.

PL/I declaración

Declaración típica de la estructura en PL/I.

Declaración High Level Assembler

Declaración típica de la estructura en lenguaje ensamblador System/390 .

Declaración de Visual Basic

Declaración típica de la estructura en Visual Basic.

Programación C

Información que te ayudará a utilizar la biblioteca « MQI » desde el lenguaje de programación C.

- “Archivos de cabecera” en la [página 205](#)
- “Funciones” en la [página 205](#)
- “Parámetros con tipo de datos no definido” en la [página 205](#)
- “Tipos de datos” en la [página 206](#)
- “Manipulación de series binarias” en la [página 206](#)

- “Manipulación de series de caracteres” en la página 206
- “Valores iniciales para estructuras” en la página 206
- “Valores iniciales para estructuras dinámicas” en la página 207
- “Utilizar desde C++” en la página 207
- “Convenios de notación” en la página 207

Archivos de cabecera

Tabla 461. Archivos de cabecera C	
Archivo	Contenido
CMQC	Prototipos de funciones, tipos de datos y constantes con nombre para el programa principal MQI
CMQXC	Prototipos de función, tipos de datos y constantes con nombre para la salida de conversión de datos
CMQEC	Prototipos de funciones, tipos de datos y constantes con nombre para la función principal `MQI`, la salida de conversión de datos y la estructura de puntos de entrada de interfaz (CMQEC incluye CMQXC y CMQC)
CMQSTRC	Funciones que convierten definiciones de constantes de « MQI » en su equivalente textual.  Atención:  Aplicable a z/OS desde IBM MQ 9.1. Los programas que utilizan este archivo de cabecera deben compilarse con la opción de compilador LONGNAME.

Para mejorar la portabilidad de aplicaciones, codifique el nombre del archivo de cabecera en minúsculas en la directiva de preprocesador `#include`:

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

Funciones

No es necesario especificar todos los parámetros que se pasan por dirección cada vez que se invoca una función.

- Pasar parámetros que son *de sólo entrada* y de tipo MQHCONN, MQHOBJ o MQLONG por valor.
- Pase todos los demás parámetros por dirección.

Cuando no sea necesario un parámetro determinado, utilice un puntero nulo como parámetro en la invocación de la función, en lugar de la dirección de los datos del parámetro. Los parámetros para los cuales esto es posible se identifican en las descripciones de llamada.

No se devuelve ningún parámetro como valor de la función; en terminología C, esto significa que todas las funciones devuelven `void`.

Los atributos de la función se definen mediante la variable de macro MQENTRY; el valor de esta variable de macro depende del entorno.

Parámetros con tipo de datos no definido

El parámetro **Buffer** en las funciones MQGET, MQPUT y MQPUT1 tiene un tipo de datos no definido. Este parámetro se utiliza para enviar y recibir los datos de mensaje de la aplicación.

Los parámetros de este tipo se muestran en los ejemplos de C como matrices de MQBYTE. Puede declarar los parámetros de esta forma, pero normalmente es más conveniente declararlos como la estructura particular que describe el diseño de los datos en el mensaje. Declare el parámetro de función

real como puntero a vacío y especifique la dirección de cualquier tipo de datos como parámetro en la invocación de la función.

Tipos de datos

Defina todos los tipos de datos utilizando la sentencia `typedef` de C. Para cada tipo de datos, defina también el tipo de datos de puntero correspondiente. El nombre del tipo de datos de puntero es el nombre del tipo de datos elemental o de estructura prefijado con la letra P para denotar un puntero. Defina los atributos del puntero utilizando la variable de macro `MQPOINTER`; el valor de esta variable de macro depende del entorno. A continuación se muestra cómo declarar tipos de datos de puntero:

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

Manipulación de series binarias

Declare series de datos binarios como uno de los tipos de datos `MQBYTEN`.

Siempre que copie, compare o establezca campos de este tipo, utilice las funciones C **`memcpy`**, **`memcmp`** o **`memset`**; por ejemplo:

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

MQMD MyMsgDesc;

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

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

No utilice las funciones de serie **`strcpy`**, **`strcmp`**, **`strncpy`** o **`strncmp`**, porque no funcionan correctamente para los datos declarados con los tipos de datos `MQBYTEN`.

Manipulación de series de caracteres

Cuando el gestor de colas devuelve los datos de caracteres a la aplicación, el gestor de colas siempre rellena los datos de caracteres con espacios en blanco hasta la longitud definida del campo. El gestor de colas *no* devuelve series terminadas en nulo.

Por lo tanto, al copiar, comparar o concatenar dichas series, utilice las funciones de serie **`strncpy`**, **`strncmp`** o **`strncat`**.

No utilice las funciones de serie que requieren que la serie termine con un valor nulo (**`strcpy`**, **`strcmp`**, **`strcat`**). Además, no utilice la función **`strlen`** para determinar la longitud de la serie; utilice en su lugar la función **`sizeof`** para determinar la longitud del campo.

Valores iniciales para estructuras

Los archivos de cabecera definen diversas variables de macro que puede utilizar para proporcionar valores iniciales para las estructuras de MQ al declarar instancias de dichas estructuras.

Estas variables de macro tienen nombre con el formato `MQxxx_DEFAULT`, donde `MQxxx` representa el nombre de la estructura. Se utilizan de la siguiente manera:

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

En el caso de algunos campos de caracteres (por ejemplo, los *StrucId* campos que aparecen en la mayoría de las estructuras, o el *Format* campo que aparece en `MQMD`), el documento « MQI » define los

valores concretos que son válidos. Para cada uno de los valores válidos, se proporcionan *dos* variables de macro:

- Una variable de macro define el valor como una serie con una longitud, excluyendo las coincidencias nulas implícitas, exactamente la longitud definida del campo. Por ejemplo, para el campo *Format* en MQMD se proporciona la siguiente variable de macro (↵ representa un único carácter en blanco):

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

Utilice este formulario con las funciones `memcpy` y `memcmp`.

- La otra variable de macro define el valor como una matriz de caracteres; el nombre de esta variable de macro es el nombre del formato de serie con el sufijo `_ARRAY`. Por ejemplo:

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

Utilice este formulario para inicializar el campo cuando declare una instancia de la estructura con valores distintos de los proporcionados por la variable de macro `MQMD_DEFAULT`. (Esto no siempre es necesario; en algunos entornos puede utilizar el formato de serie del valor en ambas situaciones. Sin embargo, puede utilizar el formato de matriz para las declaraciones, porque esto es necesario para la compatibilidad con el lenguaje de programación C++.)

Valores iniciales para estructuras dinámicas

Cuando se necesita un número variable de instancias de una estructura, las instancias normalmente se crean en el almacenamiento principal obtenido dinámicamente utilizando las funciones `calloc` o `malloc`. Para inicializar los campos en dichas estructuras, tenga en cuenta la técnica siguiente:

1. Declare una instancia de la estructura utilizando la variable de macro `MQxxx_DEFAULT` adecuada para inicializar la estructura. Esta instancia se convierte en el modelo para otras instancias:

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

Las palabras clave `static` o `auto` se pueden codificar en la declaración para proporcionar la duración estática o dinámica de la instancia de modelo, según sea necesario.

2. Utilice las funciones `calloc` o `malloc` para obtener almacenamiento para una instancia dinámica de la estructura:

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

3. Utilice la función `memcpy` para copiar la instancia de modelo en la instancia dinámica:

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

Utilizar desde C++

Para el lenguaje de programación C++, los archivos de cabecera contienen las siguientes sentencias adicionales que sólo se incluyen cuando se utiliza un compilador C++:

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

Convenios de notación

Esta información muestra cómo invocar las funciones y declarar parámetros.

En algunos casos, los parámetros son matrices con un tamaño que no es fijo. Para estos, se utiliza una *n* minúscula para representar una constante numérica. Cuando codifique la declaración para ese parámetro, sustituya el *n* por el valor numérico necesario.

COBOL programación

Información que te ayudará a utilizar el servicio « MQI » desde el lenguaje de programación « COBOL ».

- [“Archivos de copia” en la página 208](#)
- [“Estructuras” en la página 209](#)
- [“Punteros” en la página 210](#)
- [“Constantes con nombre” en la página 210](#)
- [“Convenciones de notación” en la página 210](#)

Archivos de copia

Se incluyen varios archivos COPY para ayudarte a escribir programas de aplicación COBOL es que utilicen el lenguaje de programación de alto nivel (MQI). Hay dos archivos que contienen constantes con nombre y dos archivos para cada una de las estructuras.

Cada estructura se proporciona en dos formatos: uno con valores iniciales y otro sin ellos:

- Utilice las estructuras con valores iniciales en la SECCIÓN DE ALMACENAMIENTO DE TRABAJO de un programa COBOL; se encuentran en archivos COPY con nombres que terminan en la letra V (de Values, valores).
- Utilice las estructuras sin valores iniciales en la SECCIÓN LINKAGE de un programa COBOL; se encuentran en archivos COPY cuyos nombres terminan con la letra L (de Linkage).

Los archivos COPY se resumen en la siguiente tabla. No todos los archivos enumerados están disponibles en todos los entornos.

Tabla 462. Archivos de copia COBOL		
Archivo (con valores iniciales)	Archivo (sin valores iniciales)	Contenido
CMQAIRV	CMQAIRL	Registro de información de autenticación
CMQBOV	CMQBOL	Comience la estructura de opciones
CMQCIHV	CMQCIHL	CICS Estructura del encabezado de información
CMQCN OV	CMQCNOL	Estructura de opciones de conexión
CMQD HV	CMQDHL	Estructura del colector de distribución
CMQDL HV	CMQDLHL	Estructura del encabezado de carta muerta
CMQDXPV	CMQDXPL	Estructura del parámetro de salida de conversión de datos
CMQGM OV	CMQGMOL	Obtener estructura de opciones de mensajes
CMQIIHV	CMQIIHL	IMS Estructura del encabezado de información
CMQMDV	CMQMDL	Estructura del descriptor de mensajes
CMQMDEV	CMQMDEL	Estructura de extensión del descriptor de mensaje
CMQMD1V	CMQMD1L	Estructura del descriptor de mensajes, versión 1
CMQODV	CMQODL	Estructura del descriptor de objetos
CMQORV	CMQORL	Estructura del registro de objetos
CMQPM OV	CMQPMOL	Estructura de opciones de mensajes

Tabla 462. Archivos de copia COBOL (continuación)

Archivo (con valores iniciales)	Archivo (sin valores iniciales)	Contenido
CMQRFHV	CMQRFHL	Reglas y estructura del encabezado de formato
CMQRFH2V	CMQRFH2L	Reglas y estructura de formato del encabezado, versión 2
CMQRMHV	CMQRMHL	Estructura del encabezado del mensaje de referencia
CMQRRV	CMQRRL	Estructura del registro de respuesta
CMQSCOV	CMQSCOL	TLS opciones de configuración
CMQTMV	CMQTML	Estructura del mensaje desencadenante
CMQTMCV	CMQTMCL	Estructura del mensaje de activación (formato de caracteres)
CMQTM2V	CMQTM2L	Estructura del mensaje de activación (formato de caracteres) versión 2
CMQWIHV	CMQWIHL	Estructura del encabezado de la información del trabajo
CMQXQHV	CMQXQHL	Estructura del encabezado de la cola de transmisión
CMQV	-	Constantes con nombre para el archivo « MQI »
CMQXV	-	Constantes con nombre para la salida de conversión de datos
CMQMD2V	CMQMD2L	Estructura del descriptor de mensajes, versión 2

Estructuras

En el archivo COPY, cada declaración de estructura comienza con un elemento level-10; esto permite declarar varias instancias de la estructura codificando la declaración level-01 y utilizando luego la instrucción COPY para copiar el resto de la declaración de estructura. Para hacer referencia a la instancia adecuada, utilice la palabra clave IN :

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

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

Alinee las estructuras en los límites adecuados. Si utiliza la instrucción COPY para incluir una estructura después de un elemento que no es el elemento level-01, asegúrese de que la estructura comience en la posición adecuada desde el inicio del elemento level-01. La mayoría de las estructuras de tipo « MQI » requieren una alineación de 4 bytes; las excepciones son MQCNO, MQOD y MQPMO, que requieren una alineación de 16 bytes en IBM i.

En esta sección, los nombres de los campos de las estructuras se muestran sin prefijo. En COBOL, los nombres de los campos tienen como prefijo el nombre de la estructura seguido de un guión. Sin embargo, si el nombre de la estructura termina con un dígito numérico, lo que indica que la estructura es una segunda versión o posterior de la estructura original, el dígito numérico se omite del prefijo. Los nombres de los campos en COBOL se muestran en mayúsculas (aunque se pueden utilizar minúsculas o mayúsculas y minúsculas si es necesario). Por ejemplo, el campo *MsgType* descrito en “MQMD - Descriptor de mensaje” en la página 389 se convierte en MQMD-MSGTYPE en COBOL.

Las estructuras con sufijo V se declaran con valores iniciales para todos los campos; solo es necesario establecer aquellos campos en los que se desee un valor diferente al valor inicial proporcionado.

Punteros

Algunas estructuras deben abordar datos opcionales que pueden ser discontinuos con respecto a la estructura, como los registros MQOR y MQRR abordados por la estructura MQOD.

Para tratar estos datos opcionales, las estructuras contienen campos que se declaran con el tipo de datos puntero. Sin embargo, COBOL no admite el tipo de datos puntero en todos los entornos. Por este motivo, los datos opcionales también pueden abordarse utilizando campos que contengan el desplazamiento de los datos desde el inicio de la estructura.

Si desea transferir una aplicación entre entornos, compruebe si el tipo de datos del puntero está disponible en todos los entornos previstos. Si no es así, la aplicación debe abordar los datos opcionales utilizando los campos de desplazamiento en lugar de los campos de puntero.

En aquellos entornos en los que no se admiten punteros, declare los campos de puntero como cadenas de bytes de la longitud adecuada, con el valor inicial como cadena de bytes totalmente nula. No modifique este valor inicial si está utilizando los campos de compensación.

Constantes con nombre

En esta información, se muestran los nombres de las constantes que contienen el carácter de subrayado (_) como parte del nombre. En COBOL, utilice el guión (-) en lugar del guión bajo.

Las constantes que tienen valores de cadena de caracteres utilizan el signo de comilla simple como delimitador de cadena ('). En algunos entornos, es posible que tenga que especificar una opción de compilador adecuada para que el compilador acepte el signo de comilla simple como delimitador de cadena en lugar del signo de comilla doble.

Las constantes nombradas se declaran en los archivos COPY como elementos de level-10. Para utilizar las constantes, declare explícitamente el elemento level-01 y, a continuación, utilice la instrucción COPY para copiar las declaraciones de las constantes:

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

El método anterior hace que las constantes ocupen espacio de almacenamiento en el programa, incluso si no se hace referencia a ellas. Si incluye las constantes en muchos programas separados dentro de la misma unidad de ejecución, existirán múltiples copias de las constantes, lo que consumirá innecesariamente la memoria principal. Evite este efecto utilizando una de las siguientes técnicas:

- Añade la cláusula GLOBAL a la declaración `level-01` :

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

Esto asigna almacenamiento solo para un conjunto de constantes dentro de la unidad de ejecución. Sin embargo, cualquier programa dentro de la unidad de ejecución puede hacer referencia a las constantes, no solo el programa que contiene la declaración `level-01` .

Nota: La cláusula GLOBAL no es compatible con todos los entornos.

- Copie manualmente en cada programa solo aquellas constantes a las que hace referencia ese programa. No utilice la instrucción COPY para copiar todas las constantes en el programa.

Convenciones de notación

Las secciones anteriores de este tema muestran cómo invocar llamadas y declarar parámetros. En algunos casos, los parámetros son tablas o cadenas de caracteres cuyo tamaño no es fijo. Para estos, se utiliza una n minúscula para representar una constante numérica. Cuando codifique la declaración para ese parámetro, sustituya la n por el valor numérico requerido.

Programación de High Level Assembler

Información que te ayudará a utilizar el lenguaje de programación de ensamblador « MQI » de « System/390 ».

- [“Macros” en la página 211](#)
- [“Estructuras” en la página 211](#)
- [“Macro MQVERA” en la página 212](#)
- [“Convenios de notación” en la página 212](#)

Macros

Hay dos macros para constantes con nombre y una macro para cada una de las estructuras. Estos archivos se resumen en la tabla siguiente.

Tabla 463. Macros de Assembler	
Archivo	Contenido
CMQA	Constantes con nombre (asignaciones) para la función principal ` MQI `
CMQCIHA	Estructura de cabecera de información de CICS
CMQCNOA	Estructura de opciones de conexión
CMQDLHA	Estructura de cabecera de mensaje no entregado
CMQDXPA	Estructura de parámetros de salida de conversión de datos
CMQGMOA	Estructura de opciones de obtención de mensajes
CMQIIHA	Estructura de cabecera de información de IMS
CMQMDA	Estructura del descriptor de mensaje
CMQMDEA	Estructura de extensión de descriptor de mensaje
CMQODA	Estructura de descriptor de objeto
CMQPMOA	Estructura de opciones de colocación de mensaje
CMQRFHA	Reglas y estructura de cabecera de formato
CMQRFH2A	Reglas y estructura de cabecera de formato versión 2
CMQRMHA	Estructura de cabecera de mensaje de referencia
CMQTMA	Estructura de mensajes desencadenantes
CMQTMCA	Estructura de mensaje desencadenante (formato de caracteres) versión 2
CMQVERA	Control de versión de estructura
CMQWIHA	Estructura de cabecera de información de trabajo
CMQXA	Constantes con nombre para la salida de conversión de datos
CMQXPA	Estructura de parámetros de salida cruzada de API
CMQXQHA	Estructura de cabecera de cola de transmisión

Estructuras

Las estructuras son generadas por macros que tienen varios parámetros para controlar la acción de la macro. Consulte [“Estructuras” en la página 212](#)

Macro MQVERA

Esta macro le permite establecer el valor predeterminado que se utilizará para el parámetro DCLVER en las macros de estructura.

El valor especificado por CMQVERA lo utiliza la macro de estructura sólo si omite el parámetro DCLVER de la invocación de la macro de estructura. El valor predeterminado se establece codificando la macro CMQVERA con el parámetro DCLVER :

DCLVER=ACTUAL

La versión predeterminada se establece en la versión actual (más reciente).

DCLVER=ESPECIFICADO

La versión predeterminada se establece en la versión especificada por el parámetro VERSION .

Debe especificar el parámetro **DCLVER** y el valor debe estar en mayúsculas. El valor establecido por CMQVERA sigue siendo el valor predeterminado hasta la siguiente invocación de CMQVERA o el final del ensamblaje. Si omite CMQVERA, el valor predeterminado es DCLVER=CURRENT.

Convenios de notación

Otros temas muestran cómo invocar las llamadas y declarar parámetros. En algunos casos, los parámetros son matrices o series de caracteres con un tamaño que no está fijado para el cual, se utiliza una n minúscula para representar una constante numérica. Cuando codifique la declaración para ese parámetro, sustituya el n por el valor numérico necesario.

Estructuras

Las estructuras son generadas por macros que tienen varios parámetros para controlar la acción de la macro.

Nota: De vez en cuando se introducen nuevas versiones de las estructuras IBM MQ . Los campos adicionales en una nueva versión pueden hacer que una estructura que anteriormente era menor que 256 bytes sea mayor que 256 bytes. Debido a esto, escriba las instrucciones del ensamblador que están pensadas para copiar una estructura IBM MQ , o para establecer una estructura IBM MQ en nulos, para trabajar correctamente con estructuras que pueden tener más de 256 bytes. De forma alternativa, utilice el parámetro de macro DCLVER o la macro CMQVERA con el parámetro VERSION para declarar una versión específica de la estructura.

- [“Especificación del nombre de la estructura” en la página 212](#)
- [“Especificación del formato de la estructura” en la página 213](#)
- [“Control de la versión de la estructura” en la página 213](#)
- [“Declaración de una estructura incluida en otra” en la página 213](#)
- [“Especificación del valor inicial de un campo” en la página 213](#)
- [“Control del listado” en la página 214](#)

Especificación del nombre de la estructura

Para declarar más de una instancia de una estructura, la macro prefija el nombre de cada campo de la estructura con una serie especificada por el usuario y un carácter de subrayado.

La serie utilizada es la etiqueta especificada en la invocación de la macro. Si no se especifica ninguna etiqueta, se utiliza el nombre de la estructura para construir el prefijo:

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

Las declaraciones de estructura que se muestran en esta sección utilizan el prefijo predeterminado.

Especificación del formato de la estructura

La macro puede generar declaraciones de estructura en uno de dos formatos, controlados por el parámetro DSECT :

DSECT=YES

Se utiliza una instrucción DSECT de ensamblador para iniciar una nueva sección de datos; la definición de estructura sigue inmediatamente a la sentencia DSECT . La etiqueta en la invocación de macro se usa como nombre de la sección de datos; si no se especifica ninguna etiqueta, se usará el nombre de la estructura.

DSECT=NO

Las instrucciones DC del ensamblador se utilizan para definir la estructura en la posición actual de la rutina. Los campos se inicializan con valores, que se pueden especificar codificando los parámetros relevantes en la invocación de macro. Los campos para los que no se especifican valores en la invocación de la macro se inicializan con valores predeterminados.

El valor especificado debe estar en mayúsculas. Si no se especifica el parámetro DSECT , se presupone DSECT = NO .

Control de la versión de la estructura

De forma predeterminada, las macros siempre declaran la versión más reciente de cada estructura.

Aunque puede utilizar el parámetro de macro VERSION para especificar un valor para el campo *Version* en la estructura, ese parámetro define el valor inicial para el campo *Version* y no controla la versión de la estructura declarada realmente. Para controlar la versión de la estructura que se declara, utilice el parámetro DCLVER :

DCLVER=ACTUAL

La versión declarada es la versión actual (más reciente).

DCLVER=ESPECIFICADO

La versión declarada es la versión especificada por el parámetro VERSION . Si omite el parámetro VERSION , el valor predeterminado es la versión 1.

Si especifica el parámetro VERSION , el valor debe ser una constante numérica autodefinida o la constante con nombre para la versión necesaria (por ejemplo, MQCNO_VERSION_3). Si especifica algún otro valor, la estructura se declara como si se hubiera especificado DCLVER=CURRENT , incluso si el valor de VERSION se resuelve en un valor válido.

El valor especificado debe estar en mayúsculas. Si omite el parámetro DCLVER , el valor utilizado se toma de la variable de macro global MQDCLVER . Puede establecer esta variable utilizando la macro CMQVERA.

Declaración de una estructura incluida en otra

Para declarar una estructura como componente de otra estructura, utilice el parámetro ANIDADO :

NESTED=SÍ

La declaración de estructura está anidada dentro de otra.

ANIDADO=NO

La declaración de estructura no está anidada dentro de otra.

El valor especificado debe estar en mayúsculas. Si omite el parámetro ANIDADO , se presupone NESTED=NO .

Especificación del valor inicial de un campo

Especifique el valor que se utilizará para inicializar un campo en una estructura codificando el nombre de ese campo (sin el prefijo) como parámetro en la invocación de la macro, acompañado del valor necesario.

Por ejemplo, para declarar una estructura de descriptor de mensaje con el campo *MsgType* inicializado con MQMT_REQUEST y el campo *ReplyToQ* inicializado con la serie "MY_REPLY_TO_QUEUE", utilice lo siguiente:

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

Si especifica una constante con nombre (equate) como un valor en la invocación de la macro, utilice la macro CMQA para definir la constante con nombre. No encierre los valores de serie de caracteres entre comillas simples.

Control del listado

Controle el aspecto de la declaración de estructura en el listado del ensamblador utilizando el parámetro LIST :

LIST=YES

La declaración de estructura aparece en el listado del ensamblador.

LIST=NO

La declaración de estructura no aparece en el listado del ensamblador.

El valor especificado debe estar en mayúsculas. Si omite el parámetro LIST , se presupone LIST = NO .

MQAIR - Registro de información de autenticación

La estructura MQAIR permite que una aplicación que se ejecuta como IBM MQ MQI client especifique información sobre un autenticador que se utilizará para la conexión del cliente. La estructura es un parámetro de entrada en la llamada MQCONN.

Disponibilidad

La estructura MQAIR está disponible para los siguientes clientes:

-  AIX
-  Linux
-  Windows

Conjunto de caracteres y codificación

Los datos en MQAIR deben estar en el juego de caracteres y la codificación del gestor de colas local; estos se proporcionan mediante el atributo del **CodedCharSetId** gestor de colas y MQENC_NATIVE.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 464. Campos en MQAIR		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>StrucId</u> (identificador de estructura)	MQAIR_STRUC_ID	'AIR-'
<u>Versión</u> (número de versión de la estructura)	MQAIR_VERSION_1	1
<u>AuthInfoType</u> (tipo de información de autenticación)	MQAIT_CRL_LDAP	1
<u>AuthInfoConnName</u> (nombre de conexión del servidor CRL de LDAP)	Ninguna	Cadena nula o espacios en blanco

Tabla 464. Campos en MQAIR (continuación)

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>LDAPUserNamePtr</u> (dirección de LDAP nombre de usuario)	Ninguna	Puntero nulo o bytes nulos
<u>LDAPUserNameOffset</u> (desplazamiento del nombre de usuario LDAP desde el inicio de MQSCO)	Ninguna	0
<u>LDAPUserNameLength</u> (longitud del nombre de usuario de LDAP)	Ninguna	0
<u>LDAPPassword</u> (contraseña para acceder al servidor LDAP)	Ninguna	Cadena nula o espacios en blanco
Nota: Los campos restantes se ignoran si la versión es inferior a MQAIR_VERSION_2.		
<u>OCSPResponderURL</u> (URL, donde se puede contactar con el respondedor OCSP)	Ninguna	Cadena nula o espacios en blanco
Notas: 1. El símbolo ~ representa un único carácter en blanco. 2. En el lenguaje de programación C, la variable macroMQAIR_DEFAULT contiene los valores que se enumeran en la tabla. Úselo de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQAIR MyAIR = {MQAIR_DEFAULT};</pre>		

Declaraciones lingüísticas

Declaración C para MQAIR

```
typedef struct tagMQAIR MQAIR;
struct tagMQAIR {
    MQCHAR4      StrucId;           /* Structure identifier */
    MQLONG       Version;           /* Structure version number */
    MQLONG       AuthInfoType;      /* Type of authentication
                                     information */
    MQCHAR264    AuthInfoConnName; /* Connection name of CRL LDAP
                                     server */
    PMQCHAR      LDAPUserNamePtr;   /* Address of LDAP user name */
    MQLONG       LDAPUserNameOffset; /* Offset of LDAP user name from start
                                     of MQAIR structure */
    MQLONG       LDAPUserNameLength; /* Length of LDAP user name */
    MQCHAR32     LDAPPassword;      /* Password to access LDAP server */
    MQCHAR256    OCSPResponderURL; /* URL of OCSP responder */
};
```

COBOL Declaración para MQAIR

```
**  MQAIR structure
10 MQAIR.
**  Structure identifier
15 MQAIR-STRUCID          PIC X(4).
**  Structure version number
15 MQAIR-VERSION          PIC S9(9) BINARY.
**  Type of authentication information
15 MQAIR-AUTHINFOTYPE     PIC S9(9) BINARY.
**  Connection name of CRL LDAP server
15 MQAIR-AUTHINFOCONNNAME PIC X(264).
**  Address of LDAP user name
15 MQAIR-LDAPUSERNAMEPTR  POINTER.
**  Offset of LDAP user name from start of MQAIR structure
15 MQAIR-LDAPUSERNAMEOFFSET PIC S9(9) BINARY.
```

```

**      Length of LDAP user name
15 MQAIR-LDAPUSERNAMELENGTH PIC S9(9) BINARY.
**      Password to access LDAP server
15 MQAIR-LDAPPASSWORD          PIC X(32).
**      URL of OSCP responder
15 MQAIR-OCSPRESPONDERURL      PIC X(256).

```

Declaración de Visual Basic para MQAIR

```

Type MQAIR
  StrucId          As String*4  'Structure identifier'
  Version          As Long      'Structure version number'
  AuthInfoType     As Long      'Type of authentication information'
  AuthInfoConnName As String*264 'Connection name of CRL LDAP server'
  LDAPUserNamePtr  As MQPTR     'Address of LDAP user name'
  LDAPUserNameOffset As Long    'Offset of LDAP user name from start'
                        'of MQAIR structure'
  LDAPUserNameLength As Long    'Length of LDAP user name'
  LDAPPASSWORD     As String*32 'Password to access LDAP server'
End Type

```

Detalles de campo para MQAIR

La estructura MQAIR permite que una aplicación que se ejecuta como IBM MQ MQI client especifique información sobre un autenticador que se utilizará para la conexión del cliente. La estructura es un parámetro de entrada en la llamada MQCONN.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 465. Campos en MQAIR		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>StrucId</u> (identificador de estructura)	MQAIR_STRUC_ID	'AIR-'
<u>Versión</u> (número de versión de la estructura)	MQAIR_VERSION_1	1
<u>AuthInfoType</u> (tipo de información de autenticación)	MQAIT_CRL_LDAP	1
<u>AuthInfoConnName</u> (nombre de conexión del servidor CRL de LDAP)	Ninguna	Cadena nula o espacios en blanco
<u>LDAPUserNamePtr</u> (dirección de LDAP nombre de usuario)	Ninguna	Puntero nulo o bytes nulos
<u>LDAPUserNameOffset</u> (desplazamiento del nombre de usuario LDAP desde el inicio de MQSCO)	Ninguna	0
<u>LDAPUserNameLength</u> (longitud del nombre de usuario de LDAP)	Ninguna	0
<u>LDAPPASSWORD</u> (contraseña para acceder al servidor LDAP)	Ninguna	Cadena nula o espacios en blanco
Nota: Los campos restantes se ignoran si la versión es inferior a MQAIR_VERSION_2.		
<u>OCSPResponderURL</u> (URL, donde se puede contactar con el respondedor OCSP)	Ninguna	Cadena nula o espacios en blanco

Tabla 465. Campos en MQAIR (continuación)

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
Notas: 1. El símbolo ~ representa un único carácter en blanco. 2. En el lenguaje de programación C, la variable macroMQAIR_DEFAULT contiene los valores que se enumeran en la tabla. Úselo de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQAIR MyAIR = {MQAIR_DEFAULT};</pre>		

StrucId (MQCHAR4)

Este es el identificador de estructura del registro de información de autenticación. Siempre es un campo de entrada. Su valor es MQAIR_STRUC_ID.

El valor debe ser:

MQAIR_STRUC_ID

Identificador del registro de información de autenticación.

Para el lenguaje de programación C, también se define la constante MQAIR_STRUC_ID_ARRAY. Tiene el mismo valor que MQAIR_STRUC_ID, pero es una matriz de caracteres en lugar de una cadena.

Versión (MQLONG)

Este es el número de versión de la estructura del registro de información de autenticación. Siempre es un campo de entrada.

El valor debe ser uno de los siguientes:

MQAIR_VERSION_1

Version-1 registro de información de autenticación.

Este es el valor inicial de este campo.

MQAIR_VERSION_2

Version-2 registro de información de autenticación.

La siguiente constante especifica el número de versión de la versión actual:

MQAIR_VERSIÓN_ACTUAL

Versión actual del registro de información de autenticación.

AuthInfoType (MQLONG)

Este es el tipo de información de autenticación que contiene el registro.

El valor puede ser uno de los dos parámetros siguientes:

MQAIT_CRL_LDAP

Comprobación de revocación de certificados mediante un servidor LDAP.

MQAIT_OCSP

Comprobación de revocación de certificados mediante OCSP.

Si el valor no es válido, la llamada falla con el código de motivo MQRC_AUTH_INFO_TYPE_ERROR.

Este es un campo de entrada. El valor inicial de este campo es MQAIT_CRL_LDAP.

AuthInfoConnName (MQCHAR264)

Se trata del nombre de host o la dirección de red de un host en el que se ejecuta el servidor LDAP. A continuación puede aparecer un número de puerto opcional, entre paréntesis. El número de puerto predeterminado es 389.

Si el valor es más corto que la longitud del campo, termine el valor con un carácter nulo o rellénelo con espacios en blanco hasta alcanzar la longitud del campo. Si el valor no es válido, la llamada falla con el código de motivo MQRC_AUTH_INFO_CONN_NAME_ERROR.

Este es un campo de entrada. La longitud de este campo viene dada por MQ_AUTH_INFO_CONN_NAME_LENGTH. El valor inicial de este campo es la cadena nula en C y caracteres en blanco en otros lenguajes de programación.

LDAPUserNamePtr (PMQCHAR)

Este es el nombre de usuario LDAP.

Consiste en el nombre distintivo del usuario que intenta acceder al servidor CRL LDAP. Si el valor es más corto que la longitud especificada por *LDAPUserNameLength*, termine el valor con un carácter nulo o rellénelo con espacios en blanco hasta alcanzar la longitud *LDAPUserNameLength*. El campo se ignora si *LDAPUserNameLength* es cero.

Puede proporcionar el nombre de usuario LDAP de dos maneras:

- Al utilizar el campo puntero *LDAPUserNamePtr*

En este caso, la aplicación puede declarar una cadena separada de la estructura MQAIR y establecer la dirección *LDAPUserNamePtr* de la cadena.

Considere utilizar *LDAPUserNamePtr* para lenguajes de programación que admitan el tipo de datos puntero de forma que sea portátil a diferentes entornos (por ejemplo, el lenguaje de programación C).

- Al utilizar el campo de compensación *LDAPUserNameOffset*

En este caso, la aplicación debe declarar una estructura compuesta que contenga la estructura MQSCO seguida de la matriz de registros MQAIR seguida de las cadenas de nombre de usuario LDAP, y establecer *LDAPUserNameOffset* en la compensación de la cadena de nombre adecuada desde el inicio de la estructura MQAIR. Asegúrese de que este valor sea correcto y tenga un valor que pueda acomodarse dentro de un MQLONG (el lenguaje de programación más restrictivo es COBOL, para el cual el rango válido es de -999 999 999 a +999 999 999).

Considere utilizar *LDAPUserNameOffset* para lenguajes de programación que no admiten el tipo de datos puntero, o que implementan el tipo de datos puntero de una manera que podría no ser portátil a diferentes entornos (por ejemplo, el lenguaje de programación COBOL).

Independientemente de la técnica elegida, utilice solo una de *LDAPUserNamePtr* y *LDAPUserNameOffset*; la llamada falla con el código de motivo MQRC_LDAP_USER_NAME_ERROR si ambas son distintas de cero.

Este es un campo de entrada. El valor inicial de este campo es el puntero nulo en aquellos lenguajes de programación que admiten punteros y, en los demás casos, una cadena de bytes totalmente nulos.

Nota: En plataformas donde el lenguaje de programación no admite el tipo de datos puntero, este campo se declara como una cadena de bytes de la longitud adecuada.

LDAPUserNameOffset (MQLONG)

Este es el desplazamiento en bytes del nombre de usuario LDAP desde el inicio de la estructura MQAIR.

El desplazamiento puede ser positivo o negativo. El campo se ignora si *LDAPUserNameLength* es cero.

Puede utilizar o *LDAPUserNameOffset* o *LDAPUserNamePtr* para especificar el nombre de usuario LDAP, pero no ambos; consulte la descripción del *LDAPUserNamePtr* campo para obtener más detalles.

Este es un campo de entrada. El valor inicial de este campo es 0.

LDAPUserNameLength (MQLONG)

Es la longitud en bytes del nombre de usuario LDAP al que se refiere el *LDAPUserNamePtr* campo *LDAPUserNameOffset* o.

El valor debe estar comprendido entre cero y MQ_DISTINGUISHED_NAME_LENGTH. Si el valor no es válido, la llamada falla con el código de motivo MQRC_LDAP_USER_NAME_LENGTH_ERR.

Si el servidor LDAP involucrado no requiere un nombre de usuario, establezca este campo en cero.

Este es un campo de entrada. El valor inicial de este campo es 0.

Contraseña LDAP (MQCHAR32)

Esta es la contraseña necesaria para acceder al servidor CRL LDAP. Si el valor es más corto que la longitud del campo, termine el valor con un carácter nulo o rellénelo con espacios en blanco hasta alcanzar la longitud del campo.

Si el servidor LDAP no requiere contraseña, u omite el nombre de usuario LDAP, *LDAPPASSWORD* debe ser nulo o estar en blanco. Si omite el nombre de usuario LDAP y no *LDAPPASSWORD* es nulo ni está en blanco, la llamada falla con el código de motivo MQRC_LDAP_PASSWORD_ERROR.

Este es un campo de entrada. La longitud de este campo viene determinada por MQ_LDAP_PASSWORD_LENGTH. El valor inicial de este campo es la cadena nula en C y caracteres en blanco en otros lenguajes de programación.

OCSPResponderURL (MQCHAR256)

Para una estructura MQAIR que representa los detalles de conexión de un respondedor OCSP, este campo contiene la dirección de correo electrónico (URL) en la que se puede contactar con el respondedor.

El valor de este campo es un HTTP URL. Este campo tiene prioridad sobre un campo « URL » (Aplicación de la ley) en una extensión de certificado « AuthorityInfoAccess » (AIA).

El valor se ignora a menos que ambas afirmaciones siguientes sean ciertas:

- La estructura MQAIR es la versión 2 o posterior (el campo Versión está establecido en MQAIR_VERSION_2 o superior).
- El campo « AuthInfoType » (Identificador de OCSP) está configurado como MQAIT_OCSP.

Si el campo no contiene una dirección HTTP URL con el formato correcto (y no se está ignorando), la llamada MQCONNX falla con el código de motivo MQRC_OCSP_URL_ERROR.

Este campo es sensible a las mayúsculas y minúsculas. Debe comenzar con la cadena http:// en minúsculas. El resto de URL puede distinguir entre mayúsculas y minúsculas, dependiendo de la implementación del servidor OCSP.

Este campo no está sujeto a conversión de datos.

MQBMHO: Opciones de búfer a controlador de mensajes

La estructura MQBMHO permite a las aplicaciones especificar opciones que controlan cómo se producen los identificadores de mensajes a partir de los búferes. La estructura es un parámetro de entrada en la llamada MQBUFMH.

Conjunto de caracteres y codificación

Los datos en MQBMHO deben estar en el juego de caracteres de la aplicación y en la codificación de la aplicación (MQENC_NATIVE).

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 466. Campos en MQBMHO

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
StrucId (identificador de estructura)	MQBMHO_STRUC_ID	'BMHO'
Versión (número de versión de la estructura)	MQBMHO_VERSION_1	1
Opciones (opciones que controlan la acción de MQBMHO)	MQBMHO_NINGUNO	0
Notas: 1. En el lenguaje de programación C, la variable macroMQBMHO_DEFAULT contiene los valores que se enumeran en la tabla. Úselo de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQBMHO MyBMHO = {MQBMHO_DEFAULT};</pre>		

Declaraciones lingüísticas

Declaración C para MQBMHO

```
typedef struct tagMQBMHO MQBMHO;
struct tagMQBMHO {
    MQCHAR4 StrucId;          /* Structure identifier */
    MQLONG Version;           /* Structure version number */
    MQLONG Options;           /* Options that control the action of
                               MQBUFMH */
};
```

COBOL declaración para MQBMHO

```
** MQBMHO structure
10 MQBMHO.
** Structure identifier
15 MQBMHO-STRUCID PIC X(4).
** Structure version number
15 MQBMHO-VERSION PIC S9(9) BINARY.
** Options that control the action of MQBUFMH
15 MQBMHO-OPTIONS PIC S9(9) BINARY.
```

PL/I declaración para MQBMHO

```
Dcl
1 MQBMHO based,
3 StrucId char(4), /* Structure identifier */
3 Version fixed bin(31), /* Structure version number */
3 Options fixed bin(31), /* Options that control the action
                          of MQBUFMH */
```

High Level Assembler declaración para MQBMHO

```
MQBMHO          DSECT
MQBMHO_STRUCID  DS CL4 Structure identifier
MQBMHO_VERSION  DS F   Structure version number
MQBMHO_OPTIONS  DS F   Options that control the
*               action of MQBUFMH
MQBMHO_LENGTH   EQU *-MQBMHO
MQBMHO_AREA     DS CL(MQBMHO_LENGTH)
```

Detalles de campo para MQBMHO

La estructura de opciones del búfer MQBMHO para el controlador de mensajes permite a las aplicaciones especificar opciones que controlan cómo se generan los controladores de mensajes a partir de los búferes. La estructura es un parámetro de entrada en la llamada MQBUFMH.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 467. Campos en MQBMHO		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>StrucId</u> (identificador de estructura)	MQBMHO_STRUC_ID	'BMHO '
<u>Versión</u> (número de versión de la estructura)	MQBMHO_VERSION_1	1
<u>Opciones</u> (opciones que controlan la acción de MQBMHO)	MQBMHO_NINGUNO	0
Notas: 1. En el lenguaje de programación C, la variable macroMQBMHO_DEFAULT contiene los valores que se enumeran en la tabla. Úselo de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQBMHO MyBMHO = {MQBMHO_DEFAULT};</pre>		

StrucId (MQCHAR4)

Este es el identificador de estructura de la estructura del búfer al controlador de mensajes. Siempre es un campo de entrada. Su valor es MQBMHO_STRUC_ID.

El valor debe ser:

MQBMHO_STRUC_ID

Identificador para la estructura del búfer al controlador de mensajes.

Para el lenguaje de programación C, también se define la constante MQBMHO_STRUC_ID_ARRAY. Tiene el mismo valor que MQBMHO_STRUC_ID, pero es una matriz de caracteres en lugar de una cadena.

Versión (MQLONG)

Este es el número de versión de la estructura del búfer al controlador de mensajes. Siempre es un campo de entrada.

El valor debe ser:

MQBMHO_VERSION_1

Número de versión de la estructura del búfer al identificador de mensaje.

La siguiente constante especifica el número de versión de la versión actual:

MQBMHO_VERSIÓN_ACTUAL

Versión actual de la estructura del búfer al identificador de mensaje.

Opciones (MQLONG)

Estructura del búfer al identificador de mensaje: campo Opciones.

El valor puede ser:

MQBMHO_ELIMINAR_PROPIEDADES

Las propiedades que se añaden al identificador del mensaje se eliminan del búfer. Si la llamada falla, no se eliminan ninguna propiedad.

Opción predeterminada: Si no necesita la opción descrita, utilice la siguiente opción:

MQBMHO_NINGUNO

No se ha especificado ninguna opción.

Este campo siempre es un campo de entrada. El valor inicial de este campo es MQBMHO_DELETE_PROPERTIES.

MQBNO - Opciones de equilibrio

La siguiente tabla resume los campos de la estructura.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 468. Campos en MQBNO		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>StrucId</u> (identificador de estructura)	MQBNO_STRUC_ID	'BNO¬'
<u>Versión</u> (número de versión de la estructura)	MQBNO_VERSION_1	1
<u>ApplicationType</u> (tipo de opción de equilibrio establecido en la estructura)	MQBNO_TIPO_VALOR_S IMPLE	0
<u>Tiempo de espera</u> (tiempo de espera tras el cual el reequilibrio podría interrumpir la actividad de la aplicación)	MQBNO_TIMEOUT_COM O_VALOR_PREDETERMI NADO	0
<u>BalanceOptions</u> (opciones de equilibrio establecidas por la aplicación emisora)	MQBNO_OPCIONES_NI NGUNA	0
Notas: 1. El símbolo ¬ representa un único carácter en blanco. 2. En el lenguaje de programación C, la variable macro MQBNO_DEFAULT contiene los valores que se enumeran en la tabla. Úselo de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: MQBNO MyBNO = {MQBNO_DEFAULT};		

Declaraciones lingüísticas

Declaración C para MQBNO

```
typedef struct tagMQBNO MQBNO;  
struct tagMQBNO {  
    MQCHAR4    StrucId;           /* Structure identifier */  
    MQLONG     Version;          /* Structure version number */  
    MQLONG     Type;             /* Type of balancing options set in the  
                                structure */  
    MQLONG     Timeout;          /* Timeout after which re-balancing might  
                                interrupt application activity */  
    MQLONG     BalanceOptions;    /* Balancing options set by the issuing  
                                application */  
};
```

COBOL declaración para MQBNO

```
**    MQBNO structure  
10  MQBNO.  
**    Structure identifier  
15  MQBNO-STRUCID      PIC X(4).  
**    Structure version number  
15  MQBNO-VERSION     PIC S9(9) BINARY.
```

```

**      Type of balancing options set in the structure
15 MQBNO-TYPE          PIC S9(9) BINARY.
**      Timeout after which re-balancing might interrupt application activity
15 MQBNO-TIMEOUT       PIC S9(9) BINARY.
**      Balancing options set by the issuing application
15 MQBNO-BALANCEOPTIONS PIC S9(9) BINARY.

```

PL/I declaración para MQBNO

```

dcl
  1 MQBNO based,
    3 StrucId          char(4),          /* Structure identifier */
    3 Version          fixed bin(31), /* Structure version number */
    3 Type             fixed bin(31), /* Type of balancing options set in the
                                     structure*/
    3 Timeout          fixed bin(31), /* Timeout after which re-balancing might
                                     interrupt application activity */
    3 BalanceOptions   fixed bin(31), /* Balancing options set by the issuing
                                     application*/

```

Referencia relacionada

“MQCNO - Opciones de conexión” en la página 269

La estructura MQCNO permite a la aplicación especificar opciones relacionadas con la conexión al gestor de colas. La estructura es un parámetro de entrada/salida en la llamada MQCONN.

Detalles de campo para MQBNO

Detalles de los campos de la estructura de opciones de equilibrio MQBNO.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 469. Campos en MQBNO		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
StrucId (identificador de estructura)	MQBNO_STRUC_ID	'BNO~'
Versión (número de versión de la estructura)	MQBNO_VERSION_1	1
ApplicationType (tipo de opción de equilibrio establecido en la estructura)	MQBNO_TIPO_VALOR_S IMPLE	0
Tiempo de espera (tiempo de espera tras el cual el reequilibrio podría interrumpir la actividad de la aplicación)	MQBNO_TIMEOUT_COM O_VALOR_PREDETERMI NADO	0
BalanceOptions (opciones de equilibrio establecidas por la aplicación emisora)	MQBNO OPCIONES_NI NGUNA	0
Notas: - El símbolo ~ representa un único carácter en blanco. - En el lenguaje de programación C, la variable macro MQBNO_DEFAULT contiene los valores que se enumeran en la tabla. Úselo de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQBNO MyBNO = {MQBNO_DEFAULT};</pre>		

StrucId (MQCHAR4)

Este es el identificador de estructura de la estructura de opciones de equilibrio. Siempre es un campo de entrada. Su valor inicial es BNO.

El valor debe ser:

BNO

Identificador de la estructura de opciones de equilibrio.

Para el lenguaje de programación C, también se define la constante MQBNO_STRUC_ID_ARRAY. Esta constante tiene el mismo valor que BNO, pero es una matriz de caracteres en lugar de una cadena.

Debe proporcionar un valor válido para **StrucId** o se devolverá MQRC_BNO_ERROR.

Versión (MQLONG)

Este es el número de versión de la estructura de opciones de equilibrio. Siempre es un campo de entrada.

El valor debe ser:

MQBNO_VERSION_1

Número de versión de la estructura de opciones de equilibrio.

Debe proporcionar un valor válido para **Version** o se devolverá MQRC_BNO_ERROR.

ApplicationType (MQLONG)

El tipo de opción de equilibrio establecido en la estructura.

Los valores posibles son:

MQBNO_TIPO_SALDO_SIMPLE

Equilibrio simple; no se aplican reglas específicas además de las descritas en [Influir en el reequilibrio de aplicaciones en clústeres uniformes](#).

MQBNO_TIPO_SALDO_SOLICITUD_DE_REP

Equilibrio de solicitud-respuesta; después de cada llamada MQPUT, se espera una llamada MQGET coincidente para un mensaje de respuesta. El equilibrio se retrasa hasta que se recibe un mensaje de este tipo o se ha excedido el mensaje de solicitud EXPIRY.

Este campo siempre es un campo de entrada. El valor inicial de este campo es MQBNO_BALTYPE_SIMPLE.

Debe proporcionar un solo valor para el **ApplicationType** campo o se devolverá MQRC_BNO_ERROR.

Nota: Se reserva un valor adicional para este campo de MQBNO_BALTYPE_RA_MANAGED para su uso por parte del IBM MQ adaptador de recursos para entornos JEE. Aunque es un error que una aplicación proporcione este valor directamente, puede, por ejemplo, informarse al consultar el estado de la aplicación.

Tiempo de espera (MQLONG)

El **Timeout** después del cual el reequilibrio podría interrumpir la actividad de la aplicación.

Los valores posibles son:

MQBNO_TIMEOUT_COMO_VALOR_PREDETERMINADO

El valor de tiempo de espera predeterminado establecido.

MQBNO_TIMEOUT_INMEDIATO

Se produce un tiempo de espera inmediato.

MQBNO_TIMEOUT_NEVER

No se produce ningún tiempo de espera.

El valor inicial de este campo es MQBNO_TIMEOUT_AS_DEFAULT.

Debe proporcionar solo un valor de entre los valores definidos, o un valor de 0 a 999999999 segundos, para el **Timeout** campo, o se devolverá MQRC_BNO_ERROR.

BalanceOptions (MQLONG)

Las opciones de equilibrio establecidas por la aplicación emisora.

Los valores posibles son:

MQBNO_OPCIONES_NINGUNA

No hay opciones configuradas

MQBNO_OPTIONS_IGNORE_TRANS

Establecer esta opción permite que las aplicaciones se reequilibren incluso si están en medio de una transacción.

El valor inicial de este campo es MQBNO_OPTIONS_NONE.

Puede proporcionar cualquier combinación de los valores definidos utilizando el operador lógico OR o el carácter para el **BalanceOptions** campo. Cualquier valor que no sea válido provocará que se devuelva un MQRC_BNO_ERROR.

Referencia relacionada

“MQCNO - Opciones de conexión” en la página 269

La estructura MQCNO permite a la aplicación especificar opciones relacionadas con la conexión al gestor de colas. La estructura es un parámetro de entrada/salida en la llamada MQCONN.

MQBO - Opciones de inicio

La estructura MQBO permite a la aplicación especificar opciones relacionadas con la creación de una unidad de trabajo. La estructura es un parámetro de entrada/salida en la llamada MQBEGIN.

Disponibilidad

La estructura MQBO está disponible en las siguientes plataformas:

-  AIX
-  IBM i
-  Linux
-  Windows

La estructura MQBO no está disponible para IBM MQ MQI clients.

Conjunto de caracteres y codificación

Los datos en MQBO deben estar en el juego de caracteres proporcionado por el atributo del **CodedCharSetId** gestor de colas y la codificación del gestor de colas local proporcionada por MQENC_NATIVE. Sin embargo, si la aplicación se ejecuta como un cliente MQ MQI, la estructura debe estar en el conjunto de caracteres y la codificación del cliente.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 470. Campos en MQBO para MQBO		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>StrucId</u> (identificador de estructura)	MQBO_STRUC_ID	'B0-11'
<u>Versión</u> (número de versión de la estructura)	MQBO_VERSION_1	1

Tabla 470. Campos en MQBO para MQBO (continuación)

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
Opciones (opciones que controlan la acción de MQBEGIN)	MQBO_NONE	0

Notas:

1. El símbolo ~ representa un único carácter en blanco.
2. En el lenguaje de programación C, la variable macroMQBO_DEFAULT contiene los valores que se enumeran en la tabla. Úselo de la siguiente manera para proporcionar valores iniciales para los campos de la estructura:

```
MQBO MyBO = {MQBO_DEFAULT};
```

Declaraciones lingüísticas

Declaración C para MQBO

```
typedef struct tagMQBO MQBO;
struct tagMQBO {
    MQCHAR4  StrucId; /* Structure identifier */
    MQLONG   Version; /* Structure version number */
    MQLONG   Options; /* Options that control the action of MQBEGIN */
};
```

COBOL declaración para MQBO

```
** MQBO structure
10 MQBO.
**   Structure identifier
15 MQBO-STRUCID PIC X(4).
**   Structure version number
15 MQBO-VERSION PIC S9(9) BINARY.
**   Options that control the action of MQBEGIN
15 MQBO-OPTIONS PIC S9(9) BINARY.
```

PL/I declaración para MQBO

```
dcl
  1 MQBO based,
  3 StrucId char(4),      /* Structure identifier */
  3 Version fixed bin(31), /* Structure version number */
  3 Options fixed bin(31); /* Options that control the action of
                           MQBEGIN */
```

Declaración de Visual Basic para MQBO

```
Type MQBO
    StrucId As String*4 'Structure identifier'
    Version As Long     'Structure version number'
    Options As Long     'Options that control the action of MQBEGIN'
End Type
```

Detalles de campo para MQBO

La estructura de opciones MQBO Begin permite a la aplicación especificar opciones relacionadas con la creación de una unidad de trabajo. La estructura es un parámetro de entrada/salida en la llamada MQBEGIN.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 471. Campos en MQBO

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
StrucId (identificador de estructura)	MQBO_STRUC_ID	'B0↵↵'
Versión (número de versión de la estructura)	MQBO_VERSION_1	1
Opciones (opciones que controlan la acción de MQBEGIN)	MQBO_NONE	0
Notas: 1. El símbolo ↵ representa un único carácter en blanco. 2. En el lenguaje de programación C, la variable macroMQBO_DEFAULT contiene los valores que se enumeran en la tabla. Úselo de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQBO MyBO = {MQBO_DEFAULT};</pre>		

StrucId (MQCHAR4)

Este es el identificador de estructura de la estructura de opciones de inicio. Siempre es un campo de entrada. Su valor es MQBO_STRUC_ID.

El valor debe ser:

MQBO_STRUC_ID

Identificador para la estructura de opciones de inicio.

Para el lenguaje de programación C, también se define la constante MQBO_STRUC_ID_ARRAY. Tiene el mismo valor que MQBO_STRUC_ID, pero es una matriz de caracteres en lugar de una cadena.

Versión (MQLONG)

Este es el número de versión de la estructura de opciones iniciales. Siempre es un campo de entrada.

El valor debe ser:

MQBO_VERSION_1

Número de versión para la estructura de opciones iniciales.

La siguiente constante especifica el número de versión de la versión actual:

MQBO_VERSIÓN_ACTUAL

Versión actual de la estructura de opciones de inicio.

Opciones (MQLONG)

Este campo es siempre un campo de entrada. Su valor inicial es MQBO_NONE.

El valor debe ser:

MQBO_NONE

No se ha especificado ninguna opción.

MQCBC - Contexto de devolución de llamada

La estructura MQCBC se utiliza para especificar la información de contexto que se pasa a una función de devolución de llamada. La estructura es un parámetro de entrada/salida en la llamada a una rutina consumidora de mensajes.

Disponibilidad

La estructura MQCBC está disponible en las siguientes plataformas:

-  AIX
-  IBM i
-  Linux
-  Windows
-  z/OS

y para IBM MQ MQI clients conectados a estos sistemas.

Versión

La versión actual de MQCBC es MQCBC_VERSION_2.

Conjunto de caracteres y codificación

Los datos en MQCBC deben estar en el juego de caracteres proporcionado por el atributo del **CodedCharSetId** gestor de colas y la codificación del gestor de colas local proporcionada por MQENC_NATIVE. Sin embargo, si la aplicación se ejecuta como un cliente MQ MQI, la estructura estará en el conjunto de caracteres y la codificación del cliente.

Campos

Notas:

- En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.
- No hay valores iniciales para la estructura MQCBC. La estructura se pasa como parámetro a una rutina de devolución de llamada. El gestor de colas inicializa la estructura; las aplicaciones nunca la inicializan.

Tabla 472. Campos en MQCBC	
Campo	Descripción
StrucID	Identificador de la estructura
Versión	Número de versión de la estructura
CallType	Por qué se ha llamado a la función
Hobj	Descriptor de contexto del objeto
CallbackArea	Campo para la función de devolución de llamada que se va a utilizar
ConnectionArea	Campo para la función de devolución de llamada que se va a utilizar
CompCode	Código de terminación
Razón	Código de razón
Estado	Indicación del estado del consumidor actual
DataLength	Longitud del mensaje
BufferLength	Longitud del búfer de mensajes en bytes
Indicadores	Distintivos generales
Nota: El campo restante se ignora si la versión es inferior a MQCBC_VERSION_2	

Tabla 472. Campos en MQCBC (continuación)

Campo	Descripción
ReconnectDelay	Número de milisegundos antes del intento de reconexión

Declaraciones lingüísticas

Declaración C para MQCBC

```
typedef struct tagMQCBC MQCBC;
struct tagMQCBC {
    MQCHAR4    StructId;           /* Structure identifier */
    MQLONG     Version;           /* Structure version number */
    MQLONG     CallType;          /* Why Function was called */
    MQHOBj     Hobj;             /* Object Handle */
    MQPTR      CallbackArea;      /* Callback data passed to the function */
    MQPTR      ConnectionArea;    /* MQCTL data area passed to the function */
    MQLONG     CompCode;          /* Completion Code */
    MQLONG     Reason;           /* Reason Code */
    MQLONG     State;            /* Consumer State */
    MQLONG     DataLength;        /* Message Data Length */
    MQLONG     BufferLength;      /* Buffer Length */
    MQLONG     Flags;            /* Flags containing information about
                                this consumer */

    /* Ver:1 */
    MQLONG     ReconnectDelay;    /* Number of milliseconds before */
    /* Ver:2 */ };               /* reconnect attempt */
```

COBOL declaración para MQCBC

```
** MQCBC structure
10  MQCBC.
** Structure Identifier
15  MQCBC-STRUCID                PIC X(4).
** Structure Version
15  MQCBC-VERSION               PIC S9(9) BINARY.
** Call Type
15  MQCBC-CALLTYPE              PIC S9(9) BINARY.
** Object Handle
15  MQCBC-HOBJ                  PIC S9(9) BINARY.
** Callback User Area
15  MQCBC-CALLBACKAREA          POINTER
** Connection Area
15  MQCBC-CONNECTIONAREA        POINTER
** Completion Code
15  MQCBC-COMPCODE              PIC S9(9) BINARY.
** Reason Code
15  MQCBC-REASON                PIC S9(9) BINARY.
** Consumer State
15  MQCBC-STATE                 PIC S9(9) BINARY.
** Data Length
15  MQCBC-DATALENGTH            PIC S9(9) BINARY.
** Buffer Length
15  MQCBC-BUFFERLENGTH          PIC S9(9) BINARY.
** Flags
15  MQCBC-FLAGS                 PIC S9(9) BINARY.
** Ver:1 **
** Number of milliseconds before reconnect attempt
15  MQCBC-RECONNECTDELAY PIC S9(9) BINARY.
** Ver:2 **
```

PL/I declaración para MQCBC

```
dcl
1  MQCBC based,
3  StructId          char(4),           /* Structure identifier */
3  Version           fixed bin(31),    /* Structure version */
3  CallType          fixed bin(31),    /* Callback type */
3  Hobj              fixed bin(31),    /* Object Handle */
3  CallbackArea      pointer,          /* User area passed to the function */
3  ConnectionArea    pointer,          /* Connection User Area */
3  CompCode          fixed bin(31);    /* Completion Code */
3  Reason            fixed bin(31);    /* Reason Code */
3  State             fixed bin(31);    /* Consumer State */
```

```

3 DataLength          fixed bin(31); /* Message Data Length */
3 BufferLength        fixed bin(31); /* Message Buffer length */
3 Flags              fixed bin(31); /* Consumer Flags */
/* Ver:1 */
3 ReconnectDelay      fixed bin(31); /* Number of milliseconds before */
/* Ver:2 */           /* reconnect attempt */

```

High Level Assembler declaración para MQCBC

```

MQCBC                DSECT
MQCBC                DS 0F      Force fullword alignment
MQCBC_STRUCID        DS CL4     Structure identifier
MQCBC_VERSION        DS F       Structure version number
MQCBC_CALLTYPE       DS F       Why Function was called
MQCBC_HOBJ           DS F       Object Handle
MQCBC_CALLBACKAREA   DS A       Callback data passed to the function
MQCBC_CONNECTIONAREA DS A       MQCTL Data area passed to the function
MQCBC_COMPCODE       DS F       Completion Code
MQCBC_REASON         DS F       Reason Code
MQCBC_STATE          DS F       Consumer State
MQCBC_DATALENGTH     DS F       Message Data Length
MQCBC_BUFFERLENGTH   DS F       Buffer Length
MQCBC_FLAGS          DS F       Flags containing information about this consumer
MQCBC_RECONNECTDELAY DS F       Number of milliseconds before reconnect
MQCBC_LENGTH         EQU *-MQCBC
                     ORG      MQCBC
MQCBC_AREA           DS CL(MQCBC_LENGTH)

```

Detalles de campo para MQCBC

La estructura de contexto MQCBC Callback se utiliza para especificar la información de contexto que se pasa a una función de devolución de llamada. La estructura es un parámetro de entrada/salida en la llamada a una rutina consumidora de mensajes.

Campos

Notas:

- En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético. Los temas secundarios siguen la misma secuencia.
- No hay valores iniciales para la estructura MQCBC. La estructura se pasa como parámetro a una rutina de devolución de llamada. El gestor de colas inicializa la estructura; las aplicaciones nunca la inicializan.

Tabla 473. Campos en MQCBC	
Campo	Descripción
StrucID	Identificador de la estructura
Versión	Número de versión de la estructura
CallType	Por qué se ha llamado a la función
Hobj	Descriptor de contexto del objeto
CallbackArea	Campo para la función de devolución de llamada que se va a utilizar
ConnectionArea	Campo para la función de devolución de llamada que se va a utilizar
CompCode	Código de terminación
Razón	Código de razón
Estado	Indicación del estado del consumidor actual
DataLength	Longitud del mensaje
BufferLength	Longitud del búfer de mensajes en bytes
distintivos	Distintivos generales

Tabla 473. Campos en MQCBC (continuación)	
Campo	Descripción
Nota: El campo restante se ignora si la versión es inferior a MQCBC_VERSION_2	
ReconnectDelay	Número de milisegundos antes del intento de reconexión

StrucID (MQCHAR4)

Este es el identificador de estructura de la estructura de contexto de devolución de llamada. Siempre es un campo de entrada. Su valor es MQCBC_STRUC_ID.

El valor debe ser:

MQCBC_STRUC_ID

Identificador para la estructura del contexto de devolución de llamada.

Para el lenguaje de programación C, también se define la constante MQCBC_STRUC_ID_ARRAY. Tiene el mismo valor que MQCBC_STRUC_ID, pero es una matriz de caracteres en lugar de una cadena.

Versión (MQLONG)

Este es el número de versión de la estructura del contexto de devolución de llamada. Siempre es un campo de entrada.

El valor debe ser:

MQCBC_VERSION_1

Estructura del contexto de devolución de llamada de la versión 1.

La siguiente constante especifica el número de versión de la versión actual:

MQCBC_VERSIÓN_ACTUAL

Versión actual de la estructura del contexto de devolución de llamada.

La función de devolución de llamada siempre recibe la última versión de la estructura.

CallType (MQLONG)

Campo que contiene información sobre por qué se ha llamado a esta función; se definen los siguientes valores.

Tipos de llamadas de entrega de mensajes: Estos tipos de llamadas contienen información sobre un mensaje. Los parámetros **DataLength** **BufferLength** y son válidos para estos tipos de llamadas.

MQCBCT_MSG_ELIMINADO

La función de consumo de mensajes se ha invocado con un mensaje que se ha eliminado de forma destructiva del identificador del objeto.

Si el valor de *CompCode* es MQCC_WARNING, el valor del *Reason* campo es MQRC_TRUNCATED_MSG_ACCEPTED o uno de los códigos que indican un problema de conversión de datos.

MQCBCT_MSG_NO_ELIMINADO

La función de consumo de mensajes se ha invocado con un mensaje que aún no se ha eliminado de forma destructiva del identificador del objeto. El mensaje se puede eliminar de forma destructiva del identificador del objeto utilizando el método *MsgToken*.

Es posible que el mensaje no se haya eliminado porque:

- Las opciones MQGMO solicitaron una operación de exploración, MQGMO_BROWSE_*
- El mensaje es más grande que el búfer disponible y las opciones MQGMO no especifican MQGMO_ACCEPT_TRUNCATED_MSG

Si el valor de *CompCode* es MQCC_WARNING, el valor del *Reason* campo es MQRC_TRUNCATED_MSG_FAILED o uno de los códigos que indican un problema de conversión de datos.

Tipos de llamadas de control de devolución de llamada: Estos tipos de llamadas contienen información sobre el control de la devolución de llamada y no contienen detalles sobre un mensaje. Estos tipos de llamadas se solicitan utilizando Opciones en la estructura MQCBD.

Los parámetros **DataLength** **BufferLength** y no son válidos para estos tipos de llamadas.

MQCBCT_REGISTRAR_LLAMADA

El propósito de este tipo de llamada es permitir que la función de devolución de llamada realice alguna configuración inicial.

La función de devolución de llamada se invoca inmediatamente después de registrarse la devolución de llamada, es decir, al regresar de una llamada MQCB utilizando un valor para el *Operation* campo de MQOP_REGISTER.

Este tipo de llamada se utiliza tanto para consumidores de mensajes como para controladores de eventos.

Si se solicita, esta es la primera invocación de la función de devolución de llamada.

El valor del campo *Reason* es MQRC_NONE.

MQCBCT_INICIAR_LLAMADA

El propósito de este tipo de llamada es permitir que la función de devolución de llamada realice alguna configuración cuando se inicia, por ejemplo, restablecer los recursos que se limpiaron cuando se detuvo anteriormente.

La función de devolución de llamada se invoca cuando se inicia la conexión utilizando MQOP_START o MQOP_START_WAIT.

Si una función de devolución de llamada se registra dentro de otra función de devolución de llamada, este tipo de llamada se invoca cuando la devolución de llamada regresa.

Este tipo de llamada se utiliza únicamente para los consumidores de mensajes.

El valor del campo *Reason* es MQRC_NONE.

MQCBCT_DETENER_LLAMADA

El objetivo de este tipo de llamada es permitir que la función de devolución de llamada realice algunas tareas de limpieza cuando se detiene durante un tiempo, por ejemplo, limpiar los recursos adicionales que se han adquirido durante el consumo de mensajes.

La función de devolución de llamada se invoca cuando se emite una llamada MQCTL utilizando un valor para el *Operation* campo de MQOP_STOP.

Este tipo de llamada se utiliza únicamente para los consumidores de mensajes.

El valor del campo *Reason* se establece para indicar el motivo de la parada.

MQCBCT_DEREGISTRAR_LLAMADA

El propósito de este tipo de llamada es permitir que la función de devolución de llamada realice la limpieza final al final del proceso de consumo. La función de devolución de llamada se invoca cuando:

- La función de devolución de llamada se da de baja mediante una llamada MQCB con MQOP_DEREGISTER.
- La cola está cerrada, lo que provoca una baja implícita. En este caso, la función de devolución de llamada recibe MQHO_UNUSABLE_HOBJ como identificador de objeto.
- La llamada MQDISC finaliza, lo que provoca un cierre implícito y, por lo tanto, una baja del registro. En este caso, la conexión no se desconecta inmediatamente y cualquier transacción en curso aún no se ha confirmado.

Si alguna de estas acciones se realiza dentro de la propia función de devolución de llamada, la acción se invoca una vez que la devolución de llamada regresa.

Este tipo de llamada se utiliza tanto para consumidores de mensajes como para controladores de eventos.

Si se solicita, esta es la última invocación de la función de devolución de llamada.

El valor del campo *Reason* se establece para indicar el motivo de la parada.

MQCBCT_EVENTO_LLAMADA

Función del controlador de eventos

La función del controlador de eventos se ha invocado sin un mensaje cuando el gestor de colas o la conexión se detienen o se inactivan.

Esta llamada se puede utilizar para tomar las medidas adecuadas para todas las funciones de devolución de llamada.

Función de consumidor de mensajes

La función de consumidor de mensajes se ha invocado sin un mensaje cuando se ha detectado un error (*CompCode* = MQCC_FAILED) específico del identificador del objeto; por ejemplo, *Reason* código = MQRC_GET_INHIBITED.

El valor del *Reason* campo se establece para indicar el motivo de la llamada.

MQCBCT_MC_EVENTO_LLAMADA

La función del controlador de eventos se ha invocado para eventos multicast; el controlador de eventos se envía IBM MQ eventos multicast en lugar de eventos IBM MQ «normales».

Para obtener más información sobre MQCBCT_MC_EVENT_CALL, consulte [Informes de excepciones de multidifusión](#).

Hobj (MQHOBJ)

Este es el identificador del objeto para las llamadas al consumidor de mensajes.

Para un controlador de eventos, este valor es MQHO_NONE

La aplicación puede utilizar este identificador y el token del mensaje en el bloque Obtener opciones de mensaje para obtener el mensaje si este no se ha eliminado de la cola.

Este campo siempre es un campo de entrada. El valor inicial de este campo es MQHO_UNUSABLE_HOBJ

CallbackArea (MQPTR)

Este campo está disponible para que lo utilice la función de devolución de llamada.

El gestor de colas no toma ninguna decisión basada en el contenido de este campo y se transfiere sin cambios desde el [CallbackArea](#) campo de la estructura MQCBD, que es un parámetro de la llamada MQCB utilizado para definir la función de devolución de llamada.

Los cambios en el *CallbackArea* se conservan en todas las invocaciones de la función de devolución de llamada para un *Hobj*. Este campo no se comparte con funciones de devolución de llamada para otros identificadores.

Este es un campo de entrada/salida para la función de devolución de llamada. El valor inicial de este campo es un puntero nulo o bytes nulos.

ConnectionArea (MQPTR)

Este campo está disponible para que lo utilice la función de devolución de llamada.

El gestor de colas no toma ninguna decisión basada en el contenido de este campo y se transfiere sin cambios desde el [ConnectionArea](#) campo de la estructura MQCTLO, que es un parámetro de la llamada MQCTL utilizada para controlar la función de devolución de llamada.

Cualquier cambio realizado en este campo por las funciones de devolución de llamada se conserva en todas las invocaciones de la función de devolución de llamada. Esta área se puede utilizar para pasar información que deben compartir todas las funciones de devolución de llamada. A diferencia de *CallbackArea*, esta área es común a todas las devoluciones de llamada para un identificador de conexión.

Este es un campo de entrada y salida. El valor inicial de este campo es un puntero nulo o bytes nulos.

CompCode (MQLONG)

Este campo es el código de finalización. Indica si hubo algún problema al consumir el mensaje.

El valor puede ser uno de los siguientes:

MQCC_OK

Realización satisfactoria

MQCC_WARNING

Aviso (terminación parcial)

MQCC_FAILED

Llamada fallida

Este es un campo de entrada. El valor inicial de este campo es MQCC_OK.

Razón (MQLONG)

Este es el código de motivo que califica el *CompCode*.

Este es un campo de entrada. El valor inicial de este campo es MQRC_NONE.

Estado (MQLONG)

Una indicación sobre el estado actual del consumidor. Este campo es de gran valor para una aplicación cuando se pasa un código de motivo distinto de cero a la función del consumidor.

Puede utilizar este campo para simplificar la programación de aplicaciones, ya que no es necesario codificar el comportamiento para cada código de motivo.

Este es un campo de entrada. El valor inicial de este campo es MQCS_NONE

Tabla 474.		
Estado	Acción del gestor de colas	Valor de la constante
MQCS_NONE Este código de motivo representa una llamada normal sin información adicional sobre el motivo	Ninguna; este es el funcionamiento normal.	0
MQCS_SUSPENDED_TEMPORARY Estos códigos de motivo representan condiciones temporales.	La rutina de devolución de llamada se invoca para informar de la condición y, a continuación, se suspende. Después de un tiempo, el sistema podría intentar la operación de nuevo, lo que puede provocar que se vuelva a producir la misma situación.	1

Tabla 474. (continuación)

Estado	Acción del gestor de colas	Valor de la constante
MQCS_SUSPENDED_USER_ACTION Estos códigos de motivo representan situaciones en las que la devolución de llamada debe tomar medidas para resolver la situación.	El consumidor se suspende y se llama a la rutina de devolución de llamada para informar de la condición. La rutina de devolución de llamada debe resolver la condición si es posible y, a continuación, REANUDAR o cerrar la conexión.	2
MQCS_SUSPENDED Estos códigos de motivo representan fallos que impiden nuevas devoluciones de llamada de mensajes.	El gestor de colas suspende automáticamente la función de devolución de llamada. Si se reanuda la función de devolución de llamada, es probable que vuelva a recibir el mismo código de motivo.	3
MQCS_STOPPED Estos códigos de motivo representan el final del consumo de mensajes.	Entregado al controlador de excepciones y a las devoluciones de llamada que especificaron MQCBDO_STOP_CALL. No se pueden consumir más mensajes.	4

DataLength (MQLONG)

Es la longitud en bytes de los datos de la aplicación en el mensaje. Si el valor es cero, significa que el mensaje no contiene datos de aplicación.

El campo « DataLength » contiene la longitud del mensaje, pero no necesariamente la longitud de los datos del mensaje transmitidos al consumidor. Es posible que el mensaje se haya truncado. Utilice el [ReturnedLength](#) campo en MQGMO para determinar cuántos datos se han transferido realmente al consumidor.

Si el código de motivo indica que el mensaje ha sido truncado, puede utilizar el campo « DataLength » (Tamaño del mensaje original) para determinar el tamaño real del mensaje. Esto le permite determinar el tamaño del búfer necesario para alojar los datos del mensaje y, a continuación, emitir una llamada MQCB para actualizar el [MaxMsgLength](#) con un valor adecuado.

Si se especifica la opción MQGMO_CONVERT, el mensaje convertido podría ser mayor que el valor devuelto para DataLength. En tales casos, es probable que la aplicación tenga que emitir una llamada MQCB para actualizar el [MaxMsgLength](#) to a un valor superior al devuelto por el gestor de colas para DataLength.

Para evitar problemas de truncamiento de mensajes, especifique MaxMsgLength como MQCBD_FULL_MSG_LENGTH. Esto hace que el gestor de colas asigne un búfer para la longitud completa del mensaje después de la conversión de datos. Sin embargo, tenga en cuenta que, aunque se especifique esta opción, es posible que no haya suficiente espacio de almacenamiento disponible para procesar correctamente la solicitud. Las aplicaciones siempre deben comprobar el código de motivo devuelto. Por ejemplo, si no es posible asignar suficiente almacenamiento para convertir el mensaje, este se devuelve a la aplicación sin convertir.

Este es un campo de entrada para la función de consumo de mensajes; no es relevante para una función de controlador de eventos.

BufferLength (MQLONG)

Este campo es la longitud en bytes del búfer de mensajes que se ha pasado a esta función.

El búfer puede ser mayor que el valor « MaxMsgLength » definido para el consumidor y que el valor « ReturnedLength » en el MQGMO.

La longitud real del mensaje se indica en [DataLength](#) el campo.

La aplicación puede utilizar todo el búfer para sus propios fines durante la duración de la función de devolución de llamada.

Este es un campo de entrada para la función de consumo de mensajes; no es relevante para una función de controlador de excepciones.

Flags (MQLONG)

Banderas que contienen información sobre este consumidor.

Se define la siguiente opción:

MQCBCF_READA_BUFFER_VACÍO

Esta bandera se puede devolver si una llamada MQCLOSE anterior que utilizaba la opción MQCO_QUIESCE falló con un código de motivo MQRC_READ_AHEAD_MSGS.

Este código indica que se está devolviendo el último mensaje leído por adelantado y que el búfer ahora está vacío. Si la aplicación emite otra llamada MQCLOSE utilizando la opción MQCO_QUIESCE), se ejecuta correctamente.

Tenga en cuenta que no se garantiza que una aplicación reciba un mensaje con esta bandera activada, ya que aún podría haber mensajes en el búfer de lectura anticipada que no coincidan con los criterios de selección actuales. En este caso, la función de consumidor se invoca con el código de motivo MQRC_HOBJ_QUIESCED.

Si el búfer de lectura anticipada está completamente vacío, se invoca al consumidor con el indicador MQCBCF_READA_BUFFER_EMPTY y el código de motivo MQRC_HOBJ_QUIESCED_NO_MSGS.

Este es un campo de entrada para la función de consumo de mensajes; no es relevante para una función de controlador de eventos.

ReconnectDelay (MQLONG)

ReconnectDelay indica cuánto tiempo esperará el gestor de colas antes de intentar volver a conectarse. El campo puede ser modificado por un controlador de eventos para cambiar el retraso o detener la reconexión por completo.

Utilice el ReconnectDelay campo solo si el valor del campo Motivo en el contexto de devolución de llamada es MQRC_RECONNECTING.

Al entrar en el controlador de eventos, el valor de ReconnectDelay es el número de milisegundos que el gestor de colas va a esperar antes de intentar volver a conectarse. La siguiente tabla enumera los valores que se pueden establecer para modificar el comportamiento del gestor de colas al regresar del controlador de eventos.

Tabla 475. ReconnectDelay valores		
Nombre	Valor	Descripción
MQRD_NO_RECONNECT	-1	No vuelva a intentar reconectarse. Se devuelve un error a la aplicación.
MQRD_NO_DELAY	0	Intenta volver a conectarte inmediatamente.
Milliseconds	>0	Espera estos milisegundos antes de volver a intentar la conexión.

MQCBD - Descriptor de devolución de llamada

La estructura MQCBD se utiliza para especificar una función de devolución de llamada y las opciones que controlan su uso por parte del gestor de colas. La estructura es un parámetro de entrada en la llamada MQCB.

Disponibilidad

La estructura MQCBD está disponible en las siguientes plataformas:

-  AIX
-  IBM i
-  Linux
-  Windows
-  z/OS

y para IBM MQ MQI clients conectados a estos sistemas.

Versión

La versión actual de MQCBD es MQCBD_VERSION_1.

Conjunto de caracteres y codificación

Los datos en MQCBD deben estar en el conjunto de caracteres proporcionado por el atributo del **CodedCharSetId** gestor de colas y la codificación del gestor de colas local proporcionada por MQENC_NATIVE. Sin embargo, si la aplicación se ejecuta como un cliente MQ MQI, la estructura debe estar en el conjunto de caracteres y la codificación del cliente.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 476. Campos en MQCBD		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>StrucID</u> (identificador de estructura)	MQCBD_STRUC_ID	'CBD→'
<u>Versión</u> (número de versión de la estructura)	MQCBD_VERSION_1	1
<u>CallbackType</u> (tipo de función de devolución de llamada)	MQCBT_MENSAJE_CONSUMIDOR	1
<u>Opciones</u> (opciones que controlan el consumo de mensajes)	MQCBDO_NINGUNO	0
<u>CallbackArea</u> (campo para la función de devolución de llamada que se va a utilizar)	Ninguna	Puntero nulo o espacios en blanco nulos
<u>CallbackFunction</u> (si la función se invoca como una llamada a la API)	Ninguna	Puntero nulo o espacios en blanco nulos
<u>CallbackName</u> (si la función se invoca como un programa vinculado dinámicamente)	Ninguna	Cadena nula o espacios en blanco
<u>MaxMsgLength</u> (longitud del mensaje más largo que se puede leer)	MQCBD_FULL_MSG_LENGTH	-1

Tabla 476. Campos en MQCBD (continuación)

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
Notas: 1. El símbolo ~ representa un único carácter en blanco. 2. El valor «Cadena nula o espacios en blanco» denota la cadena nula en el lenguaje de programación C y los caracteres en blanco en otros lenguajes de programación. 3. En el lenguaje de programación C, la variable macroMQCBD_DEFAULT contiene los valores que se enumeran en la tabla. Úselo de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQCBD MyCBD = {MQCBD_DEFAULT};</pre>		

Declaraciones lingüísticas

Declaración C para MQCBD

```
typedef struct tagMQCBD MQCBD;
struct tagMQCBD {
    MQCHAR4      StrucId;           /* Structure identifier */
    MQLONG       Version;          /* Structure version number */
    MQLONG       CallbackType;     /* Callback function type */
    MQLONG       Options;          /* Options controlling message
                                   consumption */
    MQPTR        CallbackArea;     /* User data passed to the function */
    MQPTR        CallbackFunction; /* Callback function pointer */
    MQCHAR128    CallbackName;     /* Callback name */
    MQLONG       MaxMsgLength;     /* Maximum message length */
};
```

COBOL declaración para MQCBD

```
** MQCBDC structure
10  MQCBD.
** Structure Identifier
15  MQCBD-STRUCID                PIC X(4).
** Structure Version
15  MQCBD-VERSION                PIC S9(9) BINARY.
** Callback Type
15  MQCBD-CALLBACKTYPE          PIC S9(9) BINARY.
** Options
15  MQCBD-OPTIONS                PIC S9(9) BINARY.
** Callback User Area
15  MQCBD-CALLBACKAREA           POINTER
** Callback Function Pointer
15  MQCBD-CALLBACKFUNCTION       FUNCTION-POINTER
** Callback Program Name
15  MQCBD-CALLBACKNAME           PIC X(128)
** Maximum Message Length
15  MQCBD-MAXMSGLNGTH           PIC S9(9) BINARY.
```

PL/I declaración para MQCBD

```
dcl
1  MQCBD based,
3  StrucId          char(4),          /* Structure identifier*/
3  Version          fixed bin(31),    /* Structure version*/
3  CallbackType     fixed bin(31),    /* Callback function type */
3  Options          fixed bin(31),    /* Options */
3  CallbackArea     pointer,          /* User area passed to the function */
3  CallbackFunction pointer,          /* Callback Function Pointer */
3  CallbackName     char(128),        /* Callback Program Name */
3  MaxMsgLength     fixed bin(31);    /* Maximum Message Length */
```

Detalles de campo para MQCBD

La estructura del descriptor MQCBD Callback se utiliza para especificar una función de devolución de llamada y las opciones que controlan su uso por parte del gestor de colas. La estructura es un parámetro de entrada en la llamada MQCB.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético. Los temas secundarios siguen la misma secuencia.

Tabla 477. Campos en MQCBD		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>StrucID</u> (identificador de estructura)	MQCBD_STRUC_ID	'CBD↵'
<u>Versión</u> (número de versión de la estructura)	MQCBD_VERSION_1	1
<u>CallbackType</u> (tipo de función de devolución de llamada)	MQCBT_MENSAJE_CONSUMIDOR	1
<u>Opciones</u> (opciones que controlan el consumo de mensajes)	MQCBDO_NINGUNO	0
<u>CallbackArea</u> (campo para la función de devolución de llamada que se va a utilizar)	Ninguna	Puntero nulo o espacios en blanco nulos
<u>CallbackFunction</u> (si la función se invoca como una llamada a la API)	Ninguna	Puntero nulo o espacios en blanco nulos
<u>CallbackName</u> (si la función se invoca como un programa vinculado dinámicamente)	Ninguna	Cadena nula o espacios en blanco
<u>MaxMsgLength</u> (longitud del mensaje más largo que se puede leer)	MQCBD_FULL_MSG_LENGTH	-1
Notas: 1. El símbolo ↵ representa un único carácter en blanco. 2. El valor «Cadena nula o espacios en blanco» denota la cadena nula en el lenguaje de programación C y los caracteres en blanco en otros lenguajes de programación. 3. En el lenguaje de programación C, la variable macroMQCBD_DEFAULT contiene los valores que se enumeran en la tabla. Úselo de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQCBD MyCBD = {MQCBD_DEFAULT};</pre>		

StrucID (MQCHAR4)

Este es el identificador de estructura de la estructura descriptora de devolución de llamada. Siempre es un campo de entrada. Su valor es MQCBD_STRUC_ID.

El valor debe ser:

MQCBD_STRUC_ID

Identificador para la estructura del descriptor de devolución de llamada.

Para el lenguaje de programación C, también se define la constante MQCBD_STRUC_ID_ARRAY. Tiene el mismo valor que MQCBD_STRUC_ID, pero es una matriz de caracteres en lugar de una cadena.

Versión (MQLONG)

Este es el número de versión de la estructura del descriptor de devolución de llamada. Siempre es un campo de entrada.

El valor debe ser:

MQCBD_VERSION_1

Estructura del descriptor de devolución de llamada de la versión 1.

La siguiente constante especifica el número de versión de la versión actual:

MQCBD_VERSION_1_ACTUAL

Versión actual de la estructura del descriptor de devolución de llamada.

CallbackType (MQLONG)

Estructura del descriptor de devolución de llamada: campo « CallbackType ».

Este es el tipo de la función de devolución de llamada. El valor debe ser uno de los siguientes:

MQCBT_MENSAJE_CONSUMIDOR

Define esta devolución de llamada como una función consumidora de mensajes.

La función de devolución de llamada del consumidor de mensajes se invoca cuando hay un mensaje disponible en un identificador de objeto que cumple los criterios de selección especificados y se inicia la conexión.

MQCBT_EVENT_HANDLER

Define esta devolución de llamada como la rutina de eventos asíncronos; no está diseñada para consumir mensajes para un identificador.

Hob j no es necesario en la llamada MQCB que define el controlador de eventos y se ignora si se especifica.

El controlador de eventos se activa en condiciones que afectan a todo el entorno del consumidor de mensajes. La función de consumidor se invoca sin un mensaje cuando se produce un evento, por ejemplo, la detención o la inactividad de un gestor de colas o una conexión. No se requiere para condiciones específicas de un único consumidor de mensajes, por ejemplo, MQRC_GET_INHIBITED.

Los eventos se envían a la aplicación, independientemente de si la conexión está iniciada o detenida, excepto en los siguientes entornos:

- CICS sobre z/OS medio ambiente
- aplicaciones sin hilo

Si el llamante no pasa uno de estos valores, la llamada falla con un *Reason* código de MQRC_CALLBACK_TYPE_ERROR

Este campo siempre es un campo de entrada. El valor inicial de este campo es MQCBT_MESSAGE_CONSUMER.

Opciones (MQLONG)

Estructura del descriptor de devolución de llamada - Campo Opciones

Puede especificar una o varias de estas opciones. Para especificar más de una opción, suma los valores (sin añadir la misma constante más de una vez) o combínalos mediante la operación OR bit a bit (si el lenguaje de programación admite operaciones bit a bit).

MQCBDO_FAIL_IF QUIESCING

La llamada MQCB falla si el gestor de colas se encuentra en estado de inactividad.

En z/OS, esta opción también fuerza el fallo de la llamada MQCB si la conexión (para una CICS o IMS aplicación) se encuentra en estado de inactividad.

Especifique MQGMO_FAIL_IF QUIESCING en las opciones MQGMO pasadas en la llamada MQCB para que se envíe una notificación a los consumidores de mensajes cuando estén inactivos.

Opciones de control : Las siguientes opciones controlan si se llama a la función de devolución de llamada, sin un mensaje, cuando cambia el estado del consumidor:

MQCBDO_REGISTRAR_LLAMADA

La función de devolución de llamada se invoca con el tipo de llamada MQCBCT_REGISTER_CALL.

MQCBDO_INICIAR_LLAMADA

La función de devolución de llamada se invoca con el tipo de llamada MQCBCT_START_CALL.

MQCBDO_DETENER_LLAMADA

La función de devolución de llamada se invoca con el tipo de llamada MQCBCT_STOP_CALL.

MQCBDO_DEREGISTRAR_LLAMADA

La función de devolución de llamada se invoca con el tipo de llamada MQCBCT_DEREGISTER_CALL.

MQCBDO_EVENTO_LLAMADA

La función de devolución de llamada se invoca con el tipo de llamada MQCBCT_EVENT_CALL.

MQCBDO_MC_EVENTO_LLAMADA

La función de devolución de llamada se invoca con el tipo de llamada MQCBCT_MC_EVENT_CALL.

Consulte [CallType](#) para obtener más detalles sobre estos tipos de llamadas.

Opción predeterminada : Si no necesita ninguna de las opciones descritas, utilice la siguiente opción:

MQCBDO_NINGUNO

Utilice este valor para indicar que no se han especificado otras opciones; todas las opciones toman sus valores predeterminados.

MQCBDO_NONE se define para facilitar la documentación del programa; no está previsto que esta opción se utilice con ninguna otra, pero como su valor es cero, dicho uso no puede detectarse.

Este es un campo de entrada. El valor inicial del campo *Options* es MQCBDO_NONE.

CallbackArea (MQPTR)

Estructura del descriptor de devolución de llamada - Campo « CallbackArea »

Este es un campo que está disponible para que lo utilice la función de devolución de llamada.

El gestor de colas no toma ninguna decisión basada en el contenido de este campo y se transfiere sin cambios desde el [CallbackArea](#) campo de la estructura MQCBC, que es un parámetro de la declaración de la función de devolución de llamada.

El valor solo se utiliza en un *Operation* que tenga un valor MQOP_REGISTER, sin ninguna devolución de llamada definida actualmente, y no sustituye a una definición anterior.

Este es un campo de entrada y salida para la función de devolución de llamada. El valor inicial de este campo es un puntero nulo o bytes nulos.

CallbackFunction (MQPTR)

Estructura del descriptor de devolución de llamada - Campo « CallbackFunction »

La función de devolución de llamada se invoca como una llamada a función.

Utilice este campo para especificar un puntero a la función de devolución de llamada.

Debe especificar *CallbackFunction* o *CallbackName*. Si se especifican ambos, se devuelve el código de motivo MQRC_CALLBACK_ROUTINE_ERROR.

Si no se *CallbackFunction* establece ni *CallbackName* ni, la llamada falla con el código de motivo MQRC_CALLBACK_ROUTINE_ERROR.

Esta opción no es compatible con el siguiente entorno: lenguajes de programación y compiladores que no admiten referencias a punteros de función. En tales situaciones, la llamada falla con el código de motivo MQRC_CALLBACK_ROUTINE_ERROR.

 En z/OS, la función debe esperar ser llamada con las convenciones de enlace del sistema operativo. Por ejemplo, en el lenguaje de programación C, especifique:

```
#pragma linkage(MQCB_FUNCTION,OS)
```

Este es un campo de entrada. El valor inicial de este campo es un puntero nulo o bytes nulos.

CallbackName (MQCHAR128)

Estructura del descriptor de devolución de llamada - Campo « CallbackName »

La función de devolución de llamada se invoca como un programa vinculado dinámicamente.

Debe especificar *CallbackFunction* o *CallbackName*. Si se especifican ambos, se devuelve el código de motivo MQRC_CALLBACK_ROUTINE_ERROR.

Si no se *CallbackFunction* establece ni *CallbackName* ni, la llamada falla con el código de motivo MQRC_CALLBACK_ROUTINE_ERROR.

El módulo se carga cuando se registra la primera rutina de devolución de llamada que se va a utilizar y se descarga cuando se da de baja la última rutina de devolución de llamada que lo utiliza.

Salvo que se indique lo contrario en el texto siguiente, el nombre se alinea a la izquierda dentro del campo, sin espacios en blanco incrustados; el nombre se rellena con espacios en blanco hasta alcanzar la longitud del campo. En las descripciones que siguen, los corchetes ([]) indican información opcional:

IBM i

El nombre de la devolución de llamada puede tener uno de los siguientes formatos:

- Biblioteca «/» Programa
- Biblioteca «/» ServiceProgram «("FunctionName")»

Por ejemplo, MyLibrary/MyProgram(MyFunction).

El nombre de la biblioteca puede ser *LIBL. Tanto el nombre de la biblioteca como el del programa tienen un límite máximo de 10 caracteres.

AIX and Linux

El nombre de la llamada de retorno es el nombre de un módulo o biblioteca que se puede cargar dinámicamente, con el sufijo del nombre de una función que reside en esa biblioteca. El nombre de la función debe ir entre paréntesis. El nombre de la biblioteca puede ir precedido opcionalmente de una ruta de directorio:

```
[path]library(function)
```

Si no se especifica la ruta, se utiliza la ruta de búsqueda del sistema.

El nombre está limitado a un máximo de 128 caracteres.

Windows

El nombre de la llamada de retorno es el nombre de una biblioteca de enlace dinámico, seguido del nombre de una función que reside en esa biblioteca. El nombre de la función debe ir entre paréntesis. El nombre de la biblioteca puede ir precedido opcionalmente por una ruta de directorio y una unidad:

```
[d:][path]library(function)
```

Si no se especifican la unidad y la ruta, se utiliza la ruta de búsqueda del sistema.

El nombre está limitado a un máximo de 128 caracteres.

z/OS

El nombre de la llamada de retorno es el nombre de un módulo de carga válido para la especificación en el parámetro EP de la macro LINK o LOAD.

El nombre está limitado a un máximo de 8 caracteres.

z/OS CICS

El nombre de la devolución de llamada es el nombre de un módulo de carga válido para la especificación en el parámetro PROGRAM de la macro de comando EXEC CICS LINK.

El nombre está limitado a un máximo de 8 caracteres.

El programa se puede definir como remoto utilizando la opción REMOTESYTEM de la definición del PROGRAMA instalado o mediante el programa de enrutamiento dinámico.

La región CICS remota debe estar conectada para IBM MQ que el programa pueda utilizar llamadas IBM MQ API. Sin embargo, tenga en cuenta que el campo Hobj de la estructura MQCBC no es válido en un sistema remoto.

Si se produce un error al intentar cargar *CallbackName*, se devuelve uno de los siguientes códigos de error a la aplicación:

- MQRC_MÓDULO_NO_ENCONTRADO
- MQRC_MÓDULO_INVÁLIDO
- MQRC_MODULE_ENTRY_NOT_FOUND

También se escribe un mensaje en el registro de errores que contiene el nombre del módulo para el que se intentó la carga y el código de motivo del fallo del sistema operativo.

Este es un campo de entrada. El valor inicial de este campo es una cadena nula o espacios en blanco.

MaxMsgLength (MQLONG)

Es la longitud en bytes del mensaje más largo que se puede leer desde el identificador y pasar a la rutina de devolución de llamada.

Si un mensaje tiene una longitud mayor, la rutina de devolución de llamada recibe *MaxMsgLength* bytes del mensaje y el código de motivo:

- MQRC_TRUNCATED_MSG_FAILED o
- MQRC_TRUNCATED_MSG_ACCEPTED si ha especificado MQGMO_ACCEPT_TRUNCATED_MSG.

La longitud real del mensaje se indica en el DataLength campo de la estructura MQCBC.

Se define el siguiente valor especial:

MQCBD_FULL_MSG_LENGTH

El sistema ajusta la longitud del búfer para devolver los mensajes sin truncar.

Si no hay suficiente memoria disponible para asignar un búfer para recibir el mensaje, el sistema llama a la función de devolución de llamada con un código de motivo MQRC_STORAGE_NOT_AVAILABLE.

Si, por ejemplo, solicita la conversión de datos y no hay suficiente memoria disponible para convertir los datos del mensaje, el mensaje sin convertir se pasa a la función de devolución de llamada.

Este es un campo de entrada. El valor inicial del campo *MaxMsgLength* es MQCBD_FULL_MSG_LENGTH.

MQCHARV - Cadena de longitud variable

Utilice la estructura MQCHARV para describir una cadena de longitud variable.

Disponibilidad

La estructura MQCHARV está disponible en las siguientes plataformas:

-  AIX
-  IBM i
-  Linux
-  Windows

y para IBM MQ MQI clients conectados a estos sistemas.

Conjunto de caracteres y codificación

Los datos del MQCHARV deben estar codificados según el gestor de colas local proporcionado por MQENC_NATIVE y el juego de caracteres del campo VSCCSID dentro de la estructura. Si la aplicación se ejecuta como un cliente MQ, la estructura debe estar en la codificación del cliente. Algunos conjuntos de caracteres tienen una representación que depende de la codificación. Si VSCCSID es uno de estos conjuntos de caracteres, la codificación utilizada es la misma que la de los demás campos de MQCHARV. El conjunto de caracteres identificado por VSCCSID puede ser un conjunto de caracteres de doble byte (DBCS).

Utilización

La estructura MQCHARV aborda datos que pueden ser discontinuos con la estructura que los contiene. Para abordar estos datos, se pueden utilizar campos declarados con el tipo de datos puntero. Tenga en cuenta que COBOL no admite el tipo de datos puntero en todos los entornos. Por este motivo, los datos también se pueden abordar utilizando campos que contengan el desplazamiento de los datos desde el inicio de la estructura que contiene el MQCHARV.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 478. Campos en MQCHARV		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>VSPtr</u> (puntero a la cadena de longitud variable)	Ninguna	Puntero nulo o bytes nulos.
<u>VSOOffset</u> (desplazamiento en bytes de la cadena de longitud variable desde el inicio de la estructura que contiene esta estructura MQCHARV)	Ninguna	0
<u>VSBufSize</u> (tamaño en bytes del búfer al que apunta el campo VSPtr o VSOOffset)	MQVS_USE_VSLENGTH	0
<u>VSLength</u> (longitud en bytes de la cadena de longitud variable a la que apunta el campo VSPtr o VSOOffset)	Ninguna	0
<u>VSCCSID</u> (identificador del conjunto de caracteres de la cadena de longitud variable a la que apunta el campo VSPtr o VSOOffset)	MQCCSI_APPL	-3
Nota: En el lenguaje de programación C, la variable macro MQCHARV_DEFAULT contiene los valores que se enumeran en la tabla. Se puede utilizar de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQCHARV MyVarStr = {MQCHARV_DEFAULT};</pre>		

Declaraciones lingüísticas

Declaración C para MQCHARV

```
typedef struct tagMQCHARV MQCHARV;  
struct tagMQCHARV {  
    MQPTR    VSPtr;           /* Address of variable length string */  
    MQLONG   VSOFFSET;        /* Offset of variable length string */  
    MQLONG   VBufSize;        /* Size of buffer */  
    MQLONG   VSLength;        /* Length of variable length string */  
    MQLONG   VSCCSID;         /* CCSID of variable length string */  
};
```

COBOL declaración para MQCHARV

```
** MQCHARV structure  
10 MQCHARV.  
** Address of variable length string  
15 MQCHARV-VSPTR          POINTER.  
** Offset of variable length string  
15 MQCHARV-VSOFFSET      PIC S9(9) BINARY.  
** Size of buffer  
15 MQCHARV-VBUFSIZE      PIC S9(9) BINARY.  
** Length of variable length string  
15 MQCHARV-VSLNGTH      PIC S9(9) BINARY.  
** CCSID of variable length string  
15 MQCHARV-VSCCSID      PIC S9(9) BINARY.
```

Nota: Si desea transferir una aplicación COBOL entre entornos, debe averiguar si el tipo de datos del puntero está disponible en todos los entornos previstos. Si no es así, la aplicación debe abordar los datos utilizando los campos de desplazamiento en lugar de los campos de puntero. En entornos que no admiten punteros, puede declarar los campos de puntero como cadenas de bytes de la longitud adecuada, con el valor inicial como cadena de bytes totalmente nula. No modifique este valor inicial si está utilizando los campos de compensación. Una forma de hacerlo sin cambiar los libros de copia suministrados es utilizar lo siguiente:

```
COPY CMQCHRVV REPLACING POINTER BY ==BINARY PIC S9(9)==.
```

donde se CMQCHRVV puede canjear por el cuaderno que se va a utilizar.

PL/I declaración para MQCHARV

```
dcl  
1 MQCHARV based,  
3 VSPtr          pointer,      /* Address of variable length string */  
3 VSOFFSET       fixed bin(31), /* Offset of variable length string */  
3 VBufSize       fixed bin(31), /* Size of buffer */  
3 VSLength       fixed bin(31), /* Length of variable length string */  
3 VSCCSID        fixed bin(31); /* CCSID of variable length string */
```

High Level Assembler declaración para MQCHARV

MQCHARV	DSECT	
MQCHARV_VSPTR	DS F	Address of variable length string
MQCHARV_VSOFFSET	DS F	Offset of variable length string
MQCHARV_VBUFSIZE	DS F	Size of buffer
MQCHARV_VSLNGTH	DS F	Length of variable length string
MQCHARV_VSCCSID	DS F	CCSID of variable length string
*		
MQCHARV_LENGTH	EQU	*-MQCHARV
	ORG	MQCHARV
MQCHARV_AREA	DS	CL(MQCHARV_LENGTH)

Detalles de campo para MQCHARV

Utilice la estructura MQCHARV para describir una cadena de longitud variable.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 479. Campos en MQCHARV

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>VSPtr</u> (puntero a la cadena de longitud variable)	Ninguna	Puntero nulo o bytes nulos.
<u>VSOOffset</u> (desplazamiento en bytes de la cadena de longitud variable desde el inicio de la estructura que contiene esta estructura MQCHARV)	Ninguna	0
<u>VSBufSize</u> (tamaño en bytes del búfer al que apunta el campo VSPtr o VSOOffset)	MQVS_USE_VSLENGTH	0
<u>VSLength</u> (longitud en bytes de la cadena de longitud variable a la que apunta el campo VSPtr o VSOOffset)	Ninguna	0
<u>VSCCSID</u> (identificador del conjunto de caracteres de la cadena de longitud variable a la que apunta el campo VSPtr o VSOOffset)	MQCCSI_APPL	-3
Nota: En el lenguaje de programación C, la variable macro MQCHARV_DEFAULT contiene los valores que se enumeran en la tabla. Se puede utilizar de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQCHARV MyVarStr = {MQCHARV_DEFAULT};</pre>		

VSPtr (MQPTR)

Este es un puntero a la cadena de longitud variable.

Puede utilizar el campo VSPtr o VSOOffset para especificar la cadena de longitud variable, pero no ambos.

El valor inicial de este campo es un puntero nulo o bytes nulos.

VSOOffset (MQLONG)

El desplazamiento en bytes de la cadena de longitud variable desde el inicio de MQCHARV, o la estructura que la contiene.

Cuando la estructura MQCHARV está incrustada dentro de otra estructura, este valor es la compensación en bytes de la cadena de longitud variable desde el inicio de la estructura que contiene esta estructura MQCHARV. Cuando la estructura MQCHARV no está incrustada dentro de otra estructura, por ejemplo, si se especifica como parámetro en una llamada a una función, el desplazamiento es relativo al inicio de la estructura MQCHARV.

El desplazamiento puede ser positivo o negativo.

Puede utilizar el campo VSPtr o VSOOffset para especificar la cadena de longitud variable, pero no ambos.

El valor inicial de este campo es 0.

VSBufSize (MQLONG)

Este es el tamaño en bytes del búfer al que apunta el campo VSPtr o VSOOffset.

Cuando la estructura MQCHARV se utiliza como campo de salida en una llamada a función, este campo debe inicializarse con la longitud del búfer proporcionado. Si el valor de VSLength es mayor que VSBufSize, solo se devuelven VSBufSize bytes de datos al llamante en el búfer.

Este valor debe ser mayor o igual a cero, o el siguiente valor especial que se reconoce:

MQVS_USE_VSLENGTH

Cuando se especifica, la longitud del búfer se toma del campo VSLength de la estructura MQCHARV. No utilice este valor cuando utilice la estructura como campo de salida y se proporcione un búfer.

Este es el valor inicial de este campo.

VSLength (MQLONG)

La longitud en bytes de la cadena de longitud variable a la que apunta el campo VSPtr o VSOOffset.

El valor inicial de este campo es 0. El valor debe ser mayor o igual a cero o el siguiente valor especial que se reconoce:

MQVS_TERMINADO_CON_NULL

Si no se especifica MQVS_NULL_TERMINATED, se incluyen bytes VSLength como parte de la cadena. Si hay caracteres nulos, estos no delimitan la cadena.

Si se especifica MQVS_NULL_TERMINATED, la cadena se delimita por el primer carácter nulo que se encuentra en la cadena. El valor nulo en sí mismo no se incluye como parte de esa cadena.

Nota: El carácter nulo utilizado para terminar una cadena si se especifica MQVS_NULL_TERMINATED es un nulo del juego de códigos especificado por VSCCSID.

Por ejemplo, en UTF-16 (CCSID 1200, 13488 y 17584), se trata de la codificación Unicode de dos bytes en la que un valor nulo se representa mediante un número de 16 bits compuesto únicamente por ceros. En UTF-16 es habitual encontrar bytes individuales establecidos en cero que forman parte de caracteres (por ejemplo, caracteres ASCII de 7 bits), pero las cadenas solo terminarán en nulo cuando se encuentren dos bytes «cero» en un límite de byte par. Es posible obtener dos bytes «cero» en un límite impar cuando ambos forman parte de caracteres válidos. Por ejemplo, x'01' x'00 x'00' x'30' representa dos caracteres Unicode válidos y no termina la cadena con un valor nulo.

VSCCSID (MQLONG)

Este es el identificador del conjunto de caracteres de la cadena de longitud variable a la que se refiere el **VSPtr** campo **VSOOffset** o.

El valor inicial de este campo es *MQCCSI_APPL*, definido por MQ para indicar que debe cambiarse al identificador de conjunto de caracteres real del proceso actual. Como resultado, el valor de la constante *MQCCSI_APPL* nunca se asocia con una cadena de longitud variable.

El valor inicial de este campo se puede cambiar definiendo un valor diferente para la constante *MQCCSI_APPL* para su unidad de compilación. La forma de hacerlo depende del lenguaje de programación de tu aplicación.

 En z/OS los sistemas, la aplicación CCSID predeterminada utilizada por *MQCCSI_APPL* se define de la siguiente manera:

- Para aplicaciones LE por lotes que utilizan la interfaz DLL, el valor predeterminado es el CODESET asociado a la configuración regional actual en el momento **MQCONN** en que se emite (el valor predeterminado es 1047).
- En el caso de las aplicaciones LE por lotes vinculadas a uno de los stubs de MQ, el valor predeterminado es el CODESET asociado a la configuración regional vigente en el momento de la primera llamada a MQI realizada tras **MQCONN** (el valor predeterminado es 1047).
- En el caso de las aplicaciones por lotes que no son de LE y se ejecutan en un z/OS UNIX System Services hilo, el valor predeterminado es el valor de THLCCSID en el momento de la primera llamada a `MQI` realizada tras **MQCONN** (el valor predeterminado es 1047).
- Para otras aplicaciones por lotes, el valor predeterminado es el CCSID del gestor de colas.

Nota: Para aplicaciones que no son por lotes, es necesario definir manualmente el CCSID.

Redefinición de MQCCSI_APPL

Los siguientes ejemplos muestran cómo se puede anular el valor de MQCCSI_APPL en varios lenguajes de programación. Debe cambiar el valor de MQCCSI_APPL, eliminando la necesidad de establecer el VSCCSID para cada cadena de longitud variable por separado y asegurándose de que se utiliza el CCSID correcto.

En estos ejemplos, el CCSID está establecido en 1208; cámbielo al valor que necesite. Este se convierte en el valor predeterminado, que puede anular configurando el VSCCSID en cualquier instancia específica de MQCHARV.

Uso de C

```
#define MQCCSI_APPL 1208
#include <cmqc.h>
```

Utilización de COBOL

```
COPY CMQXYZV REPLACING -3 BY 1208.
```

PL/I uso

```
%MQCCSI_APPL = '1208';
%include syslib(cmqp);
```

High Level Assembler uso

```
MQCCSI_APPL EQU 1208
CMQA LIST=NO
```

MQCIH - CICS bridge encabezado

La estructura MQCIH describe la información del encabezado de un mensaje enviado a CICS a través de CICS bridge.

Para cualquier IBM MQ plataforma compatible, puede crear y transmitir un mensaje que incluya la estructura MQCIH, pero solo un IBM MQ for z/OS gestor de colas puede utilizar el CICS bridge. Por lo tanto, para que el mensaje llegue a CICS desde un gestor de colasz/OS que no sea el suyo, su red de gestores de colas debe incluir al menos un z/OS gestor de colas a través del cual se pueda enrutar el mensaje.

Todas las CICS versiones compatibles con IBM MQ 9.0.0, y posteriores utilizan la versión CICS suministrada del puente. Para obtener más información sobre cómo configurar el adaptador de IBM MQ CICS y los componentes de IBM MQ CICS bridge, consulte la sección [Configuración de conexiones con MQ](#) de la documentación de CICS.

Disponibilidad

La estructura MQCIH está disponible en las siguientes plataformas:

-  AIX
-  Linux
-  Windows
-  z/OS

y para IBM MQ MQI clientes conectados a estos sistemas.

Nombre de formato

MQFMT_CICS

Versión

La versión actual de MQCIH es MQCIH_VERSION_2. Los campos que solo existen en la versión más reciente de la estructura se identifican como tales en las descripciones que siguen.

Los archivos de encabezado, COPY e INCLUDE proporcionados para los lenguajes de programación compatibles contienen la versión más reciente de MQCIH, con el valor inicial del *Version* campo establecido en MQCIH_VERSION_2.

Conjunto de caracteres y codificación

Se aplican condiciones especiales al conjunto de caracteres y la codificación utilizados para la estructura MQCIH y los datos de los mensajes de aplicación:

- Las aplicaciones que se conectan al gestor de colas propietario de CICS bridge la cola deben proporcionar una estructura MQCIH que esté en el juego de caracteres y la codificación del gestor de colas. Esto se debe a que, en este caso, no se realiza la conversión de datos de la estructura MQCIH.
- Las aplicaciones que se conectan a otros gestores de colas pueden proporcionar una estructura MQCIH en cualquiera de los conjuntos de caracteres y codificaciones compatibles; el agente del canal de mensajes receptor conectado al gestor de colas propietario de la CICS bridge cola convierte la estructura MQCIH.
- Los datos del mensaje de aplicación que siguen la estructura MQCIH deben estar en el mismo conjunto de caracteres y codificación que la estructura MQCIH. No se pueden utilizar los campos *CodedCharSetId* *Encoding* y de la estructura MQCIH para especificar el juego de caracteres y la codificación de los datos del mensaje de la aplicación.

Debe proporcionar una salida de conversión de datos para convertir los datos del mensaje de la aplicación si los datos no están en uno de los formatos integrados compatibles con el gestor de colas.

Utilización

Si la aplicación requiere valores que sean iguales a los valores iniciales mostrados en [Tabla 481 en la página 250](#), y el puente se está ejecutando con AUTH=LOCAL o AUTH=IDENTIFY, puede omitir la estructura MQCIH del mensaje. En todos los demás casos, la estructura debe estar presente.

El puente acepta una estructura MQCIH de tipo « version-1 » (entrada de datos) o « version-2 » (entrada de datos), pero para las transacciones 3270, debe utilizar una estructura « version-2 » (entrada de datos).

La aplicación debe garantizar que los campos documentados como campos de solicitud tengan valores adecuados en el mensaje enviado al puente; estos campos se introducen en el puente.

Los campos documentados como campos de respuesta son establecidos por el CICS bridge en el mensaje de respuesta que el puente envía a la aplicación. La información sobre errores se devuelve en los campos *ReturnCode*, *Function*, *CompCodeAbendCode*, *Reason* y, pero no todos ellos se establecen en todos los casos. La siguiente tabla muestra qué campos se establecen para diferentes valores de *ReturnCode*.

Tabla 480. Contenido de los campos de información de error en la estructura MQCIH para MQCIH				
ReturnCode	Function	CompCode	Reason	AbendCode
MQCRC_OK	-	-	-	-
ERROR_PUENTE_MQCRC	-	-	MQFB_CICS_*	-
MQCRC_MQ_API_ERROR MQCRC_BRIDGE_TIMEOUT	MQ nombre de llamada	MQ CompCode	MQ Reason	-
MQCRC_CICS_ERROR_DE_EJECUCIÓN MQCRC_ERROR_DE_SEGURIDAD MQCRC_PROGRAMA_NO_DISPONIBLE MQCRC_TRANSID_NO_DISPONIBLE	CICS EIBFN	CICS EIBRESP	CICS EIBRESP2	-

Tabla 480. Contenido de los campos de información de error en la estructura MQCIH para MQCIH (continuación)				
ReturnCode	Function	CompCode	Reason	AbendCode
MQCRC_BRIDGE_ABEND MQCRC_APPLICATION_ABEND	-	-	-	CICS ABCODE

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 481. Campos en MQCIH		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>StrucId</u> (identificador de estructura)	MQCIH_STRUC_ID	'CIH~'
<u>Versión</u> (número de versión de la estructura)	MQCIH_VERSION_2	2
<u>StrucLength</u> (longitud de la estructura MQCIH)	MQCIH_LENGTH_2	180
<u>Codificación</u> (reservado)	Ninguna	0
<u>CodedCharSetId</u> (reservado)	Ninguna	0
<u>Formato</u> (nombre del formato MQ e de los datos que siguen a MQCIH)	MQFMT_NONE	Espacios en blanco
<u>Banderas</u> (banderas)	MQCIH_NINGUNO	0
<u>ReturnCode</u> (código de retorno del puente)	MQCRC_OK	0
<u>CompCode</u> (código de finalización MQ o CICS EIBRESP)	MQCC_OK	0
<u>Motivo</u> (MQ reason o código de comentarios, o CICS EIBRESP2)	MQRC_NONE	0
<u>UOWControl</u> (control de unidad de trabajo)	MQCUOWC_SOLO	273
<u>GetWaitInterval</u> (intervalo de espera para la llamada MQGET emitida por la tarea puente)	MQCGWI_DEFAULT	-2
<u>LinkType</u> (tipo de enlace)	PROGRAMA_MQCLT	1
<u>OutputDataLength</u> (longitud de los datos COMMAREA de salida)	MQCODL_AS_ENTRADA	-1
<u>FacilityKeepTime</u> (hora de liberación de la instalación puente)	Ninguna	0
<u>ADSDescriptor</u> (enviar/recibir descriptor ADS)	MQCADSD_NINGUNO	0
<u>ConversationalTask</u> (si la tarea puede ser conversacional)	MQCCT_NO	0
<u>TaskEndStatus</u> (estado al final de la tarea)	MQCTES_NO SINCRONIZADO	0
<u>Instalación</u> (fichas de instalación puente)	MQCFAC_NINGUNO	Nulos
<u>Función</u> (nombre de llamada MQ o función CICS EIBFN)	MQCFUNC_NONE	Espacios en blanco
<u>AbendCode</u> (código abend)	Ninguna	Espacios en blanco

Tabla 481. Campos en MQCIH (continuación)

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>Autenticador</u> (contraseña o ticket de acceso)	Ninguna	Espacios en blanco
<u>Reserved1</u> (reservado)	Ninguna	Espacios en blanco
<u>ReplyToFormat</u> (Nombre del formato MQ e del mensaje de respuesta)	MQFMT_NONE	Espacios en blanco
<u>RemoteSysId</u> (Id. del sistema CICS remoto que se va a utilizar)	Ninguna	Espacios en blanco
<u>RemoteTransId</u> (CICS RTRANSID para usar)	Ninguna	Espacios en blanco
<u>TransactionId</u> (transacción para adjuntar)	Ninguna	Espacios en blanco
<u>FacilityLike</u> (atributos emulados por terminal)	Ninguna	Espacios en blanco
<u>AttentionId</u> (Clave AID)	Ninguna	Espacios en blanco
<u>StartCode</u> (código de inicio de transacción)	MQCSC_NINGUNO	Espacios en blanco
<u>CancelCode</u> (código de transacción abend)	Ninguna	Espacios en blanco
<u>NextTransactionId</u> (próxima transacción que se adjuntará)	Ninguna	Espacios en blanco
<u>Reserved2</u> (reservado)	Ninguna	Espacios en blanco
<u>Reserved3</u> (reservado)	Ninguna	Espacios en blanco
Nota: Los campos restantes no están presentes si <i>Version</i> es menor que MQCIH_VERSION_2.		
<u>CursorPosition</u> (posición del cursor)	Ninguna	0
<u>ErrorOffset</u> (desviación del error en el mensaje)	Ninguna	0
<u>InputItem</u> (elemento de entrada)	Ninguna	0
<u>Reserved4</u> (reservado)	Ninguna	0
Notas: 1. El símbolo ~ representa un único carácter en blanco. 2. En el lenguaje de programación C, la variable macroMQCIH_DEFAULT contiene los valores que se enumeran en la tabla. Úselo de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQCIH MyCIH = {MQCIH_DEFAULT};</pre>		

Declaraciones lingüísticas

Declaración C para MQCIH

```
typedef struct tagMQCIH MQCIH;
struct tagMQCIH {
    MQCHAR4  StrucId;           /* Structure identifier */
    MQLONG   Version;          /* Structure version number */
    MQLONG   StrucLength;      /* Length of MQCIH structure */
    MQLONG   Encoding;         /* Reserved */
    MQLONG   CodedCharSetId;    /* Reserved */
    MQCHAR8  Format;           /* MQ format name of data that follows
                               MQCIH */
    MQLONG   Flags;            /* Flags */
    MQLONG   ReturnCode;       /* Return code from bridge */
}
```

```

MQLONG    CompCode;           /* MQ completion code or CICS EIBRESP */
MQLONG    Reason;             /* MQ reason or feedback code, or CICS
                               EIBRESP2 */
MQLONG    UOWControl;         /* Unit-of-work control */
MQLONG    GetWaitInterval;    /* Wait interval for MQGET call issued
                               by bridge task */
MQLONG    LinkType;           /* Link type */
MQLONG    OutputDataLength;   /* Output COMMAREA data length */
MQLONG    FacilityKeepTime;   /* Bridge facility release time */
MQLONG    ADSDescriptor;      /* Send/receive ADS descriptor */
MQLONG    ConversationalTask; /* Whether task can be conversational */
MQLONG    TaskEndStatus;      /* Status at end of task */
MQBYTE8    Facility;          /* Bridge facility token */
MQCHAR4    Function;          /* MQ call name or CICS EIBFN
                               function */
MQCHAR4    AbendCode;         /* Abend code */
MQCHAR8    Authenticator;     /* Password or passticket */
MQCHAR8    Reserved1;         /* Reserved */
MQCHAR8    ReplyToFormat;     /* MQ format name of reply message */
MQCHAR4    RemoteSysId;       /* Reserved */
MQCHAR4    RemoteTransId;     /* Reserved */
MQCHAR4    TransactionId;     /* Transaction to attach */
MQCHAR4    FacilityLike;      /* Terminal emulated attributes */
MQCHAR4    AttentionId;       /* AID key */
MQCHAR4    StartCode;         /* Transaction start code */
MQCHAR4    CancelCode;        /* Abend transaction code */
MQCHAR4    NextTransactionId; /* Next transaction to attach */
MQCHAR8    Reserved2;         /* Reserved */
MQCHAR8    Reserved3;         /* Reserved */
MQLONG    CursorPosition;     /* Cursor position */
MQLONG    ErrorOffset;        /* Offset of error in message */
MQLONG    InputItem;          /* Reserved */
MQLONG    Reserved4;          /* Reserved */
};

```

COBOL declaración para MQCIH

```

**  MQCIH structure
10 MQCIH.
**    Structure identifier
15 MQCIH-STRUCID          PIC X(4).
**    Structure version number
15 MQCIH-VERSION         PIC S9(9) BINARY.
**    Length of MQCIH structure
15 MQCIH-STRUCLNGTH     PIC S9(9) BINARY.
**    Reserved
15 MQCIH-ENCODING        PIC S9(9) BINARY.
**    Reserved
15 MQCIH-CODEDCHARSETID  PIC S9(9) BINARY.
**    MQ format name of data that follows MQCIH
15 MQCIH-FORMAT          PIC X(8).
**    Flags
15 MQCIH-FLAGS           PIC S9(9) BINARY.
**    Return code from bridge
15 MQCIH-RETURNCODE      PIC S9(9) BINARY.
**    MQ completion code or CICS EIBRESP
15 MQCIH-COMPCODE        PIC S9(9) BINARY.
**    MQ reason or feedback code, or CICS EIBRESP2
15 MQCIH-REASON          PIC S9(9) BINARY.
**    Unit-of-work control
15 MQCIH-UOWCONTROL      PIC S9(9) BINARY.
**    Wait interval for MQGET call issued by bridge task
15 MQCIH-GETWAITINTERVAL PIC S9(9) BINARY.
**    Link type
15 MQCIH-LINKTYPE        PIC S9(9) BINARY.
**    Output COMMAREA data length
15 MQCIH-OUTPUTDATALENGTH PIC S9(9) BINARY.
**    Bridge facility release time
15 MQCIH-FACILITYKEEPTIME PIC S9(9) BINARY.
**    Send/receive ADS descriptor
15 MQCIH-ADSDESCRIPTOR   PIC S9(9) BINARY.
**    Whether task can be conversational
15 MQCIH-CONVERSATIONALTASK PIC S9(9) BINARY.
**    Status at end of task
15 MQCIH-TASKENDSTATUS   PIC S9(9) BINARY.
**    Bridge facility token
15 MQCIH-FACILITY        PIC X(8).
**    MQ call name or CICS EIBFN function
15 MQCIH-FUNCTION        PIC X(4).
**    Abend code

```



```

15 MQCIH-ABENDCODE          PIC X(4).
** Password or passticket
15 MQCIH-AUTHENTICATOR      PIC X(8).
** Reserved
15 MQCIH-RESERVED1          PIC X(8).
** MQ format name of reply message
15 MQCIH-REPLYTOFORMAT      PIC X(8).
** Reserved
15 MQCIH-REMOTESYSID        PIC X(4).
** Reserved
15 MQCIH-REMOtetransid      PIC X(4).
** Transaction to attach
15 MQCIH-TRANSACTIONID      PIC X(4).
** Terminal emulated attributes
15 MQCIH-FACILITYLIKE        PIC X(4).
** AID key
15 MQCIH-ATTENTIONID        PIC X(4).
** Transaction start code
15 MQCIH-STARTCODE          PIC X(4).
** Abend transaction code
15 MQCIH-CANCELCODE          PIC X(4).
** Next transaction to attach
15 MQCIH-NEXTTRANSACTIONID  PIC X(4).
** Reserved
15 MQCIH-RESERVED2          PIC X(8).
** Reserved
15 MQCIH-RESERVED3          PIC X(8).
** Cursor position
15 MQCIH-CURSORPOSITION      PIC S9(9) BINARY.
** Offset of error in message
15 MQCIH-ERROROFFSET        PIC S9(9) BINARY.
** Reserved
15 MQCIH-INPUTITEM          PIC S9(9) BINARY.
** Reserved
15 MQCIH-RESERVED4          PIC S9(9) BINARY.

```

PL/I declaración para MQCIH

```

dcl
1 MQCIH based,
3 StructId          char(4),          /* Structure identifier */
3 Version           fixed bin(31),    /* Structure version number */
3 StructLength      fixed bin(31),    /* Length of MQCIH structure */
3 Encoding          fixed bin(31),    /* Reserved */
3 CodedCharSetId    fixed bin(31),    /* Reserved */
3 Format             char(8),          /* MQ format name of data that
                                     follows MQCIH */
3 Flags             fixed bin(31),    /* Flags */
3 ReturnCode        fixed bin(31),    /* Return code from bridge */
3 CompCode          fixed bin(31),    /* MQ completion code or CICS
                                     EIBRESP */
3 Reason            fixed bin(31),    /* MQ reason or feedback code, or
                                     CICS EIBRESP2 */
3 UOWControl        fixed bin(31),    /* Unit-of-work control */
3 GetWaitInterval   fixed bin(31),    /* Wait interval for MQGET call
                                     issued by bridge task */
3 LinkType          fixed bin(31),    /* Link type */
3 OutputDataLength  fixed bin(31),    /* Output COMMAREA data length */
3 FacilityKeepTime  fixed bin(31),    /* Bridge facility release time */
3 ADSDescriptor     fixed bin(31),    /* Send/receive ADS descriptor */
3 ConversationalTask fixed bin(31),    /* Whether task can be
                                     conversational */
3 TaskEndStatus     fixed bin(31),    /* Status at end of task */
3 Facility          char(8),          /* Bridge facility token */
3 Function          char(4),          /* MQ call name or CICS EIBFN
                                     function */
3 AbendCode         char(4),          /* Abend code */
3 Authenticator      char(8),          /* Password or passticket */
3 Reserved1         char(8),          /* Reserved */
3 ReplyToFormat     char(8),          /* MQ format name of reply
                                     message */
3 RemoteSysId       char(4),          /* Reserved */
3 RemoteTransId     char(4),          /* Reserved */
3 TransactionId      char(4),          /* Transaction to attach */
3 FacilityLike      char(4),          /* Terminal emulated attributes */
3 AttentionId       char(4),          /* AID key */
3 StartCode         char(4),          /* Transaction start code */
3 CancelCode        char(4),          /* Abend transaction code */
3 NextTransactionId char(4),          /* Next transaction to attach */
3 Reserved2         char(8),          /* Reserved */

```

3 Reserved3	char(8),	/* Reserved */
3 CursorPosition	fixed bin(31),	/* Cursor position */
3 ErrorOffset	fixed bin(31),	/* Offset of error in message */
3 InputItem	fixed bin(31),	/* Reserved */
3 Reserved4	fixed bin(31);	/* Reserved */

High Level Assembler declaración para MQCIH

MQCIH	DSECT	
MQCIH_STRUCID	DS	CL4 Structure identifier
MQCIH_VERSION	DS	F Structure version number
MQCIH_STRUCLNGTH	DS	F Length of MQCIH structure
MQCIH_ENCODING	DS	F Reserved
MQCIH_CODEDCHARSETID	DS	F Reserved
MQCIH_FORMAT	DS	CL8 MQ format name of data that follows
*		MQCIH
MQCIH_FLAGS	DS	F Flags
MQCIH_RETURNCODE	DS	F Return code from bridge
MQCIH_COMPCODE	DS	F MQ completion code or CICS EIBRESP
MQCIH_REASON	DS	F MQ reason or feedback code, or CICS
*		EIBRESP2
MQCIH_UOWCONTROL	DS	F Unit-of-work control
MQCIH_GETWAITINTERVAL	DS	F Wait interval for MQGET call issued
*		by bridge task
MQCIH_LINKTYPE	DS	F Link type
MQCIH_OUTPUTDATALENGTH	DS	F Output COMMAREA data length
MQCIH_FACILITYKEEPTIME	DS	F Bridge facility release time
MQCIH_ADSDESCRIPTOR	DS	F Send/receive ADS descriptor
MQCIH_CONVERSATIONALTASK	DS	F Whether task can be conversational
MQCIH_TASKENDSTATUS	DS	F Status at end of task
MQCIH_FACILITY	DS	XL8 Bridge facility token
MQCIH_FUNCTION	DS	CL4 MQ call name or CICS EIBFN function
MQCIH_ABENDCODE	DS	CL4 Abend code
MQCIH_AUTHENTICATOR	DS	CL8 Password or passticket
MQCIH_RESERVED1	DS	CL8 Reserved
MQCIH_REPLYTOFORMAT	DS	CL8 MQ format name of reply message
MQCIH_REMOTESYSID	DS	CL4 Reserved
MQCIH_REMOTETRANSID	DS	CL4 Reserved
MQCIH_TRANSACTIONID	DS	CL4 Transaction to attach
MQCIH_FACILITYLIKE	DS	CL4 Terminal emulated attributes
MQCIH_ATTENTIONID	DS	CL4 AID key
MQCIH_STARTCODE	DS	CL4 Transaction start code
MQCIH_CANCELCODE	DS	CL4 Abend transaction code
MQCIH_NEXTTRANSACTIONID	DS	CL4 Next transaction to attach
MQCIH_RESERVED2	DS	CL8 Reserved
MQCIH_RESERVED3	DS	CL8 Reserved
MQCIH_CURSORPOSITION	DS	F Cursor position
MQCIH_ERROROFFSET	DS	F Offset of error in message
MQCIH_INPUTITEM	DS	F Reserved
MQCIH_RESERVED4	DS	F Reserved
*		
MQCIH_LENGTH	EQU	*-MQCIH
	ORG	MQCIH
MQCIH_AREA	DS	CL(MQCIH_LENGTH)

Declaración de Visual Basic para MQCIH

Type MQCIH		
StrucId	As String*4	'Structure identifier'
Version	As Long	'Structure version number'
StrucLength	As Long	'Length of MQCIH structure'
Encoding	As Long	'Reserved'
CodedCharSetId	As Long	'Reserved'
Format	As String*8	'MQ format name of data that follows'
		'MQCIH'
Flags	As Long	'Flags'
ReturnCode	As Long	'Return code from bridge'
CompCode	As Long	'MQ completion code or CICS EIBRESP'
Reason	As Long	'MQ reason or feedback code, or CICS'
		'EIBRESP2'
UOWControl	As Long	'Unit-of-work control'
GetWaitInterval	As Long	'Wait interval for MQGET call issued'
		'by bridge task'
LinkType	As Long	'Link type'
OutputDataLength	As Long	'Output COMMAREA data length'
FacilityKeepTime	As Long	'Bridge facility release time'
ADSDescriptor	As Long	'Send/receive ADS descriptor'
ConversationalTask	As Long	'Whether task can be conversational'
TaskEndStatus	As Long	'Status at end of task'

```

Facility      As MQBYTE8  'Bridge facility token'
Function      As String*4  'MQ call name or CICS EIBFN function'
AbendCode     As String*4  'Abend code'
Authenticator As String*8  'Password or passticket'
Reserved1     As String*8  'Reserved'
ReplyToFormat As String*8  'MQ format name of reply message'
RemoteSysId   As String*4  'Reserved'
RemoteTransId As String*4  'Reserved'
TransactionId As String*4  'Transaction to attach'
FacilityLike  As String*4  'Terminal emulated attributes'
AttentionId   As String*4  'AID key'
StartCode     As String*4  'Transaction start code'
CancelCode    As String*4  'Abend transaction code'
NextTransactionId As String*4 'Next transaction to attach'
Reserved2     As String*8  'Reserved'
Reserved3     As String*8  'Reserved'
CursorPosition As Long    'Cursor position'
ErrorOffset   As Long    'Offset of error in message'
InputItem     As Long    'Reserved'
Reserved4     As Long    'Reserved'
End Type

```

Detalles de campo para MQCIH

La estructura MQCIH describe la información del encabezado de un mensaje enviado a CICS a través de CICS bridge.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 482. Campos en MQCIH para MQCIH		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
StrucId (identificador de estructura)	MQCIH_STRUC_ID	'CIH'
Versión (número de versión de la estructura)	MQCIH_VERSION_2	2
StrucLength (longitud de la estructura MQCIH)	MQCIH_LENGTH_2	180
Codificación (reservado)	Ninguna	0
CodedCharSetId (reservado)	Ninguna	0
Formato (nombre del formato MQ e de los datos que siguen a MQCIH)	MQFMT_NONE	Espacios en blanco
Banderas (banderas)	MQCIH_NINGUNO	0
ReturnCode (código de retorno del puente)	MQCRC_OK	0
CompCode (código de finalización MQ o CICS EIBRESP)	MQCC_OK	0
Motivo (MQ reason o código de comentarios, o CICS EIBRESP2)	MQRC_NONE	0
UOWControl (control de unidad de trabajo)	MQCUOWC_SOLO	273
GetWaitInterval (intervalo de espera para la llamada MQGET emitida por la tarea puente)	MQCGWI_DEFAULT	-2
LinkType (tipo de enlace)	PROGRAMA_MQCLT	1
OutputDataLength (longitud de los datos COMMAREA de salida)	MQCODL_AS_ENTRADA	-1
FacilityKeepTime (hora de liberación de la instalación puente)	Ninguna	0

Tabla 482. Campos en MQCIH para MQCIH (continuación)

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>ADSDescriptor</u> (enviar/recibir descriptor ADS)	MQCADSD_NINGUNO	0
<u>ConversationalTask</u> (si la tarea puede ser conversacional)	MQCCT_NO	0
<u>TaskEndStatus</u> (estado al final de la tarea)	MQCTES_NO SINCRONIZADO	0
<u>Instalación</u> (fichas de instalación puente)	MQCFAC_NINGUNO	Nulos
<u>Función</u> (nombre de llamada MQ o función CICS EIBFN)	MQCFUNC_NONE	Espacios en blanco
<u>AbendCode</u> (código abend)	Ninguna	Espacios en blanco
<u>Autenticador</u> (contraseña o ticket de acceso)	Ninguna	Espacios en blanco
<u>Reserved1</u> (reservado)	Ninguna	Espacios en blanco
<u>ReplyToFormat</u> (Nombre del formato MQ e del mensaje de respuesta)	MQFMT_NONE	Espacios en blanco
<u>RemoteSysId</u> (Id. del sistema CICS remoto que se va a utilizar)	Ninguna	Espacios en blanco
<u>RemoteTransId</u> (CICS RTRANSID para usar)	Ninguna	Espacios en blanco
<u>TransactionId</u> (transacción para adjuntar)	Ninguna	Espacios en blanco
<u>FacilityLike</u> (atributos emulados por terminal)	Ninguna	Espacios en blanco
<u>AttentionId</u> (Clave AID)	Ninguna	Espacios en blanco
<u>StartCode</u> (código de inicio de transacción)	MQCSC_NINGUNO	Espacios en blanco
<u>CancelCode</u> (código de transacción abend)	Ninguna	Espacios en blanco
<u>NextTransactionId</u> (próxima transacción que se adjuntará)	Ninguna	Espacios en blanco
<u>Reserved2</u> (reservado)	Ninguna	Espacios en blanco
<u>Reserved3</u> (reservado)	Ninguna	Espacios en blanco
Nota: Los campos restantes no están presentes si <i>Version</i> es menor que MQCIH_VERSION_2.		
<u>CursorPosition</u> (posición del cursor)	Ninguna	0
<u>ErrorOffset</u> (desviación del error en el mensaje)	Ninguna	0
<u>InputItem</u> (elemento de entrada)	Ninguna	0
<u>Reserved4</u> (reservado)	Ninguna	0
Notas: 1. El símbolo ~ representa un único carácter en blanco. 2. En el lenguaje de programación C, la variable macroMQCIH_DEFAULT contiene los valores que se enumeran en la tabla. Úselo de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQCIH MyCIH = {MQCIH_DEFAULT};</pre>		

StrucId (MQCHAR4)

Este es el identificador de estructura de la estructura CICS de cabecera de información. Siempre es un campo de entrada. Su valor es MQCIH_STRUC_ID.

El valor debe ser:

MQCIH_STRUC_ID

Identificador de la estructura del CICS encabezado de información.

Para el lenguaje de programación C, también se define la constante MQCIH_STRUC_ID_ARRAY. Tiene el mismo valor que MQCIH_STRUC_ID, pero es una matriz de caracteres en lugar de una cadena.

Versión (MQLONG)

Este campo es un campo de solicitud. Su valor inicial es MQCIH_VERSION_2.

El valor debe ser uno de los siguientes:

MQCIH_VERSION_1

Version-1 Estructura del CICS encabezado de información.

MQCIH_VERSION_2

Version-2 Estructura del CICS encabezado de información.

Los campos que solo existen en la versión más reciente de la estructura se identifican como tales en las descripciones de los campos. La siguiente constante especifica el número de versión de la versión actual:

MQCIH_VERSIÓN_ACTUAL

Versión actual de la estructura del CICS encabezado de información.

StrucLength (MQLONG)

Este campo es un campo de solicitud, con un valor inicial de MQCIH_LENGTH_2.

El valor debe ser uno de los siguientes:

MQCIH_LENGTH_1

Longitud de la estructura del encabezado de CICS información de version-1.

MQCIH_LENGTH_2

Longitud de la estructura del encabezado de CICS información de version-2.

La siguiente constante especifica la longitud de la versión actual:

MQCIH_LONGITUD_ACTUAL

Longitud de la versión actual de la estructura del CICS encabezado de información.

Encoding (MQLONG)

Este campo es un campo reservado. Su valor no es significativo. Su valor inicial es 0.

La codificación de las estructuras compatibles que siguen una estructura MQCIH es la misma que la codificación de la propia estructura MQCIH y se toma de cualquier encabezado IBM MQ anterior.

CodedCharSetId (MQLONG)

CodedCharSetId es un campo reservado. Su valor no es significativo. El valor inicial de este campo es 0.

El identificador del conjunto de caracteres para las estructuras compatibles que siguen una estructura MQCIH es el mismo que el identificador del conjunto de caracteres de la propia estructura MQCIH y se toma de cualquier encabezado IBM MQ anterior.

Format (MQCHAR8)

Este campo muestra el nombre IBM MQ del formato de los datos que siguen la estructura MQCIH.

En la llamada MQPUT o MQPUT1, la aplicación debe establecer este campo con el valor adecuado para los datos. Las reglas para codificar este campo son las mismas que las reglas para codificar el *Format* campo en MQMD.

Este nombre de formato también se utiliza para el mensaje de respuesta, si el *ReplyToFormat* campo tiene el valor MQFMT_NONE.

- Para las solicitudes DPL, *Format* debe ser el nombre del formato de COMMAREA.
- Para 3270 solicitudes, *Format* debe ser CSQCBDCI, y el puente establece el formato en CSQCBDCO para los mensajes de respuesta.

Las salidas de conversión de datos para estos formatos deben instalarse en el gestor de colas donde se van a ejecutar.

Si el mensaje de solicitud genera un mensaje de respuesta de error, el mensaje de respuesta de error tiene un nombre de formato MQFMT_STRING.

Este campo es un campo de solicitud. La longitud de este campo viene dada por MQ_FORMAT_LENGTH. El valor inicial de este campo es MQFMT_NONE.

Flags (MQLONG)

Este campo es un campo de solicitud. El valor inicial de este campo es MQCIH_NONE.

El valor debe ser:

MQCIH_NINGUNO

Sin banderas.

MQCIH_PLAZO_DE_VIGENCIA

El mensaje de respuesta contiene:

- Las mismas opciones de informe de caducidad que el mensaje de solicitud.
- El tiempo restante hasta la expiración del mensaje de solicitud sin ajustes por el tiempo de procesamiento del puente.

Si omite este valor, el tiempo de caducidad se establece como *ilimitado*.

MQCIH_RESPUESTA_SIN_NULOS

La longitud del mensaje de respuesta de una solicitud del CICS programa DPL se ajusta para excluir los caracteres nulos finales (X'00') al final del COMMAREA devuelto por el programa DPL. Si este valor no se establece, los valores nulos podrían ser significativos y se devolvería el COMMAREA completo.

MQCIH_SINCRONIZAR_AL_RETORNO

El CICS enlace para solicitudes DPL utiliza la opción SYNCONRETURN, lo que provoca CICS que se tome un punto de sincronización cuando el programa finaliza si se envía a otra CICS región. El puente no especifica a qué CICS región enviar la solicitud; eso lo controla la definición del CICS programa o las funciones de equilibrio de la carga de trabajo.

ReturnCode (MQLONG)

El valor de este campo es el código de retorno de la CICS bridge que describe el resultado del procesamiento realizado por el puente. Este campo es un campo de respuesta, con un valor inicial de MQCRC_OK.

Los *AbendCode* campos *Function*, *CompCode Reason*, y pueden contener información adicional (véase [Tabla 480 en la página 249](#)). El valor puede ser uno de los siguientes:

MQCRC_APLICACIÓN_ABEND

(5, X'005') La aplicación finalizó de forma anómala.

MQCRC_PUENTE_ABEND

(4, X'004') CICS bridge finalizó de forma anómala.

ERROR_PUENTE_MQCRC

(3, X'003') CICS bridge detectó un error.

MQCRC_BRIDGE_TIMEOUT

(8, X'008') Segundo mensaje o mensajes posteriores dentro de la unidad de trabajo actual no recibidos dentro del tiempo especificado.

MQCRC_CICS_EXEC_ERROR

(1, X'001') La CICS instrucción EXEC detectó un error.

MQCRC_MQ_API_ERROR

(2, X'002') MQ call detectó un error.

MQCRC_OK

(0, X'000') Sin errores.

MQCRC_PROGRAMA_NO_DISPONIBLE

(7, X'007') Programa no disponible.

MQCRC_ERROR_DE_SEGURIDAD

(6, X'006') Se ha producido un error de seguridad.

MQCRC_TRANSID_NO_DISPONIBLE

(9, X'009') Transacción no disponible.

CompCode (MQLONG)

Este campo es un campo de respuesta. Su valor inicial es MQCC_OK

El valor devuelto en este campo depende de *ReturnCode*. Consulte [Tabla 480 en la página 249](#).

Razón (MQLONG)

Este campo es un campo de respuesta. Su valor inicial es MQRC_NONE

El valor devuelto en este campo depende de *ReturnCode*. Consulte [Tabla 480 en la página 249](#).

UOWControl (MQLONG)

Este campo es un campo de solicitud que controla el procesamiento de la unidad de trabajo realizado por el CICS bridge. El valor inicial de este campo es MQCUOWC_ONLY

Puede solicitar al puente que ejecute una sola transacción o uno o varios programas dentro de una unidad de trabajo. El campo indica si el CICS bridge inicia una unidad de trabajo, realiza la función solicitada dentro de la unidad de trabajo actual o finaliza la unidad de trabajo confirmándola o cancelándola. Se admiten varias combinaciones para optimizar los flujos de transmisión de datos.

El valor debe ser uno de los siguientes:

MQCUOWC_SOLO

Inicie la unidad de trabajo, realice la función y, a continuación, confirme la unidad de trabajo.

MQCUOWC_CONTINUAR

Datos adicionales para la unidad de trabajo actual (solo 3270).

MQCUOWC_PRIMERO

Iniciar la unidad de trabajo y ejecutar la función.

MQCUOWC_MEDIO

Realizar funciones dentro de la unidad de trabajo actual

MQCUOWC_ÚLTIMO

Realiza la función y, a continuación, confirma la unidad de trabajo.

MQCUOWC_COMMIT

Confirmar la unidad de trabajo (solo DPL).

MQCUOWC_RETIRADA

Retirar la unidad de trabajo (solo DPL).

GetWaitInterval (MQLONG)

Este campo es un campo de solicitud. Su valor inicial es MQCGWI_DEFAULT

Este campo solo se aplica cuando *UOWControl* tiene el valor MQCUOWC_FIRST. Permite a la aplicación emisora especificar el tiempo aproximado en milisegundos que las llamadas MQGET emitidas por el puente esperarán por el segundo mensaje de solicitud y los mensajes de solicitud posteriores para la unidad de trabajo iniciada por este mensaje. Esta función anula el intervalo de espera predeterminado utilizado por el puente. Puede utilizar los siguientes valores especiales:

MQCGWI_DEFAULT

Intervalo de espera predeterminado.

Este valor hace que CICS bridge el espere el tiempo especificado cuando se inició el puente.

MQWI_ILIMITADO

Intervalo de espera ilimitado.

LinkType (MQLONG)

Este campo es un campo de solicitud. Su valor inicial es MQCLT_PROGRAM

Este valor indica el tipo de objeto que el puente intenta vincular. Tiene que ser uno de los valores siguientes:

PROGRAMA_MQCLT

Programa DPL.

MQCLT_TRANSACCIÓN

Transacción 3270.

OutputDataLength (MQLONG)

Este campo es un campo de solicitud que se utiliza únicamente para programas DPL. Su valor inicial es MQCODL_AS_INPUT

Este valor es la longitud de los datos de usuario que se devolverán al cliente en un mensaje de respuesta. Esta longitud incluye el nombre del programa de 8 bytes. La longitud de COMMAREA pasada al programa vinculado es el máximo de este campo y la longitud de los datos de usuario en el mensaje de solicitud, menos 8.

Nota: La longitud de los datos de usuario en un mensaje es la longitud del mensaje sin incluir la estructura MQCIH.

Si la longitud de los datos de usuario en el mensaje de solicitud es menor que *OutputDataLength*, se utiliza la DATALENGTH opción del LINK comando, lo que permite que LINK el se envíe de manera eficiente a otra CICS región.

Puede utilizar el siguiente valor especial:

MQCODL_AS_ENTRADA

La longitud de salida es igual a la longitud de entrada.

Este valor puede ser necesario incluso si no se solicita ninguna respuesta, con el fin de garantizar que el COMMAREA pasado al programa vinculado tenga un tamaño suficiente.

FacilityKeepTime (MQLONG)

FacilityKeepTime Es el tiempo en segundos que se mantiene la conexión puente después de que finaliza la transacción del usuario.

Para las transacciones pseudoconversacionales, especifique un valor que corresponda a la duración prevista de una pseudoconversación; especifique cero para la última transacción de una pseudoconversación y, para otros tipos de transacciones, especifique cero.

Este campo es un campo de solicitud que solo se utiliza para transacciones 3270. El valor inicial de este campo es 0.

Descriptor de anuncio (MQLONG)

Este campo es un indicador que especifica si se deben enviar descriptores ADS en las solicitudes BMS SEND y RECEIVE.

Los valores siguientes están definidos:

MQCADSD_NINGUNO

No envíe ni reciba descriptores ADS.

MQCADSD_ENVIAR

Enviar descriptores ADS.

MQCADSD_RECEPCIÓN

Recibir descriptores ADS.

MQCADSD_FORMATO DE MENSAJE

Utilice el formato de mensaje para los descriptores ADS.

Esto envía o recibe los descriptores ADS utilizando la forma larga del descriptor ADS. El formulario largo tiene campos alineados en límites de 4 bytes.

Configure el *ADSDDescriptor* campo de la siguiente manera:

- Si no utiliza descriptores ADS, establezca el campo en MQCADSD_NONE.
- Si utiliza descriptores ADS con el *mismo* CCSID en cada entorno, establezca el campo en la suma de MQCADSD_SEND y MQCADSD_RECV.
- Si utiliza descriptores ADS con *diferentes* CCSID en cada entorno, establezca el campo en la suma de MQCADSD_SEND, MQCADSD_RECV y MQCADSD_MSGFORMAT.

Este es un campo de solicitud que se utiliza únicamente para transacciones 3270. El valor inicial de este campo es MQCADSD_NONE.

ConversationalTask (MQLONG)

Este campo es un indicador que especifica si se permite que la tarea envíe solicitudes de más información o si se detiene la tarea y se emite un mensaje de error.

El valor debe ser una de las siguientes opciones:

MQCCT_SÍ

La tarea es conversacional.

MQCCT_NO

La tarea no es conversacional.

Este campo es un campo de solicitud que solo se utiliza para transacciones 3270. El valor inicial de este campo es MQCCT_NO

TaskEndStatus (MQLONG)

Este campo es un campo de respuesta que muestra el estado de la transacción del usuario al finalizar la tarea. El campo solo se utiliza para transacciones 3270 y su valor inicial es MQCTES_NOSYNC

Se devuelve uno de los siguientes valores:

MQCTES_NO SINCRONIZADO

No sincronizado.

La transacción del usuario aún no se ha completado y no se ha sincronizado. El *MsgType* campo en MQMD es MQMT_REQUEST en este caso.

MQCTES_COMMIT

Confirmar unidad de trabajo.

La transacción del usuario aún no se ha completado, pero se ha sincronizado la primera unidad de trabajo. El *MsgType* campo en MQMD es MQMT_DATAGRAM en este caso.

MQCTES_RETIRADA

Retirar unidad de trabajo.

La transacción del usuario aún no se ha completado. La unidad de trabajo actual se cancela. El *MsgType* campo en MQMD es MQMT_DATAGRAM en este caso.

MQCTES_FINALIZAR_TAREA

Finalizar tarea.

La transacción del usuario ha finalizado (o se ha cancelado). El *MsgType* campo en MQMD es MQMT_REPLY en este caso.

Instalaciones (MQBYTE8)

Este campo muestra el token de 8 bytes de la función puente.

Un token de puente permite que varias transacciones en una pseudoconversación utilicen el mismo puente (terminal virtual 3270). En el primer mensaje, o único mensaje, de una pseudoconversación, establezca un valor de MQCFAC_NONE. Este valor indica CICS que se debe asignar una nueva instalación puente para este mensaje. Se devuelve un token de instalación puente en los mensajes de respuesta cuando se especifica un valor *FacilityKeepTime* distinto de cero en el mensaje de entrada. Los mensajes de entrada posteriores dentro de una pseudoconversación deben utilizar entonces el mismo token de puente.

Se define el siguiente valor especial:

MQCFAC_NINGUNO

No se ha especificado ningún token de instalación.

Para el lenguaje de programación C, también se define la constante MQCFAC_NONE_ARRAY, que tiene el mismo valor que MQCFAC_NONE, pero es una matriz de caracteres en lugar de una cadena.

Este campo es tanto un campo de solicitud como de respuesta que se utiliza únicamente para transacciones 3270. La longitud de este campo viene dada por MQ_FACILITY_LENGTH. El valor inicial de este campo es MQCFAC_NONE.

Función (MQCHAR4)

Este campo es un campo de respuesta. La longitud de este campo viene dada por MQ_FUNCTION_LENGTH. El valor inicial de este campo es MQCFUNC_NONE

El valor devuelto en este campo depende de *ReturnCode* ; véase [Tabla 480 en la página 249](#). Los siguientes valores son posibles cuando *Function* contiene un nombre IBM MQ de llamada:

MQCFUNC_MQCONN

Llamada MQCONN.

MQCFUNC_MQGET

Llamada MQGET.

MQCFUNC_MQINQ

Llamada MQINQ.

MQCFUNC_MQOPEN

Llamada MQOPEN.

MQCFUNC_MQPUT

Llamada MQPUT.

MQCFUNC_MQPUT1

MQPUT1 llamada.

MQCFUNC_NONE

No hay llamada.

En todos los casos, para el lenguaje de programación C también se definen las constantes MQCFUNC_*_ARRAY; estas constantes tienen los mismos valores que las constantes MQCFUNC_* correspondientes, pero son matrices de caracteres en lugar de cadenas.

AbendCode (MQCHAR4)

AbendCode es un campo de respuesta. La longitud de este campo viene dada por MQ_ABEND_CODE_LENGTH. El valor inicial de este campo es 4 caracteres en blanco.

El valor devuelto en este campo solo es significativo si el *ReturnCode* campo tiene el valor MQCRC_APPLICATION_ABEND o MQCRC_BRIDGE_ABEND. Si es así, *AbendCode* contiene el valor CICS ABCODE.

Autenticador (MQCHAR8)

El valor de este campo es la contraseña o el ticket de acceso.

Si la autenticación por identificador de usuario está activa para el CICS bridge, *Authenticator* se utiliza con el identificador de usuario en el contexto de identidad MQMD para autenticar al remitente del mensaje.

Este es un campo de solicitud. La longitud de este campo viene dada por MQ_AUTHENTICATOR_LENGTH. El valor inicial de este campo es 8 espacios en blanco.

Reserved1 (MQCHAR8)

Este campo es un campo reservado. El valor debe ser 8 espacios en blanco.

ReplyToFormat (MQCHAR8)

El valor de este campo es el nombre IBM MQ del formato del mensaje de respuesta que se envía en respuesta al mensaje actual.

Las reglas para codificar este campo son las mismas que las reglas para codificar el *Format* campo en MQMD.

Este campo es un campo de solicitud que se utiliza únicamente para programas DPL. La longitud de este campo viene dada por MQ_FORMAT_LENGTH. El valor inicial de este campo es MQFMT_NONE.

RemoteSysId (MQCHAR4)

Este campo muestra el CICS identificador del CICS sistema que procesa la solicitud.

Si este campo está en blanco, la CICS solicitud del sistema se procesa en el mismo CICS sistema que el monitor puente. El SYSID utilizado se devuelve en el mensaje de respuesta.

Para una pseudoconversación 3270, todos los mensajes posteriores de la conversación deben especificar el SYSID remoto devuelto en la respuesta inicial. Si se especifica, el SYSID debe:

- Mantente activo.
- Tener acceso a la cola IBM MQ de solicitudes.
- Accesible mediante los enlaces CICS ISC desde el CICS sistema del monitor del puente.

RemoteTransId (MQCHAR4)

Este campo es un campo de solicitud opcional. La longitud de este campo viene dada por MQ_TRANSACTION_ID_LENGTH

Si se especifica, el campo se utiliza como valor RTRANSID de un START d CICS.

TransactionId (MQCHAR4)

Este campo es un campo de solicitud. Su longitud viene dada por MQ_TRANSACTION_ID_LENGTH. El valor inicial de este campo es cuatro espacios en blanco.

Si *LinkType* tiene el valor MQCLT_TRANSACTION, *TransactionId* es el identificador de la transacción de usuario que se va a ejecutar. Especifique un valor que no esté en blanco en este caso.

Si *LinkType* tiene el valor MQCLT_PROGRAM, *TransactionId* es el código de transacción bajo el cual se ejecutarán todos los programas dentro de la unidad de trabajo. Si se especifica un valor en blanco, se utiliza el código de transacción CICS predeterminado del puente DPL (CKBP). Si el valor no está en blanco, debe haberlo definido como CICS una transacción local con un programa inicial que sea CSQCBP00. Este campo solo se aplica cuando *UOWControl* tiene el valor MQCUOWC_FIRST o MQCUOWC_ONLY.

FacilityLike (MQCHAR4)

FacilityLike es el nombre de un terminal instalado que se utilizará como modelo para la instalación del puente.

Un valor en blanco significa que *FacilityLike* se toma de la definición del perfil de transacción del puente, o se utiliza un valor predeterminado.

Este campo es un campo de solicitud que solo se utiliza para transacciones 3270. La longitud de este campo viene dada por MQ_FACILITY_LIKE_LENGTH. El valor inicial de este campo es cuatro espacios en blanco.

AttentionId (MQCHAR4)

El valor de este campo determina el valor inicial de la clave AID cuando se inicia la transacción. Es un valor de 1 byte, alineado a la izquierda.

AttentionId Es un campo de solicitud que se utiliza únicamente para transacciones 3270. La longitud de este campo viene dada por MQ_ATTENTION_ID_LENGTH. El valor inicial de este campo es cuatro espacios en blanco.

StartCode (MQCHAR4)

El valor de este campo es un indicador que especifica si el puente emula una transacción de terminal o una transacción iniciada con START.

El valor debe ser uno de los siguientes:

MQCSC_INICIO

Inicio.

MQCSC_INICIO_DATOS

Datos iniciales.

MQCSC_TERMINPUT

Entrada de terminal.

MQCSC_NINGUNO

Ninguna.

En todos los casos, para el lenguaje de programación C también se definen las constantes MQCSC_*_ARRAY; estas constantes tienen los mismos valores que las constantes MQCSC_* correspondientes, pero son matrices de caracteres en lugar de cadenas.

En la respuesta del puente, este campo se establece en el código de inicio adecuado para el siguiente ID de transacción contenido en el *NextTransactionId* campo. Los siguientes códigos de inicio son posibles en la respuesta:

- MQCSC_INICIO
- MQCSC_INICIO_DATOS

- MQCSC_TERMINPUT

Para CICS Transaction Server 1.2, este campo es solo un campo de solicitud; su valor en la respuesta es indefinido.

Para CICS Transaction Server 1.3 y versiones posteriores, este campo es tanto un campo de solicitud como de respuesta.

Este campo solo se utiliza para transacciones 3270. La longitud de este campo viene dada por MQ_START_CODE_LENGTH. El valor inicial de este campo es MQCSC_NONE.

CancelCode (MQCHAR4)

El valor de este campo es el código de interrupción que se utilizará para terminar la transacción (normalmente una transacción conversacional que solicita más datos). De lo contrario, este campo se deja en blanco.

Este campo es un campo de solicitud que solo se utiliza para transacciones 3270. La longitud de este campo viene dada por MQ_CANCEL_CODE_LENGTH. El valor inicial de este campo es cuatro espacios en blanco.

NextTransactionId (MQCHAR4)

Este valor es el nombre de la siguiente transacción devuelta por la transacción del usuario (normalmente mediante EXEC CICS RETURN TRANSID). Si no hay ninguna transacción siguiente, este campo se deja en blanco.

Este campo es un campo de respuesta que se utiliza únicamente para transacciones 3270. La longitud de este campo viene dada por MQ_TRANSACTION_ID_LENGTH. El valor inicial de este campo es cuatro espacios en blanco.

Reserved2 (MQCHAR8)

Este campo es un campo reservado. El valor debe ser 8 espacios en blanco.

Reserved3 (MQCHAR8)

Este campo es un campo reservado. El valor debe ser 8 espacios en blanco.

CursorPosition (MQLONG)

El valor de este campo muestra la posición inicial del cursor cuando se inicia la transacción. Para las transacciones conversacionales, la posición del cursor se encuentra en el vector RECEIVE.

Este campo es un campo de solicitud que solo se utiliza para transacciones 3270. El valor inicial de este campo es 0. Este campo no está presente si *Version* es menor que MQCIH_VERSION_2

ErrorOffset (MQLONG)

El campo « ErrorOffset » muestra la posición de los datos no válidos detectados por la salida del puente. Este campo proporciona la compensación desde el inicio del mensaje hasta la ubicación de los datos no válidos.

ErrorOffset Es un campo de respuesta que se utiliza únicamente para transacciones 3270. El valor inicial de este campo es 0. Este campo no está presente si *Version* es menor que MQCIH_VERSION_2

InputItem (MQLONG)

Este campo es un campo reservado. El valor debe ser 0.

Este campo no está presente si *Version* es menor que MQCIH_VERSION_2

Reserved4 (MQLONG)

Este campo es un campo reservado. El valor debe ser 0.

Este campo no está presente si *Version* es menor que MQCIH_VERSION_2

MQCMHO: crear opciones de controlador de mensajes

La **MQCMHO** estructura permite a las aplicaciones especificar opciones que controlan cómo se crean los identificadores de mensajes. La estructura es un parámetro de entrada en la **MQCRTMH** llamada.

Disponibilidad

La **MQCMHO** estructura está disponible en las siguientes plataformas:

-  AIX
-  IBM i
-  Linux
-  Windows
-  z/OS

y para IBM MQ MQI clients conectados a estos sistemas.

Conjunto de caracteres y codificación

Los datos en **MQCMHO** deben estar en el juego de caracteres de la aplicación y la codificación de la aplicación (**MQENC_NATIVE**).

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 483. Campos en MQCMHO		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>StrucId</u> (identificador de estructura)	MQCMHO_STRUC_ID	' CMHO '
<u>Versión</u> (número de versión de la estructura)	MQCMHO_VERSION_1	1
<u>Opciones</u> (opciones)	MQCMHO_VALIDACIÓN_POR_DEFAULT	0
Notas: 1. En el lenguaje de programación C, la variable macroMQCMHO_DEFAULT contiene los valores que se enumeran en la tabla. Se puede utilizar de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQCMHO MyCMHO = {MQCMHO_DEFAULT};</pre>		

Declaraciones lingüísticas

Declaración C para MQCMHO

```
struct tagMQCMHO {  
    MQCHAR4    StrucId;           /* Structure identifier */  
    MQLONG     Version;          /* Structure version number */  
};
```

```

    MQLONG    Options;          /* Options that control the action of MQCRTMH */
};

```

COBOL declaración para MQCMHO

```

**  MQCMHO structure
10 MQCMHO.
**  Structure identifier
15 MQCMHO-STRUCID      PIC X(4).
**  Structure version number
15 MQCMHO-VERSION      PIC S9(9) BINARY.
**  Options that control the action of MQCRTMH
15 MQCMHO-OPTIONS      PIC S9(9) BINARY.

```

PL/I declaración para MQCMHO

```

dcl
  1 MQCMHO based,
    3 StrucId      char(4),          /* Structure identifier */
    3 Version      fixed bin(31), /* Structure version number */
    3 Options      fixed bin(31), /* Options that control the action of MQCRTMH */

```

High Level Assembler declaración para MQCMHO

```

MQCMHO          DSECT
MQCMHO_STRUCID  DS   CL4   Structure identifier
MQCMHO_VERSION  DS   F     Structure version number
MQCMHO_OPTIONS  DS   F     Options that control the action of
*                MQCRTMH
MQCMHO_LENGTH   EQU   *-MQCMHO
MQCMHO_AREA     DS   CL(MQCMHO_LENGTH)

```

Detalles de campo para MQCMHO

La estructura **MQCMHO** Create message handle options permite a las aplicaciones especificar opciones que controlan cómo se crean los identificadores de mensajes. La estructura es un parámetro de entrada en la **MQCRTMH** llamada.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 484. Campos en MQCMHO		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>StrucId</u> (identificador de estructura)	MQCMHO_STRUC_ID	'CMHO'
<u>Versión</u> (número de versión de la estructura)	MQCMHO_VERSION_1	1
<u>Opciones</u> (opciones)	MQCMHO_VALIDACIÓN _POR_DEFAULT	0
Notas: 1. En el lenguaje de programación C, la variable macro MQCMHO_DEFAULT contiene los valores que se enumeran en la tabla. Se puede utilizar de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQCMHO MyCMHO = {MQCMHO_DEFAULT};</pre>		

StrucId (MQCHAR4)

Este es el identificador de estructura de la estructura de opciones del identificador de mensajes creados. Siempre es un campo de entrada. Su valor es MQCMHO_STRUC_ID.

El valor debe ser:

MQCMHO_STRUC_ID

Identificador para crear la estructura de opciones del identificador de mensajes.

Para el lenguaje de programación C, también se **MQCMHO_STRUC_ID_ARRAY** define la constante. Esto tiene el mismo valor que **MQCMHO_STRUC_ID**, pero es una matriz de caracteres en lugar de una cadena.

Versión (MQLONG)

Este campo es siempre un campo de entrada. Su valor inicial es MQCMHO_VERSION_1.

Este es el número de versión de la estructura; el valor debe ser:

MQCMHO_VERSION_1

Version-1 Crear estructura de opciones de control de mensajes.

La siguiente constante especifica el número de versión de la versión actual:

MQCMHO_VERSIÓN_ACTUAL

Versión actual de la estructura de opciones para crear un identificador de mensaje.

Opciones (MQLONG)

Este campo es siempre un campo de entrada. Su valor inicial es MQCMHO_DEFAULT_VALIDATION.

Se puede especificar una de las siguientes opciones:

MQCMHO_VALIDAR

Cuando **MQSETMP** se llama para establecer una propiedad en este identificador de mensaje, se valida el nombre de la propiedad para garantizar que:

- no contiene caracteres no válidos.
- no comienza JMS o usr.JMS excepto en los siguientes casos:
 - JMSCorrelationID
 - JMSReplyTo
 - JMSType
 - JMSXGroupID
 - JMSXGroupSeq

Estos nombres están reservados para JMS propiedades.

- no es una de las siguientes palabras clave, en cualquier combinación de mayúsculas o minúsculas:
 - Y
 - BETWEEN (entre)
 - ESCAPE
 - FALSE
 - ENTRADA
 - ES
 - LIKE
 - NOT
 - Nulo
 - O
 - TRUE
- no comienza Cuerpo. o Raíz. (excepto Root.MQMD.).

Si la propiedad está definida MQ mente (mq.*) y se reconoce el nombre, los campos descriptores de la propiedad se establecen en los valores correctos para la propiedad. Si no se reconoce la propiedad, el *Support* campo del descriptor de la propiedad se establece en **MQPD_OPTIONAL**.

MQCMHO_VALIDACIÓN_POR_DEFAULT

Este valor especifica que se produce el nivel predeterminado de validación de los nombres de propiedades.

El nivel predeterminado de validación es equivalente al nivel especificado por **MQCMHO_VALIDATE**.

Este es el valor predeterminado.

MQCMHO_SIN_VALIDACIÓN

No se realiza ninguna validación del nombre de la propiedad. Véase la descripción de **MQCMHO_VALIDATE**.

Opción predeterminada: Si no se requiere ninguna de las opciones descritas anteriormente, se puede utilizar la siguiente opción:

MQCMHO_NINGUNO

Todas las opciones adoptan sus valores predeterminados. Utilice este valor para indicar que no se han especificado otras opciones. **MQCMHO_NONE** documentación del programa de ayuda; no está previsto que esta opción se utilice con ninguna otra, pero como su valor es cero, dicho uso no puede detectarse.

MQCNO - Opciones de conexión

La estructura MQCNO permite a la aplicación especificar opciones relacionadas con la conexión al gestor de colas. La estructura es un parámetro de entrada/salida en la llamada MQCONN.

Para obtener más información sobre el uso de identificadores compartidos y la llamada MQCONN, consulte [Conexiones compartidas \(independientes del subproceso\) con MQCONN](#).

Disponibilidad

La estructura MQCNO está disponible en las siguientes plataformas:

-  AIX
-  IBM i
-  Linux
-  Windows

y para IBM MQ MQI clientes conectados a estos sistemas.

Versión

Los archivos de encabezado, COPY e INCLUDE proporcionados para los lenguajes de programación compatibles contienen la versión más reciente de MQCNO, pero con el valor inicial del *Version* campo establecido en MQCNO_VERSION_1. Para utilizar campos que no están presentes en la estructura version-1, la aplicación debe establecer el *Version* campo en el número de versión requerido.

Conjunto de caracteres y codificación

Los datos en MQCNO deben estar en el juego de caracteres proporcionado por el atributo del **CodedCharSetId** gestor de colas y la codificación del gestor de colas local proporcionada por MQENC_NATIVE. Sin embargo, si la aplicación se ejecuta como un IBM MQ MQI client, la estructura debe estar en el juego de caracteres y en la codificación del cliente.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 485. Campos en MQCNO para MQCNO		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>StrucId</u> (identificador de estructura)	MQCNO_STRUC_ID	'CNO~'
<u>Versión</u> (número de versión de la estructura)	MQCNO_VERSION_1	1
<u>Opciones</u> (opciones que controlan la acción de MQCONN)	MQCNO_NONE	0
Nota: Los campos restantes se ignoran si <i>Versión</i> es menor que MQCNO_VERSION_2.		
<u>ClientConnOffset</u> (desplazamiento de la estructura MQCD para la conexión del cliente)	Ninguna	0
<u>ClientConnPtr</u> (dirección de la estructura MQCD para la conexión del cliente)	Ninguna	Puntero nulo o bytes nulos
Nota: Los campos restantes se ignoran si <i>Versión</i> es menor que MQCNO_VERSION_3.		
<u>ConnTag</u> (etiqueta de conexión del gestor de colas)	MQCT_NINGUNO	Nulos
Nota: Los campos restantes se ignoran si <i>Versión</i> es menor que MQCNO_VERSION_4.		
<u>SSLConfigPtr</u> (dirección de la estructura MQSCO para la conexión del cliente)	Ninguna	Puntero nulo o bytes nulos
<u>SSLConfigOffset</u> (desplazamiento de la estructura MQSCO para la conexión del cliente)	Ninguna	0
Nota: Los campos restantes se ignoran si <i>Versión</i> es menor que MQCNO_VERSION_5.		
<u>ConnectionId</u> (ID de conexión único)	Ninguna	Puntero nulo o bytes nulos
<u>SecurityParmsOffset</u> (desplazamiento de la estructura MQCSP para los parámetros de seguridad)	Ninguna	Puntero nulo o bytes nulos
<u>SecurityParmsPtr</u> (dirección de la estructura MQCSP para los parámetros de seguridad)	Ninguna	Puntero nulo o bytes nulos
Nota: Los campos restantes se ignoran si <i>Versión</i> es menor que MQCNO_VERSION_6.		
<u>Reservado</u> (campo reservado)	Ninguna	Campo reservado para rellenar la estructura hasta un límite de 64 bits.
<u>CCDTUrlLength</u> (Longitud del CCDT URL)	Ninguna	Longitud de la cadena identificada por <i>CCDTUrlPtr</i> o <i>CCDTUrlOffset</i>

Tabla 485. Campos en MQCNO para MQCNO (continuación)

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>CCDTUrlPtr</u> (Puntero CCDT URL)	Ninguna	Puntero a una cadena que contiene un identificador de canal de conexión (URL) para identificar la ubicación de la tabla de canales de conexión del cliente que se utilizará para la conexión.
<u>CCDTUrlOffset</u> (Desviación del CCDT (URL))	Ninguna	Desplazamiento en bytes desde una cadena que contiene un identificador de canal (URL) que identifica la ubicación de la tabla de canales de conexión del cliente que se utilizará para la conexión.
Nota: Los campos restantes se ignoran si <i>Version</i> es menor que MQCNO_VERSION_7.		
<u>ApplName</u> (nombre establecido por la aplicación)	Ninguna	Nombre establecido por la aplicación para identificar la conexión con el gestor de colas
<u>Reserved2</u> (campo reservado)	Ninguna	Campo reservado para rellenar la estructura hasta un límite de 64 bits.
Nota: Los campos restantes se ignoran si <i>Version</i> es menor que MQCNO_VERSION_8.		
<u>BalanceParms Offset</u>	Ninguna	Desplazamiento en bytes a la estructura MQBNO
<u>BalanceParmsPtr</u>	Ninguna	Puntero a la ubicación de la estructura MQBNO
Notas: 1. El símbolo – representa un único carácter en blanco. 2. En el lenguaje de programación C, la variable macroMQCNO_DEFAULT contiene los valores que se enumeran en la tabla. Úselo de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQCNO MycNO = {MQCNO_DEFAULT};</pre>		

Declaraciones lingüísticas

Declaración C para MQCNO

```
typedef struct tagMQCNO MQCNO;
struct tagMQCNO {
    MQCHAR4    StructId;          /* Structure identifier */
    MQLONG     Version;           /* Structure version number */
    MQLONG     Options;           /* Options that control the action of
    MQCONNX */
    MQLONG     ClientConnOffset; /* Offset of MQCD structure for client
    connection */
    MQPTR      ClientConnPtr;     /* Address of MQCD structure for client
    connection */
    MQBYTE128  ConnTag;           /* Queue manager connection tag */
    PMQSCO     SSLConfigPtr;      /* Address of MQSCO structure for client
    connection */
    MQLONG     SSLConfigOffset;   /* Offset of MQSCO structure for client
    connection */
    MQBYTE24   ConnectionId;      /* Unique connection identifier */
    MQLONG     SecurityParmsOffset /* Security fields */
    PMQCSP     SecurityParmsPtr  /* Security parameters */
    MQLONG     CCDTUrlLength      /* Length of string identified by Ptr or offset */
    MQLONG     CCDTUrlOffset      /* Offset in bytes to URL of client connection channel */
    PMQURL     CCDTUrlPtr        /* Address of string containing URL */
    MQBYTE4    Reserved          /* Reserved field to pad out to 64 bit boundary */
    MQCHAR28   ApplName          /* Name set by the application to identify the connection to
    the queue manager */
    MQBYTE4    Reserved2         /* Reserved field to pad out to 64 bit boundary */
    MQLONG     BalanceParmsOffset /* Offset of the MQBMO structure */
    PMQBMO     BalanceParmsPtr   /* Address of the location of the MQBMO structure */
};
```

COBOL declaración para MQCNO

```
** MQCNO structure
10 MQCNO.
** Structure identifier
15 MQCNO-STRUCID PIC X(4).
** Structure version number
15 MQCNO-VERSION PIC S9(9) BINARY.
** Options that control the action of MQCONNX
15 MQCNO-OPTIONS PIC S9(9) BINARY.
** Offset of MQCD structure for client connection
15 MQCNO-CLIENTCONNOFFSET PIC S9(9) BINARY.
** Address of MQCD structure for client connection
15 MQCNO-CLIENTCONNPTR POINTER.
** Queue manager connection tag
15 MQCNO-CONNTAG PIC X(128).
** Address of MQSCO structure for client connection
15 MQCNO-SSLCONFIGPTR POINTER.
** Offset of MQSCO structure for client connection
15 MQCNO-SSLCONFIGOFFSET PIC S9(9) BINARY.
** Unique connection identifier
15 MQCNO-CONNECTIONID PIC X(24).
** Offset of MQCSP structure for security parameters
15 MQCNO-SECURITYPARMSOFFSET PIC S9(9) BINARY.
** Address of MQCSP structure for security parameters
15 MQCNO-SECURITYPARMSPTR POINTER.
** Length of string identified by CCDTURLOFFSET or CCDTURLPTR
15 MQCNO-CCDTURLLENGTH
** Pointer to a string which contains a URL, to identify the location of the client
connection channel
15 MQCNO-CCDTURLPTR
** Address of string which contains a URL that identifies the location of the client
connection channel table
15 MQCNO-CCDTURLOFFSET
** Reserved field to pad to 64 bit boundary
15 MQCNO-RESERVED
** Name set by the application to identify the connection to the queue manager
15 MQCNO-APPLNAME
** Reserved field to pad to 64 bit boundary
15 MQCNO-RESERVED2
** Address of the MQBMO structure
15 MQCNO-BALANCEPARMSOFFSET
** Pointer to the MQBMO structure
15 MQCNO-BALANCEPARMSPTR
```

PL/I declaración para MQCNO

```

dcl
  1 MQCNO based,
    3 StrucId          char(4),          /* Structure identifier */
    3 Version          fixed bin(31),    /* Structure version number */
    3 Options          fixed bin(31),    /* Options that control the action
                                     of MQCONN */
    3 ClientConnOffset fixed bin(31),    /* Offset of MQCD structure for
                                     client connection */
    3 ClientConnPtr    pointer,          /* Address of MQCD structure for
                                     client connection */
    3 ConnTag          char(128),        /* Queue manager connection tag */
    3 SSLConfigPtr     pointer,          /* Address of MQSCO structure for
                                     client connection */
    3 SSLConfigOffset  fixed bin(31),    /* Offset of MQSCO structure for
                                     client connection */
    3 ConnectionId     char(24),        /* Unique connection identifier */
    3 SecurityParmsOffset fixed bin(31) /* Offset of MQCSP structure for
                                     security parameters */
    3 SecurityParmsPtr pointer,          /* Address of MQCSP structure for
                                     security parameters */
    3 CCDURLLength     fixed bin(31)     /* Length of string identified by CCDURLPtr
                                     or CCDURLOffset */
    3 CCDURLOffset     fixed bin(31)     /* Offset in bytes to URL of client connection channel */
    3 CCDURLPtr        pointer           /* Pointer to string containing URL */
    3 Reserved         char(4)          /* Reserved field to pad out to 64 bit boundary */
    3 ApplName         char(28)         /* Name set by the application to identify the
                                     connection to
                                     the queue manager */
    3 Reserved2        char(4)          /* Reserved field to pad out to 64 bit boundary */
    3 BalanceParmsOffset fixed bin(31) /* Offset of the MQBMO structure */
    3 BalanceParmsPtr  pointer          /* Address of the MQBMO structure */

```

High Level Assembler declaración para MQCNO

MQCNO	DSECT	
MQCNO_STRUCID	DS CL4	Structure identifier
MQCNO_VERSION	DS F	Structure version number
MQCNO_OPTIONS	DS F	Options that control the action of MQCONN
MQCNO_CLIENTCONNOFFSET	DS F	Offset of MQCD structure for client connection
MQCNO_CLIENTCONNPTR	DS F	Address of MQCD structure for client connection
MQCNO_CONNTAG	DS XL128	Queue manager connection tag
MQCNO_CONNECTIONID	DS XL24	Unique connection identifier
MQCNO_SSLCONFIGOFFSET	DS F	Offset of MQCSP structure for security parameters
MQCNO_SSLCONFIGPTR	DS F	Address of MQCSP structure for security parameters
MQCNO_LENGTH	EQU *-MQCNO	
MQCNO_AREA	ORG MQCNO	
MQCNO_CCDURLLENGTH	DS CL(MQCNO_LENGTH)	
MQCNO_CCDTURLOFFSET	DS F	Length of string identified by CCDURLPTR or CCDTURLOFFSET
MQCNO_CCDTURLPTR	DS F	Offset in bytes to URL of client connection channel
RESERVED	DS XL4	Pointer to string containing URL
APPLNAME	DS CL28	Reserved field to pad out to 64 bit boundary
RESERVED2	DS XL4	Name set by the application to identify the connection to the queue manager
MQCNO_BALANCEPARMSOFFSET	DS F	Reserved field to pad out to 64 bit boundary
MQCNO_BALANCEPARMSPTR	DS F	Offset of the MQBMO structure
		Address of the MQBMO structure

Declaración de Visual Basic para MQCNO

Type MQCNO		
StrucId	As String*4	'Structure identifier'
Version	As Long	'Structure version number'
Options	As Long	'Options that control the action of 'MQCONN'
ClientConnOffset	As Long	'Offset of MQCD structure for client 'connection'
ClientConnPtr	As MQPTR	'Address of MQCD structure for client 'connection'
ConnTag	As MQBYTE128	'Queue manager connection tag'

SSLConfigPtr	As MQPTR	'Address of MQSCO structure for client' 'connection'
SSLConfigOffset	As Long	'Offset of MQSCO structure for client' 'connection'
ConnectionId	As MQBYTE24	'Unique connection identifier'
SecurityParmsOffset	As Long	'Offset of MQCSP structure for security' 'parameters'
SecurityParmsPtr	As MQPTR	'Address of MQCSP structure for security' 'parameters'
CCDTUrlLength	As Long	'Length of string identified by <i>CCDTUrlPtr</i> ' 'or <i>CCDTUrlOffset</i> '
CCDTUrlOffset	As Long	'Offset in bytes to URL of client connection channel'
CCDTUrlPtr	As MQPTR	'Pointer to string containing URL'
Reserved	As MQBYTE4	'Reserved field to pad out to 64 bit boundary'
ApplName	As String*28	'Name set by the application to identify the connection to' 'the queue manager'
Reserved2	As MQBYTE4	'Reserved field to pad out to 64 bit boundary'
BalanceParmsOffset	As Long	'Offset in bytes to MQBNO structure'
BalanceParmsPtr	As MQPTR	'Address of MQBNO structure'
End Type		

Tareas relacionadas

Utilización de MQCONN

Detalles de campo para MQCNO

La estructura de opciones MQCNO Connect permite a la aplicación especificar opciones relacionadas con la conexión al gestor de colas. La estructura es un parámetro de entrada/salida en la llamada MQCONN.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 486. Campos en MQCNO		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>StrucId</u> (identificador de estructura)	MQCNO_STRUC_ID	'CNO-'
<u>Versión</u> (número de versión de la estructura)	MQCNO_VERSION_1	1
<u>Opciones</u> (opciones que controlan la acción de MQCONN)	MQCNO_NONE	0
Nota: Los campos restantes se ignoran si <i>Versión</i> es menor que MQCNO_VERSION_2.		
<u>ClientConnOffset</u> (desplazamiento de la estructura MQCD para la conexión del cliente)	Ninguna	0
<u>ClientConnPtr</u> (dirección de la estructura MQCD para la conexión del cliente)	Ninguna	Puntero nulo o bytes nulos
Nota: Los campos restantes se ignoran si <i>Versión</i> es menor que MQCNO_VERSION_3.		
<u>ConnTag</u> (etiqueta de conexión del gestor de colas)	MQCT_NINGUNO	Nulos
Nota: Los campos restantes se ignoran si <i>Versión</i> es menor que MQCNO_VERSION_4.		
<u>SSLConfigPtr</u> (dirección de la estructura MQSCO para la conexión del cliente)	Ninguna	Puntero nulo o bytes nulos
<u>SSLConfigOffset</u> (desplazamiento de la estructura MQSCO para la conexión del cliente)	Ninguna	0
Nota: Los campos restantes se ignoran si <i>Versión</i> es menor que MQCNO_VERSION_5.		

Tabla 486. Campos en MQCNO (continuación)		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>ConnectionId</u> (ID de conexión único)	Ninguna	Puntero nulo o bytes nulos
<u>SecurityParmsOffset</u> (desplazamiento de la estructura MQSCO para los parámetros de seguridad)	Ninguna	Puntero nulo o bytes nulos
<u>SecurityParmsPtr</u> (dirección de la estructura MQSCO para los parámetros de seguridad)	Ninguna	Puntero nulo o bytes nulos
Nota: Los campos restantes se ignoran si <i>Version</i> es menor que MQCNO_VERSION_6.		
<u>Reservado</u> (campo reservado)	Ninguna	Campo reservado para rellenar la estructura hasta un límite de 64 bits.
<u>CCDTUrlLength</u> (Longitud del CCDT URL)	Ninguna	Longitud de la cadena identificada por <i>CCDTUrlPtr</i> o <i>CCDTUrlOffset</i>
<u>CCDTUrlPtr</u> (Puntero CCDT URL)	Ninguna	Puntero a una cadena que contiene un identificador de canal (URL) que identifica la ubicación de la tabla de canales de conexión del cliente que se utilizará para la conexión.
<u>CCDTUrlOffset</u> (Desviación del CCDT (URL))	Ninguna	Desplazamiento en bytes desde una cadena que contiene un identificador de canal de conexión (URL) que identifica la ubicación de la tabla de canales de conexión del cliente que se utilizará para la conexión.
Nota: Los campos restantes se ignoran si <i>Version</i> es menor que MQCNO_VERSION_7.		
<u>ApplName</u> (nombre establecido por la aplicación)	Ninguna	Nombre establecido por la aplicación para identificar la conexión con el gestor de colas
<u>Reserved2</u> (campo reservado)	Ninguna	Campo reservado para rellenar la estructura hasta un límite de 64 bits.
Nota: Los campos restantes se ignoran si <i>Version</i> es menor que MQCNO_VERSION_8.		

Tabla 486. Campos en MQCNO (continuación)

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>BalanceParms Offset</u>	Ninguna	Desplazamiento en bytes a la estructura MQBNO
<u>BalanceParmsPtr</u>	Ninguna	Puntero a la ubicación de la estructura MQBNO
Notas: 1. El símbolo ↵ representa un único carácter en blanco. 2. En el lenguaje de programación C, la variable macroMQCNO_DEFAULT contiene los valores que se enumeran en la tabla. Úselo de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQCNO MycNO = {MQCNO_DEFAULT};</pre>		

StrucId (MQCHAR4)

Este es el identificador de estructura de la estructura de opciones de conexión. Siempre es un campo de entrada. Su valor es MQCNO_STRUC_ID.

El valor debe ser:

MQCNO_STRUC_ID

Identificador para la estructura de opciones de conexión.

Para el lenguaje de programación C, también se define la constante MQCNO_STRUC_ID_ARRAY. Esta constante tiene el mismo valor que MQCNO_STRUC_ID, pero es una matriz de caracteres en lugar de una cadena.

Versión (MQLONG)

La versión es siempre un campo de entrada. Su valor inicial es MQCNO_VERSION_1.

El valor debe ser uno de los siguientes:

MQCNO_VERSION_1

Version-1 Estructura de opciones de conexión.

MQCNO_VERSION_2

Version-2 Estructura de opciones de conexión.

MQCNO_VERSION_3

Version-3 Estructura de opciones de conexión.

MQCNO_VERSION_4

Version-4 Estructura de opciones de conexión.

MQCNO_VERSION_5

Version-5 Estructura de opciones de conexión.

MQCNO_VERSION_6

Version-6 Estructura de opciones de conexión.

MQCNO_VERSION_7

Version-7 Estructura de opciones de conexión.

MQCNO_VERSION_8

Version-8 Estructura de opciones de conexión.

Los campos que solo existen en las versiones más recientes de la estructura se identifican como tales en las descripciones de los campos. La siguiente constante especifica el número de versión de la versión actual:

MQCNO_VERSIÓN_ACTUAL

Versión actual de la estructura connect-options.

Opciones (MQLONG)

Opciones que controlan la acción de MQCONN.

Opciones contables

Las siguientes opciones controlan el tipo de contabilidad si el atributo del **AccountingConnOverride** gestor de colas está establecido en MQMON_ENABLED:

MQCNO_ACCOUNTING_MQI_ENABLED

Cuando la recopilación de datos de supervisión está desactivada en la definición del gestor de colas al establecer el **MQIAccounting** atributo en MQMON_OFF, al activar este indicador se habilita la recopilación de datos de contabilidad de MQI.

MQCNO_ACCOUNTING_MQI_DISABLED

Cuando se desactiva la recopilación de datos de supervisión en la definición del gestor de colas estableciendo el **MQIAccounting** atributo en MQMON_OFF, al activar este indicador se detiene la recopilación de datos de contabilidad de MQI.

MQCNO_ACCOUNTING_Q_ENABLED

Cuando la recopilación de datos de contabilidad de colas está desactivada en la definición del gestor de colas estableciendo el **MQIAccounting** atributo en MQMON_OFF, al establecer este indicador se habilita la recopilación de datos de contabilidad para aquellas colas que especifican un gestor de colas en el *MQIAccounting* campo de su definición de cola.

MQCNO_ACCOUNTING_Q_DISABLED

Cuando la recopilación de datos de contabilidad de colas está desactivada en la definición del gestor de colas estableciendo el **MQIAccounting** atributo en MQMON_OFF, al establecer este indicador se desactiva la recopilación de datos de contabilidad para aquellas colas que especifican un gestor de colas en el *MQIAccounting* campo de su definición de cola.

Si no se define ninguna de estas opciones, la contabilidad de la conexión se realiza según lo definido en los atributos del gestor de colas.

Opciones de encuadración

Las siguientes opciones controlan el tipo de IBM MQ enlace que se va a utilizar. Especifique solo una de estas opciones:

MQCNO_STANDARD_BINDING

La aplicación y el agente gestor de colas local (el componente que gestiona las operaciones de cola) se ejecutan en unidades de ejecución separadas (normalmente, en procesos separados). Esta disposición mantiene la integridad del gestor de colas; es decir, protege al gestor de colas de programas erróneos.

Si el gestor de colas admite varios tipos de enlace y se establece MQCNO_STANDARD_BINDING, el gestor de colas utiliza el **DefaultBindType** atributo en la *Connection* estrofa del *qm.ini* archivo para seleccionar el tipo de enlace real. Si esta estrofa no está definida, o el valor no se puede utilizar o no es adecuado para la aplicación, el gestor de colas selecciona un tipo de enlace adecuado. El gestor de colas establece el tipo de enlace real utilizado en las opciones de conexión.

Utilice MQCNO_STANDARD_BINDING en situaciones en las que la aplicación podría no haber sido probada completamente, o podría ser poco fiable o poco segura. MQCNO_STANDARD_BINDING es el valor predeterminado.

Esta opción es compatible con todos los entornos.

Si se va a enlazar a la biblioteca mqm, primero se intentará establecer una conexión de servidor estándar mediante el tipo de enlace predeterminado. Si no se ha podido cargar la biblioteca de servidor subyacente, en su lugar se intenta una conexión de cliente.

- Para cambiar el comportamiento de MQCONN (o MQCONNX si se especifica MQCNO_STANDARD_BINDING), establezca la variable de entorno MQ_CONNECT_TYPE en una de las siguientes opciones. Tenga en cuenta que hay una excepción a esto: si se especifica MQCNO_FASTPATH_BINDING con MQ_CONNECT_TYPE establecido en LOCAL o STANDARD, el administrador puede degradar las conexiones fastpath sin realizar ningún cambio relacionado en la aplicación.

Tabla 487. Valores para MQ_CONNECT_TYPE que cambian el comportamiento de MQCONN o MQCONNX	
Valor	Significado
CLIENTE	Sólo se intenta una conexión de cliente.
FASTPATH	Este valor estaba soportado en los releases anteriores, pero ahora se pasa por alto si se ha especificado.
LOCAL	Sólo se intenta una conexión con el servidor. Las conexiones fastpath se reducen a una conexión con el servidor estándar.
ESTÁNDAR	Soportado por motivos de compatibilidad con releases anteriores. Este valor se trata ahora como LOCAL.

- Si la variable de entorno MQ_CONNECT_TYPE no está definida cuando se llama a MQCONNX, se intenta establecer una conexión de servidor estándar utilizando el tipo de enlace predeterminado. Si no se puede cargar la biblioteca del servidor, se intenta una conexión de cliente.

MQCNO_FASTPATH_BINDING

La aplicación y el agente gestor de colas local forman parte de la misma unidad de ejecución. Esto contrasta con el método típico de enlace, en el que la aplicación y el agente gestor de colas local se ejecutan en unidades de ejecución separadas.

MQCNO_FASTPATH_BINDING se ignora si el gestor de colas no admite este tipo de enlace; el procesamiento continúa como si la opción no se hubiera especificado.

MQCNO_FASTPATH_BINDING puede ser ventajoso en situaciones en las que varios procesos consumen más recursos que el total utilizado por la aplicación. Una aplicación que utiliza el enlace fastpath se conoce como *aplicación de confianza*.

Tenga en cuenta los siguientes puntos importantes a la hora de decidir si utilizar el enlace fastpath:

- El uso de la opción MQCNO_FASTPATH_BINDING no impide que una aplicación altere o dañe los mensajes y otras áreas de datos que pertenecen al gestor de colas. Utilice esta opción solo en situaciones en las que haya evaluado completamente estas cuestiones.
- La aplicación no debe utilizar señales asíncronas ni interrupciones de temporizador (como sigkill) con MQCNO_FASTPATH_BINDING. También existen restricciones en el uso de segmentos de memoria compartida.
- La aplicación debe utilizar la llamada MQDISC para desconectarse del gestor de colas.
- La aplicación debe finalizar antes de cerrar el gestor de colas con el endmqm comando.
-  En IBM i, el trabajo debe ejecutarse con un perfil de usuario que pertenezca al QMQMADM grupo. Además, el programa no debe detenerse de forma anómala, ya que de lo contrario podrían producirse resultados impredecibles.
-   En AIX and Linux, el identificador mqm de usuario debe ser el identificador de usuario efectivo, y el identificador mqm de grupo debe ser el identificador de grupo efectivo. Para que la aplicación se ejecute de esta manera, configure el programa para que sea

propiedad del identificador mqm de usuario y del mqm identificador de grupo, y luego establezca los bits de setgid permiso setuid y en el programa.

El Administrador IBM MQ de Autoridad de Objetos (OAM) sigue utilizando el ID de usuario real para la comprobación de autoridad.

-  En Windows, el usuario que ejecute el programa debe ser miembro del mqm grupo. La vinculación Fastpath no es compatible con aplicaciones de 32 bits.

La opción MQCNO_FASTPATH_BINDING es compatible con los siguientes entornos:

-  AIX
-  IBM i
-  Linux
-  Windows

-  En z/OS, la opción se acepta pero se ignora.

Para obtener más información sobre las implicaciones del uso de aplicaciones de confianza, consulte [Restricciones para aplicaciones de confianza](#).

MQCNO_SHARED_BINDING

Con MQCNO_SHARED_BINDING, la aplicación y el agente gestor de colas local comparten algunos recursos. MQCNO_SHARED_BINDING se omite si el gestor de colas no soporta este tipo de enlace. El proceso continuará como si no se hubiese especificado la opción.

MQCNO_ISOLATED_BINDING

En este caso, el proceso de solicitud y el agente gestor de colas local están aislados entre sí, ya que no comparten recursos. Si el gestor de colas no da soporte a este tipo de enlace, se ignora MQCNO_ISOLATED_BINDING. El proceso continuará como si no se hubiese especificado la opción.

MQCNO_CLIENT_BINDING

Especifique esta opción para que la aplicación intente únicamente una conexión con el cliente. Esta opción tiene las siguientes limitaciones:

-  En z/OS se omite MQCNO_CLIENT_BINDING.
- MQCNO_CLIENT_BINDING se rechaza con MQRC_OPTIONS_ERROR, si se ha especificado con cualquier opción de enlace MQCNO distinta de MQCNO_STANDARD_BINDING.
- MQCNO_CLIENT_BINDING no está disponible para Java o .NET ya que tienen sus propios mecanismos para elegir el tipo de enlace.

MQCNO_LOCAL_BINDING

Especifique esta opción para hacer que la aplicación intente una conexión con el servidor. Si también se ha especificado MQCNO_FASTPATH_BINDING, MQCNO_ISOLATED_BINDING o MQCNO_SHARED_BINDING, la conexión será de dicho tipo, y se ha documentado en esta sección. De lo contrario, se intenta efectuar una conexión de servidor estándar utilizando el tipo de enlace predeterminado. MQCNO_LOCAL_BINDING tiene las limitaciones siguientes:

-  Se omite MQCNO_LOCAL_BINDING en z/OS.
- MQCNO_LOCAL_BINDING se rechaza con MQRC_OPTIONS_ERROR si se ha especificado con cualquier opción de reconexión MQCNO que no sea MQCNO_RECONNECT_AS_DEF.
- MQCNO_LOCAL_BINDING no está disponible para Java o .NET ya que tienen sus propios mecanismos para elegir el tipo de enlace.

En las siguientes plataformas, puede utilizar la variable de entorno MQ_CONNECT_TYPE con el tipo de enlace especificado por el Options campo para controlar el tipo de enlace utilizado.

-  AIX
-  Linux
-  Windows

Si se especifica esta variable de entorno, debe tener el valor FASTPATH o STANDARD ; si tiene un valor diferente, se ignora. El valor de la variable de entorno distingue entre mayúsculas y minúsculas; consulte la variable de entorno MQCONNx para obtener más información.

La variable de entorno y *Options* el campo interactúan de la siguiente manera:

- Si omite la variable de entorno o le asigna un valor no compatible, el uso del enlace fastpath vendrá determinado únicamente por el Options campo.
- Si le asignas a la variable de entorno un valor compatible, el enlace fastpath solo se utilizará si tanto la variable de entorno como Options el campo especifican el enlace fastpath.

Opciones de etiquetas de conexión

Las siguientes opciones controlan el uso de la etiqueta ConnTag de conexión. Solo se admiten cuando se conectan a un IBM MQ for z/OS gestor de colas. No son válidos si Version es menor que MQCNO_VERSION_3. Especifique solo una de estas opciones. Si se conecta desde un gestor de colas multiplataforma, especifique MQCNO_GENERATE_CONN_TAG.

MQCNO_GENERAR_ETIQUETA_CONEXIÓN

Devuelve la etiqueta de conexión que el gestor de colas ha asociado a esta conexión, en la estructura de salida MQCNO.

La etiqueta de conexión devuelta es idéntica para todas las conexiones que el gestor de colas considera como una única instancia de aplicación.

MQCNO_SERIALIZE_CONN_TAG_Q_MGR

Esta opción solicita el uso exclusivo de la etiqueta de conexión dentro del gestor de colas local. Si la etiqueta de conexión ya está en uso en el gestor de colas local, la llamada MQCONNx falla con el código de motivo MQRC_CONN_TAG_IN_USE. El resultado de la llamada no se ve afectado por el uso de la etiqueta de conexión en otra parte del grupo de uso compartido de colas al que pertenece el gestor de colas local.

MQCNO_SERIALIZE_CONN_TAG_QSG

Esta opción solicita el uso exclusivo de la etiqueta de conexión dentro del grupo de uso compartido de colas al que pertenece el gestor de colas local. Si la etiqueta de conexión ya está en uso en el grupo de uso compartido de colas, la llamada MQCONNx falla con el código de motivo MQRC_CONN_TAG_IN_USE.

MQCNO_RESTRICT_CONN_TAG_Q_MGR

Esta opción solicita el uso compartido de la etiqueta de conexión dentro del gestor de colas local. Si la etiqueta de conexión ya está en uso en el gestor de colas local, la llamada MQCONNx puede tener éxito si la aplicación solicitante se está ejecutando en el mismo ámbito de procesamiento que el usuario existente de la etiqueta. Si no se cumple esta condición, la llamada MQCONNx falla con el código de motivo MQRC_CONN_TAG_IN_USE. El resultado de la llamada no se ve afectado por el uso de la etiqueta de conexión en otra parte del grupo de uso compartido de colas al que pertenece el gestor de colas local.

- Las aplicaciones deben ejecutarse dentro del mismo espacio de direcciones MVS para compartir la etiqueta de conexión. Si la aplicación que utiliza la etiqueta de conexión es una aplicación cliente, no se permite MQCNO_RESTRICT_CONN_TAG_Q_MGR.

► **z/OS** **MQCNO_RESTRICT_CONN_TAG_QSG**

Esta opción solicita el uso compartido de la etiqueta de conexión dentro del grupo de uso compartido de colas al que pertenece el gestor de colas local. Si la etiqueta de conexión ya se está utilizando en el grupo de uso compartido de colas, la llamada MQCONN puede realizarse correctamente siempre que la aplicación solicitante se esté ejecutando en el mismo ámbito de procesamiento y esté conectada al mismo gestor de colas que el usuario existente de la etiqueta.

Si no se cumplen estas condiciones, la llamada MQCONN falla con el código de motivo MQRC_CONN_TAG_IN_USE.

- Las aplicaciones deben ejecutarse dentro del mismo espacio de direcciones MVS para compartir la etiqueta de conexión. Si la aplicación que utiliza la etiqueta de conexión es una aplicación cliente, no se permite MQCNO_RESTRICT_CONN_TAG_QSG.

Si no se especifica ninguna opción, no ConnTag se utiliza.

► **Multi** **Opciones para compartir el control**

Estas opciones son compatibles con los siguientes entornos:

- ► **AIX** AIX
- ► **IBM i** IBM i
- ► **Linux** Linux
- ► **Windows** Windows

Controlan el intercambio de identificadores entre diferentes subprocesos (unidades de procesamiento paralelo) dentro del mismo proceso. Solo puede especificar una de estas opciones:

MQCNO_HANDLE_SHARE_NONE

Esta opción indica que las conexiones y los identificadores de objetos solo pueden ser utilizados por el subproceso que provocó la asignación del identificador (es decir, el subproceso que emitió la llamada MQCONN, MQCONNX o MQOPEN). Los manejadores no pueden ser utilizados por otros subprocesos pertenecientes al mismo proceso.

MQCNO_MANEJAR_COMPARTIR_BLOQUE

Esta opción indica que las conexiones y los identificadores de objetos asignados por un subproceso de un proceso pueden ser utilizados por otros subprocesos que pertenezcan al mismo proceso. Sin embargo, solo un subproceso a la vez puede utilizar un identificador concreto; es decir, solo se permite el uso secuencial de un identificador. Si un subproceso intenta utilizar un identificador que ya está siendo utilizado por otro subproceso, la llamada se bloquea (espera) hasta que el identificador esté disponible.

MQCNO_HANDLE_SHARE_NO_BLOCK

Es igual que MQCNO_HANDLE_SHARE_BLOCK, salvo que si el identificador está siendo utilizado por otro subproceso, la llamada se completa inmediatamente con MQCC_FAILED y MQRC_CALL_IN_PROGRESS, en lugar de bloquearse hasta que el identificador esté disponible.

Un subproceso puede tener cero o un identificador no compartido:

- Cada llamada MQCONN o MQCONNX que especifica MQCNO_HANDLE_SHARE_NONE devuelve un nuevo identificador no compartido en la primera llamada y el mismo identificador no compartido en la segunda llamada y en las siguientes (suponiendo que no haya ninguna llamada MQDISC intermedia). El código de motivo es MQRC_ALREADY_CONNECTED para la segunda llamada y las siguientes.
- Cada llamada a MQCONNX que especifica MQCNO_HANDLE_SHARE_BLOCK o MQCNO_HANDLE_SHARE_NO_BLOCK devuelve un nuevo identificador compartido en cada llamada.

Los identificadores de objeto heredan las mismas propiedades de uso compartido que el identificador de conexión especificado en la llamada MQOPEN que creó el identificador de objeto. Además, las

unidades de trabajo heredan las mismas propiedades de uso compartido que el identificador de conexión utilizado para iniciar la unidad de trabajo; si la unidad de trabajo se inicia en un subproceso utilizando un identificador compartido, la unidad de trabajo se puede actualizar en otro subproceso utilizando el mismo identificador.

Si no se especifica una opción para compartir identificadores, el valor predeterminado viene determinado por el entorno:

-  En el entorno Microsoft del servidor de transacciones (MTS), el valor predeterminado es el mismo que MQCNO_HANDLE_SHARE_BLOCK.
- En otros entornos, el valor predeterminado es el mismo que MQCNO_HANDLE_SHARE_NONE.

Opciones de reconexión

Las opciones de reconexión determinan si una conexión se puede volver a conectar. Solo las conexiones de los clientes se pueden volver a conectar.

MQCNO_RECONECTAR_COMO_DEF

La opción de reconexión se establece en su valor predeterminado. Si no se establece ningún valor predeterminado, el valor de esta opción se resuelve como DISABLED. El valor de la opción se transmite al servidor y puede ser consultado por PCF y MQSC.

MQCNO_RECONECTAR

La aplicación se puede volver a conectar a cualquier gestor de colas que sea compatible con el valor del **QmgrName** parámetro de MQCONN. Utilice la opción MQCNO_RECONNECT solo si no existe afinidad entre la aplicación cliente y el gestor de colas con el que estableció inicialmente la conexión. El valor de la opción se transmite al servidor y puede ser consultado por PCF y MQSC.

MQCNO_RECONECTAR_DESACTIVADO

La aplicación no se puede volver a conectar. El valor de la opción no se transmite al servidor.

MQCNO_RECONECTAR_Q_MGR

La aplicación solo se puede volver a conectar al gestor de colas con el que se conectó originalmente. Utilice este valor si un cliente puede volver a conectarse, pero existe una afinidad entre la aplicación cliente y el gestor de colas con el que estableció originalmente una conexión. Elija este valor si desea que el cliente se reconecte automáticamente a la instancia en espera de un gestor de colas con elevada disponibilidad. El valor de la opción se transmite al servidor y puede ser consultado por PCF y MQSC.

Utilice las opciones MQCNO_RECONNECT, MQCNO_RECONNECT_DISABLED y MQCNO_RECONNECT_Q_MGR solo para conexiones de cliente. Si las opciones se utilizan para una conexión vinculante, MQCONN falla con el código de finalización MQCC_FAILED y el código MQRC_OPTIONS_ERROR de motivo. La reconexión automática del cliente no es compatible con IBM MQ classes for Java

Opciones para compartir conversaciones

Las siguientes opciones solo se aplican a las conexiones de clientes de « TCP/IP ». Para los canales SNA, SPX y NetBios, estos valores se ignoran y el canal funciona como en versiones anteriores del producto

MQCNO_NO_CONV_COMPARTIR

Esta opción no permite compartir conversaciones.

Puede utilizar MQCNO_NO_CONV_SHARING en situaciones en las que las conversaciones tienen una carga elevada y, por lo tanto, existe la posibilidad de que se produzcan conflictos en el extremo de la conexión del servidor de la instancia de canal en la que existen las conversaciones compartidas. MQCNO_NO_CONV_SHARING se comporta como SHARECNV(1) cuando se conecta a un canal que admite el uso compartido de conversaciones, y como SHARECNV(0) cuando se conecta a un canal que no admite el uso compartido de conversaciones.

MQCNO_TODAS_CONVERSACIONES_COMPARTIDAS

Esta opción permite compartir conversaciones; la aplicación no impone ningún límite al número de conexiones en la instancia del canal. Esta opción es el valor predeterminado.

Si la aplicación indica que la instancia del canal puede compartir, pero la definición *SharingConversations* (SHARECNV) en el extremo de conexión del servidor del canal está establecida en uno, no se produce ningún intercambio y no se envía ninguna advertencia a la aplicación.

Del mismo modo, si la aplicación indica que se permite compartir, pero la definición de *SharingConversations* conexión al servidor está establecida en cero, no se muestra ninguna advertencia y la aplicación muestra el mismo comportamiento que un cliente en versiones anteriores a IBM WebSphere MQ 7.0; se ignora la configuración de la aplicación relacionada con compartir conversaciones.

MQCNO_NO_CONV_SHARING y MQCNO_ALL_CONVS_SHARE son mutuamente excluyentes. Si se especifican ambas opciones en una conexión concreta, la conexión se rechaza con un código de motivo MQRC_OPTIONS_ERROR.

Opciones de definición de canal

Las siguientes opciones controlan el uso de la estructura de definición de canal pasada en MQCNO:

MQCNO_CD_SOLO_PARA_SALIDA

Esta opción permite que la estructura de definición de canales en MQCNO se utilice únicamente para devolver el nombre del canal utilizado en una llamada MQCONNX correcta.

Si no se proporciona una estructura de definición de canal válida, la llamada falla con el código de motivo MQRC_CD_ERROR.

Si la aplicación no se está ejecutando como cliente, se ignora la opción.

El nombre del canal devuelto se puede utilizar en una llamada MQCONNX posterior utilizando la opción MQCNO_USE_CD_SELECTION para volver a conectarse utilizando la misma definición de canal. Esto puede resultar útil cuando hay varias definiciones de canal aplicables en la tabla de canales del cliente.

MQCNO_USAR_SELECCIÓN_DE_CD

Esta opción permite que la llamada MQCONNX se conecte utilizando el nombre del canal contenido en la estructura de definición del canal pasada en MQCNO.

Si se establece la variable de entorno MQSERVER, se utiliza la definición de canal definida por ella. Si MQSERVER no está configurado, se utiliza la tabla de canales del cliente.

Si no se encuentra una definición de canal con el nombre de canal y el nombre del gestor de colas coincidentes, la llamada falla con el código de motivo MQRC_Q_MGR_NAME_ERROR.

Si no se proporciona una estructura de definición de canal válida, la llamada falla con el código de motivo MQRC_CD_ERROR.

Si la aplicación no se está ejecutando como cliente, se ignora la opción.

Opción predeterminada

Si no necesita ninguna de las opciones descritas anteriormente, puede utilizar la siguiente opción:

MQCNO_NONE

No se especifica ninguna opción.

Utilice MQCNO_NONE para facilitar la documentación del programa. No está previsto que esta opción se utilice con ninguna otra opción MQCNO_*, pero dado que su valor es cero, dicho uso no puede detectarse.

ClientConnOffset (MQLONG)

ClientConnOffset Es el desplazamiento en bytes de una estructura de definición de canal MQCD desde el inicio de la estructura MQCNO. El desplazamiento puede ser positivo o negativo. Este campo es un campo de entrada con un valor inicial de 0.

Úselo *ClientConnOffset* solo cuando la aplicación que emite la llamada MQCONNX se esté ejecutando como un IBM MQ MQI client. Para obtener información sobre cómo utilizar este campo, consulte la descripción del *ClientConnPtr* campo.

Este campo se ignora si *Version* es menor que MQCNO_VERSION_2.

ClientConnPtr (MQPTR)

ClientConnPtr es un campo de entrada. Su valor inicial es el puntero nulo en aquellos lenguajes de programación que admiten punteros, y una cadena de bytes totalmente nula en los demás casos.

Utilice *ClientConnOffset* y *ClientConnPtr* solo cuando la aplicación que emite la llamada MQCONNX se esté ejecutando como un IBM MQ MQI client. Al especificar uno u otro de estos campos, la aplicación puede controlar la definición del canal de conexión del cliente proporcionando una estructura de definición de canal MQCD que contenga los valores requeridos.

Si la aplicación se ejecuta como IBM MQ MQI client, pero no proporciona una estructura MQCD, se utiliza la variable MQSERVER de entorno para seleccionar la definición del canal. Si no MQSERVER está configurado, se utiliza la tabla de canales del cliente.

Si la aplicación no se está ejecutando como un IBM MQ MQI client, *ClientConnOffset* y *ClientConnPtr* se ignoran.

Si la aplicación proporciona una estructura MQCD, establezca los campos enumerados con los valores requeridos; los demás campos de MQCD se ignorarán. Puede rellenar cadenas de caracteres con espacios en blanco hasta alcanzar la longitud del campo, o terminarlas con un carácter nulo. Consulte [“Campos” en la página 1503](#) para obtener más información sobre los campos de la estructura MQCD.

Tabla 488. Campos en MQCD

Campo en MQCD	Valor
<i>ChannelName</i>	Nombre de canal.
<i>Version</i>	Número de versión de la estructura. No debe ser inferior a MQCD_VERSION_7.
<i>TransportType</i>	Cualquier tipo de transporte compatible.
<i>ModeName</i>	LU Nombre del modo « 6.2 ».
<i>TpName</i>	LU Nombre del programa de transacciones « 6.2 ».
<i>SecurityExit</i>	Nombre de la salida de seguridad del canal.
<i>SendExit</i>	Nombre del canal de salida de envío.
<i>ReceiveExit</i>	Nombre del canal de salida de recepción.
<i>MaxMsgLength</i>	Longitud máxima en bytes de los mensajes que se pueden enviar a través del canal de conexión del cliente.
<i>SecurityUserData</i>	Datos de usuario para salida de seguridad.
<i>SendUserData</i>	Datos de usuario para enviar salida.
<i>ReceiveUserData</i>	Datos de usuario para salida de recepción.
<i>UserIdentifier</i>	Identificador de usuario que se utilizará para establecer una sesión LU 6.2.
<i>Password</i>	Contraseña que se utilizará para establecer una sesión LU 6.2.

Tabla 488. Campos en MQCD (continuación)

Campo en MQCD	Valor
<i>ConnectionName</i>	nombre de conexión.
<i>HeartbeatInterval</i>	Tiempo en segundos entre latidos cardíacos.
<i>StructLength</i>	Longitud de la estructura MQCD.
<i>ExitNameLength</i>	Longitud de los nombres de salida abordados por <i>SendExitPtr</i> y <i>ReceiveExitPtr</i> . Debe ser mayor que cero si <i>SendExitPtr</i> o <i>ReceiveExitPtr</i> se establece en un valor que no sea el puntero nulo.
<i>ExitDataLength</i>	Longitud de los datos de salida tratados por <i>SendUserDataPtr</i> y <i>ReceiveUserDataPtr</i> . Debe ser mayor que cero si <i>SendUserDataPtr</i> o <i>ReceiveUserDataPtr</i> se establece en un valor que no sea el puntero nulo.
<i>SendExitsDefined</i>	Número de salidas de envío abordadas por <i>SendExitPtr</i> . Si es cero, <i>SendExit</i> y <i>SendUserData</i> proporcionan el nombre y los datos de salida. Si es mayor que cero, <i>SendExitPtr</i> y <i>SendUserDataPtr</i> proporcionan los nombres y datos de salida, y <i>SendExit</i> y <i>SendUserData</i> deben estar en blanco.
<i>ReceiveExitsDefined</i>	Número de salidas de recepción abordadas por <i>ReceiveExitPtr</i> . Si es cero, <i>ReceiveExit</i> y <i>ReceiveUserData</i> proporcionan el nombre y los datos de salida. Si es mayor que cero, <i>ReceiveExitPtr</i> y <i>ReceiveUserDataPtr</i> proporcionan los nombres y datos de salida, y <i>ReceiveExit</i> y <i>ReceiveUserData</i> deben estar en blanco.
<i>SendExitPtr</i>	Dirección del nombre de la primera salida de envío.
<i>SendUserDataPtr</i>	Dirección de los datos para la primera salida de envío.
<i>ReceiveExitPtr</i>	Dirección del nombre de la primera salida recibida.
<i>ReceiveUserDataPtr</i>	Dirección de los datos para la primera salida recibida.
<i>LongRemoteUserIdLength</i>	Longitud del identificador de usuario remoto largo.
<i>LongRemoteUserIdPtr</i>	Dirección del identificador de usuario remoto largo.
<i>RemoteSecurityId</i>	Identificador de seguridad remoto.
<i>SSLCipherSpec</i>	TLS CipherSpec.
<i>SSLPeerNamePtr</i>	Dirección de un nombre de par de TLS.
<i>SSLPeerNameLength</i>	Longitud del nombre de un par de TLS.
<i>KeepAliveInterval</i>	Valor pasado a la pila de comunicaciones para la sincronización de mantenimiento del canal
<i>LocalAddress</i>	La dirección de comunicaciones local, incluida la dirección IP del adaptador de red local que se va a utilizar y un rango de puertos que se van a utilizar para las conexiones salientes.

Proporcione la estructura de definición del canal de una de estas dos maneras:

- Al utilizar el campo de compensación *ClientConnOffset*

En este caso, la aplicación debe declarar una estructura compuesta que contenga un MQCNO seguido de la estructura de definición de canal MQCD, y establecer *ClientConnOffset* en la compensación de la estructura de definición de canal desde el inicio del MQCNO. Asegúrese de que este desplazamiento sea correcto. *ClientConnPtr* debe establecerse en el puntero nulo o en bytes nulos.

Utilícelo *ClientConnOffset* para lenguajes de programación que no admiten el tipo de datos puntero o que implementan el tipo de datos puntero de una forma que no es portable a diferentes entornos (por ejemplo, el lenguaje de programación COBOL).

Para el lenguaje de programación Visual Basic, una estructura compuesta llamada MQCNOCD se proporciona en el archivo de encabezado CMQXB.BAS; esta estructura contiene una estructura MQCNO seguida de una estructura MQCD. Inicialice MQCNOCD invocando la subrutina MQCNOCD_DEFAULTS. MQCNOCD se utiliza con el MQCONN. Cualquier variante de la llamada MQCONN; consulte la descripción de la llamada MQCONN para obtener más detalles.

- Al utilizar el campo puntero *ClientConnPtr*

En este caso, la aplicación puede declarar la estructura de definición del canal por separado de la estructura MQCNO y establecer *ClientConnPtr* en la dirección de la estructura de definición del canal. Establecer *ClientConnOffset* en cero.

Utilícelo *ClientConnPtr* para lenguajes de programación que admitan el tipo de datos puntero de forma que sea portátil a diferentes entornos (por ejemplo, el lenguaje de programación C).

En el lenguaje de programación C, puede utilizar la variable macro MQCD_CLIENT_CONN_DEFAULT para proporcionar valores iniciales para la estructura que son más adecuados para su uso en la llamada MQCONN que los valores iniciales proporcionados por MQCD_DEFAULT.

Independientemente de la técnica que elija, solo puede utilizar una de las opciones *ClientConnOffset* y *ClientConnPtr* ; la llamada fallará con el código de motivo MQRC_CLIENT_CONN_ERROR si ambas son distintas de cero.

Cuando finaliza la llamada MQCONN, no se vuelve a hacer referencia a la estructura MQCD.

Este campo se ignora si *Version* es menor que MQCNO_VERSION_2.

Nota: En plataformas donde el lenguaje de programación no admite el tipo de datos puntero, este campo se declara como una cadena de bytes de la longitud adecuada, con el valor inicial siendo la cadena de bytes totalmente nula.

ConnTag (MQBYTE128)



Una etiqueta de conexión es conceptualmente similar a un identificador de conexión, pero puede abarcar varias conexiones relacionadas, identificándolas como una única instancia de aplicación. En multiplataformas, la etiqueta de conexión es generada por el gestor de colas en el momento de la conexión.

Para obtener más información, consulte [identificador de conexión](#) e [instancia de aplicación](#).

Las etiquetas de conexión generadas son legibles semi para los humanos. Es decir, se pueden mostrar y filtrar en MQSC como si fueran cadenas en el juego de caracteres local. Las conexiones que se sabe que IBM MQ están relacionadas se asignan automáticamente la misma etiqueta de conexión. Esta tarea es especialmente importante para [el equilibrio de aplicaciones](#).

La etiqueta de conexión generada es visible de tres maneras:

- En la estructura MQCNO de salida en una llamada MQCONN, cuando se especifica [MQCNO_GENERATE_CONN_TAG](#).
- En la salida de [DISPLAY CONN](#) (o equivalentes programáticos).
- En la salida de [DISPLAY APSTATUS](#) (o equivalentes).

La etiqueta deja de ser válida cuando la aplicación finaliza o emite la llamada MQDISC.

ConnTag (MQBYTE128) en IBM MQ for z/OS



Una etiqueta de conexión es conceptualmente similar a un identificador de conexión, pero puede abarcar varias conexiones relacionadas, identificándolas como una única instancia de aplicación. En IBM MQ for z/OS, la etiqueta de conexión es un campo de entrada proporcionado por la aplicación y utilizado junto con las opciones MQCNO_*_CONN_TAG para serializar las conexiones desde esa instancia de la aplicación.

Cuando hay varias instancias de una aplicación que están destinadas a conectarse simultáneamente, cada una de ellas debe proporcionar un valor único para este campo. Consulte las descripciones de estas [opciones de etiquetas de conexión](#) para obtener más detalles.

Notas:

- En IBM MQ for z/OS, no hay forma de determinar administrativamente la etiqueta de conexión asociada a una aplicación en tiempo de ejecución.
- Los valores de etiqueta de conexión que comienzan con MQ en mayúsculas, minúsculas o mixtas, ya sea en ASCII o EBCDIC, están reservados para su uso por IBM productos. No utilice valores de etiqueta de conexión que comiencen con estas letras.

Utilice el siguiente valor especial si no necesita ninguna etiqueta:

MQCT_NINGUNO

El valor es un cero binario para la longitud del campo.

Para el lenguaje de programación C, también se define la constante MQCT_NONE_ARRAY; esta constante tiene el mismo valor que MQCT_NONE, pero es una matriz de caracteres en lugar de una cadena.

El campo « ConnTag » se utiliza al conectarse a un gestor z/OS de colas.

La longitud de este campo viene dada por MQ_CONN_TAG_LENGTH. Este campo se ignora si *Version* es menor que MQCNO_VERSION_3.

 Consulte “ConnTag (MQBYTE128)” en la página 286 para obtener información sobre cómo utilizar la etiqueta de conexión en IBM MQ for Multiplatforms.

SSLConfigPtr (PMQSCO)

SSLConfigPtr es un campo de entrada. Su valor inicial es el puntero nulo en aquellos lenguajes de programación que admiten punteros, y una cadena de bytes totalmente nula en los demás casos.

Utilice *SSLConfigPtr* y *SSLConfigOffset* solo cuando la aplicación que realiza la llamada MQCONN se ejecute como un IBM MQ MQI client y el protocolo del canal sea TCP/IP. Si la aplicación no se está ejecutando como cliente IBM MQ, o si el protocolo del canal no es « TCP/IP », *SSLConfigPtr* y *SSLConfigOffset* se ignoran.

Al especificar *SSLConfigPtr* o *SSLConfigOffset*, más o *ClientConnPtr* *ClientConnOffset*, la aplicación puede controlar el uso de TLS para la conexión del cliente. Cuando la información TLS se especifica de esta manera, se ignoran las variables de entorno MQSSLKEYR y MQSSLCRYP; también se ignora cualquier información relacionada con TLS en la tabla de definición de canales de cliente (CCDT).

La información « TLS » solo puede especificarse en:

- La primera llamada MQCONN del proceso cliente, o
- Una llamada MQCONN posterior cuando todas las conexiones TLS anteriores al gestor de colas se hayan concluido utilizando MQDISC.

Estos son los únicos estados en los que se puede inicializar el entorno de proceso completo « TLS ». Si se emite una llamada MQCONN especificando información TLS cuando ya existe el entorno TLS, se ignora la información TLS de la llamada y la conexión se establece utilizando el entorno TLS existente; en este caso, la llamada devuelve el código de finalización MQCC_WARNING y el código de motivo MQRC_SSL_ALREADY_INITIALIZED.

Puede proporcionar la estructura MQSCO de la misma manera que la estructura MQCD, ya sea especificando una dirección en *SSLConfigPtr*, o especificando un desplazamiento en

SSLConfigOffset ; consulte la descripción de *ClientConnPtr* para obtener más detalles sobre cómo hacerlo. Sin embargo, no se puede utilizar más de uno de *SSLConfigPtr* y *SSLConfigOffset* ; la llamada falla con el código de motivo MQRC_SSL_CONFIG_ERROR. si ambos son distintos de cero.

Una vez completada la llamada MQCONN, no se vuelve a hacer referencia a la estructura MQSCO.

Este campo se ignora si *Version* es menor que MQCNO_VERSION_4.

Nota: En plataformas donde el lenguaje de programación no admite el tipo de datos puntero, este campo se declara como una cadena de bytes de la longitud adecuada.

SSLConfigOffset (MQLONG)

SSLConfigOffset es la compensación en bytes de una estructura MQSCO desde el inicio de la estructura MQCNO. El desplazamiento puede ser positivo o negativo. Este campo es un campo de entrada, con un valor inicial de 0.

Úselo *SSLConfigOffset* solo cuando la aplicación que emite la llamada MQCONN se esté ejecutando como un IBM MQ MQI client. Para obtener información sobre cómo utilizar este campo, consulte la descripción del *SSLConfigPtr* campo.

Este campo se ignora si *Version* es menor que MQCNO_VERSION_4.

ConnectionId (MQBYTE24)

ConnectionId es un identificador único de 24 bytes que permite IBM MQ identificar de forma fiable una aplicación. Una aplicación puede utilizar este identificador para la correlación en llamadas PUT y GET. Este parámetro de salida tiene un valor inicial de 24 bytes nulos en todos los lenguajes de programación.

El gestor de colas asigna un identificador único a todas las conexiones, independientemente de cómo se establezcan. Si un MQCONN establece la conexión con un MQCNO versión 5, la aplicación puede determinar el MQCNO (ConnectionId) a partir del MQCNO devuelto. Se garantiza que el identificador asignado es único entre todos los demás identificadores que IBM MQ genera, como CorrelId, MsgID, y GroupId.

Utilice el comando « ConnectionId » para identificar las unidades de trabajo de larga duración mediante el comando PCF «Inquire Connection» o el comando MQSC «DISPLAY CONN». El identificador de sesión de cliente (ConnectionId) utilizado por los comandos MQSC (CONN) se deriva del identificador de sesión de cliente (ConnectionId) que se devuelve aquí. Los comandos PCF Inquire y Stop Connection pueden utilizar el valor de retorno de ConnectionId sin modificaciones.

Puede utilizar el servicio de conexión de trabajo (ConnectionId) para forzar el final de una unidad de trabajo de larga duración, especificando el número de conexión de trabajo (ConnectionId) mediante el comando PCF Stop Connection o el comando MQSC STOP CONN. Consulte [Detener conexión y STOP CONN](#) para obtener más información sobre el uso de estos comandos.

Este campo no se devuelve si la versión es inferior a MQCNO_VERSION_5.

La longitud de este campo viene dada por MQ_CONNECTION_ID_LENGTH.

SecurityParmsOffset (MQLONG)

SecurityParmsOffset es la compensación en bytes de la estructura MQCSP desde el inicio de la estructura MQCNO. El desplazamiento puede ser positivo o negativo. Este campo es un campo de entrada, con un valor inicial de 0.

Este campo se ignora si *la versión* es inferior a MQCNO_VERSION_5.

La estructura del MQCSP se define en [“MQCSP - Parámetros de seguridad” en la página 291.](#)

SecurityParmsPtr (PMQCSP)

SecurityParmsPtr es la dirección de la estructura MQCSP, utilizada para especificar un ID de usuario y una contraseña para la autenticación por parte del servicio de autorización. Este campo es un campo de entrada y su valor inicial es un puntero nulo o bytes nulos.

Este campo se ignora si *la versión* es inferior a MQCNO_VERSION_5.

La estructura del MQCSP se define en [“MQCSP - Parámetros de seguridad” en la página 291.](#)

Reserved

Un campo reservado para rellenar la estructura hasta un límite de 64 bits. El valor inicial del campo es un cero binario para la longitud del campo.

Este campo se ignora si Version es menor que MQCNO_VERSION_6.

CCDTUrlLength (MQLONG)

CCDTUrlLength es la longitud de la cadena identificada por CCDUrlPtr o CCDUrlOffset. Esta cadena contiene un identificador de canal (URL) que identifica la ubicación de la tabla de canales de conexión del cliente que se utilizará para la conexión. El valor inicial del campo es cero. El valor máximo es la longitud máxima de la ruta de archivo del sistema operativo.

Úselo CCDUrlLength solo cuando la aplicación que emite la llamada MQCONN se esté ejecutando como un IBM MQ MQI client.

Esta es una alternativa programática a la configuración de las variables de entorno [MQCHLLIB](#) y [MQCHLTAB](#).

Si la aplicación no se está ejecutando como cliente, CCDUrlLength se ignora.

Este campo se ignora si Version es menor que MQCNO_VERSION_6.

CCDTUrlPtr (PMQCHAR)

CCDTUrlPtr Es un puntero opcional a una cadena que contiene un identificador de canal (URL) para identificar la ubicación de la tabla de canales de conexión del cliente que se utilizará para la conexión. Este campo es un campo de entrada, con un valor inicial de un puntero nulo en los lenguajes de programación que admiten punteros, y una cadena de bytes totalmente nulos en los demás casos.

Úselo CCDUrlPtr solo cuando la aplicación que emite la llamada MQCONN se esté ejecutando como un IBM MQ MQI client.

Importante: Solo se puede utilizar uno de CCDUrlPtr y CCDUrlOffset. La llamada falla con el código de motivo MQRC_CCDT_URL_ERROR si ambos campos son distintos de cero.

Esta es una alternativa programática a la configuración de las variables de entorno [MQCHLLIB](#) y [MQCHLTAB](#).

Si la aplicación no se está ejecutando como cliente, CCDUrlPtr se ignora.

Este campo se ignora si Version es menor que MQCNO_VERSION_6.

CCDTUrlOffset (MQLONG)

CCDTUrlOffset es la compensación en bytes, desde el inicio de la estructura MQCNO, hasta una cadena que contiene un identificador de canal de conexión (URL) que identifica la ubicación de la tabla de canales de conexión del cliente que se utilizará para la conexión. El desplazamiento puede ser positivo o negativo y el valor inicial del campo es cero.

Úselo CCDUrlOffset solo cuando la aplicación que emite la llamada MQCONN se esté ejecutando como un IBM MQ MQI client.

Importante: Solo se puede utilizar uno de `CCDTUrlPtr` y `CCDTUrlOffset`. La llamada falla con el código de motivo `MQRC_CCDT_URL_ERROR` si ambos campos son distintos de cero.

Esta es una alternativa programática a la configuración de las variables de entorno [MQCHLLIB](#) y [MQCHLTAB](#).

Si la aplicación no se está ejecutando como cliente, `CCDTUrlOffset` se ignora.

Este campo se ignora si `Version` es menor que `MQCNO_VERSION_6`.

ApplName (MQCHAR28)

El nombre establecido por la aplicación para identificar la conexión con el gestor de colas. El valor inicial del campo es `MQAN_NONE_ARRAY` (caracteres en blanco).

Este campo se ignora si `Version` es menor que `MQCNO_VERSION_7`, o si el valor está establecido en espacios en blanco.

 No se puede establecer este campo en z/OS. Si intenta hacerlo, recibirá el código de motivo `MQRC_CNO_ERROR`.

Reserved2 (MQBYTE4)

Un campo reservado para rellenar la estructura hasta un límite de 64 bits. El valor inicial del campo es un cero binario para la longitud del campo.

Este campo se ignora si `Version` es menor que `MQCNO_VERSION_7`.

BalanceParmsOffset

La ubicación de memoria para una estructura de tipo `MQBNO` que contiene información sobre el comportamiento de equilibrio de la aplicación. La estructura se ignora por completo a menos que la aplicación se conecte a través de un canal de cliente.

Este campo se ignora si `Version` es menor que `MQCNO_VERSION_8`.

Consulte [MQBNO](#) para obtener más información.

Si rellenas este campo, no podrás rellenar el “[BalanceParmsPtr](#)” en la [página 290](#) campo. Si intenta proporcionar ambos campos, recibirá un error `MQRC_CNO_ERROR`. Dado que este campo solo es relevante para las conexiones de cliente, proporcionar este campo en cualquier otro tipo de conexión también da como resultado `MQRC_CNO_ERROR`.

BalanceParmsPtr

Puntero a la ubicación de memoria para una estructura de tipo `MQBNO` que contiene información sobre el comportamiento de equilibrio de la aplicación. La estructura se ignora por completo a menos que la aplicación se conecte a través de un canal de cliente.

Este campo se ignora si `Version` es menor que `MQCNO_VERSION_8`.

Consulte [MQBNO](#) para obtener más información.

Si rellenas este campo, no podrás rellenar el “[BalanceParmsOffset](#)” en la [página 290](#) campo. Si intenta proporcionar ambos campos, recibirá un error `MQRC_CNO_ERROR`. Dado que este campo solo es relevante para las conexiones de cliente, proporcionar este campo en cualquier otro tipo de conexión también da como resultado `MQRC_CNO_ERROR`.

Tareas relacionadas

[Utilización de MQCONN](#)

MQCSP - Parámetros de seguridad

La estructura del parámetro de seguridad de IBM MQ conexión es utilizada por las aplicaciones para transmitir información de autenticación en una llamada MQCONN al gestor de colas. También puede utilizarlo para proporcionar la clave inicial que se utiliza con el sistema de IBM MQ protección con contraseña que cifra los datos confidenciales.

Establezca *AuthenticationType* en MQCSP_AUTH_USER_ID_AND_PWD para incluir el ID de usuario y la contraseña de la versión 1.

Cuando se proporciona información clave inicial en la versión 2, el *AuthenticationType* valor predeterminado es MQCSP_AUTH_NONE.

Desde IBM MQ 9.3.4, utilice *AuthenticationType* para incluir la información del token de autenticación.

Puede utilizar MQCSP_AUTH_USER_ID_AND_PWD o MQCSP_AUTH_ID_TOKEN, pero no ambos.

Aviso: En algunos casos, la contraseña o el token de autenticación de una estructura MQCSP para una aplicación cliente se envía a través de la red en texto sin formato. Para garantizar que las contraseñas y los tokens de autenticación de las aplicaciones cliente estén protegidos adecuadamente, consulte [Protección de contraseñas MQCSP](#).

Disponibilidad

La estructura MQCSP está disponible en todas las plataformas IBM MQ compatibles.

Versión

Los archivos de encabezado, COPY e INCLUDE proporcionados para los lenguajes de programación compatibles contienen la versión más reciente de MQCSP, pero con el valor inicial del *Version* campo establecido en MQCSP_VERSION_1. Para utilizar campos que no están presentes en la estructura version-1, la aplicación debe establecer el *Version* campo en el número de versión requerido.

Conjunto de caracteres y codificación

Los datos en MQCSP deben estar en el juego de caracteres y la codificación del gestor de colas local, que se indican mediante el atributo del **CodedCharSetId** gestor de colas y MQENC_NATIVE, respectivamente.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 489. Campos en MQCSP para MQCSP		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>StrucId</u> (identificador de estructura)	MQCSP_STRUC_ID	'CSP-'
<u>Versión</u> (número de versión de la estructura)	MQCSP_VERSION_1	1
<u>AuthenticationType</u> (tipo de autenticación)	Ninguna	MQCSP_AUTH_NONE
<u>Reserved1</u> (requerido para la alineación del puntero en IBM i)	Ninguna	Cadena nula o espacios en blanco
<u>CSPUserIdPtr</u> (dirección de ID de usuario)	Ninguna	Puntero nulo o bytes nulos
<u>CSPUserIdOffset</u> (desplazamiento del ID de usuario)	Ninguna	0

Tabla 489. Campos en MQCSP para MQCSP (continuación)

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>CSPUserIdLength</u> (longitud del ID de usuario)	Ninguna	0
<u>Reserved2</u> (requerido para la alineación del puntero en IBM i)	Ninguna	Cadena nula o espacios en blanco
<u>CSPPasswordPtr</u> (dirección de la contraseña)	Ninguna	Puntero nulo o bytes nulos
<u>CSPPasswordOffset</u> (desplazamiento de la contraseña)	Ninguna	0
<u>CSPPasswordLength</u> (longitud de la contraseña)	Ninguna	0
Nota: Los campos restantes se ignoran si <i>Version</i> es menor que MQCSP_VERSION_2.		
<u>Reserved3</u> (requerido para la alineación del puntero en IBM i)	Ninguna	Cadena nula o espacios en blanco
<u>InitialKeyPtr</u>	Ninguna	Puntero nulo o espacios en blanco
<u>InitialKeyOffset</u> (desplazamiento de la clave inicial para el sistema de protección con contraseña)	Ninguna	0
<u>InitialKeyLength</u> (longitud de la clave inicial para el sistema de protección con contraseña)	Ninguna	0
Nota: Los campos restantes se ignoran si <i>Version</i> es menor que MQCSP_VERSION_3.		
<u>Reserved4</u> (requerido para la alineación del puntero en IBM i)	Ninguna	Cadena nula o espacios en blanco
<u>TokenPtr</u> (dirección del token de autenticación)	Ninguna	Puntero nulo o espacios en blanco
<u>TokenKeyOffset</u> (desplazamiento del token de autenticación)	Ninguna	0
<u>TokenLength</u> (longitud del token de autenticación)	Ninguna	0
Notas: 1. El símbolo ~ representa un único carácter en blanco. 2. En el lenguaje de programación C, la variable macroMQCSP_DEFAULT contiene los valores que se enumeran en la tabla. Se puede utilizar de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQCSP MyCSP = {MQCSP_DEFAULT};</pre>		

Declaraciones lingüísticas

Declaración C para MQCSP

```
typedef struct tagMQCSP MQCSP;
struct tagMQCSP {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQLONG     AuthenticationType; /* Type of authentication */
    MQBYTE4    Reserved1;        /* Required for IBM i pointer alignment */
    MQPTR      CSPUserIdPtr;     /* Address of user ID */
}
```



```

    MQLONG      CSPUserIdOffset;    /* Offset of user ID */
    MQLONG      CSPUserIdLength;    /* Length of user ID */
    MQBYTE8     Reserved2;          /* Required for IBM i pointer alignment */
    MQPTR        CSPPasswordPtr;     /* Address of password */
    MQLONG      CSPPasswordOffset;   /* Offset of password */
    MQLONG      CSPPasswordLength;   /* Length of password */
/* Ver:1 */

    MQBYTE8     Reserved3;          /* Required for IBM i pointer alignment */
    MQPTR        InitialKeyPtr;      /* Address of initial key */
    MQLONG      InitialKeyOffset;    /* Offset of initial key */
    MQLONG      InitialKeyLength;    /* Length of initial key */
/* Ver:2 */
};

```

```

typedef struct tagMQCSP MQCSP;
struct tagMQCSP {
    MQCHAR4      StrucId;            /* Structure identifier */
    MQLONG       Version;            /* Structure version number */
    MQLONG       AuthenticationType; /* Type of authentication */
    MQBYTE4      Reserved1;          /* Required for IBM i pointer alignment */
    MQPTR        CSPUserIdPtr;       /* Address of user ID */
    MQLONG      CSPUserIdOffset;     /* Offset of user ID */
    MQLONG      CSPUserIdLength;     /* Length of user ID */
    MQBYTE8     Reserved2;          /* Required for IBM i pointer alignment */
    MQPTR        CSPPasswordPtr;     /* Address of password */
    MQLONG      CSPPasswordOffset;   /* Offset of password */
    MQLONG      CSPPasswordLength;   /* Length of password */
/* Ver:1 */

    MQBYTE8     Reserved3;          /* Required for IBM i pointer alignment */
    MQPTR        InitialKeyPtr;      /* Address of initial key */
    MQLONG      InitialKeyOffset;    /* Offset of initial key */
    MQLONG      InitialKeyLength;    /* Length of initial key */
/* Ver:2 */

    MQBYTE8     Reserved4;          /* Required for IBM i pointer alignment */
    MQPTR        TokenPtr;           /* Address of token */
    MQLONG      TokenOffset;         /* Offset of token */
    MQLONG      TokenLength;         /* Length of token */
/* Ver:3 */
};

```

COBOL declaración para MQCSP

```

**      MQCSP structure
10 MQCSP.
**      Structure identifier
15 MQCSP-STRUCID          PIC X(4).
**      Structure version number
15 MQCSP-VERSION         PIC S9(9) BINARY.
**      Type of authentication
15 MQCSP-AUTHENTICATIONTYPE PIC S9(9) BINARY.
**      Required for IBM i pointer alignment
15 MQCSP-RESERVED1       PIC X(4).
**      Address of user ID
15 MQCSP-CSPUSERIDPTR    POINTER.
**      Offset of user ID
15 MQCSP-CSPUSERIDOFFSET PIC S9(9) BINARY.
**      Length of user ID
15 MQCSP-CSPUSERIDLENGTH PIC S9(9) BINARY.
**      Required for IBM i pointer alignment
15 MQCSP-RESERVED2       PIC X(4).
**      Address of password
15 MQCSP-CSPPASSWORDPTR  POINTER.
**      Offset of password
15 MQCSP-CSPPASSWORDOFFSET PIC S9(9) BINARY.
**      Length of password
15 MQCSP-CSPPASSWORDLENGTH PIC S9(9) BINARY.
** Ver:1 **

**      Reserved
15 MQCSP-RESERVED3       PIC X(8).
**      Address of initial key
15 MQCSP-INITIALKEYPTR   POINTER.
**      Offset of initial key
15 MQCSP-INITIALKEYOFFSET PIC S9(9) BINARY.
**      Length of initial key

```

```

15 MQCSP-INITIALKEYLENGTH PIC S9(9) BINARY.
** Ver:2 **

```

```

** MQCSP structure
10 MQCSP.
** Structure identifier
15 MQCSP-STRUCID PIC X(4).
** Structure version number
15 MQCSP-VERSION PIC S9(9) BINARY.
** Type of authentication
15 MQCSP-AUTHENTICATIONTYPE PIC S9(9) BINARY.
** Required for IBM i pointer alignment
15 MQCSP-RESERVED1 PIC X(4).
** Address of user ID
15 MQCSP-CSPUSERIDPTR POINTER.
** Offset of user ID
15 MQCSP-CSPUSERIDOFFSET PIC S9(9) BINARY.
** Length of user ID
15 MQCSP-CSPUSERIDLENGTH PIC S9(9) BINARY.
** Required for IBM i pointer alignment
15 MQCSP-RESERVED2 PIC X(4).
** Address of password
15 MQCSP-CSPPASSWORDPTR POINTER.
** Offset of password
15 MQCSP-CSPPASSWORDOFFSET PIC S9(9) BINARY.
** Length of password
15 MQCSP-CSPPASSWORDLENGTH PIC S9(9) BINARY.
** Ver:1 **

** Reserved
15 MQCSP-RESERVED3 PIC X(8).
** Address of initial key
15 MQCSP-INITIALKEYPTR POINTER.
** Offset of initial key
15 MQCSP-INITIALKEYOFFSET PIC S9(9) BINARY.
** Length of initial key
15 MQCSP-INITIALKEYLENGTH PIC S9(9) BINARY.
** Ver:2 **

** Reserved
15 MQCSP-RESERVED4 PIC X(8).
** Address of token
15 MQCSP-TOKENPTR POINTER.
** Offset of token
15 MQCSP-TOKENOFFSET PIC S9(9) BINARY.
** Length of token
15 MQCSP-TOKENLENGTH PIC S9(9) BINARY.
** Ver:3 **

```

PL/I declaración para MQCSP

```

dcl
  1 MQCSP based,
    3 StrucId          char(4),          /* Structure identifier */
    3 Version          fixed bin(31),    /* Structure version number */
    3 AuthenticationType fixed bin(31),  /* Type of authentication */
    3 Reserved1        char(4),          /* Required for IBM i pointer
                                          alignment */
    3 CSPUserIdPtr      pointer,          /* Address of user ID */
    3 CSPUserIdOffset   fixed bin(31),    /* Offset of user ID */
    3 CSPUserIdLength   fixed bin(31),    /* Length of user ID */
    3 Reserved2        char(8),          /* Required for IBM i pointer
                                          alignment */
    3 CSPPasswordPtr    pointer,          /* Address of password */
    3 CSPPasswordOffset fixed bin(31),    /* Offset of user ID */
    3 CSPPasswordLength fixed bin(31),    /* Length of user ID */
/* Version 1 */

    3 Reserved3        char(8),          /* Reserved */
    3 InitialKeyPtr     pointer,          /* Address of initial key */
    3 InitialKeyOffset   fixed bin(31),    /* Offset of initial key */
    3 InitialKeyLength   fixed bin(31); /* Length of initial key */
/* Version 2 */

```

```

dcl
  1 MQCSP based,

```

```

3 StrucId          char(4),          /* Structure identifier */
3 Version          fixed bin(31),    /* Structure version number */
3 AuthenticationType fixed bin(31),  /* Type of authentication */
3 Reserved1        char(4),          /* Required for IBM i pointer
                                     alignment */
3 CSPUserIdPtr     pointer,           /* Address of user ID */
3 CSPUserIdOffset  fixed bin(31),    /* Offset of user ID */
3 CSPUserIdLength  fixed bin(31),    /* Length of user ID */
3 Reserved2        char(8),          /* Required for IBM i pointer
                                     alignment */
3 CSPPasswordPtr   pointer,           /* Address of password */
3 CSPPasswordOffset fixed bin(31),    /* Offset of user ID */
3 CSPPasswordLength fixed bin(31),    /* Length of user ID */
/* Version 1 */

3 Reserved3        char(8),          /* Reserved */
3 InitialKeyPtr    pointer,           /* Address of initial key */
3 InitialKeyOffset fixed bin(31),    /* Offset of initial key */
3 InitialKeyLength fixed bin(31);    /* Length of initial key */
/* Version 2 */

3 Reserved4        char(8),          /* Reserved */
3 TokenPtr         pointer,           /* Address of Token */
3 TokenOffset      fixed bin(31),    /* Offset of Token */
3 TokenLength      fixed bin(31);    /* Length of Token */
/* Version 3 */

```

Declaración de Visual Basic para MQCSP

```

Type MQCSP
  StrucId          As String*4      'Structure identifier'
  Version          As Long          'Structure version number'
  AuthenticationType As Long        'Type of authentication'
  Reserved1        As MBYTE4       'Required for IBM i pointer'
                                     'alignment'
  CSPUserIdPtr     As MQPTR         'Address of user ID'
  CSPUserIdOffset  As Long          'Offset of user ID'
  CSPUserIdLength  As Long          'Length of user ID'
  Reserved2        As MBYTE8       'Required for IBM i pointer'
                                     'alignment'
  CSPPasswordPtr   As MQPTR         'Address of password'
  CSPPasswordOffset As Long          'Offset of password'
  CSPPasswordLength As Long          'Length of password'
End Type

```

Detalles de campo para MQCSP

La estructura del parámetro de seguridad de IBM MQ conexión es utilizada por las aplicaciones para transmitir información de autenticación en una llamada MQCONN al gestor de colas. También puede utilizarlo para proporcionar la clave inicial que se utiliza con el sistema de IBM MQ protección con contraseña que cifra los datos confidenciales.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 490. Campos en MQCSP para MQCSP		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>StrucId</u> (identificador de estructura)	MQCSP_STRUC_ID	'CSP-'
<u>Versión</u> (número de versión de la estructura)	MQCSP_VERSION_1	1
<u>AuthenticationType</u> (tipo de autenticación)	Ninguna	MQCSP_AUTH_NONE
<u>Reserved1</u> (requerido para la alineación del puntero en IBM i)	Ninguna	Cadena nula o espacios en blanco
<u>CSPUserIdPtr</u> (dirección de ID de usuario)	Ninguna	Puntero nulo o bytes nulos

Tabla 490. Campos en MQCSP para MQCSP (continuación)

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>CSPUserIdOffset</u> (desplazamiento del ID de usuario)	Ninguna	0
<u>CSPUserIdLength</u> (longitud del ID de usuario)	Ninguna	0
Reserved2 (requerido para la alineación del puntero en IBM i)	Ninguna	Cadena nula o espacios en blanco
<u>CSPPasswordPtr</u> (dirección de la contraseña)	Ninguna	Puntero nulo o bytes nulos
<u>CSPPasswordOffset</u> (desplazamiento de la contraseña)	Ninguna	0
<u>CSPPasswordLength</u> (longitud de la contraseña)	Ninguna	0
Nota: Los campos restantes se ignoran si <i>Version</i> es menor que MQCSP_VERSION_2.		
Reserved3 (requerido para la alineación del puntero en IBM i)	Ninguna	Cadena nula o espacios en blanco
<u>InitialKeyPtr</u>	Ninguna	Puntero nulo o espacios en blanco
<u>InitialKeyOffset</u> (desplazamiento de la clave inicial para el sistema de protección con contraseña)	Ninguna	0
<u>InitialKeyLength</u> (longitud de la clave inicial para el sistema de protección con contraseña)	Ninguna	0
Nota: Los campos restantes se ignoran si <i>Version</i> es menor que MQCSP_VERSION_3.		
Reserved4 (requerido para la alineación del puntero en IBM i)	Ninguna	Cadena nula o espacios en blanco
<u>TokenPtr</u> (dirección del token de autenticación)	Ninguna	Puntero nulo o espacios en blanco
<u>TokenKeyOffset</u> (desplazamiento del token de autenticación)	Ninguna	0
<u>TokenLength</u> (longitud del token de autenticación)	Ninguna	0
Notas: 1. El símbolo ~ representa un único carácter en blanco. 2. En el lenguaje de programación C, la variable macro MQCSP_DEFAULT contiene los valores que se enumeran en la tabla. Se puede utilizar de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQCSP MyCSP = {MQCSP_DEFAULT};</pre>		

StrucId (MQCHAR4)

Este es el identificador de estructura de la estructura de parámetros de seguridad. Siempre es un campo de entrada. Su valor es MQCSP_STRUC_ID.

El valor debe ser:

MQCSP_STRUC_ID

Identificador de la estructura de parámetros de seguridad.

Para el lenguaje de programación C, también se define la constante MQCSP_STRUC_ID_ARRAY. Tiene el mismo valor que MQCSP_STRUC_ID, pero es una matriz de caracteres en lugar de una cadena.

Versión (MQLONG)

Número de versión de la estructura MQCSP.

El valor debe ser:

MQCSP_VERSION_1

Version-1 Estructura de los parámetros de seguridad. En la versión 1, puede incluir un ID de usuario y una contraseña en la estructura MQCSP para autenticarse con el gestor de colas.

MQCSP_VERSION_2

Version-2 Estructura de los parámetros de seguridad. En la versión 2, puede incluir un ID de usuario y una contraseña para autenticarse con el administrador de colas y especificar la clave inicial que se utiliza para proteger las contraseñas.

MQCSP_VERSION_3

Version-3 Estructura de los parámetros de seguridad. En la versión 3, puede incluir un ID de usuario y una contraseña o un token de autenticación en la estructura MQCSP para autenticarse con el gestor de colas. También puede especificar la clave inicial que se utiliza para proteger las contraseñas.

La siguiente constante especifica el número de versión de la versión actual:

MQCSP_VERSIÓN_ACTUAL

Versión actual de la estructura de los parámetros de seguridad.

Este campo siempre es un campo de entrada. El valor inicial de este campo es MQCSP_VERSION_3.

AuthenticationType (MQLONG)

AuthenticationType es un campo de entrada. Su valor inicial es MQCSP_AUTH_NONE.

Este es el tipo de autenticación que se debe realizar. Los valores válidos son:

MQCSP_AUTH_NONE

No utilice los campos de ID de usuario y contraseña o token de autenticación.

MQCSP_AUTH_ID_DE_USUARIO_Y_CONTRASEÑA

Auténtíquese utilizando el ID de usuario y la contraseña en la estructura MQCSP.

MQCSP_AUTH_ID_TOKEN

Autenticar utilizando el token de autenticación en la estructura MQCSP.

El valor predeterminado es MQCSP_AUTH_NONE. Con el valor predeterminado, no se lleva a cabo ninguna protección por contraseña.

Si necesita autenticación, debe configurar **MQCSP.AuthenticationType** a MQCSP_AUTH_USER_ID_AND_PWD o MQCSP_AUTH_ID_TOKEN.

Para obtener más información, consulte

- [Protección por contraseña MQCSP](#)
- [Trabajar con tokens de autenticación](#)

Reserved1 (MQBYTE4)



Un campo reservado, necesario para la alineación del puntero en IBM i.

El valor inicial de este campo es nulo.

CSPUserIdPtr (MQPTR)

La dirección para el ID de usuario que se utilizará en la autenticación.

Este es un campo de entrada. El valor inicial de este campo es el puntero nulo en aquellos lenguajes de programación que admiten punteros y, en los demás casos, una cadena de bytes totalmente nulos. Este campo se ignora si *Version* es menor que MQCNO_VERSION_5.

Este campo puede contener un ID de usuario del sistema operativo cuando se nombra un **AUTHTYPE** de *IDPWOS* en el campo CONNAUTH del gestor de colas.

En Windows esto puede ser un ID de usuario de dominio totalmente calificado.

Este campo puede contener un ID de usuario LDAP cuando se nombra un **AUTHTYPE** de *IDPWLDAP* en el campo CONNAUTH del gestor de colas.

CSPUserIdOffset (MQLONG)

El desplazamiento en bytes para el ID de usuario que se utilizará en la autenticación. El desplazamiento puede ser positivo o negativo.

Este es un campo de entrada. El valor inicial de este campo es 0.

CSPUserIdLength (MQLONG)

La longitud del ID de usuario que se utilizará en la autenticación.

La longitud máxima del ID de usuario depende de la plataforma; consulte ID de usuario. Si la longitud del ID de usuario es mayor que la longitud máxima permitida, la solicitud de autenticación falla con MQRC_CSP_ERROR. En versiones anteriores de IBM MQ, el error devuelto es MQRC_NOT_AUTHORIZED.

Este campo es un campo de entrada. El valor inicial de este campo es 0.

Reserved2 (MQBYTE8)



Un campo reservado, necesario para la alineación del puntero en IBM i.

El valor inicial de este campo es nulo.

CSPPasswordPtr (MQPTR)

La dirección para la contraseña que se utilizará en la autenticación.

Este es un campo de entrada. El valor inicial de este campo es el puntero nulo en aquellos lenguajes de programación que admiten punteros y, en los demás casos, una cadena de bytes totalmente nulos. Este campo se ignora si *Version* es menor que MQCNO_VERSION_5.

Este campo puede contener una contraseña vacía que es rechazada por el sistema operativo o por la comprobación de contraseñas de LDAP, dependiendo de la configuración, pero no es rechazada por IBM MQ antes de ser pasada al método de autenticación.

CSPPasswordOffset (MQLONG)

Este es el desplazamiento en bytes para la contraseña que se utilizará en la autenticación. El desplazamiento puede ser positivo o negativo.

Este es un campo de entrada. El valor inicial de este campo es 0.

CSPPasswordLength (MQLONG)

La longitud de la contraseña que se utilizará en la autenticación.

La longitud máxima de la contraseña es MQ_CSP_PASSWORD_LENGTH, que es de 256 caracteres. Si la longitud de la contraseña es superior a la longitud máxima permitida, la solicitud de autenticación falla con MQRC_CSP_ERROR. En versiones anteriores de IBM MQ, el error devuelto es MQRC_NOT_AUTHORIZED.

Este campo es un campo de entrada. El valor inicial de este campo es 0.

Reserved3 (MQBYTE8)



Un campo reservado, necesario para la alineación del puntero en IBM i.

El valor inicial de este campo es nulo.

InitialKeyPtr (MQPTR)



La dirección de la clave inicial para el sistema de protección con contraseña.

Este es un campo de entrada. El valor inicial de este campo es el puntero nulo en aquellos lenguajes de programación que admiten punteros y, en los demás casos, una cadena de bytes totalmente nulos. Este campo se ignora si *Version* es menor que MQCSP_VERSION_2.

Este campo solo es relevante para IBM MQ los clientes de « MQI » que se ejecutan en entornos multiplataforma.

IBM MQMQI Los clientes pueden introducir valores cifrados en algunos campos mediante el IBM MQ sistema de protección con contraseña. Si ha utilizado una clave inicial para cifrar la contraseña del repositorio de claves especificado en la estructura MQSCO, asegúrese de incluir los campos de clave inicial en la estructura MQCSP para la misma aplicación cliente.

El algoritmo de cifrado utiliza una clave inicial para cifrar y descifrar estos valores. Si se proporciona una clave inicial cuando los valores de estos campos se cifran utilizando la **runmqicred** utilidad, el cliente deberá especificar la misma clave inicial cuando se conecte al gestor de colas.

La clave inicial especificada mediante este campo anula cualquier clave inicial especificada mediante la variable de entorno MQS_MQI_KEYFILE o la MQIInitialKeyFile propiedad en la sección Security del archivo de configuración del cliente.

Puede utilizar o *InitialKeyPtrInitialKeyOffset* para especificar la clave inicial, pero no ambos.

Consulte también el apartado

- [Proporcionar una clave inicial para un IBM MQ MQI client en AIX, Linux, and Windows](#)
- [Protección de contraseñas en archivos de configuración IBM MQ de componentes](#)
- [runmqicred \(proteger las IBM MQ contraseñas de los clientes\)](#)
- [“KeyRepoPasswordPtr \(MQPTR\)” en la página 556](#)

InitialKeyOffset (MQLONG)



El desplazamiento en bytes de la clave inicial del sistema de protección con contraseña desde el inicio de la estructura MQCSP. El desplazamiento puede ser positivo o negativo.

Puede utilizar o *InitialKeyPtrInitialKeyOffset* para especificar la clave inicial, pero no ambos. Para obtener más información, consulte la descripción del *InitialKeyPtr* campo.

Este es un campo de entrada. El valor inicial de este campo es 0. Este campo se ignora si *Version* es menor que MQCSP_VERSION_2.

Consulte también el apartado

-  Proporcionar una clave inicial para un IBM MQ MQI client en AIX, Linux, and Windows
- [Protección de contraseñas en archivos de configuración IBM MQ de componentes](#)

InitialKeyLength (MQLONG)



La longitud de la clave inicial para el sistema de protección con contraseña.

Este es un campo de entrada. El valor inicial de este campo es 0. Este campo se ignora si *Version* es menor que MQCSP_VERSION_2.

Reserved4 (MQBYTE8)



Un campo reservado, necesario para la alineación del puntero en IBM i.

El valor inicial de este campo es nulo.

TokenPtr (MQPTR)



La dirección del token de autenticación que se utiliza para la autenticación con el gestor de colas.

Este es un campo de entrada. El valor inicial de este campo es el puntero nulo en aquellos lenguajes de programación que admiten punteros y, en los demás casos, una cadena de bytes totalmente nulos. Este campo se ignora si *Version* es menor que MQCSP_VERSION_3.

Este campo es relevante para IBM MQ MQI clients conectarse a IBM MQ gestores de colas que se ejecutan en AIX o Linux sistemas.

Puede utilizar o *TokenPtrTokenOffset* para especificar el token de autenticación, pero no ambos.

Consulte también el apartado

- [Uso de tokens de autenticación en una aplicación](#)
- [Trabajar con tokens de autenticación](#)

TokenOffset (MQLONG)



Este es el desplazamiento en bytes para el token de autenticación desde el inicio de la estructura MQCSP. El desplazamiento puede ser positivo o negativo.

Este es un campo de entrada. El valor inicial de este campo es 0. Este campo se ignora si *Version* es menor que MQCSP_VERSION_3.

Puede utilizar o *TokenPtrTokenOffset* para especificar el token, pero no ambos. Para obtener más información, consulte la descripción del [TokenPtr](#) campo.

Consulte también el apartado

- [Uso de tokens de autenticación en una aplicación](#)
- [Trabajar con tokens de autenticación](#)

TokenLength (MQLONG)



Esta es la longitud del token de autenticación utilizado para la autenticación con el gestor de colas.

La longitud máxima del token de autenticación es MQ_CSP_TOKEN_LENGTH, que es de 8192 bytes. Si es *TokenLength* mayor que la longitud máxima permitida, la solicitud de autenticación falla con MQRC_CSP_ERROR.

Este es un campo de entrada. El valor inicial de este campo es 0. Este campo se ignora si *Version* es menor que MQCSP_VERSION_3.

Consulte también el apartado

- [Uso de tokens de autenticación en una aplicación](#)
- [Trabajar con tokens de autenticación](#)

MQCTLO - Estructura de opciones de devolución de llamada de control

La estructura MQCTLO se utiliza para especificar opciones relacionadas con una función de devolución de llamada de control. La estructura es un parámetro de entrada y salida en la llamada MQCTL.

Disponibilidad

La estructura MQCTLO está disponible en las siguientes plataformas:

-  AIX
-  IBM i
-  Linux
-  Windows
-  z/OS

y para IBM MQ MQI clients conectados a estos sistemas.

Versión

La versión actual de MQCTLO es MQCTLO_VERSION_1.

Conjunto de caracteres y codificación

Los datos en MQCTLO deben estar en el juego de caracteres proporcionado por el atributo del **CodedCharSetId** gestor de colas y la codificación del gestor de colas local proporcionada por MQENC_NATIVE. Sin embargo, si la aplicación se ejecuta como un cliente MQ MQI, la estructura debe estar en el conjunto de caracteres y la codificación del cliente.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 491. Campos en MQCTLO		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
StrucID (identificador de estructura)	MQCTLO_STRUC_ID	'CTLO'
Versión (número de versión de la estructura)	MQCTLO_VERSION_1	1
Opciones (opciones)	MQCTLO_NINGUNO	Nulos
Opciones (reservadas)	Campo reservado	
ConnectionArea (campo para la función de devolución de llamada que se va a utilizar)	Ninguna	Puntero nulo o bytes nulos

Tabla 491. Campos en MQCTLO (continuación)		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
Notas: 1. En el lenguaje de programación C, la variable macroMQCTLO_DEFAULT contiene los valores que se enumeran en la tabla. Úselo de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQCTLO MyCTLO = {MQCTLO_DEFAULT};</pre>		

Declaraciones lingüísticas

Declaración C para MQCTLO

```
typedef struct tagMQCTLO MQCTLO;
struct tagMQCTLO {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQLONG     Options;          /* Options that control the action of MQCTL */
    MQLONG     Reserved;         /* Reserved field */

    MQPTR      ConnectionArea; /* Connection work area passed to the function */
};
```

COBOL Declaración para MQCTLO

```
** MQCTLO structure
10  MQCTLO.
** Structure Identifier
15  MQCTLO-STRUCID                PIC X(4).
** Structure Version
15  MQCTLO-VERSION               PIC S9(9) BINARY.
** Options
15  MQCTLO-OPTIONS              PIC S9(9) BINARY.
** Reserved
15  MQCTLO-RESERVED              PIC S9(9) BINARY.
** ConnectionArea
15  MQCTLO-CONNECTIONAREA        POINTER
```

PL/I Declaración para MQCTLO

```
dcl
1  MQCTLO based,
3  StrucId          char(4),           /* Structure identifier */
3  Version          fixed bin(31),    /* Structure version */
3  Options          fixed bin(31),    /* Options */
3  Reserved         fixed bin(31),
3  ConnectionArea   pointer;          /* Connection work area */
```

Detalles de campo para MQCTLO

La estructura MQCTLO se utiliza para especificar opciones relacionadas con una función de devolución de llamada de control. La estructura es un parámetro de entrada y salida en la llamada MQCTL.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 492. Campos en MQCTLO

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
StrucID (identificador de estructura)	MQCTLO_STRUC_ID	'CTLO'
Versión (número de versión de la estructura)	MQCTLO_VERSION_1	1
Opciones (opciones)	MQCTLO_NINGUNO	Nulos
Opciones (reservadas)	Campo reservado	
ConnectionArea (campo para la función de devolución de llamada que se va a utilizar)	Ninguna	Puntero nulo o bytes nulos
Notas: 1. En el lenguaje de programación C, la variable macroMQCTLO_DEFAULT contiene los valores que se enumeran en la tabla. Úselo de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQCTLO MyCTLO = {MQCTLO_DEFAULT};</pre>		

StrucID (MQCHAR4)

Este es el identificador de estructura de la estructura de opciones de control. Siempre es un campo de entrada. Su valor es MQCTLO_STRUC_ID.

El valor debe ser:

MQCTLO_STRUC_ID

Identificador de la estructura de opciones de control.

Para el lenguaje de programación C, también se define la constante MQCTLO_STRUC_ID_ARRAY. Tiene el mismo valor que MQCTLO_STRUC_ID, pero es una matriz de caracteres en lugar de una cadena.

Versión (MQLONG)

Este es el número de versión de la estructura; el valor debe ser:

MQCTLO_VERSION_1

Version-1 Estructura de opciones de control.

La siguiente constante especifica el número de versión de la versión actual:

MQCTLO_VERSIÓN_ACTUAL

Versión actual de la estructura de opciones de control.

Este campo siempre es un campo de entrada. El valor inicial de este campo es MQCTLO_VERSION_1.

Opciones (MQLONG)

Opciones que controlan la acción de MQCTL.

MQCTLO_FAIL_IF QUIESCING

Forzar el fallo de la llamada MQCTL si el gestor de colas o la conexión se encuentran en estado de inactividad.

Especifique MQGMO_FAIL_IF QUIESCING en las opciones MQGMO pasadas en la llamada MQCB para que se envíe una notificación a los consumidores de mensajes cuando estén inactivos.

Afinidad de subprocesos MQCTLO

Esta opción informa al sistema de que la aplicación requiere que todos los consumidores de mensajes, para la misma conexión, se llamen en el mismo subproceso. Este hilo se utilizará para todas las invocaciones de los consumidores hasta que se detenga la conexión.

Opción predeterminada : Si no necesita ninguna de las opciones descritas, utilice la siguiente opción:

MQCTLO_NINGUNO

Utilice este valor para indicar que no se han especificado otras opciones; todas las opciones toman sus valores predeterminados. MQCTLO_NONE se define para facilitar la documentación del programa; no está previsto que esta opción se utilice con ninguna otra, pero como su valor es cero, dicho uso no puede detectarse.

Este es un campo de entrada. El valor inicial del campo *Options* es MQCTLO_NONE.

Reservado (MQLONG)

Este es un campo reservado. El valor debe ser cero.

ConnectionArea (MQPTR)

Este es un campo que está disponible para que lo utilice la función de devolución de llamada.

El gestor de colas no toma ninguna decisión basada en el contenido de este campo y se transfiere sin cambios al ConnectionArea campo de la estructura MQCBC, que es un parámetro de entrada para la devolución de llamada.

Este campo se ignora para todas las operaciones excepto MQOP_START y MQOP_START_WAIT.

Este es un campo de entrada y salida para la función de devolución de llamada. El valor inicial de este campo es un puntero nulo o bytes nulos.

MQDH - Cabecera de distribución

La estructura MQDH describe los datos adicionales que están presentes en un mensaje cuando ese mensaje es un mensaje de lista de distribución almacenado en una cola de transmisión. Un mensaje de lista de distribución es un mensaje que se envía a varias colas de destino. Los datos adicionales consisten en la estructura MQDH seguida de una matriz de registros MQOR y una matriz de registros MQPMR. Esta estructura es utilizada por aplicaciones especializadas que colocan mensajes directamente en colas de transmisión o que eliminan mensajes de colas de transmisión (por ejemplo: agentes de canales de mensajes). Las aplicaciones que deseen enviar mensajes a listas de distribución no deben utilizar esta estructura. En su lugar, deben utilizar la estructura MQOD para definir los destinos en la lista de distribución y la estructura MQPMO para especificar las propiedades de los mensajes o recibir información sobre los mensajes enviados a los destinos individuales.

Disponibilidad

La estructura MQDH está disponible en las siguientes plataformas:

-  AIX
-  IBM i
-  Linux
-  Windows

y para IBM MQ MQI clients conectados a estos sistemas.

Nombre de formato

MQFMT_DIST_HEADER

Conjunto de caracteres y codificación

Los datos en MQDH deben estar en el juego de caracteres proporcionado por el atributo del **CodedCharSetId** gestor de colas y la codificación del gestor de colas local proporcionada por MQENC_NATIVE.

Configure el juego de caracteres y la codificación del MQDH en los campos *CodedCharSetId* *Encoding* y en:

- El MQMD (si la estructura MQDH se encuentra al inicio de los datos del mensaje), o
- La estructura de encabezado que precede a la estructura MQDH (todos los demás casos).

Utilización

Cuando una aplicación envía un mensaje a una lista de distribución y algunos o todos los destinos son remotos, el gestor de colas antepone las estructuras MQXQH y MQDH a los datos del mensaje de la aplicación y coloca el mensaje en la cola de transmisión correspondiente. Por lo tanto, los datos aparecen en la siguiente secuencia cuando el mensaje se encuentra en una cola de transmisión:

- estructura MQXQH
- Estructura MQDH más matrices de registros MQOR y MQPMR
- Datos del mensaje de la aplicación

Dependiendo de los destinos, el gestor de colas puede generar más de un mensaje de este tipo y colocarlo en diferentes colas de transmisión. En este caso, las estructuras MQDH de esos mensajes identifican diferentes subconjuntos de los destinos definidos por la lista de distribución abierta por la aplicación.

Una aplicación que coloca un mensaje de lista de distribución directamente en una cola de transmisión debe ajustarse a la secuencia descrita anteriormente y debe garantizar que la estructura MQDH sea correcta. Si la estructura MQDH no es válida, el gestor de colas puede rechazar la llamada MQPUT o MQPUT1 con el código de motivo MQRC_DH_ERROR.

Solo puede almacenar mensajes en una cola en forma de lista de distribución si ha definido la cola como capaz de admitir mensajes de lista de distribución. Consulte el **DistLists** atributo cola descrito en “Atributos para colas” en la página 841. Si una aplicación coloca un mensaje de lista de distribución directamente en una cola que no admite listas de distribución, el gestor de colas divide el mensaje de lista de distribución en mensajes individuales y los coloca en la cola.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 493. Campos en MQDH		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>StrucId</u> (identificador de estructura)	MQDH_STRUC_ID	'DH→→'
<u>Versión</u> (número de versión de la estructura)	MQDH_VERSION_1	1
<u>StrucLength</u> (longitud de la estructura MQDH más los registros siguientes)	Ninguna	0
<u>Codificación</u> (codificación numérica de los datos que siguen a la matriz de registros MQPMR)	Ninguna	0
<u>CodedCharSetId</u> (identificador del conjunto de caracteres de los datos que siguen a la matriz de registros MQPMR)	MQCCSI_INDETERMINADO	0

Tabla 493. Campos en MQDH (continuación)

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>Formato</u> (nombre del formato de los datos que siguen a la matriz de registros MQPMR)	MQFMT_NONE	Espacios en blanco
<u>Banderas</u> (banderas generales)	MQDHF_NINGUNO	0
<u>PutMsgRecFields</u> (indicadores que señalan qué campos MQPMR están presentes)	MQPMRF_NONE	0
<u>RecsPresent</u> (número de registros de objetos presentes)	Ninguna	0
<u>ObjectRecOffset</u> (desplazamiento del primer registro de objeto desde el inicio de MQDH)	Ninguna	0
<u>PutMsgRecOffset</u> (desplazamiento del primer registro de mensaje enviado desde el inicio de MQDH)	Ninguna	0
Notas: 1. El símbolo ~ representa un único carácter en blanco. 2. En el lenguaje de programación C, la variable macroMQDH_DEFAULT contiene los valores que se enumeran en la tabla. Úselo de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQDH MyDH = {MQDH_DEFAULT};</pre>		

Declaraciones lingüísticas

Declaración C para MQDH

```
typedef struct tagMQDH MQDH;
struct tagMQDH {
    MQCHAR4  StrucId;           /* Structure identifier */
    MQLONG   Version;          /* Structure version number */
    MQLONG   StrucLength;       /* Length of MQDH structure plus following
                                MQOR and MQPMR records */
    MQLONG   Encoding;         /* Numeric encoding of data that follows
                                the MQOR and MQPMR records */
    MQLONG   CodedCharSetId;    /* Character set identifier of data that
                                follows the MQOR and MQPMR records */
    MQCHAR8  Format;           /* Format name of data that follows the
                                MQOR and MQPMR records */
    MQLONG   Flags;            /* General flags */
    MQLONG   PutMsgRecFields;   /* Flags indicating which MQPMR fields are
                                present */
    MQLONG   RecsPresent;       /* Number of MQOR records present */
    MQLONG   ObjectRecOffset;   /* Offset of first MQOR record from start
                                of MQDH */
    MQLONG   PutMsgRecOffset;   /* Offset of first MQPMR record from start
                                of MQDH */
};
```

COBOL declaración para MQDH

```
**  MQDH structure
10  MQDH.
**    Structure identifier
15  MQDH-STRUCID          PIC X(4).
**    Structure version number
15  MQDH-VERSION          PIC S9(9) BINARY.
**    Length of MQDH structure plus following MQOR and MQPMR records
15  MQDH-STRUCLength     PIC S9(9) BINARY.
**    Numeric encoding of data that follows the MQOR and MQPMR records
```

```

15 MQDH-ENCODING          PIC S9(9) BINARY.
** Character set identifier of data that follows the MQOR and MQPMR
** records
15 MQDH-CODEDCHARSETID PIC S9(9) BINARY.
** Format name of data that follows the MQOR and MQPMR records
15 MQDH-FORMAT           PIC X(8).
** General flags
15 MQDH-FLAGS           PIC S9(9) BINARY.
** Flags indicating which MQPMR fields are present
15 MQDH-PUTMSGRECFIELDS PIC S9(9) BINARY.
** Number of MQOR records present
15 MQDH-RECSPRESENT     PIC S9(9) BINARY.
** Offset of first MQOR record from start of MQDH
15 MQDH-OBJECTRECOFFSET PIC S9(9) BINARY.
** Offset of first MQPMR record from start of MQDH
15 MQDH-PUTMSGRECOFFSET PIC S9(9) BINARY.

```

PL/I declaración para MQDH

```

dcl
  1 MQDH based,
    3 StrucId          char(4),          /* Structure identifier */
    3 Version          fixed bin(31), /* Structure version number */
    3 StrucLength      fixed bin(31), /* Length of MQDH structure plus
                                     following MQOR and MQPMR
                                     records */
    3 Encoding         fixed bin(31), /* Numeric encoding of data that
                                     follows the MQOR and MQPMR
                                     records */
    3 CodedCharSetId   fixed bin(31), /* Character set identifier of data
                                     that follows the MQOR and MQPMR
                                     records */
    3 Format            char(8),          /* Format name of data that follows
                                     the MQOR and MQPMR records */
    3 Flags            fixed bin(31), /* General flags */
    3 PutMsgRecFields  fixed bin(31), /* Flags indicating which MQPMR
                                     fields are present */
    3 RecsPresent      fixed bin(31), /* Number of MQOR records present */
    3 ObjectRecOffset  fixed bin(31), /* Offset of first MQOR record from
                                     start of MQDH */
    3 PutMsgRecOffset  fixed bin(31); /* Offset of first MQPMR record from
                                     start of MQDH */

```

Declaración de Visual Basic para MQDH

```

Type MQDH
  StrucId          As String*4 'Structure identifier'
  Version          As Long      'Structure version number'
  StrucLength      As Long      'Length of MQDH structure plus following'
                                'MQOR and MQPMR records'
  Encoding         As Long      'Numeric encoding of data that follows'
                                'the MQOR and MQPMR records'
  CodedCharSetId   As Long      'Character set identifier of data that'
                                'follows the MQOR and MQPMR records'
  Format           As String*8  'Format name of data that follows the'
                                'MQOR and MQPMR records'
  Flags            As Long      'General flags'
  PutMsgRecFields  As Long      'Flags indicating which MQPMR fields are'
                                'present'
  RecsPresent      As Long      'Number of MQOR records present'
  ObjectRecOffset  As Long      'Offset of first MQOR record from start'
                                'of MQDH'
  PutMsgRecOffset  As Long      'Offset of first MQPMR record from start'
                                'of MQDH'
End Type

```

Detalles de campo para MQDH

La estructura del encabezado MQDH Distribution describe los datos adicionales que están presentes en un mensaje cuando ese mensaje es un mensaje de lista de distribución almacenado en una cola de transmisión. Un mensaje de lista de distribución es un mensaje que se envía a varias colas de destino. Los datos adicionales consisten en la estructura MQDH seguida de una matriz de registros MQOR y una matriz de registros MQPMR.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 494. Campos en MQDH		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>StrucId</u> (identificador de estructura)	MQDH_STRUC_ID	'DH¬¬'
<u>Versión</u> (número de versión de la estructura)	MQDH_VERSION_1	1
<u>StrucLength</u> (longitud de la estructura MQDH más los registros siguientes)	Ninguna	0
<u>Codificación</u> (codificación numérica de los datos que siguen a la matriz de registros MQPMR)	Ninguna	0
<u>CodedCharSetId</u> (identificador del conjunto de caracteres de los datos que siguen a la matriz de registros MQPMR)	MQCCSI_INDETERMINADO	0
<u>Formato</u> (nombre del formato de los datos que siguen a la matriz de registros MQPMR)	MQFMT_NONE	Espacios en blanco
<u>Banderas</u> (banderas generales)	MQDHF_NINGUNO	0
<u>PutMsgRecFields</u> (indicadores que señalan qué campos MQPMR están presentes)	MQPMRF_NONE	0
<u>RecsPresent</u> (número de registros de objetos presentes)	Ninguna	0
<u>ObjectRecOffset</u> (desplazamiento del primer registro de objeto desde el inicio de MQDH)	Ninguna	0
<u>PutMsgRecOffset</u> (desplazamiento del primer registro de mensaje enviado desde el inicio de MQDH)	Ninguna	0
Notas: 1. El símbolo ¬ representa un único carácter en blanco. 2. En el lenguaje de programación C, la variable macro MQDH_DEFAULT contiene los valores que se enumeran en la tabla. Úselo de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQDH MyDH = {MQDH_DEFAULT};</pre>		

StrucId (MQCHAR4)

Este es el identificador de estructura de la estructura de cabecera de distribución. Siempre es un campo de entrada. Su valor es MQDH_STRUC_ID.

El valor debe ser:

MQDH_STRUC_ID

Identificador de la estructura del encabezado de distribución.

Para el lenguaje de programación C, también se define la constante MQDH_STRUC_ID_ARRAY. Tiene el mismo valor que MQDH_STRUC_ID, pero es una matriz de caracteres en lugar de una cadena.

Versión (MQLONG)

El valor debe ser:

MQDH_VERSION_1

Número de versión para la estructura del encabezado de distribución.

La siguiente constante especifica el número de versión de la versión actual:

MQDH_VERSIÓN_ACTUAL

Versión actual de la estructura del encabezado de distribución.

El valor inicial de este campo es MQDH_VERSION_1.

StrucLength (MQLONG)

Es el número de bytes desde el inicio de la estructura MQDH hasta el inicio de los datos del mensaje que siguen a las matrices de registros MQOR y MQPMR. Los datos se presentan en la siguiente secuencia:

- Estructura MQDH
- Matriz de registros MQOR
- Matriz de registros MQPMR
- Datos de mensaje

Las matrices de registros MQOR y MQPMR se abordan mediante desplazamientos contenidos en la estructura MQDH. Si estas compensaciones dan lugar a bytes no utilizados entre una o varias de las estructuras MQDH, las matrices de registros y los datos del mensaje, dichos bytes no utilizados deben incluirse en el valor de *StrucLength*, pero el gestor de colas no conserva el contenido de dichos bytes. Es válido que la matriz de registros MQPMR preceda a la matriz de registros MQOR.

El valor inicial de este campo es 0.

Encoding (MQLONG)

Se trata de la codificación numérica de los datos que siguen a las matrices de registros MQOR y MQPMR; no se aplica a los datos numéricos de la propia estructura MQDH.

En la llamada MQPUT o MQPUT1, la aplicación debe establecer este campo con el valor adecuado para los datos.

El valor inicial de este campo es 0.

CodedCharSetId (MQLONG)

Este es el identificador del conjunto de caracteres de los datos que siguen a las matrices de registros MQOR y MQPMR; no se aplica a los datos de caracteres de la propia estructura MQDH.

En la llamada MQPUT o MQPUT1, la aplicación debe establecer este campo con el valor adecuado para los datos. Puede utilizar el siguiente valor especial:

MQCCSI_HEREDAR

Heredar el identificador del conjunto de caracteres de esta estructura.

Los datos de caracteres en los datos *que siguen* esta estructura están en el mismo conjunto de caracteres que esta estructura.

El gestor de colas cambia este valor en la estructura enviada en el mensaje al identificador real del juego de caracteres de la estructura. Si no se produce ningún error, la llamada MQGET no devuelve el valor MQCCSI_INHERIT.

No puede utilizar MQCCSI_INHERIT si el valor del *PutApplType* campo en MQMD es MQAT_BROKER.

Este valor es compatible con multiplataformas y con IBM MQ clientes conectados a estos sistemas.

El valor inicial de este campo es MQCCSI_UNDEFINED.

Format (MQCHAR8)

Este es el nombre del formato de los datos que siguen a las matrices de registros MQOD y MQPMR (el que ocurra en último lugar).

En la llamada MQPUT o MQPUT1, la aplicación debe establecer este campo con el valor adecuado para los datos. Las reglas para codificar este campo son las mismas que para el *Format* campo en MQMD.

El valor inicial de este campo es MQFMT_NONE.

Flags (MQLONG)

Puede especificar el siguiente indicador:

MQDHF_NUEVO_MSG_IDS

Genera un nuevo identificador de mensaje para cada destino de la lista de distribución. Establezca esto solo cuando no haya registros de mensajes enviados, o cuando los registros estén presentes pero no contengan el *MsgId* campo.

El uso de este indicador aplaza la generación de los identificadores de mensaje hasta el momento en que el mensaje de la lista de distribución se divide finalmente en mensajes individuales. Esto minimiza la cantidad de información de control que debe fluir con el mensaje de la lista de distribución.

Cuando una aplicación envía un mensaje a una lista de distribución, el gestor de colas establece MQDHF_NEW_MSG_IDS en el MQDH que genera cuando ambas afirmaciones siguientes son ciertas:

- La aplicación no proporciona registros de mensajes enviados, o los registros proporcionados no contienen el *MsgId* campo.
- El *MsgId* campo en MQMD es MQMI_NONE, o el *Options* campo en MQPMO incluye MQPMO_NEW_MSG_ID

Si no se necesitan indicadores, especifique lo siguiente:

MQDHF_NINGUNO

No se han especificado banderas. MQDHF_NONE se define para facilitar la documentación del programa. No está previsto que esta constante se utilice con ninguna otra, pero como su valor es cero, dicho uso no puede detectarse.

El valor inicial de este campo es MQDHF_NONE.

PutMsgRecFields (MQLONG)

Puede especificar ninguna o varias de las siguientes opciones:

MQPMRF_MSG_ID

El campo identificador de mensaje está presente.

MQPMRF_CORREL_ID

El campo identificador de correlación está presente.

MQPMRF_ID_GRUPO

El campo identificador de grupo está presente.

MQPMRF_COMENTARIOS

El campo de comentarios está presente.

MQPMRF_CONTABILIDAD_TOKEN

El campo «Accounting-token» está presente.

Si no hay campos MQPMR, especifique lo siguiente:

MQPMRF_NONE

No hay campos de registro de mensajes enviados. MQPMRF_NONE se define para facilitar la documentación del programa. No está previsto que esta constante se utilice con ninguna otra, pero como su valor es cero, dicho uso no puede detectarse.

El valor inicial de este campo es MQPMRF_NONE.

RecsPresent (MQLONG)

Este es el número de destinos. Una lista de distribución debe contener siempre al menos un destino, por lo que *RecsPresent* siempre debe ser mayor que cero.

El valor inicial de este campo es 0.

ObjectRecOffset (MQLONG)

Esto proporciona la compensación en bytes del primer registro de la matriz de registros de objetos MQOR que contienen los nombres de las colas de destino. Hay *RecsPresent* registros en esta matriz. Estos registros (más cualquier byte omitido entre el primer registro de objeto y el campo anterior) se incluyen en la longitud indicada por el *StrucLength* campo.

Una lista de distribución debe contener siempre al menos un destino, por lo que *ObjectRecOffset* siempre debe ser mayor que cero.

El valor inicial de este campo es 0.

PutMsgRecOffset (MQLONG)

Esto proporciona la compensación en bytes del primer registro de la matriz de registros de mensajes MQPMR put que contienen las propiedades del mensaje. Si está presente, hay *RecsPresent* registros en esta matriz. Estos registros (más cualquier byte omitido entre el primer registro del mensaje put y el campo anterior) se incluyen en la longitud indicada por el *StrucLength* campo.

Los registros de mensajes son opcionales; si no se proporcionan registros, *PutMsgRecOffset* es cero y *PutMsgRecFields* tiene el valor MQPMRF_NONE.

El valor inicial de este campo es 0.

MQDLH - Cabecera de mensajes no entregados

La estructura MQDLH describe la información que precede a los datos del mensaje de aplicación de los mensajes en la cola de mensajes no entregados (mensajes no entregados). Un mensaje puede llegar a la cola de mensajes no entregados porque el administrador de la cola o el agente del canal de mensajes lo ha redirigido a la cola, o porque una aplicación lo ha colocado directamente en la cola.

Nombre de formato

MQFMT_DEAD_LETTER_HEADER

Conjunto de caracteres y codificación

Los campos de la estructura MQDLH están en el juego de caracteres y la codificación especificados por los campos *CodedCharSetId* *Encoding* y. Estos se especifican en la estructura de encabezado que precede al MQDLH, o en la estructura MQMD si el MQDLH se encuentra al inicio de los datos del mensaje de aplicación.

El juego de caracteres debe ser uno que tenga caracteres de un solo byte para los caracteres que son válidos en los nombres de las colas.

Si utiliza las IBM MQ clases para Java/ JMSy la página de códigos definida en MQMD no es compatible con la máquina Java virtual, MQDLH se escribe en el juego de caracteres UTF-8.

Utilización

Las aplicaciones que colocan mensajes directamente en la cola de mensajes no entregados deben anteponer a los datos del mensaje una estructura MQDLH e inicializar los campos con los valores adecuados. Sin embargo, el gestor de colas no requiere que exista una estructura MQDLH ni que se hayan especificado valores válidos para los campos.

Si un mensaje es demasiado largo para colocarlo en la cola de mensajes no entregados, la aplicación debe realizar una de las siguientes acciones:

- Truncar los datos del mensaje para que quepan en la cola de mensajes no entregados.
- Graba el mensaje en el almacenamiento auxiliar y coloca un mensaje de informe de excepción en la cola de mensajes no entregados indicando esto.
- Descartar el mensaje y devolver un error a su remitente. Si el mensaje es (o podría ser) un mensaje crítico, haga esto solo si se sabe que el remitente aún tiene una copia del mensaje; por ejemplo, un mensaje recibido por un agente de canal de mensajes desde un canal de comunicación.

Cuál de las acciones anteriores es la adecuada (si es que hay alguna) depende del diseño de la aplicación.

El gestor de colas realiza un procesamiento especial cuando se coloca un mensaje que es un segmento con una estructura MQDLH al principio; consulte la descripción de la estructura MQMDE para obtener más detalles.

Colocar mensajes en la cola de mensajes no entregados

Cuando se coloca un mensaje en la cola de mensajes no entregados, la estructura MQMD utilizada para la llamada MQPUT o MQPUT1 debe ser idéntica a la MQMD asociada al mensaje (normalmente la MQMD devuelta por la llamada MQGET), con la excepción de lo siguiente:

- Establezca los campos *CodedCharSetId* *Encoding* y con el juego de caracteres y la codificación que se utilicen para los campos de la estructura MQDLH.
- Establezca el *Format* campo en MQFMT_DEAD_LETTER_HEADER para indicar que los datos comienzan con una estructura MQDLH.
- Establezca los campos de contexto (*AccountingToken*, *ApplIdentityData*, *ApplOriginData*, *PutApplName*, *PutApplType*, *PutDate*, *PutTime*, *UserIdentifier*) utilizando una opción de contexto adecuada a las circunstancias:
 - Una aplicación que coloque en la cola de mensajes caducados un mensaje que no esté relacionado con ningún mensaje anterior debe utilizar la opción MQPMO_DEFAULT_CONTEXT; esto hace que el gestor de colas establezca todos los campos de contexto del descriptor del mensaje en sus valores predeterminados.
 - Una aplicación de servidor que coloca en la cola de mensajes no entregados un mensaje que acaba de recibir debe utilizar la opción MQPMO_PASS_ALL_CONTEXT para conservar la información contextual original.
 - Una aplicación de servidor que coloca en la cola de mensajes no entregados una *respuesta* a un mensaje que acaba de recibir debe utilizar la opción MQPMO_PASS_IDENTITY_CONTEXT; esto conserva la información de identidad, pero establece la información de origen como la de la aplicación de servidor.
 - Un agente de canal de mensajes que coloca en la cola de mensajes no entregados un mensaje que ha recibido de su canal de comunicación debe utilizar la opción MQPMO_SET_ALL_CONTEXT para conservar la información contextual original.

En la propia estructura MQDLH, configure los campos de la siguiente manera:

- Establezca los campos *CodedCharSetId*, *Encoding* *Format* y con los valores que describen los datos que siguen a la estructura MQDLH, normalmente los valores del descriptor del mensaje original.
- Establezca los campos de contexto *PutApplType*, *PutApplName*, *PutDate* y *PutTime* con valores adecuados para la aplicación que está colocando el mensaje en la cola de mensajes no entregados; estos valores no están relacionados con el mensaje original.

- Configure los demás campos según corresponda.

Asegúrese de que todos los campos tengan valores válidos y que los campos de caracteres estén rellenos con espacios en blanco hasta alcanzar la longitud definida del campo; no termine los datos de caracteres prematuramente utilizando un carácter nulo, ya que el gestor de colas no convierte el carácter nulo y los caracteres posteriores en espacios en blanco en la estructura MQDLH.

Obtener mensajes de la cola de mensajes no entregados

Las aplicaciones que reciben mensajes de la cola de mensajes no entregados deben verificar que los mensajes comiencen con una estructura MQDLH. La aplicación puede determinar si existe una estructura MQDLH examinando el *Format* campo en el descriptor de mensajes MQMD; si el campo tiene el valor MQFMT_DEAD_LETTER_HEADER, los datos del mensaje comienzan con una estructura MQDLH. Tenga en cuenta también que los mensajes que las aplicaciones obtienen de la cola de mensajes no entregados pueden truncarse si originalmente eran demasiado largos para la cola.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 495. Campos en MQDLH		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>StrucId</u> (identificador de estructura)	MQDLH_STRUC_ID	'DLH'
<u>Versión</u> (número de versión de la estructura)	MQDLH_VERSION_1	1
<u>Motivo</u> (mensaje de motivo llegado a la cola de mensajes no entregados)	MQRC_NONE	0
<u>DestQName</u> (nombre de la cola de destino original)	Ninguna	Cadena nula o espacios en blanco
<u>DestQMGrName</u> (nombre del gestor de colas de destino original)	Ninguna	Cadena nula o espacios en blanco
<u>Codificación</u> (codificación numérica de los datos que siguen a MQDLH)	Ninguna	0
<u>CodedCharSetId</u> (identificador del conjunto de caracteres de los datos que siguen a MQDLH)	MQCCSI_INDETERMINADO	0
<u>Formato</u> (nombre del formato de los datos que siguen a MQDLH)	MQFMT_NONE	Espacios en blanco
<u>PutApplType</u> (tipo de aplicación que coloca mensajes en la cola de mensajes no entregados)	Ninguna	0
<u>PutApplName</u> (nombre de la aplicación que colocó el mensaje en la cola de mensajes no entregados)	Ninguna	Cadena nula o espacios en blanco
<u>PutDate</u> (fecha en la que el mensaje se colocó en la cola de mensajes no entregados)	Ninguna	Cadena nula o espacios en blanco
<u>PutTime</u> (hora en que el mensaje se colocó en la cola de mensajes no entregados)	Ninguna	Cadena nula o espacios en blanco

Tabla 495. Campos en MQDLH (continuación)

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
Notas: 1. El símbolo ~ representa un único carácter en blanco. 2. El valor «Cadena nula o espacios en blanco» denota la cadena nula en C y los caracteres en blanco en otros lenguajes de programación. 3. En el lenguaje de programación C, la variable macroMQDLH_DEFAULT contiene los valores que se enumeran en la tabla. Úselo de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQDLH MyDLH = {MQDLH_DEFAULT};</pre>		

Declaraciones lingüísticas

Declaración C para MQDLH

```
typedef struct tagMQDLH MQDLH;
struct tagMQDLH {
    MQCHAR4    StrucId;        /* Structure identifier */
    MQLONG     Version;        /* Structure version number */
    MQLONG     Reason;         /* Reason message arrived on dead-letter
                               (undelivered-message) queue */
    MQCHAR48    DestQName;     /* Name of original destination queue */
    MQCHAR48    DestQMGrName;  /* Name of original destination queue
                               manager */
    MQLONG     Encoding;       /* Numeric encoding of data that follows
                               MQDLH */
    MQLONG     CodedCharSetId; /* Character set identifier of data that
                               follows MQDLH */
    MQCHAR8     Format;         /* Format name of data that follows
                               MQDLH */
    MQLONG     PutApplType;     /* Type of application that put message on
                               dead-letter (undelivered-message)
                               queue */
    MQCHAR28    PutApplName;    /* Name of application that put message on
                               dead-letter (undelivered-message)
                               queue */
    MQCHAR8     PutDate;        /* Date when message was put on dead-letter
                               (undelivered-message) queue */
    MQCHAR8     PutTime;        /* Time when message was put on the
                               dead-letter (undelivered-message)
                               queue */
};
```

COBOL declaración para MQDLH

```
**  MQDLH structure
10  MQDLH.
**    Structure identifier
15  MQDLH-STRUCID      PIC X(4).
**    Structure version number
15  MQDLH-VERSION      PIC S9(9) BINARY.
**    Reason message arrived on dead-letter (undelivered-message) queue
15  MQDLH-REASON      PIC S9(9) BINARY.
**    Name of original destination queue
15  MQDLH-DESTQNAME    PIC X(48).
**    Name of original destination queue manager
15  MQDLH-DESTMGRNAME  PIC X(48).
**    Numeric encoding of data that follows MQDLH
15  MQDLH-ENCODING     PIC S9(9) BINARY.
**    Character set identifier of data that follows MQDLH
15  MQDLH-CODEDCHARSETID PIC S9(9) BINARY.
**    Format name of data that follows MQDLH
15  MQDLH-FORMAT       PIC X(8).
**    Type of application that put message on dead-letter
**    (undelivered-message) queue
15  MQDLH-PUTAPPLTYPE  PIC S9(9) BINARY.
```

```

**      Name of application that put message on dead-letter
**      (undelivered-message) queue
15 MQDLH-PUTAPPLNAME      PIC X(28).
**      Date when message was put on dead-letter (undelivered-message)
**      queue
15 MQDLH-PUTDATE          PIC X(8).
**      Time when message was put on the dead-letter (undelivered-message)
**      queue
15 MQDLH-PUTTIME          PIC X(8).

```

PL/I declaración para MQDLH

```

dcl
  1 MQDLH based,
    3 StrucId      char(4),      /* Structure identifier */
    3 Version      fixed bin(31), /* Structure version number */
    3 Reason       fixed bin(31), /* Reason message arrived on
                                   dead-letter (undelivered-message)
                                   queue */
    3 DestQName    char(48),     /* Name of original destination
                                   queue */
    3 DestQMgrName char(48),     /* Name of original destination queue
                                   manager */
    3 Encoding     fixed bin(31), /* Numeric encoding of data that
                                   follows MQDLH */
    3 CodedCharSetId fixed bin(31), /* Character set identifier of data
                                   that follows MQDLH */
    3 Format        char(8),      /* Format name of data that follows
                                   MQDLH */
    3 PutApplType   fixed bin(31), /* Type of application that put
                                   message on dead-letter
                                   (undelivered-message) queue */
    3 PutApplName   char(28),     /* Name of application that put
                                   message on dead-letter
                                   (undelivered-message) queue */
    3 PutDate       char(8),      /* Date when message was put on
                                   dead-letter (undelivered-message)
                                   queue */
    3 PutTime       char(8);      /* Time when message was put on the
                                   dead-letter (undelivered-message)
                                   queue */

```

High Level Assembler declaración para MQDLH

MQDLH	DSECT	
MQDLH_STRUCID	DS	CL4 Structure identifier
MQDLH_VERSION	DS	F Structure version number
MQDLH_REASON	DS	F Reason message arrived on dead-letter
*		(undelivered-message) queue
MQDLH_DESTQNAME	DS	CL48 Name of original destination queue
MQDLH_DESTQMGRNAME	DS	CL48 Name of original destination queue
*		manager
MQDLH_ENCODING	DS	F Numeric encoding of data that follows
*		MQDLH
MQDLH_CODEDCHARSETID	DS	F Character set identifier of data that
*		follows MQDLH
MQDLH_FORMAT	DS	CL8 Format name of data that follows MQDLH
MQDLH_PUTAPPLTYPE	DS	F Type of application that put message on
*		dead-letter (undelivered-message) queue
MQDLH_PUTAPPLNAME	DS	CL28 Name of application that put message on
*		dead-letter (undelivered-message) queue
MQDLH_PUTDATE	DS	CL8 Date when message was put on
*		dead-letter (undelivered-message) queue
MQDLH_PUTTIME	DS	CL8 Time when message was put on the
*		dead-letter (undelivered-message) queue
*		
MQDLH_LENGTH	EQU	*-MQDLH
	ORG	MQDLH
MQDLH_AREA	DS	CL(MQDLH_LENGTH)

Declaración de Visual Basic para MQDLH

Type MQDLH		
StrucId	As String*4	'Structure identifier'
Version	As Long	'Structure version number'
Reason	As Long	'Reason message arrived on dead-letter'
		'(undelivered-message) queue'
DestQName	As String*48	'Name of original destination queue'

```

DestQMgrName  As String*48 'Name of original destination queue'
                        'manager'
Encoding       As Long      'Numeric encoding of data that follows'
                        'MQDLH'
CodedCharSetId As Long      'Character set identifier of data that'
                        'follows MQDLH'
Format         As String*8  'Format name of data that follows MQDLH'
PutApplType    As Long      'Type of application that put message on'
                        'dead-letter (undelivered-message) queue'
PutApplName    As String*28 'Name of application that put message on'
                        'dead-letter (undelivered-message) queue'
PutDate        As String*8  'Date when message was put on dead-letter'
                        '(undelivered-message) queue'
PutTime        As String*8  'Time when message was put on the'
                        'dead-letter (undelivered-message) queue'
End Type

```

Detalles de campo para MQDLH

La estructura del encabezado de correo no entregado MQDLH describe la información que precede a los datos del mensaje de aplicación de los mensajes en la cola de correo no entregado (mensajes no entregados). Un mensaje puede llegar a la cola de mensajes no entregados porque el administrador de la cola o el agente del canal de mensajes lo ha redirigido a la cola, o porque una aplicación lo ha colocado directamente en la cola.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 496. Campos en MQDLH		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>StrucId</u> (identificador de estructura)	MQDLH_STRUC_ID	'DLH~'
<u>Versión</u> (número de versión de la estructura)	MQDLH_VERSION_1	1
<u>Motivo</u> (mensaje de motivo llegado a la cola de mensajes no entregados)	MQRC_NONE	0
<u>DestQName</u> (nombre de la cola de destino original)	Ninguna	Cadena nula o espacios en blanco
<u>DestQMgrName</u> (nombre del gestor de colas de destino original)	Ninguna	Cadena nula o espacios en blanco
<u>Codificación</u> (codificación numérica de los datos que siguen a MQDLH)	Ninguna	0
<u>CodedCharSetId</u> (identificador del conjunto de caracteres de los datos que siguen a MQDLH)	MQCCSI_INDETERMINADO	0
<u>Formato</u> (nombre del formato de los datos que siguen a MQDLH)	MQFMT_NONE	Espacios en blanco
<u>PutApplType</u> (tipo de aplicación que coloca mensajes en la cola de mensajes no entregados)	Ninguna	0
<u>PutApplName</u> (nombre de la aplicación que colocó el mensaje en la cola de mensajes no entregados)	Ninguna	Cadena nula o espacios en blanco
<u>PutDate</u> (fecha en la que el mensaje se colocó en la cola de mensajes no entregados)	Ninguna	Cadena nula o espacios en blanco
<u>PutTime</u> (hora en que el mensaje se colocó en la cola de mensajes no entregados)	Ninguna	Cadena nula o espacios en blanco

Tabla 496. Campos en MQDLH (continuación)

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
Notas: 1. El símbolo ~ representa un único carácter en blanco. 2. El valor «Cadena nula o espacios en blanco» denota la cadena nula en C y los caracteres en blanco en otros lenguajes de programación. 3. En el lenguaje de programación C, la variable macro MQDLH_DEFAULT contiene los valores que se enumeran en la tabla. Úselo de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQDLH MyDLH = {MQDLH_DEFAULT};</pre>		

StrucId (MQCHAR4)

Este es el identificador de estructura de la estructura del encabezado de carta muerta. Siempre es un campo de entrada. Su valor es MQDLH_STRUC_ID.

El valor debe ser:

MQDLH_STRUC_ID

Identificador de la estructura del encabezado de carta muerta.

Para el lenguaje de programación C, también se define la constante MQDLH_STRUC_ID_ARRAY. Tiene el mismo valor que MQDLH_STRUC_ID, pero es una matriz de caracteres en lugar de una cadena.

Versión (MQLONG)

La versión es el número de versión de la estructura.

El valor debe ser:

MQDLH_VERSION_1

Número de versión para la estructura del encabezado de correo no entregado.

La siguiente constante especifica el número de versión de la versión actual:

MQDLH_VERSIÓN_ACTUAL

Versión actual de la estructura del encabezado de correo no entregado.

El valor inicial de este campo es MQDLH_VERSION_1.

Razón (MQLONG)

El campo Motivo identifica el motivo por el que el mensaje se colocó en la cola de mensajes no entregados en lugar de en la cola de destino original.

Esto identifica el motivo por el cual el mensaje se colocó en la cola de mensajes no entregados en lugar de en la cola de destino original. Debe ser uno de los valores MQFB_* o MQRC_* (por ejemplo, MQRC_Q_FULL). Consulte la descripción del *Feedback* campo en “MQMD - Descriptor de mensaje” en la [página 389](#) para obtener más información sobre los valores MQFB_* comunes que pueden aparecer.

Si el valor está comprendido entre MQFB_IMS_FIRST y MQFB_IMS_LAST, el código de error IMS real puede determinarse restando MQFB_IMS_ERROR del valor del *Reason* campo.

Algunos valores MQFB_* solo aparecen en este campo. Se refieren a mensajes de repositorio, mensajes de activación o mensajes de cola de transmisión que se han transferido a la cola de mensajes no entregados. Son las siguientes:

MQFB_APPL_NO_SE_PUEDE_INICIAR (X'00000109')

Una aplicación que procesa un mensaje de activación no puede iniciar la aplicación indicada en el *ApplId* campo del mensaje de activación (véase [“MQTM - Mensaje de activación”](#) en la página 595).

En z/OS, la transacción CICS CKTI es un ejemplo de una aplicación que procesa mensajes de activación.

MQFB_APPL_TYPE_ERROR (X'0000010B')

Una aplicación que procesa un mensaje de activación no puede iniciar la aplicación porque el *ApplType* campo del mensaje de activación no es válido (véase [“MQTM - Mensaje de activación”](#) en la página 595).

En z/OS, la transacción CICS CKTI es un ejemplo de una aplicación que procesa mensajes de activación.

MQFB_BIND_OPEN_CLUSRCVR_DEL (X'00000119')

El mensaje estaba en el canal de recepción remota (SYSTEM.CLUSTER.TRANSMIT.QUEUE) destinado a una cola de clúster que se abrió con la opción MQOO_BIND_ON_OPEN, pero el canal receptor del clúster remoto que se iba a utilizar para transmitir el mensaje a la cola de destino se eliminó antes de que se pudiera enviar el mensaje. Debido a que se especificó MQOO_BIND_ON_OPEN, solo se puede utilizar el canal seleccionado al abrir la cola para transmitir el mensaje. Como este canal ya no está disponible, el mensaje se coloca en la cola de mensajes no entregados.

MQFB_NOT_A_REPOSITORY_MSG (X'00000118')

El mensaje no es un mensaje de repositorio.

MQFB_DETENIDO_POR_SALIDA_DE_CHAD (X'00000115')

El mensaje fue detenido por la salida de la definición automática del canal.

MQFB_DETENIDO_POR_SALIDA_DE_MENSAJE (X'0000010D')

El mensaje fue detenido por la salida del mensaje del canal.

MQFB_TM_ERROR (X'0000010A')

El *Format* campo en MQMD especifica MQFMT_TRIGGER, pero el mensaje no comienza con una estructura MQTM válida. Por ejemplo, el *StrucId* mnemotécnico llamativo podría no ser válido, el *Version* podría no ser reconocido, o la longitud del mensaje desencadenante podría ser insuficiente para contener la estructura MQTM.

En z/OS, la transacción CICS CKTI es un ejemplo de una aplicación que procesa mensajes de activación y puede generar este código de respuesta.

MQFB_XMIT_Q_MSG_ERROR (X'0000010F')

Un agente del canal de mensajes ha detectado que un mensaje de la cola de transmisión no tiene el formato correcto. El agente del canal de mensajes coloca el mensaje en la cola de mensajes no entregados utilizando este código de respuesta.

Una causa común es que un mensaje se ha colocado directamente en la cola de transmisión, por lo que el mensaje no tiene el encabezado XQH esperado. Los mensajes deben colocarse en una cola de transmisión a través de una cola remota, a menos que la aplicación cree el encabezado MQXQH.

El valor inicial de este campo es MQRC_NONE.

DestQName (MQCHAR48)

DestQName es el nombre de la cola de mensajes que era el destino original del mensaje.

La longitud de este campo viene determinada por MQ_Q_NAME_LENGTH. El valor inicial de este campo es la cadena nula en C y 48 caracteres en blanco en otros lenguajes de programación.

DestQMgrName (MQCHAR48)

DestQMgrName es el nombre del gestor de colas que era el destino original del mensaje.

La longitud de este campo viene determinada por MQ_Q_MGR_NAME_LENGTH. El valor inicial de este campo es la cadena nula en C y 48 caracteres en blanco en otros lenguajes de programación.

Encoding (MQLONG)

La codificación es la codificación numérica de los datos que siguen la estructura MQDLH (normalmente los datos del mensaje original); no se aplica a los datos numéricos de la propia estructura MQDLH.

En la llamada MQPUT o MQPUT1, la aplicación debe establecer este campo con el valor adecuado para los datos.

El valor inicial de este campo es 0.

CodedCharSetId (MQLONG)

CodedCharSetId Es el identificador del conjunto de caracteres de los datos que fluyen a través de la estructura MQDLH (normalmente los datos del mensaje original); no se aplica a los datos de caracteres de la propia estructura MQDLH.

En la llamada MQPUT o MQPUT1, la aplicación debe establecer este campo con el valor adecuado para los datos. Se puede utilizar el siguiente valor especial:

MQCCSI_HEREDAR

Los datos de caracteres en los datos que siguen esta estructura están en el mismo conjunto de caracteres que esta estructura.

El gestor de colas cambia este valor en la estructura enviada en el mensaje al identificador real del juego de caracteres de la estructura. Si no se produce ningún error, la llamada MQGET no devuelve el valor MQCCSI_INHERIT.

No puede utilizar MQCCSI_INHERIT si el valor del *PutApplType* campo en MQMD es MQAT_BROKER.

Este valor es compatible con los siguientes entornos:

-  AIX
-  IBM i
-  Linux
-  Windows

y para IBM MQ los clientes conectados a estos sistemas.

El valor inicial de este campo es MQCCSI_UNDEFINED.

Format (MQCHAR8)

El formato es el nombre del formato de los datos que siguen la estructura MQDLH (normalmente los datos del mensaje original).

En la llamada MQPUT o MQPUT1, la aplicación debe establecer este campo con el valor adecuado para los datos. Las reglas para codificar este campo son las mismas que las reglas para codificar el *Format* campo en MQMD.

La longitud de este campo viene dada por MQ_FORMAT_LENGTH. El valor inicial de este campo es MQFMT_NONE.

PutApplType (MQLONG)

PutApplType es el tipo de aplicación que coloca el mensaje en la cola de mensajes no entregados (mensajes no entregados).

Este campo tiene el mismo significado que el *PutApplType* campo del descriptor de mensajes MQMD (véase [“MQMD - Descriptor de mensaje” en la página 389](#) para más detalles).

Si el gestor de colas redirige el mensaje a la cola de mensajes no entregados, *PutApplType* tiene el valor MQAT_QMGR.

El valor inicial de este campo es 0.

PutApplName (MQCHAR28)

PutApplName es el nombre de la aplicación que colocó el mensaje en la cola de mensajes no entregados (mensajes no entregados).

El formato del nombre depende del campo *PutApplType*. El formato puede variar de una versión a otra. Consulte la descripción del campo *PutApplName* en [“MQMD - Descriptor de mensaje” en la página 389](#).

Si el gestor de colas redirige el mensaje a la cola de mensajes no entregados, *PutApplName* contiene los primeros 28 caracteres del nombre del gestor de colas, rellenos con espacios en blanco si es necesario.

La longitud de este campo viene dada por MQ_PUT_APPL_NAME_LENGTH. El valor inicial de este campo es la cadena nula en C y 28 caracteres en blanco en otros lenguajes de programación.

PutDate (MQCHAR8)

PutDate es la fecha en la que el mensaje se colocó en la cola de mensajes no entregados (mensajes no entregados).

El formato utilizado para la fecha cuando este campo es generado por el gestor de colas es:

- AAAAMMDD

donde los caracteres representan:

AAAA

año (cuatro dígitos numéricos)

MM

mes del año (del 01 al 12)

DD

día del mes (del 01 al 31)

La hora del meridiano de Greenwich (GMT) se utiliza para los campos *PutDate* *PutTime* y, siempre que el reloj del sistema esté ajustado con precisión a la GMT.

La longitud de este campo viene dada por MQ_PUT_DATE_LENGTH. El valor inicial de este campo es la cadena nula en C y ocho caracteres en blanco en otros lenguajes de programación.

PutTime (MQCHAR8)

PutTime es el momento en que el mensaje se colocó en la cola de mensajes no entregados (mensajes no entregados).

El formato utilizado para la hora en que este campo es generado por el gestor de colas es:

- HHMMSSSTH

donde los caracteres representan:

HH

horas (de 00 a 23)

MM

minutos (de 00 a 59)

SS

segundos (de 00 a 59; véase la nota)

T

décimas de segundo (0 a 9)

H

centésimas de segundo (0 a 9)

Nota: Si el reloj del sistema está sincronizado con un estándar de tiempo muy preciso, en raras ocasiones es posible que se devuelva 60 o 61 para los segundos en *PutTime*. Esto ocurre cuando se insertan segundos intercalares en el estándar horario global.

La hora del meridiano de Greenwich (GMT) se utiliza para los campos *PutDate* *PutTime* y, siempre que el reloj del sistema esté ajustado con precisión a la GMT.

La longitud de este campo viene dada por MQ_PUT_TIME_LENGTH. El valor inicial de este campo es la cadena nula en C y ocho caracteres en blanco en otros lenguajes de programación.

MQDMHO: opciones para eliminar el identificador de mensaje

La **MQDMHO** estructura permite a las aplicaciones especificar opciones que controlan cómo se eliminan los identificadores de mensajes. La estructura es un parámetro de entrada en la **MQDLTMH** llamada.

Conjunto de caracteres y codificación

Los datos en **MQDMHO** deben estar en el juego de caracteres de la aplicación y la codificación de la aplicación (**MQENC_NATIVE**).

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 497. Campos en MQDMHO		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>StrucId</u> (identificador de estructura)	MQDMHO_STRUC_ID	'DMHO'
<u>Versión</u> (número de versión de la estructura)	MQDMHO_VERSION_1	1
<u>Opciones</u> (opciones)	MQDMHO_NINGUNO	0
Notas: 1. En el lenguaje de programación C, la variable macroMQDMHO_DEFAULT contiene los valores que se enumeran en la tabla. Se puede utilizar de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQDMHO MyDMHO = {MQDMHO_DEFAULT};</pre>		

Declaraciones lingüísticas

Declaración C para MQDMHO

```
typedef struct tagMQDMHO;  
struct tagMQDMHO {  
    MQCHAR4    StrucId;           /* Structure identifier */  
    MQLONG     Version;          /* Structure version number */  
    MQLONG     Options;          /* Options that control the action of MQDLTMH */  
};
```

COBOL Declaración para MQDMHO

```
**    MQDMHO structure  
10 MQDMHO.  
**    Structure identifier  
15 MQDMHO-STRUCID      PIC X(4).  
**    Structure version number  
15 MQDMHO-VERSION     PIC S9(9) BINARY.
```

```

**      Options that control the action of MQDLTMH
15 MQDMHO-OPTIONS      PIC S9(9) BINARY.

```

PL/I Declaración para MQDMHO

```

dcl
  1 MQDMHO based,
    3 StrucId      char(4),          /* Structure identifier */
    3 Version      fixed bin(31), /* Structure version number */
    3 Options      fixed bin(31), /* Options that control the action of MQDLTMH */

```

High Level Assembler Declaración para MQDMHO

```

MQDMHO          DSECT
MQDMHO_STRUCID  DS    CL4    Structure identifier
MQDMHO_VERSION  DS    F      Structure version number
MQDMHO_OPTIONS  DS    F      Options that control the action of
*                               MQDLTMH
MQDMHO_LENGTH   EQU    *-MQDMHO
MQDMHO_AREA     DS      CL(MQDMHO_LENGTH)

```

Detalles de campo para MQDMHO

La estructura Opciones de **MQDMHO** eliminación de identificadores de mensajes permite a las aplicaciones especificar opciones que controlan cómo se eliminan los identificadores de mensajes. La estructura es un parámetro de entrada en la **MQDLTMH** llamada.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 498. Campos en MQDMHO		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>StrucId</u> (identificador de estructura)	MQDMHO_STRUC_ID	'DMHO'
<u>Versión</u> (número de versión de la estructura)	MQDMHO_VERSION_1	1
<u>Opciones</u> (opciones)	MQDMHO_NINGUNO	0
Notas: 1. En el lenguaje de programación C, la variable macro MQDMHO_DEFAULT contiene los valores que se enumeran en la tabla. Se puede utilizar de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQDMHO MyDMHO = {MQDMHO_DEFAULT};</pre>		

StrucId (MQCHAR4)

Este es el identificador de estructura de la estructura de opciones del identificador de mensajes eliminados. Siempre es un campo de entrada. Su valor es MQDMHO_STRUC_ID.

El valor debe ser:

MQDMHO_STRUC_ID

Identificador para la estructura de opciones del identificador de mensajes eliminados.

Para el lenguaje de programación C, también se **MQDMHO_STRUC_ID_ARRAY** define la constante. Esto tiene el mismo valor que **MQDMHO_STRUC_ID**, pero es una matriz de caracteres en lugar de una cadena.

Versión (MQLONG)

Este es el número de versión de la estructura. El valor debe ser:

MQDMHO_VERSION_1

Version-1 Eliminar la estructura de opciones del identificador de mensaje.

La siguiente constante especifica el número de versión de la versión actual:

MQDMHO_VERSIÓN_ACTUAL

Versión actual de la estructura de opciones de eliminación de mensajes.

Este campo siempre es un campo de entrada. El valor inicial de este campo es **MQDMHO_VERSION_1**.

Opciones (MQLONG)

El valor debe ser:

MQDMHO_NINGUNO

No se ha especificado ninguna opción.

Este campo siempre es un campo de entrada. El valor inicial de este campo es **MQDMHO_NONE**.

MQDMPO - Eliminar opciones de propiedades del mensaje

La estructura MQDMPO permite a las aplicaciones especificar opciones que controlan cómo se eliminan las propiedades de los mensajes. La estructura es un parámetro de entrada en la llamada MQDLTMP.

Conjunto de caracteres y codificación

Los datos en MQDMPO deben estar en el juego de caracteres de la aplicación y en la codificación de la aplicación (MQENC_NATIVE).

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 499. Campos en MQDPMO		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>StrucId</u> (identificador de estructura)	MQDMPO_STRUC_ID	' DPMO '
<u>Versión</u> (número de versión de la estructura)	MQDMPO_VERSION_1	1
Opciones (opciones que controlan la acción de MQDMPO)	Opciones que controlan la acción de MQDLTMP	MQDMPO_NINGUNO
Notas: 1. En el lenguaje de programación C, la variable macroMQDMPO_DEFAULT contiene los valores que se enumeran en la tabla. Úselo de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQDMPO MyDPMO = {MQDMPO_DEFAULT};</pre>		

Declaraciones lingüísticas

Declaración C para MQDMPO

```
typedef struct tagMQDMPO MQDMPO;  
struct tagMQDMPO {  
    MQCHAR4 StrucId;          /* Structure identifier */
```

```

    MQLONG    Version;          /* Structure version number */
    MQLONG    Options;          /* Options that control the action of
                                MQDLTMP */
};

```

COBOL declaración para MQDMPO

```

**  MQDMPO structure
10  MQDMPO.
**  Structure identifier
15  MQDMPO-STRUCID          PIC X(4).
**  Structure version number
15  MQDMPO-VERSION          PIC S9(9) BINARY.
**  Options that control the action of MQDLTMP
15  MQDMPO-OPTIONS          PIC S9(9) BINARY.

```

PL/I declaración para MQDMPO

```

Dcl
  1 MQDMPO based,
    3 StrucId      char(4),          /* Structure identifier */
    3 Version      fixed bin(31),    /* Structure version number */
    3 Options      fixed bin(31),    /* Options that control the action
                                     of MQDLTMP */

```

High Level Assembler declaración para MQDMPO

```

MQDMPO          DSECT
MQDMPO_STRUCID  DS    CL4  Structure identifier
MQDMPO_VERSION  DS    F    Structure version number
MQDMPO_OPTIONS  DS    F    Options that control the
*                  action of MQDLTMP
MQDMPO_LENGTH   EQU    *-MQDMPO
MQDMPO_AREA     DS    CL(MQDMPO_LENGTH)

```

Detalles de campo para MQDMPO

La estructura de opciones de propiedades de mensajes MQDMPO Delete permite a las aplicaciones especificar opciones que controlan cómo se eliminan las propiedades de los mensajes. La estructura es un parámetro de entrada en la llamada MQDLTMP.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 500. Campos en MQDPMO		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
StrucId (identificador de estructura)	MQDMPO_STRUC_ID	'DPMO'
Versión (número de versión de la estructura)	MQDMPO_VERSION_1	1
Opciones (opciones que controlan la acción de MQDMPO)	Opciones que controlan la acción de MQDLTMP	MQDMPO_NINGUNO
Notas: En el lenguaje de programación C, la variable macro MQDMPO_DEFAULT contiene los valores que se enumeran en la tabla. Úselo de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQDMPO MyDPMO = {MQDMPO_DEFAULT};</pre>		

StrucId (MQCHAR4)

Este es el identificador de estructura de la estructura de opciones de la propiedad del mensaje de eliminación. Siempre es un campo de entrada. Su valor es MQDMPO_STRUC_ID.

El valor debe ser:

MQDMPO_STRUC_ID

Identificador de la estructura de opciones de la propiedad del mensaje de eliminación.

Para el lenguaje de programación C, también se define la constante MQDMPO_STRUC_ID_ARRAY. Tiene el mismo valor que MQDMPO_STRUC_ID, pero es una matriz de caracteres en lugar de una cadena.

Versión (MQLONG)

Eliminar estructura de opciones de propiedad del mensaje: campo Versión.

Este es el número de versión de la estructura. El valor debe ser:

MQDMPO_VERSION_1

Número de versión para la estructura de opciones de la propiedad de eliminación de mensajes.

La siguiente constante especifica el número de versión de la versión actual:

MQDMPO_VERSIÓN_ACTUAL

Estructura actual de las opciones de eliminación de propiedades de mensajes.

Este campo siempre es un campo de entrada. El valor inicial de este campo es MQDMPO_VERSION_1.

Opciones (MQLONG)

Eliminar estructura de opciones de propiedad del mensaje - Campo Opciones.

Opciones de ubicación: Las siguientes opciones se refieren a la ubicación relativa de la propiedad en comparación con el cursor de la propiedad.

MQDMPO_DEL_FIRST

Elimina la primera propiedad que coincide con el nombre especificado.

MQDMPO_DEL_PROP_BAJO_EL_CURSOR

Elimina la propiedad a la que apunta el cursor de propiedad; es decir, la última propiedad consultada mediante la opción MQIMPO_INQ_FIRST o MQIMPO_INQ_NEXT.

El cursor de propiedad se restablece cuando se reutiliza el identificador de mensaje. También se restablece cuando se especifica el identificador de mensaje en el *MsgHandle* campo de la estructura MQGMO en una llamada MQGET, o en la estructura MQPMO en una llamada MQPUT.

Si se utiliza esta opción cuando aún no se ha establecido el cursor de propiedad, la llamada falla con el código de finalización MQCC_FAILED y el motivo MQRC_PROPERTY_NOT_AVAILABLE. Si la propiedad a la que apunta el cursor de propiedad ya se ha eliminado, la llamada también falla con el código de finalización MQCC_FAILED y el motivo MQRC_PROPERTY_NOT_AVAILABLE.

Si ninguna de estas opciones es necesaria, se puede utilizar la siguiente opción:

MQDMPO_NINGUNO

No se ha especificado ninguna opción.

Este campo es siempre un campo de entrada. El valor inicial de este campo es MQDMPO_DEL_FIRST.

MQEPH - Cabecera PCF incrustada

La estructura MQEPH describe los datos adicionales que están presentes en un mensaje cuando dicho mensaje es un mensaje de formato de comando programable (PCF). El *PCFHeader* campo define los parámetros PCF que siguen esta estructura y esto le permite seguir los datos del mensaje PCF con otros encabezados.

Nombre de formato

MQFMT_EMBEDDED_PCF

Conjunto de caracteres y codificación

Los datos en MQEPH deben estar en el juego de caracteres proporcionado por el atributo del **CodedCharSetId** gestor de colas y la codificación del gestor de colas local proporcionada por MQENC_NATIVE.

Establezca el juego de caracteres y la codificación del MQEPH en los campos *CodedCharSetId* *Encoding* y del MQMD (si la estructura MQEPH se encuentra al principio de los datos del mensaje), o en la estructura de encabezado que precede a la estructura MQEPH (en todos los demás casos).

Utilización

No se pueden utilizar estructuras MQEPH para enviar comandos al servidor de comandos ni a ningún otro servidor que acepte PCF del gestor de colas.

Del mismo modo, el servidor de comandos o cualquier otro servidor gestor de colas que acepte PCF no generan respuestas ni eventos que contengan estructuras MQEPH.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 501. Campos en MQEPH para MQEPH		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>StrucId</u> (identificador de estructura)	MQEPH_STRUC_ID	'EPH~'
<u>Versión</u> (número de versión de la estructura)	MQEPH_VERSION_1	1
<u>StrucLength</u> (longitud de la estructura MQEPH más las estructuras MQCFH y de parámetros que la siguen)	MQEPH_ESTRUCTURA_LONGITUD_FIJA	68
<u>Codificación</u> (codificación numérica de los datos que siguen la estructura del último parámetro PCF)	Ninguna	0
<u>CodedCharSetId</u> (identificador del conjunto de caracteres de los datos que siguen a la última estructura de parámetros PCF)	MQCCSI_INDETERMINADO	0
<u>Formato</u> (nombre del formato de los datos que siguen a la última estructura de parámetros PCF)	MQFMT_NONE	Espacios en blanco
<u>Banderas</u> (banderas)	MQEPH_NONE	0
<u>PCFHeader</u> (encabezado de formato de comando programable (PCF))	Nombres y valores tal y como se definen en Tabla 503 en la página 331	0

Tabla 501. Campos en MQEPH para MQEPH (continuación)

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
Notas: 1. El símbolo ~ representa un único carácter en blanco. 2. En el lenguaje de programación C, la variable macroMQEPH_DEFAULT contiene los valores que se enumeran en la tabla. Úselo de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQEPH MyEPH = {MQEPH_DEFAULT};</pre>		

Declaraciones lingüísticas

Declaración C para MQEPH

```
typedef struct tagMQEPH MQEPH;
struct tagMQDH {
    MQCHAR4  StrucId;           /* Structure identifier */
    MQLONG   Version;          /* Structure version number */
    MQLONG   StrucLength;      /* Total length of MQEPH including the MQCFH
                               and parameter structures that follow it */
    MQLONG   Encoding;         /* Numeric encoding of data that follows last
                               PCF parameter structure */
    MQLONG   CodedCharSetId;    /* Character set identifier of data that
                               follows last PCF parameter structure */
    MQCHAR8   Format;          /* Format name of data that follows last PCF
                               parameter structure */
    MQLONG   Flags;            /* Flags */
    MQCFH     PCFHeader;       /* Programmable command format header */
};
```

COBOL declaración para MQEPH

```
** MQEPH structure
10 MQEPH.
** Structure identifier
15 MQEPH-STRUCID PIC X(4).
** Structure version number
15 MQEPH-VERSION PIC S9(9) BINARY.
** Total length of MQEPH structure including the MQCFH
** and parameter structures that follow it
15 MQEPH-STRUCLength PIC S9(9) BINARY.
** Numeric encoding of data that follows last
** PCF structure
15 MQEPH-ENCODING PIC S9(9) BINARY.
** Character set identifier of data that
** follows last PCF parameter structure
15 MQEPH-CODEDCHARSETID PIC S9(9) BINARY.
** Format name of data that follows last PCF
** parameter structure
15 MQEPH-FORMAT PIC X(8).
** Flags
15 MQEPH-FLAGS PIC S9(9) BINARY.
** Programmable command format header
15 MQEPH-PCFHEADER.
** Structure type
20 MQEPH-PCFHEADER-TYPE PIC S9(9) BINARY.
** Structure length
20 MQEPH-PCFHEADER-STRUCLength PIC S9(9) BINARY.
** Structure version number
20 MQEPH-PCFHEADER-VERSION PIC S9(9) BINARY.
** Command identifier
20 MQEPH-PCFHEADER-COMMAND PIC S9(9) BINARY.
** Message sequence number
20 MQEPH-PCFHEADER-MSGSEQNUMBER PIC S9(9) BINARY.
** Control options
20 MQEPH-PCFHEADER-CONTROL PIC S9(9) BINARY.
** Completion code
20 MQEPH-PCFHEADER-COMPCODE PIC S9(9) BINARY.
** Reason code qualifying completion code
```

```

20 MQEPH-PCFHEADER-REASON      PIC S9(9) BINARY.
**      Count of parameter structures
20 MQEPH-PCFHEADER-PARAMETERCOUNT PIC S9(9) BINARY.

```

PL/I declaración para MQEPH

```

dcl
  1 MQEPH based,
    3 StrucId      char(4),      /* Structure identifier */
    3 Version      fixed bin(31), /* Structure version number */
    3 StrucLength  fixed bin(31), /* Total Length of MQEPH including the
                                   MQCFH and parameter structures that
                                   follow it
    3 Encoding      fixed bin(31), /* Numeric encoding of data that follows
                                   last PCF parameter structure
    3 CodedCharSetId fixed bin(31), /* Character set identifier of data that
                                   follows last PCF parameter structure
    3 Format         char(8),      /* Format name of data that follows last
                                   PCF parameter structure */
    3 Flags         fixed bin(31), /* Flags */
    3 PCFHeader,    /* Programmable command format header
    5 Type          fixed bin(31), /* Structure type */
    5 StrucLength  fixed bin(31), /* Structure length */
    5 Version      fixed bin(31), /* Structure version number */
    5 Command       fixed bin(31), /* Command identifier */
    5 MsgseqNumber fixed bin(31), /* Message sequence number */
    5 Control       fixed bin(31), /* Control options */
    5 CompCode      fixed bin(31), /* Completion code */
    5 Reason        fixed bin(31), /* Reason code qualifying completion code */
    5 ParameterCount fixed bin(31); /* Count of parameter structures */

```

High Level Assembler declaración para MQEPH

```

MQEPH          DSECT
MQEPH_STRUCID  DS   CL4   Structure identifier
MQEPH_VERSION  DS   F     Structure version number
MQEPH_STRUCLNGTH *      DS   F     Total length of MQEPH including the
                                   MQCFH and parameter structures that
                                   follow it
MQEPH_ENCODING DS   F     Numeric encoding of data that follows
                                   last PCF parameter structure
MQEPH_CODEDCHARSETID *   DS   F     Character set identifier of data that
                                   follows last PCF parameter structure
MQEPH_FORMAT   DS   CL8   Format name of data that follows last
                                   PCF parameter structure
MQEPH_FLAGS    DS   F     Flags
MQEPH_PCFHEADER DS   0F   Force fullword alignment
MQEPH_PCFHEADER_TYPE DS   F     Structure type
MQEPH_PCFHEADER_STRUCLNGTH DS   F     Structure length
MQEPH_PCFHEADER_VERSION DS   F     Structure version number
MQEPH_PCFHEADER_COMMAND DS   F     Command identifier
MQEPH_PCFHEADER_MSGSEQUENumber DS   F     Structure length
MQEPH_PCFHEADER_CONTROL DS   F     Control options
MQEPH_PCFHEADER_COMPCODE DS   F     Completion code
MQEPH_PCFHEADER_REASON DS   F     Reason code qualifying completion code
MQEPH_PCFHEADER_PARAMETER COUNT DS   F     Count of parameter structures
MQEPH_PCFHEADER_LENGTH EQU   *-MQEPH_PCFHEADER
                                ORG   MQEPH_PCFHEADER
MQEPH_PCFHEADER_AREA DS   CL(MQEPH_PCFHEADER_LENGTH)
MQEPH_LENGTH   EQU   *-MQEPH
                                ORG   MQEPH
MQEPH_AREA     DS   CL(MQEPH_LENGTH)

```

Declaración de Visual Basic para MQEPH

```

Type MQEPH
  StrucId      As String*4 'Structure identifier'
  Version      As Long     'Structure version number'
  StrucLength  As Long     'Total length of MQEPH structure including the MQCFH'
                                   'and parameter structures that follow it'
  Encoding     As Long     'Numeric encoding of data that follows last'
                                   'PCF parameter structure'
  CodedCharSetId As Long   'Character set identifier of data that'
                                   'follows last PCF parameter structure'
  Format       As String*8 'Format name of data that follows last PCF'
                                   'parameter structure'
  Flags       As Long     'Flags'

```

```

PCFHeader      As MQCFH      'Programmable command format header'
End Type

Global MQEPH_DEFAULT As MQEPH

```

Detalles de campo para MQEPH

La estructura del encabezado PCF integrado MQEPH describe los datos adicionales que están presentes en un mensaje cuando dicho mensaje es un mensaje de formato de comando programable (PCF). El *PCFHeader* campo define los parámetros PCF que siguen esta estructura y esto le permite seguir los datos del mensaje PCF con otros encabezados.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 502. Campos en MQEPH		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
StrucId (identificador de estructura)	MQEPH_STRUC_ID	'EPH↵'
Versión (número de versión de la estructura)	MQEPH_VERSION_1	1
StrucLength (longitud de la estructura MQEPH más las estructuras MQCFH y de parámetros que la siguen)	MQEPH_ESTRUCTURA_LONGITUD_FIJA	68
Codificación (codificación numérica de los datos que siguen la estructura del último parámetro PCF)	Ninguna	0
CodedCharSetId (identificador del conjunto de caracteres de los datos que siguen a la última estructura de parámetros PCF)	MQCCSI_INDETERMINADO	0
Formato (nombre del formato de los datos que siguen a la última estructura de parámetros PCF)	MQFMT_NONE	Espacios en blanco
Banderas (banderas)	MQEPH_NONE	0
PCFHeader (encabezado de formato de comando programable (PCF))	Nombres y valores tal y como se definen en Tabla 503 en la página 331	0
Notas: - El símbolo ↵ representa un único carácter en blanco. - En el lenguaje de programación C, la variable macroMQEPH_DEFAULT contiene los valores que se enumeran en la tabla. Úselo de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQEPH MyEPH = {MQEPH_DEFAULT};</pre>		

StrucId (MQCHAR4)

Este es el identificador de estructura de la estructura de encabezado PCF incrustada. Siempre es un campo de entrada. Su valor es MQEPH_STRUC_ID.

El valor debe ser:

MQEPH_STRUC_ID

Identificador de la estructura de encabezado PCF incrustada.

Para el lenguaje de programación C, también se define la constante MQEPH_STRUC_ID_ARRAY. Tiene el mismo valor que MQEPH_STRUC_ID, pero es una matriz de caracteres en lugar de una cadena.

Versión (MQLONG)

El valor debe ser:

MQEPH_VERSION_1

Número de versión para la estructura de encabezado PCF incrustada.

La siguiente constante especifica el número de versión de la versión actual:

MQCFH_VERSION_3

Versión actual de la estructura del encabezado PCF integrado.

El valor inicial de este campo es MQEPH_VERSION_1.

StrucLength (MQLONG)

Es la cantidad de datos que preceden a la siguiente estructura de encabezado. Incluye:

- La longitud del encabezado MQEPH
- La longitud de todos los parámetros PCF que siguen al encabezado
- Cualquier relleno en blanco que siga a esos parámetros

StrucLength debe ser múltiplo de 4.

La parte de longitud fija de la estructura se define mediante MQEPH_STRUC_LENGTH_FIXED.

El valor inicial de este campo es 68.

Encoding (MQLONG)

Se trata de la codificación numérica de los datos que siguen la estructura MQEPH y los parámetros PCF asociados; no se aplica a los datos de caracteres de la propia estructura MQEPH.

El valor inicial de este campo es 0.

CodedCharSetId (MQLONG)

Este es el identificador del conjunto de caracteres de los datos que siguen a la estructura MQEPH y los parámetros PCF asociados; no se aplica a los datos de caracteres de la propia estructura MQEPH.

El valor inicial de este campo es MQCCSI_UNDEFINED.

Format (MQCHAR8)

Este es el nombre del formato de los datos que siguen la estructura MQEPH y los parámetros PCF asociados.

El valor inicial de este campo es MQFMT_NONE.

Flags (MQLONG)

Están disponibles los valores siguientes:

MQEPH_NONE

No se han especificado banderas. MQEPH_NONE se define para facilitar la documentación del programa. No está previsto que esta constante se utilice con ninguna otra, pero como su valor es cero, dicho uso no puede detectarse.

MQEPH_CCSID_EMBEDDED

El conjunto de caracteres de los parámetros que contienen datos de caracteres se especifica individualmente dentro del campo « CodedCharSetId » (Conjunto de caracteres de parámetros) en

cada estructura. El conjunto de caracteres de los campos « StrucId » y «Format» se define mediante el campo « CodedCharSetId » de la estructura de encabezado que precede a la estructura MQEPH, o mediante el campo « CodedCharSetId » de MQMD si MQEPH se encuentra al principio del mensaje.

El valor inicial de este campo es MQEPH_NONE.

PCFHeader (MQCFH)

Este es el encabezado del formato de comando programable (PCF), que define los parámetros PCF que siguen la estructura MQEPH. Esto le permite seguir los datos del mensaje PCF con otras cabeceras.

El encabezado PCF se define inicialmente con los siguientes valores:

Tabla 503. Campos en MQCFH		
Nombre de campo	Nombre de la constante	Valor de la constante
Type	MQCFT_NINGUNO	0
StrucLength	MQCFH_STRUC_LENGTH	36
Version	MQCFH_VERSION_3	3
StrucLength	Ninguna	0
Command	MQCMD_NINGUNO	0
MsgSeqNumber	Ninguna	1
Control	MQCFC_ÚLTIMO	1
CompCode	MQCC_OK	0
Reason	MQRC_NONE	0
ParameterCount	Ninguna	0

La aplicación debe cambiar el Type de MQCFT_NONE a un tipo de estructura válido para el uso que está haciendo del encabezado PCF incrustado.

MQGMO - Opciones para obtener mensajes

La estructura MQGMO permite a la aplicación controlar cómo se eliminan los mensajes de las colas. La estructura es un parámetro de entrada/salida en la llamada MQGET.

Versión

La versión actual de MQGMO es MQGMO_VERSION_4. Algunos campos solo están disponibles en determinadas versiones de MQGMO. Si necesita transferir aplicaciones entre varios entornos, debe asegurarse de que la versión de MQGMO sea la misma en todos los entornos. Los campos que solo existen en versiones concretas de la estructura se identifican como tales en [“MQGMO - Opciones para obtener mensajes”](#) en la página 331 y en las descripciones de los campos.

Los archivos de encabezado, COPY e INCLUDE proporcionados para los lenguajes de programación compatibles contienen la versión más reciente de MQGMO compatible con el entorno, pero con el valor inicial del Version campo establecido en MQGMO_VERSION_1. Para utilizar campos que no están presentes en la estructura version-1, establezca el Version campo en el número de versión de la versión requerida.

Conjunto de caracteres y codificación

Los datos en MQGMO deben estar en el conjunto de caracteres proporcionado por el atributo del CodedCharSetId gestor de colas y la codificación del gestor de colas local proporcionada por

MQENC_NATIVE. Sin embargo, si la aplicación se ejecuta como un cliente MQ MQI, la estructura debe estar en el conjunto de caracteres y la codificación del cliente.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 504. Campos en MQGMO		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>StrucId</u> (identificador de estructura)	MQGMO_STRUC_ID	'GMO↵'
<u>Versión</u> (número de versión de la estructura)	MQGMO_VERSION_1	1
MQGMO - Campo de opciones (opciones que controlan la acción de MQGET)	MQGMO_NO_WAIT	0
<u>WaitInterval</u> (intervalo de espera)	Ninguna	0
<u>Signal1</u> (señal)	Ninguna	Puntero nulo en z/OS ; 0 de lo contrario
<u>Signal2</u> (identificador de señal)	Ninguna	0
<u>ResolvedQName</u> (nombre resuelto de la cola de destino)	Ninguna	Cadena nula o espacios en blanco
Nota: Los campos restantes se ignoran si <i>Version</i> es menor que MQGMO_VERSION_2.		
MatchOptions (opciones que controlan los criterios de selección utilizados para MQGET)	MQMO_MATCH_MSG_ID + MQMO_MATCH_CORREL_ID	3
<u>GroupStatus</u> (indicador que señala si el mensaje recuperado pertenece a un grupo)	MQGS_NO_EN_GRUPO	'↵'
<u>SegmentStatus</u> (indicador que señala si el mensaje recuperado es un segmento de un mensaje lógico)	MQSS_NO_ES_UN_SEG MENTO	'↵'
<u>Segmentación</u> (indicador que señala si se permite una mayor segmentación del mensaje recuperado)	MQSEG_INHIBIDO	'↵'
<u>Reserved1</u> (reservado)	Ninguna	'↵'
Nota: Los campos restantes se ignoran si <i>Version</i> es menor que MQGMO_VERSION_3.		
<u>MsgToken</u> (token de mensaje)	MQMTOK_NONE	Nulos
<u>ReturnedLength</u> (longitud en bytes de los datos del mensaje devueltos)	MQRL_INDETERMINAD O	-1
Nota: Los campos restantes se ignoran si <i>Version</i> es menor que MQGMO_VERSION_4.		
<u>Reserved2</u> (reservado)	Ninguna	'↵'
<u>MsgHandle</u> (identificador de un mensaje que se rellenará con las propiedades del mensaje que se recupera de la cola)	MQHM_NINGUNO	0

Tabla 504. Campos en MQGMO (continuación)

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
Notas: 1. El símbolo ~ representa un único carácter en blanco. 2. El valor «Cadena nula o espacios en blanco» denota la cadena nula en C y los caracteres en blanco en otros lenguajes de programación. 3. En el lenguaje de programación C, la variable macroMQGMO_DEFAULT contiene los valores que se enumeran en la tabla. Se puede utilizar de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQGMO MyGMO = {MQGMO_DEFAULT};</pre>		

Declaraciones lingüísticas

Declaración C para MQGMO

```
typedef struct tagMQGMO MQGMO;
struct tagMQGMO {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQLONG     Options;          /* Options that control the action of */
                                /* MQGET */
    MQLONG     WaitInterval;     /* Wait interval */
    MQLONG     Signal1;          /* Signal */
    MQLONG     Signal2;          /* Signal identifier */
    MQCHAR48    ResolvedQName;   /* Resolved name of destination queue */
    /* Ver:1 */
    MQLONG     MatchOptions;     /* Options controlling selection */
                                /* criteria used for MQGET */
    MQCHAR     GroupStatus;      /* Flag indicating whether message */
                                /* retrieved is in a group */
    MQCHAR     SegmentStatus;    /* Flag indicating whether message */
                                /* retrieved is a segment of a logical */
                                /* message */
    MQCHAR     Segmentation;     /* Flag indicating whether further */
                                /* segmentation is allowed for the */
                                /* message retrieved */
    MQCHAR     Reserved1;        /* Reserved */
    /* Ver:2 */
    MQBYTE16    MsgToken;        /* Message token */
    MQLONG     ReturnedLength;   /* Length of message data returned */
                                /* (bytes) */
    /* Ver:3 */
    MQLONG     Reserved2;        /* Reserved */
    MQHMSG     MsgHandle;        /* Message handle */
    /* Ver:4 */
};
```

Nota: En z/OS, el *Signal1* campo se declara como PMQLONG.

COBOL declaración para MQGMO

```
** MQGMO structure
10 MQGMO.
**   Structure identifier
15 MQGMO-STRUCID      PIC X(4).
**   Structure version number
15 MQGMO-VERSION      PIC S9(9) BINARY.
**   Options that control the action of MQGET
15 MQGMO-OPTIONS      PIC S9(9) BINARY.
**   Wait interval
15 MQGMO-WAITINTERVAL PIC S9(9) BINARY.
**   Signal
15 MQGMO-SIGNAL1      PIC S9(9) BINARY.
**   Signal identifier
15 MQGMO-SIGNAL2      PIC S9(9) BINARY.
**   Resolved name of destination queue
```

```

15 MQGMO-RESOLVEDQNAME PIC X(48).
** Options controlling selection criteria used for MQGET
15 MQGMO-MATCHOPTIONS PIC S9(9) BINARY.
** Flag indicating whether message retrieved is in a group
15 MQGMO-GROUPSTATUS PIC X.
** Flag indicating whether message retrieved is a segment of a
** logical message
15 MQGMO-SEGMENTSTATUS PIC X.
** Flag indicating whether further segmentation is allowed for the
** message retrieved
15 MQGMO-SEGMENTATION PIC X.
** Reserved
15 MQGMO-RESERVED1 PIC X.
** Message token
15 MQGMO-MSGTOKEN PIC X(16).
** Length of message data returned (bytes)
15 MQGMO-RETURNEDLENGTH PIC S9(9) BINARY.
** Reserved
15 MQGMO-RESERVED2 PIC S9(9) BINARY.
** Message handle
15 MQGMO-MSGHANDLE PIC S9(18) BINARY.

```

Nota: En z/OS, el *Signal1* campo se declara como POINTER.

PL/I declaración para MQGMO

```

dcl
  1 MQGMO based,
    3 StrucId      char(4),      /* Structure identifier */
    3 Version      fixed bin(31), /* Structure version number */
    3 Options      fixed bin(31), /* Options that control the action of
                                   MQGET */
    3 WaitInterval fixed bin(31), /* Wait interval */
    3 Signal1      fixed bin(31), /* Signal */
    3 Signal2      fixed bin(31), /* Signal identifier */
    3 ResolvedQName char(48),    /* Resolved name of destination
                                   queue */
    3 MatchOptions fixed bin(31), /* Options controlling selection
                                   criteria used for MQGET */
    3 GroupStatus  char(1),      /* Flag indicating whether message
                                   retrieved is in a group */
    3 SegmentStatus char(1),    /* Flag indicating whether message
                                   retrieved is a segment of a logical
                                   message */
    3 Segmentation char(1),    /* Flag indicating whether further
                                   segmentation is allowed for the
                                   message retrieved */
    3 Reserved1    char(1),      /* Reserved */
    3 MsgToken     char(16),     /* Message token */
    3 ReturnedLength fixed bin(31); /* Length of message data returned
                                   (bytes) */
    3 Reserved2    fixed bin(31); /* Reserved */
    3 MsgHandle    fixed bin(63); /* Message handle */

```

Nota: En z/OS, el *Signal1* campo se declara como pointer.

High Level Assembler declaración para MQGMO

MQGMO	DSECT		
MQGMO_STRUCID	DS	CL4	Structure identifier
MQGMO_VERSION	DS	F	Structure version number
MQGMO_OPTIONS	DS	F	Options that control the action of MQGET
* MQGMO_WAITINTERVAL	DS	F	Wait interval
MQGMO_SIGNAL1	DS	F	Signal
MQGMO_SIGNAL2	DS	F	Signal identifier
MQGMO_RESOLVEDQNAME	DS	CL48	Resolved name of destination queue
MQGMO_MATCHOPTIONS	DS	F	Options controlling selection criteria used for MQGET
* MQGMO_GROUPSTATUS	DS	CL1	Flag indicating whether message retrieved is in a group
* MQGMO_SEGMENTSTATUS	DS	CL1	Flag indicating whether message retrieved is a segment of a logical message
* MQGMO_SEGMENTATION	DS	CL1	Flag indicating whether further segmentation is allowed for the message retrieved
* MQGMO_RESERVED1	DS	CL1	Reserved
MQGMO_MSGTOKEN	DS	XL16	Message token

```

MQGMO_RETURNEDLENGTH DS F Length of message data returned (bytes)
MQGMO_RESERVED2 DS F Reserved
MQGMO_MSGHANDLE DS D Message handle
MQGMO_LENGTH EQU *-MQGMO
ORG MQGMO
MQGMO_AREA DS CL(MQGMO_LENGTH)

```

High Level Assembler declaración para MQGMO

```

Type MQGMO
  StrucId      As String*4 'Structure identifier'
  Version     As Long      'Structure version number'
  Options      As Long      'Options that control the action of MQGET'
  WaitInterval As Long      'Wait interval'
  Signal1     As Long      'Signal'
  Signal2     As Long      'Signal identifier'
  ResolvedQName As String*48 'Resolved name of destination queue'
  MatchOptions As Long      'Options controlling selection criteria'
                                'used for MQGET'
  GroupStatus  As String*1  'Flag indicating whether message'
                                'retrieved is in a group'
  SegmentStatus As String*1 'Flag indicating whether message'
                                'retrieved is a segment of a logical'
                                'message'
  Segmentation As String*1  'Flag indicating whether further'
                                'segmentation is allowed for the message'
                                'retrieved'
  Reserved1    As String*1  'Reserved'
  MsgToken     As MQBYTE16  'Message token'
  ReturnedLength As Long    'Length of message data returned (bytes)'
End Type

```

PROPCTL Opciones de canal para MQGMO

Utilice el **PROPCTL** atributo channel para controlar qué propiedades del mensaje se incluyen en un mensaje que se envía desde un gestor IBM MQ 10.0 de colas a un gestor de colas asociado desde una versión anterior de IBM MQ

Tabla 505. Valores de atributos de propiedades de mensaje de canal	
PROPCTL	Descripción
todos	Utilice esta opción si las aplicaciones conectadas al gestor de colas asociado desde una versión anterior pueden procesar cualquier propiedad colocada en un mensaje por una IBM MQ 10.0 aplicación. Todas las propiedades se envían al gestor de colas asociado, además de cualquier par nombre-valor colocado en el MQRFH2. Debe contemplar dos posibilidades de diseño de aplicaciones: 1. Una aplicación conectada al gestor de colas asociado debe poder procesar mensajes que contengan MQRFH2 encabezados generados en un IBM MQ 10.0 gestor de colas. 2. La aplicación conectada al gestor de colas de asociado debe procesar nuevas propiedades de mensaje que están marcadas con MQPD_SUPPORT_REQUIRED correctamente. Con la opción ALL de canal establecida, JMS las aplicaciones pueden interoperar entre sí IBM MQ 10.0 y con una versión anterior utilizando el canal. Las nuevas aplicaciones IBM MQ 10.0 que utilizan propiedades de mensaje pueden interactuar con aplicaciones de una versión anterior, dependiendo de cómo la aplicación de la versión anterior gestione las cabeceras MQRFH2.

Tabla 505. Valores de atributos de propiedades de mensaje de canal (continuación)

PROPCTL	Descripción
COMPAT	Utilice esta opción para enviar propiedades de mensaje a aplicaciones conectadas a un gestor de colas de asociado de una versión anterior en algunos casos, pero no siempre. Las propiedades de mensaje sólo se envían si se cumplen dos condiciones: 1. Ninguna propiedad debe marcarse de modo que requiera proceso de propiedad de mensaje. 2. Una de las propiedades de mensaje debe estar como mínimo en una carpeta "reservada"; consulte Nota. Con la opción del canal COMPAT activada, JMS las aplicaciones pueden interoperar entre sí IBM MQ 10.0 y con una versión anterior utilizando el canal. El canal no está disponible para cada aplicación utilizando propiedades de mensaje, sólo para aquellas aplicaciones que utilizan las carpetas reservadas. Las reglas relativas a si el mensaje o la propiedad se envía son: 1. Si el mensaje tiene propiedades, pero ninguna de las propiedades está asociada con una carpeta "reservada", no se envía ninguna propiedad de mensaje. 2. Si se ha creado cualquier propiedad de mensaje en una carpeta de propiedad "reservada", se envían todas las propiedades de mensaje asociadas con el mensaje. No obstante: a. Si alguna de las propiedades de mensaje está marcada para requerir soporte, MQPD_SUPPORT_REQUIRED o MQPD_SUPPORT_REQUIRED_IF_LOCAL, se rechaza todo el mensaje. Se devuelve, descarga o envía a la cola de mensajes no entregados según el valor de sus opciones de informe. b. Si no hay ninguna propiedad de mensaje que requiera soporte, es posible que una propiedad individual no se envíe. Si alguno de los campos de descriptor de propiedad de mensaje está establecido en valores no predeterminados, la propiedad individual no se envía. El mensaje se envía de todos modos. Un ejemplo de valor de campo de descriptor de propiedad no predeterminado es MQPD_USER_CONTEXT. Nota: Los nombres de carpetas "reservadas" empiezan por mcd., jms., usr. o mqext.. Estas carpetas se crean para aplicaciones que utilizan la interfaz JMS. En IBM MQ 10.0 cualquier par nombre-valor que se coloque en estas carpetas se trata como propiedades del mensaje. Las propiedades del mensaje se envían en un MQRFH2 encabezado, además de cualquier par nombre-valor colocado en un MQRFH2 encabezado. Cualquier par nombre-valor colocado en un MQRFH2 encabezado se envía siempre que el mensaje no sea rechazado.
NONE	Utilice esta opción para impedir que se envíe cualquier propiedad de mensaje a aplicaciones conectadas a un gestor de colas de asociado de una versión anterior. Se sigue enviando un MQRFH2 que contiene pares nombre-valor y propiedades de mensaje, pero solo con los pares nombre-valor. Con la opción de canal NONE establecida, se envía un mensaje JMS como JMSTextMessage o un JMSBytesMessage sin ninguna propiedad de mensaje JMS. Si una aplicación de versión anterior puede ignorar todas las propiedades establecidas en una IBM MQ 10.0 aplicación, puede interoperar con ella.

PROPCTL Opciones de cola para MQGMO

Utilice el atributo de cola **PROPCTL** para controlar cómo se devuelven las propiedades de mensaje a una aplicación que llama a **MQGET** sin establecer ninguna opción de propiedad de mensaje **MQGMO**.

Tabla 506. Valores de atributos de propiedades de mensaje de cola

PROPCTL	Descripción
todos	Utilice la opción ALL para que las diferentes aplicaciones que leen un mensaje de la misma cola puedan procesar el mensaje de diferentes maneras. • Una aplicación migrada sin cambios desde una versión anterior puede leer la cabecera MQRFH2 directamente. Las propiedades son accesibles directamente en la cabecera MQRFH2. Debe modificar la aplicación para manejar cualquier propiedad nueva y atributos de propiedad nuevos. Es posible que la aplicación se pueda ver afectada por cambios en el diseño y el número de cabeceras MQRFH2. Algunos atributos de carpeta pueden ser eliminados o puede que IBM MQ notifique un error en el diseño de la cabecera MQRFH2 que pasó por alto en una versión anterior. • Una aplicación nueva o modificada puede utilizar la propiedad de mensajes MQI para consultar las propiedades de los mensajes y leer directamente los pares nombre-valor del MQRFH2 encabezado. Todas las propiedades del mensaje se devuelven a la aplicación. • Si la aplicación llama a MQCRTMH para crear un manejador de mensajes, debe consultar las propiedades de mensaje utilizando MQINQMP. Los pares nombre-valor que no son propiedades del mensaje permanecen en el MQRFH2, que se despoja de cualquier propiedad del mensaje. • Si la aplicación no crea un identificador de mensaje, todas las propiedades del mensaje y los pares nombre-valor permanecen en el MQRFH2. ALL solo tiene este efecto si la aplicación receptora no ha establecido una MQGMO_PROPERTIES opción, o la ha establecido en MQGMO_PROPERTIES_AS_Q_DEF.

Tabla 506. Valores de atributos de propiedades de mensaje de cola (continuación)

PROPCTL	Descripción
COMPAT (predeterminado)	COMPAT es la opción predeterminada. Si <code>GM0_PROPERTIES_*</code> no se ha establecido, como en una aplicación no modificada de una versión anterior, se presupone COMPAT. Mediante el uso de COMPAT como opción predeterminada, una aplicación de una versión anterior que no creó explícitamente una cabecera MQRFH2, puede funcionar sin cambios en IBM MQ 10.0. Utilice esta opción si ha escrito una aplicación MQI de una versión anterior para leer mensajes JMS. - Las JMS propiedades, que se almacenan en un MQRFH2 encabezado, se devuelven a la aplicación en un MQRFH2 encabezado en carpetas cuyos nombres comienzan por <code>mcld.</code>, <code>jms.</code>, <code>usr.</code>, o <code>mqext.</code> - Si el mensaje tiene JMS carpetas y si una IBM MQ 10.0 aplicación añade nuevas carpetas de propiedades al mensaje, estas propiedades también se devuelven en el MQRFH2. Por lo tanto, debe modificar la aplicación para manejar cualquier propiedad nueva y nuevos atributos de propiedad. Es posible que una aplicación no modificada se pueda ver afectada por cambios en el diseño y el número de cabeceras MQRFH2. Puede encontrar que algunos atributos de carpeta se han eliminado o que IBM MQ encuentra errores en el diseño de la cabecera MQRFH2 que se pasan por alto en una versión anterior. Nota: En este caso, el comportamiento de la aplicación es el mismo tanto si está conectada a un gestor de colas de una versión anterior o IBM MQ 10.0. Si el atributo de canal PROPCTL está establecido en COMPAT o ALL, todas las propiedades de mensaje nuevas se envían al gestor de colas de la versión anterior. - Si el mensaje no es un mensaje JMS, pero contiene otras propiedades, dichas propiedades no se devuelven a la aplicación en una cabecera MQRFH2.¹ - Esta opción también permite que, en muchos casos, las aplicaciones de una versión anterior que crean explícitamente una cabecera MQRFH2 funcionen correctamente. Por ejemplo, un programa de la interfaz de colas de mensajes que crea una MQRFH2 que contiene propiedades de mensaje JMS continúa funcionando correctamente. Si un mensaje se crea sin propiedades de mensaje JMS pero con algunas otras carpetas MQRFH2, las carpetas se devuelven a la aplicación. Sólo se eliminarán dichas carpetas específicas de la MQRFH2 si son carpetas de propiedades de mensaje. Las carpetas de propiedades de mensaje se identifican por tener el nuevo atributo de carpeta <code>content='properties'</code>, o son carpetas cuyo nombre aparece en <u>Nombre de carpeta de propiedades definidas</u> o <u>Nombre de carpeta de propiedades sin agrupar</u>. - Si la aplicación llama a MQCRTMH para crear un manejador de mensajes, debe consultar las propiedades de mensaje utilizando MQINQMP. Las propiedades de mensaje se eliminan de las cabeceras MQRFH2. Los pares nombre-valor que no son propiedades del mensaje permanecen en el MQRFH2. - Si la aplicación llama a MQCRTMH para crear un manejador de mensajes, puede consultar a todas las propiedades de mensaje, independientemente de si el mensaje tiene carpetas JMS. - Si la aplicación no crea un identificador de mensaje, todas las propiedades del mensaje y los pares nombre-valor permanecen en el MQRFH2. Si un mensaje contiene nuevas carpetas de propiedades de usuario, puede deducir que el mensaje lo creó una aplicación IBM MQ 10.0 nueva o cambiada. Si la aplicación receptora va a procesar estas propiedades directamente en una MQRFH2, debe modificar la aplicación para utilizar la opción ALL. Con la opción predeterminada COMPAT establecida, una aplicación no modificada puede procesar el resto de la cabecera MQRFH2, sin las propiedades de IBM MQ 10.0. El objetivo de la interfaz PROPCTL es dar soporte a las aplicaciones antiguas que leen carpetas MQRFH2 y a las aplicaciones nuevas y cambiadas utilizando la interfaz de propiedades de mensaje. Está pensada para que las nuevas aplicaciones utilicen la interfaz de propiedades de mensaje para todas las propiedades de mensaje de usuario y para evitar leer y escribir cabeceras MQRFH2 directamente.

Tabla 506. Valores de atributos de propiedades de mensaje de cola (continuación)

PROPCTL	Descripción
FORCE	La opción FORCE coloca todas las propiedades de mensaje en cabeceras MQRFH2. Todas las propiedades del mensaje y los pares nombre-valor de los MQRFH2 encabezados permanecen en el mensaje. Las propiedades de mensaje no se eliminan de MQRFH2 y están disponibles a través de un manejador de mensajes. El efecto de elegir la opción FORCE es habilitar una aplicación recién migrada para leer propiedades de mensaje de cabeceras MQRFH2. Suponga que ha modificado una aplicación para procesar propiedades de mensaje de la IBM MQ 10.0 pero también ha mantenido la posibilidad de que funcione directamente con cabeceras MQRFH2, como se hacía anteriormente. Puede decidir cuándo la aplicación pasará a utilizar propiedades de mensaje estableciendo inicialmente el atributo de cola PROPCTL en FORCE. Establezca el atributo de cola PROPCTL en otro valor cuando esté preparado para empezar a utilizar propiedades de mensaje. Si la nueva función en la aplicación no se comporta según lo previsto, establezca la opción PROPCTL en FORCE. FORCE solo tiene este efecto si la aplicación receptora no ha establecido una MQGMO_PROPERTIES opción, o la ha establecido en MQGMO_PROPERTIES_AS_Q_DEF.
NONE	Utilice la opción NONE para que una aplicación existente pueda procesar un mensaje, ignorando todas las propiedades del mensaje, y una aplicación nueva o modificada pueda consultar las propiedades del mensaje. • Si la aplicación llama a MQCRTMH para crear un manejador de mensajes, debe consultar las propiedades de mensaje utilizando MQINQMP. Los pares nombre-valor que no son propiedades del mensaje permanecen en el MQRFH2, que se despoja de cualquier propiedad del mensaje. • Si la aplicación no crea un manejador de mensajes, todas las propiedades de mensaje se eliminan de MQRFH2. Los pares nombre-valor de los MQRFH2 encabezados permanecen en el mensaje. NONE solo tiene este efecto si la aplicación receptora no ha establecido una MQGMO_PROPERTIES opción, o la ha establecido en MQGMO_PROPERTIES_AS_Q_DEF.
V6COMPAT	Utilice esta opción para recibir MQRFH2 en el mismo formato que se envió. Si la aplicación emisora o el gestor de colas crea propiedades de mensaje adicionales, se devuelven en el manejador de mensajes. Esta opción se debe establecer en las colas emisora y receptora y en cualquier cola de transmisión que intervenga. Prevalece sobre cualquier opción PROPCTL establecido en definiciones de cola en la vía de acceso de resolución de nombres de colas. Utilice la V6COMPAT opción solo en circunstancias excepcionales. Por ejemplo, si está migrando aplicaciones de una versión anterior a IBM MQ 10.0, la opción es valiosa porque conserva el comportamiento de la versión anterior. La opción es probable que afecte al rendimiento de los mensajes. También es más difícil de administrar; deberá asegurarse de que la opción esté establecida en las colas emisora, receptora e interviniente. V6COMPAT solo tiene este efecto si la aplicación receptora no ha establecido una MQGMO_PROPERTIES opción, o la ha establecido en MQGMO_PROPERTIES_AS_Q_DEF.

Para obtener más información sobre las propiedades de los mensajes y los pares nombre-valor, consulte [“NameValueData \(MQCHARn\)”](#) en la página 517.

¹ La existencia de carpetas de propiedades específicas creadas por el IBM MQ classes for JMS indica un JMS mensaje. Las carpetas de la propiedad son mcd., jms., usr., o mqext.

Opciones de propiedades de mensaje para MQGMO

Utilice las opciones de la **MQGMO** propiedad message para controlar cómo se devuelven las propiedades del mensaje a una aplicación.

Tabla 507. Valores de las opciones de propiedades de mensaje MQGMO

MQGMO Option	Descripción
MQGMO_PROPERTIES_AS_Q_DEF	IBM MQ Las aplicaciones que leen de la misma cola y no establecen GMO_PROPERTIES_*, reciben las propiedades del mensaje de forma diferente. IBM MQ Las aplicaciones que no crean un identificador de mensaje se controlan mediante el atributo PROPCTL de cola. Una IBM MQ aplicación puede optar por recibir las propiedades del mensaje en el MQRFH2, o crear un identificador de mensaje y consultar las propiedades del mensaje. Si la aplicación crea un manejador de mensajes, las propiedades se eliminan de MQRFH2. - Una aplicación IBM MQ nueva o cambiada que no define GMO_PROPERTIES_* o que establece su valor en MQGMO_PROPERTIES_AS_Q_DEF puede elegir consultar propiedades de mensaje. Debe definir MQCRTMH para crear un manejador de mensajes y consultar propiedades de mensaje utilizando la llamada de MQI MQINQMP. - Si una aplicación nueva o modificada no crea un identificador de mensaje, debe leer las propiedades de los mensajes que recibe directamente de los MQRFH2 encabezados. - Si el atributo de cola PROPCTL se establece en FORCE, no se devuelven propiedades en el manejador de mensajes. Todas las propiedades se devuelven en cabeceras MQRFH2. - Si el atributo de cola PROPCTL está establecido en NONE o COMPAT, una IBM MQ aplicación que crea un identificador de mensaje recibe todas las propiedades del mensaje.
MQGMO_PROPERTIES_IN_HANDLE	Fuerza una aplicación a utilizar propiedades de mensaje. Utilice esta opción para detectar si una aplicación modificada no logra crear un manejador de mensajes. La aplicación podría estar intentando leer propiedades de mensaje directamente de una MQRFH2, en lugar de llamar a MQINQMP.
MQGMO_NO_PROPERTIES	- Se eliminan todas las propiedades.. Las propiedades generadas por el gestor de colas como, por ejemplo, propiedades JMS, se eliminan. - Las propiedades se eliminan aunque se cree un manejador de mensajes. Los pares nombre-valor de otras MQRFH2 carpetas están disponibles en los datos del mensaje.
MQGMO_PROPERTIES_FORCE_MQRFH2	Las propiedades se devuelven en las cabeceras MQRFH2, aunque se cree un manejador de mensajes. MQINQMP no devuelve ninguna propiedad de mensaje, aunque se cree un manejador de mensajes. Se devuelve MQRC_PROPERTY_NOT_AVAILABLE si se consulta una propiedad.

Tabla 507. Valores de las opciones de propiedades de mensaje MQGMO (continuación)

MQGMO Option	Descripción
MQGMO_PROPERTIES_COMPATIBILITY	Si el mensaje proviene de un cliente JMS, las propiedades JMS se devuelven en las cabeceras MQRFH2. Las aplicaciones IBM MQ nuevas o modificadas que crean un manejador de mensajes se comportan de forma diferente. • Todas las propiedades en cualquier carpeta de propiedades de mensaje se devuelven si el mensaje contiene una carpeta mcd., jms., usr., o mqext. • Si el mensaje contiene carpetas de propiedad, pero no una carpeta mcd., jms., usr., o mqext, no se devolverá ninguna propiedad de mensaje en una MQRFH2. • Si una aplicación nueva o modificada de la IBM MQ crea un descriptor de contexto de mensajes, consulte las propiedades de mensajes utilizando la llamada MQI MQINQMP. Todas las propiedades de mensaje se eliminan de MQRFH2. • Si una aplicación IBM MQ nueva o modificada crea un manejador de mensajes, se pueden consultar todas las propiedades contenidas en el mensaje. Incluso si el mensaje no contiene una carpeta mcd., jms., usr., o mqext, se pueden consultar todas las propiedades de mensaje.

Referencia relacionada

PROPCTL

2471 (09A7) (RC2471): MQRC_PROPERTY_NOT_AVAILABLE

Detalles de campo para MQGMO

La estructura de opciones Get-message de MQGMO permite a la aplicación controlar cómo se eliminan los mensajes de las colas. La estructura es un parámetro de entrada/salida en la llamada MQGET.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 508. Campos en MQGMO		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>StrucId</u> (identificador de estructura)	MQGMO_STRUC_ID	'GMO-'
<u>Versión</u> (número de versión de la estructura)	MQGMO_VERSION_1	1
MQGMO - Campo de opciones (opciones que controlan la acción de MQGET)	MQGMO_NO_WAIT	0
<u>WaitInterval</u> (intervalo de espera)	Ninguna	0
<u>Signal1</u> (señal)	Ninguna	Puntero nulo en z/OS ; 0 de lo contrario
<u>Signal2</u> (identificador de señal)	Ninguna	0
<u>ResolvedQName</u> (nombre resuelto de la cola de destino)	Ninguna	Cadena nula o espacios en blanco

Tabla 508. Campos en MQGMO (continuación)

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
Nota: Los campos restantes se ignoran si <i>Version</i> es menor que MQGMO_VERSION_2.		
MatchOptions (opciones que controlan los criterios de selección utilizados para MQGET)	MQMO_MATCH_MSG_ID + MQMO_MATCH_CORREL_ID	3
GroupStatus (indicador que señala si el mensaje recuperado pertenece a un grupo)	MQGS_NO_EN_GRUPO	'¬'
SegmentStatus (indicador que señala si el mensaje recuperado es un segmento de un mensaje lógico)	MQSS_NO_ES_UN_SEG_MENTO	'¬'
Segmentación (indicador que señala si se permite una mayor segmentación del mensaje recuperado)	MQSEG_INHIBIDO	'¬'
Reserved1 (reservado)	Ninguna	'¬'
Nota: Los campos restantes se ignoran si <i>Version</i> es menor que MQGMO_VERSION_3.		
MsgToken (token de mensaje)	MQMTOK_NONE	Nulos
ReturnedLength (longitud en bytes de los datos del mensaje devueltos)	MQRL_INDETERMINADO	-1
Nota: Los campos restantes se ignoran si <i>Version</i> es menor que MQGMO_VERSION_4.		
Reserved2 (reservado)	Ninguna	'¬'
MsgHandle (identificador de un mensaje que se rellenará con las propiedades del mensaje que se recupera de la cola)	MQHM_NINGUNO	0
Notas: 1. El símbolo ¬ representa un único carácter en blanco. 2. El valor «Cadena nula o espacios en blanco» denota la cadena nula en C y los caracteres en blanco en otros lenguajes de programación. 3. En el lenguaje de programación C, la variable macro MQGMO_DEFAULT contiene los valores que se enumeran en la tabla. Se puede utilizar de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQGMO MyGMO = {MQGMO_DEFAULT};</pre>		

StrucId (MQCHAR4)

Este es el identificador de estructura de la estructura de opciones del mensaje get. Siempre es un campo de entrada. Su valor es MQGMO_STRUC_ID.

El valor debe ser:

MQGMO_STRUC_ID

Identificador para la estructura de opciones del mensaje get.

Para el lenguaje de programación C, también se define la constante MQGMO_STRUC_ID_ARRAY. Tiene el mismo valor que MQGMO_STRUC_ID, pero es una matriz de caracteres en lugar de una cadena.

Versión (MQLONG)

La versión es el número de versión de la estructura.

El valor debe ser uno de los siguientes:

MQGMO_VERSION_1

Version-1 Estructura de opciones de get-message.

Esta versión es compatible con todos los entornos.

MQGMO_VERSION_2

Version-2 Estructura de opciones de get-message.

Esta versión es compatible con todos los entornos.

MQGMO_VERSION_3

Version-3 Estructura de opciones de get-message.

Esta versión es compatible con todos los entornos.

MQGMO_VERSION_4

Version-4 Estructura de opciones de get-message.

Esta versión es compatible con todos los entornos.

Los campos que solo existen en las versiones más recientes de la estructura se identifican como tales en las descripciones de los campos. La siguiente constante especifica el número de versión de la versión actual:

MQGMO_VERSION_ACTUAL

Versión actual de la estructura de opciones de get-message.

Este campo siempre es un campo de entrada. El valor inicial de este campo es MQGMO_VERSION_1.

Opciones (MQLONG)

MQGMO Las opciones controlan la acción de MQGET. Puede especificar cero o más opciones. Si necesita más de un valor opcional:

- Añade los valores (no añadas la misma constante más de una vez), o
- Combine los valores utilizando la operación OR bit a bit (si el lenguaje de programación admite operaciones bit a bit).

Las combinaciones de opciones que no son válidas se indican; todas las demás combinaciones son válidas.

Opciones de espera

Las siguientes opciones se refieren a la espera de mensajes que llegan a la cola:

MQGMO_WAIT

La aplicación espera hasta que llega un mensaje adecuado. El tiempo máximo que espera la aplicación se especifica en *WaitInterval*.

Importante: No hay espera ni retraso si hay un mensaje adecuado disponible de inmediato.

Si MQGET las solicitudes se inhiben, o MQGET las solicitudes se inhiben mientras esperan, la espera se cancela. La llamada finaliza con MQCC_FAILED y el código de motivo MQRC_GET_INHIBITED, independientemente de si hay mensajes adecuados en la cola.

Puede utilizar MQGMO_WAIT con las opciones MQGMO_BROWSE_FIRST MQGMO_BROWSE_NEXT o.

Si varias aplicaciones están esperando en la misma cola compartida, las siguientes reglas seleccionan qué aplicación se activa cuando llega un mensaje adecuado:

Tabla 509. Reglas para activar MQGET llamadas en una cola compartida.		
Número de MQGET llamadas en espera de ser activadas		Resultado
Con una opción EXPLORAR	Sin opción EXPLORAR ²	
Ninguna	uno o más	Se activa una MQGET llamada sin opción BROWSE.
uno o más	Ninguna	Todas MQGET las llamadas con la opción BROWSE están activadas.
uno o más	uno o más	Se activa una MQGET llamada sin opción BROWSE. El número de MQGET llamadas con una opción BROWSE que se activan es impredecible.

Si hay más de una MQGET llamada sin opción BROWSE esperando en la misma cola, solo se activa una. El gestor de colas intenta dar prioridad a las llamadas en espera en el siguiente orden:

1. Solicitudes específicas de obtención-espera que solo pueden satisfacerse con determinados mensajes, por ejemplo, aquellos con un o MsgIdCorrelId específico (o ambos).
2. Solicitudes generales de espera que pueden satisfacerse con cualquier mensaje.

Nota:

- Dentro de la primera categoría, no se da prioridad adicional a las solicitudes get-wait más específicas. Por ejemplo, solicitudes que especifican tanto MsgId como CorrelId .
- Dentro de cualquiera de las dos categorías, no se puede predecir qué aplicación se seleccionará. En particular, la solicitud que lleva más tiempo en espera no es necesariamente la seleccionada.
- La longitud de la ruta y las consideraciones de programación de prioridades del sistema operativo pueden hacer que una aplicación en espera con una prioridad del sistema operativo inferior a la esperada recupere el mensaje.
- También puede ocurrir que una aplicación que no está en espera recupere el mensaje antes que otra que sí lo está.

 En z/OS, se aplican los siguientes puntos:

- Si desea que la aplicación continúe con otras tareas mientras espera la llegada del mensaje, considere utilizar la opción de señal (MQGMO_SET_SIGNAL) en su lugar. Sin embargo, la opción de señal es específica del entorno; las aplicaciones que se transfieren entre diferentes entornos no deben utilizarla.
- Si hay más de una MQGET llamada en espera para el mismo mensaje, con una combinación de opciones de espera y señal, cada llamada en espera se considera por igual. Es un error especificar MQGMO_SET_SIGNAL con MQGMO_WAIT. También es un error especificar esta opción con un identificador de cola para el que hay una señal pendiente.
- Si se especifica MQGMO_WAIT o MQGMO_SET_SIGNAL para una cola que tiene un IndexType de MQIT_MSG_TOKEN, no se permiten criterios de selección. Esto significa que:
 - Si utiliza un version-1 MQGMO, establezca los campos MsgId CorrelId y en el MQMD especificado en la MQGET llamada en MQMI_NONE y MQCI_NONE.
 - Si utiliza un version-2 o posterior MQGMO, establezca el MatchOptions campo en MQMO_NONE.
- Para una MQGET llamada en una cola compartida y la llamada es una solicitud de exploración, o una obtención destructiva de un mensaje de grupo, y ni MsgId ni CorrelId se van a emparejar, su ECB de señal se envía MQEC_MSG_ARRIVED después de 200 milisegundos.

Esto ocurre, aunque puede que no haya llegado ningún mensaje adecuado a la cola, hasta que expire el intervalo de espera, momento en el que se envía a la cola

² Una MQGET llamada que especifica la MQGMO_LOCK opción se trata como una llamada sin exploración.

MQEC_WAIT_INTERVAL_EXPIRED. Cuando se envía MQEC_MSG_ARRIVED, debe volver a emitir una segunda MQGET llamada para recuperar el mensaje, si hay alguno disponible.

Esta técnica se utiliza para garantizar que se le informe oportunamente de la llegada de un mensaje, pero puede parecer una sobrecarga de procesamiento inesperada en comparación con una secuencia de llamadas similar en una cola no compartida.

MQGMO_WAIT Se ignora si se especifica con MQGMO_BROWSE_MSG_UNDER_CURSOR o MQGMO_MSG_UNDER_CURSOR ; no se genera ningún error.

MQGMO_NO_WAIT

La aplicación no espera si no hay ningún mensaje adecuado disponible. MQGMO_NO_WAIT es lo contrario de MQGMO_WAIT. MQGMO_NO_WAIT Se define para facilitar la documentación del programa. Es el valor predeterminado si no se especifica ninguno.

MQGMO_SET_SIGNAL

Utilice esta opción con los campos Signal1 Signal2 y. Permite a las aplicaciones continuar con otras tareas mientras esperan la llegada de un mensaje. También permite (si se dispone de las funciones adecuadas del sistema operativo) que las aplicaciones esperen la llegada de mensajes en más de una cola.

Nota: La MQGMO_SET_SIGNAL opción es específica del entorno; no la utilice para aplicaciones que desee portar.

En dos circunstancias, la llamada se completa de la misma manera que si no se hubiera especificado esta opción:

1. Si un mensaje actualmente disponible cumple los criterios especificados en el descriptor del mensaje.
2. Si se detecta un error de parámetro u otro error sincrónico.

Si actualmente no hay ningún mensaje que cumpla los criterios especificados en el descriptor de mensajes, el control vuelve a la aplicación sin esperar a que llegue un mensaje. Los parámetros **CompCode Reason** y se establecen en MQCC_WARNING y MQRC_SIGNAL_REQUEST_ACCEPTED. Los demás campos de salida del descriptor de mensajes y los parámetros de salida de la MQGET llamada no están configurados. Cuando llega un mensaje adecuado más tarde, la señal se transmite mediante la publicación del BCE.

A continuación, la persona que llama debe volver a realizar la MQGET llamada para recuperar el mensaje. La aplicación puede esperar esta señal utilizando las funciones proporcionadas por el sistema operativo.

Si el sistema operativo proporciona un mecanismo de espera múltiple, puede utilizarlo para esperar a que llegue un mensaje a cualquiera de varias colas.

Si se especifica WaitInterval un valor distinto de cero, la señal se envía una vez transcurrido el intervalo de espera. El gestor de colas también puede cancelar la espera, en cuyo caso se envía la señal.

Más de una MQGET llamada puede establecer una señal para el mismo mensaje. El orden en que se activan las aplicaciones es el mismo que se describe para MQGMO_WAIT.

Si hay más de una MQGET llamada en espera para el mismo mensaje, cada llamada en espera se considera por igual. Las llamadas pueden incluir una combinación de opciones de espera y señal.

En determinadas condiciones, la MQGET llamada puede recuperar un mensaje y se puede enviar una señal resultante de la llegada de ese mismo mensaje. Cuando se envía una señal, la aplicación debe estar preparada para que no haya ningún mensaje disponible.

Un identificador de cola no puede tener más de una solicitud de señal pendiente.

Esta opción no es válida con ninguna de las siguientes opciones:

- MQGMO_UNLOCK
- MQGMO_WAIT

Para una MQGET llamada en una cola compartida y la llamada es una solicitud de exploración, o una obtención destructiva de un mensaje de grupo, y ni `MsgId` ni `CorrelId` se van a emparejar, la ECB de señal del usuario se envía MQEC_MSG_ARRIVED después de 200 milisegundos.

Esto ocurre aunque no haya llegado un mensaje adecuado a la cola, hasta que expire el intervalo de espera, momento en el que la cola se envía con MQEC_WAIT_INTERVAL_EXPIRED. Cuando MQEC_MSG_ARRIVED se publique, deberá volver a emitir una segunda MQGET llamada para recuperar el mensaje, si hay alguno disponible.

Esta técnica se utiliza para garantizar que se le informe oportunamente de la llegada de un mensaje, pero puede parecer una sobrecarga de procesamiento inesperada en comparación con una secuencia de llamadas similar en una cola no compartida.

Este no es un método eficaz para recuperar mensajes cuando estos se añaden con poca frecuencia. Para evitar esta sobrecarga en el caso de la exploración, especifique `MsgId` (si no está indexado o está indexado por `MsgId`) o `CorrelId` (si está indexado por `CorrelId`) coincidiendo en la MQGET llamada.

► **z/OS** Esta opción solo es compatible z/OS con.

MQGMO_FAIL_IF QUIESCING

Forzar el fracaso de la MQGET llamada si el gestor de colas se encuentra en estado de inactividad.

► **z/OS** En z/OS, esta opción también fuerza el fracaso MQGET de la llamada si la conexión (para una CICS o IMS aplicación) se encuentra en estado de inactividad.

Si esta opción se especifica con MQGMO_WAIT o MQGMO_SET_SIGNAL, y la espera o la señal están pendientes en el momento en que el gestor de colas entra en estado de inactividad:

- La espera se cancela y la llamada devuelve el código de finalización MQCC_FAILED con el código de motivo MQRC_Q_MGR QUIESCING o MQRC_CONNECTION QUIESCING.
- La señal se cancela con un código de finalización de señal específico del entorno.

► **z/OS** En z/OS, la señal se completa con el código de finalización MQEC_Q_MGR QUIESCING del evento o MQEC_CONNECTION QUIESCING.

Si no MQGMO_FAIL_IF QUIESCING se especifica y el gestor de colas o la conexión entran en estado de inactividad, la espera o la señal no se cancelan.

Opciones de punto de sincronización

Las siguientes opciones se refieren a la participación de la MQGET llamada dentro de una unidad de trabajo:

MQGMO_SYNCPOINT

La solicitud es operar dentro de los protocolos normales de unidad de trabajo. El mensaje se marca como no disponible para otras aplicaciones, pero solo se elimina de la cola cuando se confirma la unidad de trabajo. El mensaje vuelve a estar disponible si se cancela la unidad de trabajo.

Puedes dejar MQGMO_SYNCPOINT y MQGMO_NO_SYNCPOINT desactivar. En tal caso, la inclusión de la solicitud GET en los protocolos de unidad de trabajo viene determinada por el entorno en el que se ejecuta el gestor de colas. No está determinado por el entorno en el que se ejecuta la aplicación.

- ► **z/OS** En z/OS, la solicitud GET se encuentra dentro de una unidad de trabajo.
- En todos los entornos excepto z/OS, la solicitud GET no se encuentra dentro de una unidad de trabajo.

Debido a estas diferencias, una aplicación que desee portar no debe permitir que esta opción sea predeterminada; especifique MQGMO_SYNCPOINT o MQGMO_NO_SYNCPOINT explícitamente.

Esta opción no es válida con ninguna de las siguientes opciones:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR

- MQGMO_BROWSE_NEXT
- MQGMO_LOCK
- MQGMO_NO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT
- MQGMO_UNLOCK

MQGMO_SYNCPOINT_IF_PERSISTENT

La solicitud es operar dentro de los protocolos normales de unidad de trabajo, pero solo si el mensaje recuperado es persistente. Un mensaje persistente tiene el valor MQPER_PERSISTENT en el Persistence campo en MQMD.

- Si el mensaje es persistente, el gestor de colas procesa la llamada como si la aplicación hubiera especificado MQGMO_SYNCPOINT.
- Si el mensaje no es persistente, el gestor de colas procesa la llamada como si la aplicación hubiera especificado MQGMO_NO_SYNCPOINT.

Esta opción no es válida con ninguna de las siguientes opciones:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_BROWSE_NEXT
- MQGMO_COMPLETE_MSG
- MQGMO_MARK_SKIP_BACKOUT
- MQGMO_NO_SYNCPOINT
- MQGMO_SYNCPOINT
- MQGMO_UNLOCK

Esta opción es compatible con los siguientes entornos:

-  AIX
-  IBM i
-  Linux
-  z/OS

y para IBM MQ MQI clients conectados a estos sistemas.

MQGMO_NO_SYNCPOINT

La solicitud es operar fuera de los protocolos normales de unidad de trabajo. Si recibes un mensaje sin opción de exploración, se elimina inmediatamente de la cola. El mensaje no puede volver a estar disponible al retroceder la unidad de trabajo.

Esta opción se asume si se especifica MQGMO_BROWSE_FIRST o MQGMO_BROWSE_NEXT.

Puedes dejar MQGMO_SYNCPOINT y MQGMO_NO_SYNCPOINT desactivar. En tal caso, la inclusión de la solicitud GET en los protocolos de unidad de trabajo viene determinada por el entorno en el que se ejecuta el gestor de colas. No está determinado por el entorno en el que se ejecuta la aplicación.

-  En z/OS, la solicitud GET se encuentra dentro de una unidad de trabajo.
- En todos los entornos excepto z/OS, la solicitud GET no se encuentra dentro de una unidad de trabajo.

Debido a estas diferencias, una aplicación que desee portar no debe permitir que esta opción sea predeterminada; especifique o MQGMO_NO_SYNCPOINT o MQGMO_SYNCPOINT explícitamente.

Esta opción no es válida con ninguna de las siguientes opciones:

- MQGMO_MARK_SKIP_BACKOUT

- MQGMO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT

MQGMO_MARK_SKIP_BACKOUT

Retirar una unidad de trabajo sin volver a colocar en la cola el mensaje que se marcó con esta opción.

Esta opción solo es compatible con z/OS.

Si se especifica esta opción, también MQGMO_SYNCPOINT debe especificarse.

MQGMO_MARK_SKIP_BACKOUT no es válido con ninguna de las siguientes opciones:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_BROWSE_NEXT
- MQGMO_LOCK
- MQGMO_NO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT
- MQGMO_UNLOCK

Nota: En IMS y CICS, es posible que tenga que emitir una llamada IBM MQ extran después de revertir una unidad de trabajo que contenga un mensaje marcado con MQGMO_MARK_SKIP_BACKOUT. Debe emitir una IBM MQ llamada antes de confirmar la nueva unidad de trabajo que contiene el mensaje marcado. La llamada puede ser cualquier IBM MQ llamada que desees.

1. IMSSi no ha aplicado IMS APAR PN60855 y está ejecutando una aplicación IMS MPP o BMP.
2. Si está ejecutando alguna aplicación CICS.

En ambos casos, emita cualquier IBM MQ llamada antes de confirmar la nueva unidad de trabajo que contiene el mensaje deshecho.

Nota: Dentro de una unidad de trabajo, solo puede haber una solicitud GET marcada como «skipping backout», así como ninguna o varias solicitudes GET sin marcar.

Si una aplicación se retira de una unidad de trabajo, un mensaje que se recuperó utilizando no MQGMO_MARK_SKIP_BACKOUT se restaura a su estado anterior. Se han revertido otras actualizaciones de recursos. El mensaje se trata como si se hubiera recuperado en una nueva unidad de trabajo iniciada por la solicitud de retroceso. El mensaje se recupera sin la MQGMO_MARK_SKIP_BACKOUT opción.

MQGMO_MARK_SKIP_BACKOUT Es útil si, después de que se hayan modificado algunos recursos, resulta evidente que la unidad de trabajo no puede completarse con éxito. Si omite esta opción, al retirar la unidad de trabajo se restablece el mensaje en la cola. La misma secuencia de eventos se repite cuando se recupera el mensaje.

Sin embargo, si se especifica MQGMO_MARK_SKIP_BACKOUT en la llamada MQGET original, al deshacer la unidad de trabajo se deshacen las actualizaciones de los demás recursos. El mensaje se trata como si se hubiera recuperado en una nueva unidad de trabajo. La aplicación puede realizar el manejo adecuado de errores. Puede enviar un mensaje de informe al remitente del mensaje original o colocar el mensaje original en la cola de mensajes no entregados. A continuación, puede confirmar la nueva unidad de trabajo. Al confirmar la nueva unidad de trabajo, el mensaje se elimina de forma permanente de la cola original.

MQGMO_MARK_SKIP_BACKOUT marca un único mensaje físico. Si el mensaje pertenece a un grupo de mensajes, los demás mensajes del grupo no se marcan. Del mismo modo, si el mensaje marcado es un segmento de un mensaje lógico, los demás segmentos del mensaje lógico no se marcan.

Se puede marcar cualquier mensaje de un grupo, pero si los mensajes se recuperan utilizando MQGMO_LOGICAL_ORDER, es conveniente marcar el primer mensaje del grupo. Si se cancela la unidad de trabajo, el primer mensaje (marcado) se traslada a la nueva unidad de trabajo. El segundo mensaje y los siguientes del grupo se reincorporan a la cola. Los mensajes dejados en la cola no pueden ser recuperados por otra aplicación utilizando MQGMO_LOGICAL_ORDER. El primer mensaje

del grupo ya no está en la cola. Sin embargo, la aplicación que respaldaba la unidad de trabajo puede recuperar el segundo mensaje y los mensajes posteriores en la nueva unidad de trabajo utilizando la MQGMO_LOGICAL_ORDER opción. El primer mensaje ya se ha recuperado.

En ocasiones, es posible que necesite deshacer la nueva unidad de trabajo. Por ejemplo, porque la cola de mensajes no entregados está llena y el mensaje no debe descartarse. Al retirar la nueva unidad de trabajo, se restablece el mensaje en la cola original, lo que evita que se pierda. Sin embargo, en esta situación no se puede continuar con el procesamiento. Después de cancelar la nueva unidad de trabajo, la aplicación debe informar al operador o administrador de que se ha producido un error irrecuperable y, a continuación, finalizar.

MQGMO_MARK_SKIP_BACKOUT Solo funciona si la unidad de trabajo que contiene la solicitud GET es interrumpida por la aplicación que la respalda. Si la unidad de trabajo que contiene la solicitud GET se cancela debido a un fallo de la transacción o del sistema, MQGMO_MARK_SKIP_BACKOUT se ignora. Cualquier mensaje recuperado utilizando esta opción se restablece en la cola de la misma manera que los mensajes recuperados sin esta opción.

Opciones de navegación

Las siguientes opciones se refieren a la exploración de mensajes en la cola:

MQGMO_BROWSE_FIRST

Cuando se abre una cola con la opción MQGMO_BROWSE, se establece un cursor de exploración, situado lógicamente antes del primer mensaje de la cola. A continuación, puede utilizar MQGET llamadas especificando la MQGMO_BROWSE_MSG_UNDER_CURSOR opción MQGMO_BROWSE_FIRST, MQGMO_BROWSE_NEXT, o para recuperar mensajes de la cola de forma no destructiva. El cursor de exploración marca la posición, dentro de los mensajes de la cola, desde la que la siguiente MQGET llamada con MQGMO_BROWSE_NEXT busca un mensaje adecuado.

MQGMO_BROWSE_FIRST no es válido con ninguna de las siguientes opciones:

- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_BROWSE_NEXT
- MQGMO_MARK_SKIP_BACKOUT
- MQGMO_MSG_UNDER_CURSOR
- MQGMO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT
- MQGMO_UNLOCK

También es un error si la cola no se abrió para su exploración.

Una MQGET llamada a MQGMO_BROWSE_FIRST ignora la posición anterior del cursor de exploración. Se recupera el primer mensaje de la cola que cumple las condiciones especificadas en el descriptor de mensajes. El mensaje permanece en la cola y el cursor de exploración se coloca sobre este mensaje.

Después de esta llamada, el cursor de navegación se coloca en el mensaje que se ha devuelto. El mensaje podría eliminarse de la cola antes de que se emita la siguiente MQGETMQGMO_BROWSE_NEXT llamada con. En este caso, el cursor de exploración permanece en la posición que ocupaba el mensaje en la cola, aunque ahora esa posición esté vacía.

Utilice la MQGMO_MSG_UNDER_CURSOR opción con una llamada que no MQGET sea de exploración para eliminar el mensaje de la cola.

El cursor de exploración no se mueve con una llamada que no MQGET sea de exploración, incluso si se utiliza el mismo *Hobj* identificador. Tampoco se mueve por una llamada de MQGET exploración que devuelve un código de finalización de MQCC_FAILED, o un código de motivo de MQRC_TRUNCATED_MSG_FAILED.

Especifique la MQGMO_LOCK opción con esta opción para bloquear el mensaje que se está navegando.

Puede especificar MQGMO_BROWSE_FIRST con cualquier combinación válida de las opciones MQGMO_* MQMO_* y que controlan el procesamiento de mensajes en grupos y segmentos de mensajes lógicos.

Si se especifica MQGMO_LOGICAL_ORDER, los mensajes se examinan en orden lógico. Si omite esa opción, los mensajes se examinan en orden físico. Si se especifica MQGMO_BROWSE_FIRST, se puede cambiar entre el orden lógico y el orden físico. Las llamadas MQGET posteriores utilizan la cola MQGMO_BROWSE_NEXT en el mismo orden que la llamada más reciente que se especificó MQGMO_BROWSE_FIRST para el identificador de cola.

El gestor de colas conserva dos conjuntos de información de grupos y segmentos para MQGET las llamadas. La información del grupo y del segmento para las llamadas de exploración se conserva por separado de la información de las llamadas que eliminan mensajes de la cola. Si se especifica MQGMO_BROWSE_FIRST, el gestor de colas ignora la información del grupo y del segmento para la exploración. Escanea la cola como si no hubiera ningún grupo actual ni ningún mensaje lógico actual. Si la MQGET llamada tiene éxito, el código de finalización MQCC_OK o MQCC_WARNING, la información del grupo y del segmento para la exploración se establece en la del mensaje devuelto. Si la llamada falla, la información del grupo y del segmento permanece igual que antes de la llamada.

MQGMO_BROWSE_NEXT

Avance el cursor de exploración al siguiente mensaje de la cola que cumpla los criterios de selección especificados en la MQGET llamada. El mensaje se devuelve a la aplicación, pero permanece en la cola.

MQGMO_BROWSE_NEXT no es válido con ninguna de las siguientes opciones:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_MARK_SKIP_BACKOUT
- MQGMO_MSG_UNDER_CURSOR
- MQGMO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT
- MQGMO_UNLOCK

También es un error si la cola no se abrió para su exploración.

MQGMO_BROWSE_NEXT Se comporta de la misma manera que MQGMO_BROWSE_FIRST, si es la primera llamada para explorar una cola, después de que la cola se haya abierto para su exploración.

El mensaje situado bajo el cursor puede eliminarse de la cola antes de que se emita la siguiente MQGETMQGMO_BROWSE_NEXT llamada con. El cursor de exploración permanece lógicamente en la posición que ocupaba el mensaje en la cola, aunque ahora esa posición esté vacía.

Los mensajes se almacenan en la cola de dos maneras:

- FIFO dentro de la prioridad (MQMDS_PRIORITY), o
- FIFO independientemente de la prioridad (MQMDS_FIFO)

El **MsgDeliverySequence** atributo queue indica qué método se aplica (véase [“Atributos para colas”](#) en la página 841 para más detalles).

Una cola puede tener un MsgDeliverySequence de MQMDS_PRIORITY. Llega un mensaje a la cola que tiene una prioridad mayor que el que señala actualmente el cursor de exploración. En tal caso, el mensaje de mayor prioridad no se encuentra durante el barrido actual de la cola utilizando MQGMO_BROWSE_NEXT. Solo se puede encontrar después de restablecer el cursor de exploración con MQGMO_BROWSE_FIRST, o volviendo a abrir la cola.

La MQGMO_MSG_UNDER_CURSOR opción se puede utilizar con una llamada no MQGET navegable si es necesario, para eliminar el mensaje de la cola.

El cursor de exploración no se mueve por llamadas que no sean MQGET de exploración que utilicen el mismo Hobj controlador.

Especifique la MQGMO_LOCK opción con esta opción para bloquear el mensaje que se está navegando.

Puede especificar MQGMO_BROWSE_NEXT con cualquier combinación válida de las opciones MQGMO_* MQMO_* y que controlan el procesamiento de mensajes en grupos y segmentos de mensajes lógicos.

Si se especifica MQGMO_LOGICAL_ORDER, los mensajes se examinan en orden lógico. Si omite esa opción, los mensajes se examinan en orden físico. Si se especifica MQGMO_BROWSE_FIRST, se puede cambiar entre el orden lógico y el orden físico. Las llamadas MQGET posteriores utilizan la cola MQGMO_BROWSE_NEXT en el mismo orden que la llamada más reciente que se especificó MQGMO_BROWSE_FIRST para el identificador de cola. La llamada falla con el código de motivo MQRC_INCONSISTENT_BROWSE si no se cumple esta condición.

Nota: Ten especial cuidado al utilizar una MQGET llamada para navegar más allá del final de un grupo de mensajes si no MQGMO_LOGICAL_ORDER se especifica. Por ejemplo, supongamos que el último mensaje del grupo precede al primer mensaje del grupo en la cola. Al utilizar MQGMO_BROWSE_NEXT para navegar más allá del final del grupo, especificando MQGMO_MATCH_MSG_SEQ_NUMBER con MsgSeqNumber establecido en 1 se muestra el primer mensaje del grupo ya navegado. Este resultado puede producirse inmediatamente o varias MQGET llamadas más tarde si hay grupos intermedios. La misma consideración se aplica a un mensaje lógico que no está en un grupo.

La información del grupo y del segmento para las llamadas de exploración se conserva por separado de la información de las llamadas que eliminan mensajes de la cola.

MQGMO_BROWSE_MSG_UNDER_CURSOR

Recuperar el mensaje al que apunta el cursor de exploración de forma no destructiva, independientemente de las MQMO_* opciones especificadas en el MatchOptions campo de MQGMO.

MQGMO_BROWSE_MSG_UNDER_CURSOR no es válido con ninguna de las siguientes opciones:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_NEXT
- MQGMO_MARK_SKIP_BACKOUT
- MQGMO_MSG_UNDER_CURSOR
- MQGMO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT
- MQGMO_UNLOCK

También es un error si la cola no se abrió para su exploración.

El mensaje señalado por el cursor de navegación es el último que se ha recuperado utilizando la opción MQGMO_BROWSE_FIRST o la MQGMO_BROWSE_NEXT opción. La llamada falla si ninguna de estas llamadas se ha emitido para esta cola desde que se abrió. La llamada también falla si el mensaje que estaba bajo el cursor de exploración se ha recuperado de forma destructiva.

La posición del cursor de exploración no cambia con esta llamada.

La MQGMO_MSG_UNDER_CURSOR opción se puede utilizar con una llamada no MQGET navegable para eliminar el mensaje de la cola.

El cursor de exploración no se mueve con una llamada que no MQGET sea de exploración, incluso si se utiliza el mismo Hobj identificador. Tampoco se mueve por una llamada de MQGET exploración que devuelve un código de finalización de MQCC_FAILED, o un código de motivo de MQRC_TRUNCATED_MSG_FAILED.

Si MQGMO_BROWSE_MSG_UNDER_CURSOR se especifica con MQGMO_LOCK:

- Si ya hay un mensaje bloqueado, debe ser el que se encuentra debajo del cursor, por lo que se devuelve sin desbloquearlo y volver a bloquearlo. El mensaje permanece bloqueado.
- Si no hay ningún mensaje bloqueado y hay un mensaje debajo del cursor de exploración, se bloquea y se devuelve a la aplicación. Si no hay ningún mensaje debajo del cursor de exploración, la llamada falla.

Si MQGMO_BROWSE_MSG_UNDER_CURSOR se especifica sin MQGMO_LOCK:

- Si ya hay un mensaje bloqueado, debe ser el que se encuentra debajo del cursor. El mensaje se devuelve a la aplicación y, a continuación, se desbloquea. Dado que el mensaje ahora está desbloqueado, no hay garantía de que pueda volver a consultarse o recuperarse de forma

destruccion mediante la misma aplicacion. Es posible que otra aplicacion haya recuperado el mensaje de forma destructiva al obtener mensajes de la cola.

- Si no hay ningun mensaje bloqueado y hay un mensaje debajo del cursor de exploracion, se devuelve a la aplicacion. Si no hay ningun mensaje debajo del cursor de exploracion, la llamada falla.

Si MQGMO_COMPLETE_MSG se especifica con MQGMO_BROWSE_MSG_UNDER_CURSOR, el cursor de exploracion debe identificar un mensaje cuyo Offset campo en MQMD sea cero. Si no se cumple esta condicion, la llamada falla con el codigo de motivo MQRC_INVALID_MSG_UNDER_CURSOR.

La informacion del grupo y del segmento para las llamadas de exploracion se conserva por separado de la informacion de las llamadas que eliminan mensajes de la cola.

MQGMO_MSG_UNDER_CURSOR

Recupera el mensaje al que apunta el cursor de exploracion, independientemente de las MQMO_* opciones especificadas en el MatchOptions campo de MQGMO. El mensaje se elimina de la cola.

El mensaje senalado por el cursor de navegacion es el ultimo que se ha recuperado utilizando la opcion MQGMO_BROWSE_FIRST o la MQGMO_BROWSE_NEXT opcion.

Si MQGMO_COMPLETE_MSG se especifica con MQGMO_MSG_UNDER_CURSOR, el cursor de exploracion debe identificar un mensaje cuyo Offset campo en MQMD sea cero. Si no se cumple esta condicion, la llamada falla con el codigo de motivo MQRC_INVALID_MSG_UNDER_CURSOR.

Esta opcion no es valida con ninguna de las siguientes opciones:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_BROWSE_NEXT
- MQGMO_UNLOCK

Tambien es un error si la cola no se abrio tanto para la exploracion como para la entrada. Si el cursor de exploracion no esta apuntando actualmente a un mensaje recuperable, la MQGET llamada devuelve un error.

MQGMO_MARK_BROWSE_HANDLE

El mensaje que devuelve un MQGET, o identificado por el devuelto MsgToken, se marca. La marca es especifica del identificador de objeto utilizado en la llamada.

El mensaje no se elimina de la cola.

MQGMO_MARK_BROWSE_HANDLE solo es valido si tambien se especifica una de las siguientes opciones:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_BROWSE_NEXT

MQGMO_MARK_BROWSE_HANDLE no es valido con ninguna de las siguientes opciones:

- MQGMO_ALL_MSGS_AVAILABLE
- MQGMO_ALL_SEGMENTS_AVAILABLE
- MQGMO_COMPLETE_MSG
- MQGMO_LOCK
- MQGMO_LOGICAL_ORDER
- MQGMO_UNLOCK

El mensaje permanece en este estado hasta que se produce uno de los siguientes eventos:

- El identificador del objeto en cuestion se cierra, ya sea de forma normal o de otro modo.
- El mensaje no esta marcado para este identificador mediante una llamada a MQGET con la opcion MQGMO_UNMARK_BROWSE_HANDLE.

- El mensaje se devuelve desde una llamada a destructivo MQGET, que se completa con MQCC_OK o MQCC_WARNING. El estado del mensaje permanece cambiado incluso si posteriormente se revierte MQGET el.
- El mensaje caduca.

MQGMO_MARK_BROWSE_CO_OP

El mensaje devuelto por un exitoso MQGET, o identificado por el devuelto *MsgToken*, se marca para todos los manejadores del conjunto cooperativo.

La marca de nivel cooperativo se suma a cualquier marca de nivel de control que se haya establecido.

El mensaje no se elimina de la cola.

MQGMO_MARK_BROWSE_CO_OP solo es válido si el identificador de objeto utilizado fue devuelto por una llamada a MQOPEN el especificado MQOO_CO_OP. También debe especificar una de las siguientes opciones de MQGMO :

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_BROWSE_NEXT

Esta opción no es válida con ninguna de las siguientes opciones:

- MQGMO_ALL_MSGS_AVAILABLE
- MQGMO_ALL_SEGMENTS_AVAILABLE
- MQGMO_COMPLETE_MSG
- MQGMO_LOCK
- MQGMO_LOGICAL_ORDER
- MQGMO_UNLOCK

Si el mensaje ya está marcado y no se especifica MQGMO_UNMARKED_BROWSE_MSG la opción, la llamada falla con MQCC_FAILED y el código de motivo MQRC_MSG_MARKED_BROWSE_CO_OP.

El mensaje permanece en este estado hasta que se produce uno de los siguientes eventos:

- Se cierran todos los identificadores de objetos del conjunto cooperativo.
- El mensaje no está marcado para los navegadores que cooperan mediante una llamada a MQGET con la opción MQGMO_UNMARK_BROWSE_CO_OP.
- El mensaje es desmarcado automáticamente por el gestor de colas.
- El mensaje se devuelve desde una llamada a un elemento que no es de navegación MQGET. El estado del mensaje permanece cambiado incluso si posteriormente se revierte MQGET el.
- El mensaje caduca.

MQGMO_UNMARKED_BROWSE_MSG

Una llamada a MQGET que especifica MQGMO_UNMARKED_BROWSE_MSG devuelve un mensaje que se considera sin marcar para su identificador. No devuelve ningún mensaje si el mensaje fue marcado para su controlador. Tampoco devuelve el mensaje si la cola se abrió mediante una llamada a MQOPEN, con la opción MQOO_CO_OP, y el mensaje ha sido marcado por un miembro del conjunto cooperativo.

Esta opción no es válida con ninguna de las siguientes opciones:

- MQGMO_ALL_MSGS_AVAILABLE
- MQGMO_ALL_SEGMENTS_AVAILABLE
- MQGMO_COMPLETE_MSG
- MQGMO_LOCK
- MQGMO_LOGICAL_ORDER
- MQGMO_UNLOCK

MQGMO_UNMARK_BROWSE_CO_OP

Después de una llamada a MQGET que especifica esta opción, el mensaje ya no es considerado por ningún identificador abierto en el conjunto de identificadores cooperantes que se marcarán para el conjunto cooperante. El mensaje sigue considerándose marcado a nivel de controlador si estaba marcado a nivel de controlador antes de esta llamada.

El uso de solo MQGMO_UNMARK_BROWSE_CO_OP es válido con un identificador devuelto por una llamada correcta a MQOPEN con la opción MQOO_CO_OP. El MQGET tiene éxito incluso si el mensaje no se considera marcado por el conjunto cooperativo de identificadores.

MQGMO_UNMARK_BROWSE_CO_OP no es válido en una llamada sin MQGET navegación ni con ninguna de las siguientes opciones:

- MQGMO_ALL_MSGS_AVAILABLE
- MQGMO_ALL_SEGMENTS_AVAILABLE
- MQGMO_COMPLETE_MSG
- MQGMO_LOCK
- MQGMO_LOGICAL_ORDER
- MQGMO_MARK_BROWSE_CO_OP
- MQGMO_UNLOCK
- MQGMO_UNMARKED_BROWSE_MSG

MQGMO_UNMARK_BROWSE_HANDLE

Después de una llamada a MQGET que especifique esta opción, el mensaje localizado ya no se considera marcado por este identificador.

La llamada se realiza correctamente aunque el mensaje no esté marcado para este identificador.

Esta opción no es válida en una llamada que no MQGET sea de exploración, ni con ninguna de las siguientes opciones:

- MQGMO_ALL_MSGS_AVAILABLE
- MQGMO_ALL_SEGMENTS_AVAILABLE
- MQGMO_COMPLETE_MSG
- MQGMO_LOCK
- MQGMO_LOGICAL_ORDER
- MQGMO_MARK_BROWSE_CO_OP
- MQGMO_UNLOCK
- MQGMO_UNMARKED_BROWSE_MSG

Opciones de bloqueo

Las siguientes opciones se refieren al bloqueo de mensajes en la cola:

MQGMO_LOCK

Bloquea el mensaje que se está explorando, de modo que el mensaje se vuelva invisible para cualquier otro identificador abierto para la cola. La opción solo se puede especificar si también se especifica una de las siguientes opciones:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_NEXT
- MQGMO_BROWSE_MSG_UNDER_CURSOR

Solo se puede bloquear un mensaje por cada identificador de cola. El mensaje puede ser un mensaje lógico o un mensaje físico:

- Si se especifica MQGMO_COMPLETE_MSG, todos los segmentos de mensaje que componen el mensaje lógico se bloquean en el identificador de cola. Todos los mensajes deben estar presentes en la cola y disponibles para su recuperación.

- Si omite MQGMO_COMPLETE_MSG, solo se bloqueará un único mensaje físico en el identificador de la cola. Si este mensaje es un segmento de un mensaje lógico, el segmento bloqueado impide que otras aplicaciones utilicen MQGMO_COMPLETE_MSG para recuperar o examinar el mensaje lógico.

El mensaje bloqueado es siempre el que se encuentra debajo del cursor de navegación. El mensaje se puede eliminar de la cola mediante una llamada MQGET posterior que especifique la MQGMO_MSG_UNDER_CURSOR opción. Otras MQGET llamadas que utilicen el identificador de cola también pueden eliminar el mensaje (por ejemplo, una llamada que especifique el identificador del mensaje bloqueado).

Si la llamada devuelve el código de MQCC_FAILED finalización, o MQCC_WARNING con el código de motivo MQRC_TRUNCATED_MSG_FAILED, no se bloquea ningún mensaje.

Si la aplicación no elimina el mensaje de la cola, el bloqueo se libera mediante una de las siguientes acciones:

- Emitir otra MQGET llamada para este identificador, especificando MQGMO_BROWSE_FIRST o MQGMO_BROWSE_NEXT. El bloqueo se libera si la llamada finaliza con MQCC_OK o MQCC_WARNING. El mensaje permanece bloqueado si la llamada finaliza con MQCC_FAILED. No obstante, se aplican las excepciones siguientes:

- El mensaje no se desbloquea si MQCC_WARNING se devuelve con MQRC_TRUNCATED_MSG_FAILED.
- El mensaje se desbloquea si MQCC_FAILED se devuelve con MQRC_NO_MSG_AVAILABLE.

Si también se especifica MQGMO_LOCK, el mensaje devuelto está bloqueado. Si omite MQGMO_LOCK, no habrá ningún mensaje bloqueado después de la llamada.

Si se especifica MQGMO_WAITy no hay ningún mensaje disponible inmediatamente, el mensaje original se desbloquea antes del inicio de la espera.

- Emitiendo otra MQGET llamada para este identificador, con MQGMO_BROWSE_MSG_UNDER_CURSOR, sin MQGMO_LOCK. El bloqueo se libera si la llamada finaliza con MQCC_OK o MQCC_WARNING. El mensaje permanece bloqueado si la llamada finaliza con MQCC_FAILED. Sin embargo, se aplica la siguiente excepción:
 - El mensaje no se desbloquea si MQCC_WARNING se devuelve con MQRC_TRUNCATED_MSG_FAILED.
- Emitiendo otra MQGET llamada para este identificador con MQGMO_UNLOCK.
- Emisión de una MQCLOSE llamada utilizando el identificador. El MQCLOSE puede ser implícito, causado por el final de la aplicación.

No se requiere ninguna MQOPEN opción especial para especificar MQGMO_LOCK, salvo MQOO_BROWSE, que es necesaria para especificar una opción de exploración acompañante.

MQGMO_LOCK no es válido con ninguna de las siguientes opciones:

- MQGMO_MARK_SKIP_BACKOUT
- MQGMO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT
- MQGMO_UNLOCK

MQGMO_UNLOCK

El mensaje que se va a desbloquear debe haber sido bloqueado previamente mediante una MQGET llamada con la MQGMO_LOCK opción. Si no hay ningún mensaje bloqueado para este identificador, la llamada finaliza con MQCC_WARNING y MQRC_NO_MSG_LOCKED.

Los parámetros **MsgDesc**, **BufferLength** **Buffer**, **DataLength** y no se comprueban ni modifican si se especifica MQGMO_UNLOCK. No se devuelve ningún mensaje en *Buffer*.

No se requiere ninguna opción especial de apertura para especificar MQGMO_UNLOCK (aunque MQOO_BROWSE es necesario para emitir la solicitud de bloqueo en primer lugar).

Esta opción no es válida con ninguna otra opción, excepto las siguientes:

- MQGMO_NO_WAIT
- MQGMO_NO_SYNCPOINT

Ambas opciones se asumen independientemente de si se especifican o no.

Opciones de datos de mensajes

Las siguientes opciones se refieren al procesamiento de los datos del mensaje cuando este se lee desde la cola:

MQGMO_ACCEPT_TRUNCATED_MSG

Si el búfer de mensajes es demasiado pequeño para contener el mensaje completo, permita que MQGET la llamada llene el búfer. MQGET llena el búfer con la mayor parte del mensaje que puede. Emite un código de finalización de advertencia y completa su procesamiento. Esto significa que:

- Al navegar por los mensajes, el cursor de navegación avanza hasta el mensaje devuelto.
- Al eliminar mensajes, el mensaje devuelto se elimina de la cola.
- Se devuelve el código de motivo MQRC_TRUNCATED_MSG_ACCEPTED si no se produce ningún otro error.

Sin esta opción, el búfer se sigue llenando con todo el mensaje que pueda contener. Se emite un código de finalización de advertencia, pero el procesamiento no se completa. Esto significa que:

- Al navegar por los mensajes, el cursor de navegación no avanza.
- Al eliminar mensajes, el mensaje no se elimina de la cola.
- Se devuelve MQRC_TRUNCATED_MSG_FAILED el código de motivo si no se produce ningún otro error.

MQGMO_CONVERT

Esta opción convierte los datos de la aplicación en el mensaje para que se ajusten a los CodedCharSetId valores Encoding y especificados en el **MsgDesc** parámetro de la MQGET llamada. Los datos se convierten antes de copiarlos al **Buffer** parámetro.

El Format campo especificado cuando se envió el mensaje es asumido por el proceso de conversión para identificar la naturaleza de los datos del mensaje. Los datos del mensaje son convertidos por el gestor de colas para los formatos integrados y por una salida escrita por el usuario para otros formatos. Consulte [“salida de conversión de datos” en la página 915](#) para obtener más información sobre la salida de conversión de datos.

- Si la conversión se realiza correctamente, los campos CodedCharSetId Encoding y especificados en el **MsgDesc** parámetro no se modifican al regresar de la MQGET llamada.
- Si solo falla la conversión, los datos del mensaje se devuelven sin convertir. Los campos CodedCharSetId Encoding y en MsgDesc se establecen en los valores del mensaje sin convertir. El código de finalización es MQCC_WARNING en este caso.

En cualquier caso, estos campos describen el identificador del conjunto de caracteres y la codificación de los datos del mensaje que se devuelven en el **Buffer** parámetro.

Consulte el *Format* campo descrito en [“MQMD - Descriptor de mensaje” en la página 389](#) para obtener una lista de los nombres de formato para los que el gestor de colas realiza la conversión.

Opciones de grupo y segmento

Las siguientes opciones se refieren al procesamiento de mensajes en grupos y segmentos de mensajes lógicos. Antes de describir las opciones, aquí hay algunas definiciones de términos importantes:

Mensaje físico

Un mensaje físico es la unidad más pequeña de información que se puede colocar en una cola o eliminar de ella. A menudo corresponde a la información especificada o recuperada en una sola llamada MQPUT, MQPUT1MQGET , o. Cada mensaje físico tiene su propio descriptor de mensaje, MQMD. Normalmente, los mensajes físicos se distinguen por valores diferentes para el identificador de mensaje, el MsgId campo en MQMD. El gestor de colas no aplica valores diferentes.

Mensaje lógico

Un mensaje lógico es una unidad única de información de aplicación. En ausencia de restricciones del sistema, un mensaje lógico es lo mismo que un mensaje físico. Si los mensajes lógicos son grandes, las limitaciones del sistema pueden hacer que sea aconsejable o necesario dividir un mensaje lógico en dos o más mensajes físicos, denominados segmentos.

Un mensaje lógico que ha sido segmentado consta de dos o más mensajes físicos que tienen el mismo identificador de grupo no nulo, `GroupId` campo en MQMD. Tienen el mismo número de secuencia de mensajes, `MsgSeqNumber` campo en MQMD. Los segmentos se distinguen por valores diferentes para el campo de `Offset` desplazamiento del segmento en MQMD. El desplazamiento del segmento es el desplazamiento de los datos en el mensaje físico desde el inicio de los datos en el mensaje lógico. Dado que cada segmento es un mensaje físico, los segmentos de un mensaje lógico suelen tener identificadores de mensaje diferentes.

Un mensaje lógico que no ha sido segmentado, pero para el que la aplicación emisora ha permitido la segmentación, también tiene un identificador de grupo distinto de nulo. En este caso, solo hay un mensaje físico con ese identificador de grupo si el mensaje lógico no pertenece a un grupo de mensajes. Los mensajes lógicos, cuya segmentación ha sido inhibida por la aplicación emisora, tienen un identificador de grupo nulo, `MQGI_NONE`, a menos que el mensaje lógico pertenezca a un grupo de mensajes.

Grupo de mensajes

Un grupo de mensajes es un conjunto de uno o más mensajes lógicos que tienen el mismo identificador de grupo no nulo. Los mensajes lógicos del grupo se distinguen por diferentes valores del número de secuencia del mensaje. El número de secuencia es un número entero comprendido entre 1 y n, donde n es el número de mensajes lógicos del grupo. Si uno o más de los mensajes lógicos están segmentados, hay más de n mensajes físicos en el grupo.

MQGMO_LOGICAL_ORDER

`MQGMO_LOGICAL_ORDER` controla el orden en que se devuelven los mensajes por llamadas `MQGET` sucesivas para el identificador de la cola. La opción debe especificarse en cada llamada.

Si `MQGMO_LOGICAL_ORDER` se especifica para llamadas `MQGET` sucesivas para el mismo identificador de cola, los mensajes en grupos se devuelven en el orden de sus números de secuencia de mensajes. Los segmentos de mensajes lógicos se devuelven en el orden indicado por sus desplazamientos de segmento. Este orden puede diferir del orden en que esos mensajes y segmentos aparecen en la cola.

Nota: Especificar no `MQGMO_LOGICAL_ORDER` tiene consecuencias adversas en los mensajes que no pertenecen a grupos y que no son segmentos. En efecto, dichos mensajes se tratan como si cada uno perteneciera a un grupo de mensajes compuesto por un solo mensaje. Es seguro especificar `MQGMO_LOGICAL_ORDER` al recuperar mensajes de colas que contienen una mezcla de mensajes en grupos, segmentos de mensajes y mensajes no segmentados que no están en grupos.

Para devolver los mensajes en el orden requerido, el gestor de colas conserva la información de grupo y segmento entre llamadas `MQGET` sucesivas. La información de grupo y segmento identifica el grupo de mensajes actual y el mensaje lógico actual para el identificador de cola. También identifica la posición actual dentro del grupo y el mensaje lógico, y si los mensajes se están recuperando dentro de una unidad de trabajo. Dado que el gestor de colas conserva esta información, la aplicación no necesita establecer la información de grupo y segmento antes de cada `MQGET` llamada. En concreto, esto significa que la aplicación no necesita configurar los campos `GroupId`, `MsgSeqNumberOffset` y en MQMD. Sin embargo, la aplicación debe establecer correctamente la opción `MQGMO_SYNCPOINT` o `MQGMO_NO_SYNCPOINT` en cada llamada.

Cuando se abre la cola, no hay ningún grupo de mensajes actual ni ningún mensaje lógico actual. Un grupo de mensajes se convierte en el grupo de mensajes actual cuando la `MQGET` llamada devuelve un mensaje que tiene la `MQMF_MSG_IN_GROUP` bandera. Con `MQGMO_LOGICAL_ORDER` especificado en llamadas sucesivas, ese grupo permanece como el grupo actual hasta que se devuelve un mensaje que tiene:

- `MQMF_LAST_MSG_IN_GROUP` sin `MQMF_SEGMENT` (es decir, el último mensaje lógico del grupo no está segmentado), o

- MQMF_LAST_MSG_IN_GROUP con MQMF_LAST_SEGMENT (es decir, el mensaje devuelto es el último segmento del último mensaje lógico del grupo).

Cuando se devuelve un mensaje de este tipo, el grupo de mensajes se termina y, una vez completada con éxito la MQGET llamada, ya no hay ningún grupo activo. De manera similar, un mensaje lógico se convierte en el mensaje lógico actual cuando un mensaje que tiene la MQMF_SEGMENT bandera es devuelto por la MQGET llamada. El mensaje lógico finaliza cuando se devuelve el mensaje que tiene la MQMF_LAST_SEGMENT bandera.

Si no se especifican criterios de selección, las MQGET llamadas sucesivas devuelven, en el orden correcto, los mensajes del primer grupo de mensajes de la cola. A continuación, devuelven los mensajes del segundo grupo de mensajes, y así sucesivamente, hasta que no quedan más mensajes disponibles. Es posible seleccionar los grupos de mensajes concretos que se devuelven especificando una o varias de las siguientes opciones en el MatchOptions campo:

- MQMO_MATCH_MSG_ID
- MQMO_MATCH_CORREL_ID
- MQMO_MATCH_GROUP_ID

Sin embargo, estas opciones solo son efectivas cuando no hay ningún grupo de mensajes actual ni ningún mensaje lógico. Consulte el MatchOptions campo descrito en [“MQGMO - Opciones para obtener mensajes”](#) en la página 331 para obtener más detalles.

Tabla 510 en la página 358 muestra los valores de los campos MsgId, CorrelId, GroupIdOffset MsgSeqNumber, y que el gestor de colas busca cuando intenta encontrar un mensaje para devolver en la MQGET llamada. Las reglas se aplican tanto a la eliminación de mensajes de la cola como a la exploración de mensajes en la cola. En la tabla, «Either» significa «Sí» o «No»:

LOG ORD

Indica si la MQGMO_LOGICAL_ORDER opción se ha especificado en la llamada.

Cur grp

Indica si existe un grupo de mensajes actual antes de la llamada.

Cur log msg

Indica si existe un mensaje lógico actual antes de la llamada.

Otras columnas

Mostrar los valores que busca el gestor de colas. Previous indica el valor devuelto para el campo en el mensaje anterior para el identificador de cola.

Tabla 510. Opciones de MQGET relacionadas con mensajes en grupos y segmentos de mensajes lógicos							
Opciones especificadas	Estado de grupo y mensaje lógico antes de la llamada		Valores que busca el gestor de colas				
	Cur grp	Cur log msg	MsgId	CorrelId	GroupId	MsgSeqNumber	Offset
Sí	No	No	Controlado por MatchOptions	Controlado por MatchOptions	Controlado por MatchOptions	1	0
Sí	No	Sí	Cualquier identificador de mensaje	Cualquier identificador de correlación	Identificador de grupo anterior	1	Desplazamiento anterior + longitud de segmento anterior

Tabla 510. Opciones de MQGET relacionadas con mensajes en grupos y segmentos de mensajes lógicos (continuación)

Opciones especificadas	Estado de grupo y mensaje lógico antes de la llamada		Valores que busca el gestor de colas				
	Sí	No	Cualquier identificador de mensaje	Cualquier identificador de correlación	Identificador de grupo anterior	Número de secuencia anterior + 1	0
Sí	Sí	Sí	Cualquier identificador de mensaje	Cualquier identificador de correlación	Identificador de grupo anterior	Número de secuencia anterior	Desplazamiento anterior + longitud de segmento anterior
No	Cualquiera de los dos	Cualquiera de los dos	Controlado por <i>MatchOptions</i>	Controlado por <i>MatchOptions</i>	Controlado por <i>MatchOptions</i>	Controlado por <i>MatchOptions</i>	Controlado por <i>MatchOptions</i>

Si hay varios grupos de mensajes en la cola y son aptos para su devolución, los grupos se devuelven en el orden determinado por la posición en la cola del primer segmento del primer mensaje lógico de cada grupo. Es decir, los mensajes físicos que tienen números de secuencia de mensajes 1 y desplazamientos 0 determinan el orden en que se devuelven los grupos elegibles.

La MQGMO_LOGICAL_ORDER opción afecta a las unidades de trabajo de la siguiente manera:

- Si el primer mensaje lógico o segmento de un grupo se recupera dentro de una unidad de trabajo, todos los demás mensajes lógicos y segmentos del grupo deben recuperarse dentro de una unidad de trabajo, si se utiliza el mismo identificador de cola. Sin embargo, no es necesario recuperarlos dentro de la misma unidad de trabajo. Esto permite dividir un grupo de mensajes compuesto por muchos mensajes físicos en dos o más unidades de trabajo consecutivas para el controlador de cola.
- Si el primer mensaje lógico o segmento de un grupo no se recupera dentro de una unidad de trabajo y se utiliza el mismo identificador de cola, ninguno de los demás mensajes lógicos y segmentos del grupo podrá recuperarse dentro de una unidad de trabajo.

Si no se cumplen estas condiciones, la MQGET llamada falla con el código de motivo MQRC_INCONSISTENT_UOW.

Cuando MQGMO_LOGICAL_ORDER se especifica, el MQGMO suministrado en la MQGET llamada no debe ser inferior a MQGMO_VERSION_2, y el MQMD no debe ser inferior a MQMD_VERSION_2. Si no se cumple esta condición, la llamada falla con el código de motivo MQRC_WRONG_GMO_VERSION o MQRC_WRONG_MD_VERSION, según corresponda.

Si no MQGMO_LOGICAL_ORDER se especifica para llamadas MQGET sucesivas para el identificador de cola, los mensajes se devuelven sin tener en cuenta si pertenecen a grupos de mensajes o si son segmentos de mensajes lógicos. Esto significa que los mensajes o segmentos de un grupo concreto o mensaje lógico pueden devolverse desordenados o mezclados con mensajes o segmentos de otros grupos o mensajes lógicos, o con mensajes que no pertenecen a ningún grupo ni son segmentos. En esta situación, los mensajes concretos que devuelven las MQGET llamadas sucesivas están controlados por las MQMO_* opciones especificadas en dichas llamadas (consulte el *MatchOptions* campo descrito en “MQGMO - Opciones para obtener mensajes” en la página 331 para obtener más información sobre estas opciones).

Esta es la técnica que se puede utilizar para reiniciar un grupo de mensajes o un mensaje lógico a mitad de camino, después de que se haya producido un fallo del sistema. Cuando el sistema se reinicia, la aplicación puede establecer los campos GroupId, MsgSeqNumberMatchOptions, Offset y con los valores adecuados y, a continuación, emitir la

MQGET llamada con MQGMO_SYNCPOINT o MQGMO_NO_SYNCPOINT establecidos, pero sin especificar MQGMO_LOGICAL_ORDER. Si esta llamada tiene éxito, el gestor de colas conserva la información del grupo y del segmento, y las llamadas MQGET posteriores que utilicen ese identificador de cola pueden especificar MQGMO_LOGICAL_ORDER como de costumbre.

La información de grupo y segmento que el gestor de colas conserva para la MQGET llamada es independiente de la información de grupo y segmento que conserva para la llamada MQPUT. Además, el gestor de colas conserva información separada para:

- MQGET llamadas que eliminan mensajes de la cola.
- MQGET llamadas que examinan los mensajes de la cola.

Para cualquier identificador de cola dado, la aplicación puede mezclar MQGET llamadas que especifican MQGMO_LOGICAL_ORDER con MQGET llamadas que no lo hacen. Sin embargo, tenga en cuenta los siguientes puntos:

- Si se omite MQGMO_LOGICAL_ORDER, cada llamada MQGET exitosa hace que el gestor de colas establezca la información guardada del grupo y del segmento con los valores correspondientes al mensaje devuelto; esto sustituye la información existente del grupo y del segmento conservada por el gestor de colas para el identificador de cola. Solo se modifica la información adecuada a la acción de la llamada (navegar o eliminar).
- Si se omite MQGMO_LOGICAL_ORDER, la llamada no falla si hay un grupo de mensajes actual o un mensaje lógico; la llamada podría tener éxito con un código MQCC_WARNING de finalización. La [Tabla 511 en la página 360](#) muestra los distintos casos que se pueden presentar. En estos casos, si el código de finalización no es MQCC_OK, el código de motivo es uno de los siguientes (según corresponda):
 - MQRC_INCOMPLETE_GROUP
 - MQRC_INCOMPLETE_MSG
 - MQRC_INCONSISTENT_UOW

Nota: El gestor de colas no comprueba la información del grupo y del segmento al examinar una cola, ni al cerrar una cola que se abrió para examinarla pero no para introducir datos; en esos casos, el código de finalización es siempre MQCC_OK (suponiendo que no haya otros errores).

<i>Tabla 511. Resultado cuando la llamada MQGET o MQCLOSE no es coherente con la información del grupo y del segmento</i>		
La llamada actual es	La llamada anterior fue MQGET con MQGMO_LOGICAL_ORDER	La llamada anterior fue MQGET sin MQGMO_LOGICAL_ORDER
MQGET con MQGMO_LOGICAL_ORDER	MQCC_FAILED	MQCC_FAILED
MQGET sin MQGMO_LOGICAL_ORDER	MQCC_WARNING	MQCC_OK
MQCLOSE con un grupo o mensaje lógico sin terminar	MQCC_WARNING	MQCC_OK

Se recomienda a las aplicaciones que deseen recuperar mensajes y segmentos en orden lógico que especifiquen MQGMO_LOGICAL_ORDER, ya que es la opción más sencilla de usar. Esta opción hace que la aplicación no tenga que gestionar la información de grupo y segmento, pues el gestor de colas lo hace en su lugar. Sin embargo, las aplicaciones especializadas pueden necesitar más control que el que proporciona la MQGMO_LOGICAL_ORDER opción, y esto se puede conseguir no especificando dicha opción. A continuación, la aplicación debe asegurarse de que los campos MsgId, CorrelId, GroupIdOffset, MsgSeqNumber, y en MQMD, y las MQMO_* opciones en MatchOptions en MQGMO, estén configurados correctamente antes de cada MQGET llamada.

Por ejemplo, una aplicación que desee reenviar los mensajes físicos que recibe, independientemente de si dichos mensajes se encuentran en grupos o segmentos de mensajes lógicos, no debe especificar

MQGMO_LOGICAL_ORDER. En una red compleja con múltiples rutas entre los gestores de colas de emisión y recepción, los mensajes físicos pueden llegar fuera de secuencia. Al no especificar ni MQGMO_LOGICAL_ORDER, ni el correspondiente MQPMO_LOGICAL_ORDER en la llamada MQPUT, la aplicación de reenvío puede recuperar y reenviar cada mensaje físico tan pronto como llega, sin tener que esperar a que llegue el siguiente en orden lógico.

Puede especificar MQGMO_LOGICAL_ORDER con cualquiera de las otras MQGMO_* opciones, y con varias de las MQMO_* opciones en circunstancias apropiadas (véase la sección anterior).

-  En z/OS, esta opción es compatible con colas privadas y compartidas, pero la cola debe tener un tipo de índice MQIT_GROUP_ID. Para las colas compartidas, el objeto CFSTRUCT al que se asigna la cola debe estar en CFLEVEL(3) o superior.
- Esta opción es compatible con todas las colas locales para las siguientes plataformas:

-  AIX
-  Linux
-  IBM i
-  Windows

y para IBM MQ MQI clients conectados a estos sistemas.

MQGMO_COMPLETE_MSG

La MQGET llamada solo puede devolver un mensaje lógico completo. Si el mensaje lógico está segmentado, el gestor de colas vuelve a ensamblar los segmentos y devuelve el mensaje lógico completo a la aplicación; el hecho de que el mensaje lógico estuviera segmentado no es evidente para la aplicación que lo recupera.

Nota: Esta es la única opción que hace que el gestor de colas vuelva a ensamblar los segmentos de mensajes. Si no se especifica, los segmentos se devuelven individualmente a la aplicación si están presentes en la cola (y cumplen los demás criterios de selección especificados en la MQGET llamada). Las aplicaciones que no deseen recibir segmentos individuales deben especificarlo siempre MQGMO_COMPLETE_MSG.

Para utilizar esta opción, la aplicación debe proporcionar un búfer lo suficientemente grande como para alojar el mensaje completo, o especificar la MQGMO_ACCEPT_TRUNCATED_MSG opción.

Si la cola contiene mensajes segmentados en los que faltan algunos segmentos (quizás porque se han retrasado en la red y aún no han llegado), especificar MQGMO_COMPLETE_MSG impide la recuperación de segmentos que pertenecen a mensajes lógicos incompletos. Sin embargo, esos segmentos de mensaje siguen contribuyendo al valor del atributo **CurrentQDepth** queue; esto significa que puede que no haya mensajes lógicos recuperables, aunque *CurrentQDepth* sea mayor que cero.

En el caso de los mensajes persistentes, el gestor de colas solo puede reensamblar los segmentos dentro de una unidad de trabajo:

- Si la MQGET llamada se está ejecutando dentro de una unidad de trabajo definida por el usuario, se utiliza esa unidad de trabajo. Si la llamada falla durante el proceso de reensamblaje, el gestor de colas restablece en la cola cualquier segmento que se haya eliminado durante el reensamblaje. Sin embargo, el fallo no impide que la unidad de trabajo se confirme correctamente.
- Si la llamada se realiza fuera de una unidad de trabajo definida por el usuario y no existe ninguna unidad de trabajo definida por el usuario, el gestor de colas crea una unidad de trabajo para la duración de la llamada. Si la llamada se realiza correctamente, el gestor de colas confirma automáticamente la unidad de trabajo (la aplicación no necesita hacerlo). Si la llamada falla, el gestor de colas revierte la unidad de trabajo.
- Si la llamada se ejecuta fuera de una unidad de trabajo definida por el usuario, pero existe una unidad de trabajo definida por el usuario, el gestor de colas no puede volver a ensamblar. Si el mensaje no requiere reensamblaje, la llamada aún puede tener éxito. Pero si el mensaje requiere reensamblaje, la llamada falla con el código de motivo MQRC_UOW_NOT_AVAILABLE.

Para los mensajes no persistentes, el gestor de colas no requiere que haya una unidad de trabajo disponible para realizar el reensamblaje.

Cada mensaje físico que es un segmento tiene su propio descriptor de mensaje. Para los segmentos que constituyen un único mensaje lógico, la mayoría de los campos del descriptor del mensaje son los mismos para todos los segmentos del mensaje lógico; normalmente, solo `MsgId` los campos, `Offset` `MsgFlags` y difieren entre los segmentos del mensaje lógico. Sin embargo, si un segmento se coloca en una cola de mensajes no entregados en un gestor de colas intermedio, el controlador DLQ recupera el mensaje especificando la `MQGMO_CONVERT` opción, lo que puede provocar que se modifique el juego de caracteres o la codificación del segmento. Si el controlador DLQ envía correctamente el segmento, es posible que este tenga un juego de caracteres o una codificación diferentes a los de los demás segmentos del mensaje lógico cuando llegue al gestor de colas de destino.

Un mensaje lógico compuesto por segmentos en los que los campos `CodedCharSetId` `Encoding` y difieren no puede ser reensamblado por el gestor de colas en un único mensaje lógico. En su lugar, el gestor de colas vuelve a ensamblar y devuelve los primeros segmentos consecutivos al principio del mensaje lógico que tienen los mismos identificadores de juego de caracteres y codificaciones, y la `MQGET` llamada se completa con el código de finalización `MQCC_WARNING` y el código de motivo `MQRC_INCONSISTENT_CCIDS` o `MQRC_INCONSISTENT_ENCODINGS`, según corresponda. Esto ocurre independientemente de si `MQGMO_CONVERT` está especificado. Para recuperar los segmentos restantes, la aplicación debe volver a emitir la `MQGET` llamada sin la `MQGMO_COMPLETE_MSG` opción, recuperando los segmentos uno por uno. `MQGMO_LOGICAL_ORDER` se puede utilizar para recuperar los segmentos restantes en orden.

Una aplicación que coloca segmentos también puede establecer otros campos en el descriptor del mensaje con valores que difieren entre segmentos. Sin embargo, no hay ninguna ventaja en hacerlo si la aplicación receptora utiliza `MQGMO_COMPLETE_MSG` para recuperar el mensaje lógico. Cuando el gestor de colas vuelve a ensamblar un mensaje lógico, devuelve en el descriptor del mensaje los valores del descriptor del mensaje del primer segmento; la única excepción es el `MsgFlags` campo, que el gestor de colas establece para indicar que el mensaje reensamblado es el único segmento.

Si `MQGMO_COMPLETE_MSG` se especifica para un mensaje de informe, el gestor de colas realiza un procesamiento especial. El gestor de colas comprueba la cola para ver si todos los mensajes de informe de ese tipo de informe relacionados con los diferentes segmentos del mensaje lógico están presentes en la cola. Si es así, se pueden recuperar como un único mensaje especificando `MQGMO_COMPLETE_MSG`. Para que esto sea posible, los mensajes de informe deben ser generados por un gestor de colas o MCA que admita segmentación, o bien la aplicación de origen debe solicitar al menos 100 bytes de datos de mensaje (es decir, deben especificarse las opciones `MQRO_*_WITH_FULL_DATA` o `MQRO_*_WITH_DATA` adecuadas). Si no se dispone de la totalidad de los datos de la solicitud para un segmento, los bytes que faltan se sustituyen por valores nulos en el mensaje de informe devuelto.

Si `MQGMO_COMPLETE_MSG` se especifica con `MQGMO_MSG_UNDER_CURSOR` o `MQGMO_BROWSE_MSG_UNDER_CURSOR`, el cursor de exploración debe colocarse en un mensaje cuyo `Offset` campo en `MQMD` tenga un valor de 0. Si no se cumple esta condición, la llamada falla con el código de motivo `MQRC_INVALID_MSG_UNDER_CURSOR`.

`MQGMO_COMPLETE_MSG` implica `MQGMO_ALL_SEGMENTS_AVAILABLE`, por lo que no es necesario especificarlo.

`MQGMO_COMPLETE_MSG` Se puede especificar con cualquiera de las otras `MQGMO_*` opciones, excepto `MQGMO_SYNCPOINT_IF_PERSISTENT`, y con cualquiera de las `MQMO_*` opciones, excepto `MQMO_MATCH_OFFSET`.

-  En z/OS, esta opción es compatible con colas privadas y compartidas, pero la cola debe tener un tipo de índice `MQIT_GROUP_ID`. Para las colas compartidas, el objeto `CFSTRUCT` al que se asigna la cola debe estar en `CFLEVEL(3)` o superior.

- En las siguientes plataformas:

–  AIX

-  IBM i
-  Linux
-  Windows

y para IBM MQ MQI clients conectados a estos sistemas, esta opción es compatible con todas las colas locales.

MQGMO_ALL_MSGS_AVAILABLE

Los mensajes de un grupo solo están disponibles para su recuperación cuando todos los mensajes del grupo están disponibles. Si la cola contiene grupos de mensajes en los que faltan algunos mensajes (quizás porque se han retrasado en la red y aún no han llegado), especificar MQGMO_ALL_MSGS_AVAILABLE impide la recuperación de mensajes que pertenecen a grupos incompletos. Sin embargo, esos mensajes siguen contribuyendo al valor del atributo **CurrentQDepth** queue; esto significa que puede que no haya grupos de mensajes recuperables, aunque CurrentQDepth sea mayor que cero. Si no hay otros mensajes recuperables, se devuelve MQRC_NO_MSG_AVAILABLE el código de motivo una vez transcurrido el intervalo de espera especificado (si lo hay).

El procesamiento de MQGMO_ALL_MSGS_AVAILABLE depende de si también MQGMO_LOGICAL_ORDER se especifica:

- Si se especifican ambas opciones, solo MQGMO_ALL_MSGS_AVAILABLE tiene efecto cuando no hay ningún grupo actual ni mensaje lógico. Si hay un grupo actual o un mensaje lógico, MQGMO_ALL_MSGS_AVAILABLE se ignora. Esto significa que MQGMO_ALL_MSGS_AVAILABLE puede permanecer activado al procesar mensajes en orden lógico.
- Si MQGMO_ALL_MSGS_AVAILABLE se especifica sin MQGMO_LOGICAL_ORDER, MQGMO_ALL_MSGS_AVAILABLE siempre tiene efecto. Esto significa que la opción debe desactivarse después de que el primer mensaje del grupo se haya eliminado de la cola, para poder eliminar los mensajes restantes del grupo.

La finalización satisfactoria de una MQGET llamada que especifica MQGMO_ALL_MSGS_AVAILABLE significa que, en el momento en que se realizó la MQGET llamada, todos los mensajes del grupo estaban en la cola. Sin embargo, ten en cuenta que otras aplicaciones pueden seguir eliminando mensajes del grupo (el grupo no está bloqueado para la aplicación que recupera el primer mensaje del grupo).

Si omite esta opción, los mensajes pertenecientes a grupos se pueden recuperar incluso cuando el grupo está incompleto.

MQGMO_ALL_MSGS_AVAILABLE implica MQGMO_ALL_SEGMENTS_AVAILABLE, por lo que no es necesario especificarlo.

MQGMO_ALL_MSGS_AVAILABLE Se puede especificar con cualquiera de las otras MQGMO_* opciones y con cualquiera de las MQMO_* opciones.

-  En z/OS, esta opción es compatible con colas privadas y compartidas, pero la cola debe tener un tipo de índice MQIT_GROUP_ID. Para las colas compartidas, el objeto CFSTRUCT al que se asigna la cola debe estar en CFLEVEL(3) o superior.
- En las siguientes plataformas:

-  AIX
-  IBM i
-  Linux
-  Windows

y para IBM MQ MQI clients conectados a estos sistemas, esta opción es compatible con todas las colas locales.

MQGMO_ALL_SEGMENTS_AVAILABLE

Los segmentos de un mensaje lógico solo están disponibles para su recuperación cuando todos los segmentos del mensaje lógico están disponibles. Si la cola contiene mensajes segmentados en los que faltan algunos segmentos (quizás porque se han retrasado en la red y aún no han llegado), especificar MQGMO_ALL_SEGMENTS_AVAILABLE impide la recuperación de segmentos que pertenecen a mensajes lógicos incompletos. Sin embargo, esos segmentos siguen contribuyendo al valor del atributo **CurrentQDepth** de cola; esto significa que puede que no haya mensajes lógicos recuperables, aunque CurrentQDepth sea mayor que cero. Si no hay otros mensajes recuperables, se devuelve MQRC_NO_MSG_AVAILABLE el código de motivo una vez transcurrido el intervalo de espera especificado (si lo hay).

El procesamiento de MQGMO_ALL_SEGMENTS_AVAILABLE depende de si también MQGMO_LOGICAL_ORDER se especifica:

- Si se especifican ambas opciones, solo MQGMO_ALL_SEGMENTS_AVAILABLE tiene efecto cuando no hay ningún mensaje lógico actual. Si hay un mensaje lógico actual, MQGMO_ALL_SEGMENTS_AVAILABLE se ignora. Esto significa que MQGMO_ALL_SEGMENTS_AVAILABLE puede permanecer activado al procesar mensajes en orden lógico.
- Si MQGMO_ALL_SEGMENTS_AVAILABLE se especifica sin MQGMO_LOGICAL_ORDER, MQGMO_ALL_SEGMENTS_AVAILABLE siempre tiene efecto. Esto significa que la opción debe desactivarse después de que el primer segmento del mensaje lógico se haya eliminado de la cola, para poder eliminar los segmentos restantes del mensaje lógico.

Si no se especifica esta opción, los segmentos del mensaje se pueden recuperar incluso cuando el mensaje lógico está incompleto.

Si bien tanto MQGMO_COMPLETE_MSG como MQGMO_ALL_SEGMENTS_AVAILABLE requieren que todos los segmentos estén disponibles antes de que se pueda recuperar cualquiera de ellos, el primero devuelve el mensaje completo, mientras que el segundo permite recuperar los segmentos uno por uno.

Si MQGMO_ALL_SEGMENTS_AVAILABLE se especifica para un mensaje de informe, el gestor de colas comprueba la cola para ver si hay al menos un mensaje de informe para cada uno de los segmentos que componen el mensaje lógico completo. Si existe, se cumple la MQGMO_ALL_SEGMENTS_AVAILABLE condición. Sin embargo, el gestor de colas no comprueba el *tipo* de los mensajes de informe presentes, por lo que puede haber una mezcla de tipos de informe en los mensajes de informe relacionados con los segmentos del mensaje lógico. Por lo tanto, el éxito de MQGMO_ALL_SEGMENTS_AVAILABLE no implica que MQGMO_COMPLETE_MSG vaya a tener éxito. Si hay una mezcla de tipos de informes presentes para los segmentos de un mensaje lógico concreto, esos mensajes de informe deben recuperarse uno por uno.

Puede especificar MQGMO_ALL_SEGMENTS_AVAILABLE con cualquiera de las otras MQGMO_* opciones, y con cualquiera de las MQMO_* opciones.

- En z/OS, esta opción es compatible con colas privadas y compartidas, pero la cola debe tener un tipo de índice MQIT_GROUP_ID. Para las colas compartidas, el objeto CFSTRUCT al que se asigna la cola debe estar en CFLEVEL(3) o superior.
- En las siguientes plataformas:

-  AIX
-  IBM i
-  Linux
-  Windows

y para IBM MQ MQI clients conectados a estos sistemas, esta opción es compatible con todas las colas locales.

Opciones de propiedad

Las opciones siguientes hacen relación a las propiedades del mensaje:

MQGMO_PROPERTIES_AS_Q_DEF

Las propiedades del mensaje, excepto las contenidas en el descriptor del mensaje (o extensión), deben representarse tal y como se define en el **PropertyControl** atributo de cola. Si se proporciona `MsgHandle` un, esta opción se ignora y las propiedades del mensaje están disponibles a través del `MsgHandle`, a menos que el valor del atributo **PropertyControl** de cola sea `MQPROP_FORCE_MQRFH2`.

Se trata de la acción predeterminada si no se especifican opciones de propiedad.

MQGMO_PROPERTIES_IN_HANDLE

Las propiedades del mensaje deben estar disponibles a través de `MsgHandle`. Si no se proporciona ningún manejador de mensajes, la llamada fallará con la razón `MQRC_HMSG_ERROR`.

Nota: Si posteriormente el mensaje es leído por una aplicación que no crea un identificador de mensaje, el gestor de colas coloca las propiedades del mensaje en una `MQRFH2` estructura. Es posible que la presencia de un encabezado `MQRFH2` inesperado altere el comportamiento de una aplicación existente.

MQGMO_NO_PROPERTIES

No se recuperarán las propiedades del mensaje, excepto aquellas contenidas en el descriptor del mensaje (o extensión). Si se proporciona `MsgHandle` un, se ignorará.

MQGMO_PROPERTIES_FORCE_MQRFH2

Las propiedades del mensaje, excepto las contenidas en el descriptor del mensaje (o extensión), deben representarse mediante `MQRFH2` encabezados. Esto proporciona compatibilidad con versiones anteriores para aplicaciones que esperan recuperar propiedades, pero que no pueden modificarse para utilizar identificadores de mensajes. Si se proporciona `MsgHandle` un, se ignora.

MQGMO_PROPERTIES_COMPATIBILITY

Si el mensaje contiene una propiedad con un prefijo, "**mcd.**", "**jms.**", "**usr.**", o "**mqext.**", todas las propiedades del mensaje se envían a la aplicación en un `MQRFH2` encabezado. De lo contrario, todas las propiedades del mensaje, excepto las que se encuentran en el descriptor de mensaje (o extensión), se descartan y dejan de estar accesibles para la aplicación.

Opción predeterminada

Si no se requiere ninguna de las opciones descritas, se puede utilizar la siguiente opción:

MQGMO_NONE

Utilice este valor para indicar que no se han especificado otras opciones; todas las opciones toman sus valores predeterminados. `MQGMO_NONE` documentación del programa de ayuda; no está previsto que esta opción se utilice con ninguna otra, pero como su valor es cero, dicho uso no puede detectarse.

El valor inicial del campo `Options` es `MQGMO_NO_WAIT` más `MQGMO_PROPERTIES_AS_Q_DEF`.

WaitInterval (MQLONG)

Este es el tiempo aproximado, expresado en milisegundos, que la llamada `MQGET` espera a que llegue un mensaje adecuado (es decir, un mensaje que cumpla los criterios de selección especificados en el **MsgDesc** parámetro de la llamada `MQGET`). Consulte el *MsgId* campo descrito en "[MQMD - Descriptor de mensaje](#)" en la [página 389](#) para obtener más detalles). Si no se ha recibido ningún mensaje adecuado tras transcurrido este tiempo, la llamada finaliza con `MQCC_FAILED` y el código de motivo `MQRC_NO_MSG_AVAILABLE`.

Importante: No hay espera ni retraso si hay un mensaje adecuado disponible de inmediato.

En z/OS, el tiempo que la llamada `MQGET` espera realmente se ve afectado por la carga del sistema y consideraciones de programación del trabajo, y puede variar entre el valor especificado para *WaitInterval* y aproximadamente 100 milisegundos más que *WaitInterval*.

WaitInterval Se utiliza junto con la opción MQGMO_WAIT o MQGMO_SET_SIGNAL. Se ignora si no se especifica ninguno de los dos. Si se especifica uno de estos, *WaitInterval* debe ser mayor o igual a cero, o el siguiente valor especial:

MQWI_ILIMITADO

Intervalo de espera ilimitado.

El valor inicial de este campo es 0.

Signal1 (MQLONG)

Este es un campo de entrada que solo se utiliza junto con la opción MQGMO_SET_SIGNAL; identifica una señal que se enviará cuando haya un mensaje disponible.

Nota: El tipo de datos y el uso de este campo vienen determinados por el entorno; por este motivo, las aplicaciones que desee transferir entre diferentes entornos no deben utilizar señales.

- En z/OS, este campo debe contener la dirección de un bloque de control de eventos (ECB). El BCE debe ser autorizado por la aplicación antes de que se emita la llamada MQGET. El almacenamiento que contiene el ECB no debe liberarse hasta que se cierre la cola. El ECB es enviado por el gestor de colas con uno de los códigos de finalización de señal descritos. Estos códigos de finalización se establecen en los bits 2 a 31 del ECB, el área definida en la z/OS macro de asignación IHAECB como destinada a un código de finalización de usuario.
- En todos los demás entornos, este es un campo reservado; su valor no es significativo.

Los códigos de finalización de señal son:

MQEC_MSG_LLEGADO

Ha llegado un mensaje adecuado a la cola. Este mensaje no se ha reservado para la persona que llama; se debe emitir una segunda solicitud MQGET, pero otra aplicación podría recuperar el mensaje antes de que se realice la segunda solicitud.

MQEC_WAIT_INTERVAL_EXPIRED

El especificado *WaitInterval* ha caducado sin que haya llegado ningún mensaje adecuado.

MQEC_WAIT_CANCELADO

La espera se canceló por un motivo indeterminado (por ejemplo, el administrador de la cola se cerró o la cola se desactivó). Vuelva a enviar la solicitud si desea un diagnóstico más detallado.

MQEC_Q_MGR_INACTIVO

La espera se canceló porque el gestor de colas ha entrado en estado de inactividad (se especificó MQGMO_FAIL_IF QUIESCING en la llamada MQGET).

MQEC_CONEXIÓN QUIESCING

La espera se canceló porque la conexión ha entrado en estado de inactividad (se especificó MQGMO_FAIL_IF QUIESCING en la llamada MQGET).

El valor inicial de este campo viene determinado por el entorno:

- En z/OS, el valor inicial es el puntero nulo.
- En todos los demás entornos, el valor inicial es 0.

Signal2 (MQLONG)

Este es un campo de entrada que solo se utiliza junto con la opción MQGMO_SET_SIGNAL. Es un campo reservado; su valor no es significativo.

El valor inicial de este campo es 0.

ResolvedQName (MQCHAR48)

Este es un campo de salida que el gestor de colas establece con el nombre local de la cola desde la que se recuperó el mensaje, tal y como se define en el gestor de colas local. Esto difiere del nombre utilizado para abrir la cola si:

- Se abrió una cola alias (en cuyo caso, se devuelve el nombre de la cola local a la que se resolvió el alias), o
- Se abrió una cola modelo (en cuyo caso, se devuelve el nombre de la cola local dinámica).

La longitud de este campo viene determinada por MQ_Q_NAME_LENGTH. El valor inicial de este campo es la cadena nula en C y 48 caracteres en blanco en otros lenguajes de programación.

MatchOptions (MQLONG)

Estas opciones permiten a la aplicación elegir qué campos del **MsgDesc** parámetro utilizar para seleccionar el mensaje devuelto por la llamada MQGET. La aplicación establece las opciones requeridas en este campo y, a continuación, establece los campos correspondientes en el **MsgDesc** parámetro con los valores requeridos para esos campos. Solo los mensajes que tienen esos valores en el MQMD para el mensaje son candidatos para la recuperación utilizando ese **MsgDesc** parámetro en la llamada MQGET. Los campos para los que no se especifica la opción de coincidencia correspondiente se ignoran al seleccionar el mensaje que se va a devolver. Si no se especifican criterios de selección en la llamada MQGET (es decir, se acepta *cualquier* mensaje), establezca *MatchOptions* en MQMO_NONE.

- En z/OS, los criterios de selección que se pueden utilizar pueden estar restringidos por el tipo de índice utilizado para la cola. Consulte el **IndexType** atributo cola para obtener más detalles.

Si se especifica MQGMO_LOGICAL_ORDER, solo determinados mensajes podrán ser devueltos por la siguiente llamada MQGET:

- Si no hay ningún grupo o mensaje lógico actual, solo los mensajes que sean *MsgSeqNumber* iguales a 1 e *Offset* iguales a 0 podrán ser devueltos. En esta situación, puede utilizar una o varias de las siguientes opciones de coincidencia para seleccionar cuál de los mensajes elegibles se devuelve:
 - MQMO_MATCH_MSG_ID
 - MQMO_MATCH_CORREL_ID
 - MQMO_MATCH_GROUP_ID
- Si hay un mensaje de grupo o lógico actual, solo el siguiente mensaje del grupo o el siguiente segmento del mensaje lógico es elegible para su devolución, y esto no se puede modificar especificando las opciones MQMO_*.

En ambos casos anteriores, puede especificar opciones de coincidencia que no se apliquen, pero el valor del campo relevante en el **MsgDesc** parámetro debe coincidir con el valor del campo correspondiente en el mensaje que se va a devolver; la llamada falla con el código de motivo MQRC_MATCH_OPTIONS_ERROR si no se cumple esta condición.

MatchOptions se ignora si se especifica MQGMO_MSG_UNDER_CURSOR o MQGMO_BROWSE_MSG_UNDER_CURSOR.

La recepción de mensajes basados en propiedades de mensajes no se realiza mediante opciones de coincidencia; para obtener más información, consulte [“SelectionString \(MQCHARV\)”](#) en la página 469.

Puede especificar una o varias de las siguientes opciones de coincidencia:

MQMO_MATCH_MSG_ID

El mensaje que se va a recuperar debe tener un identificador de mensaje que coincida con el valor del *MsgId* campo en el **MsgDesc** parámetro de la llamada MQGET. Esta coincidencia se suma a cualquier otra coincidencia que pueda aplicarse (por ejemplo, el identificador de correlación).

Si omite esta opción, se ignorará el *MsgId* campo del **MsgDesc** parámetro y coincidirá cualquier identificador de mensaje.

Nota: El identificador de mensaje MQMI_NONE es un valor especial que coincide con cualquier identificador de mensaje en el MQMD para el mensaje. Por lo tanto, especificar MQMO_MATCH_MSG_ID con MQMI_NONE es lo mismo que no especificar MQMO_MATCH_MSG_ID.

MQMO_MATCH_CORREL_ID

El mensaje que se va a recuperar debe tener un identificador de correlación que coincida con el valor del *CorrelId* campo en el **MsgDesc** parámetro de la llamada MQGET. Esta coincidencia se suma a cualquier otra coincidencia que pueda aplicarse (por ejemplo, el identificador del mensaje).

Si omite esta opción, se ignorará el *CorrelId* campo del **MsgDesc** parámetro y coincidirá cualquier identificador de correlación.

Nota: El identificador de correlación MQCI_NONE es un valor especial que coincide con *cualquier* identificador de correlación en el MQMD para el mensaje. Por lo tanto, especificar MQMO_MATCH_CORREL_ID con MQCI_NONE es lo mismo que no especificar MQMO_MATCH_CORREL_ID.

MQMO_MATCH_GROUP_ID

El mensaje que se va a recuperar debe tener un identificador de grupo que coincida con el valor del *GroupId* campo en el **MsgDesc** parámetro de la llamada MQGET. Esta coincidencia se suma a cualquier otra coincidencia que pueda aplicarse (por ejemplo, el identificador de correlación).

Si omite esta opción, se ignorará el *GroupId* campo del **MsgDesc** parámetro y coincidirá cualquier identificador de grupo.

Nota: El identificador de grupo MQGI_NONE es un valor especial que coincide con *cualquier* identificador de grupo en el MQMD para el mensaje. Por lo tanto, especificar MQMO_MATCH_GROUP_ID con MQGI_NONE es lo mismo que no especificar MQMO_MATCH_GROUP_ID.

MQMO_MATCH_MSG_SEQ_NUMBER

El mensaje que se va a recuperar debe tener un número de secuencia de mensaje que coincida con el valor del *MsgSeqNumber* campo del **MsgDesc** parámetro de la llamada MQGET. Esta coincidencia se suma a cualquier otra coincidencia que pueda aplicarse (por ejemplo, el identificador de grupo).

Si omite esta opción, se ignorará el *MsgSeqNumber* campo del **MsgDesc** parámetro y coincidirá cualquier número de secuencia de mensajes.

MQMO_MATCH_OFFSET

El mensaje que se va a recuperar debe tener un desplazamiento que coincida con el valor del *Offset* campo en el **MsgDesc** parámetro de la llamada MQGET. Esta coincidencia se suma a cualquier otra coincidencia que pueda aplicarse (por ejemplo, el número de secuencia del mensaje).

Si se omite esta opción sin especificarla, se ignorará el *Offset* campo del **MsgDesc** parámetro y cualquier desplazamiento coincidirá.

- Esta opción no es compatible con z/OS.

MQMO_MATCH_MSG_TOKEN

El mensaje que se va a recuperar debe tener un token de mensaje que coincida con el valor del *MsgToken* campo de la estructura MQGMO especificada en la llamada MQGET.

Puede especificar esta opción para todas las colas locales. Si lo especifica para una cola que tiene un *IndexType* de MQIT_MSG_TOKEN (una cola gestionada por WLM), no puede especificar otras opciones de coincidencia con MQMO_MATCH_MSG_TOKEN.

No se puede especificar MQMO_MATCH_MSG_TOKEN con MQGMO_WAIT o MQGMO_SET_SIGNAL. Si la aplicación desea esperar a que llegue un mensaje a una cola que tiene un *IndexType* de MQIT_MSG_TOKEN, especifique MQMO_NONE.

Si omite esta opción, se ignorará el *MsgToken* campo en MQGMO y coincidirá cualquier token de mensaje.

Si no se especifica ninguna de las opciones descritas, se puede utilizar la siguiente opción:

MQMO_NONE

No utilice coincidencias al seleccionar el mensaje que se va a devolver; todos los mensajes de la cola son aptos para su recuperación (pero están sujetos al control de las opciones MQGMO_ALL_MSGS_AVAILABLE, MQGMO_ALL_SEGMENTS_AVAILABLE y MQGMO_COMPLETE_MSG).

MQMO_NONE ayuda con la documentación del programa. No está previsto que esta opción se utilice con ninguna otra opción MQMO_*, pero como su valor es cero, dicho uso no puede detectarse.

Este es un campo de entrada. El valor inicial de este campo es MQMO_MATCH_MSG_ID con MQMO_MATCH_CORREL_ID. Este campo se ignora si *Version* es menor que MQGMO_VERSION_2.

Nota: El valor inicial del *MatchOptions* campo se define para garantizar la compatibilidad con los gestores de MQSeries colas anteriores. Sin embargo, al leer una serie de mensajes de una cola sin utilizar criterios de selección, este valor inicial requiere que la aplicación restablezca los campos *MsgId CorrelId* y a MQMI_NONE y MQCI_NONE antes de cada llamada MQGET. Evita la necesidad de restablecer *MsgId* y *CorrelId* estableciendo *Version* en MQGMO_VERSION_2 y *MatchOptions* en MQMO_NONE.

GroupStatus (MQCHAR)

Esta bandera indica si el mensaje recuperado pertenece a un grupo.

Tiene uno de los siguientes valores:

MQGS_NO_EN_GRUPO

El mensaje no está en un grupo.

MQGS_MSG_EN_GRUPO

El mensaje está en un grupo, pero no es el último del grupo.

Último mensaje en el grupo

El mensaje es el último del grupo.

Este es también el valor devuelto si el grupo consta de un solo mensaje.

Se trata de un campo de salida. El valor inicial de este campo es MQGS_NOT_IN_GROUP. Este campo se ignora si *Version* es menor que MQGMO_VERSION_2.

SegmentStatus (MQCHAR)

Esta es una bandera que indica si el mensaje recuperado es un segmento de un mensaje lógico. Tiene uno de los siguientes valores:

MQSS_NO_ES_UN_SEGMENTO

El mensaje no es un segmento.

MQSS_SEGMENTO

El mensaje es un segmento, pero no es el último segmento del mensaje lógico.

MQSS_ÚLTIMO_SEGMENTO

El mensaje es el último segmento del mensaje lógico.

Este es también el valor devuelto si el mensaje lógico consta de un solo segmento.

En z/OS, el gestor de colas siempre establece este campo en MQSS_NOT_A_SEGMENT.

Se trata de un campo de salida. El valor inicial de este campo es MQSS_NOT_A_SEGMENT. Este campo se ignora si *Version* es menor que MQGMO_VERSION_2.

Segmentación (MQCHAR)

Esta es una bandera que indica si se permite una mayor segmentación del mensaje recuperado. Tiene uno de los siguientes valores:

MQSEG_INHIBIDO

No se permite la segmentación.

MQSEG_PERMITIDO

Segmentación permitida.

En z/OS, el gestor de colas siempre establece este campo en MQSEG_INHIBITED.

Se trata de un campo de salida. El valor inicial de este campo es MQSEG_INHIBITED. Este campo se ignora si *Version* es menor que MQGMO_VERSION_2.

Reserved1 (MQCHAR)

Este es un campo reservado. El valor inicial de este campo es un carácter en blanco. Este campo se ignora si *Version* es menor que MQGMO_VERSION_2.

MsgToken (MQBYTE16)

MsgToken campo - Estructura MQGMO. Este campo es utilizado por el gestor de colas para identificar de forma única un mensaje.

Se trata de una cadena de bytes generada por el gestor de colas para identificar de forma única un mensaje en una cola. El token del mensaje se genera cuando el mensaje se coloca por primera vez en el gestor de colas y permanece con el mensaje hasta que este se elimina permanentemente del gestor de colas, a menos que se reinicie el gestor de colas.

Cuando el mensaje se elimina de la cola, el *MsgToken* que identificaba esa instancia del mensaje ya no es válido y nunca se vuelve a utilizar. Si se reinicia el gestor de colas, es posible que la *MsgToken* que identificaba un mensaje en la cola antes del reinicio no sea válida después del reinicio. Sin embargo, el nunca *MsgToken* se reutiliza para identificar una instancia de mensaje diferente. El *MsgToken* es generado por el gestor de colas y no es visible para ninguna aplicación externa.

Cuando se devuelve un mensaje mediante una llamada a MQGET en la que se proporciona un MQGMO de versión 3 o superior, el *MsgToken* que identifica el mensaje en la cola es devuelto en el MQGMO por el gestor de colas. Hay una excepción a esto: cuando el mensaje se elimina de la cola fuera del punto de sincronización, es posible que el gestor de colas no devuelva un *MsgToken* porque no es útil para identificar el mensaje devuelto en una llamada MQGET posterior. Las aplicaciones solo deben utilizar *MsgToken* para hacer referencia al mensaje en llamadas MQGET posteriores.

Si se proporciona *MsgToken* un y se especifica el *MatchOption* MQMO_MATCH_MSG_TOKEN y no se especifica ni MQGMO_MSG_UNDER_CURSOR ni MQGMO_BROWSE_MSG_UNDER_CURSOR, solo se puede devolver el mensaje *MsgToken* identificado por ese. La opción es válida en todas las colas locales independientemente de INDXTYPE, y en z/OS debe utilizar INDXTYPE(MSGTOKEN) solo en las colas del Gestor de carga de trabajo (WLM).

Se comprueban todos los demás *MatchOptions* especificados y, si no coinciden, se devuelve MQRC_NO_MSG_AVAILABLE. Si MQGMO_BROWSE_NEXT se codifica con MQMO_MATCH_MSG_TOKEN, el mensaje identificado por solo se devuelve si se *MsgToken* encuentra más allá del cursor de exploración del identificador de llamada.

Si se especifica MQGMO_MSG_UNDER_CURSOR o MQGMO_BROWSE_MSG_UNDER_CURSOR, se ignora MQMO_MATCH_MSG_TOKEN.

MQMO_MATCH_MSG_TOKEN no es válido con las siguientes opciones de obtención de mensajes:

- MQGMO_WAIT
- MQGMO_SET_SIGNAL

Para una llamada MQGET que especifique MQMO_MATCH_MSG_TOKEN, se debe proporcionar un MQGMO de versión 3 o posterior a la llamada; de lo contrario, se devolverá MQRC_WRONG_GMO_VERSION.

Si el no *MsgToken* es válido en este momento, se devuelve MQCC_FAILED con MQRC_NO_MSG_AVAILABLE, a menos que haya otro error.

ReturnedLength (MQLONG)

Este es un campo de salida que el gestor de colas establece con la longitud en bytes de los datos del mensaje devueltos por la llamada MQGET en el **Buffer** parámetro. Si el gestor de colas no admite esta capacidad, *ReturnedLength* se establece en el valor MQRL_UNDEFINED.

Cuando los mensajes se convierten entre codificaciones o conjuntos de caracteres, los datos del mensaje pueden cambiar de tamaño en ocasiones. Al regresar de la llamada MQGET:

- Si no *ReturnedLength* es MQRL_UNDEFINED, el número de bytes de datos del mensaje devueltos viene dado por *ReturnedLength*.
- Si *ReturnedLength* tiene el valor MQRL_UNDEFINED, el número de bytes de datos del mensaje devueltos suele ser el menor de *BufferLength* y *DataLength*, pero puede ser inferior a este si la llamada MQGET se completa con el código de motivo MQRC_TRUNCATED_MSG_ACCEPTED. Si esto ocurre, los bytes insignificantes del **Buffer** parámetro se establecen en nulos.

Se define el siguiente valor especial:

MQRL_INDETERMINADO

Longitud de los datos devueltos no definida.

En z/OS, el valor devuelto para el *ReturnedLength* campo es siempre MQRL_UNDEFINED.

El valor inicial de este campo es MQRL_UNDEFINED. Este campo se ignora si *Version* es menor que MQGMO_VERSION_3.

Reserved2 (MQLONG)

Este es un campo reservado. El valor inicial de este campo es un carácter en blanco. Este campo se ignora si *Version* es menor que MQGMO_VERSION_4.

MsgHandle (MQHMSG)

Si se especifica la opción MQGMO_PROPERTIES_AS_Q_DEF y el **PropertyControl** atributo de cola no está establecido en MQPROP_FORCE_MQRFH2, este es el identificador de un mensaje que se rellenará con las propiedades del mensaje que se recupera de la cola. El identificador se crea mediante una llamada MQCRTMH. Cualquier propiedad ya asociada con el identificador se borrará antes de recuperar un mensaje.

También se puede especificar el siguiente valor:

MQHM_NINGUNO

No se ha proporcionado ningún identificador de mensaje.

No se requiere ningún descriptor de mensaje en la llamada MQGET si se proporciona un identificador de mensaje válido y se utiliza en la salida para contener las propiedades del mensaje; el descriptor de mensaje asociado al identificador de mensaje se utiliza para los campos de entrada.

Si se especifica un descriptor de mensaje en la llamada MQGET, siempre tiene prioridad sobre el descriptor de mensaje asociado a un identificador de mensaje.

Si se especifica MQGMO_PROPERTIES_FORCE_MQRFH2, o se especifica MQGMO_PROPERTIES_AS_Q_DEF y el **PropertyControl** atributo de cola es MQPROP_FORCE_MQRFH2, la llamada falla con el código de motivo MQRC_MD_ERROR cuando no se especifica ningún parámetro descriptor de mensaje.

Al regresar de la llamada MQGET, las propiedades y el descriptor de mensajes asociados con este identificador de mensajes se actualizan para reflejar el estado del mensaje recuperado (así como el descriptor de mensajes, si se proporcionó uno en la llamada MQGET). Las propiedades del mensaje se pueden consultar utilizando la llamada MQINQMP.

A excepción de las extensiones del descriptor de mensajes, cuando están presentes, una propiedad que se puede consultar con la llamada MQINQMP no está contenida en los datos del mensaje; si el mensaje de la cola contenía propiedades en los datos del mensaje, estas se eliminan de los datos del mensaje antes de que los datos se devuelvan a la aplicación.

Si no se proporciona ningún identificador de mensaje o la versión es inferior a MQGMO_VERSION_4, debe proporcionar un descriptor de mensaje válido en la llamada MQGET. Cualquier propiedad del mensaje (excepto las contenidas en el descriptor del mensaje) se devuelve en los datos del mensaje, sujeto al valor de las opciones de propiedad en la estructura MQGMO y el **PropertyControl** atributo de cola.

Este es siempre un campo de entrada. El valor inicial de este campo es MQHM_NONE. Este campo se ignora si **Version** es menor que MQGMO_VERSION_4.

MQIIH – Cabecera información de IMS

La estructura MQIIH describe la información del encabezado de un mensaje enviado a IMS a través del IMS puente. Para cualquier IBM MQ plataforma compatible, puede crear y transmitir un mensaje que incluya la estructura MQIIH, pero solo un IBM MQ for z/OS gestor de colas puede utilizar el IMS puente. Por lo tanto, para que el mensaje llegue a IMS desde un gestor de colas z/OS que no sea el suyo, su red de gestores de colas debe incluir al menos un z/OS gestor de colas a través del cual se pueda enrutar el mensaje.

Disponibilidad

Todos IBM MQ los sistemas y IBM MQ clientes.

Nombre de formato

MQFMT_IMS

Conjunto de caracteres y codificación

Se aplican condiciones especiales al conjunto de caracteres y la codificación utilizados para la estructura MQIIH y los datos de los mensajes de aplicación:

- Las aplicaciones que se conectan al gestor de colas propietario de la cola puente IMS deben proporcionar una estructura MQIIH que esté en el juego de caracteres y la codificación del gestor de colas. Esto se debe a que, en este caso, no se realiza la conversión de datos de la estructura MQIIH.
- Las aplicaciones que se conectan a otros gestores de colas pueden proporcionar una estructura MQIIH en cualquiera de los conjuntos de caracteres y codificaciones compatibles; el agente del canal de mensajes receptor conectado al gestor de colas propietario de la cola puente IMS convierte el MQIIH.
- Los datos del mensaje de aplicación que siguen la estructura MQIIH deben tener el mismo juego de caracteres y codificación que la estructura MQIIH. No utilice los campos *CodedCharSetId* *Encoding* y de la estructura MQIIH para especificar el juego de caracteres y la codificación de los datos del mensaje de la aplicación.

Debe proporcionar una salida de conversión de datos para convertir los datos del mensaje de la aplicación si los datos no están en uno de los formatos integrados compatibles con el gestor de colas.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 512. Campos en MQIIH para MQIIH		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>StrucId</u> (identificador de estructura)	MQIIH_STRUC_ID	'IIH~'
<u>Versión</u> (número de versión de la estructura)	MQIIH_VERSION_1	1
<u>StrucLength</u> (longitud de la estructura MQIIH)	MQIIH_LENGTH_1	84
<u>Codificación</u> (reservado - véase “Conjunto de caracteres y codificación” en la página 372)	Ninguna	0
<u>CodedCharSetId</u> (reservado - ver “Conjunto de caracteres y codificación” en la página 372)	Ninguna	0

Tabla 512. Campos en MQIIH para MQIIH (continuación)

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>Formato</u> (nombre del formato MQ e de los datos que siguen a MQIIH)	MQFMT_NONE	Espacios en blanco
<u>Banderas</u> (banderas)	MQIIH_NINGUNO	0
<u>LTermOverride</u> (anulación lógica del terminal)	Ninguna	Espacios en blanco
<u>MFSMapName</u> (formato del mensaje, nombre del mapa de servicios)	Ninguna	Espacios en blanco
<u>ReplyToFormat</u> (Nombre del formato MQ e del mensaje de respuesta)	MQFMT_NONE	Espacios en blanco
<u>Autenticador</u> (RACF contraseña o pasaporte)	MQIAUT_NINGUNO	Espacios en blanco
<u>TranInstanceId</u> (identificador de instancia de transacción)	MQITII_NINGUNO	Nulos
<u>TranState</u> (estado de la transacción)	MQITS_NO_EN_CONVER SACIÓN	'¬'
<u>CommitMode</u> (modo de compromiso)	MQICM_COMPROMETER _Y_ENVIAR	'0'
<u>SecurityScope</u> (ámbito de seguridad)	MQISS_COMPROBAR	'C'
<u>Reservado</u> (reservado)	Ninguna	'¬'

Notas:

1. El símbolo ¬ representa un único carácter en blanco.
2. En el lenguaje de programación C, la variable macro MQIIH_DEFAULT contiene los valores que se enumeran en la tabla. Se puede utilizar de la siguiente manera para proporcionar valores iniciales para los campos de la estructura:

```
MQIIH MyIIH = {MQIIH_DEFAULT};
```

Declaraciones lingüísticas

Declaración C para MQIIH

```
typedef struct tagMQIIH MQIIH;
struct tagMQIIH {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQLONG     StrucLength;      /* Length of MQIIH structure */
    MQLONG     Encoding;         /* Reserved */
    MQLONG     CodedCharSetId;   /* Reserved */
    MQCHAR8    Format;           /* MQ format name of data that follows
                                MQIIH */
    MQLONG     Flags;            /* Flags */
    MQCHAR8    LTermOverride;    /* Logical terminal override */
    MQCHAR8    MFSMapName;       /* Message format services map name */
    MQCHAR8    ReplyToFormat;    /* MQ format name of reply message */
    MQCHAR8    Authenticator;    /* RACF password or passticket */
    MQBYTE16   TranInstanceId;   /* Transaction instance identifier */
    MQCHAR     TranState;        /* Transaction state */
    MQCHAR     CommitMode;       /* Commit mode */
    MQCHAR     SecurityScope;    /* Security scope */
    MQCHAR     Reserved;        /* Reserved */
};
```

COBOL declaración para MQIIH

```

** MQIIH structure
10 MQIIH.
**   Structure identifier
15 MQIIH-STRUCID      PIC X(4).
**   Structure version number
15 MQIIH-VERSION      PIC S9(9) BINARY.
**   Length of MQIIH structure
15 MQIIH-STRUCLNGTH   PIC S9(9) BINARY.
**   Reserved
15 MQIIH-ENCODING      PIC S9(9) BINARY.
**   Reserved
15 MQIIH-CODEDCHARSETID PIC S9(9) BINARY.
**   MQ format name of data that follows MQIIH
15 MQIIH-FORMAT        PIC X(8).
**   Flags
15 MQIIH-FLAGS         PIC S9(9) BINARY.
**   Logical terminal override
15 MQIIH-LTERMOVERRIDE PIC X(8).
**   Message format services map name
15 MQIIH-MFSMAPNAME     PIC X(8).
**   MQ format name of reply message
15 MQIIH-REPLYTOFORMAT PIC X(8).
**   RACF password or passticket
15 MQIIH-AUTHENTICATOR PIC X(8).
**   Transaction instance identifier
15 MQIIH-TRANINSTANCEID PIC X(16).
**   Transaction state
15 MQIIH-TRANSTATE      PIC X.
**   Commit mode
15 MQIIH-COMMITMODE      PIC X.
**   Security scope
15 MQIIH-SECURITYSCOPE  PIC X.
**   Reserved
15 MQIIH-RESERVED        PIC X.

```

PL/I declaración para MQIIH

```

dcl
1 MQIIH based,
3 StrucId      char(4),      /* Structure identifier */
3 Version      fixed bin(31), /* Structure version number */
3 StrucLength   fixed bin(31), /* Length of MQIIH structure */
3 Encoding      fixed bin(31), /* Reserved */
3 CodedCharSetId fixed bin(31), /* Reserved */
3 Format        char(8),      /* MQ format name of data that follows
                               MQIIH */
3 Flags        fixed bin(31), /* Flags */
3 LTermOverride char(8),      /* Logical terminal override */
3 MFSMapName    char(8),      /* Message format services map name */
3 ReplyToFormat char(8),      /* MQ format name of reply message */
3 Authenticator char(8),      /* RACF password or passticket */
3 TranInstanceId char(16),    /* Transaction instance identifier */
3 TranState     char(1),      /* Transaction state */
3 CommitMode     char(1),      /* Commit mode */
3 SecurityScope  char(1),      /* Security scope */
3 Reserved      char(1);      /* Reserved */

```

High Level Assembler declaración para MQIIH

MQIIH	DSECT	
MQIIH_STRUCID	DS	CL4 Structure identifier
MQIIH_VERSION	DS	F Structure version number
MQIIH_STRUCLNGTH	DS	F Length of MQIIH structure
MQIIH_ENCODING	DS	F Reserved
MQIIH_CODEDCHARSETID	DS	F Reserved
MQIIH_FORMAT	DS	CL8 MQ format name of data that follows
*		MQIIH
MQIIH_FLAGS	DS	F Flags
MQIIH_LTERMOVERRIDE	DS	CL8 Logical terminal override
MQIIH_MFSMAPNAME	DS	CL8 Message format services map name
MQIIH_REPLYTOFORMAT	DS	CL8 MQ format name of reply message
MQIIH_AUTHENTICATOR	DS	CL8 RACF password or passticket
MQIIH_TRANINSTANCEID	DS	XL16 Transaction instance identifier
MQIIH_TRANSTATE	DS	CL1 Transaction state
MQIIH_COMMITMODE	DS	CL1 Commit mode
MQIIH_SECURITYSCOPE	DS	CL1 Security scope

MQIIH_RESERVED	DS	CL1	Reserved
*MQIIH_LENGTH	EQU	*-MQIIH	
	ORG	MQIIH	
MQIIH_AREA	DS	CL(MQIIH_LENGTH)	

Declaración de Visual Basic para MQIIH

```

Type MQIIH
  StrucId      As String*4 'Structure identifier'
  Version      As Long     'Structure version number'
  StrucLength  As Long     'Length of MQIIH structure'
  Encoding     As Long     'Reserved'
  CodedCharSetId As Long   'Reserved'
  Format       As String*8 'MQ format name of data that follows MQIIH'
  Flags       As Long     'Flags'
  LTermOverride As String*8 'Logical terminal override'
  MFSSMapName  As String*8 'Message format services map name'
  ReplyToFormat As String*8 'MQ format name of reply message'
  Authenticator As String*8 'RACF password or passticket'
  TranInstanceId As MQBYTE16 'Transaction instance identifier'
  TranState     As String*1 'Transaction state'
  CommitMode    As String*1 'Commit mode'
  SecurityScope As String*1 'Security scope'
  Reserved      As String*1 'Reserved'
End Type

```

Detalles de campo para MQIIH

La estructura del encabezado de IMS información MQIIH describe la información del encabezado de un mensaje enviado a IMS través del IMS puente.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 513. Campos en MQIIH para MQIIH		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>StrucId</u> (identificador de estructura)	MQIIH_STRUC_ID	' I IH~ '
<u>Versión</u> (número de versión de la estructura)	MQIIH_VERSION_1	1
<u>StrucLength</u> (longitud de la estructura MQIIH)	MQIIH_LENGTH_1	84
<u>Codificación</u> (reservado - véase “Conjunto de caracteres y codificación” en la página 372)	Ninguna	0
<u>CodedCharSetId</u> (reservado - ver “Conjunto de caracteres y codificación” en la página 372)	Ninguna	0
<u>Formato</u> (nombre del formato MQ e de los datos que siguen a MQIIH)	MQFMT_NONE	Espacios en blanco
<u>Banderas</u> (banderas)	MQIIH_NINGUNO	0
<u>LTermOverride</u> (anulación lógica del terminal)	Ninguna	Espacios en blanco
<u>MFSSMapName</u> (formato del mensaje, nombre del mapa de servicios)	Ninguna	Espacios en blanco
<u>ReplyToFormat</u> (Nombre del formato MQ e del mensaje de respuesta)	MQFMT_NONE	Espacios en blanco
<u>Autenticador</u> (RACF contraseña o pasaporte)	MQIAUT_NINGUNO	Espacios en blanco
<u>TranInstanceId</u> (identificador de instancia de transacción)	MQITII_NINGUNO	Nulos

Tabla 513. Campos en MQIIH para MQIIH (continuación)

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>TranState</u> (estado de la transacción)	MQITS_NO_EN_CONVER SACIÓN	'↵'
<u>CommitMode</u> (modo de compromiso)	MQICM_COMPROMETER _Y_ENVIAR	'0'
<u>SecurityScope</u> (ámbito de seguridad)	MQISS_COMPROBAR	'C'
<u>Reservado</u> (reservado)	Ninguna	'↵'
Notas: 1. El símbolo ↵ representa un único carácter en blanco. 2. En el lenguaje de programación C, la variable macroMQIIH_DEFAULT contiene los valores que se enumeran en la tabla. Se puede utilizar de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQIIH MyIIH = {MQIIH_DEFAULT};</pre>		

StrucId (MQCHAR4)

Este es el identificador de estructura de la estructura IMS de cabecera de información. Siempre es un campo de entrada. Su valor es MQIIH_STRUC_ID.

El valor debe ser:

MQIIH_STRUC_ID

Identificador de la estructura del IMS encabezado de información.

Para el lenguaje de programación C, también se define la constante MQIIH_STRUC_ID_ARRAY. Tiene el mismo valor que MQIIH_STRUC_ID, pero es una matriz de caracteres en lugar de una cadena.

Versión (MQLONG)

Este es el número de versión de la estructura. El valor debe ser:

MQIIH_VERSION_1

Número de versión para la estructura del IMS encabezado de información.

La siguiente constante especifica el número de versión de la versión actual:

MQIIH_VERSIÓN_ACTUAL

Versión actual de la estructura del IMS encabezado de información.

El valor inicial de este campo es MQIIH_VERSION_1.

StrucLength (MQLONG)

Esta es la longitud de la estructura MQIIH. El valor debe ser:

MQIIH_LENGTH_1

Longitud de la estructura del IMS encabezado de información.

El valor inicial de este campo es MQIIH_LENGTH_1.

Encoding (MQLONG)

Este es un campo reservado; su valor no es significativo. El valor inicial de este campo es 0.

La codificación para las estructuras compatibles que siguen una estructura MQIIH es la misma que la de la propia estructura MQIIH y se toma de cualquier encabezado MQ anterior.

CodedCharSetId (MQLONG)

Este es un campo reservado; su valor no es significativo. El valor inicial de este campo es 0.

El identificador del conjunto de caracteres para las estructuras compatibles que siguen una estructura MQIIH es el mismo que el de la propia estructura MQIIH y se toma de cualquier encabezado MQ anterior.

Format (MQCHAR8)

Especifica el nombre del formato MQ de los datos que siguen a la estructura MQIIH.

En la llamada MQPUT o MQPUT1, la aplicación debe establecer este campo con el valor adecuado para los datos.

La longitud de este campo viene dada por MQ_FORMAT_LENGTH. El valor inicial de este campo es MQFMT_NONE.

Flags (MQLONG)

El valor de las banderas debe ser:

MQIIH_NINGUNO

Sin banderas.

MQIIH_PLAZO_DE_VIGENCIA

El mensaje de respuesta contiene:

- Las mismas opciones de informe de caducidad que el mensaje de solicitud
- El tiempo restante hasta la expiración del mensaje de solicitud sin ajustes por el tiempo de procesamiento del puente

Si no se establece este valor, el tiempo de caducidad se establece como *ilimitado*.

MQIIH_FORMATO_DE_RESPUESTA_NINGUNO

Establece el campo « MQIIH.Format » (Estado de la respuesta) de la respuesta en MQFMT_NONE.

MQIIH_IGNORAR_PURGA

Establece el indicador TMAMIPRG en el prefijo OTMA, lo que solicita que OTMA ignore las llamadas PURG en la PCB TP para transacciones e CM0.

MQIIH_CM0_REQUEST_RESPONSE

Para las transacciones del modo de compromiso 0 (CM0), este indicador establece el indicador TMAMHRSP en el prefijo OTMA. Al configurar este indicador, se solicita que OTMA/ IMS genere un mensaje « DFS2082 RESPONSE MODE TRANSACTION TERMINATED WITHOUT REPLY» cuando el programa de aplicación IMS original no responda al IOPCB ni cambie el mensaje a otra transacción.

El valor inicial de este campo es MQIIH_NONE.

LTermOverride (MQCHAR8)

La anulación lógica del terminal, situada en el campo IO PCB. Es opcional; si no se especifica, se utiliza el nombre TPIPE. Se ignora si el primer byte está en blanco o es nulo.

La longitud de este campo viene dada por MQ_LTERM_OVERRIDE_LENGTH. El valor inicial de este campo es 8 caracteres en blanco.

MFSMapName (MQCHAR8)

El nombre del mapa de servicios de formato de mensajes, situado en el campo IO PCB. Es opcional. En la entrada representa el MID, en la salida representa el MOD. Se ignora si el primer byte está en blanco o es nulo.

La longitud de este campo viene dada por MQ_MFS_MAP_NAME_LENGTH. El valor inicial de este campo es 8 caracteres en blanco.

ReplyToFormat (MQCHAR8)

Este es el nombre del formato MQ del mensaje de respuesta que se envía en respuesta al mensaje actual. La longitud de este campo viene dada por MQ_FORMAT_LENGTH. El valor inicial de este campo es MQFMT_NONE.

Para convertir los datos del mensaje de respuesta utilizando MQGMO_CONVERT, especifique MQIIH.replyToFormat=MQFMT_STRING o MQIIH.replyToFormat=MQFMT_IMS_VAR_STRING. Para obtener una explicación sobre el uso de estos campos, consulte [“Format \(MQCHAR8\)”](#) en la página 417.

Si se utiliza el valor predeterminado (MQIIH.replyToFormat=MQFMT_NONE) en el mensaje de solicitud y el mensaje de respuesta se recupera utilizando MQGMO_CONVERT, no se realiza ninguna conversión de datos.

Autenticador (MQCHAR8)

Esta es la RACF contraseña o PassTicket. Es opcional; si se especifica, se utiliza con el ID de usuario en el contexto de seguridad MQMD para crear un UTOKEN que se envía a IMS para proporcionar un contexto de seguridad. Si no se especifica, se utiliza el ID de usuario sin verificación. Esto depende de la configuración de los RACF interruptores, que pueden requerir la presencia de un autenticador.

Esto se ignora si el primer byte está en blanco o es nulo. Se puede utilizar el siguiente valor especial:

MQIAUT_NINGUNO

Sin autenticación.

Para el lenguaje de programación C, también se define la constante MQIAUT_NONE_ARRAY, que tiene el mismo valor que MQIAUT_NONE, pero es una matriz de caracteres en lugar de una cadena.

La longitud de este campo viene dada por MQ_AUTHENTICATOR_LENGTH. El valor inicial de este campo es MQIAUT_NONE.

TranInstanceId (MQBYTE16)

Este es el identificador de la instancia de transacción. Este campo es utilizado por los mensajes de salida de IMS, por lo que se ignora en la primera entrada. Si se establece *TranState* en MQITS_IN_CONVERSATION, esto debe proporcionarse en la siguiente entrada, y en todas las entradas posteriores, para permitir IMS que correlacione los mensajes con la conversación correcta. Puede utilizar el siguiente valor especial:

MQITII_NINGUNO

Sin identificador de instancia de transacción.

Para el lenguaje de programación C, también se define la constante MQITII_NONE_ARRAY, que tiene el mismo valor que MQITII_NONE, pero es una matriz de caracteres en lugar de una cadena.

La longitud de este campo viene dada por MQ_TRAN_INSTANCE_ID_LENGTH. El valor inicial de este campo es MQITII_NONE.

TranState (MQCHAR)

Esto indica el IMS estado de la conversación. Esto se ignora en la primera entrada porque no existe ninguna conversación. En entradas posteriores, indica si una conversación está activa o no. En la salida, se establece mediante IMS. El valor debe ser uno de los siguientes:

MQITS_EN_CONVERSACIÓN

En conversación.

MQITS_NO_EN_CONVERSACIÓN

No en conversación.

MQITS_ARCHITECTED

Devolver los datos del estado de la transacción en formato estructurado.

Este valor solo se utiliza con el IMS/DISPLAY TRAN comando. Devuelve los datos del estado de la transacción en el IMS formato diseñado en lugar del formato de caracteres.  Para obtener más información, consulte [Escribir IMS programas de transacción a través de IBM MQ](#).

El valor inicial de este campo es MQITS_NOT_IN_CONVERSATION.

CommitMode (MQCHAR)

Este es el modo IMS de confirmación. Consulte la *Referencia OTMA* para obtener más información sobre los IMS modos de confirmación. El valor debe ser uno de los siguientes:

MQICM_COMPROMETER_Y_ENVIAR

Confirmar y enviar.

Este modo implica una doble cola de salida, pero con tiempos de ocupación de la región más cortos. Las transacciones de ruta rápida y conversacionales no pueden ejecutarse con este modo.

MQICM_ENVIAR_Y_CONFIRMAR

Enviar y luego confirmar.

Cualquier IMS transacción iniciada como resultado de un modo de confirmación de MQICM_SEND_THEN_COMMIT se ejecuta en modo RESPONSE independientemente de cómo se defina la transacción en la IMS definición del sistema (parámetro MSGTYPE en la macro TRANSACT). Esto también se aplica a las transacciones iniciadas mediante un conmutador de transacciones.

El valor inicial de este campo es MQICM_COMMIT_THEN_SEND.

SecurityScope (MQCHAR)

Esto indica el procesamiento IMS de seguridad requerido. Los valores siguientes están definidos:

MQISS_COMPROBAR

Comprueba el ámbito de seguridad: se crea un ACEE en la región de control, pero no en la región dependiente.

MQISS_COMPLETO

Ámbito de seguridad completo: se crea un ACEE almacenado en caché en la región de control y un ACEE no almacenado en caché en la región dependiente. Si utiliza MQISS_FULL, asegúrese de que el ID de usuario para el que se ha creado el ACEE tenga acceso a los recursos utilizados en la región dependiente.

Si no se especifica ni MQISS_CHECK ni MQISS_FULL para este campo, se asume MQISS_CHECK.

El valor inicial de este campo es MQISS_CHECK.

Reservado (MQCHAR)

Este es un campo reservado. Debe estar en blanco.

MQIMPO - Consultar opciones de propiedades de mensajes

La estructura MQIMPO permite a las aplicaciones especificar opciones que controlan cómo se consultan las propiedades de los mensajes. La estructura es un parámetro de entrada en la llamada MQINQMP.

Disponibilidad

Todos IBM MQ los sistemas y IBM MQ clientes.

Conjunto de caracteres y codificación

Los datos en MQIMPO deben estar en el juego de caracteres de la aplicación y en la codificación de la aplicación (MQENC_NATIVE).

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 514. Campos en MQIMPO		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>StrucId</u> (identificador de estructura)	MQIMPO_STRUC_ID	'IMPO'
<u>Versión</u> (número de versión de la estructura)	MQIMPO_VERSION_1	1
<u>Opciones</u> (opciones que controlan la acción de MQINQMP)	MQIMPO_INQ_PRIMERO	
<u>RequestedEncoding</u> (codificación a la que se debe convertir la propiedad consultada)	MQENC_NATIVE	
<u>RequestedCCSID</u> (conjunto de caracteres de la propiedad consultada)	MQCCSI_APPL	
<u>ReturnedEncoding</u> (codificación del valor devuelto)	MQENC_NATIVE	
<u>ReturnedCCSID</u>	0	
<u>Reserved1</u> (campo reservado)	carácter en blanco (campo de 4 bytes)	
<u>ReturnedName</u> (nombre de la propiedad solicitada)	MQCHARV_DEFAULT	
<u>TypeString</u> (representación en cadena del tipo de datos de la propiedad)	Cadena nula o espacios en blanco	
Notas: 1. El valor «Cadena nula o espacios en blanco» denota la cadena nula en C y los caracteres en blanco en otros lenguajes de programación. 2. En el lenguaje de programación C, la variable macroMQIMPO_DEFAULT contiene los valores que se enumeran en la tabla. Úselo de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQIMPO MyIMPO = {MQIMPO_DEFAULT};</pre>		

Declaraciones lingüísticas

Declaración C para MQIMPO

```
typedef struct tagMQIMPO MQIMPO;  
struct tagMQIMPO {  
    MQCHAR4  StrucId;           /* Structure identifier */  
    MQLONG   Version;          /* Structure version number */  
    MQLONG   Options;          /* Options that control the action of  
                               MQINQMP */  
    MQLONG   RequestedEncoding; /* Requested encoding of Value */  
    MQLONG   RequestedCCSID;    /* Requested character set identifier  
                               of Value */  
    MQLONG   ReturnedEncoding;  /* Returned encoding of Value */  
    MQLONG   ReturnedCCSID;     /* Returned character set identifier  
                               of Value */  
    MQCHAR   Reserved1         /* Reserved field */  
};
```



```

MQCHARV  ReturnedName;      /* Returned property name */
MQCHAR8  TypeString;        /* Property data type as a string */
};

```

COBOL declaración para MQIMPO

```

**  MQIMPO structure
10 MQIMPO.
**    Structure identifier
15 MQIMPO-STRUCID          PIC X(4).
**    Structure version number
15 MQIMPO-VERSION          PIC S9(9) BINARY.
**    Options that control the action of MQINQMP
15 MQIMPO-OPTIONS          PIC S9(9) BINARY.
**    Requested encoding of VALUE
15 MQIMPO-REQUESTEDENCODING PIC S9(9) BINARY.
**    Requested character set identifier of VALUE
15 MQIMPO-REQUESTEDCCSID    PIC S9(9) BINARY.
**    Returned encoding of VALUE
15 MQIMPO-RETURNEDENCODING  PIC S9(9) BINARY.
**    Returned character set identifier of VALUE
15 MQIMPO-RETURNEDCCSID     PIC S9(9) BINARY.
**    Reserved field
15 MQIMPO-RESERVED1
**    Returned property name
15 MQIMPO-RETURNEDNAME.
**    Address of variable length string
20 MQIMPO-RETURNEDNAME-VSPTR POINTER.
**    Offset of variable length string
20 MQIMPO-RETURNEDNAME-VSOFFSET PIC S9(9) BINARY.
**    CCSID of variable length string
20 MQIMPO-RETURNEDNAME-VSCCSID PIC S9(9) BINARY.
**    Property data type as string
15 MQIMPO-TYPESTRING        PIC S9(9) BINARY.

```

PL/I declaración para MQIMPO

```

dcl
1 MQIMPO based,
3 StrucId          char(4),      /* Structure identifier */
3 Version          fixed bin(31), /* Structure version number */
3 Options          fixed bin(31), /* Options that control the
                                   action of MQINQMP */
3 RequestedEncoding fixed bin(31), /* Requested encoding of
                                   Value */
3 RequestedCCSID    fixed bin(31), /* Requested character set
                                   identifier of Value */
3 ReturnedEncoding  fixed bin(31), /* Returned encoding of
                                   Value */
3 ReturnedCCSID     fixed bin(31), /* Returned character set
                                   identifier of Value */
3 Reserved1        fixed bin(31), /* Reserved field */
3 ReturnedName,     /* Returned property name */
5 ReturnedName_VSPtr  pointer,    /* Address of returned
                                   name */
5 5 ReturnedName_VSOffset fixed bin(31), /* Offset of returned
                                   name */
5 5 ReturnedName_VSCCSID fixed bin(31), /* CCSID of returned
                                   name */
3 TypeString        char(8);      /* Property data type as
                                   string */

```

High Level Assembler declaración para MQIMPO

MQIMPO	DSECT		
MQIMPO_STRUCID	DS	CL4	Structure identifier
MQIMPO_VERSION	DS	F	Structure version number
MQIMPO_OPTIONS	DS	F	Options that control the
*			action of MQINQMP
MQIMPO_REQUESTEDENCODING	DS	F	Requested encoding of VALUE
MQIMPO_REQUESTEDCCSID	DS	F	Requested character set
*			identifier of VALUE
MQIMPO_RETURNEDENCODING	DS	F	Returned encoding of VALUE
MQIMPO_RETURNEDCCSID	DS	F	Returned character set
*			identifier of VALUE
MQIMPO_RESERVED1	DS	F	Reserved field
MQIMPO_RETURNEDNAME	DS	0F	Force fullword alignment
MQIMPO_RETURNEDNAME_VSPTR	DS	F	Address of returned name

MQIMPO_RETURNEDNAME_VSOFFSET	DS	F	Offset of returned name
MQIMPO_RETURNEDNAME_VSLENGTH	DS	F	Length of returned name
MQIMPO_RETURNEDNAME_VSCCSID	DS	F	CCSID of returned name
MQIMPO_RETURNEDNAME_LENGTH	EQU	*	MQIMPO_RETURNEDNAME
	ORG		MQIMPO_RETURNEDNAME
MQIMPO_RETURNEDNAME_AREA	DS		CL(MQIMPO_RETURNEDNAME_LENGTH)
*			
MQIMPO_TYPESTRING	DS	CL8	Property data type as string
MQIMPO_LENGTH	EQU	*	MQIMPO
MQIMPO_AREA	DS		CL(MQIMPO_LENGTH)

Detalles de campo para MQIMPO

La estructura de opciones de propiedades de mensajes MQIMPO Inquire permite a las aplicaciones especificar opciones que controlan cómo se consultan las propiedades de los mensajes. La estructura es un parámetro de entrada en la llamada MQINQMP.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 515. Campos en MQIPMO		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
StrucId (identificador de estructura)	MQIMPO_STRUC_ID	'IMPO'
Versión (número de versión de la estructura)	MQIMPO_VERSION_1	1
Opciones (opciones que controlan la acción de MQINQMP)	MQIMPO_INQ_PRIMER O	
RequestedEncoding (codificación a la que se debe convertir la propiedad consultada)	MQENC_NATIVE	
RequestedCCSID (conjunto de caracteres de la propiedad consultada)	MQCCSI_APPL	
ReturnedEncoding (codificación del valor devuelto)	MQENC_NATIVE	
ReturnedCCSID	0	
Reserved1 (campo reservado)	carácter en blanco (campo de 4 bytes)	
ReturnedName (nombre de la propiedad solicitada)	MQCHARV_DEFAULT	
TypeString (representación en cadena del tipo de datos de la propiedad)	Cadena nula o espacios en blanco	
Notas: 1. El valor «Cadena nula o espacios en blanco» denota la cadena nula en C y los caracteres en blanco en otros lenguajes de programación. 2. En el lenguaje de programación C, la variable macroMQIMPO_DEFAULT contiene los valores que se enumeran en la tabla. Úselo de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQIMPO MyIMPO = {MQIMPO_DEFAULT};</pre>		

StrucId (MQCHAR4)

Este es el identificador de estructura de la estructura de opciones de propiedad del mensaje de consulta. Siempre es un campo de entrada. Su valor es MQIMPO_STRUC_ID.

El valor debe ser:

MQIMPO_STRUC_ID

Identificador de la estructura de opciones de la propiedad del mensaje de consulta.

Para el lenguaje de programación C, también se define la constante MQIMPO_STRUC_ID_ARRAY. Tiene el mismo valor que MQIMPO_STRUC_ID, pero es una matriz de caracteres en lugar de una cadena.

Versión (MQLONG)

Estructura de opciones de propiedad del mensaje de consulta: campo Versión.

Este es el número de versión de la estructura. El valor debe ser:

MQIMPO_VERSION_1

Número de versión para la estructura de opciones de propiedades del mensaje de consulta.

La siguiente constante especifica el número de versión de la versión actual:

MQIMPO_VERSIÓN_ACTUAL

Versión actual de la estructura de opciones de propiedades de los mensajes de consulta.

Este campo siempre es un campo de entrada. El valor inicial de este campo es MQIMPO_VERSION_1.

Opciones (MQLONG)

Estructura de opciones de propiedad del mensaje de consulta: campo Opciones.

Las siguientes opciones controlan la acción de MQINQMP. Puede especificar una o varias de estas opciones. Para especificar más de una opción, suma los valores (sin añadir la misma constante más de una vez) o combínalos mediante la operación OR bit a bit (si el lenguaje de programación admite operaciones bit a bit).

Las combinaciones de opciones que no son válidas se indican; todas las demás combinaciones son válidas.

Opciones de datos de valor: Las siguientes opciones se refieren al procesamiento de los datos de valor cuando se recupera la propiedad del mensaje.

MQIMPO_CONVERTIR_VALOR

Esta opción solicita que el valor de la propiedad se convierta para ajustarse a los *RequestedCCSID* valores *RequestedEncoding* y especificados antes de que la llamada MQINQMP devuelva el valor de la propiedad en el *Value* área.

- Si la conversión se realiza correctamente, los campos *ReturnedCCSID* *ReturnedEncoding* y se establecen igual que *RequestedCCSID* y *RequestedEncoding* al regresar de la llamada MQINQMP.
- Si la conversión falla, pero la llamada MQINQMP se completa sin errores, el valor de la propiedad se devuelve sin convertir.

Si la propiedad es una cadena, los campos *ReturnedCCSID* *ReturnedEncoding* y se establecen en el juego de caracteres y la codificación de la cadena sin convertir.

El código de finalización es MQCC_WARNING en este caso, con el código de motivo MQRC_PROP_VALUE_NOT_CONVERTED. El cursor de propiedad avanza hasta la propiedad devuelta.

Si el valor de la propiedad se amplía durante la conversión y supera el tamaño del **Value** parámetro, el valor se devuelve sin convertir, con el código de finalización MQCC_FAILED; el código de motivo se establece en MQRC_PROPERTY_VALUE_TOO_BIG.

El **DataLength** parámetro de la llamada MQINQMP devuelve la longitud a la que se habría convertido el valor de la propiedad, con el fin de permitir que la aplicación determine el tamaño del búfer necesario para alojar el valor de la propiedad convertido. El cursor de propiedad no ha cambiado.

Esta opción también solicita que:

- Si el nombre de la propiedad contiene un comodín y
- El *ReturnedName* campo se inicializa con una dirección o desplazamiento para el nombre devuelto,

A continuación, el nombre devuelto se convierte para ajustarse a los *RequestedCCSID* valores *RequestedEncoding* y.

- Si la conversión se realiza correctamente, el *VSCCSID* campo de *ReturnedName* y la codificación del nombre devuelto se establecen en el valor de entrada de *RequestedCCSID* y *RequestedEncoding*.
- Si la conversión falla, pero la llamada MQINQMP se completa sin errores ni advertencias, el nombre devuelto no se convierte. El código de finalización es MQCC_WARNING en este caso, con el código de motivo MQRC_PROP_NAME_NOT_CONVERTED.

El cursor de propiedad avanza hasta la propiedad devuelta. Se devuelve MQRC_PROP_VALUE_NOT_CONVERTED si ni el valor ni el nombre se han convertido.

Si el nombre devuelto se expande durante la conversión y excede el tamaño del *VSBufsize* campo del *RequestedName*, la cadena devuelta se deja sin convertir, con el código de finalización MQCC_FAILED y el código de motivo establecido en MQRC_PROPERTY_NAME_TOO_BIG.

El *VSLength* campo de la estructura MQCHARV devuelve la longitud a la que se habría convertido el valor de la propiedad, con el fin de permitir que la aplicación determine el tamaño del búfer necesario para alojar el valor de la propiedad convertido. El cursor de propiedad no ha cambiado.

MQIMPO_CONVERTIR_TIPO

Esta opción solicita que el valor de la propiedad se convierta de su tipo de datos actual al tipo de datos especificado en el **Type** parámetro de la llamada MQINQMP.

- Si la conversión se realiza correctamente, el **Type** parámetro no cambia al devolver la llamada MQINQMP.
- Si la conversión falla, pero la llamada MQINQMP se completa sin errores, la llamada falla con el motivo MQRC_PROP_CONV_NOT_SUPPORTED. El cursor de propiedad no ha cambiado.

Si la conversión del tipo de datos provoca que el valor se amplíe durante la conversión y el valor convertido supera el tamaño del **Value** parámetro, el valor se devuelve sin convertir, con el código de finalización MQCC_FAILED y el código de motivo establecido en MQRC_PROPERTY_VALUE_TOO_BIG.

El **DataLength** parámetro de la llamada MQINQMP devuelve la longitud a la que se habría convertido el valor de la propiedad, con el fin de permitir que la aplicación determine el tamaño del búfer necesario para alojar el valor de la propiedad convertido. El cursor de propiedad no ha cambiado.

Si el valor del **Type** parámetro de la llamada MQINQMP no es válido, la llamada falla con el motivo MQRC_PROPERTY_TYPE_ERROR.

Si la conversión de tipo de datos solicitada no es compatible, la llamada falla con el motivo MQRC_PROP_CONV_NOT_SUPPORTED. Se admiten las siguientes conversiones de tipos de datos:

Tabla 516. Conversiones de tipos de datos compatibles	
Tipo de datos de propiedad	Tipos de datos de destino soportados
MQTYPE_BOOLEAN	MQTYPE_STRING, MQTYPE_INT8, MQTYPE_INT16, MQTYPE_INT32, MQTYPE_INT64
MQTYPE_BYTE_STRING	MQTYPE_STRING
MQTYPE_INT8	MQTYPE_STRING, MQTYPE_INT16, MQTYPE_INT32, MQTYPE_INT64
MQTYPE_INT16	MQTYPE_STRING, MQTYPE_INT32, MQTYPE_INT64
MQTYPE_INT32	MQTYPE_STRING, MQTYPE_INT64
MQTYPE_INT64	MQTYPE_STRING
MQTYPE_FLOAT32	MQTYPE_STRING, MQTYPE_FLOAT64

Tabla 516. Conversiones de tipos de datos compatibles (continuación)

Tipo de datos de propiedad	Tipos de datos de destino soportados
MQTYPE_FLOAT64	MQTYPE_STRING
MQTYPE_STRING	MQTYPE_BOOLEAN, MQTYPE_INT8, MQTYPE_INT16, MQTYPE_INT32, MQTYPE_INT64, MQTYPE_FLOAT32, MQTYPE_FLOAT64
MQTYPE_NULL	Ninguna

Las normas generales que rigen las conversiones soportadas son las siguientes:

- Los valores numéricos de las propiedades se pueden convertir de un tipo de datos a otro, siempre que no se pierdan datos durante la conversión.

Por ejemplo, el valor de una propiedad con el tipo de datos MQTYPE_INT32 puede convertirse en un valor con el tipo de datos MQTYPE_INT64, pero no puede convertirse en un valor con el tipo de datos MQTYPE_INT16.

- Un valor de propiedad de cualquier tipo de datos se puede convertir a una serie.
- Un valor de propiedad de serie se puede convertir a cualquier otro tipo de datos, siempre que la serie tenga el formato correcto para la conversión. Si una aplicación intenta convertir un valor de propiedad de cadena que no está formateado correctamente, IBM MQ devuelve el código de motivo MQRC_PROP_NUMBER_FORMAT_ERROR.
- Si una aplicación intenta una conversión que no es compatible, IBM MQ devuelve el código de motivo MQRC_PROP_CONV_NOT_SUPPORTED.

Las reglas específicas para convertir el valor de una propiedad de un tipo de datos a otro son las siguientes:

- Al convertir un valor de propiedad MQTYPE_BOOLEAN en una cadena, el valor TRUE se convierte en la cadena «TRUE» y el valor false se convierte en la cadena «FALSE».
- Al convertir un valor de propiedad MQTYPE_BOOLEAN a un tipo de datos numérico, el valor TRUE se convierte en uno y el valor FALSE se convierte en cero.
- Al convertir el valor de una propiedad de cadena en un valor MQTYPE_BOOLEAN, la cadena «TRUE» o «1» se convierte en TRUE, y la cadena «FALSE» o «0» se convierte en FALSE.

Tenga en cuenta que los términos «TRUE» y «FALSE» no distinguen entre mayúsculas y minúsculas.

No se puede convertir ninguna otra cadena; IBM MQ devuelve el código de motivo MQRC_PROP_NUMBER_FORMAT_ERROR.

- Al convertir el valor de una propiedad de cadena en un valor con el tipo de datos MQTYPE_INT8, MQTYPE_INT16, MQTYPE_INT32 o MQTYPE_INT64, la cadena debe tener el siguiente formato:

```
[blanks][sign]digits
```

El significado de los componentes de la serie es el siguiente:

blanks

Caracteres en blanco opcionales al principio

sign

Un carácter opcional de signo más (+) o signo menos (-).

digits

Una secuencia continua de caracteres de dígitos (0-9). Al menos, debe estar presente un carácter de dígito.

Después de la secuencia de caracteres de dígitos, la serie puede contener otros caracteres que no son caracteres de dígitos, pero la conversión se detiene tan pronto como se llega al primero de estos caracteres. Se da por sentado que la serie representa un entero decimal.

IBM MQ devuelve el código de motivo MQRC_PROP_NUMBER_FORMAT_ERROR si la cadena no tiene el formato correcto.

- Al convertir el valor de una propiedad de cadena en un valor con el tipo de datos MQTYPE_FLOAT32 o MQTYPE_FLOAT64, la cadena debe tener el siguiente formato:

```
[blanks][sign]digits[.digits][e_char[e_sign]e_digits]
```

El significado de los componentes de la serie es el siguiente:

blanks

Caracteres en blanco opcionales al principio

sign

Un carácter opcional de signo más (+) o signo menos (-).

digits

Una secuencia continua de caracteres de dígitos (0-9). Al menos, debe estar presente un carácter de dígito.

e_char

Un carácter exponente, que puede ser «E» o «e».

e_sign

Un signo más (+) o menos (-) opcional para el exponente.

e_digits

Una secuencia contigua de caracteres de dígitos (0-9) para el exponente. Al menos, debe estar presente un carácter de dígito, si la serie contiene un carácter de exponente.

Detrás de la secuencia de caracteres de dígitos, o los caracteres opcionales que representan un exponente, la serie puede contener otros caracteres que no son caracteres de dígitos, pero la conversión se detiene tan pronto como se llega al primero de estos caracteres. Se da por supuesto que la serie representa un número de coma flotante decimal con un exponente que es una potencia de 10.

IBM MQ devuelve el código de motivo MQRC_PROP_NUMBER_FORMAT_ERROR si la cadena no tiene el formato correcto.

- Al convertir un valor de propiedad numérico en una cadena, el valor se convierte en la representación de cadena del valor como número decimal, no en la cadena que contiene el carácter ASCII para ese valor. Por ejemplo, el número entero 65 se convierte en la cadena «65», no en la cadena «A».
- Al convertir el valor de una propiedad de cadena de bytes a una cadena, cada byte se convierte en los dos caracteres hexadecimales que representan el byte. Por ejemplo, la matriz de bytes {0xF1, 0x12, 0x00, 0xFF} se convierte en la cadena "F11200FF".

MQIMPO_LONGITUD_CONSULTA

Consultar el tipo y la longitud del valor de la propiedad. La longitud se devuelve en el **DataLength** parámetro de la llamada MQINQMP. El valor de la propiedad no se devuelve.

Si se especifica un **ReturnedName** búfer, el **VSLength** campo de la estructura MQCHARV se rellena con la longitud del nombre de la propiedad. El nombre de la propiedad no se devuelve.

Opciones de iteración: Las siguientes opciones se refieren a la iteración sobre propiedades, utilizando un nombre con un carácter comodín

MQIMPO_INQ_PRIMERO

Consultar la primera propiedad que coincida con el nombre especificado. Después de esta llamada, se establece un cursor en la propiedad que se devuelve.

Éste es el valor predeterminado.

La opción MQIMPO_INQ_PROP_UNDER_CURSOR se puede utilizar posteriormente con una llamada MQINQMP, si es necesario, para volver a consultar la misma propiedad.

Tenga en cuenta que solo hay un cursor de propiedad; por lo tanto, si cambia el nombre de la propiedad especificado en la llamada MQINQMP, el cursor se restablece.

Esta opción no es válida con ninguna de las siguientes opciones:

MQIMPO_INQ_SIGUIENTE
MQIMPO_INQ_PROP_BAJO_EL_CURSOR

MQIMPO_INQ_SIGUIENTE

Consulta la siguiente propiedad que coincide con el nombre especificado, continuando la búsqueda desde el cursor de la propiedad. El cursor avanza hasta la propiedad que se devuelve.

Si esta es la primera llamada MQINQMP para el nombre especificado, se devuelve la primera propiedad que coincide con el nombre especificado.

La opción MQIMPO_INQ_PROP_UNDER_CURSOR se puede utilizar posteriormente con una llamada MQINQMP si es necesario, para volver a consultar la misma propiedad.

Si la propiedad situada bajo el cursor se ha eliminado, MQINQMP devuelve la siguiente propiedad coincidente después de la que se ha eliminado.

Si se añade una propiedad que coincide con el comodín mientras se está realizando una iteración, es posible que la propiedad se devuelva o no durante la finalización de la iteración. La propiedad se devuelve una vez que se reinicia la iteración utilizando MQIMPO_INQ_FIRST.

Una propiedad que coincida con el comodín que se eliminó, mientras la iteración estaba en curso, no se devuelve después de su eliminación.

Esta opción no es válida con ninguna de las siguientes opciones:

MQIMPO_INQ_PRIMERO
MQIMPO_INQ_PROP_BAJO_EL_CURSOR

MQIMPO_INQ_PROP_BAJO_EL_CURSOR

Recupera el valor de la propiedad apuntada por el cursor de propiedad. La propiedad señalada por el cursor de propiedad es la última consultada, utilizando la opción MQIMPO_INQ_FIRST o MQIMPO_INQ_NEXT.

El cursor de propiedad se restablece cuando se reutiliza el identificador de mensaje, cuando el identificador de mensaje se especifica en el *MsgHandle* campo de MQGMO en una llamada MQGET, o cuando el identificador de mensaje se especifica en los campos *OriginalMsgHandle* o *NewMsgHandle* o de la estructura MQPMO en una llamada MQPUT.

Si se utiliza esta opción cuando aún no se ha establecido el cursor de propiedad, o si se ha eliminado la propiedad a la que apunta el cursor de propiedad, la llamada falla con el código de finalización MQCC_FAILED y el motivo MQRC_PROPERTY_NOT_AVAILABLE.

Esta opción no es válida con ninguna de las siguientes opciones:

MQIMPO_INQ_PRIMERO
MQIMPO_INQ_SIGUIENTE

Si no se requiere ninguna de las opciones descritas anteriormente, se puede utilizar la siguiente opción:

MQIMPO_NINGUNO

Utilice este valor para indicar que no se han especificado otras opciones; todas las opciones toman sus valores predeterminados.

MQIMPO_NONE ayuda a la documentación del programa; no está previsto que esta opción se utilice con ninguna otra, pero como su valor es cero, dicho uso no puede detectarse.

Este campo siempre es un campo de entrada. El valor inicial de este campo es MQIMPO_INQ_FIRST.

RequestedEncoding (MQLONG)

Estructura de opciones de propiedad del mensaje de consulta: campo « RequestedEncoding ».

Esta es la codificación a la que se debe convertir el valor de la propiedad consultada cuando se especifica MQIMPO_CONVERT_VALUE o MQIMPO_CONVERT_TYPE.

El valor inicial de este campo es MQENC_NATIVE.

RequestedCCSID (MQLONG)

Estructura de opciones de propiedad del mensaje de consulta: campo « RequestedCCSID ».

El conjunto de caracteres al que se convertirá el valor de la propiedad consultada si el valor es una cadena de caracteres. Este es también el conjunto de caracteres al que se debe convertir *ReturnedName* el cuando se especifica MQIMPO_CONVERT_VALUE o MQIMPO_CONVERT_TYPE.

El valor inicial de este campo es MQCCSI_APPL.

ReturnedEncoding (MQLONG)

Estructura de opciones de propiedad del mensaje de consulta: campo « ReturnedEncoding ».

En la salida, esta es la codificación del valor devuelto.

Si se especifica la opción MQIMPO_CONVERT_VALUE y la conversión se ha realizado correctamente, el *ReturnedEncoding* campo, al regresar, tendrá el mismo valor que el valor pasado.

El valor inicial de este campo es MQENC_NATIVE.

ReturnedCCSID (MQLONG)

Estructura de opciones de propiedad del mensaje de consulta: campo « ReturnedCCSID ».

En la salida, este es el conjunto de caracteres del valor devuelto si el **Type** parámetro de la llamada MQINQMP es MQTYPE_STRING.

Si se especifica la opción MQIMPO_CONVERT_VALUE y la conversión se ha realizado correctamente, el *ReturnedCCSID* campo, al regresar, tendrá el mismo valor que el valor pasado.

El valor inicial de este campo es cero.

Reserved1 (MQCHAR)

Este es un campo reservado. El valor inicial de este campo es un carácter en blanco (campo de 4 bytes).

ReturnedName (MQCHARV)

Estructura de opciones de propiedad del mensaje de consulta: campo « ReturnedName ».

El nombre real de la propiedad solicitada.

Al introducir datos, se puede pasar un búfer de cadena utilizando el *VSPtr* campo *VSOffset* o de la estructura *MQCHARV*. La longitud del búfer de cadena se especifica utilizando el *VSBufsize* campo de la estructura *MQCHARV*.

Al regresar de la llamada MQINQMP, el búfer de cadena se completa con el nombre de la propiedad que se consultó, siempre que el búfer de cadena fuera lo suficientemente largo como para contener el nombre completo. El *VSLength* campo de la estructura *MQCHARV* se rellena con la longitud del nombre de la propiedad. El *VSCCSID* campo de la estructura *MQCHARV* se rellena para indicar el juego de caracteres del nombre devuelto, independientemente de si la conversión del nombre ha fallado o no.

Este es un campo de entrada/salida. El valor inicial de este campo es MQCHARV_DEFAULT.

TypeString (MQCHAR8)

Estructura de opciones de propiedad del mensaje de consulta: campo « TypeString ».

Una representación de cadena del tipo de datos de la propiedad.

Si la propiedad se especificó en un encabezado MQRFH2 y no se reconoce el atributo dt MQRFH2, este campo se puede utilizar para determinar el tipo de datos de la propiedad. *TypeString* Se devuelve en el conjunto de caracteres codificados 1208 (UTF-8) y son los primeros ocho bytes del valor del dt atributo de la propiedad que no se ha podido reconocer

Este campo siempre es de salida. El valor inicial de este campo es la cadena nula en el lenguaje de programación C y 8 caracteres en blanco en otros lenguajes de programación.

MQMD - Descriptor de mensaje

La estructura MQMD contiene la información de control que acompaña a los datos de la aplicación cuando un mensaje viaja entre las aplicaciones emisora y receptora. La estructura es un parámetro de entrada/salida en las llamadas MQGET, MQPUT y MQPUT1.

Disponibilidad

Todos IBM MQ los sistemas, además de IBM MQ MQI clients los conectados a estos sistemas.

Versión

La versión actual de MQMD es MQMD_VERSION_2. Las aplicaciones que estén diseñadas para ser portables entre varios entornos deben garantizar que la versión requerida de MQMD sea compatible con todos los entornos en cuestión. Los campos que solo existen en las versiones más recientes de la estructura se identifican como tales en las descripciones que siguen.

Los archivos de encabezado, COPY e INCLUDE proporcionados para los lenguajes de programación compatibles contienen la versión más reciente de MQMD que es compatible con el entorno, pero con el valor inicial del *Version* campo establecido en MQMD_VERSION_1. Para utilizar campos que no están presentes en la estructura version-1, la aplicación debe establecer el *Version* campo en el número de versión requerido.

Hay disponible una declaración para la estructura `version-1` con el nombre `MQMD1`.

Conjunto de caracteres y codificación

Los datos en MQMD deben estar en el juego de caracteres y la codificación del gestor de colas local; estos se proporcionan mediante el atributo del **CodedCharSetId** gestor de colas y MQENC_NATIVE. Sin embargo, si la aplicación se ejecuta como un IBM MQ MQI client, la estructura debe estar en el juego de caracteres y en la codificación del cliente.

Si los gestores de colas de envío y recepción utilizan diferentes conjuntos de caracteres o codificaciones, los datos de MQMD se convierten automáticamente. No es necesario que la aplicación convierta el MQMD.

Uso de diferentes versiones de MQMD

Un MQMD « version-2 » (Mensaje de datos de movimiento) es equivalente a utilizar un MQMD « version-1 » (Mensaje de datos de movimiento) y anteponer a los datos del mensaje una estructura MQMDE. Sin embargo, si todos los campos de la estructura MQMDE tienen sus valores predeterminados, se puede omitir MQMDE. version-1 Se utilizan MQMD y MQMDE tal y como se describe:

- En las llamadas MQPUT y MQPUT1, si la aplicación proporciona un MQMD version-1, la aplicación puede, opcionalmente, anteponer a los datos del mensaje un MQMDE, estableciendo el *Format* campo en MQMD en MQFMT_MD_EXTENSION para indicar que hay un MQMDE presente. Si la aplicación no proporciona un MQMDE, el gestor de colas asume los valores predeterminados para los campos del MQMDE.

Nota: Varios de los campos que existen en el MQMD version-2 pero no en el MQMD version-1 son campos de entrada/salida en las llamadas MQPUT y MQPUT1. Sin embargo, el gestor de colas no devuelve ningún valor en los campos equivalentes del MQMDE en la salida de las llamadas MQPUT y MQPUT1; si la aplicación requiere esos valores de salida, debe utilizar un MQMD version-2.

- En la llamada MQGET, si la aplicación proporciona un MQMD version-1, el gestor de colas antepone al mensaje devuelto un MQMDE, pero solo si uno o varios de los campos del MQMDE tienen un valor distinto del predeterminado. El *Format* campo en MQMD tendrá el valor MQFMT_MD_EXTENSION para indicar que hay un MQMDE presente.

Los valores predeterminados que utiliza el gestor de colas para los campos del MQMDE son los mismos que los valores iniciales de dichos campos, que se muestran en [Tabla 522](#) en la [página 448](#).

Cuando un mensaje se encuentra en una cola de transmisión, algunos de los campos de MQMD se establecen en valores concretos; consulte [“MQXQH - Cabecera de cola de transmisión”](#) en la [página 618](#) para obtener más detalles.

Contexto de mensaje

Algunos campos de MQMD contienen el contexto del mensaje. Hay dos tipos de contexto de mensaje: *contexto de identidad* y *contexto de origen*. Por norma, se utiliza para:

- El contexto de identidad se refiere a la aplicación que envió *originalmente* el mensaje
- El contexto de origen se refiere a la aplicación que envió el mensaje *más recientemente*.

Estas dos aplicaciones pueden ser la misma aplicación, pero también pueden ser aplicaciones diferentes (por ejemplo, cuando se reenvía un mensaje de una aplicación a otra).

Aunque el contexto de identidad y origen suele tener los significados descritos, el contenido de ambos tipos de campos de contexto en MQMD depende de las opciones MQPMO_*_CONTEXT que se especifican cuando se envía el mensaje. Como resultado, el contexto de identidad no se relaciona necesariamente con la aplicación que envió originalmente el mensaje, y el contexto de origen no se relaciona necesariamente con la aplicación que envió el mensaje más recientemente; depende del diseño del conjunto de aplicaciones.

El agente del canal de mensajes (MCA) nunca altera el contexto del mensaje. Los MCA que reciben mensajes de gestores de colas remotos utilizan la opción de contexto MQPMO_SET_ALL_CONTEXT en la llamada MQPUT o MQPUT1. Esto permite al MCA receptor conservar exactamente el contexto del mensaje que viajó con el mensaje desde el MCA emisor. Sin embargo, el resultado es que el contexto de origen no se relaciona con ninguno de los MCA que enviaron y recibieron el mensaje. El contexto de origen se refiere a una aplicación anterior que envió el mensaje. Si todas las aplicaciones intermedias han pasado el contexto del mensaje, el contexto de origen se refiere a la propia aplicación de origen.

En las descripciones, los campos de contexto se describen como si se utilizaran tal y como se ha explicado anteriormente. Para obtener más información sobre el contexto del mensaje, consulte [Contexto de mensaje](#).

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 517. Campos en MQMD para MQMD		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
StrucId (identificador de estructura)	MQMD_STRUC_ID	'MD'
Versión (número de versión de la estructura)	MQMD_VERSION_1	1
Informe (opciones para los mensajes de informe)	MQRO_NONE	0
MsgType (tipo de mensaje)	MQMT_DATAGRAM	8
MQMD - Campo de caducidad (vida útil del mensaje)	MQEI_UNLIMITED	-1
MQMD - Campo de comentarios (comentarios o código de motivo)	MQFB_NONE	0

Tabla 517. Campos en MQMD para MQMD (continuación)

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>Codificación</u> (codificación numérica de los datos del mensaje)	MQENC_NATIVE	Depende del entorno
<u>CodedCharSetId</u> (identificador del conjunto de caracteres de los datos del mensaje)	MQCCSI_Q_MGR	0
<u>Formato</u> (nombre del formato de los datos del mensaje)	MQFMT_NONE	Espacios en blanco
<u>Prioridad</u> (prioridad del mensaje)	MQPRI_PRIORITY_AS_Q_DEF	-1
<u>Persistencia</u> (persistencia de mensajes)	MQPER_PERSISTENCE_AS_Q_DEF	2
<u>MQMD - Campo « MsgId »</u> (identificador de mensaje)	MQMI_NONE	Nulos
<u>CorrelId</u> (identificador de correlación)	MQCI_NONE	Nulos
<u>BackoutCount</u> (contador de retroceso)	Ninguna	0
<u>ReplyToQ</u> (nombre de la cola de respuestas)	Ninguna	Cadena nula o espacios en blanco
<u>ReplyToQMgr</u> (nombre del gestor de colas de respuesta)	Ninguna	Cadena nula o espacios en blanco
<u>UserIdentifier</u> (identificador de usuario)	Ninguna	Cadena nula o espacios en blanco
<u>AccountingToken</u> (ficha contable)	MQACT_NONE	Nulos
<u>ApplIdentityData</u> (datos de solicitud relacionados con la identidad)	Ninguna	Cadena nula o espacios en blanco
<u>PutApplType</u> (tipo de aplicación que envió el mensaje)	MQAT_SIN_CONTEXTO	0
<u>PutApplName</u> (nombre de la aplicación que envió el mensaje)	Ninguna	Cadena nula o espacios en blanco
<u>PutDate</u> (fecha en que se publicó el mensaje)	Ninguna	Cadena nula o espacios en blanco
<u>PutTime</u> (hora en que se envió el mensaje)	Ninguna	Cadena nula o espacios en blanco
<u>ApplOriginData</u> (datos de la solicitud relativos al origen)	Ninguna	Cadena nula o espacios en blanco
Nota: Los campos restantes se ignoran si <i>Version</i> es menor que MQMD_VERSION_2.		
<u>GroupId</u> (identificador de grupo)	MQGI_NONE	Nulos
<u>MsgSeqNumber</u> (número de secuencia del mensaje lógico dentro del grupo)	Ninguna	1
<u>Desplazamiento</u> (desplazamiento de los datos en el mensaje físico desde el inicio del mensaje lógico)	Ninguna	0

Tabla 517. Campos en MQMD para MQMD (continuación)

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
MQMD - Campo « MsgFlags » (indicadores de mensajes)	MQMF_NINGUNO	0
OriginalLength (longitud del mensaje original)	MQOL_INDETERMINAD 0	-1
Notas: 1. El valor «Cadena nula o espacios en blanco» denota la cadena nula en C y los caracteres en blanco en otros lenguajes de programación. 2. En el lenguaje de programación C, la variable macroMQMD_DEFAULT contiene los valores que se enumeran en la tabla. Se puede utilizar de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQMD MyMD = {MQMD_DEFAULT};</pre>		

Declaraciones lingüísticas

Declaración C para MQMD

```
typedef struct tagMQMD MQMD;
struct tagMQMD {
    MQCHAR4   StrucId;           /* Structure identifier */
    MQLONG    Version;          /* Structure version number */
    MQLONG    Report;           /* Options for report messages */
    MQLONG    MessageType;      /* Message type */
    MQLONG    Expiry;           /* Message lifetime */
    MQLONG    Feedback;         /* Feedback or reason code */
    MQLONG    Encoding;         /* Numeric encoding of message data */
    MQLONG    CodedCharSetId;    /* Character set identifier of message
                                data */
    MQCHAR8   Format;           /* Format name of message data */
    MQLONG    Priority;          /* Message priority */
    MQLONG    Persistence;      /* Message persistence */
    MQBYTE24  MsgId;            /* Message identifier */
    MQBYTE24  CorrelId;         /* Correlation identifier */
    MQLONG    BackoutCount;     /* Backout counter */
    MQCHAR48  ReplyToQ;         /* Name of reply queue */
    MQCHAR48  ReplyToQMgr;      /* Name of reply queue manager */
    MQCHAR12  UserIdentifier;    /* User identifier */
    MQBYTE32  AccountingToken;  /* Accounting token */
    MQCHAR32  ApplIdentityData; /* Application data relating to
                                identity */
    MQLONG    PutApplType;       /* Type of application that put the
                                message */
    MQCHAR28  PutApplName;      /* Name of application that put the
                                message */
    MQCHAR8   PutDate;          /* Date when message was put */
    MQCHAR8   PutTime;          /* Time when message was put */
    MQCHAR4   ApplOriginData;   /* Application data relating to origin */
    MQBYTE24  GroupId;          /* Group identifier */
    MQLONG    MsgSeqNumber;     /* Sequence number of logical message
                                within group */
    MQLONG    Offset;           /* Offset of data in physical message
                                from start of logical message */
    MQLONG    MsgFlags;         /* Message flags */
    MQLONG    OriginalLength;    /* Length of original message */
};
```

COBOL declaración para MQMD

```
** MQMD structure
10 MQMD.
** Structure identifier
15 MQMD-STRUCID PIC X(4).
** Structure version number
```

```

15 MQMD-VERSION          PIC S9(9) BINARY.
** Options for report messages
15 MQMD-REPORT           PIC S9(9) BINARY.
** Message type
15 MQMD-MSGTYPE          PIC S9(9) BINARY.
** Message lifetime
15 MQMD-EXPIRY           PIC S9(9) BINARY.
** Feedback or reason code
15 MQMD-FEEDBACK         PIC S9(9) BINARY.
** Numeric encoding of message data
15 MQMD-ENCODING         PIC S9(9) BINARY.
** Character set identifier of message data
15 MQMD-CODEDCHARSETID   PIC S9(9) BINARY.
** Format name of message data
15 MQMD-FORMAT           PIC X(8).
** Message priority
15 MQMD-PRIORITY         PIC S9(9) BINARY.
** Message persistence
15 MQMD-PERSISTENCE      PIC S9(9) BINARY.
** Message identifier
15 MQMD-MSGID            PIC X(24).
** Correlation identifier
15 MQMD-CORRELID         PIC X(24).
** Backout counter
15 MQMD-BACKOUTCOUNT    PIC S9(9) BINARY.
** Name of reply queue
15 MQMD-REPLYTOQ         PIC X(48).
** Name of reply queue manager
15 MQMD-REPLYTOQMGR      PIC X(48).
** User identifier
15 MQMD-USERIDENTIFIER   PIC X(12).
** Accounting token
15 MQMD-ACCOUNTINGTOKEN  PIC X(32).
** Application data relating to identity
15 MQMD-APPLIDENTITYDATA PIC X(32).
** Type of application that put the message
15 MQMD-PUTAPPLTYPE      PIC S9(9) BINARY.
** Name of application that put the message
15 MQMD-PUTAPPLNAME      PIC X(28).
** Date when message was put
15 MQMD-PUTDATE          PIC X(8).
** Time when message was put
15 MQMD-PUTTIME          PIC X(8).
** Application data relating to origin
15 MQMD-APPLORIGINDATA   PIC X(4).
** Group identifier
15 MQMD-GROUPID          PIC X(24).
** Sequence number of logical message within group
15 MQMD-MSGSEQNUMBER     PIC S9(9) BINARY.
** Offset of data in physical message from start of logical message
15 MQMD-OFFSET           PIC S9(9) BINARY.
** Message flags
15 MQMD-MSGFLAGS         PIC S9(9) BINARY.
** Length of original message
15 MQMD-ORIGINALLENGTH   PIC S9(9) BINARY.

```

PL/I declaración para MQMD

```

dcl
1 MQMD based,
3 StructId          char(4),          /* Structure identifier */
3 Version           fixed bin(31),    /* Structure version number */
3 Report            fixed bin(31),    /* Options for report messages */
3 MsgType           fixed bin(31),    /* Message type */
3 Expiry            fixed bin(31),    /* Message lifetime */
3 Feedback          fixed bin(31),    /* Feedback or reason code */
3 Encoding          fixed bin(31),    /* Numeric encoding of message
                                     data */
3 CodedCharSetId    fixed bin(31),    /* Character set identifier of
                                     message data */
3 Format            char(8),           /* Format name of message data */
3 Priority           fixed bin(31),    /* Message priority */
3 Persistence       fixed bin(31),    /* Message persistence */
3 MsgId            char(24),          /* Message identifier */
3 CorrelId          char(24),          /* Correlation identifier */
3 BackoutCount      fixed bin(31),    /* Backout counter */
3 ReplyToQ          char(48),         /* Name of reply queue */
3 ReplyToQMgr       char(48),         /* Name of reply queue manager */
3 UserIdentifier    char(12),         /* User identifier */
3 AccountingToken   char(32),         /* Accounting token */

```

```

3 ApplIdentityData char(32),      /* Application data relating to
3 PutApplType      fixed bin(31), /* identity */
                                /* Type of application that put the
3 PutApplName      char(28),      /* message */
                                /* Name of application that put the
3 PutDate          char(8),       /* message */
3 PutTime          char(8),       /* Date when message was put */
3 ApplOriginData   char(4),       /* Time when message was put */
                                /* Application data relating to
3 GroupId          char(24),      /* origin */
3 MsgSeqNumber     fixed bin(31), /* Group identifier */
                                /* Sequence number of logical
3 Offset          fixed bin(31), /* message within group */
                                /* Offset of data in physical
                                /* message from start of logical
3 MsgFlags         fixed bin(31), /* message */
3 OriginalLength   fixed bin(31); /* Message flags */
                                /* Length of original message */

```

High Level Assembler declaración para MQMD

```

MQMD
MQMD_STRUCID      DS CL4  Structure identifier
MQMD_VERSION      DS F    Structure version number
MQMD_REPORT       DS F    Options for report messages
MQMD_MSGTYPE      DS F    Message type
MQMD_EXPIRY       DS F    Message lifetime
MQMD_FEEDBACK     DS F    Feedback or reason code
MQMD_ENCODING     DS F    Numeric encoding of message data
MQMD_CODECHARSETID DS F    Character set identifier of message
*
MQMD_FORMAT       DS CL8  Format name of message data
MQMD_PRIORITY     DS F    Message priority
MQMD_PERSISTENCE  DS F    Message persistence
MQMD_MSGID        DS XL24 Message identifier
MQMD_CORRELID     DS XL24 Correlation identifier
MQMD_BACKOUTCOUNT DS F    Backout counter
MQMD_REPLYTOQ     DS CL48 Name of reply queue
MQMD_REPLYTOQMGR  DS CL48 Name of reply queue manager
MQMD_USERIDENTIFIER DS CL12 User identifier
MQMD_ACCOUNTINGTOKEN DS XL32 Accounting token
MQMD_APPLIDENTITYDATA DS CL32 Application data relating to identity
MQMD_PUTAPPLTYPE  DS F    Type of application that put the
*
MQMD_PUTAPPLNAME  DS CL28 Name of application that put the
*
MQMD_PUTDATE      DS CL8  Date when message was put
MQMD_PUTTIME      DS CL8  Time when message was put
MQMD_APPLORIGINDATA DS CL4 Application data relating to origin
MQMD_GROUPID      DS XL24 Group identifier
MQMD_MSGSEQNUMBER DS F    Sequence number of logical message
*
MQMD_OFFSET       DS F    Offset of data in physical message
*
MQMD_MSGFLAGS     DS F    Message flags
MQMD_ORIGINALLENGTH DS F    Length of original message
*
MQMD_LENGTH       EQU *-MQMD
ORG MQMD
MQMD_AREA         DS CL(MQMD_LENGTH)

```

Declaración de Visual Basic para MQMD

```

Type MQMD
  StrucId      As String*4 'Structure identifier'
  Version      As Long     'Structure version number'
  Report       As Long     'Options for report messages'
  MsgType      As Long     'Message type'
  Expiry       As Long     'Message lifetime'
  Feedback     As Long     'Feedback or reason code'
  Encoding     As Long     'Numeric encoding of message data'
  CodedCharSetId As Long   'Character set identifier of message'
  'data'
  Format       As String*8 'Format name of message data'
  Priority     As Long     'Message priority'
  Persistence  As Long     'Message persistence'
  MsgId       As MQBYTE24 'Message identifier'
  CorrelId    As MQBYTE24 'Correlation identifier'
  BackoutCount As Long     'Backout counter'

```

```

ReplyToQ      As String*48 'Name of reply queue'
ReplyToQMgr   As String*48 'Name of reply queue manager'
UserIdentifier As String*12 'User identifier'
AccountingToken As MQBYTE32 'Accounting token'
ApplIdentityData As String*32 'Application data relating to identity'
PutApplType    As Long      'Type of application that put the'
                                'message'
PutApplName    As String*28 'Name of application that put the'
                                'message'
PutDate        As String*8  'Date when message was put'
PutTime        As String*8  'Time when message was put'
ApplOriginData As String*4  'Application data relating to origin'
GroupId        As MQBYTE24  'Group identifier'
MsgSeqNumber   As Long      'Sequence number of logical message'
                                'within group'
Offset         As Long      'Offset of data in physical message'
                                'from start of logical message'
MsgFlags       As Long      'Message flags'
OriginalLength As Long      'Length of original message'
End Type

```

Detalles de campo para MQMD

La estructura del descriptor de mensajes MQMD contiene la información de control que acompaña a los datos de la aplicación cuando un mensaje viaja entre las aplicaciones emisora y receptora. La estructura es un parámetro de entrada/salida en las llamadas MQGET, MQPUT y MQPUT1.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 518. Campos en MQMD para MQMD		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>StrucId</u> (identificador de estructura)	MQMD_STRUC_ID	'MD'
<u>Versión</u> (número de versión de la estructura)	MQMD_VERSION_1	1
<u>Informe</u> (opciones para los mensajes de informe)	MQRO_NONE	0
<u>MsgType</u> (tipo de mensaje)	MQMT_DATAGRAM	8
<u>Caducidad</u> (vida útil del mensaje)	MQEI_UNLIMITED	-1
<u>Comentarios</u> (comentarios o código de motivo)	MQFB_NONE	0
<u>Codificación</u> (codificación numérica de los datos del mensaje)	MQENC_NATIVE	Depende del entorno
<u>CodedCharSetId</u> (identificador del conjunto de caracteres de los datos del mensaje)	MQCCSI_Q_MGR	0
<u>Formato</u> (nombre del formato de los datos del mensaje)	MQFMT_NONE	Espacios en blanco
<u>Prioridad</u> (prioridad del mensaje)	MQPRI_PRIORITY_AS_Q_DEF	-1
<u>Persistencia</u> (persistencia de mensajes)	MQPER_PERSISTENCE_AS_Q_DEF	2
<u>MsgId</u> (identificador del mensaje)	MQMI_NONE	Nulos
<u>CorrelId</u> (identificador de correlación)	MQCI_NONE	Nulos
<u>BackoutCount</u> (contador de retroceso)	Ninguna	0

Tabla 518. Campos en MQMD para MQMD (continuación)

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>ReplyToQ</u> (nombre de la cola de respuestas)	Ninguna	Cadena nula o espacios en blanco
<u>ReplyToQMgr</u> (nombre del gestor de colas de respuesta)	Ninguna	Cadena nula o espacios en blanco
<u>UserIdentifier</u> (identificador de usuario)	Ninguna	Cadena nula o espacios en blanco
<u>AccountingToken</u> (ficha contable)	MQACT_NONE	Nulos
<u>AppIdentityData</u> (datos de solicitud relacionados con la identidad)	Ninguna	Cadena nula o espacios en blanco
<u>PutApplType</u> (tipo de aplicación que envió el mensaje)	MQAT_SIN_CONTEXTO	0
<u>PutApplName</u> (nombre de la aplicación que envió el mensaje)	Ninguna	Cadena nula o espacios en blanco
<u>PutDate</u> (fecha en que se publicó el mensaje)	Ninguna	Cadena nula o espacios en blanco
<u>PutTime</u> (hora en que se envió el mensaje)	Ninguna	Cadena nula o espacios en blanco
<u>ApplOriginData</u> (datos de la solicitud relativos al origen)	Ninguna	Cadena nula o espacios en blanco
Nota: Los campos restantes se ignoran si <i>Version</i> es menor que MQMD_VERSION_2.		
<u>GroupId</u> (identificador de grupo)	MQGI_NONE	Nulos
<u>MsgSeqNumber</u> (número de secuencia del mensaje lógico dentro del grupo)	Ninguna	1
<u>Desplazamiento</u> (desplazamiento de los datos en el mensaje físico desde el inicio del mensaje lógico)	Ninguna	0
<u>MsgFlags</u> (indicadores de mensajes)	MQMF_NINGUNO	0
<u>OriginalLength</u> (longitud del mensaje original)	MQOL_INDETERMINADO	-1
Notas: 1. El valor «Cadena nula o espacios en blanco» denota la cadena nula en C y los caracteres en blanco en otros lenguajes de programación. 2. En el lenguaje de programación C, la variable macroMQMD_DEFAULT contiene los valores que se enumeran en la tabla. Se puede utilizar de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQMD MyMD = {MQMD_DEFAULT};</pre>		

StrucId (MQCHAR4)

Este es el identificador de estructura de la estructura descriptora del mensaje. Siempre es un campo de entrada. Su valor es MQMD_STRUC_ID.

El valor debe ser:

MQMD_STRUC_ID

Identificador de la estructura del descriptor del mensaje.

Para el lenguaje de programación C, también se define la constante MQMD_STRUC_ID_ARRAY. Tiene el mismo valor que MQMD_STRUC_ID, pero es una matriz de caracteres en lugar de una cadena.

Versión (MQLONG)

Este es el número de versión de la estructura y debe ser uno de los siguientes:

MQMD_VERSION_1

Version-1 Estructura del descriptor de mensajes.

Esta versión es compatible con todos los entornos.

MQMD_VERSION_2

Version-2 Estructura del descriptor de mensajes.

Esta versión es compatible con todos los entornos IBM MQV6.0 y posteriores, además de IBM MQ MQI clients estar conectada a estos sistemas.

Nota: Cuando se utiliza un MQMD (version-2), el gestor de colas realiza comprobaciones adicionales en cualquier estructura de encabezado MQPUT (MQ) que pueda estar presente al principio de los datos del mensaje de la aplicación; para obtener más detalles, consulte las notas de uso de la llamada MQPUT.

Los campos que solo existen en la versión más reciente de la estructura se identifican como tales en las descripciones de los campos. La siguiente constante especifica el número de versión de la versión actual:

MQMD_VERSION_ACTUAL

Versión actual de la estructura del descriptor de mensajes.

Este campo siempre es un campo de entrada. El valor inicial de este campo es MQMD_VERSION_1.

Report (MQLONG)

Un mensaje de informe es un mensaje sobre otro mensaje, utilizado para informar a una aplicación sobre eventos esperados o inesperados relacionados con el mensaje original. El *Report* campo permite a la aplicación que envía el mensaje original especificar qué mensajes de informe se requieren, si los datos del mensaje de la aplicación deben incluirse en ellos y también (tanto para los informes como para las respuestas) cómo deben configurarse los identificadores de mensaje y correlación en el mensaje de informe o respuesta. Se puede solicitar cualquiera o todos (o ninguno) de los siguientes tipos de mensajes de informe:

- Excepción
- Caducidad
- Confirmar a la llegada (COA)
- Confirmar en el momento de la entrega (COD)
- notificación de acción positiva (PAN)
- notificación de acción negativa (NAN)

Puede especificar una o varias de estas opciones. Para especificar más de una opción, suma los valores (sin añadir la misma constante más de una vez) o combínalos mediante la operación OR bit a bit (si el lenguaje de programación admite operaciones bit a bit).

La aplicación que recibe el mensaje de informe puede determinar el motivo por el que se generó el informe examinando el *Feedback* campo en el MQMD; consulte el *Feedback* campo para obtener más detalles.

El uso de opciones de informe al enviar un mensaje a un tema puede provocar que se generen y envíen a la aplicación cero, uno o muchos mensajes de informe. Esto se debe a que el mensaje de publicación puede enviarse a cero, una o muchas aplicaciones suscritas.

Opciones de excepción

Especifique una de las opciones enumeradas para solicitar un mensaje de informe de excepción.

MQRO_EXCEPTION

Un agente de canal de mensajes genera este tipo de informe cuando se envía un mensaje a otro gestor de colas y el mensaje no se puede entregar a la cola de destino especificada. Por ejemplo, la cola de destino o una cola de transmisión intermedia podrían estar llenas, o el mensaje podría ser demasiado grande para la cola.

La generación del mensaje del informe de excepción depende de la persistencia del mensaje original y de la velocidad del canal de mensajes (normal o rápido) por el que viaja el mensaje original:

- Para todos los mensajes persistentes y para los mensajes no persistentes que viajan a través de canales de mensajes normales, el informe de excepciones solo se genera si la acción especificada por la aplicación de envío para la condición de error se puede completar correctamente. La aplicación emisora puede especificar una de las siguientes acciones para controlar la disposición del mensaje original cuando se produce la condición de error:
 - MQRO_DEAD_LETTER_Q (esto coloca el mensaje original en la cola de mensajes no entregados).
 - MQRO_DISCARD_MSG (esto descarta el mensaje original).

Si la acción especificada por la aplicación emisora no se puede completar correctamente, el mensaje original permanece en la cola de transmisión y no se genera ningún mensaje de informe de excepción.

- En el caso de los mensajes no persistentes que viajan a través de canales de mensajes rápidos, el mensaje original se elimina de la cola de transmisión y se genera el informe de excepción, *incluso* si la acción especificada para la condición de error no se puede completar correctamente. Por ejemplo, si se especifica MQRO_DEAD_LETTER_Q, pero el mensaje original no se puede colocar en la cola de mensajes no entregados porque dicha cola está llena, se genera el mensaje de informe de excepción y se descarta el mensaje original.

Para obtener más información sobre los canales de mensajes normales y rápidos, consulte [Velocidad de mensajes no persistentes \(NPMSPEED\)](#).

No se genera un informe de excepción si la aplicación que envió el mensaje original puede ser notificada de forma sincrónica del problema mediante el código de motivo devuelto por la llamada MQPUT o MQPUT1.

Las aplicaciones también pueden enviar informes de excepción para indicar que un mensaje no se puede procesar (por ejemplo, porque se trata de una transacción de débito que haría que la cuenta superara su límite de crédito).

Los datos del mensaje original no se incluyen en el mensaje del informe.

No especifique más de uno de los siguientes: MQRO_EXCEPTION, MQRO_EXCEPTION_WITH_DATA y MQRO_EXCEPTION_WITH_FULL_DATA.

MQRO_EXCEPTION_WITH_DATA

Es igual que MQRO_EXCEPTION, salvo que los primeros 100 bytes de los datos del mensaje de la aplicación del mensaje original se incluyen en el mensaje del informe. Si el mensaje original contiene una o más estructuras de encabezado MQ, estas se incluyen en el mensaje de informe, además de los 100 bytes de datos de la aplicación.

No especifique más de uno de los siguientes: MQRO_EXCEPTION, MQRO_EXCEPTION_WITH_DATA y MQRO_EXCEPTION_WITH_FULL_DATA.

MQRO_EXCEPTION_WITH_FULL_DATA

Informes de excepciones con todos los datos necesarios.

Es igual que MQRO_EXCEPTION, salvo que todos los datos del mensaje de aplicación del mensaje original se incluyen en el mensaje de informe.

No especifique más de uno de los siguientes: MQRO_EXCEPTION, MQRO_EXCEPTION_WITH_DATA y MQRO_EXCEPTION_WITH_FULL_DATA.

Opciones de vencimiento

Especifique una de las opciones enumeradas para solicitar un mensaje de informe de caducidad.

MQRO_EXPIRATION

Este tipo de informe es generado por el gestor de colas si el mensaje se descarta antes de su entrega a una aplicación porque ha superado su tiempo de caducidad (véase el *Expiry* campo). Si esta opción no está configurada, no se generará ningún mensaje de informe si se descarta un mensaje por este motivo (incluso si se especifica una de las opciones MQRO_EXCEPTION_*).

Los datos del mensaje original no se incluyen en el mensaje del informe.

No especifique más de uno de los siguientes: MQRO_EXPIRATION, MQRO_EXPIRATION_WITH_DATA y MQRO_EXPIRATION_WITH_FULL_DATA.

MQRO_EXPIRATION_WITH_DATA

Es igual que MQRO_EXPIRATION, salvo que los primeros 100 bytes de los datos del mensaje de aplicación del mensaje original se incluyen en el mensaje de informe. Si el mensaje original contiene una o más estructuras de encabezado MQ, estas se incluyen en el mensaje de informe, además de los 100 bytes de datos de la aplicación.

No especifique más de uno de los siguientes: MQRO_EXPIRATION, MQRO_EXPIRATION_WITH_DATA y MQRO_EXPIRATION_WITH_FULL_DATA.

MQRO_EXPIRATION_WITH_FULL_DATA

Es igual que MQRO_EXPIRATION, salvo que todos los datos del mensaje de aplicación del mensaje original se incluyen en el mensaje de informe.

No especifique más de uno de los siguientes: MQRO_EXPIRATION, MQRO_EXPIRATION_WITH_DATA y MQRO_EXPIRATION_WITH_FULL_DATA.

Opciones de confirmación a la llegada

Especifique una de las opciones enumeradas para solicitar un mensaje de informe de confirmación a la llegada.

MQRO_COA

Este tipo de informe lo genera el gestor de colas propietario de la cola de destino cuando el mensaje se coloca en la cola de destino. Los datos del mensaje original no se incluyen en el mensaje del informe.

Si el mensaje se incluye como parte de una unidad de trabajo y la cola de destino es una cola local, el mensaje de informe COA generado por el gestor de colas solo se puede recuperar si se ha confirmado la unidad de trabajo.

No se genera un informe COA si el *Format* campo del descriptor del mensaje es MQFMT_XMIT_Q_HEADER o MQFMT_DEAD_LETTER_HEADER. Esto evita que se genere un informe COA si el mensaje se coloca en una cola de transmisión o si no se puede entregar y se coloca en una cola de mensajes no entregados.

En el caso de una cola de puente e IMS, el informe COA se genera cuando el mensaje llega a IMS la cola (se recibe la confirmación de IMS) y no cuando el mensaje se coloca en la cola de puente e MQ. Esto significa que si no IMS está activo, no se genera ningún informe COA hasta que IMS se inicia y se pone un mensaje en la IMS cola.

El usuario que ejecuta un programa que envía un mensaje con MQMD.Report =MQRO_COA debe tener la autoridad +passid en la cola de respuesta. Si el usuario no tiene la autoridad +passid, el mensaje del informe COA no llega a la cola de respuesta. Se intenta colocar el mensaje del informe en la cola de mensajes no entregados.

No especifique más de uno de MQRO_COA, MQRO_COA_WITH_DATA y MQRO_COA_WITH_FULL_DATA.

MQRO_COA_WITH_DATA

Es igual que MQRO_COA, salvo que los primeros 100 bytes de los datos del mensaje de aplicación del mensaje original se incluyen en el mensaje de informe. Si el mensaje original contiene una o más estructuras de encabezado MQ, estas se incluyen en el mensaje de informe, además de los 100 bytes de datos de la aplicación.

No especifique más de uno de MQRO_COA, MQRO_COA_WITH_DATA y MQRO_COA_WITH_FULL_DATA.

MQRO_COA_WITH_FULL_DATA

Es igual que MQRO_COA, salvo que todos los datos del mensaje de aplicación del mensaje original se incluyen en el mensaje de informe.

No especifique más de uno de MQRO_COA, MQRO_COA_WITH_DATA y MQRO_COA_WITH_FULL_DATA.

Opciones de confirmación en el momento de la entrega

Especifique una de las opciones enumeradas para solicitar un mensaje de informe de confirmación de entrega.

MQRO_COD

Este tipo de informe es generado por el gestor de colas cuando una aplicación recupera el mensaje de la cola de destino de tal forma que se elimina el mensaje de la cola. Los datos del mensaje original no se incluyen en el mensaje del informe.

Si el mensaje se recupera como parte de una unidad de trabajo, el mensaje de informe se genera dentro de la misma unidad de trabajo, por lo que el informe no estará disponible hasta que se confirme la unidad de trabajo. Si se cancela la unidad de trabajo, no se envía el informe.

No siempre se genera un informe COD si se recupera un mensaje con la opción MQGMO_MARK_SKIP_BACKOUT. Si se cancela la unidad de trabajo principal pero se confirma la unidad de trabajo secundaria, el mensaje se elimina de la cola, pero no se genera un informe COD.

No se genera un informe COD si el *Format* campo del descriptor del mensaje es MQFMT_DEAD_LETTER_HEADER. Esto evita que se genere un informe COD si el mensaje no se puede entregar y se coloca en una cola de mensajes no entregados.

MQRO_COD no es válido si la cola de destino es una cola XCF.

No especifique más de uno de MQRO_COD, MQRO_COD_WITH_DATA y MQRO_COD_WITH_FULL_DATA.

MQRO_COD_WITH_DATA

Es igual que MQRO_COD, salvo que los primeros 100 bytes de los datos del mensaje de aplicación del mensaje original se incluyen en el mensaje del informe. Si el mensaje original contiene una o más estructuras de encabezado MQ, estas se incluyen en el mensaje de informe, además de los 100 bytes de datos de la aplicación.

Si se especifica MQGMO_ACCEPT_TRUNCATED_MSG en la llamada MQGET para el mensaje original y el mensaje recuperado está truncado, la cantidad de datos del mensaje de la aplicación que se coloca en el mensaje del informe depende del entorno:

- En z/OS, es el mínimo de:
 - La longitud del mensaje original
 - La longitud del búfer utilizado para recuperar el mensaje
 - 100 bytes.
- En otros entornos, es el mínimo de:
 - La longitud del mensaje original
 - 100 bytes.

MQRO_COD_WITH_DATA no es válido si la cola de destino es una cola XCF.

No especifique más de uno de MQRO_COD, MQRO_COD_WITH_DATA y MQRO_COD_WITH_FULL_DATA.

MQRO_COD_WITH_FULL_DATA

Es igual que MQRO_COD, salvo que todos los datos del mensaje de aplicación del mensaje original se incluyen en el mensaje de informe.

MQRO_COD_WITH_FULL_DATA no es válido si la cola de destino es una cola XCF.

No especifique más de uno de MQRO_COD, MQRO_COD_WITH_DATA y MQRO_COD_WITH_FULL_DATA.

Opciones de notificación de acciones

Especifique una o ambas opciones enumeradas para solicitar que la aplicación receptora envíe un mensaje de informe de acción positiva o negativa.

MQRO_PAN

Este tipo de informe es generado por la aplicación que recupera el mensaje y actúa en consecuencia. Indica que la acción solicitada en el mensaje se ha realizado correctamente. La aplicación que genera el informe determina si se debe incluir algún dato en el informe.

Aparte de transmitir esta solicitud a la aplicación que recupera el mensaje, el gestor de colas no realiza ninguna acción basada en esta opción. La aplicación de recuperación debe generar el informe si procede.

MQRO_NAN

Este tipo de informe es generado por la aplicación que recupera el mensaje y actúa en consecuencia. Indica que la acción solicitada en el mensaje no se ha realizado correctamente. La aplicación que genera el informe determina si se debe incluir algún dato en el informe. Por ejemplo, es posible que desee incluir algunos datos que indiquen por qué no se pudo realizar la solicitud.

Aparte de transmitir esta solicitud a la aplicación que recupera el mensaje, el gestor de colas no realiza ninguna acción basada en esta opción. La aplicación de recuperación debe generar el informe si procede.

La aplicación debe determinar qué condiciones corresponden a una acción positiva y cuáles a una acción negativa. Sin embargo, si la solicitud solo se ha cumplido parcialmente, genere un informe NAN en lugar de un informe PAN, si así se solicita. Cada condición posible debe corresponder a una acción positiva o a una acción negativa, pero no a ambas.

Opciones de identificador de mensaje

Especifique una de las opciones enumeradas para controlar cómo se debe configurar el *MsgId* del mensaje de informe (o del mensaje de respuesta).

MQRO_NEW_MSG_ID

Esta es la acción predeterminada e indica que si se genera un informe o una respuesta como resultado de este mensaje, se generará *MsgId* un nuevo para el informe o el mensaje de respuesta.

MQRO_PASS_MSG_ID

Si se genera un informe o una respuesta como resultado de este mensaje, el *MsgId* de este mensaje se copia en el *MsgId* del informe o del mensaje de respuesta.

El *MsgId* de un mensaje de publicación será diferente para cada suscriptor que reciba una copia de la publicación y, por lo tanto, el *MsgId* copiado en el informe o mensaje de respuesta será diferente para cada uno.

Si no se especifica esta opción, se asume MQRO_NEW_MSG_ID.

Opciones de identificador de correlación

Especifique una de las opciones enumeradas para controlar cómo se debe configurar el *CorrelId* del mensaje de informe (o del mensaje de respuesta).

MQRO_COPY_MSG_ID_TO_CORREL_ID

Esta es la acción predeterminada e indica que, si se genera un informe o una respuesta como resultado de este mensaje, el *MsgId* de este mensaje se copia en el *CorrelId* del informe o mensaje de respuesta.

El *MsgId* de un mensaje de publicación será diferente para cada suscriptor que reciba una copia de la publicación y, por lo tanto, el *MsgId* copiado en el *CorrelId* del informe o mensaje de respuesta será diferente para cada uno.

MQRO_PASS_CORREL_ID

Si se genera un informe o una respuesta como resultado de este mensaje, el *CorrelId* de este mensaje se copia en el *CorrelId* del informe o del mensaje de respuesta.

El *CorrelId* de un mensaje de publicación será específico para un suscriptor, a menos que utilice la opción MQSO_SET_CORREL_ID y establezca el campo SubCorrelId en MQSD como MQCI_NONE. Por lo tanto, es posible que el *CorrelId* copiado en el *CorrelId* del informe o mensaje de respuesta sea diferente para cada uno.

Si no se especifica esta opción, se asume MQRO_COPY_MSG_ID_TO_CORREL_ID.

Los servidores que responden a solicitudes o generan mensajes de informe deben comprobar si las opciones MQRO_PASS_MSG_ID o MQRO_PASS_CORREL_ID se establecieron en el mensaje original. Si fuera así, los servidores deben tomar las medidas descritas para esas opciones. Si no se establece ninguno de los dos, los servidores deben realizar la acción predeterminada correspondiente.

Opciones de disposición

Especifique una de las opciones enumeradas para controlar la disposición del mensaje original cuando no se pueda entregar a la cola de destino. La aplicación puede configurar las opciones de disposición independientemente de los informes de excepción solicitados.

MQRO_DEAD_LETTER_Q

Esta es la acción predeterminada, y coloca el mensaje en la cola de mensajes no entregados si el mensaje no se puede entregar a la cola de destino. Esto ocurre en las siguientes situaciones:

- Cuando la aplicación que envió el mensaje original no puede ser notificada de forma sincrónica del problema mediante el código de motivo devuelto por la llamada MQPUT o MQPUT1. Se genera un mensaje de informe de excepción, si el remitente lo ha solicitado.
- Cuando la aplicación que publicó el mensaje original lo estaba colocando en un tema

MQRO_DISCARD_MSG

Esto descarta el mensaje si no se puede entregar a la cola de destino. Esto ocurre en las siguientes situaciones:

- Cuando la aplicación que envió el mensaje original no puede ser notificada de forma sincrónica del problema mediante el código de motivo devuelto por la llamada MQPUT o MQPUT1. Se genera un mensaje de informe de excepción, si el remitente lo ha solicitado.
- Cuando la aplicación que publicó el mensaje original lo estaba colocando en un tema

Si desea devolver el mensaje original al remitente, sin que el mensaje original se coloque en la cola de mensajes no entregados, el remitente debe especificar MQRO_DISCARD_MSG con MQRO_EXCEPTION_WITH_FULL_DATA.

MQRO_PASS_DISCARD_AND_EXPIRY

Si esta opción está activada en un mensaje y se genera un informe o una respuesta debido a ello, el descriptor del mensaje del informe hereda:

- MQRO_DISCARD_MSG si se ha establecido.
- El tiempo restante hasta la expiración del mensaje (si no se trata de un informe de expiración). Si se trata de un informe de caducidad, el tiempo de caducidad se establece en 60 segundos.

Opción de actividad

MQRO_ACTIVITY

El uso de este valor permite rastrear la ruta de **cualquier** mensaje a través de una red de gestores de colas. La opción de informe se puede especificar en cualquier mensaje de usuario actual, lo que le permite comenzar a calcular instantáneamente la ruta del mensaje a través de la red.

Si la aplicación que genera el mensaje no puede habilitar la generación de informes de actividad, los informes se pueden habilitar mediante una salida de cruce de API proporcionada por los administradores del gestor de colas.

Nota:

1. Cuanto menor sea el número de gestores de colas de la red capaces de generar informes de actividad, menos detallada será la ruta.
2. Puede resultar difícil ordenar correctamente los informes de actividad para determinar la ruta seguida.
3. Es posible que los informes de actividad no puedan encontrar una ruta hacia el destino solicitado.
4. Los mensajes con esta opción de informe activada deben ser aceptados por cualquier gestor de colas, incluso si no comprenden la opción. Esto permite configurar la opción de informe en cualquier mensaje de usuario, incluso si son procesados por un gestor de colas no compatible IBM WebSphere MQ 6.0 o posterior.
5. Si un proceso, ya sea un gestor de colas o un proceso de usuario, realiza una actividad en un mensaje con esta opción activada, puede optar por generar y enviar un informe de actividad.

Opción predeterminada

Especifique lo siguiente si no se requieren opciones de informe:

MQRO_NONE

Utilice este valor para indicar que no se han especificado otras opciones. MQRO_NONE se define para facilitar la documentación del programa. No está previsto que esta opción se utilice con ninguna otra, pero como su valor es cero, dicho uso no puede detectarse.

Información general

1. Todos los tipos de informes requeridos deben ser solicitados específicamente por la aplicación que envía el mensaje original. Por ejemplo, si se solicita un informe COA pero no un informe de excepción, se genera un informe COA cuando el mensaje se coloca en la cola de destino, pero no se genera ningún informe de excepción si la cola de destino está llena cuando llega el mensaje. Si no se establece ninguna *Report* opción, el gestor de colas o el agente de canal de mensajes (MCA) no generan mensajes de informe.

Algunas opciones de informe se pueden especificar aunque el gestor de colas local no las reconozca; esto resulta útil cuando la opción debe ser procesada por el gestor de colas *de destino*. Consulte [“Opciones de informe y distintivos de mensaje” en la página 911](#) para obtener más detalles.

Si se solicita un mensaje de informe, se debe especificar el nombre de la cola a la que se debe enviar el informe en el *ReplyToQ* campo. Cuando se recibe un mensaje de informe, la naturaleza del informe puede determinarse examinando el *Feedback* campo en el descriptor del mensaje.

2. Si el gestor de colas o MCA que genera un mensaje de informe no puede colocar el mensaje de informe en la cola de respuesta (por ejemplo, porque la cola de respuesta o la cola de transmisión están llenas), el mensaje de informe se coloca en su lugar en la cola de mensajes no entregados. Si eso *también* falla, o no hay una cola de mensajes no entregados, la acción que se tome dependerá del tipo de mensaje del informe:
 - Si el mensaje del informe es un informe de excepción, el mensaje que generó el informe de excepción se deja en su cola de transmisión; esto garantiza que el mensaje no se pierda.
 - Para todos los demás tipos de informes, el mensaje del informe se descarta y el procesamiento continúa con normalidad. Esto se hace porque el mensaje original ya se ha entregado correctamente (en el caso de los mensajes de informe COA o COD) o porque ya no tiene interés (en el caso de los mensajes de informe de caducidad).

Una vez que un mensaje de informe se ha colocado correctamente en una cola (ya sea la cola de destino o una cola de transmisión intermedia), el mensaje ya no está sujeto a un procesamiento especial, sino que se trata como cualquier otro mensaje.

3. Cuando se genera el informe, se abre la *ReplyToQ* cola y se coloca el mensaje del informe utilizando la autoridad del *UserIdentifier* en el MQMD del mensaje que provoca el informe, excepto en los siguientes casos:

- Los informes de excepción generados por un MCA receptor se incluyen con la autoridad que utilizó el MCA cuando intentó enviar el mensaje que provocó el informe.
- Los informes COA generados por el gestor de colas se incluyen con la autoridad que se utilizó cuando el mensaje que provocó el informe se colocó en el gestor de colas que generó el informe. Por ejemplo, si el mensaje fue enviado por un MCA receptor utilizando el identificador de usuario del MCA, el gestor de colas envía el informe COA utilizando el identificador de usuario del MCA.

Las aplicaciones que generan informes deben utilizar la misma autoridad que utilizan para generar una respuesta; normalmente se trata de la autoridad del identificador de usuario del mensaje original.

Si el informe tiene que enviarse a un destino remoto, los remitentes y destinatarios pueden decidir si aceptarlo, del mismo modo que lo hacen con otros mensajes.

4. Si se solicita un mensaje de informe con datos:

- El mensaje de informe siempre se genera con la cantidad de datos solicitada por el remitente del mensaje original. Si el mensaje del informe es demasiado grande para la cola de respuestas, se produce el procesamiento descrito anteriormente; el mensaje del informe nunca se trunca para que quepa en la cola de respuestas.
- Si el *Format* del mensaje original es MQFMT_XMIT_Q_HEADER, los datos incluidos en el informe no incluyen el MQXQH. Los datos del informe comienzan con el primer byte de los datos más allá del MQXQH en el mensaje original. Esto ocurre independientemente de si la cola es una cola de transmisión o no.

5. Si se recibe un mensaje de informe COA, COD o de caducidad en la cola de respuesta, se garantiza que el mensaje original llegó, se entregó o caducó, según corresponda. Sin embargo, si se solicita uno o más de estos mensajes de informe y no se reciben, no se puede asumir lo contrario, ya que podría haber ocurrido alguna de las siguientes situaciones:

- a. El mensaje del informe se retrasa porque un enlace no funciona.
- b. El mensaje de informe se retiene porque existe una condición de bloqueo en una cola de transmisión intermedia o en la cola de respuesta (por ejemplo, la cola está llena o inhibida para inserciones).
- c. El mensaje del informe se encuentra en una cola de mensajes no entregados.
- d. Cuando el gestor de colas intentaba generar el mensaje de informe, no podía colocarlo ni en la cola adecuada ni en la cola de mensajes no entregados, por lo que no se pudo generar el mensaje de informe.
- e. Se produjo un fallo en el gestor de colas entre la acción notificada (llegada, entrega o vencimiento) y la generación del mensaje de informe correspondiente. (Esto no ocurre con los mensajes de informe COD si la aplicación recupera el mensaje original dentro de una unidad de trabajo, ya que el mensaje de informe COD se genera dentro de la misma unidad de trabajo)

Los mensajes de informe de excepciones pueden retenerse de la misma manera por las razones 1, 2 y 3 anteriores. Sin embargo, cuando un MCA no puede generar un mensaje de informe de excepción (el mensaje de informe no se puede colocar ni en la cola de respuesta ni en la cola de mensajes no entregados), el mensaje original permanece en la cola de transmisión del remitente y el canal se cierra. Esto ocurre independientemente de si el mensaje de informe debía generarse en el extremo emisor o receptor del canal.

6. Si el mensaje original se bloquea temporalmente (lo que da lugar a la generación de un mensaje de informe de excepción y a que el mensaje original se coloque en una cola de mensajes no entregados), pero el bloqueo se despeja y una aplicación lee el mensaje original de la cola de mensajes no entregados y lo vuelve a colocar en su destino, puede ocurrir lo siguiente:

- Aunque se haya generado un mensaje de informe de excepción, el mensaje original finalmente llega correctamente a su destino.
- Se genera más de un mensaje de informe de excepción con respecto a un único mensaje original, ya que el mensaje original podría encontrar otro bloqueo más adelante.

Informar de los mensajes al publicar en un tema

1. Se pueden generar informes al publicar un mensaje en un tema. Este mensaje se enviará a todos los suscriptores del tema, que pueden ser cero, uno o muchos. Esto debe tenerse en cuenta al elegir las opciones de informe, ya que podrían generarse muchos mensajes de informe como resultado.
2. Al enviar un mensaje a un tema, puede haber muchas colas de destino a las que se les debe enviar una copia del mensaje. Si alguna de estas colas de destino tiene un problema, como estar llena, la finalización correcta de MQPUT dependerá de la configuración de NPMGDLV o PMSGDLV (dependiendo de la persistencia del mensaje). Si la configuración es tal que la entrega del mensaje a la cola de destino debe realizarse correctamente (por ejemplo, se trata de un mensaje persistente a un suscriptor duradero y PMSGDLV está establecido en ALL o ALLDUR), entonces se considera que se ha realizado correctamente si se cumple uno de los siguientes criterios:
 - Exitosa inserción en la cola de suscriptores
 - Uso de MQRO_DEAD_LETTER_Q y envío correcto a la cola de mensajes no entregados si la cola del suscriptor no puede aceptar el mensaje
 - Uso de MQRO_DISCARD_MSG si la cola del suscriptor no puede aceptar el mensaje.

Informar mensajes para segmentos de mensajes

1. Se pueden solicitar mensajes de informe para los mensajes que permiten la segmentación (véase la descripción del indicador MQMF_SEGMENTATION_ALLOWED). Si el gestor de colas considera necesario segmentar el mensaje, se puede generar un mensaje de informe para cada uno de los segmentos que posteriormente cumpla la condición pertinente. Las aplicaciones deben estar preparadas para recibir múltiples mensajes de informe para cada tipo de mensaje de informe solicitado. Utilice el *GroupId* campo del mensaje de informe para correlacionar los múltiples informes con el identificador de grupo del mensaje original, y el *Feedback* campo para identificar el tipo de cada mensaje de informe.
2. Si se utiliza MQGMO_LOGICAL_ORDER para recuperar mensajes de informe para segmentos, tenga en cuenta que las llamadas sucesivas a MQGET pueden devolver informes de *diferentes tipos*. Por ejemplo, si se solicitan informes COA y COD para un mensaje segmentado por el gestor de colas, las llamadas MQGET para los mensajes de informe pueden devolver los mensajes de informe COA y COD intercalados de forma impredecible. Evítelo utilizando la opción MQGMO_COMPLETE_MSG (opcionalmente con MQGMO_ACCEPT_TRUNCATED_MSG). MQGMO_COMPLETE_MSG hace que el gestor de colas vuelva a ensamblar los mensajes de informe que tienen el mismo tipo de informe. Por ejemplo, la primera llamada MQGET podría reensamblar todos los mensajes COA relacionados con el mensaje original, y la segunda llamada MQGET podría reensamblar todos los mensajes COD. El orden en que se vuelven a ensamblar depende del tipo de mensaje de informe que aparezca primero en la cola.
3. Las aplicaciones que colocan segmentos pueden especificar diferentes opciones de informe para cada segmento. Sin embargo, tenga en cuenta los siguientes puntos:
 - Si los segmentos se recuperan utilizando la opción MQGMO_COMPLETE_MSG, el gestor de colas solo tiene en cuenta las opciones de informe del *primer* segmento.
 - Si los segmentos se recuperan uno por uno y la mayoría de ellos tienen una de las opciones MQRO_COD_*, pero al menos un segmento no la tiene, no se puede utilizar la opción MQGMO_COMPLETE_MSG para recuperar los mensajes del informe con una sola llamada MQGET, ni utilizar la opción MQGMO_ALL_SEGMENTS_AVAILABLE para detectar cuándo han llegado todos los mensajes del informe.
4. En una red de tipo « MQ », los gestores de colas pueden tener diferentes capacidades. Si un mensaje de informe para un segmento es generado por un gestor de colas o MCA que no admite segmentación, el gestor de colas o MCA no incluye de forma predeterminada la información de segmento necesaria en el mensaje de informe, lo que puede dificultar la identificación del mensaje original que provocó

la generación del informe. Evite esta dificultad solicitando los datos con el mensaje de informe, es decir, especificando las opciones adecuadas MQRO_*_WITH_DATA o MQRO_*_WITH_FULL_DATA. Sin embargo, tenga en cuenta que si se especifica MQRO_*_WITH_DATA, es posible que se devuelvan *menos de 100 bytes* de datos de mensajes de aplicación a la aplicación que recupera el mensaje de informe, si el mensaje de informe es generado por un gestor de colas o MCA que no admite segmentación.

Contenido del descriptor de mensaje para un mensaje de informe

Cuando el gestor de colas o el agente de canal de mensajes (MCA) genera un mensaje de informe, establece los campos del descriptor del mensaje con los siguientes valores y, a continuación, envía el mensaje de la forma habitual.

Tabla 519. Valores utilizados para los campos MQMD cuando un mensaje de informe es generado por el sistema

Campo de MQMD	Valor utilizado
<i>StrucId</i>	MQMD_STRUC_ID
<i>Version</i>	MQMD_VERSION_2
<i>Report</i>	MQRO_NONE
<i>MsgType</i>	MQMT_REPORT
<i>Expiry</i>	MQEI_UNLIMITED
<i>Feedback</i>	Según corresponda a la naturaleza del informe (MQFB_COA, MQFB_COD, MQFB_EXPIRATION o un valor MQRC_*)
<i>Encoding</i>	Copiado del descriptor del mensaje original
<i>CodedCharSetId</i>	Copiado del descriptor del mensaje original
<i>Format</i>	Copiado del descriptor del mensaje original
<i>Priority</i>	Copiado del descriptor del mensaje original
<i>Persistence</i>	Copiado del descriptor del mensaje original
<i>MsgId</i>	Según lo especificado por las opciones de informe en el descriptor del mensaje original
<i>CorrelId</i>	Según lo especificado por las opciones de informe en el descriptor del mensaje original
<i>BackoutCount</i>	0
<i>ReplyToQ</i>	Espacios en blanco
<i>ReplyToQMgr</i>	Nombre de gestor de colas
<i>UserIdentifier</i>	Según lo establecido por la opción MQPMO_PASS_IDENTITY_CONTEXT
<i>AccountingToken</i>	Según lo establecido por la opción MQPMO_PASS_IDENTITY_CONTEXT
<i>ApplIdentityData</i>	Según lo establecido por la opción MQPMO_PASS_IDENTITY_CONTEXT
<i>PutApplType</i>	MQAT_QMGR, o según corresponda para el agente del canal de mensajes
<i>PutApplName</i>	Los primeros 28 bytes del nombre del gestor de colas o del agente del canal de mensajes. Para los mensajes de informe generados por el puente IMS, este campo contiene el nombre del grupo XCF y el nombre del miembro XCF del IMS sistema al que se refiere el mensaje.
<i>PutDate</i>	Fecha en la que se envía el mensaje del informe
<i>PutTime</i>	Hora en que se envía el mensaje del informe
<i>ApplOriginData</i>	Espacios en blanco

Tabla 519. Valores utilizados para los campos MQMD cuando un mensaje de informe es generado por el sistema (continuación)

Campo de MQMD	Valor utilizado
<i>GroupId</i>	Copiado del descriptor del mensaje original
<i>MsgSeqNumber</i>	Copiado del descriptor del mensaje original
<i>Offset</i>	Copiado del descriptor del mensaje original
<i>MsgFlags</i>	Copiado del descriptor del mensaje original
<i>OriginalLength</i>	Se copia del descriptor del mensaje original si no es MQOL_UNDEFINED y, en caso contrario, se establece en la longitud de los datos del mensaje original

Se recomienda que una aplicación que genere un informe establezca valores similares, excepto en los siguientes casos:

- El *ReplyToQMGr* campo se puede dejar en blanco (el gestor de colas lo cambia por el nombre del gestor de colas local cuando se envía el mensaje).
- Establezca los campos de contexto utilizando la opción que se habría utilizado para una respuesta, normalmente MQPMO_PASS_IDENTITY_CONTEXT.

Análisis del campo del informe

El *Report* campo contiene subcampos; por este motivo, las aplicaciones que necesitan comprobar si el remitente del mensaje ha solicitado un informe concreto deben utilizar una de las técnicas descritas en [“Análisis del campo de informe”](#) en la página 912.

Este es un campo de salida para la llamada MQGET y un campo de entrada para las llamadas MQPUT y MQPUT1. El valor inicial de este campo es MQRO_NONE.

MsgType (MQLONG)

Esto indica el tipo de mensaje. Los tipos de mensajes se agrupan de la siguiente manera:

MQMT_SISTEMA_PRIMERO

Valor más bajo para los tipos de mensajes definidos por el sistema.

MQMT_SISTEMA_ÚLTIMO

Valor más alto para los tipos de mensajes definidos por el sistema.

Los siguientes valores están definidos actualmente dentro del rango del sistema:

MQMT_DATAGRAM

El mensaje es uno que no requiere una respuesta.

MQMT_REQUEST

El mensaje es uno que requiere una respuesta.

Especifique el nombre de la cola a la que enviar la respuesta en el *ReplyToQ* campo. El *Report* campo indica cómo configurar el *MsgId* y *CorrelId* el de la respuesta.

MQMT_REPLY

El mensaje es la respuesta a un mensaje de solicitud anterior (MQMT_REQUEST). El mensaje debe enviarse a la cola indicada por el *ReplyToQ* campo del mensaje de solicitud. Utilice el *Report* campo de la solicitud para controlar cómo configurar los campos *MsgId* y *CorrelId* de la respuesta.

Nota: El gestor de colas no impone la relación solicitud-respuesta; esta es una responsabilidad de la aplicación.

MQMT_REPORT

El mensaje informa sobre algún suceso esperado o inesperado, normalmente relacionado con algún otro mensaje (por ejemplo, se ha recibido un mensaje de solicitud que contenía datos no válidos). Enviar el mensaje a la cola indicada por el *ReplyToQ* campo del descriptor de mensajes del mensaje

original. Configure los *Feedback* campos para indicar la naturaleza del informe. Utilice el *Report* campo del mensaje original para controlar cómo configurar el *MsgId* y *CorrelId* del mensaje del informe.

Los mensajes de informe generados por el gestor de colas o el agente del canal de mensajes siempre se envían a la *ReplyToQ* cola, con los campos *Feedback CorrelId* y configurados como se ha descrito anteriormente.

También se pueden utilizar valores definidos por la aplicación. Deben estar dentro del siguiente rango:

MQMT_APPL_FIRST

Valor más bajo para los tipos de mensajes definidos por la aplicación.

MQMT_APPL_ÚLTIMO

Valor más alto para los tipos de mensajes definidos por la aplicación.

Para las llamadas MQPUT y MQPUT1, el *MsgType* valor debe estar dentro del rango definido por el sistema o del rango definido por la aplicación; si no es así, la llamada falla con el código de motivo MQRC_MSG_TYPE_ERROR.

Este es un campo de salida para la llamada MQGET y un campo de entrada para las llamadas MQPUT y MQPUT1. El valor inicial de este campo es MQMT_DATAGRAM.

Expiry (MQLONG)

Es un periodo de tiempo expresado en décimas de segundo, establecido por la aplicación que envía el mensaje. El mensaje se debe poder seleccionar para descartarlo si no se ha suprimido de la cola de destino antes de que transcurra este período de tiempo.

Por ejemplo, para establecer un minuto como tiempo de caducidad, debe configurar **MQMD.Expiry** a 600.

El valor se reduce para reflejar el tiempo que el mensaje permanece en la cola de destino y también en cualquier cola de transmisión intermedia si el envío se realiza a una cola remota. También puede ser decrementado por los agentes del canal de mensajes para reflejar los tiempos de transmisión, si estos son significativos. Del mismo modo, una aplicación que reenvíe este mensaje a otra cola podría disminuir el valor si fuera necesario, en caso de que haya retenido el mensaje durante un tiempo considerable. Sin embargo, el tiempo de expiración se considera aproximado y no es necesario decrementar el valor para reflejar pequeños intervalos de tiempo.

Cuando una aplicación recupera el mensaje mediante la llamada MQGET, el *Expiry* campo representa el tiempo de caducidad que aún queda.

Una vez transcurrido el tiempo de caducidad de un mensaje, el gestor de colas puede descartarlo. El mensaje se descarta cuando se produce una llamada MQGET de exploración o sin exploración que habría devuelto el mensaje si no hubiera caducado ya. Por ejemplo, una llamada MQGET sin exploración con el *MatchOptions* campo en MQGMO establecido en MQMO_NONE que lee desde una cola ordenada FIFO descarta todos los mensajes caducados hasta el primer mensaje no caducado. Con una cola ordenada por prioridad, la misma llamada descartará los mensajes caducados de mayor prioridad y los mensajes de igual prioridad que llegaron a la cola antes que el primer mensaje no caducado.

Un mensaje que ha caducado nunca se devuelve a una aplicación (ya sea mediante una llamada MQGET de exploración o sin exploración), por lo que el valor del *Expiry* campo del descriptor del mensaje tras una llamada MQGET correcta es mayor que cero o el valor especial MQEI_UNLIMITED.

Si un mensaje se coloca en una cola remota, es posible que caduque (y se descarte) mientras se encuentra en una cola de transmisión intermedia, antes de llegar a la cola de destino.

Se genera un informe cuando se descarta un mensaje caducado, si el mensaje especificaba una de las opciones de informe MQRO_EXPIRATION_*. Si no se especifica ninguna de estas opciones, no se genera dicho informe; se supone que el mensaje ya no es relevante después de este período de tiempo (quizás porque un mensaje posterior lo ha sustituido).

Para un mensaje colocado dentro de un punto de sincronización, el intervalo de caducidad comienza en el momento en que se coloca el mensaje, no en el momento en que se confirma el punto de sincronización. Es posible que el intervalo de caducidad expire antes de que se confirme el punto de sincronización. En

este caso, el mensaje se descartará en algún momento después de la operación de confirmación y no se devolverá a una aplicación en respuesta a una operación MQGET.

Cualquier otro programa que descarte mensajes basándose en el tiempo de caducidad también debe enviar un mensaje de informe adecuado si se ha solicitado.

Notas:

1. Si se envía un mensaje con un *Expiry* tiempo de expiración igual a cero o un número superior a 999 999 999, la llamada MQPUT o MQPUT1 falla con el código de motivo MQRC_EXPIRY_ERROR; en este caso no se genera ningún mensaje de informe.

Para habilitar el código de motivo 2013, MQRC_EXPIRY_ERROR, debe habilitar la variable de entorno AMQ_ENFORCE_MAX_EXPIRY_ERROR.

A continuación se muestra un ejemplo de Linux:

```
$ export AMQ_ENFORCE_MAX_EXPIRY_ERROR=True
```

Tenga en cuenta que:

- Lo importante es exportar la variable
 - El valor real se ignora, sin embargo, utilizar True puede resultar útil al revisar la configuración.
2. Dado que un mensaje cuyo tiempo de caducidad ha expirado puede no descartarse hasta más tarde, es posible que haya mensajes en una cola que hayan superado su tiempo de caducidad y que, por lo tanto, no sean aptos para su recuperación. No obstante, estos mensajes cuentan para el número de mensajes en la cola a todos los efectos, incluida la activación por profundidad.

Si un suscriptor/consumidor (cliente) intenta obtener un mensaje y ese mensaje ha caducado, el cliente no recibe nada, ya que el mensaje se ha descartado por ser demasiado antiguo. Además, el cliente no recibirá ningún mensaje de error.
 3. Se genera un informe de caducidad, si se solicita, cuando se descarta el mensaje, no cuando este pasa a ser susceptible de ser descartado.
 4. El descarte de un mensaje caducado y la generación de un informe de caducidad, si se solicita, nunca forman parte de la unidad de trabajo de la aplicación, incluso si el mensaje se programó para su descarte como resultado de una llamada MQGET que opera dentro de una unidad de trabajo.
 5. Si un mensaje a punto de caducar se recupera mediante una llamada MQGET dentro de una unidad de trabajo y, posteriormente, se revoca dicha unidad de trabajo, es posible que el mensaje pueda descartarse antes de que se pueda volver a recuperar.
 6. Si un mensaje a punto de caducar se bloquea mediante una llamada MQGET con MQGMO_LOCK, es posible que el mensaje pueda descartarse antes de que pueda recuperarse mediante una llamada MQGET con MQGMO_MSG_UNDER_CURSOR; si esto ocurre, se devuelve el código de motivo MQRC_NO_MSG_UNDER_CURSOR en esta llamada MQGET posterior.
 7. Cuando se recupera un mensaje de solicitud con un tiempo de caducidad superior a cero, la aplicación puede realizar una de las siguientes acciones al enviar el mensaje de respuesta:
 - Copie el tiempo de caducidad restante del mensaje de solicitud al mensaje de respuesta.
 - Establezca el tiempo de caducidad en el mensaje de respuesta en un valor explícito mayor que cero.
 - Establezca el tiempo de caducidad en el mensaje de respuesta en MQEI_UNLIMITED.

La acción a realizar depende del diseño de la aplicación. Sin embargo, la acción predeterminada para colocar mensajes en una cola de mensajes no entregados (mensajes no entregados) debe ser conservar el tiempo de caducidad restante del mensaje y continuar disminuyéndolo.

8. Los mensajes de activación siempre se generan con MQEI_UNLIMITED.
9. Un mensaje (normalmente en una cola de transmisión) que tiene un *Format* nombre de MQFMT_XMIT_Q_HEADER tiene un segundo descriptor de mensaje dentro del MQXQH. Por lo tanto, tiene dos *Expiry* campos asociados. En este caso, deben tenerse en cuenta los siguientes puntos adicionales:

- Cuando una aplicación coloca un mensaje en una cola remota, el gestor de colas coloca inicialmente el mensaje en una cola de transmisión local y antepone a los datos del mensaje de la aplicación una estructura MQXQH. El gestor de colas establece los valores de los dos *Expiry* campos para que sean iguales a los especificados por la aplicación.

Si una aplicación coloca un mensaje directamente en una cola de transmisión local, los datos del mensaje deben comenzar con una estructura MQXQH y el nombre del formato debe ser MQFMT_XMIT_Q_HEADER. En este caso, la aplicación no necesita establecer los valores de estos dos *Expiry* campos para que sean iguales. (El gestor de colas comprueba que el *Expiry* campo dentro de MQXQH contiene un valor válido y que los datos del mensaje son lo suficientemente largos como para incluirlo). Para que una aplicación pueda escribir directamente en la cola de transmisión, la aplicación debe crear un encabezado de cola de transmisión con el descriptor de mensaje incrustado. Sin embargo, si el valor de caducidad en el descriptor de mensaje escrito en la cola de transmisión no coincide con el valor en el descriptor de mensaje incrustado, se produce un rechazo por error de caducidad.

- Cuando se recupera un mensaje con el *Format* nombre MQFMT_XMIT_Q_HEADER de una cola (ya sea una cola normal o una cola de transmisión), el gestor de colas decrementa *ambos Expiry* campos con el tiempo de espera en la cola. No se produce ningún error si los datos del mensaje no son lo suficientemente largos como para incluir el *Expiry* campo en el MQXQH.
 - El gestor de colas utiliza el *Expiry* campo del descriptor de mensajes independiente (es decir, no el del descriptor de mensajes integrado en la estructura MQXQH) para comprobar si el mensaje es susceptible de ser descartado.
 - Si los valores iniciales de los dos *Expiry* campos son diferentes, el *Expiry* tiempo en el descriptor de mensaje separado cuando se recupera el mensaje podría ser mayor que cero (por lo que el mensaje no es elegible para descartarse), mientras que el tiempo según el *Expiry* campo en el MQXQH ha transcurrido. En este caso, el *Expiry* campo en el MQXQH se establece en cero.
10. El tiempo de caducidad de un mensaje de respuesta devuelto desde el puente IMS es ilimitado, a menos que se establezca MQIIH_PASS_EXPIRATION en el campo Flags del MQIIH. Consulte [Banderas](#) para obtener más información.

Se reconoce el siguiente valor especial:

MQEI_UNLIMITED

El mensaje tiene una validez ilimitada.

Este es un campo de salida para la llamada MQGET y un campo de entrada para las llamadas MQPUT y MQPUT1. El valor inicial de este campo es MQEI_UNLIMITED.

Véase también [CAPEXPY](#): Limitar el tiempo de caducidad de los mensajes.



Mensajes caducados en z/OS

En IBM MQ for z/OS, los mensajes que han caducado se descartan en la siguiente llamada MQGET adecuada.

Sin embargo, si no se produce dicha llamada, el mensaje caducado no se descarta y, en algunas colas, se puede acumular un gran número de mensajes caducados. Para solucionar esto, configure el gestor de colas para que escanee las colas periódicamente y descarte los mensajes caducados en una o varias colas de una de las siguientes maneras:

Exploración periódica

Puede especificar un período utilizando el atributo EXPRYINT (intervalo de caducidad) del gestor de colas. Cada vez que se alcanza el intervalo de caducidad, el gestor de colas busca colas candidatas que merezcan ser escaneadas para descartar los mensajes caducados.

El gestor de colas mantiene información sobre los mensajes caducados en cada cola y sabe si merece la pena realizar un escaneo en busca de mensajes caducados. Por lo tanto, solo se escanea una selección de colas en cada momento.

Las colas compartidas son escaneadas por un solo gestor de colas en un grupo de intercambio de colas. Por lo general, es el primer gestor de colas que se reinicia o el primero en tener EXPRYINT establecido. Si este gestor de colas finaliza, otro gestor de colas del grupo de uso compartido de colas se encarga del escaneo de colas. Establezca el valor del intervalo de caducidad para todos los gestores de colas dentro de un grupo de intercambio de colas en el mismo valor.

Tenga en cuenta que el procesamiento de caducidad se lleva a cabo para todas las colas cuando se reinicia un gestor de colas, independientemente de la configuración de EXPRYINT.

Solicitud explícita

Ejecute el comando REFRESH QMGR TYPE(EXPIRY), especificando la cola o colas que desea escanear.

Feedback (MQLONG)

El campo Feedback se utiliza con un mensaje de tipo MQMT_REPORT para indicar la naturaleza del informe, y solo tiene sentido con ese tipo de mensaje.

El campo puede contener uno de los valores MQFB_* o uno de los valores MQRC_*. Los códigos de retroalimentación se agrupan de la siguiente manera:

MQFB_NONE

No se ha proporcionado ningún comentario.

MQFB_SISTEMA_PRIMERO

Valor más bajo para los comentarios generados por el sistema.

MQFB_SISTEMA_ÚLTIMO

Máximo valor para los comentarios generados por el sistema.

El rango de códigos de retroalimentación generados por el sistema, desde MQFB_SYSTEM_FIRST hasta MQFB_SYSTEM_LAST, incluye los códigos de retroalimentación generales enumerados en este tema (MQFB_*), así como los códigos de motivo (MQRC_*) que pueden aparecer cuando el mensaje no se puede colocar en la cola de destino.

MQFB_APPL_FIRST

Valor más bajo para los comentarios generados por la aplicación.

MQFB_APPL_ÚLTIMO

Máximo valor para los comentarios generados por la aplicación.

Las aplicaciones que generan mensajes de informe no deben utilizar códigos de respuesta en el rango del sistema (excepto MQFB_QUIT), a menos que deseen simular mensajes de informe generados por el gestor de colas o el agente de canal de mensajes.

En las llamadas MQPUT o MQPUT1, el valor especificado debe ser MQFB_NONE o estar dentro del rango del sistema o del rango de la aplicación. Esto se comprueba independientemente del valor de *MsgType*.

Códigos generales de comentarios :

MQFB_COA

Confirmación de llegada a la cola de destino (véase MQRO_COA).

MQFB_COD

Confirmación de entrega a la aplicación receptora (véase MQRO_COD).

MQFB_CADUCIDAD

El mensaje se descartó porque no se había eliminado de la cola de destino antes de que expirara su tiempo de validez.

MQFB_PAN

Notificación de acción positiva (véase MQRO_PAN).

MQFB_NAN

Notificación de acción negativa (véase MQRO_NAN).

MQFB_SALIR

Finalizar aplicación.

Esto puede ser utilizado por un programa de programación de cargas de trabajo para controlar el número de instancias de un programa de aplicación que se están ejecutando. El envío de un mensaje MQMT_REPORT con este código de respuesta a una instancia del programa de aplicación indica a dicha instancia que debe detener el procesamiento. Sin embargo, el cumplimiento de esta convención es una cuestión que compete a la aplicación; no es impuesta por el gestor de colas.

Códigos de retroalimentación del canal

MQFB_CANAL_COMPLETADO

Un canal terminó normalmente.

MQFB_CANAL_FALLO

Un canal ha finalizado de forma anómala y pasa al estado DETENIDO.

MQFB_CANAL_FALLO_REINTENTO

Un canal ha finalizado de forma anómala y pasa al estado RETRY.

IMS-códigos de retroalimentación de puente

Estos códigos se utilizan cuando se recibe un código de detección -OTMA IMS inesperado. El código de sentido o, cuando el código de sentido es 0x1A, el código de motivo asociado a ese código de sentido, se indica en la *retroalimentación*.

1. Para los códigos *de respuesta* comprendidos entre MQFB_IMS_FIRST (301) y MQFB_IMS_LAST (399), se ha recibido un código de sentido distinto de « 0x1A ». *El código de sentido* viene dado por la expresión (*Feedback* - MQFB_IMS_FIRST+1). El significado de los códigos de IMS sentido -OTMA y los códigos de motivo correspondientes se describen en la [sección «Códigos de sentido OTMA para mensajes NAK»](#) de la documentación de IMS.
2. Para los códigos *de retroalimentación* comprendidos entre MQFB_IMS_NACK_1A_REASON_FIRST (600) y MQFB_IMS_NACK_1A_REASON_LAST (855), se recibió un código de detección de 0x1A. *El código de motivo* asociado al código de sentido viene dado por la expresión (*Feedback* - MQFB_IMS_NACK_1A_REASON_FIRST). Consulte [los mensajes de « 001A »](#) en la documentación de « IMS ».

El puente « IMS » puede generar los siguientes códigos de respuesta:

MQFB_LONGITUD_DE_DATOS_CERO

La longitud del segmento era cero en los datos de aplicación del mensaje.

MQFB_LONGITUD_DE_DATOS_NEGATIVA

La longitud del segmento era negativa en los datos de aplicación del mensaje.

MQFB_DATA_LENGTH_TOO_BIG

La longitud de un segmento era demasiado grande en los datos de aplicación del mensaje.

MQFB_DESBORDAMIENTO_DE_BUFFER

El valor de uno de los campos de longitud provocaría que los datos desbordaran el búfer de mensajes.

MQFB_LONGITUD_DESVIADA_EN_UNO

El valor de uno de los campos de longitud era 1 byte demasiado corto.

MQFB_IIH_ERROR

El *Format* campo en MQMD especifica MQFMT_IMS, pero el mensaje no comienza con una estructura MQIIH válida.

MQFB_NO_AUTORIZADO_PARA_IMS

El ID de usuario contenido en el descriptor de mensaje MQMD, o la contraseña contenida en el *Authenticator* campo de la estructura MQIIH, no superó la validación realizada por el puente IMS. Como resultado, el mensaje no se transmitió a IMS.

MQFB_IMS_ERROR

Se ha producido un IMSerror inesperado. Consulte el registro IBM MQ de errores del sistema en el que reside el puente IMS para obtener más información sobre el error.

MQFB_IMS_PRIMERO

Cuando el código de IMS sentido -OTMA no es « 0x1A », los códigos de IMSretroalimentación generados se encuentran en el intervalo comprendido entre MQFB_IMS_FIRST (301) y

MQFB_IMS_LAST (399). El código de IMS sentido -OTMA en sí mismo es *Feedback* menos MQFB_IMS_ERROR.

MQFB_IMS_ÚLTIMO

Valor más alto para la IMSretroalimentación generada cuando el código de sentido no es « 0x1A ».

MQFB_IMS_NACK_1A_REASON_FIRST

Cuando el código de sentido es « 0x1A », los códigos de IMSretroalimentación generados se encuentran en el rango « MQFB_IMS_NACK_1A_REASON_FIRST » (600) a « MQFB_IMS_NACK_1A_REASON_LAST » (855).

MQFB_IMS_NACK_1A_REASON_LAST

Valor más alto para la IMSretroalimentación generada cuando el código de sentido es 0x1A

CICS-códigos de retroalimentación de puente

Los siguientes códigos de retroalimentación pueden ser generados por el CICS bridge:

MQFB_CICS_APPL_ABENDED

El programa de aplicación especificado en el mensaje finalizó de forma anómala. Este código de respuesta solo aparece en el *Reason* campo de la estructura MQDLH.

MQFB_CICS_APPL_NO_INICIADO

El EXEC CICS LINK para el programa de aplicación especificado en el mensaje ha fallado. Este código de respuesta solo aparece en el *Reason* campo de la estructura MQDLH.

Fallo del puente MQFB_CICS

CICS bridge Terminó de forma anómala sin completar el procesamiento normal de errores.

MQFB_CICS_CCSID_ERROR

Identificador de conjunto de caracteres no válido.

MQFB_CICS_CIH_ERROR

CICS Estructura del encabezado de información ausente o no válida.

MQFB_CICS_COMMAREA_ERROR

Longitud de CICS COMMAREA no válida.

MQFB_CICS_CORREL_ID_ERROR

Identificador de correlación no válido.

MQFB_CICS_DLQ_ERROR

La CICS bridge tarea no pudo copiar una respuesta a esta solicitud en la cola de mensajes no entregados. La solicitud fue rechazada.

MQFB_CICS_ERROR_DE_CODIFICACIÓN

Codificación no válida.

MQFB_CICS_ERROR_INTERNO

CICS bridge Se ha producido un error inesperado.

Este código de respuesta solo aparece en el *Reason* campo de la estructura MQDLH.

MQFB_CICS_NO_AUTORIZADO

Identificador de usuario no autorizado o contraseña no válida.

Este código de respuesta solo aparece en el *Reason* campo de la estructura MQDLH.

MQFB_CICS_UOW_RETIRADO

La unidad de trabajo se canceló por una de las siguientes razones:

- Se detectó un error al procesar otra solicitud dentro de la misma unidad de trabajo.
- Se produjo un CICS error mientras se estaba ejecutando la unidad de trabajo.

MQFB_CICS_UOW_ERROR

Campo de control de unidad *UOWControl* de trabajo no válido.

Códigos de respuesta de mensajes de rastreo de ruta

MQFB_ACTIVITY

Se utiliza con el formato MQFMT_EMBEDDED_PCF para permitir la opción de datos de usuario tras los informes de actividad.

MQFB_MÁXIMO_ACTIVIDADES

Se devuelve cuando se descarta el mensaje de ruta de seguimiento porque el número de actividades en las que ha participado el mensaje supera el límite máximo de actividades.

MQFB_NO_REENVIADO

Se devuelve cuando se descarta el mensaje de ruta de seguimiento porque está a punto de enviarse a un gestor de colas remoto que no admite mensajes de ruta de seguimiento.

MQFB_NO_ENTREGADO

Se devuelve cuando el mensaje de ruta de seguimiento se descarta porque está a punto de colocarse en una cola local.

MQFB_REENVÍO NO ADMITIDO

Se devuelve cuando se descarta el mensaje de ruta de seguimiento porque un valor del parámetro de reenvío no se reconoce y se encuentra en la máscara de bits rechazada.

MQFB_ENTREGA NO ADMISIBLE

Se devuelve cuando se descarta el mensaje de ruta de seguimiento porque un valor del parámetro de entrega no se reconoce y se encuentra en la máscara de bits rechazada.

IBM MQ códigos de motivo

Para los mensajes de informe de excepciones, *Feedback* contiene un código IBM MQ de motivo. Entre los posibles códigos de motivo se encuentran:

MQRC_PUT_INHIBIDO

(2051, X'803') Se inhibieron las llamadas de puesta en cola.

MQRC_Q_COMPLETO

(2053, X'805') La cola ya contiene el número máximo de mensajes.

MQRC_NOT_AUTHORIZED

(2035, X'7F3') Acceso no autorizado.

MQRC_Q_ESPACIO_NO_DISPONIBLE

(2056, X'808') No hay espacio disponible en el disco para la cola.

MQRC_PERSISTENTE_NO_PERMITIDO

(2048, X'800') La cola no admite mensajes persistentes.

MQRC_MSG_DEMASIADO_GRANDE_PARA_Q_MGR

(2031, X'7EF') La longitud del mensaje supera el máximo permitido por el gestor de colas.

MQRC_MSG_DEMASIADO_GRANDE_PARA_Q

(2030, X'7EE') La longitud del mensaje supera el máximo permitido para la cola.

Para obtener una lista completa de los códigos de motivo, consulte:

- Para obtener más información IBM MQ for z/OS, consulte [Finalización de API y códigos de motivo](#).
- Para todas las demás plataformas, consulte [la finalización de la API y los códigos de motivo](#).

Este es un campo de salida para la llamada MQGET y un campo de entrada para las llamadas MQPUT y MQPUT1. El valor inicial de este campo es MQFB_NONE.

Encoding (MQLONG)

Esto especifica la codificación numérica de los datos numéricos en el mensaje; no se aplica a los datos numéricos en la propia estructura MQMD. La codificación numérica define la representación utilizada para los enteros binarios, los enteros decimales empaquetados y los números de coma flotante.



En z/OS, la parte entera binaria del Encoding campo también se utiliza para especificar la codificación entera de los datos de caracteres en el cuerpo del mensaje cuando el identificador del juego de caracteres correspondiente indica que la representación del juego de caracteres depende de la

codificación utilizada para los enteros binarios. Esto solo afecta a determinados conjuntos de caracteres multibyte (por ejemplo, los conjuntos de caracteres UTF-16).

En la llamada MQPUT o MQPUT1, la aplicación debe establecer este campo con el valor adecuado para los datos. El gestor de colas no comprueba que el campo sea válido. Se define el siguiente valor especial:

MQENC_NATIVE

La codificación es la predeterminada para el lenguaje de programación y la máquina en la que se ejecuta la aplicación.

Nota: El valor de esta constante depende del lenguaje de programación y del entorno. Por este motivo, las aplicaciones deben compilarse utilizando los archivos de encabezado, macro, COPY o INCLUDE adecuados para el entorno en el que se ejecutará la aplicación.

Las aplicaciones que envían mensajes suelen especificar MQENC_NATIVE. Las aplicaciones que recuperan mensajes deben comparar este campo con el valor MQENC_NATIVE; si los valores difieren, es posible que la aplicación tenga que convertir los datos numéricos del mensaje. Utilice la opción MQGMO_CONVERT para solicitar al gestor de colas que convierta el mensaje como parte del procesamiento de la llamada MQGET. Consulte [“Codificaciones de máquina” en la página 908](#) para obtener más información sobre cómo se construye el Encoding campo.

Si se especifica la opción MQGMO_CONVERT en la llamada MQGET, este campo es un campo de entrada/salida. El valor especificado por la aplicación es la codificación a la que se convertirán los datos del mensaje si es necesario. Si la conversión se realiza correctamente o no es necesaria, el valor permanece sin cambios. Si la conversión no se realiza correctamente, el valor tras la llamada a MQGET representa la codificación del mensaje no convertido que se devuelve a la aplicación.

En otros casos, se trata de un campo de salida para la llamada MQGET y un campo de entrada para las llamadas MQPUT y MQPUT1. El valor inicial de este campo es MQENC_NATIVE.

CodedCharSetId (MQLONG)

Este campo especifica el identificador del conjunto de caracteres de los datos de caracteres dentro del cuerpo del mensaje.

Nota: Los datos de caracteres en MQMD y las demás estructuras de datos de MQ que son parámetros en las llamadas deben estar en el juego de caracteres del gestor de colas. Esto se define mediante el atributo **CodedCharSetId** del gestor de colas; consulte [“Atributos para el gestor de colas” en la página 801](#) para obtener más información sobre este atributo.

Si este campo se establece en MQCCSI_Q_MGR al llamar a MQGET con MQGMO_CONVERT en las opciones, el comportamiento es diferente entre las aplicaciones cliente y servidor. Para las aplicaciones de servidor, la página de códigos utilizada para la conversión de caracteres es la CodedCharSetId del gestor de colas; para las aplicaciones de cliente, la página de códigos utilizada para la conversión de caracteres es la página de códigos de la configuración regional actual.

Para las aplicaciones cliente, se rellena MQCCSI_Q_MGR, basándose en la configuración regional del cliente en lugar de la del gestor de colas. La excepción a esta regla se da cuando se envía un mensaje a una cola puente IMS; lo que se devuelve, en el *CodedCharSetId* campo de MQMD, es el CCSID del gestor de colas.

No debe utilizar el siguiente valor especial:

MQCCSI_APPL

Esto da como resultado un valor incorrecto en el CodedCharSetId campo del MQMD y provoca un código de retorno MQRC_SOURCE_CCSDID_ERROR (o MQRC_FORMAT_ERROR para z/OS) cuando se recibe el mensaje utilizando la llamada MQGET con la opción MQGMO_CONVERT.

Puede utilizar los siguientes valores especiales:

MQCCSI_Q_MGR

Los datos de caracteres del mensaje están en el juego de caracteres del gestor de colas.

En las llamadas MQPUT y MQPUT1, el gestor de colas cambia este valor en el MQMD que se envía con el mensaje al identificador de juego de caracteres verdadero del gestor de colas. Como resultado, el valor MQCCSI_Q_MGR nunca es devuelto por la llamada MQGET.

MQCCSI_POR_DEFAULT

El CodedCharSetId de los datos en el *String* campo se define mediante el CodedCharSetId campo en la estructura de cabecera que precede a la estructura MQCFH, o mediante el CodedCharSetId campo en MQMD si MQCFH se encuentra al inicio del mensaje.

MQCCSI_HEREDAR

Los datos de caracteres del mensaje están en el mismo juego de caracteres que esta estructura; este es el juego de caracteres del gestor de colas. (Solo para MQMD, MQCCSI_INHERIT tiene el mismo significado que MQCCSI_Q_MGR).

El gestor de colas cambia este valor en el MQMD que se envía con el mensaje al identificador de juego de caracteres real de MQMD. Si no se produce ningún error, la llamada MQGET no devuelve el valor MQCCSI_INHERIT.

No utilice MQCCSI_INHERIT si el valor del PutAppType campo en MQMD es MQAT_BROKER.

MQCCSI_INTEGRADO

Los datos de caracteres del mensaje se encuentran en un conjunto de caracteres con el identificador que se incluye en los propios datos del mensaje. Puede haber cualquier número de identificadores de conjunto de caracteres incrustados en los datos del mensaje, aplicables a diferentes partes de los datos. Este valor debe utilizarse para los mensajes PCF (con formato MQFMT_ADMIN, MQFMT_EVENT o MQFMT_PCF) que contengan datos en una combinación de conjuntos de caracteres. Cada estructura MQCFST, MQCFSL y MQCFSF contenida en el mensaje PCF debe tener un identificador de juego de caracteres explícito especificado y no MQCCSI_DEFAULT.

Si un mensaje con formato MQFMT_EMBEDDED_PCF va a contener datos en una mezcla de conjuntos de caracteres, no utilice MQCCSI_EMBEDDED. En su lugar, establezca MQEPH_CCSID_EMBEDDED en el campo Flags de la estructura MQEPH. Esto equivale a establecer MQCCSI_EMBEDDED en la estructura anterior. Cada estructura MQCFST, MQCFSL y MQCFSF contenida en el mensaje PCF debe tener un identificador de conjunto de caracteres explícito especificado y no MQCCSI_DEFAULT. Para obtener más información sobre la estructura del MQEPH, consulte [“MQEPH - Cabecera PCF incrustada” en la página 325](#).

Especifique este valor solo en las llamadas MQPUT y MQPUT1. Si se especifica en la llamada MQGET, impide la conversión del mensaje.

En las llamadas MQPUT y MQPUT1, el gestor de colas cambia los valores MQCCSI_Q_MGR y MQCCSI_INHERIT en el MQMD que se envía con el mensaje, tal y como se ha descrito anteriormente, pero no cambia el MQMD especificado en la llamada MQPUT o MQPUT1. No se realiza ninguna otra comprobación sobre el valor especificado.

Las aplicaciones que recuperan mensajes deben comparar este campo con el valor que espera la aplicación; si los valores difieren, es posible que la aplicación tenga que convertir los datos de caracteres del mensaje.

 En z/OS, el [Encoding](#) campo del MQMD se utiliza para especificar la codificación entera de los datos de caracteres en el cuerpo del mensaje, cuando el CodedCharSetId campo del MQMD indica que la representación del conjunto de caracteres depende de la codificación utilizada para los enteros binarios.

 En [Multiplataformas](#), se supone que el orden de bytes de los datos de caracteres es el mismo que el de la codificación entera nativa de la plataforma en la que se ejecuta el gestor de colas. Esto solo afecta a determinados conjuntos de caracteres multibyte (por ejemplo, los conjuntos de caracteres UTF-16).

Si se especifica la opción MQGMO_CONVERT en la llamada MQGET, este campo es un campo de entrada/salida. El valor especificado por la aplicación es el identificador del juego de caracteres codificados al que se deben convertir los datos del mensaje si es necesario. Si la conversión se realiza correctamente o no es necesaria, el valor no cambia (excepto que el valor MQCCSI_Q_MGR o MQCCSI_INHERIT se

convierte al valor real). Si la conversión no se realiza correctamente, el valor tras la llamada a MQGET representa el identificador del juego de caracteres codificado del mensaje no convertido que se devuelve a la aplicación.

De lo contrario, este es un campo de salida para la llamada MQGET y un campo de entrada para las llamadas MQPUT y MQPUT1. El valor inicial de este campo es MQCCSI_Q_MGR.

Format (MQCHAR8)

Es un nombre que utiliza el remitente del mensaje para indicar al destinatario la naturaleza de los datos que contiene el mensaje. Se puede especificar cualquier carácter del conjunto de caracteres del gestor de colas para el nombre, pero debe restringir el nombre a lo siguiente:

- Mayúsculas de la A a la Z
- Dígitos numéricos del 0 al 9

Si se utilizan otros caracteres, es posible que no se pueda traducir el nombre entre los conjuntos de caracteres de los gestores de colas de envío y recepción.

Rellene el nombre con espacios en blanco hasta alcanzar la longitud del campo, o utilice un carácter nulo para terminar el nombre antes del final del campo; el carácter nulo y cualquier carácter posterior se tratan como espacios en blanco. No especifique un nombre con espacios en blanco al principio o en medio. Para la llamada MQGET, el gestor de colas devuelve el nombre relleno con espacios en blanco hasta alcanzar la longitud del campo.

El gestor de colas no comprueba que el nombre cumpla con las recomendaciones descritas anteriormente.

Los nombres que comienzan p MQ, tanto en mayúsculas como en minúsculas o en mayúsculas y minúsculas, tienen significados definidos por el gestor de colas; no utilice nombres que comiencen con estas letras para sus propios formatos. Los formatos integrados del gestor de colas son:

MQFMT_NONE

La naturaleza de los datos es indefinida: los datos no se pueden convertir cuando el mensaje se recupera de una cola utilizando la opción MQGMO_CONVERT.

Si se especifica MQGMO_CONVERT en la llamada MQGET y el juego de caracteres o la codificación de los datos del mensaje difiere de la especificada en el **MsgDesc** parámetro, el mensaje se devuelve con los siguientes códigos de finalización y motivo (suponiendo que no haya otros errores):

- Código de finalización MQCC_WARNING y código de motivo MQRC_FORMAT_ERROR si los datos MQFMT_NONE se encuentran al principio del mensaje.
- Código de finalización MQCC_OK y código de motivo MQRC_NONE si los datos MQFMT_NONE se encuentran al final del mensaje (es decir, precedidos por una o varias estructuras de encabezado MQ). En este caso, las estructuras de encabezado « MQ » se convierten al juego de caracteres y la codificación solicitados.

Para el lenguaje de programación C, también se define la constante MQFMT_NONE_ARRAY, que tiene el mismo valor que MQFMT_NONE, pero es una matriz de caracteres en lugar de una cadena.

MQFMT_ADMIN

El mensaje es una solicitud o respuesta del servidor de comandos en formato de comando programable (PCF). Los mensajes de este formato se pueden convertir si se especifica la opción MQGMO_CONVERT en la llamada MQGET. Consulte [Uso de formatos de comandos programables](#) para obtener más información sobre el uso de mensajes con formatos de comandos programables.

Para el lenguaje de programación C, también se define la constante MQFMT_ADMIN_ARRAY, que tiene el mismo valor que MQFMT_ADMIN, pero es una matriz de caracteres en lugar de una cadena.

MQFMT_CICS

Los datos del mensaje comienzan con el CICS encabezado de información MQCIH, seguido de los datos de la aplicación. El nombre del formato de los datos de la aplicación viene dado por el Format campo de la estructura MQCIH.

 En z/OS, especifique la opción MQGMO_CONVERT en la llamada MQGET para convertir los mensajes que tienen el formato MQFMT_CICS.

Para el lenguaje de programación C, también se define la constante MQFMT_CICS_ARRAY, que tiene el mismo valor que MQFMT_CICS, pero es una matriz de caracteres en lugar de una cadena.

MQFMT_COMMAND_1

El mensaje es un mensaje de respuesta del servidor de comandos MQSC que contiene el recuento de objetos, el código de finalización y el código de motivo. Los mensajes de este formato se pueden convertir si se especifica la opción MQGMO_CONVERT en la llamada MQGET.

Para el lenguaje de programación C, también se define la constante `MQFMT\_COMMAND\_1\_ARRAY`; esta tiene el mismo valor que `MQFMT\_COMMAND\_1`, pero es una matriz de caracteres en lugar de una cadena.

MQFMT_COMMAND_2

El mensaje es un mensaje de respuesta del servidor de comandos MQSC que contiene información sobre los objetos solicitados. Los mensajes de este formato se pueden convertir si se especifica la opción MQGMO_CONVERT en la llamada MQGET.

Para el lenguaje de programación C, también se define la constante `MQFMT\_COMMAND\_2\_ARRAY`; esta tiene el mismo valor que `MQFMT\_COMMAND\_2`, pero es una matriz de caracteres en lugar de una cadena.

MQFMT_DEAD_LETTER_HEADER

Los datos del mensaje comienzan con el encabezado de carta muerta MQDLH. Los datos del mensaje original siguen inmediatamente a la estructura MQDLH. El nombre del formato de los datos del mensaje original viene dado por el *Format* campo de la estructura MQDLH; consulte [“MQDLH - Cabecera de mensajes no entregados” en la página 311](#) para obtener más detalles sobre esta estructura. Los mensajes de este formato se pueden convertir si se especifica la opción MQGMO_CONVERT en la llamada MQGET.

Los informes COA y COD no se generan para los mensajes que tienen un *Format* de MQFMT_DEAD_LETTER_HEADER.

Para el lenguaje de programación C, también se define la constante MQFMT_DEAD_LETTER_HEADER_ARRAY, que tiene el mismo valor que MQFMT_DEAD_LETTER_HEADER, pero es una matriz de caracteres en lugar de una cadena.

MQFMT_DIST_HEADER

Los datos del mensaje comienzan con el encabezado de la lista de distribución MQDH, que incluye las matrices de registros MQOR y MQPMR. El encabezado de la lista de distribución puede ir seguido de datos adicionales. El formato de los datos adicionales (si los hay) viene dado por el *Format* campo de la estructura MQDH; véase [“MQDH - Cabecera de distribución” en la página 304](#) para obtener más detalles sobre esta estructura. Los mensajes con formato MQFMT_DIST_HEADER se pueden convertir si se especifica la opción MQGMO_CONVERT en la llamada MQGET.

Este formato es compatible con los siguientes entornos:

-  AIX
-  IBM i
-  Linux
-  Windows

y para IBM MQ MQI clientes conectados a estos sistemas.

Para el lenguaje de programación C, también se define la constante MQFMT_DIST_HEADER_ARRAY, que tiene el mismo valor que MQFMT_DIST_HEADER, pero es una matriz de caracteres en lugar de una cadena.

MQFMT_EMBEDDED_PCF

Formato para un mensaje de ruta de seguimiento, siempre que el valor del comando PCF esté establecido en MQCMD_TRACE_ROUTE. El uso de este formato permite enviar los datos del usuario junto con el mensaje de rastreo de ruta, siempre que sus aplicaciones puedan gestionar los parámetros PCF precedentes.

El encabezado PCF debe ser el primer encabezado, o el mensaje no se tratará como un mensaje de ruta de seguimiento. Esto significa que el mensaje no puede estar en un grupo y que los mensajes de rastreo de ruta no pueden segmentarse. Si se envía un mensaje de rastreo de ruta en un grupo, el mensaje se rechaza con el código de motivo MQRC_MSG_NOT_ALLOWED_IN_GROUP.

Tenga en cuenta que MQFMT_ADMIN también se puede utilizar para el formato de un mensaje de ruta de seguimiento, pero en este caso no se pueden enviar datos de usuario junto con el mensaje de ruta de seguimiento.

MQFMT_EVENT

El mensaje es un mensaje de evento MQ que informa de un evento que ha ocurrido. Los mensajes de evento tienen la misma estructura que los comandos programables; consulte [Mensajes de comando PCF](#) para obtener más información sobre esta estructura, y [Supervisión de eventos](#) para obtener información sobre los eventos.

Version-1 Los mensajes de evento se pueden convertir en todos los entornos si se especifica la opción MQGMO_CONVERT en la llamada MQGET. Version-2 Los mensajes de evento solo se pueden convertir en z/OS.

Para el lenguaje de programación C, también se define la constante MQFMT_EVENT_ARRAY, que tiene el mismo valor que MQFMT_EVENT, pero es una matriz de caracteres en lugar de una cadena.

MQFMT_IMS

Los datos del mensaje comienzan con el IMS encabezado de información MQIIH, seguido de los datos de la aplicación. El nombre del formato de los datos de la aplicación viene dado por el Format campo de la estructura MQIIH.

Para obtener más información sobre cómo se gestiona la estructura MQIIH al utilizar MQGET con MQGMO_CONVERT, consulte [“Format \(MQCHAR8\)” en la página 377](#) y [“ReplyToFormat \(MQCHAR8\)” en la página 378](#).

Para el lenguaje de programación C, también se define la constante MQFMT_IMS_ARRAY, que tiene el mismo valor que MQFMT_IMS, pero es una matriz de caracteres en lugar de una cadena.

MQFMT_IMS_VAR_STRING

El mensaje es una IMS cadena variable, que es una cadena con el formato 11zzccc, donde:

11

Es un campo de longitud de 2 bytes que especifica la longitud total del elemento de cadena IMS variable. Esta longitud es igual a la longitud de 11 (2 bytes), más la longitud de zz (2 bytes), más la longitud de la propia cadena de caracteres. 11 es un entero binario de 2 bytes en la codificación especificada por el Encoding campo.

zz

Es un campo de 2 bytes que contiene indicadores significativos para IMS. zz es una cadena de bytes compuesta por dos campos MQBYTE y se transmite sin cambios del remitente al destinatario (es decir, no zz está sujeta a ninguna conversión).

ccc

es una cadena de caracteres de longitud variable que contiene 11-4 caracteres. ccc está en el conjunto de caracteres especificado por el CodedCharSetId campo.

En z/OS, los datos del mensaje pueden consistir en una secuencia de cadenas IMS variables unidas entre sí, siendo cada cadena de la forma 11zzccc. No debe haber bytes omitidos entre cadenas variables IMS sucesivas. Esto significa que si la primera cadena tiene una longitud impar, la segunda cadena estará desalineada, es decir, no comenzará en un límite que sea múltiplo de dos. Ten cuidado al construir cadenas de este tipo en máquinas que requieren la alineación de tipos de datos elementales.

Utilice la opción MQGMO_CONVERT en la llamada MQGET para convertir los mensajes que tienen el formato MQFMT_IMS_VAR_STRING.

Para el lenguaje de programación C, también se define la constante MQFMT_IMS_VAR_STRING_ARRAY, que tiene el mismo valor que MQFMT_IMS_VAR_STRING, pero es una matriz de caracteres en lugar de una cadena.

MQFMT_MD_EXTENSION

Los datos del mensaje comienzan con la extensión del descriptor de mensaje MQMDE y, opcionalmente, van seguidos de otros datos (normalmente, los datos del mensaje de la aplicación). El nombre del formato, el conjunto de caracteres y la codificación de los datos que siguen al MQMDE se indican en los campos Format, CodedCharSetId Encoding y del MQMDE. Consulte [“MQMDE - Extensión del descriptor de mensajes”](#) en la página 443 para obtener más información sobre esta estructura. Los mensajes de este formato se pueden convertir si se especifica la opción MQGMO_CONVERT en la llamada MQGET.

Para el lenguaje de programación C, también se define la constante MQFMT_MD_EXTENSION_ARRAY, que tiene el mismo valor que MQFMT_MD_EXTENSION, pero es una matriz de caracteres en lugar de una cadena.

MQFMT_PCF

El mensaje es un mensaje definido por el usuario que se ajusta a la estructura de un mensaje de formato de comando programable (PCF). Los mensajes de este formato se pueden convertir si se especifica la opción MQGMO_CONVERT en la llamada MQGET. Consulte [Uso de formatos de comandos programables](#) para obtener más información sobre el uso de mensajes con formatos de comandos programables.

Para el lenguaje de programación C, también se define la constante MQFMT_PCF_ARRAY, que tiene el mismo valor que MQFMT_PCF, pero es una matriz de caracteres en lugar de una cadena.

MQFMT_REF_MSG_HEADER

Los datos del mensaje comienzan con el encabezado de mensaje de referencia MQRMH y, opcionalmente, van seguidos de otros datos. El nombre del formato, el juego de caracteres y la codificación de los datos se indican en los campos Format, CodedCharSetId Encoding y del MQRMH. Consulte [“MQRMH - Cabecera de mensaje de referencia”](#) en la página 531 para obtener más información sobre esta estructura. Los mensajes de este formato se pueden convertir si se especifica la opción MQGMO_CONVERT en la llamada MQGET.

Este formato es compatible con los siguientes entornos:

-  AIX
-  IBM i
-  Linux
-  Windows

y para IBM MQ MQI clients conectados a estos sistemas.

Para el lenguaje de programación C, también se define la constante MQFMT_REF_MSG_HEADER_ARRAY, que tiene el mismo valor que MQFMT_REF_MSG_HEADER, pero es una matriz de caracteres en lugar de una cadena.

MQFMT_RF_HEADER

Los datos del mensaje comienzan con las reglas y el encabezado de formato MQRFH, y opcionalmente van seguidos de otros datos. El nombre del formato, el juego de caracteres y la codificación de los datos (si los hay) se indican en los campos Format, CodedCharSetId Encoding y del MQRFH. Los mensajes de este formato se pueden convertir si se especifica la opción MQGMO_CONVERT en la llamada MQGET.

Para el lenguaje de programación C, también se define la constante MQFMT_RF_HEADER_ARRAY, que tiene el mismo valor que MQFMT_RF_HEADER, pero es una matriz de caracteres en lugar de una cadena.

MQFMT_RF_HEADER_2

Los datos del mensaje comienzan con las reglas version-2 y el encabezado de formato MQRFH2, y opcionalmente van seguidos de otros datos. El nombre del formato, el juego de caracteres y la codificación de los datos opcionales (si los hay) se indican en los campos Format, CodedCharSetId Encoding y del campo MQRFH2. Los mensajes de este formato se pueden convertir si se especifica la opción MQGMO_CONVERT en la llamada MQGET.

Para el lenguaje de programación C, también se define la constante ``MQFMT_RF_HEADER_2_ARRAY``; esta tiene el mismo valor que ``MQFMT_RF_HEADER_2``, pero es una matriz de caracteres en lugar de una cadena.

MQFMT_STRING

Los datos del mensaje de la aplicación pueden ser una cadena SBCS (conjunto de caracteres de un solo byte) o una cadena DBCS (conjunto de caracteres de doble byte). Los mensajes de este formato se pueden convertir si se especifica la opción MQGMO_CONVERT en la llamada MQGET.

Para el lenguaje de programación C, también se define la constante `MQFMT_STRING_ARRAY`, que tiene el mismo valor que `MQFMT_STRING`, pero es una matriz de caracteres en lugar de una cadena.

MQFMT_TRIGGER

El mensaje es un mensaje desencadenante, descrito por la estructura MQTM; consulte [“MQTM - Mensaje de activación”](#) en la página 595 para obtener más detalles sobre esta estructura. Los mensajes de este formato se pueden convertir si se especifica la opción MQGMO_CONVERT en la llamada MQGET.

Para el lenguaje de programación C, también se define la constante `MQFMT_TRIGGER_ARRAY`, que tiene el mismo valor que `MQFMT_TRIGGER`, pero es una matriz de caracteres en lugar de una cadena.

MQFMT_WORK_INFO_HEADER

Los datos del mensaje comienzan con el encabezado de información de trabajo MQWIH, seguido de los datos de la aplicación. El nombre del formato de los datos de la aplicación viene dado por el Format campo de la estructura MQWIH.

 En z/OS, especifique la opción MQGMO_CONVERT en la llamada MQGET para convertir los datos de usuario en mensajes que tengan el formato MQFMT_WORK_INFO_HEADER. Sin embargo, la estructura MQWIH siempre se devuelve con el juego de caracteres y la codificación del gestor de colas (es decir, la estructura MQWIH se convierte independientemente de si se especifica o no la opción MQGMO_CONVERT).

Para el lenguaje de programación C, también se define la constante `MQFMT_WORK_INFO_HEADER_ARRAY`, que tiene el mismo valor que `MQFMT_WORK_INFO_HEADER`, pero es una matriz de caracteres en lugar de una cadena.

MQFMT_XMIT_Q_HEADER

Los datos del mensaje comienzan con el encabezado de la cola de transmisión MQXQH. Los datos del mensaje original siguen inmediatamente a la estructura MQXQH. El nombre del formato de los datos del mensaje original viene dado por el Format campo de la estructura MQMD, que forma parte de la cabecera de la cola de transmisión MQXQH. Consulte [“MQXQH - Cabecera de cola de transmisión”](#) en la página 618 para obtener más información sobre esta estructura.

Los informes COA y COD no se generan para los mensajes que tienen un Format de `MQFMT_XMIT_Q_HEADER`.

Para el lenguaje de programación C, también se define la constante `MQFMT_XMIT_Q_HEADER_ARRAY`, que tiene el mismo valor que `MQFMT_XMIT_Q_HEADER`, pero es una matriz de caracteres en lugar de una cadena.

Este es un campo de salida para la llamada MQGET y un campo de entrada para las llamadas MQPUT y MQPUT1. La longitud de este campo viene dada por `MQ_FORMAT_LENGTH`. El valor inicial de este campo es `MQFMT_NONE`.

Priority (MQLONG)

Para las llamadas MQPUT y MQPUT1, el valor debe ser mayor o igual a cero; cero es la prioridad más baja. También se puede utilizar el siguiente valor especial:

MQPRI_PRIORITY_AS_Q_DEF

- Si la cola es una cola de clúster, la prioridad del mensaje se toma del **DefPriority** atributo definido en el gestor de colas *de destino* que posee la instancia concreta de la cola en la que se coloca el mensaje.

Cuando hay varias instancias de la cola de clúster y difieren en este atributo, se seleccionará el valor de uno de ellos y no se puede predecir cuál de ellos se utilizará. Por lo tanto debe establecer este atributo en el mismo valor en todas las instancias. Si este no es el caso, se envía el mensaje de error AMQ9407 a los registros del gestor de colas. Consulte también [¿Cómo se resuelven los atributos de objeto de destino para las colas alias, remotas y de clúster?](#)

El valor de *DefPriority* se copia en el *Priority* campo cuando el mensaje se coloca en la cola de destino. Si *DefPriority* se cambia posteriormente, los mensajes que ya se han colocado en la cola no se ven afectados.

- Si la cola no es una cola de clúster, la prioridad del mensaje se toma del **DefPriority** atributo definido en el gestor de colas *local*, incluso si el gestor de colas de destino es remoto.

Si hay más de una definición en la ruta de resolución del nombre de la cola, la prioridad predeterminada se toma del valor de este atributo en la *primera* definición de la ruta. Este puede ser:

- una cola alias
- Una cola local
- Una definición local de una cola remota
- Un alias del gestor de colas
- Una cola de transmisión (por ejemplo, la *DefXmitQName* cola)

El valor de *DefPriority* se copia en el *Priority* campo cuando se envía el mensaje. Si *DefPriority* se modifica posteriormente, los mensajes que ya se hayan enviado no se verán afectados.

El valor devuelto por la llamada MQGET siempre es mayor o igual a cero; el valor MQPRI_PRIORITY_AS_Q_DEF nunca se devuelve.

Si se envía un mensaje con una prioridad superior al máximo admitido por el gestor de colas local (este máximo viene determinado por el atributo del **MaxPriority** gestor de colas), el mensaje es aceptado por el gestor de colas, pero se coloca en la cola con la prioridad máxima del gestor de colas; la llamada MQPUT o MQPUT1 se completa con MQCC_WARNING y el código de motivo MQRC_PRIORITY_EXCEEDS_MAXIMUM. Sin embargo, el *Priority* campo conserva el valor especificado por la aplicación que envió el mensaje.

Si z/OS se coloca un mensaje con un indicador de prioridad de entrega de mensajes (*MsgSeqNumber*) de 1 en una cola que tiene una secuencia de entrega de mensajes de MQMDS_PRIORITY y un tipo de índice de MQIT_GROUP_ID, la cola podría tratar el mensaje con una prioridad diferente. Si el mensaje se colocó en la cola con una prioridad de 0 o 1, se procesa como si tuviera una prioridad de 2. Esto se debe a que el orden de los mensajes colocados en este tipo de cola está optimizado para permitir pruebas de integridad de grupo eficientes. Para obtener más información sobre la secuencia de entrega de mensajes MQMDS_PRIORITY y el tipo de índice MQIT_GROUP_ID, consulte [el atributo « MsgDeliverySequence »](#).

Al responder a un mensaje, las aplicaciones deben utilizar la prioridad del mensaje de solicitud para el mensaje de respuesta. En otras situaciones, especificar MQPRI_PRIORITY_AS_Q_DEF permite ajustar la prioridad sin modificar la aplicación.

Este es un campo de salida para la llamada MQGET y un campo de entrada para las llamadas MQPUT y MQPUT1. El valor inicial de este campo es MQPRI_PRIORITY_AS_Q_DEF.

Persistence (MQLONG)

Esto indica si el mensaje sobrevive a fallos del sistema y reinicios del gestor de colas. Para las llamadas MQPUT y MQPUT1, el valor debe ser uno de los siguientes:

MQPER_PERSISTENT

El mensaje sobrevive a los fallos del sistema y a los reinicios del gestor de colas. Una vez que se ha enviado el mensaje y se ha confirmado la unidad de trabajo en la que se ha enviado (si el mensaje se envía como parte de una unidad de trabajo), el mensaje se conserva en el almacenamiento auxiliar. Permanece allí hasta que el mensaje se elimina de la cola y se ha confirmado la unidad de trabajo en la que se obtuvo (si el mensaje se recupera como parte de una unidad de trabajo).

Cuando se envía un mensaje persistente a una cola remota, un mecanismo de almacenamiento y reenvío retiene el mensaje en cada gestor de colas a lo largo de la ruta hasta el destino, hasta que se sabe que el mensaje ha llegado al siguiente gestor de colas.

Los mensajes persistentes no pueden colocarse en:

- Colas dinámicas temporales
- Colas compartidas que se asignan a un objeto CFSTRUCT en CFLEVEL(2) o inferior, o en las que el objeto CFSTRUCT se define como RECOVER(NO).

Los mensajes persistentes se pueden colocar en colas dinámicas permanentes y en colas predefinidas.

MQPER_NOT_PERSISTENT

El mensaje no suele sobrevivir a los fallos del sistema ni a los reinicios del gestor de colas. Esto se aplica incluso si se encuentra una copia intacta del mensaje en el almacenamiento auxiliar cuando se reinicia el gestor de colas.

En el caso de las colas NPMCLASS (HIGH), los mensajes no persistentes sobreviven al apagado y reinicio normales del gestor de colas.

Nota: En una configuración HA nativa, cualquier cambio de líder de una instancia del gestor de colas siempre da lugar al descarte de los mensajes no persistentes, independientemente de la configuración de **NPMCLASS**.

En el caso de las colas compartidas, los mensajes no persistentes sobreviven al reinicio del gestor de colas en el grupo de intercambio de colas, pero no sobreviven a los fallos del dispositivo de acoplamiento utilizado para almacenar mensajes en las colas compartidas.

MQPER_PERSISTENCE_AS_Q_DEF

- Si la cola es una cola de clúster, la persistencia del mensaje se toma del **DefPersistence** atributo definido en el gestor de colas *de destino* que posee la instancia concreta de la cola en la que se coloca el mensaje.

Cuando hay varias instancias de la cola de clúster y difieren en este atributo, se seleccionará el valor de uno de ellos y no se puede predecir cuál de ellos se utilizará. Por lo tanto debe establecer este atributo en el mismo valor en todas las instancias. Si este no es el caso, se envía el mensaje de error AMQ9407 a los registros del gestor de colas. Consulte también [¿Cómo se resuelven los atributos de objeto de destino para las colas alias, remotas y de clúster?](#)

El valor de *DefPersistence* se copia en el *Persistence* campo cuando el mensaje se coloca en la cola de destino. Si *DefPersistence* se cambia posteriormente, los mensajes que ya se han colocado en la cola no se ven afectados.

- Si la cola no es una cola de clúster, la persistencia del mensaje se toma del **DefPersistence** atributo definido en el gestor de colas *local*, incluso si el gestor de colas de destino es remoto.

Si hay más de una definición en la ruta de resolución del nombre de la cola, la persistencia predeterminada se toma del valor de este atributo en la *primera* definición de la ruta. Este puede ser:

- una cola alias

- Una cola local
- Una definición local de una cola remota
- Un alias del gestor de colas
- Una cola de transmisión (por ejemplo, la *DefXmitQName* cola)

El valor de *DefPersistence* se copia en el *Persistence* campo cuando se envía el mensaje. Si *DefPersistence* se modifica posteriormente, los mensajes que ya se hayan enviado no se verán afectados.

En una misma cola pueden coexistir mensajes persistentes y no persistentes.

Al responder a un mensaje, las aplicaciones deben utilizar la persistencia del mensaje de solicitud para el mensaje de respuesta.

Para una llamada MQGET, el valor devuelto es MQPER_PERSISTENT o MQPER_NOT_PERSISTENT.

Este es un campo de salida para la llamada MQGET y un campo de entrada para las llamadas MQPUT y MQPUT1. El valor inicial de este campo es MQPER_PERSISTENCE_AS_Q_DEF.

MsgId (MQBYTE24)

Se trata de una cadena de bytes que se utiliza para distinguir un mensaje de otro. Por lo general, no debe haber dos mensajes con el mismo identificador de mensaje, aunque el gestor de colas no lo prohíbe. El identificador del mensaje es una propiedad permanente del mensaje y persiste tras los reinicios del gestor de colas. Dado que el identificador del mensaje es una cadena de bytes y no una cadena de caracteres, el identificador del mensaje no se convierte entre conjuntos de caracteres cuando el mensaje fluye de un gestor de colas a otro.

Para las llamadas MQPUT y MQPUT1, si la aplicación especifica MQMI_NONE o MQPMO_NEW_MSG_ID, el gestor de colas genera un identificador de mensaje único (*MsgId*) cuando se envía el mensaje y lo coloca en el descriptor de mensaje enviado con el mensaje. El gestor de colas también devuelve este identificador de mensaje en el descriptor de mensaje perteneciente a la aplicación emisora. La aplicación puede utilizar este valor para registrar información sobre mensajes concretos y responder a consultas de otras partes de la aplicación.

Nota: Un *MsgId* generado por el gestor de colas consta de un identificador de producto de 4 bytes (AMQ- o CSQ- en ASCII o EBCDIC, donde - representa un carácter en blanco), seguido de una implementación específica del producto de una cadena única. En IBM MQ esto contiene los primeros 12 caracteres del nombre del gestor de colas y un valor derivado del reloj del sistema. Por lo tanto, todos los gestores de colas que puedan comunicarse deben tener nombres que difieran en los primeros 12 caracteres, con el fin de garantizar que los identificadores de mensajes sean únicos. La capacidad de generar una cadena única también depende de que el reloj del sistema no se haya retrasado. Para eliminar la posibilidad de que un identificador de mensaje generado por el gestor de colas duplique uno generado por la aplicación, esta última debe evitar generar identificadores con caracteres iniciales comprendidos entre A e I en ASCII o EBCDIC (X'41' a X'49' y X' C1 ' a X' C9 '). Sin embargo, no se impide que la aplicación genere identificadores con caracteres iniciales dentro de estos rangos.

Si el mensaje se coloca en un tema, el gestor de colas genera identificadores de mensaje únicos según sea necesario para cada mensaje publicado. Si la aplicación especifica MQPMO_NEW_MSG_ID, el gestor de colas genera un identificador de mensaje único para devolverlo en la salida. Si la aplicación especifica MQMI_NONE, el valor del *MsgId* campo en MQMD no cambia al regresar de la llamada.

Consulte la descripción de MQPMO_RETAIN en [“Opciones \(MQLONG\)” en la página 484](#) para obtener más detalles sobre las publicaciones retenidas.

Si el mensaje se envía a una lista de distribución, el gestor de colas genera identificadores de mensaje únicos según sea necesario, pero el valor del *MsgId* campo en MQMD no cambia al regresar de la llamada, incluso si se especificó MQMI_NONE o MQPMO_NEW_MSG_ID. Si la aplicación necesita conocer los identificadores de mensajes generados por el gestor de colas, la aplicación debe proporcionar registros MQPMR que contengan el *MsgId* campo.

La aplicación emisora también puede especificar un valor para el identificador de mensaje distinto de MQMI_NONE; esto impide que el gestor de colas genere un identificador de mensaje único. Una aplicación que reenvía un mensaje puede utilizar esto para propagar el identificador del mensaje original.

El gestor de colas no utiliza este campo, salvo para:

- Genere un valor único si se solicita, tal y como se ha descrito anteriormente
- Entregar el valor a la aplicación que emite la solicitud de obtención del mensaje
- Copie el valor en el *CorrelId* campo de cualquier mensaje de informe que genere sobre este mensaje (dependiendo de las *Report* opciones)

Cuando el gestor de colas o un agente de canal de mensajes genera un mensaje de informe, establece el *MsgId* campo de la forma especificada por el *Report* campo del mensaje original, ya sea MQRO_NEW_MSG_ID o MQRO_PASS_MSG_ID. Las aplicaciones que generan mensajes de informe también deben hacerlo.

Para la llamada MQGET, *MsgId* es uno de los cinco campos que se pueden utilizar para recuperar un mensaje concreto de la cola. Normalmente, la llamada MQGET devuelve el siguiente mensaje de la cola, pero se puede obtener un mensaje concreto especificando uno o varios de los cinco criterios de selección, en cualquier combinación; estos campos son:

- *MsgId*
- *CorrelId*
- *GroupId*
- *MsgSeqNumber*
- *Offset*

La aplicación establece uno o varios de estos campos con los valores requeridos y, a continuación, establece las opciones de coincidencia MQMO_* correspondientes en el *MatchOptions* campo de MQGMO para utilizar esos campos como criterios de selección. Solo los mensajes que tengan los valores especificados en esos campos son candidatos para su recuperación. El valor predeterminado para el *MatchOptions* campo (si no lo modifica la aplicación) es que coincida tanto con el identificador del mensaje como con el identificador de correlación.

► **z/OS** En z/OS, los criterios de selección que puede utilizar están restringidos por el tipo de índice utilizado para la cola. Consulte el **IndexType** atributo cola para obtener más detalles.

Normalmente, el mensaje devuelto es el *primer* mensaje de la cola que cumple los criterios de selección. Pero si se especifica MQGMO_BROWSE_NEXT, el mensaje devuelto es el *siguiente* mensaje que cumple los criterios de selección; la búsqueda de este mensaje comienza con el mensaje *siguiente* a la posición actual del cursor.

Nota: La cola se escanea secuencialmente en busca de un mensaje que satisfaga los criterios de selección, por lo que los tiempos de recuperación son más lentos que si no se especificaran criterios de selección, especialmente si hay que escanear muchos mensajes antes de encontrar uno adecuado. Las excepciones a esto son:

- ► **Multi** Una llamada MQGET en plataformas *CorrelId* multiplataforma de 64 bits donde el *CorrelId* índice elimina la necesidad de realizar un escaneo secuencial verdadero.
- ► **z/OS** una llamada MQGET por *IndexType* en z/OS.

En ambos casos, se mejora el rendimiento de la recuperación.

Consulte la tabla Opciones MQGET relacionadas con mensajes en grupos y segmentos de mensajes lógicos para obtener más información sobre cómo se utilizan los criterios de selección en diversas situaciones.

Especificar MQMI_NONE como identificador de mensaje tiene el mismo efecto que no especificar MQMO_MATCH_MSG_ID, es decir, *cualquier* identificador de mensaje coincide.

Este campo se ignora si se especifica la opción MQGMO_MSG_UNDER_CURSOR en el **GetMsgOpts** parámetro de la llamada MQGET.

Al regresar de una llamada MQGET, el *MsgId* campo se establece en el identificador del mensaje devuelto (si lo hay).

Se puede utilizar el siguiente valor especial:

MQMI_NONE

No se ha especificado ningún identificador de mensaje.

El valor es un cero binario para la longitud del campo.

Para el lenguaje de programación C, también se define la constante MQMI_NONE_ARRAY, que tiene el mismo valor que MQMI_NONE, pero es una matriz de caracteres en lugar de una cadena.

Este es un campo de entrada/salida para las llamadas MQGET, MQPUT y MQPUT1. La longitud de este campo viene dada por MQ_MSG_ID_LENGTH. El valor inicial de este campo es MQMI_NONE.

CorrelId (MQBYTE24)

El campo « CorrelId » es una propiedad del encabezado del mensaje que puede utilizarse para identificar un mensaje específico o un grupo de mensajes.

Se trata de una cadena de bytes que la aplicación puede utilizar para relacionar un mensaje con otro, o para relacionar el mensaje con otras tareas que la aplicación está realizando. El identificador de correlación es una propiedad permanente del mensaje y persiste tras los reinicios del gestor de colas. Dado que el identificador de correlación es una cadena de bytes y no una cadena de caracteres, el identificador de correlación no se convierte entre conjuntos de caracteres cuando el mensaje fluye de un gestor de colas a otro.

Para las llamadas MQPUT y MQPUT1, la aplicación puede especificar cualquier valor. El gestor de colas transmite este valor con el mensaje y lo entrega a la aplicación que emite la solicitud de obtención del mensaje.

Si la aplicación especifica MQPMO_NEW_CORREL_ID, el gestor de colas genera un identificador de correlación único que se envía con el mensaje y también se devuelve a la aplicación de envío en la salida de la llamada MQPUT o MQPUT1.

Un identificador de correlación generado por el gestor de colas consta de un identificador de producto de 3 bytes (AMQ o CSQ en ASCII o EBCDIC), seguido de un byte reservado y una implementación específica del producto de una cadena única. En IBM MQ esta cadena de implementación específica del producto, contiene los primeros 12 caracteres del nombre del gestor de colas y un valor derivado del reloj del sistema. Por lo tanto, todos los gestores de colas que puedan comunicarse deben tener nombres que difieran en los primeros 12 caracteres para garantizar que los identificadores de mensajes sean únicos. La capacidad de generar una cadena única también depende de que el reloj del sistema no se haya retrasado. Para eliminar la posibilidad de que un identificador de mensaje generado por el gestor de colas duplique uno generado por la aplicación, esta última debe evitar generar identificadores con caracteres iniciales comprendidos entre A e I en ASCII o EBCDIC (X'41' a X'49' y X' C1 ' a X' C9 '). Sin embargo, no se impide que la aplicación genere identificadores con caracteres iniciales dentro de estos rangos.

Este identificador de correlación generado se conserva con el mensaje si este se retiene, y se utiliza como identificador de correlación cuando el mensaje se envía como una publicación a los suscriptores que especifican MQCI_NONE en el campo « SubCorrelId » (Identificador de correlación de suscriptor) en el MQSD transmitido en la llamada MQSUB. Consulte [las opciones de MQPMO](#) para obtener más detalles sobre las publicaciones conservadas.

Cuando el gestor de colas o un agente de canal de mensajes genera un mensaje de informe, establece el *CorrelId* campo de la forma especificada por el *Report* campo del mensaje original, ya sea MQRO_COPY_MSG_ID_TO_CORREL_ID o MQRO_PASS_CORREL_ID. Las aplicaciones que generan mensajes de informe también deben hacerlo.

Para la llamada MQGET, *CorrelId* es uno de los cinco campos que se pueden utilizar para seleccionar un mensaje concreto que se va a recuperar de la cola. Consulte la descripción del campo *MsgId* para obtener más información sobre cómo especificar valores para este campo.

Especificar MQCI_NONE como identificador de correlación tiene el mismo efecto que no especificar MQMO_MATCH_CORREL_ID, es decir, *cualquier* identificador de correlación coincidirá.

Si se especifica la opción MQGMO_MSG_UNDER_CURSOR en el **GetMsgOpts** parámetro de la llamada MQGET, este campo se ignora.

Al regresar de una llamada MQGET, el *CorrelId* campo se establece en el identificador de correlación del mensaje devuelto (si lo hay).

Se pueden utilizar los siguientes valores especiales:

MQCI_NONE

No se ha especificado ningún identificador de correlación.

El valor es un cero binario para la longitud del campo.

Para el lenguaje de programación C, también se define la constante MQCI_NONE_ARRAY, que tiene el mismo valor que MQCI_NONE, pero es una matriz de caracteres en lugar de una cadena.

MQCI_NUEVA_SESIÓN

El mensaje es el inicio de una nueva sesión.

Este valor es reconocido por el CICS bridge como indicativo del inicio de una nueva sesión, es decir, el inicio de una nueva secuencia de mensajes.

Para el lenguaje de programación C, también se define la constante MQCI_NEW_SESSION_ARRAY, que tiene el mismo valor que MQCI_NEW_SESSION, pero es una matriz de caracteres en lugar de una cadena.

Para la llamada MQGET, este es un campo de entrada/salida. Para las llamadas MQPUT y MQPUT1, este es un campo de entrada si no se especifica MQPMO_NEW_CORREL_ID, y un campo de salida si se especifica MQPMO_NEW_CORREL_ID. La longitud de este campo viene dada por MQ_CORREL_ID_LENGTH. El valor inicial de este campo es MQCI_NONE.

Nota:

No puede pasar el identificador de correlación de una publicación en una jerarquía. El campo lo utiliza el gestor de colas.

BackoutCount (MQLONG)

Este es un recuento del número de veces que el mensaje ha sido devuelto previamente por la llamada MQGET como parte de una unidad de trabajo y, posteriormente, descartado. Ayuda a la aplicación a detectar errores de procesamiento basados en el contenido de los mensajes. El recuento excluye las llamadas MQGET que especifican cualquiera de las opciones MQGMO_BROWSE_*.

La precisión de este recuento se ve afectada por el **HardenGetBackout** atributo queue; véase [“Atributos para colas”](#) en la página 841.

 En z/OS, un valor de 255 significa que el mensaje se ha cancelado 255 o más veces; el valor devuelto nunca es superior a 255.

Este es un campo de salida para la llamada MQGET. Se ignora para las llamadas MQPUT y MQPUT1. El valor inicial de este campo es 0.

ReplyToQ (MQCHAR48)

Este es el nombre de la cola de mensajes a la que la aplicación que emitió la solicitud de obtención del mensaje envía los mensajes MQMT_REPLY y MQMT_REPORT. El nombre es el nombre local de una cola definida en el gestor de colas identificado por *ReplyToQMgr*. Esta cola no debe ser una cola modelo, aunque el gestor de colas de envío no lo verifica cuando se envía el mensaje.

Para las llamadas MQPUT y MQPUT1, este campo no debe estar en blanco si el *MsgType* campo tiene el valor MQMT_REQUEST, o si el *Report* campo solicita algún mensaje de informe. Sin embargo, el valor especificado (o sustituido) se transmite a la aplicación que emite la solicitud de obtención del mensaje, independientemente del tipo de mensaje.

Si el *ReplyToQMGr* campo está en blanco, el gestor de colas local busca el *ReplyToQ* nombre en sus propias definiciones de colas. Si existe una definición local de una cola remota con este nombre, el *ReplyToQ* valor del mensaje transmitido se sustituye por el valor del **RemoteQName** atributo de la definición de la cola remota, y este valor se devuelve en el descriptor del mensaje cuando la aplicación receptora emite una llamada MQGET para el mensaje. Si no existe una definición local de una cola remota, no *ReplyToQ* se modifica.

Si se especifica el nombre, puede contener espacios en blanco al final; el primer carácter nulo y los caracteres que le siguen se tratan como espacios en blanco. De lo contrario, no se comprueba que el nombre cumpla las reglas de nomenclatura para colas; esto también es válido para el nombre transmitido, si se *ReplyToQ* sustituye en el mensaje transmitido. La única comprobación que se realiza es que se haya especificado un nombre, si las circunstancias lo requieren.

Si no se requiere una cola de respuesta, establezca el *ReplyToQ* campo en blanco o (en el lenguaje de programación C) en la cadena nula, o en uno o más espacios en blanco seguidos de un carácter nulo; no deje el campo sin inicializar.

Para la llamada MQGET, el gestor de colas siempre devuelve el nombre relleno con espacios en blanco hasta alcanzar la longitud del campo.

Si un mensaje que requiere un mensaje de informe no se puede entregar y el mensaje de informe tampoco se puede entregar a la cola especificada, tanto el mensaje original como el mensaje de informe se envían a la cola de mensajes no entregados (véase el **DeadLetterQName** atributo descrito en [“Atributos para el gestor de colas”](#) en la página 801).

Este es un campo de salida para la llamada MQGET y un campo de entrada para las llamadas MQPUT y MQPUT1. La longitud de este campo viene determinada por MQ_Q_NAME_LENGTH. El valor inicial de este campo es la cadena nula en C y 48 caracteres en blanco en otros lenguajes de programación.

ReplyToQMGr (MQCHAR48)

Este es el nombre del gestor de colas al que se debe enviar el mensaje de respuesta o el mensaje de informe. *ReplyToQ* es el nombre local de una cola definida en este gestor de colas.

Si el *ReplyToQMGr* campo está en blanco, el gestor de colas local busca el *ReplyToQ* nombre en sus definiciones de colas. Si existe una definición local de una cola remota con este nombre, el *ReplyToQMGr* valor del mensaje transmitido se sustituye por el valor del **RemoteQMGrName** atributo de la definición de la cola remota, y este valor se devuelve en el descriptor del mensaje cuando la aplicación receptora emite una llamada MQGET para el mensaje. Si no existe una definición local de una cola remota, el *ReplyToQMGr* que se transmite con el mensaje es el nombre del gestor de colas local.

Si se especifica el nombre, puede contener espacios en blanco al final; el primer carácter nulo y los caracteres que le siguen se tratan como espacios en blanco. De lo contrario, no se comprueba si el nombre cumple las reglas de nomenclatura para los gestores de colas, ni si este nombre es conocido por el gestor de colas emisor; esto también es válido para el nombre transmitido, si se *ReplyToQMGr* sustituye en el mensaje transmitido.

Si no se requiere una cola de respuesta, establezca el *ReplyToQMGr* campo en blanco o (en el lenguaje de programación C) en la cadena nula, o en uno o más espacios en blanco seguidos de un carácter nulo; no deje el campo sin inicializar.

Para la llamada MQGET, el gestor de colas siempre devuelve el nombre relleno con espacios en blanco hasta alcanzar la longitud del campo.

Este es un campo de salida para la llamada MQGET y un campo de entrada para las llamadas MQPUT y MQPUT1. La longitud de este campo viene determinada por MQ_Q_MGR_NAME_LENGTH. El valor inicial de este campo es la cadena nula en C y 48 caracteres en blanco en otros lenguajes de programación.

UserIdentifier (MQCHAR12)

Esto forma parte del *contexto* de identidad del mensaje. Para obtener más información sobre el contexto del mensaje, consulte [“MQMD - Descriptor de mensaje”](#) en la página 389 y [Contexto del mensaje](#).

UserIdentifier Especifica el identificador de usuario de la aplicación que originó el mensaje. El gestor de colas trata esta información como datos de caracteres, pero no define su formato. Para obtener información sobre cómo se determina el identificador de usuario a partir del entorno, consulte [Autenticación y autorización: ID de usuario del sistema operativo \(proceso de aplicación\)](#).

 **z/OS** Cuando el gestor de colas genera esta información para una **MQPUT** llamada **MQPUT1** o, utiliza el **AlternateUserId** del **ObjDesc** parámetro de la **MQOPEN** llamada **MQPUT1** o si se ha especificado la opción **MQ00_ALTERNATE_USER_AUTHORITY** o **MQPMO_ALTERNATE_USER_AUTHORITY**. Si no se ha especificado la opción correspondiente, el gestor de colas utiliza un identificador de usuario determinado a partir del entorno.

 **Multi** En IBM MQ for Multiplatforms, el **UserIdentifier** campo suele ser un campo de salida generado por el gestor de colas y rellenado con el ID de usuario asociado a la conexión que envía el mensaje, que a su vez viene determinado por el mecanismo de autenticación y la plataforma utilizados. Sin embargo, para una **MQPUT** llamada **MQPUT1** o puede convertir este campo en un campo de entrada/salida:

- Este campo es un campo de entrada/salida si se especifica **MQPMO_SET_IDENTITY_CONTEXT** o **MQPMO_SET_ALL_CONTEXT** en el **PutMsgOpts** parámetro. Cualquier información que siga a un carácter nulo dentro del campo se descarta. El gestor de colas convierte el carácter nulo y cualquier carácter posterior en espacios en blanco.
- Si desea especificar el campo de ID de usuario en lugar de dejar que el gestor de colas genere esta información, especifique **MQPMO_SET_IDENTITY_CONTEXT** o **MQPMO_SET_ALL_CONTEXT** y, a continuación, especifique un ID de usuario en el **UserIdentifier** campo.
- Si no se especifica **MQPMO_SET_IDENTITY_CONTEXT** o **MQPMO_SET_ALL_CONTEXT**, el **UserIdentifier** campo se ignora en la entrada y es un campo de solo salida.

Para la autenticación **IDPWOS**, la forma en que se almacenan los ID de más de 12 caracteres en el **UserIdentifier** campo varía según la plataforma y IBM MQ la versión:

-  **Windows**  **V 9.4.0** Para IBM MQ 9.4.0 Long Term Support, y IBM MQ 9.4.1 y 9.4.2 Continuous Delivery, solo Windows Las aplicaciones cliente o servidor pueden adoptar un ID de usuario de conexión que tenga más de 12 caracteres. En este caso, el ID de usuario se indica almacenando los primeros 12 caracteres.
-  **Linux**  **MQ Appliance**  **AIX** A partir de IBM MQ 9.4.3, las aplicaciones cliente o servidor en todas Continuous Delivery las plataformas pueden adoptar un ID de usuario de conexión de más de 12 caracteres. En Windows, el ID de usuario se indica almacenando los primeros 12 caracteres. En AIX, Linux y IBM MQ Appliance, también es necesario configurar el **AllowLongUID=Yes** propiedad en un gestor de colas. Si se establece esta propiedad, el ID de usuario se indica almacenando los primeros 11 caracteres con el símbolo + como sufijo.

Tras la finalización satisfactoria de una llamada **MQPUT** **MQPUT1** o, este campo contiene el ID de usuario que se transmitió con el mensaje si se colocó en una cola. Este es el valor de **UserIdentifier** que se conserva con el mensaje si se retiene, pero no se utiliza como **UserIdentifier** cuando el mensaje se envía como una publicación a los suscriptores, ya que estos proporcionan un valor para anular **UserIdentifier** en todas las publicaciones que se les envían. Si el mensaje no tiene contexto, el campo queda completamente en blanco. Para obtener más información sobre las publicaciones retenidas, consulte **MQPMO_RETAIN** en las opciones de **MQPMO**.

Para la **MQGET** llamada, el **UserIdentifier** campo es un campo de salida. La longitud de este campo viene dada por **MQ_USER_ID_LENGTH**. El valor inicial de este campo es la cadena nula en C y 12 caracteres en blanco en otros lenguajes de programación.

Una vez recibido el mensaje, utilice **UserIdentifier** en el **AlternateUserID** campo del **ObjDesc** parámetro de una llamada **MQPUT1** o **MQOPEN** posterior para realizar la comprobación de autorización del **UserIdentifier** usuario en lugar de la aplicación que realiza la apertura. Por ejemplo:

1. El usuario A envía un mensaje.
2. El mensaje es retirado de una cola por la aplicación B.
3. La aplicación B desea realizar más actividades de mensajería bajo la autoridad del usuario A.

Para lograrlo, la aplicación B copia el **UserIdentifier** campo del MQMD del mensaje recibido en el **AlternateUserID** campo del MQOD para el nuevo identificador que están abriendo. Tenga en cuenta que la aplicación B también debe contar con autorizaciones específicas que lo permitan.

AccountingToken (MQBYTE32)

Este es el token contable, parte del *contexto* de identidad del mensaje. Para obtener más información sobre el contexto del mensaje, consulte [“MQMD - Descriptor de mensaje” en la página 389](#) y [Contexto del mensaje](#).

AccountingToken permite a una aplicación cobrar adecuadamente por el trabajo realizado como resultado del mensaje. El gestor de colas trata esta información como una cadena de bits y no comprueba su contenido.

El gestor de colas genera esta información de la siguiente manera:

- El primer byte del campo se establece en la longitud de la información contable presente en los bytes siguientes; esta longitud está comprendida entre cero y 30, y se almacena en el primer byte como un entero binario.
- El segundo byte y los siguientes (según lo especificado en el campo de longitud) se establecen con la información contable adecuada para el entorno.
 -  En z/OS la información contable se establece:
 - Para z/OS batch, la información contable de la tarjeta JES JOB o de un estado JES ACCT en la tarjeta EXEC (los separadores de coma se cambian a X'FF'). Esta información se trunca, si es necesario, a 31 bytes.
 - Para TSO, el número de cuenta del usuario.
 - Para CICS, el identificador de unidad de trabajo LU (6.2) (UEPUOWDS) (26 bytes).
 - Para IMS, el nombre PSB de 8 caracteres concatenado con el token de recuperación de IMS 16 caracteres.
 -  En IBM i, la información contable se establece en el código contable del trabajo.
 -   En AIX and Linux, la información contable se establece en el identificador numérico del usuario, en caracteres ASCII.
 -  En Windows, la información contable se establece en un identificador Windows de seguridad (SID) en formato comprimido. El SID identifica de forma única el identificador de usuario almacenado en el *UserIdentifier* campo. Cuando el SID se almacena en el *AccountingToken* campo, se omite la autoridad de identificación de 6 bytes (ubicada en el tercer byte y los bytes posteriores del SID). Por ejemplo, si el Windows SID tiene una longitud de 28 bytes, se almacenan 22 bytes de información del SID en el *AccountingToken* campo.
- El último byte (byte 32) del campo de contabilidad se establece en el tipo de token de contabilidad (en este caso, MQACTT_NT_SECURITY_ID, x « 0b »):

MQACTT_CICS_LUOW_ID

CICS Identificador LUOW.

MQACTT_NT_ID_DE_SEGURIDAD

Windows identificador de seguridad.

 MQACTT_OS400_ACCOUNT_TOKEN

IBM i ficha contable.

 MQACTT_UNIX_ID_NUMÉRICO

UNIX identificador numérico.

MQACTT_USUARIO

Token contable definido por el usuario.

MQACTT_DESCONOCIDO

Tipo de token contable desconocido.

El tipo de token contable se establece en un valor explícito solo en los siguientes entornos:

-  AIX
-  IBM i
-  Linux
-  Windows

y para IBM MQ MQI clientes conectados a estos sistemas. En otros entornos, el tipo de token contable se establece en el valor **MQACTT_UNKNOWN**. En estos entornos, utilice el *PutApplType* campo para deducir el tipo de token contable recibido.

- Todos los demás bytes se establecen en cero binario.

Para las llamadas **MQPUT** y **MQPUT1**, este es un campo de entrada/salida si se especifica **MQPMO_SET_IDENTITY_CONTEXT** o **MQPMO_SET_ALL_CONTEXT** en el **PutMsgOpts** parámetro. Si no se especifica ni **MQPMO_SET_IDENTITY_CONTEXT** ni **MQPMO_SET_ALL_CONTEXT**, este campo se ignora en la entrada y es un campo de solo salida. Para obtener más información sobre el contexto del mensaje, consulte [Contexto de mensaje](#).

Tras la finalización satisfactoria de una llamada **MQPUT** o **MQPUT1**, este campo contiene el **AccountingToken** que se transmitió con el mensaje si se colocó en una cola. Este será el valor de **AccountingToken** que se conservará con el mensaje si se retiene (consulte la descripción de **MQPMO_RETAIN** en “[Opciones \(MQLONG\)](#)” en la [página 484](#) para obtener más detalles sobre las publicaciones retenidas), pero no se utilizará como **AccountingToken** cuando el mensaje se envíe como publicación a los suscriptores, ya que estos proporcionan un valor para anular **AccountingToken** en todas las publicaciones que se les envían. Si el mensaje no tiene contexto, el campo es completamente binario cero.

Este es un campo de salida para la llamada **MQGET**.

Este campo no está sujeto a ninguna traducción basada en el conjunto de caracteres del gestor de colas; el campo se trata como una cadena de bits, y no como una cadena de caracteres.

El gestor de colas no hace nada con la información de este campo. La aplicación debe interpretar la información si desea utilizarla con fines contables.

Puede utilizar el siguiente valor especial para el **AccountingToken** campo:

MQACT_NONE

No se especifica ningún token contable.

El valor es un cero binario para la longitud del campo.

Para el lenguaje de programación C, también se define la constante **MQACT_NONE_ARRAY**, que tiene el mismo valor que **MQACT_NONE**, pero es una matriz de caracteres en lugar de una cadena.

La longitud de este campo viene dada por **MQ_ACCOUNTING_TOKEN_LENGTH**. El valor inicial de este campo es **MQACT_NONE**.

ApplIdentityData (MQCHAR32)

Esto forma parte del **contexto** de identidad del mensaje. Para obtener más información sobre el contexto del mensaje, consulte [“MQMD - Descriptor de mensaje”](#) en la página 389 y [Contexto del mensaje](#).

ApplIdentityData Es información definida por el conjunto de aplicaciones y que puede utilizarse para proporcionar información adicional sobre el mensaje o su remitente. El gestor de colas trata esta información como datos de caracteres, pero no define su formato. Cuando el gestor de colas genera esta información, está completamente en blanco.

Para las llamadas MQPUT y MQPUT1, este es un campo de entrada/salida si se especifica MQPMO_SET_IDENTITY_CONTEXT o MQPMO_SET_ALL_CONTEXT en el **PutMsgOpts** parámetro. Si hay un carácter nulo, el gestor de colas convierte el carácter nulo y los caracteres siguientes en espacios en blanco. Si no se especifica ni MQPMO_SET_IDENTITY_CONTEXT ni MQPMO_SET_ALL_CONTEXT, este campo se ignora en la entrada y es un campo de solo salida. Para obtener más información sobre el contexto del mensaje, consulte [Contexto de mensaje](#).

Tras la finalización satisfactoria de una llamada MQPUT o MQPUT1, este campo contiene el *ApplIdentityData* que se transmitió con el mensaje si se colocó en una cola. Este será el valor de *ApplIdentityData* que se conservará con el mensaje si se retiene (consulte la descripción de MQPMO_RETAIN para obtener más detalles sobre las publicaciones retenidas), pero no se utilizará como *ApplIdentityData* cuando el mensaje se envíe como publicación a los suscriptores, ya que estos proporcionan un valor para anular *ApplIdentityData* en todas las publicaciones que se les envían. Si el mensaje no tiene contexto, el campo queda completamente en blanco.

Este es un campo de salida para la llamada MQGET. La longitud de este campo viene dada por MQ_APPL_IDENTITY_DATA_LENGTH. El valor inicial de este campo es la cadena nula en C y 32 caracteres en blanco en otros lenguajes de programación.

PutApplType (MQLONG)

Este es el tipo de aplicación que envía el mensaje y forma parte del **contexto** de origen del mensaje. Para obtener más información sobre el contexto del mensaje, consulte [“MQMD - Descriptor de mensaje”](#) en la página 389 y [Contexto del mensaje](#).

PutApplType puede tener uno de los siguientes tipos estándar. También puede definir sus propios tipos, pero solo con valores comprendidos entre MQAT_USER_FIRST y MQAT_USER_LAST.

MQAT_AIX

AIX aplicación (mismo valor que MQAT_UNIX).

MQAT_AMQP

Aplicación del protocolo AMQP

MQAT_CORREDOR

el intermediario de ubicación.

MQAT_CICS

CICS transacción.

MQAT_CICS_PUENTE

CICS bridge.

MQAT_CICS_VSE

CICS/VSE transacción.

MQAT_DOS

IBM MQ MQI client aplicación en PC DOS.

MQAT_DQM

Agente gestor de colas distribuidas.

MQAT_GUARDIÁN

Aplicación Tandem Guardian (mismo valor que MQAT_NSK).

MQAT_IMS

IMS.

MQAT_IMS_PUENTE

IMS puente.

MQAT_JAVA

Java.

MQAT_MVS

MVS o aplicación TSO (mismo valor que MQAT_ZOS).

MQAT_NOTAS_AGENTE

Lotus Notes Solicitud de agente.

MQAT_OS390

OS/390 aplicación (mismo valor que MQAT_ZOS).

MQAT_OS400

IBM i .

MQAT_QMGR

Gestor de colas.

MQAT_UNIX

UNIX .

MQAT_VOS

Aplicación Stratus VOS.

MQAT_VENTANAS

Aplicación Windows de 16 bits.

MQAT_WINDOWS_NT

Aplicación Windows de 32 bits.

MQAT_WLM

z/OS Aplicación de gestión de la carga de trabajo.

MQAT_XCF

XCF.

MQAT_ZOS

z/OS .

MQAT_DEFAULT

Tipo de aplicación predeterminado.

Este es el tipo de aplicación predeterminado para la plataforma en la que se ejecuta la aplicación.

Nota: El valor de esta constante depende del entorno. Por este motivo, compile siempre la aplicación utilizando los archivos de encabezado, inclusión o COPY adecuados para la plataforma en la que se ejecutará la aplicación.

MQAT_DESCONOCIDO

Utilice este valor para indicar que se desconoce el tipo de aplicación, aunque exista otra información contextual.

MQAT_USUARIO_PRIMERO

Valor más bajo para el tipo de aplicación definido por el usuario.

MQAT_USUARIO_ÚLTIMO

Valor más alto para el tipo de aplicación definido por el usuario.

También puede darse el siguiente valor especial:

MQAT_SIN_CONTEXTO

Este valor lo establece el gestor de colas cuando se envía un mensaje sin contexto (es decir, cuando se especifica la opción de contexto MQPMO_NO_CONTEXT).

Cuando se recupera un mensaje, *PutApplType* se puede comprobar este valor para decidir si el mensaje tiene contexto (se recomienda que nunca *PutApplType* se establezca MQAT_NO_CONTEXT, mediante una aplicación que utilice MQPMO_SET_ALL_CONTEXT, si alguno de los otros campos de contexto no está en blanco).

Cuando el gestor de colas genera esta información como resultado de una aplicación, el campo se establece en un valor determinado por el entorno. En IBM i, está configurado en MQAT_OS400; el gestor de colas nunca utiliza MQAT_CICS en IBM i.

Para las llamadas MQPUT y MQPUT1, este es un campo de entrada/salida si se especifica MQPMO_SET_ALL_CONTEXT en el **PutMsgOpts** parámetro. Si no se especifica MQPMO_SET_ALL_CONTEXT, este campo se ignora en la entrada y es un campo de solo salida.

Este es un campo de salida para la llamada MQGET. El valor inicial de este campo es MQAT_NO_CONTEXT.

PutApplName (MQCHAR28)

Este es el nombre de la aplicación que envió el mensaje y forma parte del *contexto* de origen del mensaje. El contenido varía según la plataforma y puede variar entre versiones.

Para obtener más información sobre el contexto del mensaje, consulte [“MQMD - Descriptor de mensaje” en la página 389 y Contexto del mensaje](#).

Puede especificar el nombre de la aplicación en otros lenguajes de programación. Para obtener más información, consulte [Especificar el nombre de la aplicación en lenguajes de programación soportados](#).

El formato de *PutApplName* depende del valor de *PutApplType* y puede cambiar de una versión a otra. Los cambios son poco frecuentes, pero se producen si cambia el entorno.

Cuando el gestor de colas establece este campo (es decir, para todas las opciones excepto MQPMO_SET_ALL_CONTEXT), lo establece en un valor determinado por el entorno:

-  **z/OS** En z/OS, el gestor de colas utiliza:
 - Para z/OS batch, el nombre del trabajo de 8 caracteres de la tarjeta JES JOB
 - Para TSO, el identificador de usuario TSO de 7 caracteres
 - Para CICS, el applid de 8 caracteres, seguido del tranid de 4 caracteres
 - Para IMS, el identificador del sistema de IMS 8 caracteres, seguido del nombre PSB de 8 caracteres
 - Para XCF, el nombre del grupo XCF de 8 caracteres, seguido del nombre del miembro XCF de 16 caracteres
 - Para un mensaje generado por un gestor de colas, los primeros 28 caracteres del nombre del gestor de colas
 - Para colas distribuidas sin CICS, el nombre de trabajo de 8 caracteres del iniciador del canal seguido del nombre de 8 caracteres del módulo que coloca en la cola de mensajes no entregados seguido de un identificador de tarea de 8 caracteres.

El nombre o los nombres se rellenan a la derecha con espacios en blanco, al igual que cualquier espacio restante en el campo. Cuando hay más de un nombre, no hay separador entre ellos.

-  **Windows** En Windows los sistemas, el gestor de colas utiliza los siguientes nombres:
 - Para una CICS solicitud, el nombre CICS de la transacción
 - Para una aplicación CICS no ejecutable, los 28 caracteres más a la derecha del nombre completo del ejecutable
-  **IBM i** En IBM i, el gestor de colas utiliza el nombre completo del trabajo.
-  **Linux**  **AIX** En AIX and Linux, el gestor de colas utiliza los siguientes nombres:
 - Para una CICS solicitud, el nombre CICS de la transacción
 - Para una aplicación CICS no compatible, MQ solicita al sistema operativo el nombre del proceso. Esto se devuelve como el nombre del archivo del programa, sin la ruta completa. A continuación, MQ coloca el nombre de este proceso en el campo « MQMD.PutApplName » (Nombre del proceso) de la siguiente manera:

AIX

Si el nombre tiene menos de 28 bytes, se inserta el nombre, rellenando con espacios a la derecha.

Si el nombre tiene más de 28 bytes, se insertan los 28 bytes más a la izquierda del nombre.

Linux

Si el nombre tiene menos de 15 bytes, se inserta el nombre, rellenando con espacios a la derecha.

Si el nombre tiene más de 15 bytes, se insertan los 15 bytes más a la izquierda del nombre, rellenando con espacios a la derecha.

Por ejemplo, si ejecuta `/opt/mqm/samp/bin/amqspuT QNAME QMNAME`, entonces el `PutApplName` es `'amqspuT '`. Hay 21 caracteres de relleno en este campo `MQCHAR28`. Tenga en cuenta que la ruta completa, incluyendo, `/opt/mqm/samp/bin` no se incluye en el `PutApplName`.

Para las llamadas `MQPUT` y `MQPUT1`, este es un campo de entrada/salida si se especifica `MQPMO_SET_ALL_CONTEXT` en el **PutMsgOpts** parámetro. Cualquier información que siga a un carácter nulo dentro del campo se descarta. El carácter nulo y cualquier carácter posterior son convertidos en espacios en blanco por el gestor de colas. Si no se especifica `MQPMO_SET_ALL_CONTEXT`, este campo se ignora en la entrada y es un campo de solo salida.

PutDate (MQCHAR8)

Esta es la fecha en la que se envió el mensaje y forma parte del **contexto** de origen del mensaje. Para obtener más información sobre el contexto del mensaje, consulte [“MQMD - Descriptor de mensaje”](#) en la página 389 y [Contexto del mensaje](#).

El formato utilizado para la fecha cuando este campo es generado por el gestor de colas es:

- AAAAMMDD

donde los caracteres representan:

AAAA

año (cuatro dígitos numéricos)

MM

mes del año (del 01 al 12)

DD

día del mes (del 01 al 31)

La hora del meridiano de Greenwich (GMT) se utiliza para los campos *PutDate* *PutTime* y, siempre que el reloj del sistema esté ajustado con precisión a la GMT.

Si el mensaje se incluyó como parte de una unidad de trabajo, la fecha es aquella en la que se incluyó el mensaje, y no la fecha en la que se confirmó la unidad de trabajo.

Para las llamadas `MQPUT` y `MQPUT1`, este es un campo de entrada/salida si se especifica `MQPMO_SET_ALL_CONTEXT` en el **PutMsgOpts** parámetro. El administrador de colas no verifica el contenido del campo, excepto que cualquier información que aparezca después de un carácter nulo dentro del campo es descartada. El gestor de colas convierte el carácter nulo y cualquier carácter posterior en espacios en blanco. Si no se especifica `MQPMO_SET_ALL_CONTEXT`, este campo se ignora en la entrada y es un campo de solo salida.

Este es un campo de salida para la llamada `MQGET`. La longitud de este campo viene dada por `MQ_PUT_DATE_LENGTH`. El valor inicial de este campo es la cadena nula en C y 8 caracteres en blanco en otros lenguajes de programación.

PutTime (MQCHAR8)

Este es el momento en que se envió el mensaje y forma parte del **contexto** original del mensaje. Para obtener más información sobre el contexto del mensaje, consulte [“MQMD - Descriptor de mensaje”](#) en la [página 389](#) y [Contexto del mensaje](#).

El formato utilizado para la hora en que este campo es generado por el gestor de colas es:

- HHMMSSSTH

donde los caracteres representan (en orden):

HH

horas (de 00 a 23)

MM

minutos (de 00 a 59)

SS

segundos (de 00 a 59; véase la nota)

T

décimas de segundo (0 a 9)

H

centésimas de segundo (0 a 9)

Nota: Si el reloj del sistema está sincronizado con un estándar de tiempo muy preciso, en raras ocasiones es posible que se devuelva 60 o 61 para los segundos en *PutTime*. Esto ocurre cuando se insertan segundos intercalares en el estándar horario global.

La hora del meridiano de Greenwich (GMT) se utiliza para los campos *PutDate* *PutTime* y, siempre que el reloj del sistema esté ajustado con precisión a la GMT.

Si el mensaje se incluyó como parte de una unidad de trabajo, la hora es la hora en que se incluyó el mensaje, y no la hora en que se confirmó la unidad de trabajo.

Para las llamadas MQPUT y MQPUT1, este es un campo de entrada/salida si se especifica MQPMO_SET_ALL_CONTEXT en el **PutMsgOpts** parámetro. El gestor de colas no comprueba el contenido del campo, salvo que se descarte cualquier información que siga a un carácter nulo dentro del campo. El gestor de colas convierte el carácter nulo y cualquier carácter posterior en espacios en blanco. Si no se especifica MQPMO_SET_ALL_CONTEXT, este campo se ignora en la entrada y es un campo de solo salida.

Este es un campo de salida para la llamada MQGET. La longitud de este campo viene dada por MQ_PUT_TIME_LENGTH. El valor inicial de este campo es la cadena nula en C y 8 caracteres en blanco en otros lenguajes de programación.

ApplOriginData (MQCHAR4)

Esto forma parte del *contexto* original del mensaje. Para obtener más información sobre el contexto del mensaje, consulte [“MQMD - Descriptor de mensaje”](#) en la [página 389](#) y [Contexto del mensaje](#).

ApplOriginData Es información definida por el conjunto de aplicaciones que se puede utilizar para proporcionar información adicional sobre el origen del mensaje. Por ejemplo, podrían establecerlo las aplicaciones que se ejecutan con la autoridad de usuario adecuada para indicar si los datos de identidad son fiables.

El gestor de colas trata esta información como datos de caracteres, pero no define su formato. Cuando el gestor de colas genera esta información, está completamente en blanco.

Para las llamadas MQPUT y MQPUT1, este es un campo de entrada/salida si se especifica MQPMO_SET_ALL_CONTEXT en el **PutMsgOpts** parámetro. Cualquier información que siga a un carácter nulo dentro del campo se descarta. El gestor de colas convierte el carácter nulo y cualquier carácter posterior en espacios en blanco. Si no se especifica MQPMO_SET_ALL_CONTEXT, este campo se ignora en la entrada y es un campo de solo salida.

Este es un campo de salida para la llamada MQGET. La longitud de este campo viene dada por MQ_APPL_ORIGIN_DATA_LENGTH. El valor inicial de este campo es la cadena nula en C y 4 caracteres en blanco en otros lenguajes de programación.

Cuando se publica el mensaje, aunque ApplOriginData esté configurado, aparece en blanco en la suscripción que recibe.

GroupId (MQBYTE24)

Se trata de una cadena de bytes que se utiliza para identificar el grupo de mensajes concreto o el mensaje lógico al que pertenece el mensaje físico. *GroupId* también se utiliza si se permite la segmentación del mensaje. En todos estos casos, *GroupId* tiene un valor distinto de nulo y se establece uno o varios de los siguientes indicadores en el *MsgFlags* campo :

- MQMF_MSG_EN_GRUPO
- Último mensaje en grupo
- MQMF_SEGMENTO
- MQMF_ÚLTIMO_SEGMENTO
- MQMF_SEGMENTATION_ALLOWED

Si no se establece ninguna de estas banderas, *GroupId* tiene el valor nulo especial MQGI_NONE.

La aplicación no necesita establecer este campo en la llamada MQPUT o MQGET si:

- En la llamada MQPUT, se especifica MQPMO_LOGICAL_ORDER.
- En la llamada MQGET, no se especifica MQMO_MATCH_GROUP_ID.

Estas son las formas recomendadas de utilizar estas llamadas para mensajes que no son mensajes de informe. Sin embargo, si la aplicación requiere más control, o si la llamada es de tipo « MQPUT1 », la aplicación debe asegurarse de que *GroupId* se establezca en un valor adecuado.

Los grupos y segmentos de mensajes solo se pueden procesar correctamente si el identificador del grupo es único. Por este motivo, *las aplicaciones no deben generar sus propios identificadores de grupo*; en su lugar, las aplicaciones deben realizar una de las siguientes acciones:

- Si se especifica MQPMO_LOGICAL_ORDER, el gestor de colas genera automáticamente un identificador de grupo único para el primer mensaje del grupo o segmento del mensaje lógico, y utiliza ese identificador de grupo para los mensajes restantes del grupo o segmentos del mensaje lógico, por lo que la aplicación no necesita realizar ninguna acción especial. Este es el procedimiento recomendado.
- Si no se especifica MQPMO_LOGICAL_ORDER, la aplicación debe solicitar al gestor de colas que genere el identificador de grupo, estableciendo *GroupId* en MQGI_NONE en la primera llamada MQPUT o MQPUT1 para un mensaje del grupo o segmento del mensaje lógico. El identificador de grupo devuelto por el gestor de colas en la salida de esa llamada debe utilizarse entonces para los mensajes restantes del grupo o segmentos del mensaje lógico. Si un grupo de mensajes contiene mensajes segmentados, se debe utilizar el mismo identificador de grupo para todos los segmentos y mensajes del grupo.

Cuando no se especifica MQPMO_LOGICAL_ORDER, los mensajes de los grupos y segmentos de mensajes lógicos se pueden colocar en cualquier orden (por ejemplo, en orden inverso), pero el identificador de grupo debe ser asignado por la *primera* llamada MQPUT o MQPUT1 que se emita para cualquiera de esos mensajes.

En las llamadas de entrada a MQPUT y MQPUT1, el gestor de colas utiliza el valor descrito en Orden físico en una cola. En la salida de las llamadas MQPUT y MQPUT1, el gestor de colas establece este campo en el valor que se envió con el mensaje si el objeto abierto es una cola única y no una lista de distribución, pero lo deja sin cambios si el objeto abierto es una lista de distribución. En este último caso, si la aplicación necesita conocer los identificadores de grupo generados, deberá proporcionar registros MQPMR que contengan el *GroupId* campo.

Al introducir la llamada MQGET, el gestor de colas utiliza el valor descrito en Tabla 510 en la página 358. En la salida de la llamada MQGET, el gestor de colas establece este campo con el valor del mensaje recuperado.

Se define el siguiente valor especial:

MQGI_NONE

No se ha especificado ningún identificador de grupo.

El valor es un cero binario para la longitud del campo. Este es el valor que se utiliza para los mensajes que no están en grupos, ni son segmentos de mensajes lógicos, y para los que no se permite la segmentación.

Para el lenguaje de programación C, también se define la constante MQGI_NONE_ARRAY, que tiene el mismo valor que MQGI_NONE, pero es una matriz de caracteres en lugar de una cadena.

La longitud de este campo viene dada por MQ_GROUP_ID_LENGTH. El valor inicial de este campo es MQGI_NONE. Este campo se ignora si *Version* es menor que MQMD_VERSION_2.

MsgSeqNumber (MQLONG)

Es el número de secuencia de un mensaje lógico dentro de un grupo.

Los números de secuencia comienzan en 1 y aumentan en 1 por cada nuevo mensaje lógico del grupo, hasta un máximo de 999 999 999. Un mensaje físico que no está en un grupo tiene un número de secuencia de 1.

La aplicación no tiene que establecer este campo en la llamada MQPUT o MQGET si:

- En la llamada MQPUT, se especifica MQPMO_LOGICAL_ORDER.
- En la llamada MQGET, no se especifica MQMO_MATCH_MSG_SEQ_NUMBER.

Estas son las formas recomendadas de utilizar estas llamadas para mensajes que no son mensajes de informe. Sin embargo, si la aplicación requiere más control, o si la llamada es de tipo « MQPUT1 », la aplicación debe asegurarse de que *MsgSeqNumber* se establezca en un valor adecuado.

En las llamadas de entrada a MQPUT y MQPUT1, el gestor de colas utiliza el valor descrito en [Orden físico en una cola](#). En la salida de las llamadas MQPUT y MQPUT1, el gestor de colas establece este campo con el valor que se envió con el mensaje.

Al introducir la llamada MQGET, el gestor de colas utiliza el valor que se muestra en [Tabla 510 en la página 358](#). En la salida de la llamada MQGET, el gestor de colas establece este campo con el valor del mensaje recuperado.

El valor inicial de este campo es uno. Este campo se ignora si *Version* es menor que MQMD_VERSION_2.

Offset (MQLONG)

Es el desplazamiento en bytes de los datos del mensaje físico desde el inicio del mensaje lógico del que forman parte los datos. Estos datos se denominan *segmento*. El desplazamiento está comprendido entre 0 y 999 999 999. Un mensaje físico que no es un segmento de un mensaje lógico tiene un desplazamiento de cero.

La aplicación no necesita establecer este campo en la llamada MQPUT o MQGET si:

- En la llamada MQPUT, se especifica MQPMO_LOGICAL_ORDER.
- En la llamada MQGET, no se especifica MQMO_MATCH_OFFSET.

Estas son las formas recomendadas de utilizar estas llamadas para mensajes que no son mensajes de informe. Sin embargo, si la aplicación no cumple estas condiciones, o si la llamada es de tipo MQPUT1, la aplicación debe asegurarse de que *Offset* se establece en un valor adecuado.

En las llamadas de entrada a MQPUT y MQPUT1, el gestor de colas utiliza el valor descrito en [Orden físico en una cola](#). En la salida de las llamadas MQPUT y MQPUT1, el gestor de colas establece este campo con el valor que se envió con el mensaje.

Para un mensaje de informe que informa sobre un segmento de un mensaje lógico, el *OriginalLength* campo (siempre que no sea MQOL_UNDEFINED) se utiliza para actualizar el desplazamiento en la información del segmento conservada por el gestor de colas.

Al introducir la llamada MQGET, el gestor de colas utiliza el valor que se muestra en [Tabla 510](#) en la [página 358](#). En la salida de la llamada MQGET, el gestor de colas establece este campo con el valor del mensaje recuperado.

El valor inicial de este campo es cero. Este campo se ignora si *Version* es menor que MQMD_VERSION_2.

MsgFlags (MQLONG)

MsgFlags son indicadores que especifican los atributos del mensaje o controlan su procesamiento. Se dividen en las siguientes categorías:

- Indicadores de segmentación
- Indicadores de estado

Indicadores de segmentación

Cuando un mensaje es demasiado grande para una cola, el intento de colocarlo en la cola suele fallar. La segmentación es una técnica mediante la cual el gestor de colas o la aplicación divide el mensaje en partes más pequeñas denominadas segmentos y coloca cada segmento en la cola como un mensaje físico independiente. La aplicación que recupera el mensaje puede recuperar los segmentos uno por uno o solicitar al gestor de colas que vuelva a ensamblar los segmentos en un único mensaje que se devuelve mediante la llamada MQGET. Esto último se consigue especificando la opción MQGMO_COMPLETE_MSG en la llamada MQGET y proporcionando un búfer lo suficientemente grande como para alojar el mensaje completo. (Consulte [“MQGMO - Opciones para obtener mensajes”](#) en la [página 331](#) para obtener más información sobre la opción MQGMO_COMPLETE_MSG) Un mensaje se puede segmentar en el gestor de colas de envío, en un gestor de colas intermedio o en el gestor de colas de destino.

Puede especificar una de las siguientes opciones para controlar la segmentación de un mensaje:

MQMF_SEGMENTACIÓN_INHIBIDA

Esta opción evita que el mensaje sea dividido en segmentos por el gestor de colas. Si se especifica para un mensaje que ya es un segmento, esta opción evita que el segmento se divida en segmentos más pequeños.

El valor de esta bandera es cero binario. Este es el valor predeterminado.

MQMF_SEGMENTATION_ALLOWED

Esta opción permite que el mensaje sea dividido en segmentos por el gestor de colas. Si se especifica para un mensaje que ya es un segmento, esta opción permite dividir el segmento en segmentos más pequeños. MQMF_SEGMENTATION_ALLOWED se puede configurar sin necesidad de configurar MQMF_SEGMENT o MQMF_LAST_SEGMENT.

- En z/OS, el gestor de colas no admite la segmentación de mensajes. Si un mensaje es demasiado grande para la cola, la llamada MQPUT o MQPUT1 falla con el código de motivo MQRC_MSG_TOO_BIG_FOR_Q. Sin embargo, la opción MQMF_SEGMENTATION_ALLOWED aún se puede especificar y permite que el mensaje se segmente en un gestor de colas remoto.

Cuando el gestor de colas segmenta un mensaje, activa el indicador MQMF_SEGMENT en la copia del MQMD que se envía con cada segmento, pero no modifica la configuración de estos indicadores en el MQMD proporcionado por la aplicación en la llamada MQPUT o MQPUT1. Para el último segmento del mensaje lógico, el gestor de colas también activa el indicador MQMF_LAST_SEGMENT en el MQMD que se envía con el segmento.

Nota: Ten cuidado al enviar mensajes con MQMF_SEGMENTATION_ALLOWED pero sin MQPMO_LOGICAL_ORDER. Si el mensaje es:

- No es un segmento, y
- No en grupo, y
- No se reenvía,

La aplicación debe restablecer el *GroupId* campo a MQGI_NONE antes de cada llamada a MQPUT o MQPUT1, de modo que el gestor de colas pueda generar un identificador de grupo único para cada

mensaje. Si no se hace así, mensajes no relacionados pueden tener el mismo identificador de grupo, lo que podría llevar a un procesamiento incorrecto después. Consulte las descripciones del *GroupId* campo y la opción MQPMO_LOGICAL_ORDER para obtener más información sobre cuándo restablecer el *GroupId* campo.

El gestor de colas divide los mensajes en segmentos según sea necesario para que los segmentos (más cualquier dato de encabezado necesario) quepan en la cola. Sin embargo, existe un límite inferior para el tamaño de un segmento generado por el gestor de colas, y solo el último segmento creado a partir de un mensaje puede ser inferior a este límite (el límite inferior para el tamaño de un segmento generado por una aplicación es de un byte). Los segmentos generados por el gestor de colas pueden tener longitudes desiguales. El gestor de colas procesa el mensaje de la siguiente manera:

- Los formatos definidos por el usuario se dividen en límites que son múltiplos de 16 bytes; el gestor de colas no genera segmentos que sean menores de 16 bytes (excepto el último segmento).
- Los formatos integrados distintos de MQFMT_STRING se dividen en puntos adecuados según la naturaleza de los datos presentes. Sin embargo, el gestor de colas nunca divide un mensaje en medio de una estructura IBM MQ de encabezado. Esto significa que un segmento que contiene una única estructura de encabezado MQ no puede ser dividido por el gestor de colas y, como resultado, el tamaño mínimo posible del segmento para ese mensaje es superior a 16 bytes.

El segundo segmento o los siguientes generados por el gestor de colas comienzan con uno de los siguientes:

- Estructura de encabezado « MQ »
- El inicio de los datos del mensaje de la aplicación
- Parte del camino a través de los datos del mensaje de la aplicación
- MQFMT_STRING se divide sin tener en cuenta la naturaleza de los datos presentes (SBCS, DBCS o SBCS/DBCS mixtos). Cuando la cadena es DBCS o una mezcla de SBCS/DBCS, esto puede dar lugar a segmentos que no se pueden convertir de un juego de caracteres a otro. El gestor de colas nunca divide los mensajes MQFMT_STRING en segmentos menores de 16 bytes (excepto el último segmento).
- El gestor de colas establece los campos *Format*, *CodedCharSetId* *Encoding* y en el MQMD de cada segmento para describir correctamente los datos presentes al *inicio* del segmento; el nombre del formato es el nombre de un formato integrado o el nombre de un formato definido por el usuario.
- Se modifica el *Report* campo del MQMD de los segmentos con un valor *Offset* superior a cero. Para cada tipo de informe, si la opción de informe es MQRO_*_WITH_DATA, pero el segmento no puede contener ninguno de los primeros 100 bytes de datos de usuario (es decir, los datos que siguen a cualquier estructura IBM MQ de encabezado que pueda estar presente), la opción de informe se cambia a MQRO_*.

El gestor de colas sigue las reglas anteriores, pero por lo demás divide los mensajes de forma impredecible; no haga suposiciones sobre dónde se divide un mensaje.

En el caso de los mensajes *persistentes*, el gestor de colas solo puede realizar la segmentación dentro de una unidad de trabajo:

- Si la llamada MQPUT o MQPUT1 se ejecuta dentro de una unidad de trabajo definida por el usuario, se utiliza dicha unidad de trabajo. Si la llamada falla durante el proceso de segmentación, el gestor de colas elimina cualquier segmento que se haya colocado en la cola como resultado de la llamada fallida. Sin embargo, el fallo no impide que la unidad de trabajo se confirme correctamente.
- Si la llamada se realiza fuera de una unidad de trabajo definida por el usuario y no existe ninguna unidad de trabajo definida por el usuario, el administrador de colas crea una unidad de trabajo solo por la duración de la llamada. Si la llamada se realiza correctamente, el gestor de colas confirma la unidad de trabajo automáticamente. Si la llamada falla, el gestor de colas revierte la unidad de trabajo.

- Si la llamada se ejecuta fuera de una unidad de trabajo definida por el usuario, pero existe una unidad de trabajo definida por el usuario, el gestor de colas no puede realizar la segmentación. Si el mensaje no requiere segmentación, la llamada puede seguir realizándose correctamente. Pero si el mensaje requiere segmentación, la llamada falla con el código de motivo MQRC_UOW_NOT_AVAILABLE.

Para los mensajes *no persistentes*, el gestor de colas no requiere que haya una unidad de trabajo disponible para realizar la segmentación.

Ten especial cuidado al convertir datos en mensajes que puedan estar segmentados:

- Si la aplicación receptora convierte los datos en la llamada MQGET y especifica la opción MQGMO_COMPLETE_MSG, se pasa a la salida de conversión de datos el mensaje completo para que la salida lo convierta, y la salida puede ver claramente que el mensaje se ha segmentado.
- Si la aplicación receptora recupera un segmento cada vez, se invoca la salida de conversión de datos para convertir un segmento cada vez. Por lo tanto, la salida debe convertir los datos de un segmento independientemente de los datos de cualquiera de los otros segmentos.

Si la naturaleza de los datos del mensaje es tal que la segmentación arbitraria de los datos en límites de 16 bytes puede dar lugar a segmentos que no pueden ser convertidos por la salida, o si el formato es MQFMT_STRING y el juego de caracteres es DBCS o una mezcla de SBCS/DBCS, la aplicación emisora debe crear y colocar los segmentos, especificando MQMF_SEGMENTATION_INHIBITED para suprimir una mayor segmentación. De esta manera, la aplicación emisora puede garantizar que cada segmento contenga información suficiente para que la salida de conversión de datos pueda convertir el segmento correctamente.

- Si se especifica la conversión del remitente para un agente de canal de mensajes de envío (MCA), el MCA solo convierte los mensajes que no son segmentos de mensajes lógicos; el MCA nunca intenta convertir mensajes que son segmentos.

Esta bandera es una bandera de entrada en las llamadas MQPUT y MQPUT1, y una bandera de salida en la llamada MQGET. En la última llamada, el gestor de colas también repite el valor del indicador en el *Segmentation* campo de MQGMO.

El valor inicial de este indicador es MQMF_SEGMENTATION_INHIBITED.

Indicadores de estado

Son indicadores que señalan si el mensaje físico pertenece a un grupo de mensajes, es un segmento de un mensaje lógico, ambas cosas o ninguna. Se puede especificar uno o varios de los siguientes elementos en la llamada MQPUT o MQPUT1, o devolverlos mediante la llamada MQGET:

MQMF_MSG_EN_GRUPO

El mensaje es miembro de un grupo.

Último mensaje en grupo

El mensaje es el último mensaje lógico de un grupo.

Si se establece este indicador, el gestor de colas activa MQMF_MSG_IN_GROUP en la copia de MQMD que se envía con el mensaje, pero no modifica la configuración de estos indicadores en el MQMD proporcionado por la aplicación en la llamada MQPUT o MQPUT1.

Es válido que un grupo esté compuesto por un solo mensaje lógico. Si este es el caso, se establece MQMF_LAST_MSG_IN_GROUP, pero el *MsgSeqNumber* campo tiene el valor uno.

MQMF_SEGMENTO

El mensaje es un segmento de un mensaje lógico.

Cuando se especifica MQMF_SEGMENT sin MQMF_LAST_SEGMENT, la longitud de los datos del mensaje de aplicación en el segmento (*excluyendo* las longitudes de cualquier estructura IBM MQ de encabezado que pueda estar presente) debe ser al menos uno. Si la longitud es cero, la llamada MQPUT o MQPUT1 falla con el código de motivo MQRC_SEGMENT_LENGTH_ZERO.

En z/OS, esta opción no es compatible si el mensaje se coloca en una cola que tiene un tipo de índice MQIT_GROUP_ID.

MQMF_ÚLTIMO_SEGMENTO

El mensaje es el último segmento de un mensaje lógico.

Si se establece este indicador, el gestor de colas activa MQMF_SEGMENT en la copia de MQMD que se envía con el mensaje, pero no modifica la configuración de estos indicadores en el MQMD proporcionado por la aplicación en la llamada MQPUT o MQPUT1.

Un mensaje lógico puede constar de un solo segmento. Si es así, MQMF_LAST_SEGMENT está establecido, pero el *Offset* campo tiene el valor cero.

Cuando se especifica MQMF_LAST_SEGMENT, la longitud de los datos del mensaje de aplicación en el segmento (*excluyendo* las longitudes de cualquier estructura de encabezado que pueda estar presente) puede ser cero.

En z/OS, esta opción no es compatible si el mensaje se coloca en una cola que tiene un tipo de índice MQIT_GROUP_ID.

La aplicación debe garantizar que estas banderas se establezcan correctamente al enviar mensajes. Si se especifica MQPMO_LOGICAL_ORDER, o se especificó en la llamada MQPUT anterior para el identificador de cola, la configuración de los indicadores debe ser coherente con la información de grupo y segmento conservada por el gestor de colas para el identificador de cola. Las siguientes condiciones se aplican a las llamadas MQPUT *sucesivas* para el identificador de cola cuando se especifica MQPMO_LOGICAL_ORDER:

- Si no hay ningún grupo actual ni mensaje lógico, todas estas banderas (y sus combinaciones) son válidas.
- Una vez especificado MQMF_MSG_IN_GROUP, debe permanecer activado hasta que se especifique MQMF_LAST_MSG_IN_GROUP. La llamada falla con el código de motivo MQRC_INCOMPLETE_GROUP si no se cumple esta condición.
- Una vez especificado MQMF_SEGMENT, debe permanecer activado hasta que se especifique MQMF_LAST_SEGMENT. La llamada falla con el código de motivo MQRC_INCOMPLETE_MSG si no se cumple esta condición.
- Una vez que se ha especificado MQMF_SEGMENT sin MQMF_MSG_IN_GROUP, MQMF_MSG_IN_GROUP debe permanecer *desactivado* hasta que se haya especificado MQMF_LAST_SEGMENT. La llamada falla con el código de motivo MQRC_INCOMPLETE_MSG si no se cumple esta condición.

El orden físico en una cola muestra las combinaciones válidas de los indicadores y los valores utilizados para diversos campos.

Estas banderas son banderas de entrada en las llamadas MQPUT y MQPUT1, y banderas de salida en la llamada MQGET. En la última llamada, el gestor de colas también repite los valores de los indicadores en los campos *GroupStatus* *SegmentStatus* y de MQGMO.

No puede utilizar mensajes agrupados o segmentados con la publicación/suscripción.

Indicadores predeterminados

Se puede especificar lo siguiente para indicar que el mensaje tiene atributos predeterminados:

MQMF_NINGUNO

Sin indicadores de mensaje (atributos de mensaje predeterminados).

Esto inhibe la segmentación e indica que el mensaje no pertenece a ningún grupo y no es un segmento de un mensaje lógico. MQMF_NONE se define para facilitar la documentación del programa. No está previsto que esta bandera se utilice con ninguna otra, pero como su valor es cero, dicho uso no puede detectarse.

El *MsgFlags* campo se divide en subcampos; para más detalles, véase [“Opciones de informe y distintivos de mensaje” en la página 911](#).

El valor inicial de este campo es MQMF_NONE. Este campo se ignora si *Version* es menor que MQMD_VERSION_2.

OriginalLength (MQLONG)

Este campo solo es relevante para los mensajes de informe que son segmentos. Especifica la longitud del segmento del mensaje al que se refiere el mensaje de informe; no especifica la longitud del mensaje lógico del que forma parte el segmento, ni la longitud de los datos del mensaje de informe.

Nota: Al generar un mensaje de informe para un mensaje que es un segmento, el gestor de colas y el agente del canal de mensajes copian en el MQMD para el mensaje de informe los campos *GroupId*, *MsgSeqNumber*, *Offset* y *MsgFlags* del mensaje original. Como resultado, el mensaje del informe también es un segmento. Las aplicaciones que generan mensajes de informe deben hacer lo mismo y establecer el *OriginalLength* campo correctamente.

Se define el siguiente valor especial:

MQOL_INDETERMINADO

Longitud original del mensaje no definida.

OriginalLength es un campo de entrada en las llamadas MQPUT y MQPUT1, pero el valor que proporciona la aplicación solo se acepta en circunstancias particulares:

- Si el mensaje que se envía es un segmento y también es un mensaje de informe, el gestor de colas acepta el valor especificado. El valor debe ser:
 - Mayor que cero si el segmento no es el último segmento
 - No menos que cero si el segmento es el último segmento
 - No menos que la longitud de los datos presentes en el mensaje

Si no se cumplen estas condiciones, la llamada falla con el código de motivo MQRC_ORIGINAL_LENGTH_ERROR.

- Si el mensaje que se envía es un segmento pero no un mensaje de informe, el gestor de colas ignora el campo y utiliza en su lugar la longitud de los datos del mensaje de la aplicación.
- En todos los demás casos, el gestor de colas ignora el campo y utiliza en su lugar el valor MQOL_UNDEFINED.

Este es un campo de salida en la llamada MQGET.

El valor inicial de este campo es MQOL_UNDEFINED. Este campo se ignora si *Version* es menor que MQMD_VERSION_2.

MQMDE - Extensión del descriptor de mensajes

La estructura MQMDE describe los datos que a veces aparecen antes de los datos del mensaje de la aplicación. La estructura contiene los campos MQMD que existen en el MQMD version-2, pero no en el MQMD version-1.

Disponibilidad

Todos IBM MQ los sistemas, además de IBM MQ MQI clients los conectados a estos sistemas.

Nombre de formato

MQFMT_MD_EXTENSION

Conjunto de caracteres y codificación

Los datos en MQMDE deben estar en el juego de caracteres y la codificación del gestor de colas local; estos se proporcionan mediante el atributo del **CodedCharSetId** gestor de colas y MQENC_NATIVE para el lenguaje de programación C.

Configure el juego de caracteres y la codificación del MQMDE en los campos *CodedCharSetId* *Encoding* y en:

- El MQMD (si la estructura MQMDE se encuentra al inicio de los datos del mensaje), o
- La estructura de encabezado que precede a la estructura MQMDE (todos los demás casos).

Si el MQMDE no está en el juego de caracteres y la codificación del gestor de colas, el MQMDE se acepta pero no se respeta, es decir, el MQMDE se trata como datos de mensaje.

Nota: En Windows, las aplicaciones compiladas con Micro Focus COBOL utilizan un valor de MQENC_NATIVE que es diferente de la codificación del gestor de colas. Aunque los campos numéricos de la estructura MQMD en las llamadas MQPUT, MQPUT1 y MQGET deben estar en la codificación Micro Focus COBOL, los campos numéricos de la estructura MQMDE deben estar en la codificación del gestor de colas. Este último es proporcionado por MQENC_NATIVE para el lenguaje de programación C y tiene el valor 546.

Utilización

Las aplicaciones que utilizan un MQMD de tipo « version-2 » no encontrarán una estructura MQMDE. Sin embargo, las aplicaciones especializadas y las aplicaciones que siguen utilizando un MQMD version-1 pueden encontrarse con un MQMDE en algunas situaciones. La estructura MQMDE puede darse en las siguientes circunstancias:

- Especificado en las llamadas MQPUT y MQPUT1
- Devuelto por la llamada MQGET
- En mensajes en colas de transmisión

MQMDE especificado en llamadas MQPUT y MQPUT1

En las llamadas MQPUT y MQPUT1, si la aplicación proporciona un MQMD version-1, la aplicación puede, opcionalmente, anteponer a los datos del mensaje un MQMDE, estableciendo el *Format* campo en MQMD en MQFMT_MD_EXTENSION para indicar que hay un MQMDE presente. Si la aplicación no proporciona un MQMDE, el gestor de colas asume los valores predeterminados para los campos del MQMDE. Los valores predeterminados que utiliza el gestor de colas son los mismos que los valores iniciales de la estructura; consulte [Tabla 521 en la página 446](#).

Si la aplicación proporciona un MQMD (version-2) y antepone un MQMDE a los datos del mensaje de la aplicación, las estructuras se procesan como se muestra en la siguiente tabla.

Tabla 520. Acción del gestor de colas cuando se especifica MQMDE en MQPUT o MQPUT1 para MQMDE			
MQMD Version	Valores de los campos « version-2 » (Estado de la solicitud de pago)	Valores de los campos correspondientes en MQMDE	Medidas adoptadas por el gestor de colas
1	-	Válido	MQMDE tiene el honor de
2	Valor predeterminado	Válido	MQMDE tiene el honor de
2	No predeterminado	Válido	MQMDE se trata como datos de mensaje
1 o 2	Cualquiera	No válido	La llamada falla con un código de motivo adecuado

Tabla 520. Acción del gestor de colas cuando se especifica MQMDE en MQPUT o MQPUT1 para MQMDE (continuación)

MQMD Version	Valores de los campos « version-2 » (Estado de la solicitud de pago)	Valores de los campos correspondientes en MQMDE	Medidas adoptadas por el gestor de colas
1 o 2	Cualquiera	MQMDE tiene un juego de caracteres o una codificación incorrectos, o es una versión no compatible	MQMDE se trata como datos de mensaje
Nota: Si z/OS la aplicación especifica un MQMD version-1 con un MQMDE, el gestor de colas valida el MQMDE solo si la cola tiene un <i>IndexType</i> de MQIT_GROUP_ID.			

Hay un caso especial. Si la aplicación utiliza un MQMD de tipo « version-2 » (es decir, el indicador MQMF_SEGMENT o MQMF_LAST_SEGMENT está activado) para insertar un mensaje que es un segmento, y el nombre del formato en el MQMD es MQFMT_DEAD_LETTER_HEADER, el gestor de colas genera una estructura MQMDE y la inserta *entre* la estructura MQDLH y los datos que le siguen. En el MQMD que el gestor de colas conserva con el mensaje, los campos « version-2 » se establecen en sus valores predeterminados.

Varios de los campos que existen en el MQMD version-2 pero no en el MQMD version-1 son campos de entrada/salida en MQPUT y MQPUT1. Sin embargo, el gestor de colas no devuelve ningún valor en los campos equivalentes del MQMDE en la salida de las llamadas MQPUT y MQPUT1; si la aplicación requiere esos valores de salida, debe utilizar un MQMD version-2.

MQMDE devuelto por la llamada MQGET

En la llamada MQGET, si la aplicación proporciona un MQMD version-1, el gestor de colas antepone al mensaje devuelto un MQMDE, pero solo si uno o varios de los campos del MQMDE tienen un valor distinto del predeterminado. El gestor de colas establece el *Format* campo en MQMD con el valor MQFMT_MD_EXTENSION para indicar que hay un MQMDE presente.

Si la aplicación proporciona un MQMDE al inicio del **Buffer** parámetro, el MQMDE se ignora. Al regresar de la llamada MQGET, se sustituye por MQMDE para el mensaje (si es necesario) o se sobrescribe con los datos del mensaje de la aplicación (si no es necesario MQMDE).

Si la llamada MQGET devuelve un MQMDE, los datos del MQMDE suelen estar en el juego de caracteres y la codificación del gestor de colas. Sin embargo, el MQMDE podría estar en otro juego de caracteres y codificación si:

- El MQMDE se trató como datos sobre la llamada MQPUT o MQPUT1 (consulte [Tabla 520 en la página 444](#) para conocer las circunstancias que pueden provocar esto).
- El mensaje se recibió desde un gestor de colas remoto conectado mediante una conexión de tipo « TCP », y el agente del canal de mensajes receptor (MCA) no estaba configurado correctamente.

Nota: En Windows, las aplicaciones compiladas con Micro Focus COBOL utilizan un valor de MQENC_NATIVE que es diferente de la codificación del gestor de colas (véase más arriba).

MQMDE en mensajes en colas de transmisión

Los mensajes en las colas de transmisión tienen como prefijo la estructura MQXQH, que contiene en su interior un MQMD (version-1). También puede estar presente un MQMDE, situado entre la estructura MQXQH y los datos del mensaje de aplicación, pero normalmente solo está presente si uno o varios de los campos del MQMDE tienen un valor distinto al predeterminado.

También pueden aparecer otras estructuras de encabezado MQ entre la estructura MQXQH y los datos del mensaje de la aplicación. Por ejemplo, cuando está presente el encabezado de carta muerta MQDLH y el mensaje no es un segmento, el orden es:

- MQXQH (que contiene un MQMD e version-1)
- MQMDE
- MQDLH
- datos del mensaje de la aplicación

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 521. Campos en MQMDE		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>StrucId</u> (identificador de estructura)	MQMDE_STRUC_ID	'MDE↵'
<u>Versión</u> (número de versión de la estructura)	MQMDE_VERSION_2	2
<u>StrucLength</u> (longitud de la estructura MQMDE)	MQMDE_LENGTH_2	72
<u>Codificación</u> (codificación numérica de datos que sigue a MQMDE)	MQENC_NATIVE	Depende del entorno
<u>CodedCharSetId</u> (identificador del conjunto de caracteres de los datos que siguen a MQMDE)	MQCCSI_INDETERMINADO	0
<u>Formato</u> (nombre del formato de los datos que siguen a MQMDE)	MQFMT_NONE	Espacios en blanco
<u>Banderas</u> (banderas generales)	MQMDEF_NINGUNO	0
<u>GroupId</u> (identificador de grupo)	MQGI_NONE	Nulos
<u>MsgSeqNumber</u> (número de secuencia del mensaje lógico dentro del grupo)	Ninguna	1
<u>Desplazamiento</u> (desplazamiento de los datos en el mensaje físico desde el inicio del mensaje lógico)	Ninguna	0
<u>MsgFlags</u> (indicadores de mensajes)	MQMF_NINGUNO	0
<u>OriginalLength</u> (longitud del mensaje original)	MQOL_INDETERMINADO	-1
Notas: 1. El símbolo ↵ representa un único carácter en blanco. 2. En el lenguaje de programación C, la variable macroMQMDE_DEFAULT contiene los valores que se enumeran en la tabla. Se puede utilizar de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQMDE MyMDE = {MQMDE_DEFAULT};</pre>		

Declaraciones lingüísticas

Declaración C para MQMDE

```
typedef struct tagMQMDE MQMDE;
struct tagMQMDE {
    MQCHAR4    StrucId;          /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQLONG     StrucLength;      /* Length of MQMDE structure */
    MQLONG     Encoding;         /* Numeric encoding of data that follows
                                MQMDE */
    MQLONG     CodedCharSetId;   /* Character-set identifier of data that
                                follows MQMDE */
    MQCHAR8     Format;          /* Format name of data that follows
                                MQMDE */
    MQLONG     Flags;           /* General flags */
    MQBYTE24    GroupId;         /* Group identifier */
    MQLONG     MsgSeqNumber;     /* Sequence number of logical message
                                within group */
    MQLONG     Offset;          /* Offset of data in physical message from
                                start of logical message */
    MQLONG     MsgFlags;         /* Message flags */
    MQLONG     OriginalLength;   /* Length of original message */
};
```

COBOL declaración para MQMDE

```
** MQMDE structure
10 MQMDE.
** Structure identifier
15 MQMDE-STRUCID PIC X(4).
** Structure version number
15 MQMDE-VERSION PIC S9(9) BINARY.
** Length of MQMDE structure
15 MQMDE-STRUCLength PIC S9(9) BINARY.
** Numeric encoding of data that follows MQMDE
15 MQMDE-ENCODING PIC S9(9) BINARY.
** Character-set identifier of data that follows MQMDE
15 MQMDE-CODEDCHARSETID PIC S9(9) BINARY.
** Format name of data that follows MQMDE
15 MQMDE-FORMAT PIC X(8).
** General flags
15 MQMDE-FLAGS PIC S9(9) BINARY.
** Group identifier
15 MQMDE-GROUPID PIC X(24).
** Sequence number of logical message within group
15 MQMDE-MSGSEQNUMBER PIC S9(9) BINARY.
** Offset of data in physical message from start of logical message
15 MQMDE-OFFSET PIC S9(9) BINARY.
** Message flags
15 MQMDE-MSGFLAGS PIC S9(9) BINARY.
** Length of original message
15 MQMDE-ORIGINALLENGTH PIC S9(9) BINARY.
```

PL/I declaración para MQMDE

```
dcl
1 MQMDE based,
3 StrucId char(4), /* Structure identifier */
3 Version fixed bin(31), /* Structure version number */
3 StrucLength fixed bin(31), /* Length of MQMDE structure */
3 Encoding fixed bin(31), /* Numeric encoding of data that
                           follows MQMDE */
3 CodedCharSetId fixed bin(31), /* Character-set identifier of data
                           that follows MQMDE */
3 Format char(8), /* Format name of data that follows
                  MQMDE */
3 Flags fixed bin(31), /* General flags */
3 GroupId char(24), /* Group identifier */
3 MsgSeqNumber fixed bin(31), /* Sequence number of logical message
                           within group */
3 Offset fixed bin(31), /* Offset of data in physical message
                           from start of logical message */
3 MsgFlags fixed bin(31), /* Message flags */
3 OriginalLength fixed bin(31); /* Length of original message */
```

High Level Assembler declaración para MQMDE

```

MQMDE          DSECT
MQMDE_STRUCID  DS   CL4   Structure identifier
MQMDE_VERSION  DS   F     Structure version number
MQMDE_STRUCLength DS   F     Length of MQMDE structure
MQMDE_ENCODING DS   F     Numeric encoding of data that follows
*              MQMDE
MQMDE_CODEDCHARSETID DS   F     Character-set identifier of data that
*              follows MQMDE
MQMDE_FORMAT   DS   CL8   Format name of data that follows MQMDE
MQMDE_FLAGS    DS   F     General flags
MQMDE_GROUPID  DS   XL24  Group identifier
MQMDE_MSGSEQNUMBER DS   F     Sequence number of logical message
*              within group
MQMDE_OFFSET   DS   F     Offset of data in physical message from
*              start of logical message
MQMDE_MSGFLAGS DS   F     Message flags
MQMDE_ORIGINALLENGTH DS   F     Length of original message
*
MQMDE_LENGTH   EQU   *-MQMDE
ORG   MQMDE
MQMDE_AREA     DS   CL(MQMDE_LENGTH)

```

Declaración de Visual Basic para MQMDE

```

Type MQMDE
  StrucId      As String*4 'Structure identifier'
  Version      As Long     'Structure version number'
  StrucLength  As Long     'Length of MQMDE structure'
  Encoding     As Long     'Numeric encoding of data that follows'
                  'MQMDE'
  CodedCharSetId As Long   'Character-set identifier of data that'
                  'follows MQMDE'
  Format       As String*8 'Format name of data that follows MQMDE'
  Flags       As Long     'General flags'
  GroupId     As MQBYTE24 'Group identifier'
  MsgSeqNumber As Long     'Sequence number of logical message within'
                  'group'
  Offset      As Long     'Offset of data in physical message from'
                  'start of logical message'
  MsgFlags    As Long     'Message flags'
  OriginalLength As Long   'Length of original message'
End Type

```

Detalles de campo para MQMDE

La estructura de extensión del descriptor de mensajes MQMDE describe los datos que a veces aparecen antes de los datos del mensaje de la aplicación. La estructura contiene los campos MQMD que existen en el MQMD version-2, pero no en el MQMD version-1.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 522. Campos en MQMDE		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>StrucId</u> (identificador de estructura)	MQMDE_STRUC_ID	'MDE~'
<u>Versión</u> (número de versión de la estructura)	MQMDE_VERSION_2	2
<u>StrucLength</u> (longitud de la estructura MQMDE)	MQMDE_LENGTH_2	72
<u>Codificación</u> (codificación numérica de datos que sigue a MQMDE)	MQENC_NATIVE	Depende del entorno
<u>CodedCharSetId</u> (identificador del conjunto de caracteres de los datos que siguen a MQMDE)	MQCCSI_INDETERMINADO	0

Tabla 522. Campos en MQMDE (continuación)

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>Formato</u> (nombre del formato de los datos que siguen a MQMDE)	MQFMT_NONE	Espacios en blanco
<u>Banderas</u> (banderas generales)	MQMDEF_NINGUNO	0
<u>GroupId</u> (identificador de grupo)	MQGI_NONE	Nulos
<u>MsgSeqNumber</u> (número de secuencia del mensaje lógico dentro del grupo)	Ninguna	1
<u>Desplazamiento</u> (desplazamiento de los datos en el mensaje físico desde el inicio del mensaje lógico)	Ninguna	0
<u>MsgFlags</u> (indicadores de mensajes)	MQMF_NINGUNO	0
<u>OriginalLength</u> (longitud del mensaje original)	MQOL_INDETERMINADO	-1
Notas: 1. El símbolo ~ representa un único carácter en blanco. 2. En el lenguaje de programación C, la variable macroMQMDE_DEFAULT contiene los valores que se enumeran en la tabla. Se puede utilizar de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQMDE MyMDE = {MQMDE_DEFAULT};</pre>		

StrucId (MQCHAR4)

Este es el identificador de estructura de la estructura de extensión del descriptor de mensaje. Siempre es un campo de entrada. Su valor es MQMDE_STRUC_ID.

El valor debe ser:

MQMDE_STRUC_ID

Identificador de la estructura de extensión del descriptor de mensaje.

Para el lenguaje de programación C, también se define la constante MQMDE_STRUC_ID_ARRAY. Tiene el mismo valor que MQMDE_STRUC_ID, pero es una matriz de caracteres en lugar de una cadena.

Versión (MQLONG)

Este es el número de versión de la estructura; el valor debe ser:

MQMDE_VERSION_2

Version-2 Estructura de extensión del descriptor de mensajes.

La siguiente constante especifica el número de versión de la versión actual:

MQMDE_VERSIÓN_ACTUAL

Versión actual de la estructura de extensión del descriptor de mensajes.

El valor inicial de este campo es MQMDE_VERSION_2.

StrucLength (MQLONG)

Esta es la longitud de la estructura MQMDE; se define el siguiente valor:

MQMDE_LENGTH_2

Longitud de la estructura de extensión del descriptor de mensajes « version-2 ».

El valor inicial de este campo es MQMDE_LENGTH_2.

Encoding (MQLONG)

Esto especifica la codificación numérica de los datos que siguen a la estructura MQMDE; no se aplica a los datos numéricos de la propia estructura MQMDE.

En la llamada MQPUT o MQPUT1, la aplicación debe establecer este campo con el valor adecuado para los datos. El gestor de colas no comprueba que el campo sea válido. Consulte el campo [Codificación](#) descrito en “MQMD - Descriptor de mensaje” en la página 389 para obtener más información sobre las codificaciones de datos.

El valor inicial de este campo es MQENC_NATIVE.

CodedCharSetId (MQLONG)

Especifica el identificador del conjunto de caracteres de los datos que siguen a la estructura MQMDE; no se aplica a los datos de caracteres de la propia estructura MQMDE.

En la llamada MQPUT o MQPUT1, la aplicación debe establecer este campo con el valor adecuado para los datos. El gestor de colas no comprueba que este campo sea válido. Se puede utilizar el siguiente valor especial:

MQCCSI_HEREDAR

Los datos de caracteres en los datos *que siguen* esta estructura están en el mismo conjunto de caracteres que esta estructura.

El gestor de colas cambia este valor en la estructura enviada en el mensaje al identificador real del juego de caracteres de la estructura. Si no se produce ningún error, la llamada MQGET no devuelve el valor MQCCSI_INHERIT.

MQCCSI_INHERIT no se puede utilizar si el valor del *PutApplType* campo en MQMD es MQAT_BROKER.

Este valor es compatible con los siguientes entornos:

-  AIX
-  IBM i
-  Linux
-  Windows

y para IBM MQ los clientes conectados a estos sistemas.

El valor inicial de este campo es MQCCSI_UNDEFINED.

Format (MQCHAR8)

Especifica el nombre del formato de los datos que siguen a la estructura MQMDE.

En la llamada MQPUT o MQPUT1, la aplicación debe establecer este campo con el valor adecuado para los datos. El gestor de colas no comprueba que este campo sea válido. Consulte el campo [Formato](#) descrito en “MQMD - Descriptor de mensaje” en la página 389 para obtener más información sobre los nombres de formato.

El valor inicial de este campo es MQFMT_NONE.

Flags (MQLONG)

Se puede especificar la siguiente bandera:

MQMDEF_NINGUNO

Sin banderas.

El valor inicial de este campo es MQMDEF_NONE.

GroupId (MQBYTE24)

Véase el GroupId campo descrito en “MQMD - Descriptor de mensaje” en la página 389. El valor inicial de este campo es MQGI_NONE.

MsgSeqNumber (MQLONG)

Véase el MsgSeqNumber campo descrito en “MQMD - Descriptor de mensaje” en la página 389. El valor inicial de este campo es 1.

Offset (MQLONG)

Consulte el campo Desplazamiento descrito en “MQMD - Descriptor de mensaje” en la página 389. El valor inicial de este campo es 0.

MsgFlags (MQLONG)

Véase el MsgFlags campo descrito en “MQMD - Descriptor de mensaje” en la página 389. El valor inicial de este campo es MQMF_NONE.

OriginalLength (MQLONG)

Véase el OriginalLength campo descrito en “MQMD - Descriptor de mensaje” en la página 389. El valor inicial de este campo es MQOL_UNDEFINED.

MQMHBO - Opciones del controlador de mensajes al búfer

La estructura MQMHBO permite a las aplicaciones especificar opciones que controlan cómo se producen los búferes a partir de los identificadores de mensajes. La estructura es un parámetro de entrada en la llamada MQMHBUF.

Conjunto de caracteres y codificación

Los datos en MQMHBO deben estar en el juego de caracteres de la aplicación y en la codificación de la aplicación (MQENC_NATIVE).

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 523. Campos en MQMHBO		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>StrucId</u> (identificador de estructura)	MQMHBO_STRUC_ID	'MHBO'
<u>Versión</u> (número de versión de la estructura)	MQMHBO_VERSION_1	1
<u>Opciones</u> (opciones que controlan la acción de MQMHBUF)	MQMHBO_PROPERTIES_I N_MQRFH2	

Tabla 523. Campos en MQMHBO (continuación)

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
Notas: 1. El valor «Cadena nula o espacios en blanco» denota la cadena nula en C y los caracteres en blanco en otros lenguajes de programación. 2. En el lenguaje de programación C, la variable macroMQMHBO_DEFAULT contiene los valores que se enumeran en la tabla. Úselo de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQMHBO MyMHBO = {MQMHBO_DEFAULT};</pre>		

Declaraciones lingüísticas

Declaración C para MQMHBO

```
typedef struct tagMQMHBO MQMHBO;
struct tagMQMHBO {
    MQCHAR4 StrucId;          /* Structure identifier */
    MQLONG  Version;         /* Structure version number */
    MQLONG  Options;         /* Options that control the action of
                             MQMHBUFF */
};
```

COBOL declaración para MQMHBO

```
** MQMHBO structure
10 MQMHBO.
**   Structure identifier
15 MQMHBO-STRUCID          PIC X(4).
**   Structure version number
15 MQMHBO-VERSION          PIC S9(9) BINARY.
**   Options that control the action of MQMHBUFF
15 MQMHBO-OPTIONS          PIC S9(9) BINARY.
```

PL/I declaración para MQMHBO

```
Dcl
1 MQMHBO based,
3 StrucId      char(4),          /* Structure identifier */
3 Version      fixed bin(31), /* Structure version number */
3 Options      fixed bin(31), /* Options that control the action
                             of MQMHBUFF */
```

High Level Assembler declaración para MQMHBO

```
MQMHBO          DSECT
MQMHBO_STRUCID  DS   CL4  Structure identifier
MQMHBO_VERSION  DS   F    Structure version number
MQMHBO_OPTIONS  DS   F    Options that control the
*                    action of MQMHBUFF
MQMHBO_LENGTH   EQU   *-MQMHBO
MQMHBO_AREA     DS    CL(MQMHBO_LENGTH)
```

Detalles de campo para MQMHBO

La estructura MQMHBO Message handle to buffer options permite a las aplicaciones especificar opciones que controlan cómo se producen los búferes a partir de los identificadores de mensajes. La estructura es un parámetro de entrada en la llamada MQMHBUFF.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 524. Campos en MQMHBO

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
StrucId (identificador de estructura)	MQMHBO_STRUC_ID	'MHBO'
Versión (número de versión de la estructura)	MQMHBO_VERSION_1	1
Opciones (opciones que controlan la acción de MQMHBUF)	MQMHBO_PROPERTIES_IN_MQRFH2	

Notas:

1. El valor «Cadena nula o espacios en blanco» denota la cadena nula en C y los caracteres en blanco en otros lenguajes de programación.
2. En el lenguaje de programación C, la variable macro MQMHBO_DEFAULT contiene los valores que se enumeran en la tabla. Úselo de la siguiente manera para proporcionar valores iniciales para los campos de la estructura:

```
MQMHBO MyMHBO = {MQMHBO_DEFAULT};
```

StrucId (MQCHAR4)

Este es el identificador de estructura del identificador de mensaje a la estructura de opciones del búfer. Siempre es un campo de entrada. Su valor es MQMHBO_STRUC_ID.

El valor debe ser:

MQMHBO_STRUC_ID

Identificador para el controlador de mensajes a la estructura de opciones del búfer.

Para el lenguaje de programación C, también se define la constante MQMHBO_STRUC_ID_ARRAY. Tiene el mismo valor que MQMHBO_STRUC_ID, pero es una matriz de caracteres en lugar de una cadena.

Versión (MQLONG)

Estructura de opciones del controlador de mensajes al búfer: campo Versión. Este es el número de versión de la estructura. El valor debe ser:

MQMHBO_VERSION_1

Número de versión para el identificador de mensaje a la estructura de opciones del búfer.

La siguiente constante especifica el número de versión de la versión actual:

MQMHBO_VERSIÓN_ACTUAL

Versión actual del identificador de mensaje para la estructura de opciones del búfer.

Este campo siempre es un campo de entrada. El valor inicial de este campo es MQMHBO_VERSION_1.

Opciones (MQLONG)

Manejador de mensajes a estructura de opciones de búfer - Campo de opciones. Estas opciones controlan la acción de MQMHBUF.

Debe especificar la siguiente opción:

MQMHBO_PROPERTIES_IN_MQRFH2

Al convertir propiedades de un identificador de mensaje a un búfer, conviértalas al formato MQRFH2.

Opcionalmente, también puede especificar la siguiente opción. Para especificar más de una opción, suma los valores (sin añadir la misma constante más de una vez) o combínalos mediante la operación OR bit a bit (si el lenguaje de programación admite operaciones bit a bit).

MQMHBO_ELIMINAR_PROPIEDADES

Las propiedades que se añaden al búfer se eliminan del identificador del mensaje. Si la llamada falla, no se eliminan ninguna propiedad.

Este campo siempre es un campo de entrada. El valor inicial de este campo es MQMHBO_PROPERTIES_IN_MQRFH2.

MQOD - Descriptor de objeto

La estructura MQOD se utiliza para especificar un objeto por su nombre. La estructura es un parámetro de entrada/salida en las llamadas MQOPEN y MQPUT1.

Los siguientes tipos de objetos son válidos:

- Cola o lista de distribución
- Lista de nombres
- Definición de proceso
- Gestor de colas
- Tema

Disponibilidad

Todos IBM MQ los sistemas, además de IBM MQ MQI clients los conectados a esos sistemas.

Versión

La versión actual de MQOD es MQOD_VERSION_4. Las aplicaciones que desee transferir entre varios entornos deben garantizar que la versión requerida de MQOD sea compatible con todos los entornos en cuestión. Los campos que solo existen en las versiones más recientes de la estructura se identifican como tales en las descripciones que siguen.

Los archivos de encabezado, COPY e INCLUDE proporcionados para los lenguajes de programación compatibles contienen la versión más reciente de MQOD compatible con el entorno, pero con el valor inicial del *Version* campo establecido en MQOD_VERSION_1. Para utilizar campos que no están presentes en la estructura version-1, la aplicación debe establecer el *Version* campo en el número de versión requerido.

Para abrir una lista de distribución, *Version* debe ser MQOD_VERSION_2 o superior.

Conjunto de caracteres y codificación

Los datos en MQOD deben estar en el juego de caracteres proporcionado por el atributo del **CodedCharSetId** gestor de colas y la codificación del gestor de colas local proporcionada por MQENC_NATIVE. Sin embargo, si la aplicación se ejecuta como un cliente MQ MQI, la estructura debe estar en el conjunto de caracteres y la codificación del cliente.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>StrucId</u> (identificador de estructura)	MQOD_STRUC_ID	'0D-11'
<u>Versión</u> (número de versión de la estructura)	MQOD_VERSION_1	1
<u>ObjectType</u> (tipo de objeto)	MQOT_Q	1
<u>ObjectName</u> (nombre del objeto)	Ninguna	Cadena nula o espacios en blanco

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>ObjectQMgrName</u> (nombre del gestor de colas de objetos)	Ninguna	Cadena nula o espacios en blanco
<u>DynamicQName</u> (nombre de la cola dinámica)	Ninguna	'CSQ.*' en z/OS ; 'AMQ.*' de lo contrario
<u>AlternateUserId</u> (identificador de usuario alternativo)	Ninguna	Cadena nula o espacios en blanco
Nota: Los campos restantes se ignoran si <i>Version</i> es menor que MQOD_VERSION_2.		
<u>RecsPresent</u> (número de registros de objetos presentes)	Ninguna	0
<u>KnownDestCount</u> (número de colas locales abiertas correctamente)	Ninguna	0
<u>UnknownDestCount</u> (número de colas remotas abiertas correctamente)	Ninguna	0
<u>InvalidDestCount</u> (número de colas que no se pudieron abrir)	Ninguna	0
<u>ObjectRecOffset</u> (desplazamiento del primer registro de objeto desde el inicio de MQOD)	Ninguna	0
<u>ResponseRecOffset</u> (desplazamiento del primer registro de respuesta desde el inicio de MQOD)	Ninguna	0
<u>ObjectRecPtr</u> (dirección del primer registro de objeto)	Ninguna	Puntero nulo o bytes nulos
<u>ResponseRecPtr</u> (dirección del primer registro de respuesta)	Ninguna	Puntero nulo o bytes nulos
Nota: Los campos restantes se ignoran si <i>Version</i> es menor que MQOD_VERSION_3.		
<u>AlternateSecurityId</u> (identificador de seguridad alternativo)	MQSID_NINGUNO	Nulos
<u>ResolvedQName</u> (nombre de la cola resuelta)	Ninguna	Cadena nula o espacios en blanco
<u>ResolvedQMgrName</u> (nombre del gestor de colas resuelto)	Ninguna	Cadena nula o espacios en blanco
Nota: Los campos restantes se ignoran si <i>Version</i> es menor que MQOD_VERSION_4.		
<u>ObjectString</u> (nombre largo del objeto)	MQCHARV_DEFAULT	Según se define para MQCHARV
<u>SelectionString</u> (cadena de selección)	MQCHARV_DEFAULT	Según se define para MQCHARV
<u>ResObjectString</u> (nombre de objeto largo resuelto)	MQCHARV_DEFAULT	Según se define para MQCHARV
<u>ResolvedType</u> (tipo de objeto resuelto)	MQOT_NINGUNO	0

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
Notas: 1. El símbolo ~ representa un único carácter en blanco. 2. El valor «Cadena nula o espacios en blanco» denota la cadena nula en C y los caracteres en blanco en otros lenguajes de programación. 3. En el lenguaje de programación C, la variable macroMQOD_DEFAULT contiene los valores que se enumeran en la tabla. Se puede utilizar de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQOD MyOD = {MQOD_DEFAULT};</pre>		

Declaraciones lingüísticas

Declaración C para MQOD

```
typedef struct tagMQOD MQOD;
struct tagMQOD {
    MQCHAR4      StructId;          /* Structure identifier */
    MQLONG       Version;           /* Structure version number */
    MQLONG       ObjectType;        /* Object type */
    MQCHAR48     ObjectName;        /* Object name */
    MQCHAR48     ObjectQMgrName;    /* Object queue manager name */
    MQCHAR48     DynamicQName;      /* Dynamic queue name */
    MQCHAR12     AlternateUserId;    /* Alternate user identifier */
    /* Ver:1 */
    MQLONG       RecsPresent;        /* Number of object records present */
    MQLONG       KnownDestCount;     /* Number of local queues opened
                                     successfully */
    MQLONG       UnknownDestCount;   /* Number of remote queues opened
                                     successfully */
    MQLONG       InvalidDestCount;   /* Number of queues that failed to
                                     open */
    MQLONG       ObjectRecOffset;    /* Offset of first object record from
                                     start of MQOD */
    MQLONG       ResponseRecOffset;  /* Offset of first response record
                                     from start of MQOD */
    MQPTR        ObjectRecPtr;       /* Address of first object record */
    MQPTR        ResponseRecPtr;     /* Address of first response record */
    /* Ver:2 */
    MQBYTE40     AlternateSecurityId; /* Alternate security identifier */
    MQCHAR48     ResolvedQName;      /* Resolved queue name */
    MQCHAR48     ResolvedQMgrName;   /* Resolved queue manager name */
    /* Ver:3 */
    MQCHARV      ObjectString;       /* Object Long name */
    MQCHARV      SelectionString;     /* Message Selector */
    MQCHARV      ResObjectString;    /* Resolved Long object name*/
    MQLONG       ResolvedType        /* Alias queue resolved
                                     object type */
    /* Ver:4 */
};
```

COBOL declaración para MQOD

```
** MQOD structure
10  MQOD.
** Structure identifier
15  MQOD-STRUCID                PIC X(4).
** Structure version number
15  MQOD-VERSION                PIC S9(9) BINARY.
** Object type
15  MQOD-OBJECTTYPE            PIC S9(9) BINARY.
** Object name
15  MQOD-OBJECTNAME            PIC X(48).
** Object queue manager name
15  MQOD-OBJECTQMGRNAME        PIC X(48).
** Dynamic queue name
15  MQOD-DYNAMICQNAME          PIC X(48).
** Alternate user identifier
15  MQOD-ALTERNATEUSERID        PIC X(12).
```

```

** Number of object records present
15 MQOD-RECSPRESENT PIC S9(9) BINARY.
** Number of local queues opened successfully
15 MQOD-KNOWNDESTCOUNT PIC S9(9) BINARY.
** Number of remote queues opened successfully
15 MQOD-UNKNOWNDESTCOUNT PIC S9(9) BINARY.
** Number of queues that failed to open
15 MQOD-INVALIDDESTCOUNT PIC S9(9) BINARY.
** Offset of first object record from start of MQOD
15 MQOD-OBJECTRECOFFSET PIC S9(9) BINARY.
** Offset of first response record from start of MQOD
15 MQOD-RESPONSERECOFFSET PIC S9(9) BINARY.
** Address of first object record
15 MQOD-OBJECTRECPTR POINTER.
** Address of first response record
15 MQOD-RESPONSERECPTR POINTER.
** Alternate security identifier
15 MQOD-ALTERNATESECURITYID PIC X(40).
** Resolved queue name
15 MQOD-RESOLVEDQNAME PIC X(48).
** Resolved queue manager name
15 MQOD-RESOLVEDQMGRNAME PIC X(48).
** Object Long name
15 MQOD-OBJECTSTRING.
** Address of variable length string
20 MQOD-OBJECTSTRING-VSPTR POINTER.
** Offset of variable length string
20 MQOD-OBJECTSTRING-VSOFFSET PIC S9(9) BINARY.
** size of buffer
20 MQOD-OBJECTSTRING-VSBUFSIZE PIC S9(9) BINARY.
** Length of variable length string
20 MQOD-OBJECTSTRING-VSLENGTH PIC S9(9) BINARY.
** CCSID of variable length string
20 MQOD-OBJECTSTRING-VSCCSID PIC S9(9) BINARY.
** Message Selector
15 MQOD-SELECTIONSTRING.
** Address of variable length string
20 MQOD-SELECTIONSTRING-VSPTR POINTER.
** Offset of variable length string
20 MQOD-SELECTIONSTRING-VSOFFSET PIC S9(9) BINARY.
** size of buffer
20 MQOD-SELECTIONSTRING-VSBUFSIZE PIC S9(9) BINARY.
** Length of variable length string
20 MQOD-SELECTIONSTRING-VSLENGTH PIC S9(9) BINARY.
** CCSID of variable length string
20 MQOD-SELECTIONSTRING-VSCCSID PIC S9(9) BINARY.
** Resolved Long object name
15 MQOD-RESOBJECTSTRING.
** Address of variable length string
20 MQOD-RESOBJECTSTRING-VSPTR POINTER.
** Offset of variable length string
20 MQOD-RESOBJECTSTRING-VSOFFSET PIC S9(9) BINARY.
** size of buffer
20 MQOD-RESOBJECTSTRING-VSBUFSIZE PIC S9(9) BINARY.
** Length of variable length string
20 MQOD-RESOBJECTSTRING-VSLENGTH PIC S9(9) BINARY.
** CCSID of variable length string
20 MQOD-RESOBJECTSTRING-VSCCSID PIC S9(9) BINARY.
** Alias queue resolved object type
15 MQOD-RESOLVEDTYPE PIC S9(9) BINARY.

```

PL/I declaración para MQOD

```

dcl
1 MQOD based,
3 StrucId char(4), /* Structure identifier */
3 Version fixed bin(31), /* Structure version number */
3 ObjectType fixed bin(31), /* Object type */
3 ObjectName char(48), /* Object name */
3 ObjectQMGrName char(48), /* Object queue manager name */
3 DynamicQName char(48), /* Dynamic queue name */
3 AlternateUserId char(12), /* Alternate user identifier */
3 RecsPresent fixed bin(31), /* Number of object records
present */
3 KnownDestCount fixed bin(31), /* Number of local queues opened
successfully */
3 UnknownDestCount fixed bin(31), /* Number of remote queues opened
successfully */
3 InvalidDestCount fixed bin(31), /* Number of queues that failed to
open */

```

3 ObjectRecOffset	fixed bin(31),	/* Offset of first object record from start of MQOD */
3 ResponseRecOffset	fixed bin(31),	/* Offset of first response record from start of MQOD */
3 ObjectRecPtr	pointer,	/* Address of first object record */
3 ResponseRecPtr	pointer,	/* Address of first response record */
3 AlternateSecurityId	char(40),	/* Alternate security identifier */
3 ResolvedQName	char(48),	/* Resolved queue name */
3 ResolvedQMGrName	char(48),	/* Resolved queue manager name */
3 ObjectString,		/* Object Long name */
5 VSPtr	pointer,	/* Address of variable length string */
5 VSOffset	fixed bin(31),	/* Offset of variable length string */
5 VSBufSize	fixed bin(31),	/* size of buffer */
5 VSLength	fixed bin(31),	/* Length of variable length string */
5 VSCCSID	fixed bin(31),	/* CCSID of variable length string */
3 SelectionString,		/* Message Selection */
5 VSPtr	pointer,	/* Address of variable length string */
5 VSOffset	fixed bin(31),	/* Offset of variable length string */
5 VSBufSize	fixed bin(31),	/* size of buffer */
5 VSLength	fixed bin(31),	/* Length of variable length string */
5 VSCCSID	fixed bin(31),	/* CCSID of variable length string */
3 ResObjectString,		/* Resolved Long object name */
5 VSPtr	pointer,	/* Address of variable length string */
5 VSOffset	fixed bin(31),	/* Offset of variable length string */
5 VSBufSize	fixed bin(31),	/* size of buffer */
5 VSLength	fixed bin(31),	/* Length of variable length string */
5 VSCCSID	fixed bin(31),	/* CCSID of variable length string */
3 ResolvedType	fixed bin(31);	/* Alias queue resolved object type */

High Level Assembler declaración para MQOD

MQOD	DSECT	
MQOD_STRUCID	DS	CL4 Structure identifier
MQOD_VERSION	DS	F Structure version number
MQOD_OBJECTTYPE	DS	F Object type
MQOD_OBJECTNAME	DS	CL48 Object name
MQOD_OBJECTQMGRNAME	DS	CL48 Object queue manager name
MQOD_DYNAMICQNAME	DS	CL48 Dynamic queue name
MQOD_ALTERNATEUSERID	DS	CL12 Alternate user identifier
MQOD_RECSPRESENT	DS	F Number of object records present
MQOD_KNOWNDESTCOUNT	DS	F Number of local queues opened successfully
* MQOD_UNKNOWNDSTCOUNT	DS	F Number of remote queues opened successfully
* MQOD_INVALIDDESTCOUNT	DS	F Number of queues that failed to open
* MQOD_OBJECTRECOFFSET	DS	F Offset of first object record from start of MQOD
* MQOD_RESPONSERECOFFSET	DS	F Offset of first response record from start of MQOD
* MQOD_OBJECTRECPtr	DS	F Address of first object record
MQOD_RESPONSERECPtr	DS	F Address of first response record
MQOD_ALTERNATESECURITYID	DS	XL40 Alternate security identifier
MQOD_RESOLVEDQNAME	DS	CL48 Resolved queue name
MQOD_RESOLVEDQMGRNAME	DS	CL48 Resolved queue manager name
MQOD_OBJECTSTRING	DS	F Object Long name
MQOD_OBJECTSTRING_VSPTR	DS	F Address of variable length string
MQOD_OBJECTSTRING_VSOFFSET	DS	F Offset of variable length string
MQOD_OBJECTSTRING_VSBUFSIZE	DS	F size of buffer
MQOD_OBJECTSTRING_VSLLENGTH	DS	F Length of variable length string
MQOD_OBJECTSTRING_VSCCSID	DS	F CCSID of variable length string
MQOD_OBJECTSTRING_LENGTH	EQU	*- MQOD_OBJECTSTRING
	ORG	MQOD_OBJECTSTRING
MQOD_OBJECTSTRING_AREA	DS	CL(MQOD_OBJECTSTRING_LENGTH)
* MQOD_SELECTIONSTRING	DS	F Message Selector
MQOD_SELECTIONSTRING_VSPTR	DS	F Address of variable length string
MQOD_SELECTIONSTRING_VSOFFSET	DS	F Offset of variable length string
MQOD_SELECTIONSTRING_VSBUFSIZE	DS	F size of buffer
MQOD_SELECTIONSTRING_VSLLENGTH	DS	F Length of variable length string
MQOD_SELECTIONSTRING_VSCCSID	DS	F CCSID of variable length string
MQOD_SELECTIONSTRING_LENGTH	EQU	*- MQOD_SELECTIONSTRING
	ORG	MQOD_SELECTIONSTRING
MQOD_SELECTIONSTRING_AREA	DS	CL(MQOD_SELECTIONSTRING_LENGTH)
* MQOD_RESOBJECTSTRING	DS	F Resolved Long object name
MQOD_RESOBJECTSTRING_VSPTR	DS	F Address of variable length string
MQOD_RESOBJECTSTRING_VSOFFSET	DS	F Offset of variable length string
MQOD_RESOBJECTSTRING_VSBUFSIZE	DS	F size of buffer

```

MQOD_RESOBJECTSTRING_VSLENGTH DS F Length of variable length string
MQOD_RESOBJECTSTRING_VSCCSID DS F CCSID of variable length string
MQOD_RESOBJECTSTRING_LENGTH EQU *- MQOD_RESOBJECTSTRING
                                ORG MQOD_RESOBJECTSTRING
MQOD_RESOBJECTSTRING_AREA DS CL(MQOD_RESOBJECTSTRING_LENGTH)
MQOD_RESOLVEDTYPE DS F Alias queue object resolved type
*
MQOD_LENGTH EQU *-MQOD
                                ORG MQOD
MQOD_AREA DS CL(MQOD_LENGTH)

```

Declaración de Visual Basic para MQOD

```

Type MQOD
  StrucId      As String*4 'Structure identifier'
  Version      As Long     'Structure version number'
  ObjectType   As Long     'Object type'
  ObjectName   As String*48 'Object name'
  ObjectQMgrName As String*48 'Object queue manager name'
  DynamicQName As String*48 'Dynamic queue name'
  AlternateUserId As String*12 'Alternate user identifier'
  RecsPresent  As Long     'Number of object records present'
  KnownDestCount As Long   'Number of local queues opened'
                                'successfully'
  UnknownDestCount As Long 'Number of remote queues opened'
                                'successfully'
  InvalidDestCount As Long 'Number of queues that failed to'
                                'open'
  ObjectRecOffset As Long   'Offset of first object record from'
                                'start of MQOD'
  ResponseRecOffset As Long 'Offset of first response record'
                                'from start of MQOD'
  ObjectRecPtr  As MQPTR   'Address of first object record'
  ResponseRecPtr As MQPTR  'Address of first response record'
  AlternateSecurityId As MQBYTE40 'Alternate security identifier'
  ResolvedQName As String*48 'Resolved queue name'
  ResolvedQMgrName As String*48 'Resolved queue manager name'
End Type

```

Detalles de campo para MQOD

La estructura del descriptor de objeto MQOD se utiliza para especificar un objeto por su nombre. La estructura es un parámetro de entrada/salida en las llamadas MQOPEN y MQPUT1.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
StrucId (identificador de estructura)	MQOD_STRUC_ID	'0D-1'
Versión (número de versión de la estructura)	MQOD_VERSION_1	1
ObjectType (tipo de objeto)	MQOT_Q	1
ObjectName (nombre del objeto)	Ninguna	Cadena nula o espacios en blanco
ObjectQMgrName (nombre del gestor de colas de objetos)	Ninguna	Cadena nula o espacios en blanco
DynamicQName (nombre de la cola dinámica)	Ninguna	'CSQ.*' en z/OS ; 'AMQ.*' de lo contrario
AlternateUserId (identificador de usuario alternativo)	Ninguna	Cadena nula o espacios en blanco
Nota: Los campos restantes se ignoran si <i>Version</i> es menor que MQOD_VERSION_2.		

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>RecsPresent</u> (número de registros de objetos presentes)	Ninguna	0
<u>KnownDestCount</u> (número de colas locales abiertas correctamente)	Ninguna	0
<u>UnknownDestCount</u> (número de colas remotas abiertas correctamente)	Ninguna	0
<u>InvalidDestCount</u> (número de colas que no se pudieron abrir)	Ninguna	0
<u>ObjectRecOffset</u> (desplazamiento del primer registro de objeto desde el inicio de MQOD)	Ninguna	0
<u>ResponseRecOffset</u> (desplazamiento del primer registro de respuesta desde el inicio de MQOD)	Ninguna	0
<u>ObjectRecPtr</u> (dirección del primer registro de objeto)	Ninguna	Puntero nulo o bytes nulos
<u>ResponseRecPtr</u> (dirección del primer registro de respuesta)	Ninguna	Puntero nulo o bytes nulos
Nota: Los campos restantes se ignoran si <i>Version</i> es menor que MQOD_VERSION_3.		
<u>AlternateSecurityId</u> (identificador de seguridad alternativo)	MQSID_NINGUNO	Nulos
<u>ResolvedQName</u> (nombre de la cola resuelta)	Ninguna	Cadena nula o espacios en blanco
<u>ResolvedQMgrName</u> (nombre del gestor de colas resuelto)	Ninguna	Cadena nula o espacios en blanco
Nota: Los campos restantes se ignoran si <i>Version</i> es menor que MQOD_VERSION_4.		
<u>ObjectString</u> (nombre largo del objeto)	MQCHARV_DEFAULT	Según se define para MQCHARV
<u>SelectionString</u> (cadena de selección)	MQCHARV_DEFAULT	Según se define para MQCHARV
<u>ResObjectString</u> (nombre de objeto largo resuelto)	MQCHARV_DEFAULT	Según se define para MQCHARV
<u>ResolvedType</u> (tipo de objeto resuelto)	MQOT_NINGUNO	0
Notas: 1. El símbolo ~ representa un único carácter en blanco. 2. El valor «Cadena nula o espacios en blanco» denota la cadena nula en C y los caracteres en blanco en otros lenguajes de programación. 3. En el lenguaje de programación C, la variable macro MQOD_DEFAULT contiene los valores que se enumeran en la tabla. Se puede utilizar de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQOD MyOD = {MQOD_DEFAULT};</pre>		

StrucId (MQCHAR4)

Este es el identificador de estructura de la estructura del descriptor de objeto. Siempre es un campo de entrada. Su valor es MQOD_STRUC_ID.

El valor debe ser:

MQOD_STRUC_ID

Identificador de la estructura del descriptor del objeto.

Para el lenguaje de programación C, también se define la constante MQOD_STRUC_ID_ARRAY. Tiene el mismo valor que MQOD_STRUC_ID, pero es una matriz de caracteres en lugar de una cadena.

Versión (MQLONG)

Este es el número de versión de la estructura; el valor debe ser uno de los siguientes:

MQOD_VERSION_1

Version-1 estructura descriptora de objetos.

MQOD_VERSION_2

Version-2 estructura descriptora de objetos.

MQOD_VERSION_3

Version-3 estructura descriptora de objetos.

MQOD_VERSION_4

Version-4 estructura descriptora de objetos.

Todas las versiones son compatibles con todos los entornos IBM MQ de V7.0.

Los campos que solo existen en las versiones más recientes de la estructura se identifican como tales en las descripciones de los campos. La siguiente constante especifica el número de versión de la versión actual:

MQOD_VERSIÓN_ACTUAL

Versión actual de la estructura del descriptor de objetos.

Este campo siempre es un campo de entrada. El valor inicial de este campo es MQOD_VERSION_1.

ObjectType (MQLONG)

El tipo de objeto que se nombra en el descriptor de objetos. Los valores posibles son:

MQOT_CLNTCONN_CANAL

Canal de conexión del cliente. El nombre del objeto se encuentra en el *ObjectName* campo.

MQOT_Q

Cola. El nombre del objeto se encuentra en el *ObjectName* campo.

LISTA_DE_NOMBRES_MQOT

Lista de nombres. El nombre del objeto se encuentra en el *ObjectName* campo

MQOT_PROCESO

. El nombre del objeto se encuentra en el *ObjectName* campo

MQOT_Q_MGR

Gestor de colas. El nombre del objeto se encuentra en el *ObjectName* campo

MQOT_TEMA

. El nombre completo del tema se puede construir a partir de dos campos diferentes: *ObjectName* y *ObjectString*.

Para obtener más información sobre cómo se utilizan esos dos campos, consulte [Combinación de cadenas de temas](#).

Este campo siempre es un campo de entrada. El valor inicial de este campo es MQOT_Q.

ObjectName (MQCHAR48)

Este es el nombre local del objeto tal y como se define en el gestor de colas identificado por *ObjectQMgrName*. El nombre puede contener los siguientes caracteres:

- Caracteres alfabéticos mayúsculos (de la A a la Z)
- Letras minúsculas (de la a a la z)
- Dígitos numéricos (del 0 al 9)
- Punto (.), barra inclinada (/), guión bajo (_), porcentaje (%)

El nombre no debe contener espacios en blanco al principio ni en el interior, pero puede contener espacios en blanco al final. Utilice un carácter nulo para indicar el final de los datos significativos en el nombre; el carácter nulo y cualquier carácter que le siga se tratan como espacios en blanco. Las siguientes restricciones se aplican en los entornos indicados:

- En los sistemas que utilizan EBCDIC Katakana, no se pueden utilizar caracteres en minúscula.
- En z/OS:
 - Evite los nombres que comiencen o terminen con un guión bajo, ya que no pueden ser procesados por los paneles de operaciones y control.
 - El carácter por ciento tiene un significado especial para RACF. Si RACF se utiliza como gestor de seguridad externo, los nombres no deben contener el símbolo %. Si lo hacen, esos nombres no se incluyen en ninguna comprobación de seguridad cuando se RACF utilizan perfiles genéricos.
- En IBM i, los nombres que contengan caracteres en minúscula, barra inclinada o porcentaje deben escribirse entre comillas cuando se especifiquen en comandos. Estas comillas no deben especificarse para nombres que aparecen como campos en estructuras o como parámetros en llamadas.

El nombre completo del tema se puede construir a partir de dos campos diferentes: *ObjectName* y *ObjectString*. Para obtener más información sobre cómo se utilizan estos dos campos, consulte [Combinación de cadenas de temas](#).

Los siguientes puntos se aplican a los tipos de objetos indicados:

- Si *ObjectName* es el nombre de una cola modelo, el gestor de colas crea una cola dinámica con los atributos de la cola modelo y devuelve en el *ObjectName* campo el nombre de la cola creada. Una cola modelo solo se puede especificar en la llamada MQOPEN; una cola modelo no es válida en la llamada MQPUT1.
- Si *ObjectName* es el nombre de una cola alias con TARGTYPE(TOPIC), primero se realiza una comprobación de seguridad en la cola alias nombrada; esto es normal cuando se utilizan colas alias. Cuando la comprobación de seguridad finaliza correctamente, la llamada MQOPEN continuará y se comportará como una llamada MQOPEN en un MQOT_TOPIC; esto incluye realizar una comprobación de seguridad en el objeto de tema administrativo.
- Si *ObjectName* e *ObjectQMgrName* identifican una cola compartida propiedad del grupo de intercambio de colas al que pertenece el gestor de colas local, no debe existir también una definición de cola con el mismo nombre en el gestor de colas local. Si existe tal definición (una cola local, una cola alias, una cola remota o una cola modelo), la llamada falla con el código de motivo MQRC_OBJECT_NOT_UNIQUE.
- Si el objeto que se abre es una lista de distribución (es decir, *RecsPresent* está presente y es mayor que cero), *ObjectName* debe estar en blanco o ser una cadena nula. Si no se cumple esta condición, la llamada falla con el código de motivo MQRC_OBJECT_NAME_ERROR.
- Si *ObjectType* es MQOT_Q_MGR, se aplican reglas especiales; en este caso, el nombre debe estar completamente en blanco hasta el primer carácter nulo o el final del campo.

Este es un campo de entrada/salida para la llamada MQOPEN cuando *ObjectName* es el nombre de una cola de modelos, y un campo de solo entrada en todos los demás casos. La longitud de este campo viene determinada por MQ_Q_NAME_LENGTH. El valor inicial de este campo es la cadena nula en C y 48 caracteres en blanco en otros lenguajes de programación.

ObjectQMgrName (MQCHAR48)

Este es el nombre del gestor de colas en el que se define el *ObjectName* objeto. Los caracteres válidos en el nombre son los mismos que para *ObjectName* (véase “ObjectName (MQCHAR48)” en la página 462). Un nombre que está completamente en blanco hasta el primer carácter nulo o el final del campo indica el gestor de colas al que está conectada la aplicación (el gestor de colas local).

Los siguientes puntos se aplican a los tipos de objetos indicados:

- Si *ObjectType* es MQOT_TOPIC, MQOT_NAMELIST, MQOT_PROCESS o MQOT_Q_MGR, *ObjectQMgrName* debe estar en blanco o ser el nombre del gestor de colas local.
- Si *ObjectName* es el nombre de una cola modelo, el gestor de colas crea una cola dinámica con los atributos de la cola modelo y devuelve en el *ObjectQMgrName* campo el nombre del gestor de colas en el que se crea la cola; este es el nombre del gestor de colas local. Una cola modelo solo se puede especificar en la llamada MQOPEN; una cola modelo no es válida en la llamada MQPUT1.
- Si *ObjectName* es el nombre de una cola de clúster y *ObjectQMgrName* está en blanco, el destino de los mensajes enviados utilizando el identificador de cola devuelto por la llamada MQOPEN es elegido por el gestor de colas (o la salida de carga de trabajo del clúster, si hay una instalada) de la siguiente manera:
 - Si se especifica MQOO_BIND_ON_OPEN, el gestor de colas selecciona una instancia concreta de la cola del clúster mientras procesa la llamada MQOPEN, y todos los mensajes enviados mediante este identificador de cola se envían a esa instancia.
 - Si se especifica MQOO_BIND_NOT_FIXED, el gestor de colas puede elegir una instancia diferente de la cola de destino (que reside en un gestor de colas diferente del clúster) para cada llamada MQPUT sucesiva que utilice este identificador de cola.

Si la aplicación necesita enviar un mensaje a una instancia *específica* de una cola de clúster (es decir, una instancia de cola que reside en un gestor de colas concreto del clúster), la aplicación debe especificar el nombre de ese gestor de colas en el *ObjectQMgrName* campo. Esto obliga al gestor de colas local a enviar el mensaje al gestor de colas de destino especificado.

- Si *ObjectName* es el nombre de una cola compartida que es propiedad del grupo de uso compartido de colas al que pertenece el gestor de colas local, *ObjectQMgrName* puede ser el nombre del grupo de uso compartido de colas, el nombre del gestor de colas local o estar en blanco; el mensaje se coloca en la misma cola independientemente del valor especificado.

Los grupos de compartición de colas solo se admiten en z/OS.

- Si *ObjectName* es el nombre de una cola compartida que pertenece a un grupo de uso compartido de colas remoto (es decir, un grupo de uso compartido de colas al que no pertenece el gestor de colas local), *ObjectQMgrName* debe ser el nombre del grupo de uso compartido de colas. Puede utilizar el nombre de un gestor de colas que pertenezca a ese grupo, pero esto puede retrasar el mensaje si ese gestor de colas concreto no está disponible cuando el mensaje llega al grupo de intercambio de colas.
- Si el objeto que se abre es una lista de distribución (es decir, *RecsPresent* es mayor que cero), *ObjectQMgrName* debe estar en blanco o ser una cadena nula. Si no se cumple esta condición, la llamada falla con el código de motivo MQRC_OBJECT_Q_MGR_NAME_ERROR.

Este es un campo de entrada/salida para la llamada MQOPEN cuando *ObjectName* es el nombre de una cola de modelos, y un campo de solo entrada en todos los demás casos. La longitud de este campo viene determinada por MQ_Q_MGR_NAME_LENGTH. El valor inicial de este campo es la cadena nula en C y 48 caracteres en blanco en otros lenguajes de programación.

DynamicQName (MQCHAR48)

Este es el nombre de una cola dinámica que se creará mediante la llamada MQOPEN. Esto solo es relevante cuando *ObjectName* especifica el nombre de una cola modelo; en todos los demás casos, *DynamicQName* se ignora.

Los caracteres válidos en el nombre son los mismos que para *ObjectName*, excepto que también es válido el asterisco. Un nombre que esté en blanco (o en el que solo haya espacios en blanco antes del primer carácter nulo) no es válido si *ObjectName* es el nombre de una cola de modelos.

Si el último carácter no en blanco del nombre es un asterisco (*), el gestor de colas sustituye el asterisco por una cadena de caracteres que garantiza que el nombre generado para la cola sea único en el gestor de colas local. Para permitir un número suficiente de caracteres para ello, el asterisco solo es válido en las posiciones 1 a 33. No debe haber ningún carácter que no sea un espacio en blanco o un carácter nulo después del asterisco.

Es válido que el asterisco aparezca en la primera posición del carácter, en cuyo caso el nombre se compone únicamente de los caracteres generados por el gestor de colas.

En z/OS, no utilice un nombre con un asterisco en la primera posición, ya que no se pueden realizar comprobaciones de seguridad en una cola con un nombre completo generado automáticamente.

Este es un campo de entrada. La longitud de este campo viene determinada por MQ_Q_NAME_LENGTH. El valor inicial de este campo viene determinado por el entorno:

- En z/OS, el valor es 'CSQ.*'.
- En otras plataformas, el valor es 'AMQ.*'.

El valor es una cadena terminada en nulo en C y una cadena rellena con espacios en blanco en otros lenguajes de programación.

AlternateUserId (MQCHAR12)

Si se especifica MQOO_ALTERNATE_USER_AUTHORITY para la llamada MQOPEN, o MQPMO_ALTERNATE_USER_AUTHORITY para la llamada MQPUT1, este campo contiene un identificador de usuario alternativo que se utiliza para comprobar la autorización para la apertura, en lugar del identificador de usuario con el que se está ejecutando actualmente la aplicación. Sin embargo, algunas comprobaciones siguen realizándose con el identificador de usuario actual (por ejemplo, las comprobaciones de contexto).

Si se especifica MQOO_ALTERNATE_USER_AUTHORITY o MQPMO_ALTERNATE_USER_AUTHORITY y este campo está completamente en blanco hasta el primer carácter nulo o el final del campo, la apertura solo podrá realizarse con éxito si no se necesita autorización de usuario para abrir este objeto con las opciones especificadas.

Si no se especifica ni MQOO_ALTERNATE_USER_AUTHORITY ni MQPMO_ALTERNATE_USER_AUTHORITY, este campo se ignora.

En los entornos indicados existen las siguientes diferencias:

- En z/OS, solo se utilizan los primeros *AlternateUserId* 8 caracteres de para comprobar la autorización para la apertura. Sin embargo, el identificador de usuario actual debe estar autorizado para especificar este identificador de usuario alternativo concreto; para esta comprobación se utilizan los 12 caracteres del identificador de usuario alternativo. El identificador de usuario solo debe contener caracteres permitidos por el administrador de seguridad externo.

Si *AlternateUserId* se especifica para una cola, el valor puede ser utilizado posteriormente por el gestor de colas cuando se envían mensajes. Si las opciones MQPMO_*_CONTEXT especificadas en la llamada MQPUT o MQPUT1 hacen que el gestor de colas genere la información del contexto de identidad, el gestor de colas coloca el *AlternateUserId* en el *UserIdentifier* campo del MQMD del mensaje, en lugar del identificador de usuario actual.

- En otros entornos, solo *AlternateUserId* se utiliza para comprobaciones de control de acceso en el objeto que se está abriendo. Si el objeto es una cola, no *AlternateUserId* afecta al contenido del *UserIdentifier* campo en el MQMD de los mensajes enviados utilizando ese identificador de cola.

Este es un campo de entrada. La longitud de este campo viene determinada por MQ_USER_ID_LENGTH. El valor inicial de este campo es la cadena nula en C y 12 caracteres en blanco en otros lenguajes de programación.

RecsPresent (MQLONG)

Este es el número de registros de objetos MQOR que ha proporcionado la aplicación. Si este número es mayor que cero, indica que se está abriendo una lista de distribución, siendo *RecsPresent* el número de colas de destino en la lista. Una lista de distribución solo puede contener un destino.

El valor de no *RecsPresent* debe ser inferior a cero, y si es superior a cero, *ObjectType* debe ser MQOT_Q; la llamada falla con el código de motivo MQRC_RECS_PRESENT_ERROR si no se cumplen estas condiciones.

En z/OS, este campo debe ser cero.

Este es un campo de entrada. El valor inicial de este campo es 0. Este campo se ignora si *Version* es menor que MQOD_VERSION_2.

KnownDestCount (MQLONG)

Este es el número de colas de la lista de distribución que se resuelven en colas locales y que se han abierto correctamente. El recuento no incluye las colas que se resuelven en colas remotas (aunque inicialmente se utilice una cola de transmisión local para almacenar el mensaje). Si está presente, este campo también se establece al abrir una cola única que no está en una lista de distribución.

Se trata de un campo de salida. El valor inicial de este campo es 0. Este campo se ignora si *Version* es menor que MQOD_VERSION_1.

UnknownDestCount (MQLONG)

Este es el número de colas de la lista de distribución que se resuelven en colas remotas y que se abrieron correctamente. Si está presente, este campo también se establece al abrir una cola única que no está en una lista de distribución.

Se trata de un campo de salida. El valor inicial de este campo es 0. Este campo se ignora si *Version* es menor que MQOD_VERSION_1.

InvalidDestCount (MQLONG)

Este es el número de colas de la lista de distribución que no se pudieron abrir correctamente. Si está presente, este campo también se establece al abrir una cola única que no está en una lista de distribución.

Nota: Si está presente, este campo solo se establece si el **CompCode** parámetro en la llamada MQOPEN o MQPUT1 es MQCC_OK o MQCC_WARNING; no se establece si el **CompCode** parámetro es MQCC_FAILED.

Se trata de un campo de salida. El valor inicial de este campo es 0. Este campo se ignora si *Version* es menor que MQOD_VERSION_1.

ObjectRecOffset (MQLONG)

Es el desplazamiento en bytes del primer registro de objeto MQOR desde el inicio de la estructura MQOD. El desplazamiento puede ser positivo o negativo. *ObjectRecOffset* Se utiliza solo cuando se abre una lista de distribución. El campo se ignora si *RecsPresent* es cero.

Cuando se abre una lista de distribución, se debe proporcionar una matriz de uno o más registros de objetos MQOR para especificar los nombres de las colas de destino en la lista de distribución. Esto se puede hacer de dos maneras:

- Al utilizar el campo de compensación *ObjectRecOffset*.

En este caso, la aplicación debe declarar su propia estructura que contenga un MQOD seguido de la matriz de registros MQOR (con tantos elementos de matriz como sean necesarios) y establecer *ObjectRecOffset* en la posición del primer elemento de la matriz desde el inicio del MQOD. Asegúrese de que este desplazamiento sea correcto y tenga un valor que pueda acomodarse dentro

de un MQLONG (el lenguaje de programación más restrictivo es COBOL, para el cual el rango válido es de -999 999 999 a +999 999 999).

Utilícelo *ObjectRecOffset* para lenguajes de programación que no admiten el tipo de datos puntero o que implementan el tipo de datos puntero de una forma que no es portable a diferentes entornos (por ejemplo, el lenguaje de programación COBOL).

- Al utilizar el campo puntero *ObjectRecPtr*.

En este caso, la aplicación puede declarar la matriz de estructuras MQOR por separado de la estructura MQOD y establecer *ObjectRecPtr* en la dirección de la matriz.

Utilícelo *ObjectRecPtr* para lenguajes de programación que admitan el tipo de datos puntero de forma que sea portátil a diferentes entornos (por ejemplo, el lenguaje de programación C).

Sea cual sea la técnica que elija, utilice una de *ObjectRecOffset* y *ObjectRecPtr* ; la llamada fallará con el código de motivo MQRC_OBJECT_RECORDS_ERROR si ambas son cero o si ambas son distintas de cero.

Este es un campo de entrada. El valor inicial de este campo es 0. Este campo se ignora si *Version* es menor que MQOD_VERSION_2.

ResponseRecOffset (MQLONG)

Es el desplazamiento en bytes del primer registro de respuesta MQRR desde el inicio de la estructura MQOD. El desplazamiento puede ser positivo o negativo. *ResponseRecOffset* Se utiliza solo cuando se abre una lista de distribución. El campo se ignora si *RecsPresent* es cero.

Cuando se abre una lista de distribución, puede proporcionar una matriz de uno o más registros de respuesta MQRR para identificar las colas que no se han podido abrir (*CompCode* campo en MQRR) y el motivo de cada error (*Reason* campo en MQRR). Los datos se devuelven en la matriz de registros de respuesta en el mismo orden en que aparecen los nombres de las colas en la matriz de registros de objetos. El gestor de colas establece los registros de respuesta solo cuando el resultado de la llamada es mixto (es decir, algunas colas se abrieron correctamente, mientras que otras fallaron, o todas fallaron pero por diferentes motivos); el código de motivo MQRC_MULTIPLE_REASONS de la llamada indica este caso. Si el mismo código de motivo se aplica a todas las colas, ese motivo se devuelve en el **Reason** parámetro de la llamada MQOPEN o MQPUT1, y los registros de respuesta no se establecen. Los registros de respuesta son opcionales, pero si se proporcionan, deben ser *RecsPresent* .

Los registros de respuesta se pueden proporcionar de la misma manera que los registros de objetos, ya sea especificando un desplazamiento en *ResponseRecOffset*, o especificando una dirección en *ResponseRecPtr* ; para obtener más información sobre cómo hacerlo, consulte [ObjectRecOffset](#). Sin embargo, no se puede utilizar *ResponseRecPtr* más de uno de *ResponseRecOffset* y; la llamada falla con el código de motivo MQRC_RESPONSE_RECORDS_ERROR si ambos son distintos de cero.

En la llamada MQPUT1, estos registros de respuesta se utilizan para devolver información sobre los errores que se producen al enviar el mensaje a las colas de la lista de distribución, así como los errores que se producen al abrir las colas. El código de finalización y el código de motivo de la operación de inserción de una cola sustituyen a los de la operación de apertura de esa cola solo si el código de finalización de esta última era MQCC_OK o MQCC_WARNING.

Este es un campo de entrada. El valor inicial de este campo es 0. Este campo se ignora si *Version* es menor que MQOD_VERSION_2.

ObjectRecPtr (MQPTR)

Esta es la dirección del primer registro de objetos MQOR. *ObjectRecPtr* Se utiliza solo cuando se abre una lista de distribución. El campo se ignora si *RecsPresent* es cero.

Puede utilizar o *ObjectRecOffset* o *ObjectRecPtr* para especificar los registros de objeto, pero no ambos; para obtener una descripción del *ObjectRecOffset* campo, consulte [ObjectRecOffset](#). Si no lo utiliza *ObjectRecPtr*, establézcalo en el puntero nulo o en bytes nulos.

Este es un campo de entrada. El valor inicial de este campo es el puntero nulo en aquellos lenguajes de programación que admiten punteros y, en los demás casos, una cadena de bytes totalmente nulos. Este campo se ignora si *Version* es menor que MQOD_VERSION_2.

Nota: En plataformas donde el lenguaje de programación no admite el tipo de datos puntero, este campo se declara como una cadena de bytes de la longitud adecuada, con el valor inicial siendo la cadena de bytes totalmente nula.

ResponseRecPtr (MQPTR)

Esta es la dirección del primer registro de respuesta MQRR. *ResponseRecPtr* Se utiliza solo cuando se abre una lista de distribución. El campo se ignora si *RecsPresent* es cero.

Utilice o *ResponseRecOffsetResponseRecPtr* para especificar los registros de respuesta, pero no ambos; para obtener más información, consulte *ResponseRecOffset*. Si no lo utiliza *ResponseRecPtr*, establézcalo en el puntero nulo o en bytes nulos.

Este es un campo de entrada. El valor inicial de este campo es el puntero nulo en aquellos lenguajes de programación que admiten punteros y, en los demás casos, una cadena de bytes totalmente nulos. Este campo se ignora si *Version* es menor que MQOD_VERSION_2.

Nota: En plataformas donde el lenguaje de programación no admite el tipo de datos puntero, este campo se declara como una cadena de bytes de la longitud adecuada, con el valor inicial siendo la cadena de bytes totalmente nula.

AlternateSecurityId (MQBYTE40)

Se trata de un identificador de seguridad que se transmite junto con el *AlternateUserId* al servicio de autorización para permitir que se realicen las comprobaciones de autorización adecuadas. *AlternateSecurityId* se utiliza solo si:

- MQOO_ALTERNATE_USER_AUTHORITY se especifica en la llamada MQOPEN, o
- MQPMO_ALTERNATE_USER_AUTHORITY se especifica en la llamada a MQPUT1

y el *AlternateUserId* campo no está completamente en blanco hasta el primer carácter nulo o el final del campo.

En Windows, se *AlternateSecurityId* puede utilizar para proporcionar el identificador Windows de seguridad (SID) que identifica de forma única al *AlternateUserId*. El SID de un usuario se puede obtener del Windows sistema mediante la llamada a la LookupAccountName () Windows API.

En z/OS, este campo se ignora.

El *AlternateSecurityId* campo tiene la siguiente estructura:

- El primer byte es un entero binario que contiene la longitud de los datos significativos que le siguen; el valor excluye el propio byte de longitud. Si no hay ningún identificador de seguridad, la longitud es cero.
- El segundo byte indica el tipo de identificador de seguridad presente; son posibles los siguientes valores:

MQSIDT_NT_ID_DE_SEGURIDAD

Windows identificador de seguridad.

MQSIDT_NINGUNO

Sin identificador de seguridad.

- El tercer byte y los siguientes hasta la longitud definida por el primer byte contienen el identificador de seguridad propiamente dicho.
- Los bytes restantes en el campo se establecen en cero binario.

Puede utilizar el siguiente valor especial:

MQSID_NINGUNO

No se ha especificado ningún identificador de seguridad.

El valor es un cero binario para la longitud del campo.

Para el lenguaje de programación C, también se define la constante MQSID_NONE_ARRAY, que tiene el mismo valor que MQSID_NONE, pero es una matriz de caracteres en lugar de una cadena.

Este es un campo de entrada. La longitud de este campo viene dada por MQ_SECURITY_ID_LENGTH. El valor inicial de este campo es MQSID_NONE. Este campo se ignora si *Version* es menor que MQOD_VERSION_3.

ResolvedQName (MQCHAR48)

Este es el nombre de la cola de destino después de que el administrador de colas local resuelva el nombre. El nombre devuelto es el nombre de una cola que existe en el gestor de colas identificado por *ResolvedQMgrName*.

Solo se devuelve un valor distinto de blanco si el objeto es una cola única abierta para exploración, entrada o salida (o cualquier combinación). Si el objeto abierto es cualquiera de los siguientes, *ResolvedQName* se establece en espacios en blanco:

- No es una cola
- Una cola, pero no abierta para exploración, entrada o salida
- Una lista de distribución
- Una cola de alias que hace referencia a un objeto de tema (consulte en [ResObjectString](#) su lugar).
- Una cola de alias que se resuelve en un objeto de tema.

Se trata de un campo de salida. La longitud de este campo viene determinada por MQ_Q_NAME_LENGTH. El valor inicial de este campo es la cadena nula en C y 48 caracteres en blanco en otros lenguajes de programación. Este campo se ignora si *Version* es menor que MQOD_VERSION_3.

ResolvedQMgrName (MQCHAR48)

Este es el nombre del gestor de colas de destino después de que el gestor de colas local resuelva el nombre. El nombre devuelto es el nombre del gestor de colas propietario de la cola identificada por *ResolvedQName*. *ResolvedQMgrName* puede ser el nombre del administrador de colas local.

Si *ResolvedQName* es una cola compartida que pertenece al grupo de uso compartido de colas al que pertenece el gestor de colas local, *ResolvedQMgrName* es el nombre del grupo de uso compartido de colas. Si la cola pertenece a otro grupo de uso compartido de colas, *ResolvedQName* puede ser el nombre del grupo de uso compartido de colas o el nombre de un gestor de colas que sea miembro del grupo de uso compartido de colas (la naturaleza del valor devuelto viene determinada por las definiciones de colas que existen en el gestor de colas local).

Solo se devuelve un valor distinto de blanco si el objeto es una cola única abierta para exploración, entrada o salida (o cualquier combinación). Si el objeto abierto es cualquiera de los siguientes, *ResolvedQMgrName* se establece en espacios en blanco:

- No es una cola
- Una cola, pero no abierta para exploración, entrada o salida
- Una cola de clúster con MQOO_BIND_NOT_FIXED especificado (o con MQOO_BIND_AS_Q_DEF en vigor cuando el **DefBind** atributo de cola tiene el valor MQBND_BIND_NOT_FIXED)
- Una lista de distribución

Se trata de un campo de salida. La longitud de este campo viene determinada por MQ_Q_NAME_LENGTH. El valor inicial de este campo es la cadena nula en C y 48 caracteres en blanco en otros lenguajes de programación. Este campo se ignora si *Version* es menor que MQOD_VERSION_3.

ObjectString (MQCHARV)

El campo « ObjectString » especifica el nombre completo del objeto.

Esto especifica el nombre largo del objeto que se va a utilizar. Este campo solo se tiene en cuenta para determinados valores de *ObjectType*, y se ignora para todos los demás valores. Consulte la descripción de *ObjectType* para obtener más información sobre qué valores indican que se utiliza este campo.

Si *ObjectString* se especifica incorrectamente, según la descripción de cómo utilizar la estructura *MQCHARV*, o si supera la longitud máxima, la llamada falla con el código de motivo *MQRC_OBJECT_STRING_ERROR*.

Este es un campo de entrada. Los valores iniciales de los campos de esta estructura son los mismos que los de la estructura *MQCHARV*.

El nombre completo del tema se puede construir a partir de dos campos diferentes: *ObjectName* y *ObjectString*. Para obtener más información sobre cómo se utilizan estos dos campos, consulte [Combinación de cadenas de temas](#).

SelectionString (MQCHARV)

Esta es la cadena utilizada para proporcionar los criterios de selección que se utilizan al recuperar mensajes de una cola.

SelectionString no se debe proporcionar en los siguientes casos:

- Si no *ObjectType* es *MQOT_Q*
- Si la cola que se está abriendo no se abre utilizando una de las opciones *MQOO_BROWSE* o *MQOO_INPUT_**

Si *SelectionString* se proporciona en estos casos, la llamada falla con el código de motivo *MQRC_SELECTOR_INVALID_FOR_TYPE*.

Si *SelectionString* se especifica incorrectamente, según la descripción de cómo utilizar la “*MQCHARV* - Cadena de longitud variable” en la página 243 estructura, o si supera la longitud máxima, la llamada falla con el código de motivo *MQRC_SELECTION_STRING_ERROR*. La longitud máxima de *SelectionString* es *MQ_SELECTOR_LENGTH*.

SelectionString El uso se describe en [Selectores](#).

ResObjectString (MQCHARV)

El campo « *ResObjectString* » es el nombre largo del objeto después de que el gestor de colas resuelva el nombre proporcionado en el *ObjectName* campo «».

Este campo solo se devuelve para temas y alias de cola que hacen referencia a un objeto de tema.

Si se proporciona el nombre largo del objeto en *ObjectString* y no se proporciona nada en *ObjectName*, entonces el valor devuelto en este campo es el mismo que el proporcionado en *ObjectString*.

Si se omite este campo (es decir, si *ResObjectString.VSBufSize* es cero), no se devolverá *ResObjectString* el, pero se devolverá la longitud en *ResObjectString.VSLength*.

Si la longitud del búfer (proporcionada en *ResObjectString.VSBufSize*) es menor que la completa *ResObjectString*, la cadena se truncará y devolverá tantos caracteres de la derecha como quepan en el búfer proporcionado.

Si *ResObjectString* se especifica incorrectamente, según la descripción de cómo utilizar la estructura *MQCHARV*, o si supera la longitud máxima, la llamada falla con el código de motivo *MQRC_RES_OBJECT_STRING_ERROR*.

ResolvedType (MQLONG)

El tipo del objeto (base) resuelto que se está abriendo.

Los valores posibles son:

MQOT_Q

El objeto resuelto es una cola. Este valor se aplica cuando se abre una cola directamente o cuando se abre una cola alias que apunta a una cola.

MQOT_TEMA

El objeto resuelto es un tema. Este valor se aplica cuando se abre un tema directamente o cuando se abre una cola de alias que apunta a un objeto de tema.

MQOT_NINGUNO

El tipo resuelto no es ni una cola ni un tema.

MQOR - Registro de objetos

Utilice la estructura MQOR para especificar el nombre de la cola y el nombre del gestor de colas de una sola cola de destino. MQOR es una estructura de entrada para las llamadas MQOPEN y MQPUT1.

Disponibilidad

La estructura MQOR está disponible en las siguientes plataformas:

-  AIX
-  IBM i
-  Linux
-  Windows

y para IBM MQ MQI clients conectados a estos sistemas.

Conjunto de caracteres y codificación

Los datos en MQOR deben estar en el juego de caracteres proporcionado por el atributo del **CodedCharSetId** gestor de colas y la codificación del gestor de colas local proporcionada por MQENC_NATIVE. Sin embargo, si la aplicación se ejecuta como un cliente MQ MQI, la estructura debe estar en el conjunto de caracteres y la codificación del cliente.

Utilización

Al proporcionar una matriz de estas estructuras en la llamada MQOPEN, puede abrir una lista de colas; esta lista se denomina lista de distribución. Cada mensaje enviado utilizando el identificador de cola devuelto por esa llamada MQOPEN se coloca en cada una de las colas de la lista, siempre que la cola se haya abierto correctamente.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 525. Campos en MQOR para MQOR		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>ObjectName</u> (nombre del objeto)	Ninguna	Cadena nula o espacios en blanco
<u>ObjectQMgrName</u> (nombre del gestor de colas de objetos)	Ninguna	Cadena nula o espacios en blanco

Tabla 525. Campos en MQOR para MQOR (continuación)

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
Notas: 1. El valor «Cadena nula o espacios en blanco» denota la cadena nula en C y los caracteres en blanco en otros lenguajes de programación. 2. En el lenguaje de programación C, la variable macroMQOR_DEFAULT contiene los valores que se enumeran en la tabla. Se puede utilizar de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQOR MyOR = {MQOR_DEFAULT};</pre>		

Declaraciones lingüísticas

Declaración C para MQOR

```
typedef struct tagMQOR MQOR;
struct tagMQOR {
    MQCHAR48  ObjectName;      /* Object name */
    MQCHAR48  ObjectQMgrName; /* Object queue manager name */
};
```

COBOL declaración para MQOR

```
**  MQOR structure
10  MQOR.
**    Object name
15  MQOR-OBJECTNAME      PIC X(48).
**    Object queue manager name
15  MQOR-OBJECTQMGRNAME PIC X(48).
```

PL/I declaración para MQOR

```
dcl
1 MQOR based,
3 ObjectName      char(48), /* Object name */
3 ObjectQMgrName char(48); /* Object queue manager name */
```

Declaración de Visual Basic para MQOR

```
Type MQOR
    ObjectName      As String*48 'Object name'
    ObjectQMgrName As String*48 'Object queue manager name'
End Type
```

Detalles de campo para MQOR

Utilice la estructura de registro del objeto MQOR para especificar el nombre de la cola y el nombre del gestor de colas de una sola cola de destino. MQOR es una estructura de entrada para las llamadas MQOPEN y MQPUT1.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 526. Campos en MQOR para MQOR

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>ObjectName</u> (nombre del objeto)	Ninguna	Cadena nula o espacios en blanco
<u>ObjectQMgrName</u> (nombre del gestor de colas de objetos)	Ninguna	Cadena nula o espacios en blanco
Notas: 1. El valor «Cadena nula o espacios en blanco» denota la cadena nula en C y los caracteres en blanco en otros lenguajes de programación. 2. En el lenguaje de programación C, la variable macroMQOR_DEFAULT contiene los valores que se enumeran en la tabla. Se puede utilizar de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQOR MyOR = {MQOR_DEFAULT};</pre>		

ObjectName (MQCHAR48)

Es igual que el ObjectName campo de la estructura MQOD, salvo que:

- Debe ser el nombre de una cola.
- No debe ser el nombre de una cola modelo.

Este campo siempre es un campo de entrada. El valor inicial de este campo es la cadena nula en C y 48 caracteres en blanco en otros lenguajes de programación.

ObjectQMgrName (MQCHAR48)

Es igual que el ObjectQMgrName campo de la estructura MQOD.

Este campo siempre es un campo de entrada. El valor inicial de este campo es la cadena nula en C y 48 caracteres en blanco en otros lenguajes de programación.

MQPD - Descriptor de propiedad

La **MQPD** estructura se utiliza para definir los atributos de una propiedad. La estructura es un parámetro de entrada/salida en la llamada MQSETMP y un parámetro de salida en la llamada MQINQMP.

Disponibilidad

La **MQPD** estructura está disponible en las siguientes plataformas:

-  AIX
-  IBM i
-  Linux
-  Windows
-  z/OS

y para IBM MQ MQI clients.

Conjunto de caracteres y codificación

Los datos en **MQPD** deben estar en el juego de caracteres de la aplicación y la codificación de la aplicación (**MQENC_NATIVE**).

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 527. Campos en MQPD		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>StrucId</u> (identificador de estructura)	MQPD_STRUC_ID	'PD'
<u>Versión</u> (número de versión de la estructura)	MQPD_VERSION_1	1
<u>Opciones</u> (opciones)	MQPD_NINGUNO	0
<u>Soporte</u> (soporte requerido para la propiedad del mensaje)	MQPD_SOPORTE_OPCIONAL	0
<u>Contexto</u> (contexto del mensaje al que pertenece la propiedad)	MQPD_NO_CONTEXT	0
<u>CopyOptions</u> (copiar opciones a las que pertenece la propiedad)	MQCOPY_DEFAULT	0
Notas: 1. En el lenguaje de programación C, la variable macro MQPD_DEFAULT contiene los valores que se enumeran en la tabla. Se puede utilizar de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQPD MyPD = {MQPD_DEFAULT};</pre>		

Declaraciones lingüísticas

Declaración C para MQPD

```
typedef struct tagMQPD MQPD;
struct tagMQPD {
    MQCHAR4  StrucId;      /* Structure identifier */
    MQLONG   Version;      /* Structure version number */
    MQLONG   Options;      /* Options that control the action of
                           MQSETMP and MQINQMP */
    MQLONG   Support;      /* Property support option */
    MQLONG   Context;      /* Property context */
    MQLONG   CopyOptions; /* Property copy options */
};
```

COBOL declaración para MQPD

```
**  MQPD structure
10  MQPD.
**    Structure identifier
15  MQPD-STRUCID PIC X(4).
**    Structure version number
15  MQPD-VERSION PIC S9(9) BINARY.
**    Options that control the action of MQSETMP and
**    MQINQMP
15  MQPD-OPTIONS PIC S9(9) BINARY.
**    Property support option
15  MQPD-SUPPORT PIC S9(9) BINARY.
**    Property context
15  MQPD-CONTEXT PIC S9(9) BINARY.
```

```

**      Property copy options
15 MQPD-COPYOPTIONS PIC S9(9) BINARY.

```

PL/I declaración para MQPD

```

dcl
  1 MQPD based,
    3 StrucId      char(4),          /* Structure identifier */
    3 Version      fixed bin(31),    /* Structure version number */
    3 Options      fixed bin(31),    /* Options that control the action
                                     of MQSETMP and MQINQMP */
    3 Support      fixed bin(31),    /* Property support option */
    3 Context      fixed bin(31),    /* Property context */
    3 CopyOptions  fixed bin(31);    /* Property copy options */

```

High Level Assembler declaración para MQPD

```

MQPD          DSECT
MQPD_STRUCID  DS   CL4      Structure identifier
MQPD_VERSION  DS   F        Structure version number
MQPD_OPTIONS  DS   F        Options that control the
*               action of MQSETMP and MQINQMP
MQPD_SUPPORT  DS   F        Property support option
MQPD_CONTEXT  DS   F        Property context
MQPD_COPYOPTIONS DS   F      Property copy options
MQPD_LENGTH  EQU   *-MQPD
MQPD_AREA     DS    CL(MQPD_LENGTH)

```

Detalles de campo para MQPD

La estructura del **MQPD** descriptor de propiedad se utiliza para definir los atributos de una propiedad. La estructura es un parámetro de entrada/salida en la llamada MQSETMP y un parámetro de salida en la llamada MQINQMP.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 528. Campos en MQPD		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
StrucId (identificador de estructura)	MQPD_STRUC_ID	'PD'
Versión (número de versión de la estructura)	MQPD_VERSION_1	1
Opciones (opciones)	MQPD_NINGUNO	0
Soporte (soporte requerido para la propiedad del mensaje)	MQPD_SOPORTE_OPCIONAL	0
Contexto (contexto del mensaje al que pertenece la propiedad)	MQPD_NO_CONTEXT	0
CopyOptions (copiar opciones a las que pertenece la propiedad)	MQCOPY_DEFAULT	0
Notas: 1. En el lenguaje de programación C, la variable macro MQPD_DEFAULT contiene los valores que se enumeran en la tabla. Se puede utilizar de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQPD MyPD = {MQPD_DEFAULT};</pre>		

StrucId (MQCHAR4)

Este es el identificador de estructura de la estructura del descriptor de propiedad. Siempre es un campo de entrada. Su valor es MQPD_STRUC_ID.

El valor debe ser:

MQPD_STRUC_ID

Identificador de la estructura del descriptor de propiedad.

Para el lenguaje de programación C, también se **MQPD_STRUC_ID_ARRAY** define la constante. Esto tiene el mismo valor que **MQPD_STRUC_ID**, pero es una matriz de caracteres en lugar de una cadena.

Versión (MQLONG)

Este es el número de versión de la estructura; el valor debe ser:

MQPD_VERSION_1

Version-1 estructura del descriptor de propiedad.

La siguiente constante especifica el número de versión de la versión actual:

MQPD_VERSIÓN_ACTUAL

Versión actual de la estructura del descriptor de propiedad.

Este campo siempre es un campo de entrada. El valor inicial de este campo es **MQPD_VERSION_1**.

Opciones (MQLONG)

El valor debe ser:

MQPD_NINGUNO

No se ha especificado ninguna opción

Este campo siempre es un campo de entrada. El valor inicial de este campo es MQPD_NONE.

Soporte (MQLONG)

Este campo describe el nivel de compatibilidad con la propiedad del mensaje que se requiere del gestor de colas para que el mensaje que contiene esta propiedad se coloque en una cola. Esto solo se aplica a las propiedades IBM MQ definidas; la compatibilidad con el resto de propiedades es opcional.

El campo se establece automáticamente en el valor correcto cuando el gestor de colas conoce la propiedad IBM MQ definida. Si no se reconoce la propiedad, se asigna MQPD_SUPPORT_OPTIONAL. Cuando un gestor de colas recibe un mensaje que contiene una propiedad IBM MQ definida que el gestor de colas reconoce como incorrecta, el gestor de colas corrige el valor del *Support* campo.

Al establecer una propiedad IBM MQ definida mediante la llamada MQSETMP en un identificador de mensaje en el que se ha establecido la opción MQCMHO_NO_VALIDATION, *Support* se convierte en un campo de entrada. Esto permite que una aplicación coloque una propiedad IBM MQ definida, con el valor correcto, cuando la propiedad no es compatible con el gestor de colas conectado, pero el mensaje debe procesarse en otro gestor de colas.

El valor MQPD_SUPPORT_OPTIONAL siempre se asigna a propiedades que no son IBM MQpropiedades definidas.

La llamada MQINQMP devuelve uno de los siguientes valores, o se puede especificar uno de ellos, cuando se utiliza la llamada MQSETMP en un identificador de mensaje en el que se ha establecido la opción MQCMHO_NO_VALIDATION:

MQPD_SOPORTE OPCIONAL

La propiedad es aceptada por un gestor de colas aunque no sea compatible. La propiedad se puede descartar para que el mensaje fluya a un gestor de colas que no admite propiedades de mensajes. Este valor también se asigna a propiedades que no están IBM MQdefinidas.

MQPD_SOPORTE_NECESARIO

Se requiere soporte para la propiedad. El mensaje es rechazado por un gestor de colas que no admite la IBM MQ propiedad definida. La llamada MQPUT (MQPUT1) falla con el código de finalización MQCC_FAILED y el código de motivo MQRC_UNSUPPORTED_PROPERTY.

MQPD_SOPORTE_REQUERIDO_SI_LOCAL

El mensaje es rechazado por un gestor de colas que no admite la propiedad IBM MQ definida si el mensaje está destinado a una cola local. La llamada MQPUT (MQPUT1) falla con el código de finalización MQCC_FAILED y el código de motivo MQRC_UNSUPPORTED_PROPERTY.

La llamada MQPUT (MQPUT1) se realiza correctamente si el mensaje está destinado a un gestor de colas remoto.

Este es un campo de salida en la llamada MQINQMP y un campo de entrada en la llamada MQSETMP si el identificador de mensaje se creó con la opción MQCMHO_NO_VALIDATION establecida. El valor inicial de este campo es MQPD_SUPPORT_OPTIONAL.

Contexto (MQLONG)

Esto describe a qué contexto de mensaje pertenece la propiedad.

Cuando un gestor de colas recibe un mensaje que contiene una propiedad IBM MQ definida que el gestor de colas reconoce como incorrecta, el gestor de colas corrige el valor del *Context* campo.

Se puede especificar la siguiente opción:

MQPD_USUARIO_CONTEXTO

La propiedad está asociada al contexto de usuario.

No se requiere ninguna autorización especial para poder establecer una propiedad asociada al contexto de usuario utilizando la llamada MQSETMP.

Si no se requiere la opción descrita anteriormente, se puede utilizar la siguiente opción:

MQPD_NO_CONTEXT

La propiedad no está asociada a un contexto de mensaje.

Un valor no reconocido se rechaza con un *Reason* código de MQRC_PD_ERROR

Este es un campo de entrada/salida para la llamada MQSETMP y un campo de salida de la llamada MQINQMP. El valor inicial de este campo es MQPD_NO_CONTEXT.

CopyOptions (MQLONG)

Esto describe el tipo de mensajes en los que se debe copiar la propiedad. Este es un campo de salida exclusiva para propiedades definidas IBM MQ reconocidas; IBM MQ establece el valor adecuado.

Cuando un gestor de colas recibe un mensaje que contiene una propiedad IBM MQ definida que el gestor de colas reconoce como incorrecta, el gestor de colas corrige el valor del *CopyOptions* campo.

Puede especificar una o varias de estas opciones. Para especificar más de una opción, suma los valores (sin añadir la misma constante más de una vez) o combínalos mediante la operación OR bit a bit (si el lenguaje de programación admite operaciones bit a bit).

MQCOPY_FORWARD

Esta propiedad se copia en un mensaje que se reenvía.

MQCOPY_PUBLISH

Esta propiedad se copia en el mensaje recibido por un suscriptor cuando se publica un mensaje.

MQCOPY_RESPUESTA

Esta propiedad se copia en un mensaje de respuesta.

MQCOPY_INFORME

Esta propiedad se copia en un mensaje de informe.

MQCOPY_ALL

Esta propiedad se copia en todos los tipos de mensajes posteriores.

Opción predeterminada: Se puede especificar la siguiente opción para proporcionar el conjunto predeterminado de opciones de copia:

MQCOPY_DEFAULT

Esta propiedad se copia en un mensaje reenviado, en un mensaje de informe o en un mensaje recibido por un suscriptor cuando se publica un mensaje.

Esto equivale a especificar la combinación de opciones MQCOPY_FORWARD, más MQCOPY_REPORT, más MQCOPY_PUBLISH.

Si no se requiere ninguna de las opciones descritas anteriormente, utilice la siguiente opción:

MQCOPY_NONE

Utilice este valor para indicar que no se han especificado otras opciones de copia; desde el punto de vista de la programación, no existe ninguna relación entre esta propiedad y los mensajes posteriores. Esto siempre se devuelve para las propiedades del descriptor de mensajes.

Este es un campo de entrada/salida para la llamada MQSETMP y un campo de salida de la llamada MQINQMP. El valor inicial de este campo es MQCOPY_DEFAULT.

MQPMO - Opciones de mensaje de envío

La estructura MQPMO permite a la aplicación especificar opciones que controlan cómo se colocan los mensajes en las colas o cómo se publican en los temas. La estructura es un parámetro de entrada/salida en las llamadas MQPUT y MQPUT1.

Versión

La versión actual de MQPMO es MQPMO_VERSION_3. Algunos campos solo están disponibles en determinadas versiones de MQPMO. Si necesita transferir aplicaciones entre varios entornos, debe asegurarse de que la versión de MQPMO sea la misma en todos los entornos. Los campos que solo existen en versiones concretas de la estructura se identifican como tales en este tema y en las descripciones de los campos.

Los archivos de encabezado, COPY e INCLUDE proporcionados para los lenguajes de programación compatibles contienen la versión más reciente de MQPMO compatible con el entorno, pero con el valor inicial del *Version* campo establecido en MQPMO_VERSION_1. Para utilizar campos que no están presentes en la estructura version-1, la aplicación debe establecer el *Version* campo en el número de versión requerido.

Conjunto de caracteres y codificación

Los datos en MQPMO deben estar en el juego de caracteres proporcionado por el atributo del **CodedCharSetId** gestor de colas y la codificación del gestor de colas local proporcionada por MQENC_NATIVE. Sin embargo, si la aplicación se ejecuta como un cliente MQ MQI, la estructura debe estar en el conjunto de caracteres y la codificación del cliente.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 529. Campos en MQPMO		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>StrucId</u> (identificador de estructura)	MQPMO_STRUC_ID	'PMO~'
<u>Versión</u> (número de versión de la estructura)	MQPMO_VERSION_1	1
<u>Opciones</u> (opciones que controlan la acción de MQPUT y MQPUT1)	MQPMO_NINGUNO	0

Tabla 529. Campos en MQPMO (continuación)

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
Tiempo de espera (reservado)	Ninguna	-1
Contexto (identificador de objeto de la cola de entrada)	Ninguna	0
KnownDestCount (número de mensajes enviados correctamente a las colas locales)	Ninguna	0
UnknownDestCount (número de mensajes enviados correctamente a colas remotas)	Ninguna	0
InvalidDestCount (número de mensajes que no se pudieron enviar)	Ninguna	0
ResolvedQName (nombre resuelto de la cola de destino)	Ninguna	Cadena nula o espacios en blanco
ResolvedQMgrName (nombre resuelto del gestor de colas de destino)	Ninguna	Cadena nula o espacios en blanco
Nota: Los campos restantes se ignoran si <i>Version</i> es menor que MQPMO_VERSION_2.		
RecsPresent (número de registros de mensajes enviados o registros de respuesta presentes)	Ninguna	0
PutMsgRecFields (indicadores que señalan qué campos MQPMR están presentes)	MQPMRF_NONE	0
PutMsgRecOffset (desplazamiento del primer registro de mensaje enviado desde el inicio de MQPMO)	Ninguna	0
ResponseRecOffset (desplazamiento del primer registro de respuesta desde el inicio de MQPMO)	Ninguna	0
PutMsgRecPtr (dirección del primer registro de mensaje enviado)	Ninguna	Puntero nulo o bytes nulos
ResponseRecPtr (dirección del primer registro de respuesta)	Ninguna	Puntero nulo o bytes nulos
Nota: Los campos restantes se ignoran si <i>Version</i> es menor que MQPMO_VERSION_3.		
OriginalMsgHandle (identificador del mensaje original)	MQHM_NINGUNO	0
NewMsgHandle (nuevo identificador de mensaje)	MQHM_NINGUNO	0
Acción (tipo de operación que se está realizando y relación entre el mensaje original especificado por el <i>OriginalMsgHandle</i> campo y el nuevo mensaje especificado por el <i>NewMsgHandle</i> campo)	MQACTP_NEW	0
PubLevel (nivel de suscripción al que se dirige la publicación)	Ninguna	9

Tabla 529. Campos en MQPMO (continuación)

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
Notas: 1. El símbolo ~ representa un único carácter en blanco. 2. El valor «Cadena nula o espacios en blanco» denota la cadena nula en C y los caracteres en blanco en otros lenguajes de programación. 3. En el lenguaje de programación C, la variable macroMQPMO_DEFAULT contiene los valores que se enumeran en la tabla. Úselo de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQPMO MyPMO = {MQPMO_DEFAULT};</pre>		

Declaraciones lingüísticas

Declaración C para MQPMO

```
typedef struct tagMQPMO MQPMO;
struct tagMQPMO {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQLONG     Options;          /* Options that control the action of
                                MQPUT and MQPUT1 */
    MQLONG     Timeout;          /* Reserved */
    MQHOBJS    Context;          /* Object handle of input queue */
    MQLONG     KnownDestCount;   /* Number of messages sent
                                successfully to local queues */
    MQLONG     UnknownDestCount; /* Number of messages sent
                                successfully to remote queues */
    MQLONG     InvalidDestCount; /* Number of messages that could not
                                be sent */
    MQCHAR48    ResolvedQName;   /* Resolved name of destination
                                queue */
    MQCHAR48    ResolvedQMGrName; /* Resolved name of destination queue
                                manager */
    /* Ver:1 */
    MQLONG     RecsPresent;       /* Number of put message records or
                                response records present */
    MQLONG     PutMsgRecFields;   /* Flags indicating which MQPMR fields
                                are present */
    MQLONG     PutMsgRecOffset;   /* Offset of first put message record
                                from start of MQPMO */
    MQLONG     ResponseRecOffset; /* Offset of first response record
                                from start of MQPMO */
    MQPTR      PutMsgRecPtr;      /* Address of first put message
                                record */
    MQPTR      ResponseRecPtr;    /* Address of first response record */
    /* Ver:2 */
    MQHMSG     OriginalMsgHandle; /* Original message handle */
    MQHMSG     NewMsgHandle;       /* New message handle */
    MQLONG     Action;             /* The action being performed */
    MQLONG     PubLevel;          /* Subscription level */
    /* Ver:3 */
};
```

COBOL declaración para MQPMO

```
**      MQPMO structure
10      MQPMO.
**      Structure identifier
15      MQPMO-STRUCID          PIC X(4).
**      Structure version number
15      MQPMO-VERSION          PIC S9(9) BINARY.
**      Options that control the action of MQPUT and MQPUT1
15      MQPMO-OPTIONS          PIC S9(9) BINARY.
**      Reserved
15      MQPMO-TIMEOUT          PIC S9(9) BINARY.
**      Object handle of input queue
```

```

15 MQPMO-CONTEXT          PIC S9(9) BINARY.
** Number of messages sent successfully to local queues
15 MQPMO-KNOWNDESTCOUNT  PIC S9(9) BINARY.
** Number of messages sent successfully to remote queues
15 MQPMO-UNKNOWNDESTCOUNT PIC S9(9) BINARY.
** Number of messages that could not be sent
15 MQPMO-INVALIDDESTCOUNT PIC S9(9) BINARY.
** Resolved name of destination queue
15 MQPMO-RESOLVEDQNAME     PIC X(48).
** Resolved name of destination queue manager
15 MQPMO-RESOLVEDQMGRNAME  PIC X(48).
** Number of put message records or response records present
15 MQPMO-RECSPRESENT       PIC S9(9) BINARY.
** Flags indicating which MQPMR fields are present
15 MQPMO-PUTMSGRECFIELDS   PIC S9(9) BINARY.
** Offset of first put message record from start of MQPMO
15 MQPMO-PUTMSGRECOFFSET   PIC S9(9) BINARY.
** Offset of first response record from start of MQPMO
15 MQPMO-RESPONSERECOFFSET PIC S9(9) BINARY.
** Address of first put message record
15 MQPMO-PUTMSGRECPtr      POINTER.
** Address of first response record
15 MQPMO-RESPONSERECPtr    POINTER.
** Original message handle
15 MQPMO-ORIGINALMSGHANDLE PIC S9(18) BINARY.
** New message handle
15 MQPMO-NEWMSGHANDLE       PIC S9(18) BINARY.
** The action being performed
15 MQPMO-ACTION             PIC S9(9) BINARY.
** Publish level
15 MQPMO-PUBLEVEL          PIC S9(9) BINARY.

```

PL/I declaración para MQPMO

```

dcl
1 MQPMO based,
3 StructId          char(4),          /* Structure identifier */
3 Version           fixed bin(31),    /* Structure version number */
3 Options           fixed bin(31),    /* Options that control the action
                                     of MQPUT and MQPUT1 */
3 Timeout           fixed bin(31),    /* Reserved */
3 Context           fixed bin(31),    /* Object handle of input queue */
3 KnownDestCount    fixed bin(31),    /* Number of messages sent
                                     successfully to local queues */
3 UnknownDestCount  fixed bin(31),    /* Number of messages sent
                                     successfully to remote queues */
3 InvalidDestCount  fixed bin(31),    /* Number of messages that could
                                     not be sent */
3 ResolvedQName     char(48),         /* Resolved name of destination
                                     queue */
3 ResolvedQMgrName  char(48),         /* Resolved name of destination
                                     queue manager */
3 RecsPresent       fixed bin(31),    /* Number of put message records or
                                     response records present */
3 PutMsgRecFields   fixed bin(31),    /* Flags indicating which MQPMR
                                     fields are present */
3 PutMsgRecOffset   fixed bin(31),    /* Offset of first put message
                                     record from start of MQPMO */
3 ResponseRecOffset fixed bin(31),    /* Offset of first response record
                                     from start of MQPMO */
3 PutMsgRecPtr      pointer,          /* Address of first put message
                                     record */
3 ResponseRecPtr     pointer,          /* Address of first response
                                     record */
3 OriginalMsgHandle fixed bin(63),    /* Original message handle */
3 NewMsgHandle       fixed bin(63);   /* New message handle */
3 Action            fixed bin(31);    /* The action being performed */
3 PubLevel          fixed bin(31);    /* Publish level */

```

High Level Assembler declaración para MQPMO

MQPMO	DSECT		
MQPMO_STRUCID	DS	CL4	Structure identifier
MQPMO_VERSION	DS	F	Structure version number
MQPMO_OPTIONS	DS	F	Options that control the action of
*			MQPUT and MQPUT1
MQPMO_TIMEOUT	DS	F	Reserved
MQPMO_CONTEXT	DS	F	Object handle of input queue
MQPMO_KNOWNDESTCOUNT	DS	F	Number of messages sent successfully

```

*
MQPMO_UNKNOWNDSTCOUNT DS F to local queues
*
MQPMO_INVALIDDSTCOUNT DS F Number of messages sent successfully
*
MQPMO_UNKNOWNDSTCOUNT DS F to remote queues
*
MQPMO_INVALIDDSTCOUNT DS F Number of messages that could not be
*
MQPMO_INVALIDDSTCOUNT DS F sent
*
MQPMO_RESOLVEDQNAME DS CL48 Resolved name of destination queue
MQPMO_RESOLVEDQMGRNAME DS CL48 Resolved name of destination queue
*
MQPMO_RECSPRESENT DS F manager
*
MQPMO_RECSPRESENT DS F Number of put message records or
*
MQPMO_RECSPRESENT DS F response records present
*
MQPMO_PUTMSGRECFIELDS DS F Flags indicating which MQPMR
*
MQPMO_PUTMSGRECFIELDS DS F fields are present
*
MQPMO_PUTMSGRECOFFSET DS F Offset of first put message record
*
MQPMO_PUTMSGRECOFFSET DS F from start of MQPMO
*
MQPMO_RESPONSERECOFFSET DS F Offset of first response record
*
MQPMO_RESPONSERECOFFSET DS F from start of MQPMO
*
MQPMO_PUTMSGRECPTTR DS F Address of first put message
*
MQPMO_PUTMSGRECPTTR DS F record
*
MQPMO_RESPONSERECPTTR DS F Address of first response record
MQPMO_ORIGINALMSGHANDLE DS D Original message handle
MQPMO_NEWMSGHANDLE DS D New message handle
MQPMO_ACTION DS F The action being performed
MQPMO_PUBLEVEL DS F Publish level
*
MQPMO_LENGTH EQU *-MQPMO
ORG MQPMO
MQPMO_AREA DS CL(MQPMO_LENGTH)

```

Declaración de Visual Basic para MQPMO

```

Type MQPMO
    StrucId As String*4 'Structure identifier'
    Version As Long 'Structure version number'
    Options As Long 'Options that control the action of'
    'MQPUT and MQPUT1'
    Timeout As Long 'Reserved'
    Context As Long 'Object handle of input queue'
    KnownDestCount As Long 'Number of messages sent successfully'
    'to local queues'
    UnknownDestCount As Long 'Number of messages sent successfully'
    'to remote queues'
    InvalidDestCount As Long 'Number of messages that could not be'
    'sent'
    ResolvedQName As String*48 'Resolved name of destination queue'
    ResolvedQMgrName As String*48 'Resolved name of destination queue'
    'manager'
    RecsPresent As Long 'Number of put message records or'
    'response records present'
    PutMsgRecFields As Long 'Flags indicating which MQPMR fields'
    'are present'
    PutMsgRecOffset As Long 'Offset of first put message record'
    'from start of MQPMO'
    ResponseRecOffset As Long 'Offset of first response record from'
    'start of MQPMO'
    PutMsgRecPtr As MQPTR 'Address of first put message record'
    ResponseRecPtr As MQPTR 'Address of first response record'
End Type

```

Detalles de campo para MQPMO

La estructura de opciones de mensajes MQPMO Put permite a la aplicación especificar opciones que controlan cómo se colocan los mensajes en las colas o se publican en los temas. La estructura es un parámetro de entrada/salida en las llamadas MQPUT y MQPUT1.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 530. Campos en MQPMO		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
StrucId (identificador de estructura)	MQPMO_STRUC_ID	'PMO-'

Tabla 530. Campos en MQPMO (continuación)

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>Versión</u> (número de versión de la estructura)	MQPMO_VERSION_1	1
<u>Opciones</u> (opciones que controlan la acción de MQPUT y MQPUT1)	MQPMO_NINGUNO	0
<u>Tiempo de espera</u> (reservado)	Ninguna	-1
<u>Contexto</u> (identificador de objeto de la cola de entrada)	Ninguna	0
<u>KnownDestCount</u> (número de mensajes enviados correctamente a las colas locales)	Ninguna	0
<u>UnknownDestCount</u> (número de mensajes enviados correctamente a colas remotas)	Ninguna	0
<u>InvalidDestCount</u> (número de mensajes que no se pudieron enviar)	Ninguna	0
<u>ResolvedQName</u> (nombre resuelto de la cola de destino)	Ninguna	Cadena nula o espacios en blanco
<u>ResolvedQMgrName</u> (nombre resuelto del gestor de colas de destino)	Ninguna	Cadena nula o espacios en blanco
Nota: Los campos restantes se ignoran si <i>Version</i> es menor que MQPMO_VERSION_2.		
<u>RecsPresent</u> (número de registros de mensajes enviados o registros de respuesta presentes)	Ninguna	0
<u>PutMsgRecFields</u> (indicadores que señalan qué campos MQPMR están presentes)	MQPMRF_NONE	0
<u>PutMsgRecOffset</u> (desplazamiento del primer registro de mensaje enviado desde el inicio de MQPMO)	Ninguna	0
<u>ResponseRecOffset</u> (desplazamiento del primer registro de respuesta desde el inicio de MQPMO)	Ninguna	0
<u>PutMsgRecPtr</u> (dirección del primer registro de mensaje enviado)	Ninguna	Puntero nulo o bytes nulos
<u>ResponseRecPtr</u> (dirección del primer registro de respuesta)	Ninguna	Puntero nulo o bytes nulos
Nota: Los campos restantes se ignoran si <i>Version</i> es menor que MQPMO_VERSION_3.		
<u>OriginalMsgHandle</u> (identificador del mensaje original)	MQHM_NINGUNO	0
<u>NewMsgHandle</u> (nuevo identificador de mensaje)	MQHM_NINGUNO	0
<u>Acción</u> (tipo de operación que se está realizando y relación entre el mensaje original especificado por el <i>OriginalMsgHandle</i> campo y el nuevo mensaje especificado por el <i>NewMsgHandle</i> campo)	MQACTP_NEW	0

Tabla 530. Campos en MQPMO (continuación)

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
PubLevel (nivel de suscripción al que se dirige la publicación)	Ninguna	9
Notas: 1. El símbolo ~ representa un único carácter en blanco. 2. El valor «Cadena nula o espacios en blanco» denota la cadena nula en C y los caracteres en blanco en otros lenguajes de programación. 3. En el lenguaje de programación C, la variable macroMQPMO_DEFAULT contiene los valores que se enumeran en la tabla. Úselo de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQPMO MyPMO = {MQPMO_DEFAULT};</pre>		

StrucId (MQCHAR4)

Este es el identificador de estructura de la estructura de opciones del mensaje put. Siempre es un campo de entrada. Su valor es MQPMO_STRUC_ID.

El valor debe ser:

MQPMO_STRUC_ID

Identificador de la estructura de opciones del mensaje enviado.

Para el lenguaje de programación C, también se define la constante MQPMO_STRUC_ID_ARRAY. Tiene el mismo valor que MQPMO_STRUC_ID, pero es una matriz de caracteres en lugar de una cadena.

Versión (MQLONG)

Número de versión de la estructura.

El valor debe ser uno de los siguientes:

MQPMO_VERSION_1

Version-1 Estructura de opciones de put-message.

Esta versión es compatible con todos los entornos.

MQPMO_VERSION_2

Version-2 Estructura de opciones de put-message.

Esta versión es compatible con los siguientes entornos:

-  AIX
-  IBM i
-  Linux
-  Windows

y para IBM MQ MQI clients conectados a estos sistemas.

MQPMO_VERSION_3

Version-3 Estructura de opciones de put-message.

Esta versión es compatible con todos los entornos.

Los campos que solo existen en la versión más reciente de la estructura se identifican como tales en las descripciones de los campos. La siguiente constante especifica el número de versión de la versión actual:

MQPMO_VERSION_ACTUAL

Versión actual de la estructura de opciones de put-message.

Este campo siempre es un campo de entrada. El valor inicial de este campo es MQPMO_VERSION_1.

Opciones (MQLONG)

El campo Opciones controla el funcionamiento de las llamadas **MQPUTMQPUT1** y.

Opción de alcance

Puede especificar cualquiera de las opciones MQPMO o ninguna. Para especificar más de una opción, suma los valores (sin añadir la misma constante más de una vez) o combínalos mediante la operación OR bit a bit (si el lenguaje de programación admite operaciones bit a bit). Las combinaciones que no son válidas se indican; cualquier otra combinación es válida.

La siguiente opción controla el alcance de las publicaciones enviadas:

MQPMO_ÁMBITO_QMGR

La publicación solo se envía a los suscriptores que se han suscrito en este gestor de colas. La publicación no se reenvía a ningún gestor de colas de publicación/suscripción remoto que se haya suscrito a este gestor de colas, lo que anula cualquier comportamiento que se haya establecido mediante el atributo de tema PUBSCOPE.

Nota: Si no se establece, el ámbito de publicación viene determinado por el atributo de tema PUBSCOPE.

Opciones de publicación

Las siguientes opciones controlan la forma en que se publican los mensajes en un tema:

MQPMO_SUPPRIMIR_RESPUESTA

La información especificada en los campos *ReplyToQ* *ReplyToQMGr* y del MQMD de esta publicación no se transmite a los suscriptores. Si esta opción se utiliza con una opción de informe que requiere un *ReplyToQ*, la llamada falla con MQRC_MISSING_REPLY_TO_Q.

MQPMO_RETENER

La publicación que se envía debe ser conservada por el gestor de colas. Esta retención permite a un suscriptor solicitar una copia de esta publicación después de su publicación, utilizando la llamada MQSUBRQ. También permite enviar una publicación a las aplicaciones que se suscriben después de la fecha de publicación (a menos que elijan no recibirla utilizando la opción MQSO_NEW_PUBLICATIONS_ONLY). Si se envía una solicitud a una publicación que se ha retenido, esto se indica mediante la propiedad del mensaje « MQIsRetained » (Solicitante de publicación retenida) de dicha publicación.

Solo se puede conservar una publicación en cada nodo del árbol de temas. Por lo tanto, si ya existe una publicación retenida para este tema, publicada por cualquier otra aplicación, se sustituye por esta publicación. Por lo tanto, es mejor evitar que haya más de un editor que conserve mensajes sobre el mismo tema.

Cuando un suscriptor solicita publicaciones retenidas, la suscripción utilizada puede contener un comodín en el tema, en cuyo caso varias publicaciones retenidas pueden coincidir (en varios nodos del árbol de temas) y varias publicaciones pueden enviarse a la aplicación solicitante. Consulte la descripción de la [“MQSUBRQ - Solicitud de suscripción”](#) en la [página 797](#) convocatoria para obtener más detalles.

Para obtener información sobre cómo interactúan las publicaciones retenidas con los niveles de suscripción, consulte [Interceptar publicaciones](#).

Si se utiliza esta opción y la publicación no se puede retener, el mensaje no se publica y la llamada falla con MQRC_PUT_NOT_RETAINED.

MQPMO_NOT_OWN_SUBS

Indica al gestor de colas que la aplicación no desea enviar ninguna de sus publicaciones a las suscripciones que posee. Las suscripciones se consideran propiedad de la misma aplicación si los identificadores de conexión son los mismos.

MQPMO_ADVERTIR_SI_NO_HAY_SUSCRIPCIONES_COINCIDENTES

Si ninguna suscripción coincide con la publicación, devuelva un código de finalización (*CompCode*) de MQCC_WARNING y un código de motivo MQRC_NO_SUBS_MATCHED.

Si la operación put devuelve MQRC_NO_SUBS_MATCHED, la publicación no se ha entregado a ninguna suscripción. Sin embargo, si se especifica la opción MQPMO_RETAIN en la operación de envío, el mensaje se retiene y se entrega a cualquier suscripción coincidente definida posteriormente.

Una suscripción sobre el tema coincide con la publicación si se cumple alguna de las siguientes condiciones:

- El mensaje se envía a la cola de suscripción
- El mensaje se habría entregado a la cola de suscripción, pero un problema con la cola impide que el mensaje se coloque en ella, por lo que se ha colocado en la cola de mensajes no entregados o se ha descartado.
- Se define una salida de enrutamiento que suprime la entrega del mensaje a la suscripción

Una suscripción sobre el tema no coincide con la publicación si se cumple alguna de las siguientes condiciones:

- La suscripción tiene una cadena de selección que no coincide con la publicación
- La suscripción especificaba la opción MQSO_PUBLICATION_ON_REQUEST
- La publicación no se entrega porque se especificó la opción MQPMO_NOT_OWN_SUBS en la operación de envío y la suscripción coincide con la identidad del editor

Opciones de Syncpoint

Las siguientes opciones se refieren a la participación de la llamada MQPUT o MQPUT1 dentro de una unidad de trabajo:

MQPMO_PUNTO_DE_SINCRONIZACIÓN

La solicitud es operar dentro de los protocolos normales de unidad de trabajo. El mensaje no es visible fuera de la unidad de trabajo hasta que esta se haya confirmado. Si se cancela la unidad de trabajo, se elimina el mensaje.

Si no se especifican MQPMO_SYNCPOINT y MQPMO_NO_SYNCPOINT, la inclusión de la solicitud de inserción en los protocolos de unidad de trabajo viene determinada por el entorno en el que se ejecuta el gestor de colas y no por el entorno en el que se ejecuta la aplicación. En z/OS, la solicitud de puesta se encuentra dentro de una unidad de trabajo. En todos los demás entornos, la solicitud put no se encuentra dentro de una unidad de trabajo.

Debido a estas diferencias, una aplicación que desee portar no debe permitir que esta opción sea predeterminada; especifique explícitamente MQPMO_SYNCPOINT o MQPMO_NO_SYNCPOINT.

No especifique MQPMO_SYNCPOINT con MQPMO_NO_SYNCPOINT.

MQPMO_NO_PUNTO_DE_SINCRONIZACIÓN

La solicitud es operar fuera de los protocolos normales de unidad de trabajo. El mensaje está disponible de inmediato y no se puede eliminar revirtiendo una unidad de trabajo.

Si no se especifican MQPMO_NO_SYNCPOINT y MQPMO_SYNCPOINT, la inclusión de la solicitud de inserción en los protocolos de unidad de trabajo viene determinada por el entorno en el que se ejecuta el gestor de colas y no por el entorno en el que se ejecuta la aplicación. En z/OS, la solicitud de puesta se encuentra dentro de una unidad de trabajo. En todos los demás entornos, la solicitud put no se encuentra dentro de una unidad de trabajo.

Debido a estas diferencias, una aplicación que desee portar no debe permitir que esta opción sea predeterminada; especifique explícitamente MQPMO_SYNCPOINT o MQPMO_NO_SYNCPOINT.

No especifique MQPMO_NO_SYNCPOINT con MQPMO_SYNCPOINT.

Opciones de identificador de mensaje e identificador de correlación

Las siguientes opciones solicitan al gestor de colas que genere un nuevo identificador de mensaje o identificador de correlación:

MQPMO_NUEVO_ID_MENSAJE

El gestor de colas sustituye el contenido del *MsgId* campo en MQMD por un nuevo identificador de mensaje. Este identificador de mensaje se envía con el mensaje y se devuelve a la aplicación en la salida de la llamada MQPUT o MQPUT1.

La opción MQPMO_NEW_MSG_ID también se puede especificar cuando el mensaje se incluye en una lista de distribución; consulte la descripción del *MsgId* campo en la estructura MQPMR para obtener más detalles.

El uso de esta opción libera a la aplicación de la necesidad de restablecer el *MsgId* campo a MQMI_NONE antes de cada llamada MQPUT o MQPUT1.

MQPMO_NEW_CORREL_ID

El gestor de colas sustituye el contenido del *CorrelId* campo en MQMD por un nuevo identificador de correlación. Este identificador de correlación se envía con el mensaje y se devuelve a la aplicación en la salida de la llamada MQPUT o MQPUT1.

La opción MQPMO_NEW_CORREL_ID también se puede especificar cuando el mensaje se envía a una lista de distribución; consulte la descripción del *CorrelId* campo en la estructura MQPMR para obtener más detalles.

MQPMO_NEW_CORREL_ID es útil en situaciones en las que la aplicación requiere un identificador de correlación único.

Opciones de grupo y segmento

Las siguientes opciones se refieren al procesamiento de mensajes en grupos y segmentos de mensajes lógicos. Lea las definiciones que siguen para ayudarle a comprender la opción.



Atención: No puede utilizar los mensajes segmentados ni agrupados con la publicación/suscripción.

Mensaje físico

Es la unidad más pequeña de información que se puede colocar o eliminar de una cola; a menudo corresponde a la información especificada o recuperada en una sola llamada MQPUT, MQPUT1 o MQGET. Cada mensaje físico tiene su propio descriptor de mensaje (MQMD). Por lo general, los mensajes físicos se distinguen por valores diferentes para el identificador de mensaje (*MsgId* campo en MQMD), aunque esto no lo impone el gestor de colas.

Mensaje lógico

Un mensaje lógico es una unidad única de información de aplicación solo para plataformas z/OS no. En ausencia de restricciones del sistema, un mensaje lógico es lo mismo que un mensaje físico. Sin embargo, cuando los mensajes lógicos son extremadamente grandes, las limitaciones del sistema pueden hacer que sea aconsejable o necesario dividir un mensaje lógico en dos o más mensajes físicos, denominados *segmentos*.

Un mensaje lógico que ha sido segmentado consta de dos o más mensajes físicos que tienen el mismo identificador de grupo no nulo (*GroupId* campo en MQMD) y el mismo número de secuencia de mensajes (*MsgSeqNumber* campo en MQMD). Los segmentos se distinguen por valores diferentes para el desplazamiento del segmento (*Offset* campo en MQMD), que proporciona el desplazamiento de los datos en el mensaje físico desde el inicio de los datos en el mensaje lógico. Dado que cada segmento es un mensaje físico, los segmentos de un mensaje lógico suelen tener identificadores de mensaje diferentes.

Un mensaje lógico que no ha sido segmentado, pero cuya segmentación ha sido permitida por la aplicación emisora, también tiene un identificador de grupo distinto de nulo, aunque en este caso solo hay un mensaje físico con ese identificador de grupo si el mensaje lógico no pertenece a un grupo de

mensajes. Los mensajes lógicos cuya segmentación ha sido inhibida por la aplicación emisora tienen un identificador de grupo nulo (MQGI_NONE), a menos que el mensaje lógico pertenezca a un grupo de mensajes.

Grupo de mensajes

Un grupo de mensajes es un conjunto de uno o más mensajes lógicos que tienen el mismo identificador de grupo no nulo. Los mensajes lógicos del grupo se distinguen por valores diferentes del número de secuencia del mensaje, que es un número entero comprendido entre 1 y n , donde n es el número de mensajes lógicos del grupo. Si uno o más de los mensajes lógicos están segmentados, hay más de n mensajes físicos en el grupo.

MQPMO_LOGICAL_ORDER

Esta opción indica al gestor de colas cómo la aplicación agrupa los mensajes en segmentos de mensajes lógicos. Sólo se puede especificar en la llamada MQPUT; no es válida en la llamada MQPUT1.

Si MQPMO_LOGICAL_ORDER se especifica, esto indica que la aplicación utiliza llamadas sucesivas a MQPUT para:

1. Poner los segmentos de cada mensaje lógico en el orden de desplazamiento de segmento creciente, empezando desde 0, sin huecos.
2. Poner todos los segmentos en un mensaje lógico antes de poner los segmentos en el siguiente mensaje lógico.
3. Poner los mensajes lógicos de cada grupo de mensajes en el orden de número de secuencia de mensaje creciente, empezando desde 1, sin huecos. IBM MQ incrementa el número de secuencia de mensaje automáticamente.
4. Poner todos los mensajes lógicos en un grupo de mensajes antes de poner los mensajes lógicos en el siguiente grupo de mensajes.

Para obtener información detallada sobre MQPMO_LOGICAL_ORDER, consulte [Orden lógico y físico](#).

Opciones de contexto

Las siguientes opciones controlan el procesamiento del contexto del mensaje:

MQPMO_NO_CONTEXT

Tanto el contexto de identidad como el de origen están configurados para indicar que no hay contexto. Esto significa que los campos de contexto en MQMD se establecen en:

- Espacios en blanco para campos de caracteres
- Valores nulos para campos de bytes
- Ceros para campos numéricos

MQPMO_DEFAULT_CONTEXT

El mensaje debe tener asociada información contextual predeterminada, tanto sobre la identidad como sobre el origen. El gestor de colas establece los campos de contexto en el descriptor de mensajes de la siguiente manera:

Tabla 531. Valores predeterminados de la información contextual para los campos MQMD

Campo de MQMD	Valor utilizado
<i>UserIdentifier</i>	Determinado a partir del entorno si es posible; establecido en blanco en caso contrario.
<i>AccountingToken</i>	Determinado por el entorno si es posible; en caso contrario, establecido en MQACT.
<i>ApplIdentityData</i>	Establecido en blancos.
<i>PutApplType</i>	Determinado por el entorno.
<i>PutApplName</i>	Determinado a partir del entorno si es posible; establecido en blanco en caso contrario.

Tabla 531. Valores predeterminados de la información contextual para los campos MQMD (continuación)

Campo de MQMD	Valor utilizado
<i>PutDate</i>	Establecer la fecha en la que se envía el mensaje.
<i>PutTime</i>	Establecer la hora en la que se envía el mensaje.
<i>AppOriginData</i>	Establecido en blancos.

Para obtener más información sobre el contexto del mensaje, consulte [Contexto de mensaje](#).

Estos son los valores y acciones predeterminados si no se especifican opciones de contexto.

MQPMO_PASS_IDENTITY_CONTEXT

El mensaje debe tener información contextual asociada. El contexto de identidad se toma del identificador de cola especificado en el *Context* campo. La información del contexto de origen es generada por el gestor de colas de la misma manera que para MQPMO_DEFAULT_CONTEXT (véase la tabla anterior para los valores). Para obtener más información sobre el contexto del mensaje, consulte [Contexto de mensaje](#).

Para la llamada MQPUT, la cola debe haberse abierto con la opción MQOO_PASS_IDENTITY_CONTEXT (o una opción que la implique). Para la llamada `MQPUT1`, se lleva a cabo la misma comprobación de autorización que para la llamada `MQOPEN` con la opción `MQOO\_PASS\_IDENTITY\_CONTEXT`.

MQPMO_PASS_ALL_CONTEXT

El mensaje debe tener información contextual asociada. El contexto se toma del identificador de cola especificado en el *Context* campo. Para obtener más información sobre el contexto de los mensajes, consulte [Control de la información de contexto](#).

Para la llamada MQPUT, la cola debe haberse abierto con la opción MQOO_PASS_ALL_CONTEXT (o una opción que la implique). Para la llamada `MQPUT1`, se lleva a cabo la misma comprobación de autorización que para la llamada `MQOPEN` con la opción `MQOO\_PASS\_ALL\_CONTEXT`.

MQPMO_SET_IDENTITY_CONTEXT

El mensaje debe tener información contextual asociada. La aplicación especifica el contexto de identidad en la estructura MQMD. La información del contexto de origen es generada por el gestor de colas de la misma manera que para MQPMO_DEFAULT_CONTEXT (véase la tabla anterior para los valores). Para obtener más información sobre el contexto del mensaje, consulte [Contexto de mensaje](#).

Para la llamada MQPUT, la cola debe haberse abierto con la opción MQOO_SET_IDENTITY_CONTEXT (o una opción que la implique). Para la llamada `MQPUT1`, se lleva a cabo la misma comprobación de autorización que para la llamada `MQOPEN` con la opción `MQOO\_SET\_IDENTITY\_CONTEXT`.

MQPMO_SET_ALL_CONTEXT

El mensaje debe tener información contextual asociada. La aplicación especifica la identidad, el origen y el contexto del usuario en la estructura MQMD. Para obtener más información sobre el contexto del mensaje, consulte [Contexto de mensaje](#).

Para la llamada MQPUT, la cola debe haberse abierto con la opción MQOO_SET_ALL_CONTEXT. Para la llamada `MQPUT1`, se lleva a cabo la misma comprobación de autorización que para la llamada `MQOPEN` con la opción `MQOO\_SET\_ALL\_CONTEXT`.

Solo se puede especificar una de las opciones de contexto MQPMO_*_CONTEXT. Si no se especifica ninguno, se asume MQPMO_DEFAULT_CONTEXT.

Opciones de propiedad

La siguiente opción se refiere a las propiedades del mensaje:

MQPMO_MD_SOLO_PARA_SALIDA

El parámetro descriptor de mensaje solo debe utilizarse para la salida con el fin de devolver el descriptor de mensaje del mensaje que se ha introducido. Los campos descriptors de mensajes asociados con los campos *NewMsgHandle*, *OriginalMsgHandle*, o ambos, de la MQPMO estructura deben utilizarse para la entrada.

Si no se proporciona un identificador de mensaje válido, la llamada falla con el código de motivo **MQRC_MD_ERROR**.

Opciones de respuesta

Las siguientes opciones controlan la respuesta devuelta a una llamada MQPUT o MQPUT1. Solo puede especificar una de estas opciones. Si no se especifican MQPMO_ASYNC_RESPONSE y MQPMO_SYNC_RESPONSE, se asume MQPMO_RESPONSE_AS_Q_DEF o MQPMO_RESPONSE_AS_TOPIC_DEF.

MQPMO_RESPUESTA_ASÍNCRONA

La opción MQPMO_ASYNC_RESPONSE solicita que una operación MQPUT o MQPUT1 se complete sin que la aplicación espere a que el gestor de colas complete la llamada. El uso de esta opción puede mejorar el rendimiento de la mensajería, especialmente en aplicaciones que utilizan enlaces de cliente. Una aplicación puede comprobar periódicamente, utilizando el verbo MQSTAT, si se ha producido algún error durante llamadas asíncronas anteriores.

Con esta opción, solo se garantiza que se completen los siguientes campos en el MQMD:

- ApplIdentityData
- PutApplType
- PutApplName
- ApplOriginData

Además, si se especifican como opciones MQPMO_NEW_MSG_ID o MQPMO_NEW_CORREL_ID, también se completan los campos MsgId y CorrelId que se devuelven. (MQPMO_NEW_MSG_ID se puede especificar implícitamente indicando un campo MsgId en blanco).

Solo se completan los campos especificados anteriormente. El resto de información que normalmente se devolvería en la estructura MQMD o MQPMO no está definida.

Al solicitar una respuesta asíncrona para MQPUT1, los campos ResolvedQName y ResolvedQMgrName devueltos en la estructura MQOD no están definidos.

Cuando se solicita una respuesta de transferencia asíncrona para MQPUT o MQPUT1, un código CompCode y una razón MQCC_OK y MQRC_NONE no significan necesariamente que el mensaje se haya transferido correctamente a una cola. Al desarrollar una aplicación de MQI que utilice respuestas «put» asíncronas y requiera la confirmación de que los mensajes se han enviado a una cola, debes comprobar tanto los códigos de estado (CompCode) como los códigos de motivo (Reason) de las operaciones «put», y utilizar también MQSTAT para consultar la información sobre errores asíncronos.

Aunque el éxito o el fracaso de cada llamada individual a MQPUT o MQPUT1 puede que no se devuelva inmediatamente, el primer error que se produjo en una llamada asíncrona se puede determinar más tarde mediante una llamada a MQSTAT.

Si un mensaje persistente en syncpoint no se entrega mediante una respuesta asíncrona put y se intenta confirmar la transacción, la confirmación falla y la transacción se revierte con un código de finalización MQCC_FAILED y un motivo MQRC_BACKED_OUT. La aplicación puede realizar una llamada a MQSTAT para determinar la causa de un fallo anterior de MQPUT o MQPUT1.

Si un mensaje no persistente bajo syncpoint no se entrega utilizando una respuesta asíncrona put y se intenta confirmar la transacción, la confirmación se realiza correctamente, pero el mensaje no persistente se pierde. Para evitar esta situación, la aplicación debe realizar una llamada a MQSTAT para determinar si ha habido algún fallo anterior de MQPUT o MQPUT1 antes de llamar a MQCMIT. Si se produjeran fallos, la aplicación debería llamar a MQBACK.

MQPMO_SYNC_RESPUESTA

Especificar este tipo de respuesta garantiza que la operación MQPUT o MQPUT1 siempre se emita de forma sincrónica. Si la operación de inserción se realiza correctamente, se completan todos los campos de MQMD y MQPMO.

Esta opción garantiza una respuesta sincrónica independientemente del valor de respuesta predeterminado definido en el objeto de cola o tema.

MQPMO_RESPUESTA_COMO_Q_DEF

Si se especifica este valor para una llamada MQPUT, el tipo de respuesta de inserción utilizado se toma del valor **DEFPRESP** especificado en la cola cuando fue abierta por primera vez por la aplicación.

- Si la cola es una cola de clúster y este valor se especifica para una llamada MQPUT, el tipo de respuesta de inserción utilizado se toma del **DEFPRESP** atributo definido en el gestor de colas *de destino* que posee la instancia concreta de la cola en la que se coloca el mensaje.

Cuando hay varias instancias de la cola de clúster y difieren en este atributo, se seleccionará el valor de uno de ellos y no se puede predecir cuál de ellos se utilizará. Por lo tanto debe establecer este atributo en el mismo valor en todas las instancias. Si este no es el caso, se envía el mensaje de error AMQ9407 a los registros del gestor de colas. Consulte también [¿Cómo se resuelven los atributos de objeto de destino para las colas alias, remotas y de clúster?](#)

- Si la cola no es una cola de clúster y se especifica este valor para una llamada MQPUT, el tipo de respuesta de inserción utilizado se toma del **DEFPRESP** atributo definido en el gestor de colas *local*, incluso si el gestor de colas de destino es remoto.

Si una aplicación cliente está conectada a un gestor de colas en un nivel anterior a IBM WebSphere MQ 7.0, se comporta como si se hubiera especificado MQPMO_SYNC_RESPONSE.

Si se especifica esta opción para una llamada MQPUT1, el valor del atributo DEFPRESP no se conoce antes de enviar la solicitud al servidor. De forma predeterminada, si la llamada `MQPUT1` utiliza `MQPMO\_SYNCPOINT`, se comporta como `MQPMO\_ASYNC\_RESPONSE`, y si utiliza `MQPMO\_NO\_SYNCPOINT`, se comporta como `MQPMO\_SYNC\_RESPONSE`. Sin embargo, puede anular este comportamiento predeterminado configurando la propiedad Put1DefaultAlwaysSync en el archivo de configuración del cliente. Consulte [la sección CHANNELS del archivo de configuración del cliente](#).

MQPMO_RESPUESTA_COMO_TEMA_DEF

MQPMO_RESPONSE_AS_TOPIC_DEF es un sinónimo de MQPMO_RESPONSE_AS_Q_DEF para su uso con objetos de tema.

Otras opciones

Las siguientes opciones controlan la comprobación de autorizaciones, lo que ocurre cuando el gestor de colas está inactivo y la resolución de nombres de colas y gestores de colas:

MQPMO_ALTERNATE_USER_AUTHORITY

MQPMO_ALTERNATE_USER_AUTHORITY indica que el *AlternateUserId* campo del **ObjDesc** parámetro de la llamada MQPUT1 contiene un identificador de usuario que se utilizará para validar la autoridad para colocar mensajes en la cola. La llamada solo puede tener éxito si *AlternateUserId* está autorizado para abrir la cola con las opciones especificadas, independientemente de si el identificador de usuario con el que se ejecuta la aplicación está autorizado para hacerlo. (Sin embargo, esto no se aplica a las opciones de contexto especificadas, que siempre se comparan con el identificador de usuario bajo el cual se ejecuta la aplicación)

Esta opción solo es válida con la llamada MQPUT1.

MQPMO_FAIL_IF QUIESCING

Esta opción fuerza el fallo de la llamada MQPUT o MQPUT1 si el gestor de colas se encuentra en estado de inactividad.

En z/OS, esta opción también fuerza el fallo de la llamada MQPUT o MQPUT1 si la conexión (para una CICS o IMS aplicación) se encuentra en estado de inactividad.

La llamada devuelve el código de finalización MQCC_FAILED con el código de motivo MQRC_Q_MGR QUIESCING o MQRC_CONNECTION QUIESCING.

MQPMO_RESOLVE_LOCAL_Q

Utilice esta opción para rellenar *ResolvedQName* la estructura MQPMO con el nombre de la cola local en la que se coloca el mensaje y *ResolvedQMGrName* con el nombre del gestor de colas local que aloja la cola local. Para obtener más información sobre MQPMO_RESOLVE_LOCAL_Q, consulte el tema [MQOO_RESOLVE_LOCAL_Q](#).

Si está autorizado para poner en cola, tiene la autoridad necesaria para especificar este indicador en la llamada MQPUT; no se necesita ninguna autoridad especial.

Opción predeterminada

Si no necesita ninguna de las opciones descritas, utilice la siguiente opción:

MQPMO_NINGUNO

Utilice este valor para indicar que no se han especificado otras opciones; todas las opciones toman sus valores predeterminados. MQPMO_NONE se define para facilitar la documentación del programa; no está previsto que esta opción se utilice con ninguna otra, pero como su valor es cero, dicho uso no puede detectarse.

MQPMO_NONE es un campo de entrada. El valor inicial del campo *Options* es MQPMO_NONE.

Tiempo de espera (MQLONG)

Este es un campo reservado; su valor no es significativo. El valor inicial de este campo es -1.

Contexto (MQHOBJ)

Si se especifica MQPMO_PASS_IDENTITY_CONTEXT o MQPMO_PASS_ALL_CONTEXT, este campo debe contener el identificador de la cola de entrada desde la que se toma la información de contexto que se asociará con el mensaje que se va a enviar.

Si no se especifica ni MQPMO_PASS_IDENTITY_CONTEXT ni MQPMO_PASS_ALL_CONTEXT, este campo se ignora.

Este es un campo de entrada. El valor inicial de este campo es 0.

KnownDestCount (MQLONG)

Este es el número de mensajes que la llamada actual MQPUT o MQPUT1 ha enviado correctamente a las colas de la lista de distribución que son colas locales. El recuento no incluye los mensajes enviados a colas que se resuelven en colas remotas (aunque inicialmente se utilice una cola de transmisión local para almacenar el mensaje). Este campo también se establece cuando se envía un mensaje a una cola única que no está en una lista de distribución.

Se trata de un campo de salida. El valor inicial de este campo es 0. Este campo no se establece si *Version* es menor que MQPMO_VERSION_1.

 Este campo no está definido en z/OS porque no se admiten listas de distribución.

UnknownDestCount (MQLONG)

Este es el número de mensajes que la llamada MQPUT o MQPUT1 actual ha enviado correctamente a las colas de la lista de distribución que se resuelven en colas remotas. Los mensajes que el administrador de colas retiene temporalmente en forma de lista de distribución cuentan como el número de destinos individuales que contienen esas listas de distribución. Este campo también se establece cuando se envía un mensaje a una cola única que no está en una lista de distribución.

Se trata de un campo de salida. El valor inicial de este campo es 0. Este campo no se establece si *Version* es menor que MQPMO_VERSION_1.

 Este campo no está definido en z/OS porque no se admiten listas de distribución.

InvalidDestCount (MQLONG)

Este es el número de mensajes que no se pudieron enviar a las colas de la lista de distribución. El recuento incluye las colas que no se pudieron abrir, así como las colas que se abrieron correctamente pero en las que falló la operación de inserción. Este campo también se establece cuando se envía un mensaje a una cola única que no está en una lista de distribución.

Nota: Este campo se establece si el **CompCode** parámetro en la llamada MQPUT o MQPUT1 es MQCC_OK o MQCC_WARNING; podría establecerse si el **CompCode** parámetro es MQCC_FAILED, pero no confíe en ello en el código de la aplicación.

Se trata de un campo de salida. El valor inicial de este campo es 0. Este campo no se establece si *Version* es menor que MQPMO_VERSION_1.

 Este campo no está definido en z/OS porque no se admiten listas de distribución.

ResolvedQName (MQCHAR48)

Este es el nombre de la cola de destino después de que el administrador de colas local haya realizado la resolución de nombres. El nombre devuelto es el nombre de una cola que existe en el gestor de colas identificado por *ResolvedQMgrName*.

Solo se devuelve un valor distinto de blanco si el objeto es una cola única; si el objeto es una lista de distribución o un tema, el valor devuelto es indefinido.

Se trata de un campo de salida. La longitud de este campo viene determinada por MQ_Q_NAME_LENGTH. El valor inicial de este campo es la cadena nula en C y 48 caracteres en blanco en otros lenguajes de programación.

ResolvedQMgrName (MQCHAR48)

Este es el nombre del gestor de colas de destino después de que el gestor de colas local haya realizado la resolución de nombres. El nombre devuelto es el nombre del gestor de colas propietario de la cola identificada por *ResolvedQName*, y puede ser el nombre del gestor de colas local.

Si *ResolvedQName* es una cola compartida que pertenece al grupo de uso compartido de colas al que pertenece el gestor de colas local, *ResolvedQMgrName* es el nombre del grupo de uso compartido de colas. Si la cola pertenece a otro grupo de uso compartido de colas, *ResolvedQName* puede ser el nombre del grupo de uso compartido de colas o el nombre de un gestor de colas que sea miembro del grupo de uso compartido de colas (la naturaleza del valor devuelto viene determinada por las definiciones de colas que existen en el gestor de colas local).

Solo se devuelve un valor distinto de blanco si el objeto es una cola única; si el objeto es una lista de distribución o un tema, el valor devuelto es indefinido.

Se trata de un campo de salida. La longitud de este campo viene determinada por MQ_Q_MGR_NAME_LENGTH. El valor inicial de este campo es la cadena nula en C y 48 caracteres en blanco en otros lenguajes de programación.

RecsPresent (MQLONG)

Este es el número de registros de mensajes MQPMR o registros de respuesta MQRR que ha proporcionado la aplicación. Este número solo puede ser mayor que cero si el mensaje se envía a una lista de distribución. Los registros de mensajes y los registros de respuesta son opcionales; la aplicación no tiene por qué proporcionar ningún registro, o puede optar por proporcionar registros de un solo tipo. Sin embargo, si la aplicación proporciona registros de ambos tipos, deberá proporcionar *RecsPresent* registros de cada tipo.

El valor de no tiene *RecsPresent* por qué ser el mismo que el número de destinos de la lista de distribución. Si se proporcionan demasiados registros, los excedentes no se utilizan; si se proporcionan muy pocos registros, se utilizan valores predeterminados para las propiedades del mensaje para aquellos destinos que no tienen registros de mensajes enviados (véase *PutMsgRecOffset*).

Si *RecsPresent* es menor que cero, o es mayor que cero pero el mensaje no se está colocando en una lista de distribución, la llamada falla con el código de motivo MQRC_RECS_PRESENT_ERROR.

Este es un campo de entrada. El valor inicial de este campo es 0. Este campo se ignora si *Version* es menor que MQPMO_VERSION_2.

PutMsgRecFields (MQLONG)

Este campo contiene indicadores que señalan qué campos MQPMR están presentes en los registros de mensajes enviados proporcionados por la aplicación. Úsalo *PutMsgRecFields* solo cuando el mensaje se envíe a una lista de distribución. El campo se ignora si *RecsPresent* es cero, o si tanto *PutMsgRecOffset* como *PutMsgRecPtr* son cero.

Para los campos que están presentes, el gestor de colas utiliza para cada destino los valores de los campos del registro de mensajes de envío correspondiente. Para los campos que no están presentes, el gestor de colas utiliza los valores de la estructura MQMD.

Utilice uno o varios de los siguientes indicadores para indicar qué campos están presentes en los registros del mensaje de envío:

MQPMRF_MSG_ID

El campo identificador de mensaje está presente.

MQPMRF_CORREL_ID

El campo identificador de correlación está presente.

MQPMRF_ID_GRUPO

El campo identificador de grupo está presente.

MQPMRF_COMENTARIOS

El campo de comentarios está presente.

MQPMRF_CONTABILIDAD_TOKEN

El campo «Accounting-token» está presente.

Si se especifica este indicador, se debe especificar MQPMO_SET_IDENTITY_CONTEXT o MQPMO_SET_ALL_CONTEXT en el *Options* campo; si no se cumple esta condición, la llamada falla con el código de motivo MQRC_PMO_RECORD_FLAGS_ERROR.

Si no hay campos MQPMR, se puede especificar lo siguiente:

MQPMRF_NONE

No hay campos de registro de mensajes enviados.

Si se especifica este valor, debe ser cero *RecsPresent*, o bien *PutMsgRecOffset* y *PutMsgRecPtr* deben ser cero.

MQPMRF_NONE se define para facilitar la documentación del programa. No está previsto que esta constante se utilice con ninguna otra, pero como su valor es cero, dicho uso no puede detectarse.

Si *PutMsgRecFields* contiene indicadores que no son válidos, o se proporcionan registros de mensajes put pero *PutMsgRecFields* tiene el valor MQPMRF_NONE, la llamada falla con el código de motivo MQRC_PMO_RECORD_FLAGS_ERROR.

Este es un campo de entrada. El valor inicial de este campo es MQPMRF_NONE. Este campo se ignora si *Version* es menor que MQPMO_VERSION_2.

PutMsgRecOffset (MQLONG)

Es el desplazamiento en bytes del primer registro de mensaje MQPMR desde el inicio de la estructura MQPMO. El desplazamiento puede ser positivo o negativo. *PutMsgRecOffset* Se utiliza solo cuando el mensaje se envía a una lista de distribución. El campo se ignora si *RecsPresent* es cero.

Cuando el mensaje se envía a una lista de distribución, se puede proporcionar una matriz de uno o más registros de mensajes MQPMR para especificar determinadas propiedades del mensaje para cada destino individualmente; estas propiedades son:

- Identificador de mensaje
- Identificador de correlación
- Identificador de grupo
- Valor de retroalimentación

- Señal de contabilidad

No es necesario especificar todas estas propiedades, pero sea cual sea el subconjunto que elija, especifique los campos en el orden correcto. Consulte la descripción de la estructura MQPMR para obtener más detalles.

Por lo general, debe haber tantos registros de mensajes enviados como registros de objetos especificados por MQOD cuando se abre la lista de distribución; cada registro de mensajes enviados proporciona las propiedades del mensaje para la cola identificada por el registro de objetos correspondiente. Las colas de la lista de distribución que no se pueden abrir deben seguir teniendo registros de mensajes asignados en las posiciones adecuadas de la matriz, aunque en este caso se ignoran las propiedades del mensaje.

El número de registros de mensajes enviados puede diferir del número de registros de objetos. Si hay menos registros de mensajes enviados que registros de objetos, las propiedades de los mensajes para los destinos que no tienen registros de mensajes enviados se toman de los campos correspondientes del descriptor de mensajes MQMD. Si hay más registros de mensajes que registros de objetos, los excedentes no se utilizan (aunque debe seguir siendo posible acceder a ellos). Los registros de mensajes son opcionales, pero si se proporcionan, deben ser *RecsPresent*.

Proporcione los registros de mensajes enviados de forma similar a los registros de objetos en MQOD, ya sea especificando un desplazamiento en o especificando una *PutMsgRecOffset* dirección en *PutMsgRecPtr*; para obtener más información sobre cómo hacerlo, consulte el *ObjectRecOffset* campo descrito en “MQOD - Descriptor de objeto” en la página 454.

No se puede utilizar *PutMsgRecPtr* más de uno de *PutMsgRecOffset* y; la llamada falla con el código de motivo MQRC_PUT_MSG_RECORDS_ERROR si ambos son distintos de cero.

Este es un campo de entrada. El valor inicial de este campo es 0. Este campo se ignora si *Version* es menor que MQPMO_VERSION_2.

ResponseRecOffset (MQLONG)

Es el desplazamiento en bytes del primer registro de respuesta MQRR desde el inicio de la estructura MQPMO. El desplazamiento puede ser positivo o negativo. *ResponseRecOffset* Se utiliza solo cuando el mensaje se envía a una lista de distribución. El campo se ignora si *RecsPresent* es cero.

Al enviar el mensaje a una lista de distribución, puede proporcionar una matriz de uno o más registros de respuesta MQRR para identificar las colas a las que no se envió correctamente el mensaje (*CompCode* campo en MQRR) y el motivo de cada fallo (*Reason* campo en MQRR). Es posible que el mensaje no se haya enviado porque la cola no se ha podido abrir o porque la operación de inserción ha fallado. El gestor de colas establece los registros de respuesta solo cuando el resultado de la llamada es mixto (es decir, algunos mensajes se enviaron correctamente, mientras que otros fallaron, o todos fallaron pero por motivos diferentes); el código de motivo MQRC_MULTIPLE_REASONS de la llamada indica este caso. Si el mismo código de motivo se aplica a todas las colas, ese motivo se devuelve en el **Reason** parámetro de la llamada MQPUT o MQPUT1, y los registros de respuesta no se establecen.

Por lo general, hay tantos registros de respuesta como registros de objetos especificados por MQOD cuando se abre la lista de distribución; cuando es necesario, cada registro de respuesta se establece en el código de finalización y el código de motivo para la inserción en la cola identificada por el registro de objeto correspondiente. Las colas de la lista de distribución que no se pueden abrir deben seguir teniendo registros de respuesta asignados en las posiciones adecuadas de la matriz, aunque se establezcan con el código de finalización y el código de motivo resultantes de la operación de apertura, en lugar de la operación de inserción.

El número de registros de respuesta puede diferir del número de registros de objetos. Si hay menos registros de respuesta que registros de objetos, es posible que la aplicación no pueda identificar todos los destinos para los que falló la operación de inserción, ni los motivos de los fallos. Si hay más registros de respuesta que registros de objetos, los excedentes no se utilizan (aunque debe seguir siendo posible acceder a ellos). Los registros de respuesta son opcionales, pero si se proporcionan, deben ser *RecsPresent*.

Proporcione los registros de respuesta de forma similar a los registros de objetos en MQOD, ya sea especificando un desplazamiento en *ResponseRecOffset*, o especificando una dirección en

ResponseRecPtr ; para obtener más información sobre cómo hacerlo, consulte el *ObjectRecOffset* campo descrito en “MQOD - Descriptor de objeto” en la página 454. Sin embargo, no utilice más de uno de *ResponseRecOffset* y *ResponseRecPtr* ; la llamada fallará con el código de motivo MQRC_RESPONSE_RECORDS_ERROR si ambos son distintos de cero.

Para la llamada « MQPUT1 », este campo debe ser cero. Esto se debe a que la información de respuesta (si se solicita) se devuelve en los registros de respuesta especificados por el descriptor de objeto MQOD.

Este es un campo de entrada. El valor inicial de este campo es 0. Este campo se ignora si *Version* es menor que MQPMO_VERSION_2.

PutMsgRecPtr (MQPTR)

Esta es la dirección del primer registro de mensajes MQPMR. Úsalo *PutMsgRecPtr* solo cuando el mensaje se envíe a una lista de distribución. El campo se ignora si *RecsPresent* es cero.

Se puede utilizar o *PutMsgRecOffsetPutMsgRecPtr* para especificar los registros de mensajes enviados, pero no ambos; para obtener más información, consulte *PutMsgRecOffset*. Si no lo utiliza *PutMsgRecPtr*, establézcalo en el puntero nulo o en bytes nulos.

Este es un campo de entrada. El valor inicial de este campo es el puntero nulo en aquellos lenguajes de programación que admiten punteros y, en los demás casos, una cadena de bytes totalmente nulos. Este campo se ignora si *Version* es menor que MQPMO_VERSION_2.

Nota: En plataformas donde el lenguaje de programación no admite el tipo de datos puntero, este campo se declara como una cadena de bytes de la longitud adecuada, con el valor inicial siendo la cadena de bytes totalmente nula.

ResponseRecPtr (MQPTR)

Esta es la dirección del primer registro de respuesta MQRR. *ResponseRecPtr* Se utiliza solo cuando el mensaje se envía a una lista de distribución. El campo se ignora si *RecsPresent* es cero.

Utilice o *ResponseRecOffsetResponseRecPtr* para especificar los registros de respuesta, pero no ambos; para obtener más información, consulte *ResponseRecOffset*. Si no lo utiliza, *ResponseRecPtr* establezca el puntero nulo o los bytes nulos.

Para la llamada al método ` MQPUT1 `, este campo debe ser el puntero nulo o bytes nulos. Esto se debe a que la información de respuesta (si se solicita) se devuelve en los registros de respuesta especificados por el descriptor de objeto MQOD.

Este es un campo de entrada. El valor inicial de este campo es el puntero nulo en aquellos lenguajes de programación que admiten punteros y, en los demás casos, una cadena de bytes totalmente nulos. Este campo se ignora si *Version* es menor que MQPMO_VERSION_2.

Nota: En plataformas donde el lenguaje de programación no admite el tipo de datos puntero, este campo se declara como una cadena de bytes de la longitud adecuada, con el valor inicial siendo la cadena de bytes totalmente nula.

OriginalMsgHandle (MQHMSG)

Este es un identificador opcional para un mensaje. Es posible que se haya recuperado previamente de una cola. El uso de este identificador está sujeto al valor del campo *Action* ; véase también *NewMsgHandle*.

El contenido del identificador del mensaje original no se verá alterado por la MQPUT llamada MQPUT1 o.

Este es un campo de entrada. El valor inicial de este campo es MQHM_NONE. Este campo se ignora si la versión es inferior a MQPMO_VERSION_3.

NewMsgHandle (MQHMSG)

Este es un identificador opcional para el mensaje que se incluye en función del valor del campo Acción. Define las propiedades del mensaje y anula los valores de *OriginalMsgHandle*, si se especifican.

Al regresar de la llamada **MQPUT MQPUT1** o, el contenido del identificador refleja el mensaje que realmente se envió.

Este es un campo de entrada. El valor inicial de este campo es **MQHM_NONE**. Este campo se ignora si la versión es inferior a **MQPMO_VERSION_3**.

Acción (MQLONG)

Esto especifica el tipo de inserción que se está realizando y la relación entre el mensaje original especificado por el campo « OriginalMsgHandle » y el nuevo mensaje especificado por el campo « NewMsgHandle ». Las propiedades del mensaje son seleccionadas por el gestor de colas según el valor de la acción especificada.

Puede optar por proporcionar el contenido del descriptor del mensaje utilizando el parámetro « MsgDesc » en las llamadas MQPUT o « MQPUT1 ». Como alternativa, es posible no proporcionar el parámetro MsgDesc o especificar que es solo de salida incluyendo MQPMO_MD_FOR_OUTPUT_ONLY en el campo Opciones de la estructura MQPMO.

Si no se proporciona el parámetro MsgDesc, o si se especifica que sea solo de salida, el descriptor de mensaje para el nuevo mensaje se rellena a partir de los campos del identificador de mensaje del MQPMO, de acuerdo con las reglas descritas en este tema.

Las actividades de configuración y transferencia del contexto descritas en [Control de la información de contexto](#) surten efecto después de que se haya compuesto el descriptor del mensaje.

Si se especifica un valor de acción incorrecto, la llamada falla con el código de motivo MQRC_ACTION_ERROR.

Se puede especificar cualquiera de las siguientes acciones:

MQACTP_NEW

Se está introduciendo un nuevo mensaje y el programa no especifica ninguna relación con un mensaje anterior. El descriptor del mensaje se compone de la siguiente manera:

- Si se proporciona un MsgDesc en la llamada MQPUT o MQPUT1, y MQPMO_MD_FOR_OUTPUT_ONLY no está en MQPMO.Options, esto se utiliza como descriptor de mensaje sin modificar.
- Si no se proporciona un identificador de descripción de mensaje (MsgDesc) o si MQPMO_MD_FOR_OUTPUT_ONLY se encuentra en la lista de control de descripción de mensajes (MQPMO.Options), el gestor de colas genera el descriptor de mensaje utilizando una combinación de propiedades de OriginalMsgHandle y NewMsgHandle. Cualquier campo descriptor de mensaje establecido explícitamente en el nuevo identificador de mensaje tiene prioridad sobre los del identificador de mensaje original.

Los datos del mensaje se toman del parámetro MQPUT o del parámetro del búfer de MQPUT1.

MQACTP_FORWARD

Se está reenviando un mensaje recuperado anteriormente. El identificador del mensaje original especifica el mensaje que se recuperó anteriormente.

El nuevo identificador de mensaje especifica cualquier modificación de las propiedades (incluidas las del descriptor de mensaje) en el identificador de mensaje original.

El descriptor del mensaje se compone de la siguiente manera:

- Si se proporciona un MsgDesc en la llamada MQPUT o MQPUT1, y MQPMO_MD_FOR_OUTPUT_ONLY no está en MQPMO.Options, esto se utiliza como descriptor de mensaje sin modificar.
- Si no se proporciona un identificador de descripción de mensaje (MsgDesc) o si MQPMO_MD_FOR_OUTPUT_ONLY se encuentra en la lista de control de descripción de mensajes (MQPMO.Options), el gestor de colas genera el descriptor de mensaje utilizando una combinación de propiedades de OriginalMsgHandle y NewMsgHandle. Cualquier campo descriptor de mensaje establecido explícitamente en el nuevo identificador de mensaje tiene prioridad sobre los del identificador de mensaje original.

- Si se especifican MQPMO_NEW_MSG_ID o MQPMO_NEW_CORREL_ID en el parámetro de línea de comandos `MQPMO.Options`, se respetarán.

Las propiedades del mensaje se componen de la siguiente manera:

- Todas las propiedades del identificador de mensaje original que tienen MQCOPY_FORWARD en el campo « MQPD.CopyOptions » (Identificador de copia de reenvío)
- Todas las propiedades del nuevo identificador de mensaje. Para cada propiedad del nuevo identificador de mensaje que tenga el mismo nombre que una propiedad del identificador de mensaje original, el valor se toma del nuevo identificador de mensaje. La única excepción a esta regla es el caso especial en el que la propiedad del nuevo identificador de mensaje tiene el mismo nombre que una propiedad del identificador de mensaje original, pero el valor de la propiedad es nulo. En este caso, la propiedad se elimina del mensaje.

Los datos del mensaje que se van a reenviar se toman del parámetro MQPUT o del parámetro del búfer de reenvío (MQPUT1).

MQACTP_REPLY

Se está respondiendo a un mensaje recuperado anteriormente. El identificador del mensaje original especifica el mensaje que se recuperó anteriormente.

El nuevo identificador de mensaje especifica cualquier modificación de las propiedades (incluidas las del descriptor de mensaje) en el identificador de mensaje original.

El descriptor del mensaje se compone de la siguiente manera:

- Si se proporciona un MsgDesc en la llamada MQPUT o MQPUT1, y MQPMO_MD_FOR_OUTPUT_ONLY no está en MQPMO.Options, esto se utiliza como descriptor de mensaje sin modificar.
- Si no se proporciona un identificador de tipo de mensaje (MsgDesc) o si MQPMO_MD_FOR_OUTPUT_ONLY se encuentra en el identificador de tipo de mensaje (MQPMO.Options), los campos del descriptor de mensaje inicial se eligen de la siguiente manera:

Tabla 532. Transformación del identificador del mensaje de respuesta	
Campo de MQMD	Valor utilizado
Informe	Si MQRO_PASS_DISCARD_AND_EXPIRY y MQRO_DISCARD_MSG están configurados: MQRO_DISCARD_MSG de lo contrario MQRO_NONE
MsgType	MQMT_REPLY
Caducidad	Si MQRO_PASS_DISCARD_AND_EXPIRY está establecido: Copiado del mensaje de entrada de lo contrario MQEI_UNLIMITED
Comentarios	MQFB_NONE
MsgId	Si se establece MQPMO_NEW_MSG_ID: Se genera un nuevo identificador de mensaje si MQRO_PASS_MSG_ID está establecido: Copiado del mensaje de entrada de lo contrario MQMI_NONE

Tabla 532. Transformación del identificador del mensaje de respuesta (continuación)	
Campo de MQMD	Valor utilizado
CorrelId	Si se establece MQPMO_NEW_CORREL_ID: Se genera un nuevo identificador de correlación si MQRO_COPY_MSG_ID_TO_CORREL_ID está establecido: Copiado del campo « MsgId » (Comentarios del cliente) del mensaje de entrada si MQRO_PASS_CORREL_ID está establecido: Copiado del campo « CorrelId » (Comentarios del cliente) del mensaje de entrada de lo contrario MQCI_NONE
BackoutCount	0
ReplyToQ	Espacios en blanco
ReplyToQMgr	Espacios en blanco
GroupId	MQGI_NONE
MsgSeqNumber	1
Desplazamiento	0
MsgFlags	MQMF_NINGUNO
OriginalLength	MQOL_INDETERMINADO

- A continuación, el descriptor de mensajes se modifica mediante el nuevo identificador de mensajes: cualquier campo del descriptor de mensajes establecido explícitamente como propiedad en el nuevo identificador de mensajes tiene prioridad sobre los campos del descriptor de mensajes, tal y como se ha descrito anteriormente.

Las propiedades del mensaje se componen de la siguiente manera:

- Todas las propiedades del identificador de mensaje original que tienen MQCOPY_REPLY en el campo MQPD.CopyOptions
- Todas las propiedades del nuevo identificador de mensaje. Para cada propiedad del nuevo identificador de mensaje que tenga el mismo nombre que una propiedad del identificador de mensaje original, el valor se toma del nuevo identificador de mensaje. La única excepción a esta regla es el caso especial en el que la propiedad del nuevo identificador de mensaje tiene el mismo nombre que una propiedad del identificador de mensaje original, pero el valor de la propiedad es nulo. En este caso, la propiedad se elimina del mensaje.

Los datos del mensaje que se van a reenviar se toman del parámetro « MQPUT/MQPUT1 » (Tamaño del búfer de reenvío) del búfer.

MQACTP_REPORT

Se está generando un informe como resultado de un mensaje recuperado anteriormente. El identificador del mensaje original especifica el mensaje que ha provocado la generación del informe.

El nuevo identificador de mensaje especifica cualquier modificación de las propiedades (incluidas las del descriptor de mensaje) en el identificador de mensaje original.

El descriptor del mensaje se compone de la siguiente manera:

- Si se proporciona un MsgDesc en la llamada MQPUT o MQPUT1, y MQPMO_MD_FOR_OUTPUT_ONLY no está en MQPMO.Options, esto se utiliza como descriptor de mensaje sin modificar.

- Si no se proporciona un identificador de descripción de mensaje (MsgDesc) o si MQPMO_MD_FOR_OUTPUT_ONLY se encuentra en la tabla de descripción de mensajes (MQPMO.Options), los campos iniciales del descriptor de mensaje se eligen de la siguiente manera:

Tabla 533. Transformación del identificador del mensaje del informe	
Campo de MQMD	Valor utilizado
Informe	Si MQRO_PASS_DISCARD_AND_EXPIRY y Se establecen MQRO_DISCARD_MSG: MQRO_DISCARD_MSG de lo contrario MQRO_NONE
MsgType	MQMT_REPORT
Caducidad	Si MQRO_PASS_DISCARD_AND_EXPIRY está establecido: Copiado del mensaje de entrada de lo contrario MQEI_UNLIMITED
MsgId	Si se establece MQPMO_NEW_MSG_ID: Se genera un nuevo identificador de mensaje si MQRO_PASS_MSG_ID está establecido: Copiado del mensaje de entrada de lo contrario MQMI_NONE
CorrelId	Si se establece MQPMO_NEW_CORREL_ID: Se genera un nuevo identificador de correlación si MQRO_COPY_MSG_ID_TO_CORREL_ID está establecido: Copiado del campo « MsgId » (Comentarios del cliente) del mensaje de entrada si MQRO_PASS_CORREL_ID está establecido: Copiado del campo « CorrelId » (Comentarios del cliente) del mensaje de entrada de lo contrario MQCI_NONE
BackoutCount	0
ReplyToQ	Espacios en blanco
ReplyToQMgr	Espacios en blanco
OriginalLength	Establecido en el <i>BufferLength</i>

- A continuación, el descriptor de mensajes se modifica mediante el nuevo identificador de mensajes: cualquier campo del descriptor de mensajes establecido explícitamente como propiedad en el nuevo identificador de mensajes tiene prioridad sobre los campos del descriptor de mensajes, tal y como se ha descrito anteriormente.

Las propiedades del mensaje se componen de la siguiente manera:

- Todas las propiedades del identificador de mensaje original que tienen MQCOPY_REPORT en el campo « MQPD.CopyOptions » (Identificador de mensaje original)

- Todas las propiedades del nuevo identificador de mensaje. Para cada propiedad del nuevo identificador de mensaje que tenga el mismo nombre que una propiedad del identificador de mensaje original, el valor se toma del nuevo identificador de mensaje. La única excepción a esta regla es el caso especial en el que la propiedad del nuevo identificador de mensaje tiene el mismo nombre que una propiedad del identificador de mensaje original, pero el valor de la propiedad es nulo. En este caso, la propiedad se elimina del mensaje.

El campo Feedback (Comentarios) en el MQMD resultante representa el informe que se va a generar. Un valor de retroalimentación MQFB_NONE provoca que la llamada MQPUT o MQPUT1 falle con el código de motivo MQRC_FEEDBACK_ERROR.

Para seleccionar los datos de usuario del mensaje de informe, IBM MQ consulta los campos «Report» y «Feedback» en el MQMD resultante, y los parámetros «Buffer» y « BufferLength » de la llamada MQPUT o « MQPUT1 ».

- Si Feedback es MQFB_COA, MQFB_COD o MQFB_EXPIRATION, se comprueba el valor de Report.
- Si se da alguno de los siguientes casos, se utilizan todos los datos del mensaje del búfer con una longitud de BufferLength.
 - La respuesta es MQFB_EXPIRATION y el informe contiene MQRO_EXPIRATION_WITH_FULL_DATA
 - La retroalimentación es MQFB_COD y el informe contiene MQRO_COD_WITH_FULL_DATA
 - La retroalimentación es MQFB_COA y el informe contiene MQRO_COA_WITH_FULL_DATA
- Si se da alguno de los siguientes casos, se utilizan los primeros 100 bytes del mensaje (o un BufferLength, si esta es inferior a 100) del búfer
 - La respuesta es MQFB_EXPIRATION y el informe contiene MQRO_EXPIRATION_WITH_DATA
 - La retroalimentación es MQFB_COD y el informe contiene MQRO_COD_WITH_DATA
 - La retroalimentación es MQFB_COA y el informe contiene MQRO_COA_WITH_DATA
- Si Feedback es MQFB_EXPIRATION, MQFB_COD o MQFB_COA, y Report no contiene las opciones *_WITH_FULL_DATA o *_WITH_DATA relevantes para ese valor de Feedback, entonces no se incluyen datos de usuario con el mensaje.
- Si Feedback toma un valor diferente a los enumerados anteriormente, entonces Buffer y BufferLength se utilizan de forma normal.

La derivación de los datos de usuario descritos en la lista anterior también se muestra en la siguiente tabla:

<i>Tabla 534. Fuente de los datos del usuario</i>			
	MQFB_COA	MQFB_COD	MQFB_CADUCIDAD
MQRO_EXPIRATION_CON_DATOS_COMPLETOS	Ninguna	Ninguna	Búfer (longitud del búfer)
MQRO_CÓDIGO_CON_DATOS_COMPLETOS	Ninguna	Búfer (longitud del búfer)	Ninguna
MQRO_COA_CON_DATOS_COMPLETOS	Búfer (longitud del búfer)	Ninguna	Ninguna
MQRO_EXPIRACIÓN_CON_DATOS	Ninguna	Ninguna	Búfer (primeros 100 bytes)
MQRO_COD_WITH_DATA	Ninguna	Búfer (primeros 100 bytes)	Ninguna
MQRO_COA_WITH_DATA	Búfer (primeros 100 bytes)	Ninguna	Ninguna

PubLevel (MQLONG)

El valor inicial de este campo es 9. El nivel de suscripción al que se dirige esta publicación. Solo aquellas suscripciones con el mayor valor de SubLevel inferior o igual a este valor reciben esta publicación. Este valor debe estar comprendido entre cero y 9; cero es el nivel más bajo. Sin embargo, si una publicación se ha conservado, ya no está disponible para los suscriptores de niveles superiores, ya que se vuelve a publicar en PubLevel 1.

Para más información, véase [Interceptación de publicaciones](#).

MQPMR - Registro de mensajes enviados

Utilice la estructura MQPMR para especificar diversas propiedades de mensaje para un único destino al enviar un mensaje a una lista de distribución. MQPMR es una estructura de entrada/salida para las llamadas MQPUT y MQPUT1.

Disponibilidad

La estructura MQPMR está disponible en las siguientes plataformas:

-  AIX
-  IBM i
-  Linux
-  Windows

y para IBM MQ los clientes conectados a estos sistemas.

Conjunto de caracteres y codificación

Los datos en MQPMR deben estar en el juego de caracteres proporcionado por el atributo del **CodedCharSetId** gestor de colas y la codificación del gestor de colas local proporcionada por MQENC_NATIVE. Sin embargo, si la aplicación se ejecuta como un cliente MQ, la estructura debe estar en el juego de caracteres y la codificación del cliente.

Utilización

Al proporcionar una matriz de estas estructuras en la llamada MQPUT o MQPUT1, puede especificar diferentes valores para cada cola de destino en una lista de distribución. Algunos campos son solo de entrada, otros son de entrada/salida.

Nota: Esta estructura es inusual, ya que no tiene un diseño fijo. Los campos de esta estructura son opcionales, y la presencia o ausencia de cada campo se indica mediante los indicadores del *PutMsgRecFields* campo en MQPMO. Los campos que están presentes **deben aparecer en el siguiente orden** :

- *MsgId*
- *CorrelId*
- *GroupId*
- *Feedback*
- *AccountingToken*

Los campos que están ausentes no ocupan espacio en el registro.

Dado que MQPMR no tiene un diseño fijo, no se proporciona ninguna definición del mismo en los archivos de encabezado, COPY e INCLUDE para los lenguajes de programación compatibles. El programador de la aplicación debe crear una declaración que contenga los campos que requiere la aplicación y establecer los indicadores en *PutMsgRecFields* para indicar los campos que están presentes.

Campos

No hay valores iniciales definidos para esta estructura, ya que no se proporcionan declaraciones de estructura en los archivos de INCLUDE encabezado, COPYy para los lenguajes de programación compatibles. Las declaraciones de muestra muestran cómo declarar la estructura si todos los campos son obligatorios.

Tabla 535. Campos en MQPMR	
Nombre de campo	Descripción del campo
MsgId	Identificador de mensaje
CorrelId	Identificador de correlación
GroupId	Identificador de grupo
FEEDBACK	Código de comentario o razón
AccountingToken	Señal de contabilidad

Declaraciones lingüísticas

Declaración C para MQPMR

```
typedef struct tagMQPMR MQPMR;
struct tagMQPMR {
    MQBYTE24 MsgId;           /* Message identifier */
    MQBYTE24 CorrelId;        /* Correlation identifier */
    MQBYTE24 GroupId;         /* Group identifier */
    MQLONG Feedback;          /* Feedback or reason code */
    MQBYTE32 AccountingToken; /* Accounting token */
};
```

COBOL declaración para MQPMR

```
** MQPMR structure
10 MQPMR.
** Message identifier
15 MQPMR-MSGID PIC X(24).
** Correlation identifier
15 MQPMR-CORRELID PIC X(24).
** Group identifier
15 MQPMR-GROUPID PIC X(24).
** Feedback or reason code
15 MQPMR-FEEDBACK PIC S9(9) BINARY.
** Accounting token
15 MQPMR-ACCOUNTINGTOKEN PIC X(32).
```

PL/I declaración para MQPMR

```
dcl
1 MQPMR based,
3 MsgId char(24), /* Message identifier */
3 CorrelId char(24), /* Correlation identifier */
3 GroupId char(24), /* Group identifier */
3 Feedback fixed bin(31), /* Feedback or reason code */
3 AccountingToken char(32); /* Accounting token */
```

Declaración de Visual Basic para MQPMR

```
Type MQPMR
MsgId As MQBYTE24 'Message identifier'
CorrelId As MQBYTE24 'Correlation identifier'
GroupId As MQBYTE24 'Group identifier'
Feedback As Long 'Feedback or reason code'
AccountingToken As MQBYTE32 'Accounting token'
End Type
```

Detalles de campo para MQPMR

Utilice la estructura de registro de mensajes MQPMR Put para especificar diversas propiedades de mensaje para un único destino al enviar un mensaje a una lista de distribución. MQPMR es una estructura de entrada/salida para las llamadas MQPUT y MQPUT1.

Campos

No hay valores iniciales definidos para esta estructura, ya que no se proporcionan declaraciones de estructura en los archivos de INCLUDE encabezado, COPYy para los lenguajes de programación compatibles. Las declaraciones de muestra muestran cómo declarar la estructura si todos los campos son obligatorios.

Tabla 536. Campos en MQPMR	
Nombre de campo	Descripción del campo
<u>MsgId</u>	Identificador de mensaje
<u>CorrelId</u>	Identificador de correlación
<u>GroupId</u>	Identificador de grupo
<u>FEEDBACK</u>	Código de comentario o razón
<u>AccountingToken</u>	Señal de contabilidad

MsgId (MQBYTE24)

Este es el identificador del mensaje que se utilizará para el mensaje enviado a la cola con un nombre especificado por el elemento correspondiente en la matriz de estructuras MQOR proporcionada en la llamada MQOPEN o MQPUT1. Se procesa de la misma manera que el *MsgId* campo en MQMD para una puesta en una sola cola.

Si este campo no está presente en el registro MQPMR, o si hay menos registros MQPMR que destinos, se utiliza el valor de MQMD para aquellos destinos que no tienen un registro MQPMR que contenga un *MsgId* campo. Si ese valor es MQMI_NONE, se genera un nuevo identificador de mensaje para *cada* uno de esos destinos (es decir, no hay dos destinos que tengan el mismo identificador de mensaje).

Si se especifica MQPMO_NEW_MSG_ID, se generan nuevos identificadores de mensaje para todos los destinos de la lista de distribución, independientemente de si tienen registros MQPMR. Esto difiere de la forma en que se procesa MQPMO_NEW_CORREL_ID (véase *CorrelId* el campo).

Este es un campo de entrada/salida.

CorrelId (MQBYTE24)

Este es el identificador de correlación que se utilizará para el mensaje enviado a la cola con un nombre especificado por el elemento correspondiente en la matriz de estructuras MQOR proporcionada en la llamada MQOPEN o MQPUT1. Se procesa de la misma manera que el *CorrelId* campo en MQMD para una puesta en una sola cola.

Si este campo no está presente en el registro MQPMR, o si hay menos registros MQPMR que destinos, se utiliza el valor de MQMD para aquellos destinos que no tienen un registro MQPMR que contenga un *CorrelId* campo.

Si se especifica MQPMO_NEW_CORREL_ID, se genera un *único* identificador de correlación nuevo y se utiliza para todos los destinos de la lista de distribución, independientemente de si tienen registros MQPMR. Esto difiere de la forma en que se procesa MQPMO_NEW_MSG_ID (véase *MsgId* el campo).

Este es un campo de entrada/salida.

GroupId (MQBYTE24)

GroupId Es el identificador de grupo que se utilizará para el mensaje enviado a la cola con el nombre especificado por el elemento correspondiente en la matriz de estructuras MQOR proporcionada en la llamada MQOPEN o MQPUT1. Se procesa de la misma manera que el *GroupId* campo en MQMD para una puesta en una sola cola.

Si este campo no está presente en el registro MQPMR, o si hay menos registros MQPMR que destinos, se utiliza el valor de MQMD para aquellos destinos que no tienen un registro MQPMR que contenga un *GroupId* campo. El valor se procesa tal y como se documenta en [Orden físico en una cola](#), pero con las siguientes diferencias:

- GroupId se crea a partir del QMName y una marca de tiempo. Por lo tanto, para mantener un gestor de colas único (GroupId), mantenga también únicos los nombres de los gestores de colas. Tampoco atrases los relojes en la máquina de los gestores de colas.
- En los casos en los que se utilizaría un nuevo identificador de grupo, el gestor de colas genera un identificador de grupo diferente para cada destino (es decir, no hay dos destinos con el mismo identificador de grupo).
- En los casos en los que se utilizaría el valor del campo, la llamada falla con el código de motivo MQRC_GROUP_ID_ERROR

Este es un campo de entrada/salida.

Feedback (MQLONG)

Este es el código de respuesta que se utilizará para el mensaje enviado a la cola con el nombre especificado por el elemento correspondiente en la matriz de estructuras MQOR proporcionada en la llamada MQOPEN o MQPUT1. Se procesa de la misma manera que el *Feedback* campo en MQMD para una puesta en una sola cola.

Si este campo no está presente, se utiliza el valor de MQMD.

Este es un campo de entrada.

AccountingToken (MQBYTE32)

Este es el token contable que se utilizará para el mensaje enviado a la cola con el nombre especificado por el elemento correspondiente en la matriz de estructuras MQOR proporcionada en la llamada MQOPEN o MQPUT1. Se procesa de la misma manera que el *AccountingToken* campo en MQMD para una puesta en una sola cola. Consulte la descripción de *AccountingToken* en [“MQMD - Descriptor de mensaje” en la página 389](#) para obtener información sobre el contenido de este campo.

Si este campo no está presente, se utiliza el valor de MQMD.

Este es un campo de entrada.

MQRFH - Cabecera de reglas y formato

La estructura MQRFH define el diseño de las reglas y el encabezado de formato. Utilice este encabezado para enviar datos de cadena en forma de pares nombre-valor.

Disponibilidad

Todos IBM MQ los sistemas, además de IBM MQ MQI clients los conectados a estos sistemas.

Nombre de formato

MQFMT_RF_HEADER

Conjunto de caracteres y codificación

Los campos de la estructura MQRFH (incluido *NameValueString*) están en el juego de caracteres y la codificación especificados por los campos *CodedCharSetId* *Encoding* y de la estructura de encabezado que precede a MQRFH, o por los campos de la estructura MQMD si MQRFH se encuentra al principio de los datos del mensaje de la aplicación.

El juego de caracteres debe ser uno que tenga caracteres de un solo byte para los caracteres que son válidos en los nombres de las colas.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 537. Campos en MQRFH para MQRFH		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>StrucId</u> (identificador de estructura)	MQRFH_STRUC_ID	'RFH↵'
<u>Versión</u> (número de versión de la estructura)	MQRFH_VERSION_1	1
<u>StrucLength</u> (longitud en bytes de la estructura MQRFH)	MQRFH_ESTRUCTURA_LONGITUD_FIJA	32
<u>Codificación</u> (codificación numérica de los datos que siguen <i>NameValueString</i>)	MQENC_NATIVE	Depende del entorno
<u>CodedCharSetId</u> (especifica el identificador del juego de caracteres de los datos que siguen <i>NameValueString</i>)	MQCCSI_INDETERMINADO	0
<u>Formato</u> (nombre del formato de los datos que siguen <i>NameValueString</i>)	MQFMT_NONE	Espacios en blanco
<u>Banderas</u> (banderas)	MQRFH_NINGUNO	0
<u>NameValueString</u> (cadena de caracteres de longitud variable que contiene pares nombre-valor)	ninguno	ninguno
Notas: 1. El símbolo ↵ representa un único carácter en blanco. 2. En el lenguaje de programación C, la variable macro MQRFH_DEFAULT contiene los valores que se enumeran en la tabla. Se puede utilizar de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQRFH MyRFH = {MQRFH_DEFAULT};</pre>		

Declaraciones lingüísticas

Declaración C para MQRFH

```
typedef struct tagMQRFH MQRFH;
struct tagMQRFH {
    MQCHAR4   StrucId;           /* Structure identifier */
    MQLONG    Version;          /* Structure version number */
    MQLONG    StrucLength;      /* Total length of MQRFH including
                                NameValueString */
    MQLONG    Encoding;         /* Numeric encoding of data that follows
                                NameValueString */
    MQLONG    CodedCharSetId;   /* Character set identifier of data that
                                follows NameValueString */
    MQCHAR8   Format;           /* Format name of data that follows
                                NameValueString */
};
```

```

    MQLONG    Flags;          /* Flags */
};

```

COBOL declaración para MQRFH

```

**  MQRFH structure
10  MQRFH.
**    Structure identifier
15  MQRFH-STRUCID          PIC X(4).
**    Structure version number
15  MQRFH-VERSION          PIC S9(9) BINARY.
**    Total length of MQRFH including NAMEVALUESTRING
15  MQRFH-STRUCLNGTH      PIC S9(9) BINARY.
**    Numeric encoding of data that follows NAMEVALUESTRING
15  MQRFH-ENCODING        PIC S9(9) BINARY.
**    Character set identifier of data that follows NAMEVALUESTRING
15  MQRFH-CODEDCHARSETID  PIC S9(9) BINARY.
**    Format name of data that follows NAMEVALUESTRING
15  MQRFH-FORMAT          PIC X(8).
**    Flags
15  MQRFH-FLAGS           PIC S9(9) BINARY.

```

PL/I declaración para MQRFH

```

dcl
1  MQRFH based,
3  StrucId      char(4),      /* Structure identifier */
3  Version      fixed bin(31), /* Structure version number */
3  StrucLength  fixed bin(31), /* Total length of MQRFH including
                                NameValueString */
3  Encoding     fixed bin(31), /* Numeric encoding of data that
                                follows NameValueString */
3  CodedCharSetId fixed bin(31), /* Character set identifier of data
                                that follows NameValueString */
3  Format       char(8),      /* Format name of data that follows
                                NameValueString */
3  Flags       fixed bin(31); /* Flags */

```

High Level Assembler declaración para MQRFH

MQRFH	DSECT	
MQRFH_STRUCID	DS CL4	Structure identifier
MQRFH_VERSION	DS F	Structure version number
MQRFH_STRUCLNGTH	DS F	Total length of MQRFH including
*		NAMEVALUESTRING
MQRFH_ENCODING	DS F	Numeric encoding of data that follows
*		NAMEVALUESTRING
MQRFH_CODEDCHARSETID	DS F	Character set identifier of data that
*		follows NAMEVALUESTRING
MQRFH_FORMAT	DS CL8	Format name of data that follows
*		NAMEVALUESTRING
MQRFH_FLAGS	DS F	Flags
*		
MQRFH_LENGTH	EQU *-MQRFH	
	ORG MQRFH	
MQRFH_AREA	DS CL(MQRFH_LENGTH)	

Declaración de Visual Basic para MQRFH

Type MQRFH	
StrucId	As String*4 'Structure identifier'
Version	As Long 'Structure version number'
StrucLength	As Long 'Total length of MQRFH including 'NameValueString'
Encoding	As Long 'Numeric encoding of data that follows 'NameValueString'
CodedCharSetId	As Long 'Character set identifier of data that 'follows NameValueString'
Format	As String*8 'Format name of data that follows 'NameValueString'
Flags	As Long 'Flags'
End Type	

Detalles de campo para MQRFH

La estructura del encabezado de reglas y formato MQRFH define el diseño del encabezado de reglas y formato. Utilice este encabezado para enviar datos de cadena en forma de pares nombre-valor.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 538. Campos en MQRFH		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>StrucId</u> (identificador de estructura)	MQRFH_STRUC_ID	'RFH↵'
<u>Versión</u> (número de versión de la estructura)	MQRFH_VERSION_1	1
<u>StrucLength</u> (longitud en bytes de la estructura MQRFH)	MQRFH_ESTRUCTURA_LONGITUD_FIJA	32
<u>Codificación</u> (codificación numérica de los datos que siguen <i>NameValueString</i>)	MQENC_NATIVE	Depende del entorno
<u>CodedCharSetId</u> (especifica el identificador del juego de caracteres de los datos que siguen <i>NameValueString</i>)	MQCCSI_INDETERMINADO	0
<u>Formato</u> (nombre del formato de los datos que siguen <i>NameValueString</i>)	MQFMT_NONE	Espacios en blanco
<u>Banderas</u> (banderas)	MQRFH_NINGUNO	0
<u>NameValueString</u> (cadena de caracteres de longitud variable que contiene pares nombre-valor)	ninguno	ninguno
Notas: 1. El símbolo ↵ representa un único carácter en blanco. 2. En el lenguaje de programación C, la variable macro MQRFH_DEFAULT contiene los valores que se enumeran en la tabla. Se puede utilizar de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQRFH MyRFH = {MQRFH_DEFAULT};</pre>		

StrucId (MQCHAR4)

Este es el identificador de estructura de las reglas y la estructura del encabezado de formato. Siempre es un campo de entrada. Su valor es MQRFH_STRUC_ID.

El valor debe ser:

MQRFH_STRUC_ID

Identificador de las reglas y la estructura del encabezado de formato.

Para el lenguaje de programación C, también se define la constante MQRFH_STRUC_ID_ARRAY. Tiene el mismo valor que MQRFH_STRUC_ID, pero es una matriz de caracteres en lugar de una cadena.

Versión (MQLONG)

Este es el número de versión de la estructura; el valor debe ser:

MQRFH_VERSION_1

Version-1 normas y estructura de formato de los encabezados.

El valor inicial de este campo es MQRFH_VERSION_1.

StrucLength (MQLONG)

Es la longitud en bytes de la estructura MQRFH, incluido el *NameValueString* campo al final de la estructura. La longitud no incluye ningún dato de usuario que siga al *NameValueString* campo.

Para evitar problemas al convertir los datos de usuario en algunos entornos, *StrucLength* debe ser múltiplo de cuatro.

La siguiente constante indica la longitud de la parte *fija* de la estructura, es decir, la longitud excluyendo el *NameValueString* campo:

MQRFH_ESTRUCTURA_LONGITUD_FIJA

Longitud de la parte fija de la estructura MQRFH.

El valor inicial de este campo es MQRFH_STRUC_LENGTH_FIXED.

Encoding (MQLONG)

Esto especifica la codificación numérica de los datos que siguen *NameValueString* ; no se aplica a los datos numéricos de la propia estructura MQRFH.

En la llamada MQPUT o MQPUT1, la aplicación debe establecer este campo con el valor adecuado para los datos.

El valor inicial de este campo es MQENC_NATIVE.

CodedCharSetId (MQLONG)

Esto especifica el identificador del conjunto de caracteres de los datos que siguen *NameValueString* ; no se aplica a los datos de caracteres de la propia estructura MQRFH.

En la llamada MQPUT o MQPUT1, la aplicación debe establecer este campo con el valor adecuado para los datos. Se puede utilizar el siguiente valor especial:

MQCCSI_HEREDAR

Los datos de caracteres en los datos *que siguen* esta estructura están en el mismo conjunto de caracteres que esta estructura.

El gestor de colas cambia este valor en la estructura enviada en el mensaje al identificador real del juego de caracteres de la estructura. Si no se produce ningún error, la llamada MQGET no devuelve el valor MQCCSI_INHERIT.

MQCCSI_INHERIT no se puede utilizar si el valor del *PutApplType* campo en MQMD es MQAT_BROKER.

El valor inicial de este campo es MQCCSI_UNDEFINED.

Format (MQCHAR8)

Especifica el nombre del formato de los datos que siguen *NameValueString*.

En la llamada MQPUT o MQPUT1, la aplicación debe establecer este campo con el valor adecuado para los datos. Las reglas para codificar este campo son las mismas que para el *Format* campo en MQMD.

El valor inicial de este campo es MQFMT_NONE.

Flags (MQLONG)

Se puede especificar lo siguiente:

MQRFH_NINGUNO

Sin banderas.

El valor inicial de este campo es MQRFH_NONE.

NameValueString (MQCHARn)

Se trata de una cadena de caracteres de longitud variable que contiene pares nombre-valor con el siguiente formato:

```
name1 value1 name2 value2 name3 value3 ...
```

Cada nombre o valor debe estar separado del nombre o valor adyacente por uno o más caracteres en blanco; estos espacios en blanco no son significativos. Un nombre o valor puede contener espacios significativos anteponiendo y posponiendo el nombre o valor con comillas dobles; todos los caracteres entre la comilla doble abierta y la comilla doble cerrada correspondiente se tratan como significativos. En el siguiente ejemplo, el nombre es FAMOUS_WORDS y el valor es Hello World:

```
FAMOUS_WORDS "Hello World"
```

Un nombre o valor puede contener cualquier carácter excepto el carácter nulo (que actúa como delimitador para *NameValueString*). Sin embargo, para facilitar la interoperabilidad, una aplicación puede restringir los nombres a los siguientes caracteres:

- Primer carácter: letra mayúscula o minúscula (de la A a la Z, o de la a a la z), o guión bajo.
- Caracteres siguientes: letras mayúsculas o minúsculas, dígitos decimales (del 0 al 9), guión bajo, guión o punto.

Si un nombre o valor contiene una o más comillas dobles, el nombre o valor debe estar entre comillas dobles, y cada comilla doble dentro de la cadena debe duplicarse:

```
Famous_Words "The program displayed ""Hello World"""
```

Los nombres y los valores distinguen entre mayúsculas y minúsculas, es decir, las letras minúsculas no se consideran iguales a las mayúsculas. Por ejemplo, FAMOUS_WORDS y Famous_Words son dos nombres diferentes.

La longitud en bytes de *NameValueString* es igual a *StrucLength* menos MQRFH_STRUC_LENGTH_FIXED. Para evitar problemas al convertir los datos de usuario en algunos entornos, haz que esta longitud sea un múltiplo de cuatro. Rellene *NameValueString* con espacios en blanco hasta alcanzar esta longitud, o finalízelo antes colocando un carácter nulo después del último carácter significativo de la cadena. El carácter nulo y los bytes que le siguen, hasta la longitud especificada de *NameValueString*, se ignoran.

Nota: Debido a que la longitud de este campo no es fija, el campo se omite de las declaraciones de la estructura que se proporcionan para los lenguajes de programación compatibles.

MQRFH2 – Reglas y formato de la cabecera 2

El MQRFH2 encabezado se basa en el MQRFH encabezado, pero permite transportar cadenas Unicode sin traducción y puede transportar tipos de datos numéricos. La MQRFH2 estructura define el formato de las reglas de version-2 y el encabezado de formato. Este encabezado se utiliza para enviar datos que han sido codificados utilizando una sintaxis similar a XML. Un mensaje puede contener dos o más MQRFH2 estructuras en serie, con datos de usuario opcionalmente después de la última MQRFH2 estructura de la serie.

Disponibilidad

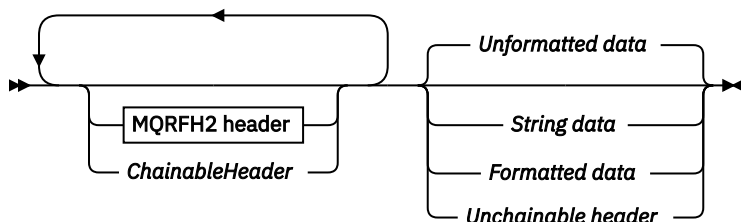
Todos IBM MQ los sistemas, además de IBM MQ MQI clients los conectados a estos sistemas.

Nombre de formato

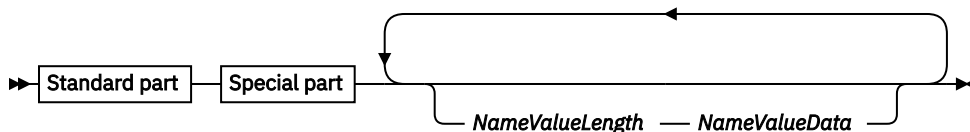
MQFMT_RF_HEADER_2

Syntax

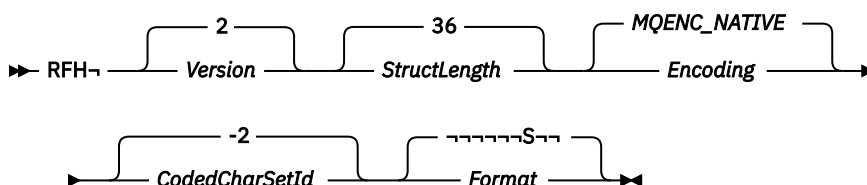
IBM MQ Message



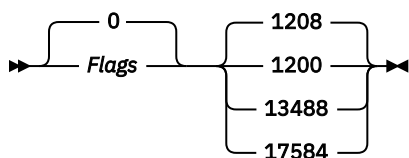
MQRFH2 header



Standard part



Special part



Conjunto de caracteres y codificación

Se aplican reglas especiales al conjunto de caracteres y la codificación utilizados para la MQRFH2 estructura:

- Los campos distintos de *NameValueData* están en el conjunto de caracteres y la codificación indicados por los campos *CodedCharSetId* *Encoding* y en la estructura de encabezado que precede MQRFH2a, o por los campos de la MQMD estructura si se MQRFH2 encuentra al principio de los datos del mensaje de la aplicación.

El juego de caracteres debe ser uno que tenga caracteres de un solo byte para los caracteres que son válidos en los nombres de las colas.

Cuando MQGMO_CONVERT se especifica en la MQGET llamada, el gestor de colas convierte los campos MQRFH2, excepto *NameValueData*, al juego de caracteres y la codificación solicitados.

- NameValueData* está en el conjunto de caracteres dado por el *NameValueCCSID* campo. Solo los conjuntos de caracteres Unicode enumerados son válidos para *NameValueCCSID* ; consulte la descripción de *NameValueCCSID* para obtener más detalles.

Algunos conjuntos de caracteres tienen una representación que depende de la codificación. Si *NameValueCCSID* es uno de estos conjuntos de caracteres, *NameValueData* debe tener la misma codificación que los demás campos del MQRFH2.

Cuando se especifica MQGMO_CONVERT en la MQGET llamada, el gestor de colas convierte *NameValueData* a la codificación solicitada, pero no cambia su juego de caracteres.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 539. Campos en MQRFH2 para MQRFH2		
Nombre de campo	Nombre de la constante	Valor de la constante
StrucId (identificador de estructura)	MQRFH_STRUC_ID	'RFH~'
Versión (número de versión de la estructura)	MQRFH_VERSION_2	2
StrucLength (longitud en bytes de la estructura MQRFH2)	MQRFH_STRUC_LENGTH_FIXED_2	36
Codificación (codificación numérica de los datos que siguen al último <i>NameValueData</i> campo)	MQENC_NATIVE	Depende del entorno
CodedCharSetId (identificador del conjunto de caracteres de los datos que siguen al último <i>NameValueData</i> campo)	MQCCSI_HEREDAR	-2
Formato (nombre del formato de los datos que siguen al último <i>NameValueData</i> campo)	MQFMT_NONE	Espacios en blanco
Banderas (banderas)	MQRFH_NINGUNO	0
NameValueCCSID (identificador del conjunto de caracteres codificados de los datos en el <i>NameValueData</i> campo)	Ninguna	1208
NameValueLength (longitud en bytes de los datos del campo <i>NameValueData</i>)	Ninguna	None
NameValueData (pares nombre-valor de las propiedades del mensaje)	Ninguna	Ninguna

Tabla 539. Campos en MQRFH2 para MQRFH2 (continuación)

Nombre de campo	Nombre de la constante	Valor de la constante
Notas: 1. El símbolo ~ representa un único carácter en blanco. 2. En el lenguaje de programación C, la variable macroMQRFH2_DEFAULT contiene los valores que se enumeran en la tabla. Úselo de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQRFH2 MyRFH2 = {MQRFH2_DEFAULT};</pre>		

Declaraciones lingüísticas

Declaración C para MQRFH2

```
typedef struct tagMQRFH2 MQRFH2;
struct tagMQRFH2 {
    MQCHAR4  StrucId;           /* Structure identifier */
    MQLONG   Version;          /* Structure version number */
    MQLONG   StrucLength;      /* Total length of MQRFH2 including all
                               NameValueLength and NameValueData
                               fields */
    MQLONG   Encoding;         /* Numeric encoding of data that follows
                               last NameValueData field */
    MQLONG   CodedCharSetId;   /* Character set identifier of data that
                               follows last NameValueData field */
    MQCHAR8  Format;           /* Format name of data that follows last
                               NameValueData field */
    MQLONG   Flags;            /* Flags */
    MQLONG   NameValueCCSID;   /* Character set identifier of
                               NameValueData */
};
```

COBOL Declaración para MQRFH2

```
** MQRFH2 structure
10 MQRFH2.
** Structure identifier
15 MQRFH2-STRUCID PIC X(4).
** Structure version number
15 MQRFH2-VERSION PIC S9(9) BINARY.
** Total length of MQRFH2 including all NAMEVALUELENGTH and
** NAMEVALUEDATA fields
15 MQRFH2-STRUCLNGTH PIC S9(9) BINARY.
** Numeric encoding of data that follows last NAMEVALUEDATA field
15 MQRFH2-ENCODING PIC S9(9) BINARY.
** Character set identifier of data that follows last NAMEVALUEDATA
** field
15 MQRFH2-CODEDCHARSETID PIC S9(9) BINARY.
** Format name of data that follows last NAMEVALUEDATA field
15 MQRFH2-FORMAT PIC X(8).
** Flags
15 MQRFH2-FLAGS PIC S9(9) BINARY.
** Character set identifier of NAMEVALUEDATA
15 MQRFH2-NAMEVALUECCSID PIC S9(9) BINARY.
```

PL/I Declaración para MQRFH2

```
dcl
1 MQRFH2 based,
3 StrucId char(4), /* Structure identifier */
3 Version fixed bin(31), /* Structure version number */
3 StrucLength fixed bin(31), /* Total length of MQRFH2 including
                               all NameValueLength and
                               NameValueData fields */
3 Encoding fixed bin(31), /* Numeric encoding of data that
                               follows last NameValueData field */
3 CodedCharSetId fixed bin(31), /* Character set identifier of data
                               that follows last NameValueData
                               field */
```

```

3 Format          char(8),          /* Format name of data that follows
                                   last NameValueData field */
3 Flags          fixed bin(31), /* Flags */
3 NameValueCCSID fixed bin(31); /* Character set identifier of
                                   NameValueData */

```

High Level Assembler Declaración para MQRFH2

```

MQRFH          DSECT
MQRFH_STRUCID  DS    CL4  Structure identifier
MQRFH_VERSION  DS    F    Structure version number
MQRFH_STRUCLNGTH DS    F    Total length of MQRFH2 including all
*                NAMEVALUELENGTH and NAMEVALUEDATA fields
MQRFH_ENCODING DS    F    Numeric encoding of data that follows
*                last NAMEVALUEDATA field
MQRFH_CODEDCHARSETID DS    F    Character set identifier of data that
*                follows last NAMEVALUEDATA field
MQRFH_FORMAT   DS    CL8  Format name of data that follows last
*                NAMEVALUEDATA field
MQRFH_FLAGS    DS    F    Flags
MQRFH_NAMEVALUECCSID DS    F    Character set identifier of
*                NAMEVALUEDATA
*
MQRFH_LENGTH   EQU    *-MQRFH
                ORG    MQRFH
MQRFH_AREA     DS     CL(MQRFH_LENGTH)

```

Declaración de Visual Basic para MQRFH2

```

Type MQRFH2
  StrucId      As String*4 'Structure identifier'
  Version      As Long     'Structure version number'
  StrucLength  As Long     'Total length of MQRFH2 including all'
                          'NameValueLength and NameValueData fields'
  Encoding     As Long     'Numeric encoding of data that follows'
                          'last NameValueData field'
  CodedCharSetId As Long   'Character set identifier of data that'
                          'follows last NameValueData field'
  Format       As String*8 'Format name of data that follows last'
                          'NameValueData field'
  Flags        As Long     'Flags'
  NameValueCCSID As Long   'Character set identifier of NameValueData'
End Type

```

Detalles del campo para MQRFH2

El encabezado MQRFH2 Rules and formatting header 2 se basa en el MQRFH encabezado header, pero permite transportar cadenas Unicode sin traducción y puede transportar tipos de datos numéricos.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 540. Campos en MQRFH2		
Nombre de campo	Nombre de la constante	Valor de la constante
StrucId (identificador de estructura)	MQRFH_STRUC_ID	'RFH~'
Versión (número de versión de la estructura)	MQRFH_VERSION_2	2
StrucLength (longitud en bytes de la estructura MQRFH2)	MQRFH_STRUC_LENGTH_FIXED_2	36

Tabla 540. Campos en MQRFH2 (continuación)

Nombre de campo	Nombre de la constante	Valor de la constante
<u>Codificación</u> (codificación numérica de los datos que siguen al último <i>NameValueData</i> campo)	MQENC_NATIVE	Depende del entorno
<u>CodedCharSetId</u> (identificador del conjunto de caracteres de los datos que siguen al último <i>NameValueData</i> campo)	MQCCSI_HEREDAR	-2
<u>Formato</u> (nombre del formato de los datos que siguen al último <i>NameValueData</i> campo)	MQFMT_NONE	Espacios en blanco
<u>Banderas</u> (banderas)	MQRFH_NINGUNO	0
<u>NameValueCCSID</u> (identificador del conjunto de caracteres codificados de los datos en el <i>NameValueData</i> campo)	Ninguna	1208
<u>NameValueLength</u> (longitud en bytes de los datos del campo <i>NameValueData</i>)	Ninguna	None
<u>NameValueData</u> (pares nombre-valor de las propiedades del mensaje)	Ninguna	Ninguna

Notas:

1. El símbolo ~ representa un único carácter en blanco.
2. En el lenguaje de programación C, la variable macro MQRFH2_DEFAULT contiene los valores que se enumeran en la tabla. Úselo de la siguiente manera para proporcionar valores iniciales para los campos de la estructura:

```
MQRFH2 MyRFH2 = {MQRFH2_DEFAULT};
```

StrucId (MQCHAR4)

Este es el identificador de estructura de las reglas y el encabezado de formato de dos estructuras. Siempre es un campo de entrada. Su valor es MQRFH2_STRUC_ID.

El valor debe ser:

MQRFH2_STRUC_ID

Identificador para las reglas y la estructura del encabezado de formato dos.

Para el lenguaje de programación C, también se define la constante `MQRFH2_STRUC_ID_ARRAY`. Esto tiene el mismo valor que `MQRFH2_STRUC_ID`, pero es una matriz de caracteres en lugar de una cadena.

Versión (MQLONG)

Este es el número de versión de la estructura; el valor debe ser:

MQRFH_VERSION_2

Version-2 normas y estructura de formato de los encabezados.

El valor inicial de este campo es `MQRFH_VERSION_2`.

StrucLength (MQLONG)

Esta es la longitud en bytes de la estructura `MQRFH2`, incluidos los campos *NameValueLength* *NameValueData* y al final de la estructura. Es válido que haya varios pares de campos *NameValueLength* *NameValueData* y al final de la estructura, en la secuencia:

```
length1, data1, length2, data2, ...
```

StrucLength no incluye ningún dato de usuario que pueda seguir al último *NameValueData* campo al final de la estructura.

Para evitar problemas con la conversión de los datos de usuario en algunos entornos, *StrucLength* debe ser múltiplo de cuatro.

La siguiente constante indica la longitud de la parte *fija* de la estructura, es decir, la longitud excluyendo los campos *NameValueLength* *NameValueData* y :

MQRFH_STRUC_LENGTH_FIXED_2

Longitud de la parte fija de una estructura `MQRFH2`.

El valor inicial de este campo es `MQRFH_STRUC_LENGTH_FIXED_2`.

Encoding (MQLONG)

Especifica la codificación numérica de los datos que siguen al último *NameValueData* campo; no se aplica a los datos numéricos de la estructura `MQRFH2`.

En la llamada `MQPUT` o `MQPUT1`, la aplicación debe establecer este campo con el valor adecuado para los datos.

El valor inicial de este campo es `MQENC_NATIVE`.

CodedCharSetId (MQLONG)

Especifica el identificador del conjunto de caracteres de los datos que siguen al último *NameValueData* campo; no se aplica a los datos de caracteres de la propia estructura `MQRFH2`.

En la llamada `MQPUT` o `MQPUT1`, la aplicación debe establecer este campo con el valor adecuado para los datos. Se puede utilizar el siguiente valor especial:

MQCCSI_HEREDAR

Los datos de caracteres en los datos *que siguen* esta estructura están en el mismo conjunto de caracteres que esta estructura.

El gestor de colas cambia este valor en la estructura enviada en el mensaje al identificador real del juego de caracteres de la estructura. Si no se produce ningún error, la llamada `MQGET` no devuelve el valor `MQCCSI_INHERIT`.

MQCCSI_INHERIT no se puede utilizar si el valor del *PutApplType* campo en MQMD es MQAT_BROKER.

El valor inicial de este campo es MQCCSI_INHERIT.

Format (MQCHAR8)

Especifica el nombre del formato de los datos que siguen al último *NameValueData* campo.

En la llamada MQPUT o MQPUT1, la aplicación debe establecer este campo con el valor adecuado para los datos. Las reglas para codificar este campo son las mismas que para el *Format* campo en MQMD.

El valor inicial de este campo es MQFMT_NONE.

Flags (MQLONG)

El valor inicial de este campo es MQRFH_NONE. MQRFH_NONE debe especificarse.

MQRFH_NONE

Sin banderas.

MQRFH_INTERNAL

El MQRFH2 encabezado contiene propiedades establecidas internamente.

MQRFH_INTERNAL Es para uso del administrador de colas.

Los 16 bits superiores, MQRFH_FLAGS_RESTRICTED_MASK, están reservados para los indicadores que establece el gestor de colas. Los indicadores que un usuario puede establecer se definen en los 16 bits inferiores.

NameValueCCSID (MQLONG)

Especifica el identificador del conjunto de caracteres codificados de los datos del *NameValueData* campo. Esto difiere del conjunto de caracteres de las demás cadenas de la estructura MQRFH2 y puede diferir del conjunto de caracteres de los datos (si los hay) que siguen al último *NameValueData* campo al final de la estructura.

NameValueCCSID debe tener uno de los siguientes valores:

CCSID	Significado
1200	UTF-16, versión soportada más reciente de Unicode
13488	UTF-16, subconjunto de Unicode versión 2.0
17584	UTF-16, subconjunto de Unicode versión 3.0 (incluye símbolo del Euro)
1208	UTF-8, versión soportada más reciente de Unicode

Para los conjuntos de caracteres UTF-16, la codificación (orden de bytes) del *NameValueData* debe ser la misma que la codificación de los demás campos de la estructura MQRFH2.

Los caracteres que se encuentran fuera del plano multilingüe básico de Unicode (los que están por encima de U+FFFF), representados en UTF-16 por puntos de código sustitutos (X' D800 ' a X'DFFF') o cuatro bytes en UTF-8, no son compatibles.

Nota: Si *NameValueCCSID* no tiene uno de los valores enumerados anteriormente y la estructura MQRFH2 requiere conversión en la llamada MQGET, la llamada se completa con el código de motivo MQRC_SOURCE_CCSD_ERROR y el mensaje se devuelve sin convertir.

El valor inicial de este campo es 1208.

NameValueLength (MQLONG)

La longitud del campo NameValueData correspondiente.

Especifica la longitud en bytes de los datos del *NameValueData* campo. *NameValueLength* debe ser múltiplo de cuatro.

Nota: Los campos *NameValueLength* *NameValueData* y son opcionales, pero si están presentes deben aparecer como un par y ser adyacentes. El par de campos se puede repetir tantas veces como sea necesario, por ejemplo:

```
length1 data1 length2 data2 length3 data3
```

Dado que estos campos son opcionales, se omiten en las declaraciones de la estructura que se proporcionan para los distintos lenguajes de programación compatibles.

NameValueData (MQCHARn)

NameValueData Es un campo de longitud variable que contiene una carpeta con pares nombre-valor de propiedades del mensaje.

Una carpeta es una cadena de caracteres de longitud variable que contiene datos codificados utilizando una sintaxis similar a XML. La longitud en bytes de la cadena de caracteres viene dada por el *NameValueLength* campo que precede al *NameValueData* campo. La longitud debe ser un múltiplo de cuatro.

Los campos *NameValueLength* *NameValueData* y son opcionales, pero si están presentes deben aparecer como un par y ser adyacentes. El par de campos se puede repetir tantas veces como sea necesario, por ejemplo:

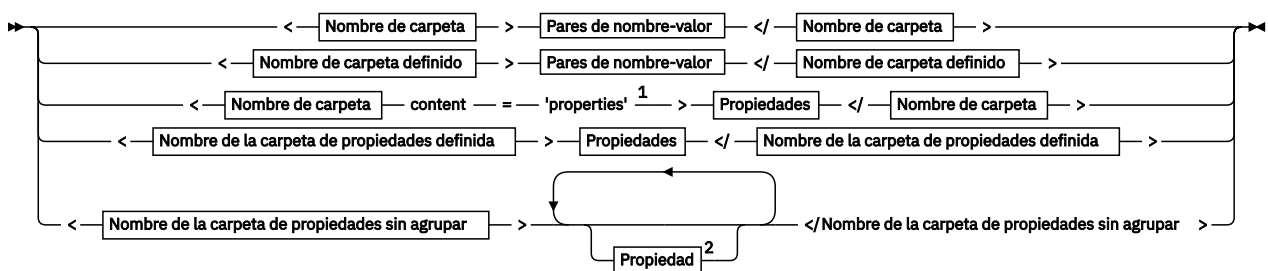
```
length1 data1 length2 data2 length3 data3
```

NameValueData no se convierte al juego de caracteres especificado en la MQGET llamada. Aunque el mensaje se recupere con la opción MQGMO_CONVERT activa, *NameValueData* permanecerá en su juego de caracteres original. Sin embargo, *NameValueData* se convierte a la codificación especificada en la MQGET llamada.

Notas:

- Dado que estos campos son opcionales, se omiten en las declaraciones de la estructura que se proporcionan para los distintos lenguajes de programación compatibles.
- Los términos "definidos" y "reservados" se utilizan en el diagrama sintáctico. "Definido" significa que el nombre es utilizado por IBM MQ. "Reservado" significa que el nombre está reservado para su uso futuro por parte de IBM MQ.

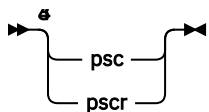
NameValueData sintaxis



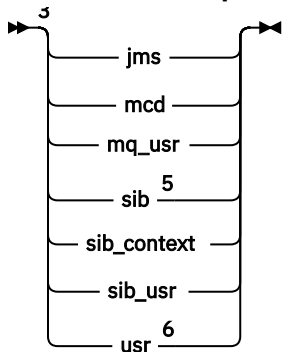
Nombre de carpeta

►► `Nombre` ◄◄

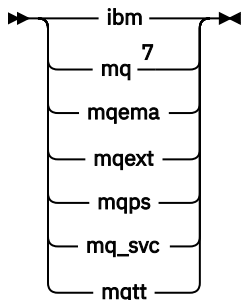
Nombre de carpeta definido



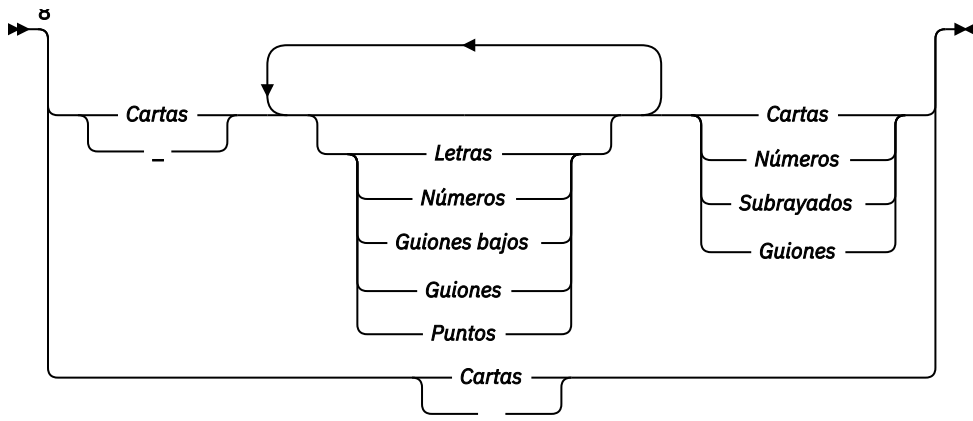
Nombre de la carpeta de propiedades definida



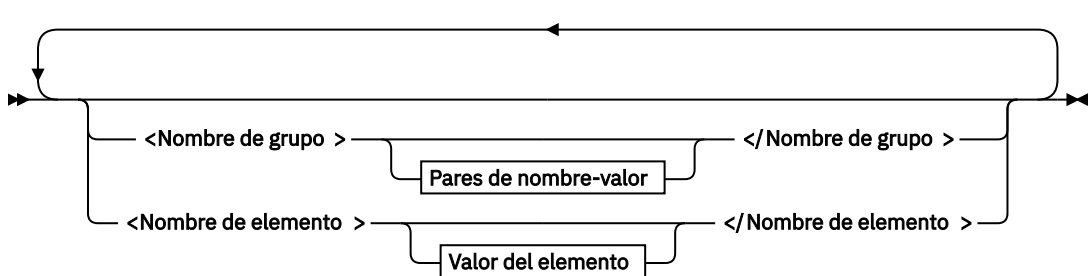
Nombre de la carpeta de propiedades sin agrupar



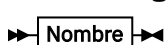
Nombre



Pares de nombre-valor



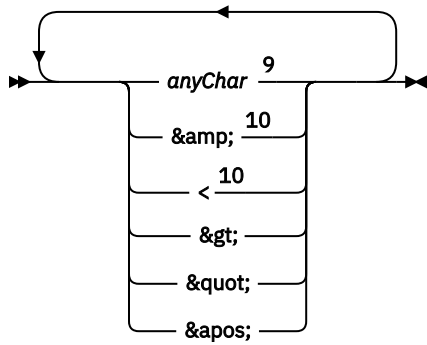
Nombre de grupo



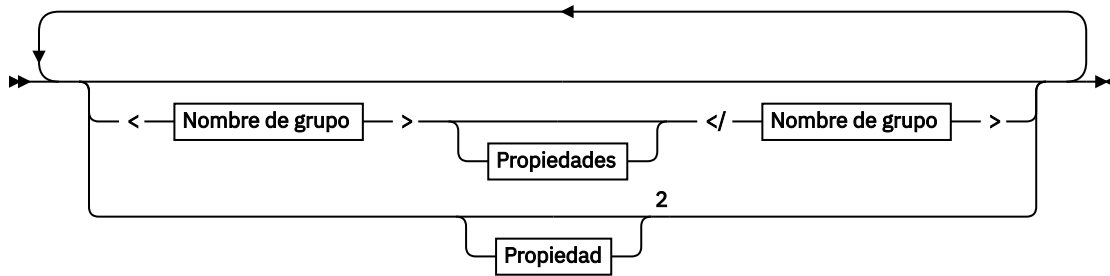
Nombre de elemento

Nombre

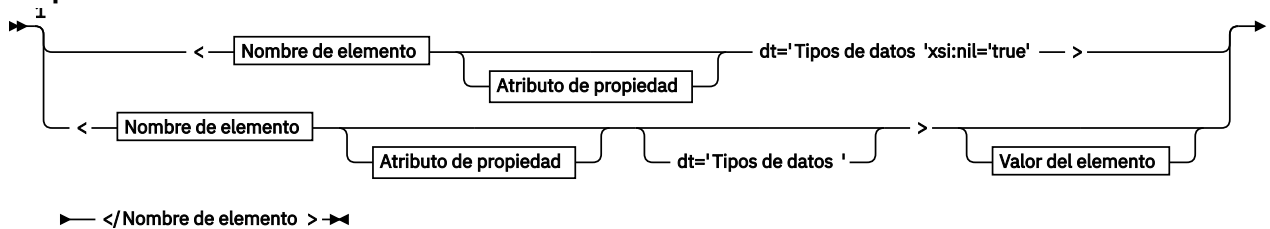
Valor del elemento



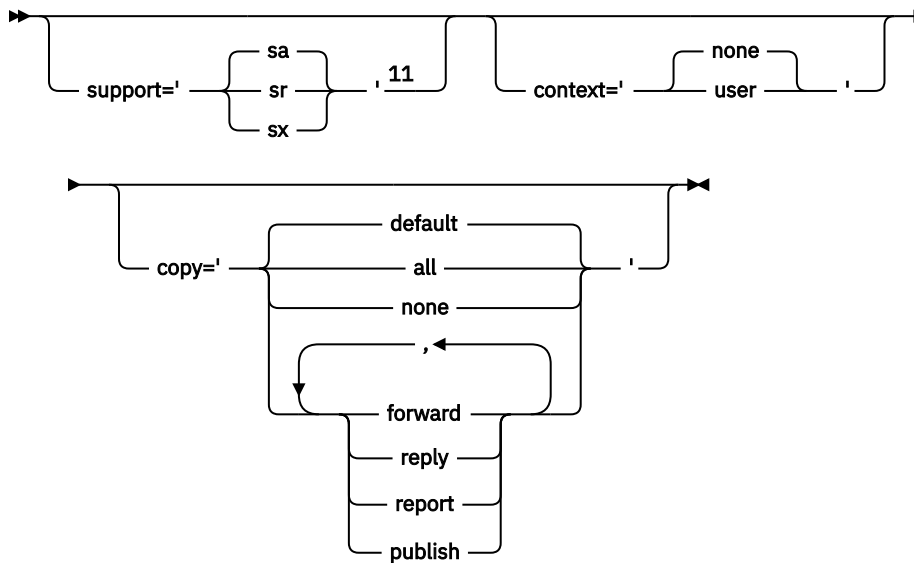
Propiedades



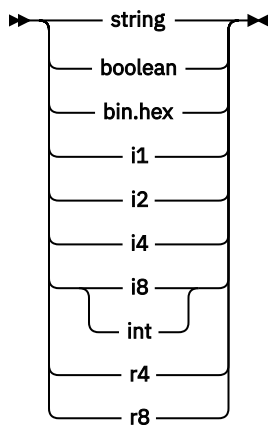
Propiedad



Atributo de propiedad



Tipos de datos



Notas:

- ¹ Las comillas dobles o simples son válidas.
- ² No utilices un nombre de propiedad no válido; consulta [Nombre de propiedad no válido](#). Utilice un nombre de propiedad reservado solo para el fin definido; consulte [Nombres de propiedades definidos](#).
- ³ El nombre debe estar en minúsculas.
- ⁴ Solo se admite una carpeta psc y pscr.
- ⁵ WebSphere Application Server El servicio Integration Bus ignora las carpetas sib, sib_context y sib_usr en los MQRFH2 encabezados posteriores, y solo las propiedades del primer MQRFH2 encabezado son significativas.
- ⁶ No debe haber más de una carpeta usr en un MQRFH2. Las propiedades de la carpeta usr no deben aparecer más de una vez.
- ⁷ Solo son importantes las propiedades de la primera carpeta mq. Si la carpeta es UTF-8, solo se admiten caracteres de un UTF-8 solo byte. El único carácter de espacio en blanco es Unicode U+0020.
- ⁸ Los caracteres válidos se definen en la especificación XML « W3C » (Sigüientes caracteres no permitidos en etiquetas) y consisten esencialmente en las categorías Unicode Ll, Lu, Lo, Lt, Nl, Mc, Mn, Lm, y Nd ; véase [Categorías de caracteres Unicode](#).
- ⁹ Todos los personajes son importantes. Los espacios en blanco al principio y al final forman parte del valor del elemento.
- ¹⁰ No utilice caracteres no válidos; consulte [Caracteres no válidos](#). Utilice una secuencia de escape en lugar de estos caracteres no válidos.
- ¹¹ El atributo de propiedad de soporte solo es válido en la carpeta mq.

Nombre de la carpeta

NameValueData contiene una sola carpeta. Para crear varias carpetas, cree varios *NameValueData* campos. Puede crear varios *NameValueData* campos en un solo MQRFH2 encabezado dentro de un mensaje. También puede crear varios encabezados MQRFH2 encadenados, cada uno con varios *NameValueData* campos.

El orden de MQRFH2 los encabezados y el orden de *NameValueData* los campos no influyen en el contenido lógico de una carpeta. Si la misma carpeta aparece más de una vez en un mensaje, la carpeta se analiza como un todo. Si la misma propiedad aparece en varias instancias de la misma carpeta, se analiza como una lista.

El análisis correcto de un no MQRFH2 se ve afectado por las diferentes formas en que una carpeta puede almacenarse físicamente en un mensaje.

Cuatro carpetas no siguen esta regla. Solo se analizan las primeras instancias de las mqcarpetas, sibsib_usr, sib_context y.

Si la misma propiedad aparece más de una vez en el contenido combinado de los encabezados MQRFH2 encadenados, solo se analiza la primera instancia de la propiedad. Si se establece una propiedad mediante una llamada a la API, como MQSETMP, y se añade a un MQRFH2 directamente desde una aplicación, la llamada a la API tiene prioridad.

El nombre de una carpeta es el nombre de una carpeta que contiene pares o grupos de nombre-valor. Los grupos y los pares nombre-valor se pueden mezclar en el mismo nivel en el árbol de carpetas; consulte [Figura 1 en la página 521](#). No combine un nombre de grupo y un nombre de elemento; consulte [Figura 2 en la página 521](#)

```
<group1><nvp1>value</nvp1></group1><group2><nvp2>value</nvp2></group2>  
<group3><nvp1>value</nvp1></group3><nvp3>value</nvp3>
```

Figura 1. Usos correctos de grupos y pares nombre-valor

```
<group1><nvp1> value </nvp1> value </group1>
```

Figura 2. Uso incorrecto de grupos y pares nombre-valor

No utilice un nombre de carpeta no válido o reservado; consulte [Nombre de ruta no válido](#) y [Nombre de carpeta o carpeta de propiedades reservado](#). Utilice un nombre de carpeta definido solo para el fin para el que ha sido definido; consulte [Nombre de carpeta definido](#).

Si añade el atributo `content='properties'` a la etiqueta del nombre de la carpeta, esta se convierte en una carpeta de propiedades; consulta [Figura 3 en la página 521](#).

```
<myFolder></myfolder>  
<myPropertyFolder content='properties'></myPropertyFolder>
```

Figura 3. Ejemplo de una carpeta y una carpeta de propiedades

Los nombres de las carpetas distinguen entre mayúsculas y minúsculas. Los nombres de las carpetas y los nombres de las carpetas de propiedades comparten el mismo espacio de nombres. Deben tener nombres diferentes. `Folder1` El nombre [Figura 4 en la página 521](#) debe ser diferente al `Folder2` nombre [Figura 5 en la página 521](#).

```
< Folder1 ><NVP1> value </NVP1></ Folder1 >
```

Figura 4. Espacio de nombres Folder1

```
< Folder2 content='properties'>< Property1 > value </ Property1 ></ Folder2 >
```

Figura 5. Espacio de nombres Folder2

Los grupos, las propiedades y los pares nombre-valor de diferentes carpetas tienen diferentes espacios de nombres. `Property1` en [Figura 5 en la página 521](#) es una propiedad diferente a `Property1` en [Figura 6 en la página 521](#).

```
<Folder3 content='properties'>< Property1 > value </ Property1 ></Folder3>
```

Figura 6. Espacio de nombres Folder3

Las carpetas de propiedades se diferencian de las carpetas que no son de propiedades en dos aspectos importantes:

1. Las carpetas de propiedades contienen propiedades, y las carpetas que no son de propiedades contienen pares nombre-valor. Las carpetas difieren ligeramente en cuanto a sintaxis.
2. Utilice las interfaces definidas, como las propiedades « MQI » o JMS las propiedades de los mensajes, para acceder a las propiedades de los mensajes. Las interfaces garantizan que las carpetas de propiedades en el MQRFH2 estén bien formadas. Una carpeta de propiedades bien formada es interoperable entre gestores de colas en diferentes plataformas y diferentes versiones.

La propiedad «message» de MQI es una forma fiable de leer y escribir un MQRFH2, y evita las dificultades que supone analizar un MQRFH2 correctamente.

Nombre de carpeta definido

Un nombre de carpeta definido es el nombre de una carpeta que está reservado para su uso por IBM MQ, u otro producto. No cree una carpeta con el mismo nombre y no añada sus propios pares nombre-valor a las carpetas. Las carpetas definidas son psc y pscx.

psc y pscx se utilizan en la publicación/suscripción en cola.

Un mensaje segmentado con o MQMF_SEGMENTATION_ALLOWEDMQMF_SEGMENT no puede contener un MQRFH2 con un nombre de carpeta definido. El MQPUT falla con el código de error 2443, MQRC_SEGMENTATION_NOT_ALLOWED.

Nombre de la carpeta de propiedades definida

Un nombre de carpeta de propiedades definido es el nombre de una carpeta de propiedades que utiliza IBM MQ, u otro producto. Para conocer los nombres de las carpetas y su contenido, consulte [Carpetas de propiedades](#). Los nombres de carpetas de propiedades definidos son un subconjunto de todos los nombres de carpetas reservados por IBM MQ; consulte [Nombre de carpeta reservado o nombre de carpeta de propiedades](#).

Cualquier elemento almacenado en una carpeta de propiedades definida es una propiedad. Un elemento almacenado en una carpeta de propiedades definida no debe tener un content= 'properties ' atributo.

Solo se pueden añadir propiedades a las carpetas de usrpropiedades definidas, mq_usr, y sib_usr. En otras carpetas de propiedades, como mq y sib, IBM MQ ignora o descarta las propiedades que no reconoce.

La descripción de cada carpeta de propiedades definida enumera las propiedades que IBM MQ ha definido y que pueden ser utilizadas por los programas de aplicación. A algunas de las propiedades se accede de forma indirecta estableciendo o obteniendo un JMS valor, mientras que a otras se accede directamente mediante las MQSETMP llamadas a ` MQI ` y MQINQMP ` getProperty `.

Las carpetas de propiedades definidas también contienen otras propiedades que IBM MQ tiene reservadas, pero a las que las aplicaciones no tienen acceso. Los nombres de las propiedades reservadas no se incluyen en la lista. No hay propiedades reservadas en las carpetas de sib_usr propiedades usr, mq_usr, y. Pero no cree propiedades con nombres de propiedad no válidos; consulte [Nombre de propiedad no válido](#).

Carpetas de propiedades

jms

jms contiene campos de cabecera JMS y propiedades JMSX que no se pueden expresar completamente en MQMD. La carpeta jms siempre está presente en un JMS MQRFH2.

<i>Tabla 541. Nombre de la propiedad jms, sinónimo, tipo de datos y carpeta</i>			
Sinónimo de propiedad	Nombre de propiedad	Tipo de datos	Carpeta
JMSDestination	jms.Dst	string	<jms><Dst> <i>destination</i> </Dst></jms>

Tabla 541. Nombre de la propiedad <i>jms</i> , sinónimo, tipo de datos y carpeta (continuación)			
Sinónimo de propiedad	Nombre de propiedad	Tipo de datos	Carpeta
JMSExpiration	jms.Exp	i8	<jms><Exp> <i>expiration</i> </Exp></jms>
Correlación JMS	jms.Cid	string	<jms><Cid> <i>correlationId</i> </Cid></jms>
Entrega JMS	jms.Dlv	i4	<jms><Dlv> <i>delivery</i> </Dlv></jms>
JMSPriority	jms.Pri	i4	<jms><Pri> <i>priority</i> </Pri></jms>
JMSReplyTo	jms.Rto	string	<jms><Rto> <i>replyToURI</i> </Rto></jms>
JMSTimestamp	jms.Tms	i8	<jms><Tms> <i>timestamp</i> </Tms></jms>
JMSXGroupID	jms.Gid	string	<jms><Gid> <i>groupId</i> </Gid></jms>
JMSXGroupSeq	jms.Seq	i4	<jms><Seq> <i>messageSequenceNo</i> </Seq></jms>

No añada sus propias propiedades en la carpeta *jms*.

mcd

mcd contiene propiedades que describen el formato del mensaje. Por ejemplo, la propiedad de dominio de servicio de mensajes *Msd* identifica un mensaje JMS como *JMSTextMessage*, *JMSBytesMessage*, *JMSStreamMessage*, *JMSMapMessage*, *JMSObjectMessage* o nulo.

La carpeta *mcd* siempre está presente en un mensaje JMS que contiene un *MQRFH2*.

Siempre está presente en un mensaje que contiene un *MQRFH2* enviado desde IBM Integration Bus. Describe el dominio, formato, tipo y conjunto de mensajes de un mensaje.

Tabla 542. <i>mcd</i> nombre de propiedad, sinónimo, tipo de datos y carpeta			
Sinónimo de propiedad	Nombre de propiedad	Tipo de datos	Carpeta
	mcd.Msd	string	<mcd><Msd> <i>messageDomain</i> </Msd></mcd>
	mcd.Set	string	<mcd><Set> <i>messageDomain</i> </Set></mcd>
	mcd.Type	string	<mcd><Type> <i>messageDomain</i> </Type></mcd>
	mcd.Fmt	string	<mcd><Fmt> <i>messageDomain</i> </Fmt></mcd>

No añada sus propias propiedades en la carpeta *mcd*.

mq_usr

mq_usr contiene propiedades definidas por la aplicación que no se exponen como propiedades JMS definidas por el usuario. Las propiedades que no cumplan JMS los requisitos pueden colocarse en esta carpeta.

Puede crear propiedades en la mq_usr carpeta. Las propiedades que creas en el mq_usr son como las propiedades que creas en carpetas nuevas con el content= 'properties ' atributo.

sib

sib contiene WebSphere Application Server las propiedades de los mensajes del sistema del WAS (SIB). sib Las propiedades no se exponen como JMS propiedades a IBM MQJMS las aplicaciones porque no son de los tipos compatibles. Por ejemplo, algunas sib propiedades no pueden exponerse como JMS propiedades porque son matrices de bytes. Algunas sib propiedades se exponen a las aplicaciones WAS /SIB como JMS_IBM_* propiedades; entre ellas se incluyen las propiedades de rutas de enrutamiento directas e inversas.

No añada sus propias propiedades en la carpeta sib.

sib_context

sib_context contiene propiedades de mensajes del sistema WAS /SIB que no están disponibles para las aplicaciones de usuario de WAS /SIB ni como JMS propiedades. sib_context contiene propiedades de seguridad y transaccionales que se utilizan para los servicios web.

No añada sus propias propiedades en la carpeta sib_context.

sib_usr

sib_usr contiene propiedades de mensajes de usuario de tipo « WAS » o «SIB» que no se muestran como JMS propiedades de usuario, ya que no pertenecen a los tipos admitidos. sib_usr está expuesto a las aplicaciones de integración de servicios (WAS, SIB) en la SIMessage interfaz; véase «Desarrollo de la integración de servicios».

El tipo de una sib_usr propiedad debe ser bin . hex, y el valor debe estar en el formato correcto. Si una IBM MQ aplicación escribe un bin . hex elemento tipado en la carpeta con un formato incorrecto, la aplicación recibe un IOException. Si el tipo de datos de la propiedad no es bin . hex la aplicación recibe un ClassCastException.

No intente hacer JMS que las propiedades de usuario estén disponibles para WAS /SIB utilizando esta carpeta; utilice, en su lugar, la usr carpeta.

Puede crear propiedades en la sib_usr carpeta.

usr

usr contiene propiedades JMS definidas por la aplicación asociadas al mensaje. La carpeta usr solo está presente si una aplicación ha establecido una propiedad definida por la aplicación.

usr es la carpeta de propiedades predeterminada. Si se establece una propiedad sin nombre de carpeta, se coloca en la usr carpeta.

Tabla 543. Nombre de la propiedad usr, sinónimo, tipo de datos y carpeta			
Sinón imo de propi edad	Nombre de propiedad	Tipo de dato s	Carpeta
	usr.contentType	stri ng	<usr><contentType>text/xml; charset=utf-8</ contentType></usr>
	usr.endpointURL	stri ng	<usr><endpointURL> URI </endpointURL></usr>

Tabla 543. Nombre de la propiedad *usr*, sinónimo, tipo de datos y carpeta (continuación)

Sinónimo de propiedad	Nombre de propiedad	Tipo de datos	Carpeta
	usr.targetService	string	<usr><targetService> serviceName </targetService></usr>
	usr.soapAction	string	<usr><soapAction> name </soapAction></usr>
	usr.transportVersion	string	<usr><transportVersion> version </transportVersion></usr>

Puede crear propiedades en la *usr* carpeta.

Un mensaje segmentado con o MQMF_SEGMENTATION_ALLOWEDMQMF_SEGMENT no puede contener un MQRFH2 con un nombre de carpeta de propiedad definido. El MQPUT falla con el código de error 2443, MQRC_SEGMENTATION_NOT_ALLOWED.

Nombre de la carpeta de propiedades sin agrupar

ibm

ibm contiene propiedades que solo utiliza IBM MQ.

Tabla 544. <i>ibm</i> nombre de propiedad, sinónimo, tipo de datos y carpeta			
Sinónimo de propiedad	Nombre de propiedad	Tipo de datos	Carpeta
	ibm.rfp	string	<ibm><rfp>fingerprint</rfp></ibm>

No añada sus propias propiedades en la carpeta *ibm*.

mq

mq contiene propiedades que solo utiliza IBM MQ.

Las siguientes restricciones se aplican a las propiedades de la *mq* carpeta:

- MQ solo actúa sobre las propiedades de la primera carpeta *mq* significativa del mensaje; las propiedades de cualquier otra *mq* carpeta del mensaje se ignoran.
- Solo se permiten caracteres de un solo byte UTF-8 en la carpeta. Un carácter multibyte en la carpeta puede provocar que el análisis falle y que el mensaje sea rechazado.
- No utilices cadenas de escape en la carpeta. Una cadena de escape se trata como el valor real del elemento.
- Solo los caracteres Unicode U+0020 se tratan como espacios en blanco dentro de la carpeta. Todos los demás caracteres se tratan como significativos y pueden provocar que el análisis de la carpeta falle y que el mensaje sea rechazado.

Si el análisis de la *mq* carpeta falla, o si la carpeta no cumple estas restricciones, el mensaje se rechaza con el código 2527 de motivo. MQRC_RFH_RESTRICTED_FORMAT_ERR

No añada sus propias propiedades en la carpeta *mq*.

mqema

mqema contiene propiedades que solo utiliza WebSphere Application Server. La carpeta ha sido sustituida por *mqext*.

No añada sus propias propiedades en la carpeta mqema.

mqext

mqext contiene los siguientes tipos de propiedad:

- Propiedades que únicamente utiliza WebSphere Application Server.
- Propiedades relacionadas con la entrega retardada de mensajes.

La carpeta está presente si la aplicación ha establecido como mínimo una de las propiedades definidas por IBM o ha utilizado el retardo el retardo de entrega.

Tabla 545. mqext nombre de propiedad, sinónimo, tipo de datos y carpeta			
Sinónimo de propiedad	Nombre de propiedad	Tipo de datos	Carpeta
JMSArmCorrelator	mqext.Arm	string	<mqext><Arm>armCorrelator</Arm></mqext>
JMSRMCorrelator	mqext.Wrm	string	<mqext><Wrm>wrmCorrelator</Wrm></mqext>
JMSDeliveryTime	mqext.Dlt	i8	<mqext><Dlt>DeliveryTime</Dlt></mqext>
JMSDeliveryDelay	mqext.Dly	i8	<mqext><Dly>DeliveryTime</Dly></mqext>

No añada sus propias propiedades en la carpeta mqext.

mmps

mmps contiene propiedades que solo son utilizadas por la publicación/suscripción de IBM MQ. La carpeta sólo está presente si la aplicación ha establecido al menos una de las propiedades de publicación/suscripción integradas.

Tabla 546. mmps nombre de propiedad, sinónimo, tipo de datos y carpeta			
Sinónimo de propiedad	Nombre de propiedad	Tipo de datos	Carpeta
MQTopicString	mmps.Top	string	<mmps><Top>topicString</Top></mmps>
MQSubUserData	mmps.Sud	string	<mmps><Sud>subscriberUserData...</Sud></mmps>
MQIsRetained	mmps.Ret	boolean	<mmps><Ret>isRetained</Ret></mmps>
MQPubOptions	mmps.Pub	i8	<mmps><Pub>publicationOptions</Pub></mmps>
MQPubLevel	mmps.Pbl	i8	<mmps><Pbl>publicationLevel</Pbl></mmps>
MQPubTime	mmpse.Pts	string	<mmps><Pts>publicationTime</Pts></mmps>
MQPubSeqNum	mmpse.Seq	i8	<mmps><Seq>publicationSequenceNumber</Seq></mmps>
MQPubStrIntData	mmpse.Sid	string	<mmps><Sid>publicationData</Sid></mmps>
MQPubFormat	mmpse.Pfmt	i8	<mmps><Pfmt>messageFormat</Pfmt></mmps>

No añada sus propias propiedades en la carpeta mmps.

mq_svc

mq_svc Contiene propiedades utilizadas por SupportPac MA93.

No añada sus propias propiedades en la carpeta mq_svc.

mqtt

mqtt contiene propiedades utilizadas por IBM MQ Telemetry

Tabla 547. Nombre de la propiedad mqtt, sinónimo, tipo de datos y carpeta			
Sinónimo de propiedad	Nombre de propiedad	Tipo de datos	Carpeta
	mqtt.clientId	string	<mqtt><clientId> <i>topicString</i> </clientId></mqtt>
	mqtt.qos	i4	<mqtt><qos> <i>qualityOfService</i> </qos></mqtt>
	mqtt.msgid	string	<mqtt><msgid> <i>messageIdentifier</i> </msgid></mqtt>

No añada sus propias propiedades en la carpeta mqtt.

Un mensaje segmentado con o MQMF_SEGMENTATION_ALLOWEDMQMF_SEGMENT no puede contener un MQRFH2 con un nombre de carpeta de propiedad no agrupado. El MQPUT falla con el código de error 2443, MQRC_SEGMENTATION_NOT_ALLOWED.

Pares nombre-valor

En el diagrama sintáctico, "los pares nombre-valor" describen el contenido de una carpeta normal. Una carpeta normal contiene grupos y elementos. Un elemento es un par nombre-valor. Un grupo contiene elementos y otros grupos.

En términos de árboles, los elementos son nodos hoja y los grupos son nodos internos. Un nodo interno y la carpeta, que es el nodo raíz, pueden contener una mezcla de nodos internos y nodos hoja. Un nodo no puede ser al mismo tiempo un nodo interno y un nodo hoja; véase [Figura 2 en la página 521](#).

Propiedades

En el diagrama sintáctico, "Propiedades" describe el contenido de una carpeta de propiedades. Una carpeta de propiedades contiene grupos y propiedades. Una propiedad es un par nombre-valor con un atributo de tipo de datos opcional. Un grupo contiene propiedades y otros grupos.

En términos de árboles, las propiedades son nodos hoja y los grupos son nodos internos. Un nodo interno y la carpeta de propiedades, que es el nodo raíz, pueden contener una mezcla de nodos internos y nodos hoja. Un nodo no puede ser al mismo tiempo un nodo interno y un nodo hoja; véase [Figura 2 en la página 521](#).

Propiedad

Una propiedad de mensaje es un par nombre-valor en una carpeta de propiedades. Opcionalmente, puede incluir un atributo de tipo de datos y un atributo de propiedad; para ver un ejemplo, consulte el siguiente código. Si se omite el atributo de tipo de datos, el tipo de propiedad es `string`.

```
<pf><p1 dt='i8' > value </p1></pf>
```

El nombre de una propiedad de mensaje es su ruta completa, con la sintaxis similar <> a XML sustituida por puntos. Por ejemplo, `myPropertyFolder1.myGroup1.myGroup2.myProperty1` se asigna a una `NameValueData` cadena de la siguiente manera. La cadena está formateada para facilitar su lectura.

```
<myPropertyFolder1>
  <myGroup1>
    <myGroup2>
      <myProperty1>value</myProperty1>
    </myGroup2>
  </myGroup1>
</myPropertyFolder1>
```

Una carpeta de propiedades puede contener varias propiedades. Por ejemplo, las propiedades de [Figura 7](#) en la [página 528](#) se asignan a la carpeta de propiedades en Cambios para aislar la cola de ventas en un nuevo clúster y separar las colas de transmisión del clúster de puerta de enlace.

```
myPropertyFolder1.myProperty4  
myPropertyFolder1.myGroup1.myGroup2.myProperty1  
myPropertyFolder1.myGroup1.myGroup2.myProperty2  
myPropertyFolder1.myGroup1.myProperty3
```

Figura 7. Varias propiedades con el mismo nombre raíz

```
<myPropertyFolder1>  
  <myProperty4>value</myProperty4>  
  <myGroup1>  
    <myGroup2>  
      <myProperty1>value</myProperty1>  
      <myProperty2>value</myProperty2>  
    </myGroup2>  
    <myProperty3>value</myProperty3>  
  </myGroup1>  
</myPropertyFolder1>
```

Figura 8. Asignación de nombres de propiedades múltiples

Nombre

Un nombre debe comenzar con una *letra* o un *guión bajo*. No debe contener dos *puntos*, no debe terminar en *punto* y solo debe contener *letras*, *números*, *guiones bajos*, *guiones* y *puntos*. Los caracteres válidos se definen en la especificación XML « W3C » (Siguiendo caracteres no permitidos en etiquetas) y consisten esencialmente en las categorías Unicode L1, Lu, Lo, Lt, N1, Mc, Mn, Lm, y Nd ; véase [Categorías de caracteres Unicode](#).

La ruta completa de una propiedad o un par nombre-valor no debe infringir la regla descrita en [Nombre de ruta no válido](#). Las rutas están limitadas a 4095 bytes, no deben contener caracteres compatibles con Unicode y no deben comenzar con la cadena XML.

Nombre del grupo

El nombre de un grupo tiene la misma sintaxis que un nombre. Los nombres de los grupos son opcionales. Las propiedades y los pares nombre-valor se pueden colocar en la raíz de una carpeta. Utiliza grupos si te ayuda a organizar propiedades y pares nombre-valor.

Nombre del elemento

El nombre de un elemento tiene la misma sintaxis que un nombre.

Valor del elemento

El valor de un elemento incluye todos los espacios en blanco entre la < *Element name* > etiqueta y la etiqueta < /*Element name* >. No utilice los dos caracteres < y & en un valor. Reemplácelos por < y & ;.

Atributos de la propiedad

Los atributos de la propiedad asignan campos descriptores de la propiedad: Las asignaciones son las siguientes:

Soporte

sa (predeterminado)

MQPD_SUPPORT_OPTIONAL

sr

MQPD_SUPPORT_REQUIRED

sx
MQPD_SUPPORT_REQUIRED_IF_LOCAL

Contexto

ninguno (predeterminado)
MQPD_NO_CONTEXT

usuario
MQPD_USER _CONTEXT

CopyOptions

hacia adelante
MQPD_COPY_FORWARD

responder
MQPD_COPY_REPLY

informe
MQPD_COPY_REPORT

publicar
MQPD_COPY_PUBLISH

todo
MQPD_COPY_ALL

No utilice todas las opciones en combinación con otras.

valor predeterminado
MQPD_COPY_DEFAULT

No utilice el valor predeterminado en combinación con otras opciones. El valor predeterminado es el mismo que reenviar + informar + publicar.

ninguno
MQPD_COPY_NONE

No utilice none en combinación con otras opciones.

Los atributos de la propiedad Support solo son aplicables a las propiedades de la mq carpeta.

Los atributos Context y CopyOptions property son aplicables a todas las carpetas de propiedades.

Tipos de datos

MQRFH2 Los tipos de datos se asignan a los tipos de propiedades de los mensajes de la siguiente manera:

Tabla 548. Correlaciones de tipos de datos	
Tipo de datos de MQRFH2	Tipo de propiedad del mensaje
bin.hex	MQBYTE[]
boolean	MQBOOL
i1	MQINT8
i2	MQINT16
i4	MQINT32
i8	MQINT64
r4	MQFLOAT32
r8	MQFLOAT64
string	MQCHAR[]

Cualquier elemento sin un tipo de datos se considera de tipo string.

Un valor nulo se indica mediante el atributo del elemento `xsi:nil='true'`. No utilice el atributo `xsi:nil='false'` para valores no nulos. Por ejemplo, la siguiente propiedad tiene un valor nulo:

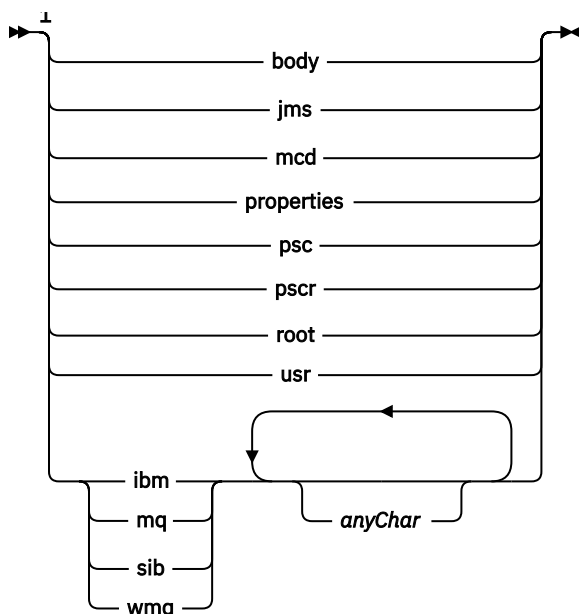
```
<NullProperty
  xsi:nil='true'></NullProperty>
```

Una propiedad de tipo byte o cadena de caracteres puede tener un valor vacío. Un valor vacío se representa mediante un MQRFH2 elemento con un valor de elemento de longitud cero. Por ejemplo, la siguiente propiedad tiene un valor vacío:

```
<EmptyProperty></EmptyProperty>
```

Nombre de la carpeta reservada o carpeta de propiedad

Restringir el nombre de una carpeta o carpeta de propiedades para que no comience con ninguna de las siguientes cadenas. Los prefijos están reservados para nombres de carpetas o propiedades creados por IBM.

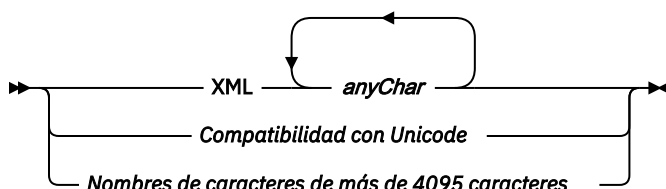


Notas:

¹ Una carpeta reservada o un nombre de propiedad contiene cualquier combinación de letras mayúsculas y minúsculas.

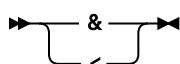
Nombre de vía de acceso no válido

Restringe la ruta completa de un par nombre-valor o una propiedad para que no incluya ninguna de las siguientes cadenas.



Caracteres no válidos

Utilice siempre las secuencias de escape `&` y `<` en lugar de los literales `"&"` y `"<"`.

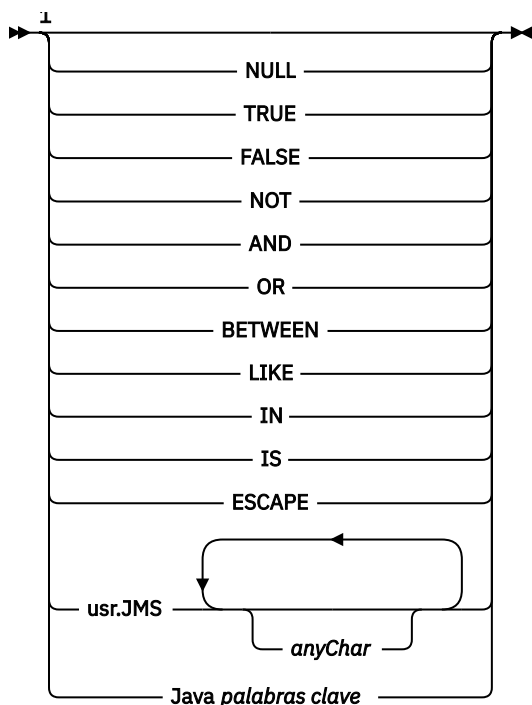


Nombres de propiedades definidos

Los nombres de propiedades definidos son los nombres de las propiedades que están definidas por IBM MQ, u otros productos, y que utilizan IBM MQ y las aplicaciones de los usuarios. Las propiedades definidas solo existen en carpetas de propiedades definidas. Los nombres de las propiedades definidas se describen en la descripción de las carpetas de propiedades; consulte [Carpetas de propiedades](#).

Nombre de propiedad no válido

No construyas nombres de propiedades que cumplan la siguiente regla. La regla se aplica a la ruta completa de la propiedad que nombra una propiedad, y no solo al nombre del elemento de la propiedad.



Notas:

¹ Un nombre de propiedad no válido puede contener cualquier combinación de mayúsculas y minúsculas.

Atributos no válidos

Las carpetas de propiedades y las propiedades solo pueden incluir [atributos de propiedad](#) y [tipos de datos](#) compatibles.

Cualquier atributo similar a XML no compatible, por ejemplo, nombres con valores de cadena entre comillas, que se incluyan en carpetas de propiedades o propiedades, podría eliminarse.

Atributos similares a XML incluidos en carpetas que no son de propiedades o elementos que no son de propiedades que permanecen en encabezados MQRFH2.

MQRMH - Cabecera de mensaje de referencia

La estructura MQRMH define el formato del encabezado de un mensaje de referencia. Este encabezado se utiliza con salidas de canal de mensajes escritas por el usuario para enviar cantidades extremadamente grandes de datos (denominados *datos masivos*) de un gestor de colas a otro. La diferencia con respecto a la mensajería normal es que los datos masivos no se almacenan en una cola, sino que solo se almacena una *referencia* a los datos masivos en la cola. Esto reduce la posibilidad de que IBM MQ los recursos se agoten por un pequeño número de mensajes extremadamente grandes.

Disponibilidad

La estructura MQRMH está disponible en las siguientes plataformas:

-  AIX
-  IBM i
-  Linux
-  Windows

y para IBM MQ los clientes conectados a estos sistemas.

Nombre de formato

MQFMT_REF_MSG_HEADER

Conjunto de caracteres y codificación

Los datos de caracteres en MQRMH y las cadenas a las que se refieren los campos de desplazamiento deben estar en el juego de caracteres del gestor de colas local; esto viene dado por el atributo del **CodedCharSetId** gestor de colas. Los datos numéricos en MQRMH deben estar en la codificación nativa de la máquina; esto viene dado por el valor de MQENC_NATIVE para el lenguaje de programación C.

Configure el juego de caracteres y la codificación del MQRMH en los campos *CodedCharSetId* *Encoding* y en:

- El MQMD (si la estructura MQRMH se encuentra al inicio de los datos del mensaje), o
- La estructura de encabezado que precede a la estructura MQRMH (todos los demás casos).

Utilización

Una aplicación envía un mensaje compuesto por un MQRMH, pero omitiendo los datos masivos. Cuando un agente de canal de mensajes (MCA) lee el mensaje de la cola de transmisión, se invoca una salida de mensaje proporcionada por el usuario para procesar el encabezado del mensaje de referencia. La salida puede añadir al mensaje de referencia los datos masivos identificados por la estructura MQRMH, antes de que el MCA envíe el mensaje a través del canal al siguiente gestor de colas.

En el extremo receptor, debe existir una salida de mensajes que espere los mensajes de referencia. Cuando se recibe un mensaje de referencia, la salida debe crear el objeto a partir de los datos masivos que siguen al MQRMH en el mensaje y, a continuación, transmitir el mensaje de referencia sin los datos masivos. El mensaje de referencia puede recuperarse posteriormente mediante una aplicación que lea el mensaje de referencia (sin los datos masivos) de una cola.

Normalmente, la estructura MQRMH es todo lo que hay en el mensaje. Sin embargo, si el mensaje se encuentra en una cola de transmisión, uno o más encabezados adicionales preceden a la estructura MQRMH.

También se puede enviar un mensaje de referencia a una lista de distribución. En este caso, la estructura MQDH y sus registros relacionados preceden a la estructura MQRMH cuando el mensaje se encuentra en una cola de transmisión.

Nota: No envíe un mensaje de referencia como mensaje segmentado, ya que la salida de mensajes no puede procesarlo correctamente.

Conversión de datos

A efectos de conversión de datos, la conversión de la estructura MQRMH incluye la conversión de los datos del entorno de origen, el nombre del objeto de origen, los datos del entorno de destino y el nombre del objeto de destino. Cualquier otro byte dentro de *StrucLength* los bytes del inicio de la estructura se descarta o tiene valores indefinidos después de la conversión de datos. Los datos masivos se convierten siempre que se cumplan todas las siguientes condiciones:

- Los datos masivos están presentes en el mensaje cuando se realiza la conversión de datos.

- El *Format* campo en MQRMH tiene un valor distinto de MQFMT_NONE.
- Existe una salida de conversión de datos escrita por el usuario con el nombre de formato especificado.

Sin embargo, tenga en cuenta que, por lo general, los datos masivos no están presentes en el mensaje cuando este se encuentra en una cola y que, como resultado, los datos masivos se convierten mediante la opción MQGMO_CONVERT.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 549. Campos en MQRMH para MQRMH		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>StrucId</u> (identificador de estructura)	MQRMH_STRUC_ID	'RMH↵'
<u>Versión</u> (número de versión de la estructura)	MQRMH_VERSION_1	1
<u>StrucLength</u> (longitud total de MQRMH, incluidas las cadenas al final de los campos fijos, pero sin los datos masivos)	Ninguna	0
<u>Codificación</u> (codificación numérica de datos masivos)	MQENC_NATIVE	Depende del entorno
<u>CodedCharSetId</u> (identificador del conjunto de caracteres de datos masivos)	MQCCSI_INDETERMINADO	0
<u>Formato</u> (nombre del formato de los datos masivos)	MQFMT_NONE	Espacios en blanco
<u>Indicadores</u> (indicadores de mensajes de referencia)	MQRMHF_NO_ÚLTIMO	0
<u>ObjectType</u> (tipo de objeto)	Ninguna	Espacios en blanco
<u>ObjectInstanceId</u> (identificador de instancia de objeto)	MQOII_NINGUNO	Nulos
<u>SrcEnvLength</u> (longitud de los datos del entorno de origen)	Ninguna	0
<u>SrcEnvOffset</u> (desviación de los datos del entorno de origen)	Ninguna	0
<u>SrcNameLength</u> (longitud del nombre del objeto fuente)	Ninguna	0
<u>SrcNameOffset</u> (desplazamiento del nombre del objeto fuente)	Ninguna	0
<u>DestEnvLength</u> (longitud de los datos del entorno de destino)	Ninguna	0
<u>DestEnvOffset</u> (desviación de los datos del entorno de destino)	Ninguna	0
<u>DestNameLength</u> (longitud del nombre del objeto de destino)	Ninguna	0
<u>DestNameOffset</u> (desplazamiento del nombre del objeto de destino)	Ninguna	0
<u>DataLogicalLength</u> (longitud de los datos masivos)	Ninguna	0

Tabla 549. Campos en MQRMH para MQRMH (continuación)

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
DataLogicalOffset (bajo desplazamiento de datos masivos)	Ninguna	0
DataLogicalOffset2 (alto desplazamiento de datos masivos)	Ninguna	0
Notas: 1. El símbolo ↵ representa un único carácter en blanco. 2. En el lenguaje de programación C, la variable macroMQRMH_DEFAULT contiene los valores que se enumeran en la tabla. Úselo de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQRMH MyRMH = {MQRMH_DEFAULT};</pre>		

Declaraciones lingüísticas

Declaración C para MQRMH

```
typedef struct tagMQRMH MQRMH;
struct tagMQRMH {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQLONG     StrucLength;       /* Total length of MQRMH, including
                                strings at end of fixed fields, but
                                not the bulk data */
    MQLONG     Encoding;         /* Numeric encoding of bulk data */
    MQLONG     CodedCharSetId;   /* Character set identifier of bulk
                                data */
    MQCHAR8    Format;           /* Format name of bulk data */
    MQLONG     Flags;            /* Reference message flags */
    MQCHAR8    ObjectType;       /* Object type */
    MQBYTE24   ObjectInstanceId; /* Object instance identifier */
    MQLONG     SrcEnvLength;      /* Length of source environment data */
    MQLONG     SrcEnvOffset;     /* Offset of source environment data */
    MQLONG     SrcNameLength;     /* Length of source object name */
    MQLONG     SrcNameOffset;    /* Offset of source object name */
    MQLONG     DestEnvLength;     /* Length of destination environment
                                data */
    MQLONG     DestEnvOffset;    /* Offset of destination environment
                                data */
    MQLONG     DestNameLength;    /* Length of destination object name */
    MQLONG     DestNameOffset;   /* Offset of destination object name */
    MQLONG     DataLogicalLength; /* Length of bulk data */
    MQLONG     DataLogicalOffset; /* Low offset of bulk data */
    MQLONG     DataLogicalOffset2; /* High offset of bulk data */
};
```

COBOL declaración para MQRMH

```
** MQRMH structure
10 MQRMH.
** Structure identifier
15 MQRMH-STRUCID PIC X(4).
** Structure version number
15 MQRMH-VERSION PIC S9(9) BINARY.
** Total length of MQRMH, including strings at end of fixed fields,
** but not the bulk data
15 MQRMH-STRUCLNGTH PIC S9(9) BINARY.
** Numeric encoding of bulk data
15 MQRMH-ENCODING PIC S9(9) BINARY.
** Character set identifier of bulk data
15 MQRMH-CODEDCHARSETID PIC S9(9) BINARY.
** Format name of bulk data
15 MQRMH-FORMAT PIC X(8).
** Reference message flags
```

```

15 MQRMH-FLAGS PIC S9(9) BINARY.
** Object type
15 MQRMH-OBJECTTYPE PIC X(8).
** Object instance identifier
15 MQRMH-OBJECTINSTANCEID PIC X(24).
** Length of source environment data
15 MQRMH-SRCENVLENGTH PIC S9(9) BINARY.
** Offset of source environment data
15 MQRMH-SRCENVOFFSET PIC S9(9) BINARY.
** Length of source object name
15 MQRMH-SRCNAMELENGTH PIC S9(9) BINARY.
** Offset of source object name
15 MQRMH-SRCNAMEOFFSET PIC S9(9) BINARY.
** Length of destination environment data
15 MQRMH-DESTENVLENGTH PIC S9(9) BINARY.
** Offset of destination environment data
15 MQRMH-DESTENVOFFSET PIC S9(9) BINARY.
** Length of destination object name
15 MQRMH-DESTNAMELENGTH PIC S9(9) BINARY.
** Offset of destination object name
15 MQRMH-DESTNAMEOFFSET PIC S9(9) BINARY.
** Length of bulk data
15 MQRMH-DATALOGICALLLENGTH PIC S9(9) BINARY.
** Low offset of bulk data
15 MQRMH-DATALOGICALOFFSET PIC S9(9) BINARY.
** High offset of bulk data
15 MQRMH-DATALOGICALOFFSET2 PIC S9(9) BINARY.

```

PL/I declaración para MQRMH

```

dcl
1 MQRMH based,
3 StrucId char(4), /* Structure identifier */
3 Version fixed bin(31), /* Structure version number */
3 StrucLength fixed bin(31), /* Total length of MQRMH,
                               including strings at end of
                               fixed fields, but not the bulk
                               data */
3 Encoding fixed bin(31), /* Numeric encoding of bulk
                             data */
3 CodedCharSetId fixed bin(31), /* Character set identifier of
                                  bulk data */
3 Format char(8), /* Format name of bulk data */
3 Flags fixed bin(31), /* Reference message flags */
3 ObjectType char(8), /* Object type */
3 ObjectInstanceId char(24), /* Object instance identifier */
3 SrcEnvLength fixed bin(31), /* Length of source environment
                                data */
3 SrcEnvOffset fixed bin(31), /* Offset of source environment
                                data */
3 SrcNameLength fixed bin(31), /* Length of source object name */
3 SrcNameOffset fixed bin(31), /* Offset of source object name */
3 DestEnvLength fixed bin(31), /* Length of destination
                                environment data */
3 DestEnvOffset fixed bin(31), /* Offset of destination
                                environment data */
3 DestNameLength fixed bin(31), /* Length of destination object
                                name */
3 DestNameOffset fixed bin(31), /* Offset of destination object
                                name */
3 DataLogicalLength fixed bin(31), /* Length of bulk data */
3 DataLogicalOffset fixed bin(31), /* Low offset of bulk data */
3 DataLogicalOffset2 fixed bin(31); /* High offset of bulk data */

```

High Level Assembler declaración para MQRMH

MQRMH	DSECT		
MQRMH_STRUCID	DS	CL4	Structure identifier
MQRMH_VERSION	DS	F	Structure version number
MQRMH_STRUCLength	DS	F	Total length of MQRMH, including
*			strings at end of fixed fields, but
*			not the bulk data
MQRMH_ENCODING	DS	F	Numeric encoding of bulk data
MQRMH_CODEDCHARSETID	DS	F	Character set identifier of bulk
*			data
MQRMH_FORMAT	DS	CL8	Format name of bulk data
MQRMH_FLAGS	DS	F	Reference message flags
MQRMH_OBJECTTYPE	DS	CL8	Object type
MQRMH_OBJECTINSTANCEID	DS	XL24	Object instance identifier

MQRMH_SRCENVLENGTH	DS	F	Length of source environment data
MQRMH_SRCENVOFFSET	DS	F	Offset of source environment data
MQRMH_SRCNAMELENGTH	DS	F	Length of source object name
MQRMH_SRCNAMEOFFSET	DS	F	Offset of source object name
MQRMH_DESTENVLENGTH	DS	F	Length of destination environment data
*			
MQRMH_DESTENVOFFSET	DS	F	Offset of destination environment data
*			
MQRMH_DESTNAMELENGTH	DS	F	Length of destination object name
MQRMH_DESTNAMEOFFSET	DS	F	Offset of destination object name
MQRMH_DATALOGICALLLENGTH	DS	F	Length of bulk data
MQRMH_DATALOGICALOFFSET	DS	F	Low offset of bulk data
MQRMH_DATALOGICALOFFSET2	DS	F	High offset of bulk data
*			
MQRMH_LENGTH	EQU	*-MQRMH	
	ORG	MQRMH	
MQRMH_AREA	DS	CL(MQRMH_LENGTH)	

Declaración de Visual Basic para MQRMH

```

Type MQRMH
  StrucId          As String*4 'Structure identifier'
  Version          As Long     'Structure version number'
  StrucLength      As Long     'Total length of MQRMH, including'
                                'strings at end of fixed fields, but'
                                'not the bulk data'
  Encoding         As Long     'Numeric encoding of bulk data'
  CodedCharSetId   As Long     'Character set identifier of bulk data'
  Format           As String*8 'Format name of bulk data'
  Flags            As Long     'Reference message flags'
  ObjectType       As String*8 'Object type'
  ObjectInstanceId As MQBYTE24 'Object instance identifier'
  SrcEnvLength     As Long     'Length of source environment data'
  SrcEnvOffset     As Long     'Offset of source environment data'
  SrcNameLength    As Long     'Length of source object name'
  SrcNameOffset    As Long     'Offset of source object name'
  DestEnvLength    As Long     'Length of destination environment'
                                'data'
  DestEnvOffset    As Long     'Offset of destination environment'
                                'data'
  DestNameLength   As Long     'Length of destination object name'
  DestNameOffset   As Long     'Offset of destination object name'
  DataLogicalLength As Long     'Length of bulk data'
  DataLogicalOffset As Long     'Low offset of bulk data'
  DataLogicalOffset2 As Long    'High offset of bulk data'
End Type

```

Detalles de campo para MQRMH

La estructura del encabezado del mensaje de referencia MQRMH define el formato del encabezado de un mensaje de referencia. Este encabezado se utiliza con salidas de canal de mensajes escritas por el usuario para enviar cantidades extremadamente grandes de datos (denominados *datos masivos*) de un gestor de colas a otro. La diferencia con respecto a la mensajería normal es que los datos masivos no se almacenan en una cola, sino que solo se almacena una *referencia* a los datos masivos en la cola. Esto reduce la posibilidad de que IBM MQ los recursos se agoten por un pequeño número de mensajes extremadamente grandes.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 550. Campos en MQRMH para MQRMH		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>StrucId</u> (identificador de estructura)	MQRMH_STRUC_ID	'RMH¬'
<u>Versión</u> (número de versión de la estructura)	MQRMH_VERSION_1	1

Tabla 550. Campos en MQRMH para MQRMH (continuación)

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>StrucLength</u> (longitud total de MQRMH, incluidas las cadenas al final de los campos fijos, pero sin los datos masivos)	Ninguna	0
<u>Codificación</u> (codificación numérica de datos masivos)	MQENC_NATIVE	Depende del entorno
<u>CodedCharSetId</u> (identificador del conjunto de caracteres de datos masivos)	MQCCSI_INDETERMINADO	0
<u>Formato</u> (nombre del formato de los datos masivos)	MQFMT_NONE	Espacios en blanco
<u>Indicadores</u> (indicadores de mensajes de referencia)	MQRMHF_NO_ÚLTIMO	0
<u>ObjectType</u> (tipo de objeto)	Ninguna	Espacios en blanco
<u>ObjectInstanceId</u> (identificador de instancia de objeto)	MQOII_NINGUNO	Nulos
<u>SrcEnvLength</u> (longitud de los datos del entorno de origen)	Ninguna	0
<u>SrcEnvOffset</u> (desviación de los datos del entorno de origen)	Ninguna	0
<u>SrcNameLength</u> (longitud del nombre del objeto fuente)	Ninguna	0
<u>SrcNameOffset</u> (desplazamiento del nombre del objeto fuente)	Ninguna	0
<u>DestEnvLength</u> (longitud de los datos del entorno de destino)	Ninguna	0
<u>DestEnvOffset</u> (desviación de los datos del entorno de destino)	Ninguna	0
<u>DestNameLength</u> (longitud del nombre del objeto de destino)	Ninguna	0
<u>DestNameOffset</u> (desplazamiento del nombre del objeto de destino)	Ninguna	0
<u>DataLogicalLength</u> (longitud de los datos masivos)	Ninguna	0
<u>DataLogicalOffset</u> (bajo desplazamiento de datos masivos)	Ninguna	0
<u>DataLogicalOffset2</u> (alto desplazamiento de datos masivos)	Ninguna	0

Tabla 550. Campos en MQRMH para MQRMH (continuación)

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
Notas: 1. El símbolo ~ representa un único carácter en blanco. 2. En el lenguaje de programación C, la variable macroMQRMH_DEFAULT contiene los valores que se enumeran en la tabla. Úselo de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQRMH MyRMH = {MQRMH_DEFAULT};</pre>		

StrucId (MQCHAR4)

Este es el identificador de estructura de la estructura del encabezado del mensaje de referencia. Siempre es un campo de entrada. Su valor es MQRMH_STRUC_ID.

El valor debe ser:

MQRMH_STRUC_ID

Identificador de la estructura del encabezado del mensaje de referencia.

Para el lenguaje de programación C, también se define la constante MQRMH_STRUC_ID_ARRAY. Tiene el mismo valor que MQRMH_STRUC_ID, pero es una matriz de caracteres en lugar de una cadena.

Versión (MQLONG)

El número de versión de la estructura. El valor debe ser:

MQRMH_VERSION_1

Version-1 Estructura del encabezado del mensaje de referencia.

La siguiente constante especifica el número de versión de la versión actual:

MQRMH_VERSIÓN_ACTUAL

Versión actual de la estructura del encabezado del mensaje de referencia.

El valor inicial de este campo es MQRMH_VERSION_1.

StrucLength (MQLONG)

La longitud total de MQRMH, incluidas las cadenas al final de los campos fijos, pero sin los datos masivos.

El valor inicial de este campo es cero.

Encoding (MQLONG)

Esto especifica la codificación numérica de los datos masivos; no se aplica a los datos numéricos de la propia estructura MQRMH.

En la llamada MQPUT o MQPUT1, la aplicación debe establecer este campo con el valor adecuado para los datos.

El valor inicial de este campo es MQENC_NATIVE.

CodedCharSetId (MQLONG)

Especifica el identificador del conjunto de caracteres de los datos masivos; no se aplica a los datos de caracteres de la propia estructura MQRMH.

En la llamada MQPUT o MQPUT1, la aplicación debe establecer este campo con el valor adecuado para los datos. Se puede utilizar el siguiente valor especial:

MQCCSI_HEREDAR

Los datos de caracteres en los datos que siguen esta estructura están en el mismo conjunto de caracteres que esta estructura.

El gestor de colas cambia este valor en la estructura enviada en el mensaje al identificador real del juego de caracteres de la estructura. Si no se produce ningún error, la llamada MQGET no devuelve el valor MQCCSI_INHERIT.

No utilice MQCCSI_INHERIT si el valor del PutApp1Type campo en MQMD es MQAT_BROKER.

Este valor es compatible con los siguientes entornos:

-  AIX
-  IBM i
-  Linux
-  Windows

y para IBM MQ los clientes conectados a estos sistemas.

El valor inicial de este campo es MQCCSI_UNDEFINED.

Format (MQCHAR8)

Esto especifica el nombre del formato de los datos masivos.

En la llamada MQPUT o MQPUT1, la aplicación debe establecer este campo con el valor adecuado para los datos. Las reglas para codificar este campo son las mismas que para el *Format* campo en MQMD.

El valor inicial de este campo es MQFMT_NONE.

Flags (MQLONG)

Estos son indicadores de mensajes de referencia. Se definen los siguientes indicadores:

MQRMHF_ÚLTIMO

Esta bandera indica que el mensaje de referencia representa o contiene la última parte del objeto al que se hace referencia.

MQRMHF_NO_ÚLTIMO

El mensaje de referencia no contiene ni representa la última parte del objeto. MQRMHF_NOT_LAST ayuda a documentar el programa. No está previsto que esta opción se utilice con ninguna otra, pero como su valor es cero, dicho uso no puede detectarse.

El valor inicial de este campo es MQRMHF_NOT_LAST.

ObjectType (MQCHAR8)

Este es un nombre que la salida de mensajes puede utilizar para reconocer los tipos de mensajes de referencia que admite. El nombre debe cumplir las mismas reglas que el *Format* campo, véase [Formato](#).

El valor inicial de este campo es 8 espacios en blanco.

ObjectInstanceId (MQBYTE24)

Utilice este campo para identificar una instancia específica de un objeto. Si no es necesario, configúrelo con el siguiente valor:

MQOII_NINGUNO

No se ha especificado ningún identificador de instancia de objeto. El valor es un cero binario para la longitud del campo.

Para el lenguaje de programación C, también se define la constante MQOII_NONE_ARRAY, que tiene el mismo valor que MQOII_NONE, pero es una matriz de caracteres en lugar de una cadena.

La longitud de este campo viene determinada por MQ_OBJECT_INSTANCE_ID_LENGTH. El valor inicial de este campo es MQOII_NONE.

SrcEnvLength (MQLONG)

La longitud de los datos del entorno de origen. Si este campo es cero, no hay datos del entorno de origen y *SrcEnvOffset* se ignora.

El valor inicial de este campo es 0.

SrcEnvOffset (MQLONG)

Este campo especifica el desplazamiento de los datos del entorno de origen desde el inicio de la estructura MQRMH. Los datos del entorno de origen pueden ser especificados por el creador del mensaje de referencia, si dichos datos son conocidos por el creador. Por ejemplo, en Windows el entorno de origen, los datos podrían ser la ruta del directorio del objeto que contiene los datos masivos. Sin embargo, si el creador no conoce los datos del entorno de origen, la salida de mensaje proporcionada por el usuario debe determinar cualquier información del entorno que sea necesaria.

La longitud de los datos del entorno de origen viene dada por *SrcEnvLength* ; si esta longitud es cero, no hay datos del entorno de origen y *SrcEnvOffset* se ignora. Si están presentes, los datos del entorno de origen deben residir completamente dentro de *StrucLength* bytes desde el inicio de la estructura.

Las aplicaciones no deben dar por sentado que los datos del entorno comienzan inmediatamente después del último campo fijo de la estructura ni que son contiguos a ninguno de los datos a los que hacen referencia los *DestNameOffset* campos *SrcNameOffset*, *DestEnvOffset* y.

El valor inicial de este campo es 0.

SrcNameLength (MQLONG)

La longitud del nombre del objeto fuente. Si este campo es cero, no hay nombre de objeto de origen y *SrcNameOffset* se ignora.

El valor inicial de este campo es 0.

SrcNameOffset (MQLONG)

Este campo especifica el desplazamiento del nombre del objeto de origen desde el inicio de la estructura MQRMH. El nombre del objeto de origen puede ser especificado por el creador del mensaje de referencia, si ese dato es conocido por el creador. Sin embargo, si el creador no conoce el nombre del objeto de origen, la salida de mensaje proporcionada por el usuario debe identificar el objeto al que se va a acceder.

La longitud del nombre del objeto fuente viene dada por *SrcNameLength* ; si esta longitud es cero, no hay nombre de objeto fuente y *SrcNameOffset* se ignora. Si está presente, el nombre del objeto fuente debe residir completamente dentro de *StrucLength* bytes desde el inicio de la estructura.

Las aplicaciones no deben asumir que el nombre del objeto de origen es contiguo a ninguno de los datos a los que hacen referencia los *SrcEnvOffsetDestNameOffset* campos, *DestEnvOffset* y.

El valor inicial de este campo es 0.

DestEnvLength (MQLONG)

Esta es la longitud de los datos del entorno de destino. Si este campo es cero, no hay datos del entorno de destino y *DestEnvOffset* se ignora.

DestEnvOffset (MQLONG)

Este campo especifica el desplazamiento de los datos del entorno de destino desde el inicio de la estructura MQRMH. El creador del mensaje de referencia puede especificar los datos del entorno de destino, si los conoce. Por ejemplo, en Windows el entorno de destino, los datos podrían ser la ruta del directorio del objeto donde se almacenarán los datos masivos. Sin embargo, si el creador no conoce los datos del entorno de destino, es responsabilidad de la salida de mensajes proporcionada por el usuario determinar cualquier información del entorno que sea necesaria.

La longitud de los datos del entorno de destino viene dada por *DestEnvLength* ; si esta longitud es cero, no hay datos del entorno de destino y *DestEnvOffset* se ignora. Si están presentes, los datos del entorno de destino deben residir completamente dentro de *StrucLength* bytes desde el inicio de la estructura.

Las aplicaciones no deben asumir que los datos del entorno de destino son contiguos con ninguno de los datos a los que se refieren los *SrcEnvOffsetDestNameOffset* campos, *SrcNameOffset* y.

El valor inicial de este campo es 0.

DestNameLength (MQLONG)

La longitud del nombre del objeto de destino. Si este campo es cero, no hay nombre de objeto de destino y *DestNameOffset* se ignora.

DestNameOffset (MQLONG)

Este campo especifica el desplazamiento del nombre del objeto de destino desde el inicio de la estructura MQRMH. El nombre del objeto de destino puede ser especificado por el creador del mensaje de referencia, si ese dato es conocido por el creador. Sin embargo, si el creador no conoce el nombre del objeto de destino, es responsabilidad de la salida de mensaje proporcionada por el usuario identificar el objeto que se va a crear o modificar.

La longitud del nombre del objeto de destino viene dada por *DestNameLength* ; si esta longitud es cero, no hay nombre de objeto de destino y *DestNameOffset* se ignora. Si está presente, el nombre del objeto de destino debe residir completamente dentro de *StrucLength* bytes desde el inicio de la estructura.

Las aplicaciones no deben asumir que el nombre del objeto de destino es contiguo a ninguno de los datos a los que se refieren los *SrcEnvOffsetDestEnvOffset* campos, *SrcNameOffset* y.

El valor inicial de este campo es 0.

DataLogicalLength (MQLONG)

El *DataLogicalLength* campo especifica la longitud de los datos masivos a los que hace referencia la estructura MQRMH.

Si los datos masivos están realmente presentes en el mensaje, los datos comienzan en un desplazamiento de *StrucLength* bytes desde el inicio de la estructura MQRMH. La longitud de todo el mensaje menos *StrucLength* da la longitud de los datos masivos presentes.

Si hay datos en el mensaje, *DataLogicalLength* especifica la cantidad de esos datos que es relevante. Lo normal es que *DataLogicalLength* tenga el mismo valor que la longitud de los datos presentes en el mensaje.

Si la estructura MQRMH representa los datos restantes del objeto (a partir del desplazamiento lógico especificado), puede utilizar el valor cero para *DataLogicalLength*, siempre que los datos masivos no estén realmente presentes en el mensaje.

Si no hay datos, el final de MQRMH coincide con el final del mensaje.

El valor inicial de este campo es 0.

DataLogicalOffset (MQLONG)

Este campo especifica el desplazamiento bajo de los datos masivos desde el inicio del objeto del que forman parte dichos datos masivos. El desplazamiento de los datos masivos desde el inicio del objeto se denomina *desplazamiento* lógico. No se trata del desplazamiento físico de los datos masivos desde el inicio de la estructura MQRMH; ese desplazamiento viene dado por *StrucLength*.

Para permitir el envío de objetos grandes mediante mensajes de referencia, el desplazamiento lógico se divide en dos campos, y el desplazamiento lógico real viene dado por la suma de estos dos campos:

- *DataLogicalOffset* representa el resto obtenido al dividir el desplazamiento lógico entre 1 000 000 000. Por lo tanto, es un valor comprendido entre 0 y 999 999 999.
- *DataLogicalOffset2* representa el resultado obtenido cuando el desplazamiento lógico se divide por 1 000 000 000. Por lo tanto, es el número de múltiplos completos de 1 000 000 000 que existen en el desplazamiento lógico. El número de múltiplos está comprendido entre 0 y 999 999 999.

El valor inicial de este campo es 0.

DataLogicalOffset2 (MQLONG)

Este campo especifica el desplazamiento alto de los datos masivos desde el inicio del objeto del que forman parte los datos masivos. Es un valor comprendido entre 0 y 999 999 999. Consulte *DataLogicalOffset* para obtener detalles.

El valor inicial de este campo es 0.

MQRR - Registro de respuesta

Utilice la estructura MQRR para recibir el código de finalización y el código de motivo resultantes de la operación de apertura o inserción para una sola cola de destino, cuando el destino sea una lista de distribución. MQRR es una estructura de salida para las llamadas MQOPEN, MQPUT y MQPUT1.

Disponibilidad

La estructura MQRR está disponible en las siguientes plataformas:

-  AIX
-  IBM i
-  Linux
-  Windows

y para IBM MQ los clientes conectados a estos sistemas.

Conjunto de caracteres y codificación

Los datos en MQRR deben estar en el juego de caracteres proporcionado por el atributo del **CodedCharSetId** gestor de colas y la codificación del gestor de colas local proporcionada por MQENC_NATIVE. Sin embargo, si la aplicación se ejecuta como un cliente MQ MQI, la estructura debe estar en el conjunto de caracteres y la codificación del cliente.

Utilización

Al proporcionar una matriz de estas estructuras en las llamadas MQOPEN y MQPUT, o en la llamada MQPUT1, puede determinar los códigos de finalización y los códigos de motivo para todas las colas de una lista de distribución cuando el resultado de la llamada es mixto, es decir, cuando la llamada tiene éxito para algunas colas de la lista, pero falla para otras. El código de motivo MQRC_MULTIPLE_REASONS de la llamada indica que los registros de respuesta (si los proporciona la aplicación) han sido establecidos por el gestor de colas.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 551. Campos en MQRR		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
CompCode (código de finalización para la cola)	MQCC_OK	0
Motivo (código de motivo para la cola)	MQRC_NONE	0
Notas: 1. En el lenguaje de programación C, la variable macroMQRR_DEFAULT contiene los valores que se enumeran en la tabla. Úselo de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQRR MyRR = {MQRR_DEFAULT};</pre>		

Declaraciones lingüísticas

Declaración C para MQRR

```
typedef struct tagMQRR MQRR;  
struct tagMQRR {  
    MQLONG CompCode; /* Completion code for queue */  
    MQLONG Reason; /* Reason code for queue */  
};
```

COBOL declaración para MQRR

```
** MQRR structure  
10 MQRR.  
** Completion code for queue  
15 MQRR-COMPCODE PIC S9(9) BINARY.  
** Reason code for queue  
15 MQRR-REASON PIC S9(9) BINARY.
```

PL/I declaración para MQRR

```
dcl  
1 MQRR based,  
3 CompCode fixed bin(31), /* Completion code for queue */  
3 Reason fixed bin(31); /* Reason code for queue */
```

Declaración de Visual Basic para MQRR

```
Type MQRR  
CompCode As Long 'Completion code for queue'  
Reason As Long 'Reason code for queue'  
End Type
```

Detalles de campo para MQRR

Utilice la estructura de registro de respuesta MQRR para recibir el código de finalización y el código de motivo resultantes de la operación de apertura o inserción para una sola cola de destino, cuando el destino sea una lista de distribución. MQRR es una estructura de salida para las llamadas MQOPEN, MQPUT y MQPUT1.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 552. Campos en MQRR

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
CompCode (código de finalización para la cola)	MQCC_OK	0
Motivo (código de motivo para la cola)	MQRC_NONE	0
Notas: 1. En el lenguaje de programación C, la variable macroMQRR_DEFAULT contiene los valores que se enumeran en la tabla. Úselo de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQRR MyRR = {MQRR_DEFAULT};</pre>		

CompCode (MQLONG)

Este es el código de finalización resultante de la operación de apertura o inserción para la cola con el nombre especificado por el elemento correspondiente en la matriz de estructuras MQOR proporcionada en la llamada MQOPEN o MQPUT1.

Este campo siempre es de salida. El valor inicial de este campo es MQCC_OK.

Razón (MQLONG)

Este es el código de motivo resultante de la operación de apertura o inserción para la cola con el nombre especificado por el elemento correspondiente en la matriz de estructuras MQOR proporcionada en la llamada MQOPEN o MQPUT1.

Este campo siempre es de salida. El valor inicial de este campo es MQRC_NONE.

MQSCO - Opciones de configuración de « SSL » (MQSCO) / « TLS » (MQSCO)

La estructura MQSCO, junto con los campos « TLS » de la estructura MQCD, permite a una aplicación que se ejecuta como un IBM MQ MQI client especificar opciones de configuración que controlan el uso de TLS para la conexión del cliente cuando el protocolo del canal es TCP/IP. La estructura es un parámetro de entrada en la llamada MQCONN.

Disponibilidad

La estructura MQSCO está disponible en los siguientes sistemas operativos:

-  AIX
-  IBM i
-  Linux
-  Windows

Si el protocolo del canal del cliente no es « TCP/IP », se ignora la estructura MQSCO.

Versión

Los archivos de encabezado, COPY e INCLUDE que se proporcionan para los lenguajes de programación compatibles contienen la versión más reciente de MQSCO, pero con el valor inicial del Version campo establecido en MQSCO_VERSION_1. Para utilizar campos que no están presentes en la estructura version-1, la aplicación debe establecer el Version campo en el número de versión requerido.

Conjunto de caracteres y codificación

Los datos en MQSCO deben estar en el juego de caracteres proporcionado por el atributo del **CodedCharSetId** gestor de colas y la codificación del gestor de colas local proporcionada por MQENC_NATIVE.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 553. Entrada de campos MQSCO		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
StrucId (identificador de estructura)	MQSCO_STRUC_ID	'SCO~'
Versión (número de versión de la estructura)	MQSCO_CURRENT_VERSION	1
KeyRepository (ubicación del repositorio de claves)	Ninguna	Cadena nula o espacios en blanco
CryptoHardware (detalles del hardware criptográfico)	Ninguna	Cadena nula o espacios en blanco
AuthInfoRecCount (número de registros MQAIR presentes)	Ninguna	0
AuthInfoRecOffset (desplazamiento del primer registro MQAIR desde el inicio de MQSCO)	Ninguna	0
AuthInfoRecPtr (dirección del primer registro MQAIR)	Ninguna	Puntero nulo o bytes nulos
Nota: Los dos campos siguientes se ignoran si <i>Versión</i> es menor que MQSCO_VERSION_2.		
KeyResetCount (Recuento de restablecimientos de la clave secreta de TLS)	MQSCO_RESET_COUNT_DEFAULT	0
“FipsRequired (MQLONG)” en la página 554 (utilizar algoritmos FIPS criptográficos certificados en IBM MQ)	MQSSL_FIPS_NO	0
Nota: Los dos campos siguientes se ignoran si <i>Versión</i> es menor que MQSCO_VERSION_3.		
EncryptionPolicySuiteB (utilizar únicamente algoritmos criptográficos Suite B)	MQ_SUITE_B_NONE, MQ_SUITE_B_NOT_AVAILABLE, MQ_SUITE_B_NOT_AVAILABLE, MQ_SUITE_B_NOT_AVAILABLE	1, 0, 0, 0
Nota: Los dos campos siguientes se ignoran si <i>Versión</i> es menor que MQSCO_VERSION_4.		
CertificateValPolicy (política de validación de certificados)	MQ_CERT_VAL_POLICY_DEFAULT	0
Nota: Los dos campos siguientes se ignoran si <i>Versión</i> es menor que MQSCO_VERSION_5.		

Tabla 553. Entrada de campos MQSCO (continuación)

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
CertificateLabel (detalla la etiqueta del certificado que se está utilizando)	Ninguna	Cadena nula o espacios en blanco
Nota: Los siguientes tres campos se ignoran si <i>Version</i> es menor que MQSCO_VERSION_6.		
KeyRepoPasswordPtr (dirección de TLS contraseña del repositorio de claves)	Ninguna	Puntero nulo o bytes nulos
KeyRepoPasswordOffset (desplazamiento de la contraseña del repositorio de claves TLS)	Ninguna	0
KeyRepoPasswordLength (longitud de la contraseña del repositorio de claves TLS)	Ninguna	0
Nota: Los siguientes cinco campos se ignoran si <i>Version</i> es menor que MQSCO_VERSION_7.		
HTTPSKeyStorePtr (dirección de una cadena HTTPSKeyStore)	ninguno	Puntero nulo
HTTPSKeyStoreOffset (desplazamiento de una cadena de e HTTPSKeyStore) desde el inicio de MQSCO	ninguno	0
HTTPSKeyStoreLength (longitud de una cadena de HTTPSKeyStore)	ninguno	0
HTTPSCertValidation (nivel de validación del certificado requerido). MQ_HTTPSCERTVAL_DEFAULT significa que se utiliza la configuración del archivo mqclient.ini , si existe)	MQ_HTTPSCERTVAL_DEFAULT MQ_HTTPSCERTVAL_ANY MQ_HTTPSCERTVAL_NONE MQ_HTTPSCERTVAL_HOSTNAMECN	0 1 2 3
HTTPSCertRevocation (nivel de comprobación de revocación de certificados requerido). MQ_HTTPSCERTREV_DEFAULT significa que se utiliza la configuración del archivo mqclient.ini , si existe)	MQ_HTTPSCERTREV_DEFAULT MQ_HTTPSCERTREV_REQUIRED MQ_HTTPSCERTREV_DISABLED MQ_HTTPSCERTREV_OPTIONAL	0 1 2 3

Notas:

1. El símbolo ~ representa un único carácter en blanco.
2. En el lenguaje de programación C, la variable macroMQSCO_DEFAULT contiene los valores que figuran en la tabla. Úselo de la siguiente manera para proporcionar valores iniciales para los campos de la estructura:

```
MQSCO MySCO = {MQSCO_DEFAULT};
```

Declaraciones lingüísticas

Declaración C para MQSCO

```
typedef struct tagMQSCO MQSCO;
struct tagMQSCO {
    MQCHAR4      StrucId;           /* Structure identifier */
    MQLONG       Version;          /* Structure version number */
    MQCHAR256    KeyRepository;    /* Location of TLS key */
    MQCHAR256    CryptoHardware;   /* repository */
    MQCHAR256    CryptoHardware;   /* Cryptographic hardware */
    MQLONG       AuthInfoRecCount; /* configuration string */
    MQLONG       AuthInfoRecOffset; /* Number of MQAIR records */
    PMQAIR       AuthInfoRecPtr;   /* present */
    MQLONG       AuthInfoRecOffset; /* Offset of first MQAIR */
    MQLONG       AuthInfoRecOffset; /* record from start of */
    MQLONG       AuthInfoRecOffset; /* MQSCO structure */
    PMQAIR       AuthInfoRecPtr;   /* Address of first MQAIR */
    MQLONG       KeyResetCount;    /* record */
    MQLONG       KeyResetCount;    /* Number of unencrypted */
    MQLONG       KeyResetCount;    /* bytes sent/received */
    MQLONG       KeyResetCount;    /* before secret key is */
    MQLONG       KeyResetCount;    /* reset */
    MQLONG       FipsRequired;     /* Using FIPS certified */
    MQLONG       EncryptionPolicySuiteB[4]; /* algorithms */
    MQLONG       EncryptionPolicySuiteB[4]; /* Use only Suite B */
    MQLONG       CertificateValPolicy; /* cryptographic algorithms */
    MQLONG       CertificateValPolicy; /* Certificate validation */
    MQLONG       CertificateValPolicy; /* policy */
    MQCHAR64     CertificateLabel; /* Certificate label */
    MQPTR        KeyRepoPasswordPtr; /* Address of key */
    MQLONG       KeyRepoPasswordOffset; /* repository password */
    MQLONG       KeyRepoPasswordOffset; /* Offset of key repository */
    MQLONG       KeyRepoPasswordLength; /* password */
    MQLONG       KeyRepoPasswordLength; /* Length of key repository */
    MQLONG       KeyRepoPasswordLength; /* password */
    MQLONG       HTTPSCertValidation; /* HTTPS certificate */
    MQLONG       HTTPSCertRevocation; /* validation level */
    MQLONG       HTTPSCertRevocation; /* HTTPS certificate */
    MQLONG       HTTPSCertRevocation; /* revocation level */
    MQPTR        HTTPSKeyStorePtr; /* Address of HTTPS */
    MQLONG       HTTPSKeyStoreOffset; /* Keystore */
    MQLONG       HTTPSKeyStoreOffset; /* Offset of HTTPS Keystore */
    MQLONG       HTTPSKeyStoreLength; /* Length of HTTPS keystore */
};
```

COBOL declaración para MQSCO

```
** MQSCO structure
10 MQSCO.
** Structure identifier
15 MQSCO-STRUCID PIC X(4).
** Structure version number
15 MQSCO-VERSION PIC S9(9) BINARY.
** Location of TLS key repository
15 MQSCO-KEYREPOSITORY PIC X(256).
** Cryptographic hardware configuration string
15 MQSCO-CRYPTOHARDWARE PIC X(256).
** Number of MQAIR records present
15 MQSCO-AUTHINFORECCOUNT PIC S9(9) BINARY.
** Offset of first MQAIR record from start of MQSCO structure
15 MQSCO-AUTHINFORECOFFSET PIC S9(9) BINARY.
** Address of first MQAIR record
15 MQSCO-AUTHINFORECPTTR POINTER.
** Version 1 **
** Number of unencrypted bytes sent/received before secret key is
** reset
15 MQSCO-KEYRESETCOUNT PIC S9(9) BINARY.
** Using FIPS certified algorithms
15 MQSCO-FIPSREQUIRED PIC S9(9) BINARY.
** Version 2 **
** Use only Suite B cryptographic algorithms
```

```

15 MQSCO-ENCRYPTIONPOLICYSUITEB PIC S9(9) BINARY OCCURS 4.
** Version 3 **
** Certificate validation policy setting
15 MQSCO-CERTIFICATEVALPOLICY PIC S9(9) BINARY.
** Version 4 **
** SSL/TLS certificate label
15 MQSCO-CERTIFICATELABEL PIC X(64).
** Version 5 **
** Add padding to ensure that pointers start on correct
** boundaries
15 FILLER PIC S9(9) BINARY VALUE 0.
** Address of key repository password
15 MQSCO-KEYREPOPASSWORDPTR POINTER.
** Offset of key repository password
15 MQSCO-KEYREPOPASSWORDOFFSET PIC S9(9) BINARY.
** Length of key repository password
15 MQSCO-KEYREPOPASSWORDLENGTH PIC S9(9) BINARY.
** Version 6 ** V 10.0.0 ** HTTPS certificate validation level
15 MQSCO-HTTPSCERTVALIDATION PIC S9(9) BINARY VALUE 0.
** HTTPS certificate revocation level
15 MQSCO-HTTPSCERTREVOCATION PIC S9(9) BINARY VALUE 0.
** Address of HTTPS Keystore
15 MQSCO-HTTPSKEYSTOREPTR POINTER VALUE NULL.
** Offset of HTTPS Keystore
15 MQSCO-HTTPSKEYSTOREOFFSET PIC S9(9) BINARY VALUE 0.
** Length of HTTPS keystore
15 MQSCO-HTTPSKEYSTORELENGTH PIC S9(9) BINARY VALUE 0.
** Ver:7 **

```

PL/I declaración para MQSCO

```

dcl
1 MQSCO based,
3 StrucId char(4), /* Structure identifier */
3 Version fixed bin(31), /* Structure version number */
3 KeyRepository char(256), /* Location of TLS key
repository */
3 CryptoHardware char(256), /* Cryptographic hardware
configuration string */
3 AuthInfoRecCount fixed bin(31), /* Number of MQAIR records
present */
3 AuthInfoRecOffset fixed bin(31), /* Offset of first MQAIR record
from start of MQSCO structure */
3 AuthInfoRecPtr pointer, /* Address of first MQAIR record */
3 KeyResetCount fixed bin(31), /* Key reset count */
/* Version 1 */
3 FipsRequired fixed bin(31), /* FIPS required */
/* Version 2 */
3 EncryptionPolicySuiteB (4) fixed bin(31), /* Suite B encryption policy */
/* Version 3 */
3 CertificateValPolicy fixed bin(31), /* Certificate validation policy */
/* Version 4 */
3 CertificateLabel char(64), /* SSL/TLS certificate label */
/* Version 5 */
3 KeyRepoPasswordPtr pointer, /* Address of key repository
password */
3 KeyRepoPasswordOffset fixed bin(31), /* Offset of key repository
password */
3 KeyRepoPasswordLength fixed bin(31); /* Length of key repository
password */
/* Version 6 */ V 10.0.0
3 HTTPSCertValidation fixed bin(31), /* HTTPS certificate */
validation level */
3 HTTPSCertRevocation fixed bin(31), /* HTTPS certificate */
revocation level */
3 HTTPSKeyStorePtr pointer, /* Address of HTTPS Keystore */
3 HTTPSKeyStoreOffset fixed bin(31), /* Offset of HTTPS Keystore */
3 HTTPSKeyStoreLength fixed bin(31); /* Length of HTTPS keystore */
/* Version 7 */

```

Declaración de Visual Basic para MQSCO

Type MQSCO		
StrucId	As String*4	'Structure identifier'
Version	As Long	'Structure version number'
KeyRepository	As String*256	'Location of TLS key repository'
CryptoHardware	As String*256	'Cryptographic hardware configuration'
		'string'
AuthInfoRecCount	As Long	'Number of MQAIR records present'

AuthInfoRecOffset	As Long	'Offset of first MQAIR record from'
AuthInfoRecPtr	As MQPTR	'start of MQSCO structure'
KeyResetCount	As Long	'Address of first MQAIR record'
is reset'		'Number of unencrypted bytes sent/received before secret key
'Version 1'		
FipsRequired	As Long	'Mandatory FIPS CipherSpecs?'
'Version 2'		
End Type		

Referencia relacionada

“MQCNO - Opciones de conexión” en la página 269

La estructura MQCNO permite a la aplicación especificar opciones relacionadas con la conexión al gestor de colas. La estructura es un parámetro de entrada/salida en la llamada MQCONN.

Detalles de campo para MQSCO

La estructura de opciones de configuración de MQSCO (SSL / TLS), junto con los campos TLS de la estructura MQCD, permite a una aplicación que se ejecuta como un IBM MQ MQI client especificar opciones de configuración que controlan el uso de TLS para la conexión del cliente cuando el protocolo del canal es TCP/IP. La estructura es un parámetro de entrada en la llamada MQCONN.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 554. Entrada de campos MQSCO		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>StrucId</u> (identificador de estructura)	MQSCO_STRUC_ID	'SCO-'
<u>Versión</u> (número de versión de la estructura)	MQSCO_CURRENT_VERSION	1
<u>KeyRepository</u> (ubicación del repositorio de claves)	Ninguna	Cadena nula o espacios en blanco
<u>CryptoHardware</u> (detalles del hardware criptográfico)	Ninguna	Cadena nula o espacios en blanco
<u>AuthInfoRecCount</u> (número de registros MQAIR presentes)	Ninguna	0
<u>AuthInfoRecOffset</u> (desplazamiento del primer registro MQAIR desde el inicio de MQSCO)	Ninguna	0
<u>AuthInfoRecPtr</u> (dirección del primer registro MQAIR)	Ninguna	Puntero nulo o bytes nulos
Nota: Los dos campos siguientes se ignoran si <i>Version</i> es menor que MQSCO_VERSION_2.		
<u>KeyResetCount</u> (Recuento de restablecimientos de la clave secreta de TLS)	MQSCO_RESET_COUNT_DEFAULT	0
“FipsRequired (MQLONG)” en la página 554 (utilizar algoritmos FIPS criptográficos certificados en IBM MQ)	MQSSL_FIPS_NO	0
Nota: Los dos campos siguientes se ignoran si <i>Version</i> es menor que MQSCO_VERSION_3.		

Tabla 554. Entrada de campos MQSCO (continuación)

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>EncryptionPolicySuiteB</u> (utilizar únicamente algoritmos criptográficos Suite B)	MQ_SUITE_B_NONE, MQ_SUITE_B_NOT_AVAILABLE, MQ_SUITE_B_NOT_AVAILABLE, MQ_SUITE_B_NOT_AVAILABLE	1, 0, 0, 0
Nota: Los dos campos siguientes se ignoran si <i>Version</i> es menor que MQSCO_VERSION_4.		
<u>CertificateValPolicy</u> (política de validación de certificados)	MQ_CERT_VAL_POLICY_DEFAULT	0
Nota: Los dos campos siguientes se ignoran si <i>Version</i> es menor que MQSCO_VERSION_5.		
<u>CertificateLabel</u> (detalla la etiqueta del certificado que se está utilizando)	Ninguna	Cadena nula o espacios en blanco
Nota: Los siguientes tres campos se ignoran si <i>Version</i> es menor que MQSCO_VERSION_6.		
<u>KeyRepoPasswordPtr</u> (dirección de TLS contraseña del repositorio de claves)	Ninguna	Puntero nulo o bytes nulos
<u>KeyRepoPasswordOffset</u> (desplazamiento de la contraseña del repositorio de claves TLS)	Ninguna	0
<u>KeyRepoPasswordLength</u> (longitud de la contraseña del repositorio de claves TLS)	Ninguna	0
Nota: Los siguientes cinco campos se ignoran si <i>Version</i> es menor que MQSCO_VERSION_7.		
v 10.0.0 <u>HTTPSKeyStorePtr</u> (dirección de una cadena HTTPSKeyStore)	ninguno	Puntero nulo
v 10.0.0 <u>HTTPSKeyStoreOffset</u> (desplazamiento de una cadena de e HTTPSKeyStore) desde el inicio de MQSCO	ninguno	0
v 10.0.0 <u>HTTPSKeyStoreLength</u> (longitud de una cadena de HTTPSKeyStore)	ninguno	0
v 10.0.0 <u>HTTPSCertValidation</u> (nivel de validación del certificado requerido). MQ_HTTPSCERTVAL_DEFAULT significa que se utiliza la configuración del archivo mqclient.ini , si existe)	MQ_HTTPSCERTVAL_DEFAULT MQ_HTTPSCERTVAL_ANY MQ_HTTPSCERTVAL_NONE MQ_HTTPSCERTVAL_HOSTNAMECN	0 1 2 3

Tabla 554. Entrada de campos MQSCO (continuación)

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
V 1000 HTTPSCertRevocation (nivel de comprobación de revocación de certificados requerido. MQ_HTTPSCERTREV_DEFAULT significa que se utiliza la configuración del archivo mqclient.ini, si existe)	MQ_HTTPSCERTREV_DEFAULT MQ_HTTPSCERTREV_REQUIRED MQ_HTTPSCERTREV_DISABLED MQ_HTTPSCERTREV_OPTIONAL	0 1 2 3

Notas:

1. El símbolo ↵ representa un único carácter en blanco.
2. En el lenguaje de programación C, la variable macroMQSCO_DEFAULT contiene los valores que figuran en la tabla. Úselo de la siguiente manera para proporcionar valores iniciales para los campos de la estructura:

```
MQSCO MySCO = {MQSCO_DEFAULT};
```

StrucId (MQCHAR4)

Este es el identificador de estructura de la estructura de opciones de configuración SSL / TLS. Siempre es un campo de entrada. Su valor es MQSCO_STRUC_ID.

El valor debe ser:

MQSCO_STRUC_ID

Identificador para la estructura de opciones de configuración SSL / TLS.

Para el lenguaje de programación C, también se define la constante MQSCO_STRUC_ID_ARRAY. Tiene el mismo valor que MQSCO_STRUC_ID, pero es una matriz de caracteres en lugar de una cadena.

Versión (MQLONG)

Este es el número de versión de la estructura; el valor debe ser una de las siguientes cadenas:

MQSCO_VERSION_1

Version-1 TLS Estructura de opciones de configuración.

MQSCO_VERSION_2

Version-2 TLS Estructura de opciones de configuración.

MQSCO_VERSION_3

Version-3 TLS Estructura de opciones de configuración.

MQSCO_VERSION_4

Version-4 TLS Estructura de opciones de configuración.

MQSCO_VERSION_5

Version-5 TLS Estructura de opciones de configuración.

MQSCO_VERSION_6

Version-6 TLS Estructura de opciones de configuración.

V 1000 MQSCO_VERSION_7

Version-7 TLS Estructura de opciones de configuración.

Los campos que solo existen en las versiones más recientes de la estructura se identifican como tales en las descripciones de los campos. La siguiente constante especifica el número de versión de la versión actual:

MQSCO_VERSION_ACTUAL

Versión actual de la estructura de opciones de configuración de TLS.

Este campo siempre es un campo de entrada. El valor inicial de este campo es MQSCO_VERSION_1.

KeyRepository (MQCHAR256)

Este campo solo es relevante para IBM MQ MQI clients ejecutar en sistemas IBM iAIX, Linux, and Windows . Especifica la ubicación del archivo de base de datos de claves en el que se almacenan las claves y los certificados. Si no se especifica la extensión del .kdb archivo, se añade automáticamente la extensión.

Cada archivo de base de datos clave puede tener un *archivo de almacenamiento de contraseñas* asociado. El archivo stash contiene contraseñas codificadas que se utilizan para permitir el acceso programático a la base de datos de claves. El archivo de almacenamiento de contraseñas debe residir en el mismo directorio y tener el mismo nombre base que la base de datos de claves, y debe terminar con la extensión .sth.

Por ejemplo, si el archivo de base de datos de claves es /xxx/yyy/key.kdb, entonces el archivo de almacenamiento de contraseñas debe ser /xxx/yyy/key.sth, donde xxx y yyy representan nombres de directorios.

La contraseña de la base de datos clave también se puede especificar utilizando el *KeyRepoPasswordPtr* campo *KeyRepoPasswordOffset* o de la estructura MQSCO.

Si el valor es más corto que la longitud del campo, termine el valor con un carácter nulo o rellénelo con espacios en blanco hasta alcanzar la longitud del campo. El valor no se comprueba; si se produce un error al acceder al repositorio de claves, la llamada falla con el código de motivo MQRC_KEY_REPOSITORY_ERROR.

Para ejecutar una conexión TLS desde un IBM MQ MQI client, establezca *KeyRepository* en un nombre de archivo de base de datos de claves válido.

Este es un campo de entrada. La longitud de este campo viene dada por MQ_SSL_KEY_REPOSITORY_LENGTH. El valor inicial de este campo es la cadena nula en C y caracteres en blanco en otros lenguajes de programación.

CryptoHardware (MQCHAR256)

Este campo proporciona detalles de configuración para el hardware criptográfico conectado al sistema cliente.

Establezca el campo en una cadena con el siguiente formato, o déjelo en blanco o nulo:

```
GSK_PKCS11=the PKCS #11 driver path and file name;the PKCS #11  
token label;the PKCS #11 token password;symmetric cipher setting;
```

Para utilizar hardware criptográfico que cumpla con la interfaz PKCS #11, por ejemplo, el IBM 4960 o IBM el 4764, se deben especificar la ruta del controlador PKCS #11, la etiqueta del token PKCS #11 y las cadenas de contraseña del token PKCS #11, cada una de ellas terminada con un punto y coma.

La ruta del controlador PKCS #11 es una ruta absoluta a la biblioteca compartida que proporciona soporte para la tarjeta PKCS #11. El nombre del archivo del controlador PKCS #11 es el nombre de la biblioteca compartida. Un ejemplo del valor requerido para la ruta PKCS #11 y el nombre del archivo es:

```
/usr/lib/pkcs11/PKCS11_API.so
```

La etiqueta del token PKCS #11 debe coincidir con la etiqueta que ha configurado en su hardware.

Si no se requiere ninguna configuración de hardware criptográfico, deje el campo en blanco o nulo.

Si el valor es más corto que la longitud del campo, termine el valor con un carácter nulo o rellénelo con espacios en blanco hasta alcanzar la longitud del campo. Si el valor no es válido o provoca un

error al utilizarse para configurar el hardware criptográfico, la llamada falla con el código de motivo MQRC_CRYPT0_HARDWARE_ERROR.

Este es un campo de entrada. La longitud de este campo viene dada por MQ_SSL_CRYPT0_HARDWARE_LENGTH. El valor inicial de este campo es la cadena nula en C y caracteres en blanco en otros lenguajes de programación.

AuthInfoRecCount (MQLONG)

Este es el número de registros de información de autenticación (MQAIR) a los que se refieren los campos *AuthInfoRecPtrAuthInfoRecOffset* o. Para obtener más información, consulte [“MQAIR - Registro de información de autenticación”](#) en la página 214. El valor debe ser mayor o igual que cero. Si el valor no es válido, la llamada falla con el código de motivo MQRC_AUTH_INFO_REC_COUNT_ERROR.

Este es un campo de entrada. El valor inicial de este campo es 0.

AuthInfoRecOffset (MQLONG)

Este es el desplazamiento en bytes del primer registro de información de autenticación desde el inicio de la estructura MQSCO. El desplazamiento puede ser positivo o negativo. El campo se ignora si *AuthInfoRecCount* es cero.

Puede utilizar o *AuthInfoRecPtrAuthInfoRecOffset* para especificar los registros MQAIR, pero no ambos; consulte la descripción del *AuthInfoRecPtr* campo para obtener más detalles.

Este es un campo de entrada. El valor inicial de este campo es 0.

AuthInfoRecPtr (PMQAIR)

Esta es la dirección del primer registro de información de autenticación. El campo se ignora si *AuthInfoRecCount* es cero.

Puede proporcionar la matriz de registros MQAIR de dos maneras:

- Al utilizar el campo puntero *AuthInfoRecPtr*

En este caso, la aplicación puede declarar una matriz de registros MQAIR separada de la estructura MQSCO y establecer *AuthInfoRecPtr* la dirección de la matriz.

Considere utilizar *AuthInfoRecPtr* para lenguajes de programación que admitan el tipo de datos puntero de forma que sea portátil a diferentes entornos (por ejemplo, el lenguaje de programación C).

- Al utilizar el campo de compensación *AuthInfoRecOffset*

En este caso, la aplicación debe declarar una estructura compuesta que contenga un MQSCO seguido de la matriz de registros MQAIR, y establecer *AuthInfoRecOffset* en la compensación del primer registro de la matriz desde el inicio de la estructura MQSCO. Asegúrese de que este valor sea correcto y tenga un valor que pueda acomodarse dentro de un MQLONG (el lenguaje de programación más restrictivo es COBOL, para el cual el rango válido es de -999 999 999 a +999 999 999).

Considere utilizar *AuthInfoRecOffset* para lenguajes de programación que no admiten el tipo de datos puntero, o que implementan el tipo de datos puntero de una manera que no es portable a diferentes entornos (por ejemplo, el lenguaje de programación COBOL).

Independientemente de la técnica que elijas, solo se puede utilizar *AuthInfoRecOffset* una de *AuthInfoRecPtr* y; la llamada falla con el código de motivo MQRC_AUTH_INFO_REC_ERROR si ambas son distintas de cero.

Este es un campo de entrada. El valor inicial de este campo es el puntero nulo en aquellos lenguajes de programación que admiten punteros y, en los demás casos, una cadena de bytes totalmente nulos.

Nota: En plataformas donde el lenguaje de programación no admite el tipo de datos puntero, este campo se declara como una cadena de bytes de la longitud adecuada.

KeyResetCount (MQLONG)

Esto representa el número total de bytes sin cifrar enviados y recibidos dentro de una conversación TLS antes de que se renegocie la clave secreta.

El número de bytes incluye la información de control que envía el MCA.

Si especifica una cuenta de restablecimiento de clave secreta TLS entre 1 byte y 32 KB, los canales TLS utilizarán una cuenta de restablecimiento de clave secreta de 32 KB. Esto es para evitar el coste de procesamiento de restablecimientos excesivos de claves que se producirían con valores pequeños de restablecimiento de claves secretas TLS.

Este es un campo de entrada. El valor es un número comprendido entre 0 y 999 999 999, con un valor predeterminado de 0. Utilice el valor 0 para indicar que las claves secretas nunca se renegocian.

FipsRequired (MQLONG)

IBM MQ Se puede configurar con hardware criptográfico para que los módulos criptográficos utilizados sean los proporcionados por el producto de hardware; estos pueden FIPS certificarse a un nivel determinado en función del producto de hardware criptográfico que se utilice.

Utilice este campo para especificar que solo se utilicen FIPS algoritmos certificados si la criptografía se proporciona en el software IBM MQ suministrado.

Notas:

-  En AIX, Linux, y Windows, IBM MQ garantiza FIPS 140-3 el cumplimiento de la normativa a través del GSKit 9 módulo IBM Crypto for C (ICC) criptográfico. La certificación del NIST correspondiente al FIPS 140-3 módulo puede consultarse en <https://csrc.nist.gov/projects/cryptographic-module-validation-program/certificate/4755>.
 -  FIPS Actualmente no hay soporte disponible para Linux s390x. Los clientes de esta plataforma que necesiten FIPS asistencia técnica deben seguir utilizando una versión anterior de IBM MQ. La compatibilidad con FIPS 140-3 se habilitará en una futura versión de FixPack una vez que el IBM Crypto for C (ICC) módulo criptográfico haya obtenido su certificación en esta plataforma.
 -  El módulo FIPS 140-3 criptográfico integrado IBM Semeru Runtime fue aprobado por el NIST en agosto de 2024. IBM MQ 10.0.0 incorpora compatibilidad con la gestión de IBM MQ classes for JMS conexiones de IBM MQ classes for Java cliente mediante TLS para FIPS 140-3 en Java 8 y IBM Semeru Runtime 11+. La certificación del NIST correspondiente al FIPS 140-3 módulo se puede consultar en <https://csrc.nist.gov/projects/cryptographic-module-validation-program/certificate/4755>.
 - En IBM MQ el caso de Containers, las imágenes de IBM MQ Operator contenedor de 3.2.0 y del gestor de colas 9.4.0.0 en adelante se basan en UBI 9. FIPS 140-3 El cumplimiento de IBM MQ la normativa para productos envasados está actualmente pendiente.
-  Si FIPS está habilitado, los procesos IBM MQ de control de contenedores utilizan un módulo FIPS 140-3 certificado de OpenSSL. Para obtener más información sobre la certificación del NIST, consulte: <https://access.redhat.com/compliance/fips>. IBM MQ Los gestores de colas que se ejecutan en imágenes de contenedor tienen el mismo nivel FIPS de certificación que la versión de la plataforma de IBM MQ la imagen base.

Cuando IBM MQ se instala, también se instala una implementación de la criptografía TLS, que proporciona algunos módulos FIPS certificados.

Los valores pueden ser:

MQSSL_FIPS_NO

Éste es el valor predeterminado. Cuando se establece en este valor:

- Se puede utilizar cualquier CipherSpec compatible con una plataforma concreta.
- Si se ejecuta sin utilizar hardware criptográfico, el CipherSpecs se ejecuta utilizando criptografía FIPS 140-2 certificada en las IBM MQ plataformas.

Para obtener una lista de los dispositivos de autenticación de huellas dactilares (CipherSpecs,) certificados por FIPS, consulte la tabla descrita en [Activación de dispositivos de autenticación de huellas dactilares \(CipherSpecs \)](#).

MQSSL_FIPS_YES

Cuando se establece este valor, a menos que se utilice hardware criptográfico para realizar la criptografía, se puede estar seguro de que

- Solo se pueden utilizar FIPS algoritmos criptográficos certificados en el CipherSpec que se aplica a esta conexión de cliente.
- Las conexiones entrantes y salientes del canal de cifrado (TLS) solo se realizan correctamente si se utilizan determinadas especificaciones de cifrado.

Consulte [Activación de CipherSpecs](#) para obtener más información.

Nota: Siempre que sea posible, si se han configurado puntos de acceso (CipherSpecs) que solo admiten FIPS, el cliente de la red de la Universidad de Maryland (MQI) rechazará las conexiones que especifiquen un punto de acceso (CipherSpec) no compatible con FIPS, devolviendo el error MQRC_SSL_INITIALIZATION_ERROR. IBM MQ no garantiza que rechace todas las conexiones de este tipo y es su responsabilidad determinar si su IBM MQ configuración es FIPS conforme.

EncryptionPolicySuiteB (MQLONG)

Este campo especifica si se utiliza criptografía compatible con Suite B y qué nivel de seguridad se emplea. El valor puede ser uno o varios de los siguientes:

- MQ_SUITE_B_NONE

No se utiliza criptografía compatible con Suite B.

- MQ_SUITE_B_128_BIT

Se utiliza seguridad Suite B de 128 bits.

- MQ_SUITE_B_192_BIT

Se utiliza seguridad Suite B de 192 bits.

Nota: El uso de MQ_SUITE_B_NONE con cualquier otro valor en este campo no es válido.

CertificateValPolicy (MQLONG)

Este campo especifica qué tipo de política de validación de certificados se utiliza.

El campo se puede configurar con uno de los siguientes valores:

MQ_CERT_VAL_POLICY_ANY

Aplica cada una de las políticas de validación de certificados compatibles con la biblioteca de sockets seguros. Acepte la cadena de certificados si alguna de las políticas considera que la cadena de certificados es válida.

MQ_CERT_VAL_POLICY_RFC5280

Aplique únicamente la política de validación de certificados compatible con RFC5280. Este valor proporciona una validación más estricta que el valor ANY, pero rechaza algunos certificados digitales más antiguos.

MQ_CERT_VAL_POLICY_NONE

No aplicar ninguna política de validación de certificados. Esta configuración es solo para aplicaciones cliente y acepta el certificado de servidor TLS sin validar la cadena de confianza.

El valor inicial de este campo es MQ_CERT_VAL_POLICY_ANY

CertificateLabel (MQCHAR64)

Este campo proporciona detalles sobre la etiqueta del certificado que se está utilizando.

IBM MQ Inicializa el valor predeterminado del *CertificateLabel* campo como espacios en blanco.

Esto se interpreta en tiempo de ejecución como el valor predeterminado y es compatible con versiones anteriores.

Por ejemplo, si se especifica una versión de MQSCO inferior a 5.0, o se utiliza el valor predeterminado en blanco para el *CertificateLabel* campo, se utiliza el valor predeterminado preexistente. *ibmwebspheremquser_id*

KeyRepoPasswordPtr (MQPTR)

Esta es la dirección de la contraseña del repositorio de claves TLS.

Este es un campo de entrada. El valor inicial de este campo es el puntero nulo en aquellos lenguajes de programación que admiten punteros y, en los demás casos, una cadena de bytes totalmente nulos. Este campo se ignora si *Version* es menor que MQSCO_VERSION_6.

Este campo solo es relevante para IBM MQ MQI clients ejecutar en sistemas IBM iAIX, Linux, and Windows .

La contraseña del repositorio de claves se puede especificar como una cadena de texto sin formato o como una contraseña que se ha cifrado con la **runmqicred** utilidad.

Si proporciona una frase de contraseña cifrada, especifique la clave inicial que se utilizó para cifrar la frase de contraseña en la estructura MQCSP proporcionada por la misma aplicación cliente.

La contraseña del repositorio de claves especificada en este campo anula cualquier contraseña del repositorio de claves especificada mediante la variable de entorno *MQKEYRPWD* o la *SSLKeyRepositoryPassword* propiedad en la sección SSL del archivo de configuración del cliente.

Puede utilizar o *KeyRepoPasswordPtrKeyRepoPasswordOffset* para especificar la frase de contraseña del repositorio de claves, pero no ambos.

KeyRepoPasswordOffset (MQLONG)

Este es el desplazamiento en bytes de la frase de contraseña del repositorio de claves TLS desde el inicio de la estructura MQSCO. El desplazamiento puede ser positivo o negativo.

Puede utilizar o *KeyRepoPasswordPtrKeyRepoPasswordOffset* para especificar la frase de contraseña del repositorio de claves, pero no ambos. Para obtener más información, consulte la descripción del *KeyRepoPasswordPtr* campo.

Este es un campo de entrada. El valor inicial de este campo es 0. Este campo se ignora si *Version* es menor que MQSCO_VERSION_6.

KeyRepoPasswordLength (MQLONG)

Esta es la longitud de la contraseña del repositorio de claves TLS.

En IBM i, la longitud máxima de la contraseña del repositorio de claves es de 128 caracteres. Si la frase de contraseña del repositorio de claves supera la longitud máxima permitida, la conexión falla con el error MQRC_KEY_REPOSITORY_ERROR.

Este es un campo de entrada. El valor inicial de este campo es 0. Este campo se ignora si *Version* es menor que MQSCO_VERSION_6.

HTTPSKeyStorePtr

V 10.0.0

Esta es la dirección de una cadena que contiene la ruta al archivo HTTPSKeyStore. El repositorio de claves PKCS #12 (HTTPSKeyStore) es un repositorio de claves PKCS #12 que el gestor de colas utiliza cuando establece la confianza en un certificado presentado por un servidor de validación de certificados (HTTPS).

Este es un campo de entrada. El valor inicial de este campo es el puntero nulo en aquellos lenguajes de programación que admiten punteros y, en los demás casos, una cadena de bytes totalmente nulos. Este campo se ignora si *Version* es menor que MQSCO_VERSION_7.

AIX Este campo solo es relevante para IBM MQ los clientes de MQI que se ejecutan en los sistemas operativos, Linux, y Windows . La longitud máxima de la ruta proporcionada es la del sistema operativo del cliente.

Puede utilizar o *HTTPSKeyStorePtrHTTPSKeyStoreOffset* para especificar la frase de contraseña del repositorio de claves, pero no ambos. Si no se especifica ninguno de los dos campos, la función que requiere HTTP seguro se deshabilita en este cliente. Si se especifican ambos campos, la llamada falla con el código de motivo MQRC_HTTPS_KEYSTORE_ERROR.

HTTPSKeyStoreOffset

V 10.0.0

Este es el desplazamiento, en bytes, de la ruta HTTPSKeyStore desde el inicio de la estructura MQSCO. El desplazamiento puede ser positivo o negativo.

Puede utilizar o *HTTPSKeyStorePtrHTTPSKeyStoreOffset* para especificar la ruta HTTPSKeyStore, pero no ambos. Si no se especifica ninguno de los dos campos, la función que requiere HTTP seguro se deshabilita en este cliente. Si se especifican ambos campos, la llamada falla con el código de motivo MQRC_HTTPS_KEYSTORE_ERROR.

Este es un campo de entrada. El valor inicial de este campo es 0. Este campo se ignora si *Version* es menor que MQSCO_VERSION_7.

HTTPSKeyStoreLength

V 10.0.0

Esta es la longitud de la ruta « HTTPSKeyStore ».

Este es un campo de entrada. El valor inicial de este campo es 0. Este campo se ignora si *Version* es menor que MQSCO_VERSION_7.

HTTPSCertValidation

V 10.0.0

Especifica el nivel de validación de certificados que se requiere para las conexiones HTTPS. Puede establecer este campo en uno de los siguientes valores:

MQ_HTTPSCERTVAL_DEFAULT

La política de validación se toma del atributo HTTPSCertValidation del mqclient.ini archivo, si se proporciona. Si no existe tal configuración, el comportamiento es el mismo que para MQ_HTTPSCERTVAL_ANY.

MQ_HTTPSCERTVAL_CUALQUIERA

Comprueba únicamente que el certificado del par sea válido y esté firmado por una autoridad de confianza.

MQ_HTTPSCERTVAL_HOSTNAMECN

Compruebe que el certificado del par sea válido y esté firmado por una autoridad de confianza, y que el nombre distintivo del certificado coincida con el del servidor HTTP con el que se está contactando.

MQ_HTTPSCERTVAL_NONE

No compruebe el certificado del par. La sesión está cifrada con TLS, pero no hay garantía de que el cliente se esté comunicando con el host deseado. Utilice esta opción con precaución; su uso podría, por ejemplo, permitir un ataque de tipo «man-in-the-middle» que redirija al cliente a un archivo CCDT malicioso.

Este campo se ignora si *Version* es menor que MQSCO_VERSION_7.

HTTPSCertRevocation

— V 1000

Especifica el nivel de comprobación de revocación de certificados que se requiere para las conexiones HTTPS. Puede establecer este campo en uno de los siguientes valores:

MQ_HTTPSCERTREV_DEFAULT

La política de validación viene determinada por el atributo HTTPSCertRevocation del `mqclient.ini` archivo, si se proporciona. Si no existe tal configuración, el comportamiento es el mismo que para MQ_HTTPSCERTREV_REQUIRED.

MQ_HTTPSCERTREV_REQUIRED

Compruebe la información del servidor de revocación de certificados, si se proporciona, para validar que el certificado no ha sido revocado. Si esta comprobación falla, por ejemplo, porque el servicio de revocación no está disponible, la conexión HTTPS falla.

MQ_HTTPSCERTREV_OPTIONAL

Ejecutar comprobaciones de revocación de certificados, pero si no se puede acceder al servicio de revocación, se permite que la conexión continúe.

MQ_HTTPSCERTREV_DESACTIVADO

No realice comprobaciones de revocación de certificados.

Este campo se ignora si *Version* es menor que MQSCO_VERSION_7.

MQSD - Descriptor de suscripción

La estructura MQSD se utiliza para especificar detalles sobre la suscripción que se está realizando. La estructura es un parámetro de entrada/salida en la llamada MQSUB. Para obtener más información, consulte [las notas de uso de MQSUB](#).

Disponibilidad

La estructura MQSD está disponible en las siguientes plataformas:

-  AIX
-  IBM i
-  Linux
-  Windows
-  z/OS

y para IBM MQ MQI clients conectados a estos sistemas.

Versión

La versión actual de MQSD es MQSD_VERSION_1.

Conjunto de caracteres y codificación

Los datos en MQSD deben estar en el juego de caracteres proporcionado por el atributo del **CodedCharSetId** gestor de colas y la codificación del gestor de colas local proporcionada por MQENC_NATIVE. Sin embargo, si la aplicación se ejecuta como un cliente MQ MQI, la estructura debe estar en el conjunto de caracteres y la codificación del cliente.

Suscripciones gestionadas

Si una aplicación no tiene una necesidad específica de utilizar una cola concreta como destino para las publicaciones que coinciden con su suscripción, puede utilizar la función de suscripción administrada. Si una aplicación opta por utilizar una suscripción gestionada, el gestor de colas informa al suscriptor sobre

el destino al que se envían los mensajes publicados, proporcionando un identificador de objeto como salida de la llamada MQSUB. Para obtener más información, consulte [Hobj \(MQHOBJ\)](#): entrada/salida.

Cuando se elimina la suscripción, el gestor de colas también se encarga de limpiar los mensajes que no se han recuperado del destino gestionado, en las siguientes situaciones:

- Cuando se elimina la suscripción (mediante el uso de MQCLOSE con MQCO_REMOVE_SUB) y se cierra el Hobj gestionado.
- De forma implícita cuando se pierde la conexión con una aplicación que utiliza una suscripción no duradera (MQSO_NON_DURABLE)
- Por caducidad, cuando se elimina una suscripción porque ha caducado y se cierra el Hobj gestionado.

Debe utilizar suscripciones gestionadas con suscripciones no duraderas, para que se pueda realizar esta limpieza y para que los mensajes de las suscripciones no duraderas cerradas no ocupen espacio en el gestor de colas. Las suscripciones duraderas también pueden utilizar destinos gestionados.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
StrucId (identificador de estructura)	MQSD_STRUC_ID	'SD→→'
Versión (número de versión de la estructura)	MQSD_VERSION_1	1
Opciones (opciones)	MQSO_NO_DURADERO	0
ObjectName (nombre del objeto)	Ninguna	Cadena nula o espacios en blanco
AlternateUserId (ID de usuario alternativo)	Ninguna	Cadena nula o espacios en blanco
AlternateSecurityId (identificador de seguridad alternativo)	MQSID_NINGUNO	Nulos
SubExpiry (caducidad de la suscripción)	MQEI_UNLIMITED	-1
ObjectString (cadena de objetos)	Ninguna	Nombres y valores tal y como se definen para MQCHARV
SubName (nombre de la suscripción)	Ninguna	Nombres y valores tal y como se definen para MQCHARV
SubUserData (datos de usuario suscrito)	Ninguna	Nombres y valores tal y como se definen para MQCHARV
SubCorrelId (ID de correlación de suscripción)	MQCI_NONE	Nulos
PubPriority (prioridad de publicación)	MQPRI_PRIORITY_AS_Q_DEF	-3
PubAccountingToken (token de contabilidad de publicación)	MQACT_NONE	Nulos
PubAppIdentityData (datos de identidad de la solicitud de publicación)	Ninguna	Cadena nula o espacios en blanco

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>SelectionString</u> (cadena que proporciona criterios de selección)	Ninguna	Nombres y valores tal y como se definen para MQCHARV
<u>SubLevel</u> (nivel de suscripción)	Ninguna	1
<u>ResObjectString</u> (nombre largo del objeto)	Ninguna	Nombres y valores tal y como se definen para MQCHARV
Notas: 1. El símbolo ¬ representa un único carácter en blanco. 2. El valor «Cadena nula o espacios en blanco» denota la cadena nula en C y los caracteres en blanco en otros lenguajes de programación. 3. En el lenguaje de programación C, la variable macroMQSD_DEFAULT contiene los valores que se enumeran en la tabla. Se puede utilizar de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQSD MySD = {MQSD_DEFAULT};</pre>		

Declaraciones lingüísticas

Declaración C para MQSD

```
typedef struct tagMQSD MQSD;
struct tagMQSD {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQLONG     Options;          /* Options associated with subscribing */
    MQCHAR48    ObjectName;       /* Object name */
    MQCHAR12    AlternateUserId;  /* Alternate user identifier */
    MQBYTE40    AlternateSecurityId; /* Alternate security identifier */
    MQLONG     SubExpiry;        /* Expiry of Subscription */
    MQCHARV     ObjectString;     /* Object Long name */
    MQCHARV     SubName;         /* Subscription name */
    MQCHARV     SubUserData;     /* Subscription User data */
    MQBYTE24    SubCorrelId;      /* Correlation Id related to this subscription */
    MQLONG     PubPriority;       /* Priority set in publications */
    MQBYTE32    PubAccountingToken; /* Accounting Token set in publications */
    MQCHAR32    PubApplIdentityData; /* Appl Identity Data set in publications */
    MQCHARV     SelectionString;  /* Message selector structure */
    MQLONG     SubLevel;         /* Subscription level */
    MQCHARV     ResObjectString;  /* Resolved Long object name*/
    /* Ver:1 */
};
```

COBOL declaración para MQSD

```
** Address of variable length string
20 MQSD-OBJECTSTRING-VSPTR      POINTER.
** Offset of variable length string
20 MQSD-OBJECTSTRING-VSOFFSET   PIC S9(9) BINARY.
** size of buffer
20 MQSD-OBJECTSTRING-VSBUFSIZE  PIC S9(9) BINARY.
** Length of variable length string
20 MQSD-OBJECTSTRING-VSLENGTH  PIC S9(9) BINARY.
** CCSID of variable length string
20 MQSD-OBJECTSTRING-VSCCSID   PIC S9(9) BINARY.
** Subscription name
15 MQSD-SUBNAME.
** Address of variable length string
20 MQSD-SUBNAME-VSPTR          POINTER.
** Offset of variable length string
20 MQSD-SUBNAME-VSOFFSET       PIC S9(9) BINARY.
** size of buffer
```

```

20 MQSD-SUBNAME-VSBUFSIZE          PIC S9(9) BINARY.
** Length of variable length string
20 MQSD-SUBNAME-VSLENGTH           PIC S9(9) BINARY.
** CCSID of variable length string
20 MQSD-SUBNAME-VSCCSID           PIC S9(9) BINARY.
** Subscription User data
15 MQSD-SUBUSERDATA.
** Address of variable length string
20 MQSD-SUBUSERDATA-VSPTR         POINTER.
** Offset of variable length string
20 MQSD-SUBUSERDATA-VSOFFSET      PIC S9(9) BINARY.
** size of buffer
20 MQSD-SUBUSERDATA-VSBUFSIZE     PIC S9(9) BINARY.
** Length of variable length string
20 MQSD-SUBUSERDATA-VSLENGTH      PIC S9(9) BINARY.
** CCSID of variable length string
20 MQSD-SUBUSERDATA-VSCCSID       PIC S9(9) BINARY.
** Correlation Id related to this subscription
15 MQSD-SUBCORRELID              PIC X(24).
** Priority set in publications
15 MQSD-PUBPRIORITY              PIC S9(9) BINARY.
** Accounting Token set in publications
15 MQSD-PUBACCOUNTINGTOKEN        PIC X(32).
** Appl Identity Data set in publications
15 MQSD-PUBAPPLIDENTITYDATA       PIC X(32).
** Message Selector
15 MQSD-SELECTIONSTRING.
** Address of variable length string
20 MQSD-SELECTIONSTRING-VSPTR     POINTER.
** Offset of variable length string
20 MQSD-SELECTIONSTRING-VSOFFSET  PIC S9(9) BINARY.
** size of buffer
20 MQSD-SELECTIONSTRING-VSBUFSIZE PIC S9(9) BINARY.
** Length of variable length string
20 MQSD-SELECTIONSTRING-VSLENGTH  PIC S9(9) BINARY.
** CCSID of variable length string
20 MQSD-SELECTIONSTRING-VSCCSID   PIC S9(9) BINARY.
** Selection criteria
20 MQSD-SELECTIONSTRING-SUBLEVEL  PIC S9(9) BINARY.
** Long object name
20 MQSD-SELECTIONSTRING-RESOBJSTRING PIC S9(9) BINARY.

```

PL/I declaración para MQSD

```

dcl
1 MQSD based,
3 StructId      char(4),      /* Structure identifier */
3 Version       fixed bin(31), /* Structure version number */
3 Options       fixed bin(31), /* Options associated with subscribing */
3 ObjectName    char(48),     /* Object name */
3 AlternateUserId char(12),    /* Alternate user identifier */
3 AlternateSecurityId char(40), /* Alternate security identifier */
3 SubExpiry     fixed bin(31), /* Expiry of Subscription */
3 ObjectString, /* Object Long name */
5 VSPtr         pointer,      /* Address of variable length string */
5 VSOffset      fixed bin(31), /* Offset of variable length string */
5 VSBufSize     fixed bin(31), /* size of buffer */
5 VSLength      fixed bin(31), /* Length of variable length string */
5 VSCCSID       fixed bin(31); /* CCSID of variable length string */
3 SubName,      /* Subscription name */
5 VSPtr         pointer,      /* Address of variable length string */
5 VSOffset      fixed bin(31), /* Offset of variable length string */
5 VSBufSize     fixed bin(31), /* size of buffer */
5 VSLength      fixed bin(31), /* Length of variable length string */
5 VSCCSID       fixed bin(31); /* CCSID of variable length string */
3 SubUserData, /* Subscription User data */
5 VSPtr         pointer,      /* Address of variable length string */
5 VSOffset      fixed bin(31), /* Offset of variable length string */
5 VSBufSize     fixed bin(31), /* size of buffer */
5 VSLength      fixed bin(31), /* Length of variable length string */
5 VSCCSID       fixed bin(31); /* CCSID of variable length string */
3 SubCorrelId   char(24),     /* Correlation Id related to this subscription */
3 PubPriority    fixed bin(31), /* Priority set in publications */
3 PubAccountingToken char(32), /* Accounting Token set in publications */
3 PubApplIdentityData char(32), /* Appl Identity Data set in publications */
3 SelectionString, /* Message Selection */
5 VSPtr         pointer,      /* Address of variable length string */
5 VSOffset      fixed bin(31), /* Offset of variable length string */
5 VSBufSize     fixed bin(31), /* size of buffer */
5 VSLength      fixed bin(31), /* Length of variable length string */

```

```

5 VSCCSID      fixed bin(31), /* CCSID of variable length string */
3 SubLevel     fixed bin(31), /* Subscription level */
3 ResObjectString, /* Resolved Long object name */
5 VSPtr        pointer, /* Address of variable length string */
5 VSOFFSET     fixed bin(31), /* Offset of variable length string */
5 VSBufSize     fixed bin(31), /* size of buffer */
5 VSLength     fixed bin(31), /* Length of variable length string */
5 VSCCSID      fixed bin(31); /* CCSID of variable length string */

```

High Level Assembler declaración para MQSD

```

MQSD          DSECT
MQSD_STRUCID   DS CL4  Structure identifier
MQSD_VERSION   DS F    Structure version number
MQSD-OPTIONS   DS F    Options associated with subscribing
MQSD_OBJECTNAME DS CL48 Object name
MQSD_ALTERNATEUSERID DS CL12 Alternate user identifier
MQSD_ALTERNATESECURITYID DS CL40 Alternate security identifier
MQSD_SUBEXPIRY DS F    Expiry of Subscription
MQSD_OBJECTSTRING DS 0F Object Long name
MQSD_OBJECTSTRING_VSPTR DS F Address of variable length string
MQSD_OBJECTSTRING_VSOFFSET DS F Offset of variable length string
MQSD_OBJECTSTRING_VSBUFFSIZE DS F size of buffer
MQSD_OBJECTSTRING_VSLLENGTH DS F Length of variable length string
MQSD_OBJECTSTRING_VSCCSID DS F CCSID of variable length string
MQSD_OBJECTSTRING_LENGTH EQU *-MQSD_OBJECTSTRING
ORG MQSD_OBJECTSTRING
MQSD_OBJECTSTRING_AREA DS CL(MQSD_OBJECTSTRING_LENGTH)
*
MQSD_SUBNAME   DS 0F Subscription name
MQSD_SUBNAME_VSPTR DS F Address of variable length string
MQSD_SUBNAME_VSOFFSET DS F Offset of variable length string
MQSD_SUBNAME_VSBUFFSIZE DS F size of buffer
MQSD_SUBNAME_VSLLENGTH DS F Length of variable length string
MQSD_SUBNAME_VSCCSID DS F CCSID of variable length string
MQSD_SUBNAME_LENGTH EQU *-MQSD_SUBNAME
ORG MQSD_SUBNAME
MQSD_SUBNAME_AREA DS CL(MQSD_SUBNAME_LENGTH)
*
MQSD_SUBUSERDATA DS 0F Subscription User data
MQSD_SUBUSERDATA_VSPTR DS F Address of variable length string
MQSD_SUBUSERDATA_VSOFFSET DS F Offset of variable length string
MQSD_SUBUSERDATA_VSBUFFSIZE DS F size of buffer
MQSD_SUBUSERDATA_VSLLENGTH DS F Length of variable length string
MQSD_SUBUSERDATA_VSCCSID DS F CCSID of variable length string
MQSD_SUBUSERDATA_LENGTH EQU *-MQSD_SUBUSERDATA
ORG MQSD_SUBUSERDATA
MQSD_SUBUSERDATA_AREA DS CL(MQSD_SUBUSERDATA_LENGTH)
*
MQSD_SUBCORRELID DS CL24 Correlation Id related to this subscription
MQSD_PUBPRIORITY DS F Priority set in publications
MQSD_PUBACCOUNTINGTOKEN DS CL32 Accounting Token set in publications
MQSD_PUBAPPLIDENTITYDATA DS CL32 Appl Identity Data set in publications
*
MQSD_SELECTIONSTRING DS F Message Selector
MQSD_SELECTIONSTRING_VSPTR DS F Address of variable length string
MQSD_SELECTIONSTRING_VSOFFSET DS F Offset of variable length string
MQSD_SELECTIONSTRING_VSBUFFSIZE DS F size of buffer
MQSD_SELECTIONSTRING_VSLLENGTH DS F Length of variable length string
MQSD_SELECTIONSTRING_VSCCSID DS F CCSID of variable length string
MQSD_SELECTIONSTRING_LENGTH EQU *-MQSD_SELECTIONSTRING
ORG MQSD_SELECTIONSTRING
MQSD_SELECTIONSTRING_AREA DS CL(MQSD_SELECTIONSTRING_LENGTH)
*
MQSD-SUBLEVEL DS F Subscription level
*
MQSD_RESOBJECTSTRING DS F Resolved Long object name
MQSD_RESOBJECTSTRING_VSPTR DS F Address of variable length string
MQSD_RESOBJECTSTRING_VSOFFSET DS F Offset of variable length string
MQSD_RESOBJECTSTRING_VSBUFFSIZE DS F size of buffer
MQSD_RESOBJECTSTRING_VSLLENGTH DS F Length of variable length string
MQSD_RESOBJECTSTRING_VSCCSID DS F CCSID of variable length string
MQSD_RESOBJECTSTRING_LENGTH EQU *-MQSD_RESOBJECTSTRING
ORG MQSD_RESOBJECTSTRING
MQSD_RESOBJECTSTRING_AREA DS CL(MQSD_RESOBJECTSTRING_LENGTH)
*
MQSD_LENGTH EQU *-MQSD
ORG MQSD
MQSD_AREA DS CL(MQSD_LENGTH)

```

Detalles de campo para MQSD

La estructura del descriptor de suscripción MQSD se utiliza para especificar detalles sobre la suscripción que se está realizando. La estructura es un parámetro de entrada/salida en la llamada MQSUB.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>StrucId</u> (identificador de estructura)	MQSD_STRUC_ID	'SD--'
<u>Versión</u> (número de versión de la estructura)	MQSD_VERSION_1	1
<u>Opciones</u> (opciones)	MQSO_NO_DURADERO	0
<u>ObjectName</u> (nombre del objeto)	Ninguna	Cadena nula o espacios en blanco
<u>AlternateUserId</u> (ID de usuario alternativo)	Ninguna	Cadena nula o espacios en blanco
<u>AlternateSecurityId</u> (identificador de seguridad alternativo)	MQSID_NINGUNO	Nulos
<u>SubExpiry</u> (caducidad de la suscripción)	MQEI_UNLIMITED	-1
<u>ObjectString</u> (cadena de objetos)	Ninguna	Nombres y valores tal y como se definen para MQCHARV
<u>SubName</u> (nombre de la suscripción)	Ninguna	Nombres y valores tal y como se definen para MQCHARV
<u>SubUserData</u> (datos de usuario suscrito)	Ninguna	Nombres y valores tal y como se definen para MQCHARV
<u>SubCorrelId</u> (ID de correlación de suscripción)	MQCI_NONE	Nulos
<u>PubPriority</u> (prioridad de publicación)	MQPRI_PRIORITY_AS_Q_DEF	-3
<u>PubAccountingToken</u> (token de contabilidad de publicación)	MQACT_NONE	Nulos
<u>PubAppIdentityData</u> (datos de identidad de la solicitud de publicación)	Ninguna	Cadena nula o espacios en blanco
<u>SelectionString</u> (cadena que proporciona criterios de selección)	Ninguna	Nombres y valores tal y como se definen para MQCHARV
<u>SubLevel</u> (nivel de suscripción)	Ninguna	1
<u>ResObjectString</u> (nombre largo del objeto)	Ninguna	Nombres y valores tal y como se definen para MQCHARV

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
Notas: 1. El símbolo <code>↵</code> representa un único carácter en blanco. 2. El valor «Cadena nula o espacios en blanco» denota la cadena nula en C y los caracteres en blanco en otros lenguajes de programación. 3. En el lenguaje de programación C, la variable macro <code>MQSD_DEFAULT</code> contiene los valores que se enumeran en la tabla. Se puede utilizar de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQSD MySD = {MQSD_DEFAULT};</pre>		

StrucId (MQCHAR4)

Este es el identificador de estructura de la estructura del descriptor de suscripción. Siempre es un campo de entrada. Su valor es `MQSD_STRUC_ID`.

El valor debe ser:

MQSD_STRUC_ID

Identificador de la estructura del descriptor de suscripción.

Para el lenguaje de programación C, también se define la constante `MQSD_STRUC_ID_ARRAY`. Tiene el mismo valor que `MQSD_STRUC_ID`, pero es una matriz de caracteres en lugar de una cadena.

Versión (MQLONG)

Este es el número de versión de la estructura; el valor debe ser:

MQSD_VERSION_1

Version-1 Estructura del descriptor de suscripción.

La siguiente constante especifica el número de versión de la versión actual:

MQSD_VERSIÓN_ACTUAL

Versión actual de la estructura del descriptor de suscripción.

Este campo siempre es un campo de entrada. El valor inicial de este campo es `MQSD_VERSION_1`.

Opciones (MQLONG)

Esto proporciona opciones para controlar la acción de la llamada `MQSUB`.

Debe especificar al menos una de las siguientes opciones:

- `MQSO_ALTER`
- `MQSO_RESUME`
- `MQSO_CREATE`

Para especificar más de una opción, suma los valores (sin añadir la misma constante más de una vez) o combínalos mediante la operación OR bit a bit (si el lenguaje de programación admite operaciones bit a bit).

Las combinaciones que no son válidas se indican en este tema; cualquier otra combinación es válida.

Opciones de acceso o creación : Las opciones de acceso y creación controlan si se crea una suscripción o si se devuelve o modifica una suscripción existente. Debe especificar al menos una de estas opciones.

Tabla 555. Combinaciones válidas de opciones de acceso y creación	
Combinación de opciones	Notas
MQSO_CREATE	Crea una suscripción si no existe ninguna. Esta combinación falla si la suscripción existe.
MQSO_RESUME	Reanuda una suscripción existente. Esta combinación falla si no existe ninguna suscripción.
MQSO_CREATE + MQSO_RESUME	Crea una suscripción si no existe ninguna y reanuda una coincidente, si existe. Esta combinación resulta útil cuando se utiliza en una aplicación que se ejecuta varias veces.
MQSO_ALTER (véase la nota)	Reanuda una suscripción existente, modificando cualquier campo para que coincida con lo especificado en el MQSD. Esta combinación falla si no existe ninguna suscripción.
MQSO_CREATE + MQSO_ALTER (véase la nota)	Crea una suscripción si no existe ninguna y reanuda una que coincida, si existe, modificando los campos para que coincidan con los especificados en el MQSD. Esta combinación resulta útil cuando se utiliza en una aplicación que desea asegurarse de que su suscripción se encuentra en un estado determinado antes de continuar.

Nota:

Las opciones que especifican MQSO_ALTER también pueden especificar MQSO_RESUME, pero esta combinación no tiene ningún efecto adicional respecto a especificar solo MQSO_ALTER. MQSO_ALTER implica MQSO_RESUME, ya que llamar a MQSUB para modificar una suscripción implica que la suscripción también se reanuda. Sin embargo, lo contrario no es cierto: reanudar una suscripción no implica que se vaya a modificar.

MQSO_CREATE

Crea una nueva suscripción para el tema especificado. Si existe una suscripción *SubName* que utiliza el mismo, la llamada falla con MQRC_SUB_ALREADY_EXISTS. Este fallo se puede evitar combinando la opción MQSO_CREATE con MQSO_RESUME. No siempre *SubName* es necesario. Para obtener más detalles, consulte la descripción de ese campo.

La combinación de MQSO_CREATE con MQSO_RESUME devuelve un identificador a una suscripción preexistente para el especificado *SubName* si se encuentra una; si no existe ninguna suscripción, se crea una nueva utilizando todos los campos proporcionados en el MQSD.

MQSO_CREATE también se puede combinar con MQSO_ALTER para obtener un efecto similar.

MQSO_RESUME

Devuelve un identificador a una suscripción preexistente que coincide con la especificada por *SubName*. No se realizan cambios en los atributos de las suscripciones coincidentes y se devuelven en la salida en la estructura MQSD. Solo se utilizan los siguientes campos MQSD: StrucId, Version, Options, AlternateUserId y AlternateSecurityId, y SubName.

La llamada falla con el código de motivo MQRC_NO_SUBSCRIPTION si no existe una suscripción que coincida con el nombre completo de la suscripción. Este fallo se puede evitar combinando la opción MQSO_CREATE con MQSO_RESUME.

El ID de usuario de la suscripción es el ID de usuario que creó la suscripción o, si posteriormente ha sido modificado por otro ID de usuario, es el ID de usuario de la última modificación realizada con éxito. Si se utiliza un AlternateUserId y se permite el uso de ID de usuario alternativos para ese

usuario, el ID de usuario alternativo se registra como el ID de usuario que creó la suscripción, en lugar del ID de usuario con el que se realizó la suscripción.

Si existe una suscripción coincidente que se creó sin la opción MQSO_ANY_USERID y el ID de usuario de la suscripción es diferente al de la aplicación que solicita un identificador para la suscripción, la llamada falla con el código de motivo MQRC_IDENTITY_MISMATCH.

Si existe una suscripción coincidente y está actualmente en uso, la llamada falla con MQRC_SUBSCRIPTION_IN_USE.

Si la suscripción indicada en SubName no es una suscripción válida para reanudar o modificar desde una aplicación, la llamada falla con MQRC_INVALID_SUBSCRIPTION.

MQSO_RESUME está implícito en MQSO_ALTER, por lo que no es necesario combinarlo con esa opción. Sin embargo, combinar las dos opciones no provoca ningún error.

MQSO_ALTER

Devuelve un identificador a una suscripción preexistente con el nombre completo de la suscripción que coincide con el especificado por el nombre en *SubName*. Cualquier atributo de la suscripción que difiera del especificado en el MQSD se modifica en la suscripción, a menos que no se permita la modificación de ese atributo. Los detalles se indican en la descripción de cada atributo y se resumen en la siguiente tabla. Si intenta modificar un atributo que no se puede cambiar, o modificar una suscripción que tiene establecida la opción MQSO_IMMUTABLE, la llamada fallará con el código de motivo que se muestra en la tabla siguiente.

La llamada falla con el código de motivo MQRC_NO_SUBSCRIPTION si no existe una suscripción que coincida con el nombre completo de la suscripción. Puede evitar este error combinando la opción MQSO_CREATE con MQSO_ALTER.

La combinación de MQSO_CREATE con MQSO_ALTER devuelve un identificador a una suscripción preexistente para el especificado *SubName* si se encuentra una; si no existe ninguna suscripción, se crea una nueva utilizando todos los campos proporcionados en el MQSD.

El ID de usuario de la suscripción es el ID de usuario que creó la suscripción o, si posteriormente es modificado por un ID de usuario diferente, es el ID de usuario de la última modificación realizada con éxito. Si se utiliza una suscripción de usuario (AlternateUserId) y se permite el uso de ID de usuario alternativos para ese usuario, entonces el ID de usuario alternativo se registra como el ID de usuario que creó la suscripción, en lugar del ID de usuario con el que se realizó la suscripción.

Si existe una suscripción coincidente que se creó sin la opción MQSO_ANY_USERID y el ID de usuario de la suscripción es diferente al de la aplicación que solicita un identificador para la suscripción, la llamada falla con el código de motivo MQRC_IDENTITY_MISMATCH.

Si existe una suscripción coincidente y está actualmente en uso, la llamada falla con MQRC_SUBSCRIPTION_IN_USE.

Si la suscripción indicada en SubName no es una suscripción válida para reanudar o modificar desde una aplicación, la llamada falla con MQRC_INVALID_SUBSCRIPTION.

La siguiente tabla muestra la capacidad de MQSO_ALTER para modificar los valores de los atributos en MQSD y MQSUB.

Tabla 556. Atributos en MQSD y MQSUB que se pueden modificar			
Descriptor de tipo de datos o llamada a función	Nombre de campo	¿Se puede modificar este atributo utilizando MQSO_ALTER?	Código de razón
MQSD	Opciones de durabilidad	No	MQRC_DURABILIDAD_NO_MODIFICABLE
MQSD	Opciones de destino	Sí	Ninguna
MQSD	Opciones de registro	Sí (véase la nota “1” en la página 567)	MQRC_GROUPING_NOT_ALTERABLE si intenta modificar MQSO_GROUP_SUB
MQSD	Opciones de publicación	Sí (véase la nota “2” en la página 567)	Ninguna
MQSD	Opciones comodín	No	MQRC_TEMA_NO_MODIFICABLE

Tabla 556. Atributos en MQSD y MQSUB que se pueden modificar (continuación)

Descriptor de tipo de datos o llamada a función	Nombre de campo	¿Se puede modificar este atributo utilizando MQSO ALTER?	Código de razón
MQSD	Otras opciones	No (véase la nota “3” en la página 567)	Ninguna
MQSD	ObjectName	No	MQRC_TEMA_NO_MODIFICABLE
MQSD	AlternateUserId	No (véase la nota “4” en la página 567)	Ninguna
MQSD	AlternateSecurityId	No (véase la nota “4” en la página 567)	Ninguna
MQSD	SubExpiry	Sí	Ninguna
MQSD	ObjectString	No	MQRC_TEMA_NO_MODIFICABLE
MQSD	SubName	No (véase la nota “5” en la página 567)	Ninguna
MQSD	SubUserData	Sí	Ninguna
MQSD	SubCorrelId	Sí (véase la nota “6” en la página 567)	MQRC_GROUPING_NOT_ALTERABLE cuando se trata de una suscripción agrupada
MQSD	PubPriority	Sí	Ninguna
MQSD	PubAccountingToken	Sí	Ninguna
MQSD	PubApplIdentityData	Sí	Ninguna
MQSD	SubLevel	No	MQRC_SUBNIVEL_NO_MODIFICABLE
MQSUB	Hobj	Sí (véase la nota “6” en la página 567)	MQRC_GROUPING_NOT_ALTERABLE cuando se trata de una suscripción agrupada

Notas:

1. MQSO_GROUP_SUB no se puede modificar.
2. MQSO_NEW_PUBLICATIONS_ONLY no se puede modificar porque no forma parte de la suscripción
3. Estas opciones no forman parte de la suscripción
4. Este atributo no forma parte de la suscripción
5. Este atributo es la identidad de la suscripción que se está modificando
6. Modificable excepto cuando forma parte de un subgrupo (MQSO_GROUP_SUB)

Opciones de durabilidad

Las siguientes opciones controlan la duración de la suscripción. Solo puede especificar una de estas opciones. Si está modificando una suscripción existente utilizando la opción MQSO ALTER, no puede cambiar la durabilidad de la suscripción. Al regresar de una llamada MQSUB utilizando MQSO_RESUME, se establece la opción de durabilidad adecuada.

MQSO_DURABLE

Solicite que la suscripción a este tema permanezca hasta que se elimine explícitamente mediante MQCLOSE con la opción MQCO_REMOVE_SUB. Si esta suscripción no se elimina explícitamente, permanecerá incluso después de que se cierre la conexión de esta aplicación al gestor de colas.

Si se solicita una suscripción duradera a un tema que está definido como no permitir suscripciones duraderas, la llamada falla con MQRC_DURABILITY_NOT_ALLOWED.

MQSO_NO_DURADERO

Solicite que se elimine la suscripción a este tema cuando se cierre la conexión de las aplicaciones al gestor de colas, si aún no se ha eliminado explícitamente. MQSO_NON_DURABLE es lo contrario de la opción MQSO_DURABLE y se define para facilitar la documentación del programa. Es el valor predeterminado si no se especifica ninguno.

Opciones de destino

La siguiente opción controla el destino al que se envían las publicaciones de un tema al que se ha suscrito. Si se modifica una suscripción existente utilizando la opción MQSO_ALTER, se puede cambiar el destino utilizado para las publicaciones de la suscripción. Al regresar de una llamada MQSUB utilizando MQSO_RESUME, esta opción se establece si procede.

MQSO_GESTIONADO

Solicite que el destino al que se envían las publicaciones sea gestionado por el gestor de colas.

El identificador de objeto devuelto en *Hobj* representa una cola gestionada por el gestor de colas y se utiliza con las llamadas MQGET, MQCB, MQINQ o MQCLOSE posteriores.

No se puede proporcionar un identificador de objeto devuelto de una llamada MQSUB anterior en el **Hobj** parámetro cuando no se especifica MQSO_MANAGED.

MQSO_SIN_MULTIDIFUSIÓN

Solicite que el destino al que se envían las publicaciones no sea una dirección de grupo de multidifusión. Esta opción solo es válida cuando se combina con la opción MQSO_MANAGED. Cuando se proporciona un identificador de cola en el **Hobj** parámetro, no se puede utilizar la multidifusión para esta suscripción y la opción no es válida.

Si el tema está definido para permitir únicamente suscripciones multicast, utilizando la MCAST (ONLY) configuración, la llamada fallará con el código de motivo MQRC_MULTICAST_REQUIRED.

Opción de alcance

La siguiente opción controla el alcance de la suscripción que se está realizando. Si se modifica una suscripción existente utilizando la opción MQSO_ALTER, esta opción de ámbito de suscripción no se puede cambiar. Al regresar de una llamada MQSUB utilizando MQSO_RESUME, se establece la opción de ámbito adecuada.

MQSO_ÁMBITO_QMGR

Esta suscripción solo se realiza en el gestor de colas local. No se distribuye ninguna suscripción proxy a otros gestores de colas de la red. Solo las publicaciones que se publican en este gestor de colas se envían a este suscriptor. Esto anula cualquier comportamiento establecido mediante el atributo de tema SUBSCOPE.

Nota: Si no se establece, el ámbito de la suscripción viene determinado por el atributo de tema SUBSCOPE.

Opciones de registro

Las siguientes opciones controlan los detalles del registro que se realiza en el gestor de colas para esta suscripción. Si se modifica una suscripción existente utilizando la opción MQSO_ALTER, se pueden cambiar estas opciones de registro. Al regresar de una llamada MQSUB utilizando MQSO_RESUME, se establecen las opciones de registro adecuadas.

MQSO_GRUPO_SUB

Esta suscripción se agrupará con otras suscripciones del mismo SubLevel utilizando la misma cola y especificando el mismo ID de correlación, de modo que cualquier publicación en temas que provoque que se envíe más de un mensaje de publicación al grupo de suscripciones, debido a un conjunto superpuesto de cadenas de temas utilizadas, solo provoque que se envíe un mensaje a la cola. Si no se utiliza esta opción, cada suscripción única (identificada por SubName) que coincida recibirá una copia de la publicación, lo que podría significar que se coloquen más de una copia de la publicación en la cola compartida por varias suscripciones.

Solo la suscripción más importante del grupo recibe una copia de la publicación. La suscripción más significativa se basa en el nombre completo del tema hasta el punto en que se encuentra un comodín. Si se utiliza una combinación de esquemas de comodines dentro del grupo, solo es importante la posición del comodín. Se recomienda no combinar diferentes esquemas de comodines dentro de un grupo de suscripciones que comparten la misma cola.

Al crear una nueva suscripción agrupada, esta debe seguir teniendo un identificador único (SubName,), pero si coincide con el nombre completo del tema de una suscripción existente en el grupo, la llamada falla con el error MQRC_DUPLICATE_GROUP_SUB.

Si la suscripción más significativa del grupo también especifica MQSO_NOT_OWN_PUBS y se trata de una publicación de la misma aplicación, no se entregará ninguna publicación a la cola.

Al modificar una suscripción realizada con esta opción, no se pueden cambiar los campos que implican la agrupación, Hobj en la llamada MQSUB (que representa el nombre de la cola y del gestor de colas) y el campo « SubCorrelId ». Si se intenta modificarlos, la llamada fallará con el error MQRC_GROUPING_NOT_ALTERABLE.

Esta opción debe combinarse con MQSO_SET_CORREL_ID con un identificador de correlación (SubCorrelId) que no esté establecido en MQCI_NONE, y no puede combinarse con MQSO_MANAGED.

MQSO_ANY_USERID

Cuando se especifica MQSO_ANY_USERID, la identidad del suscriptor no está restringida a un ID de usuario único. Esto permite que cualquier usuario modifique o reanude la suscripción cuando disponga de la autoridad adecuada. Sólo puede tener la suscripción un único usuario a la vez. Si intentas volver a usar una suscripción que ya está usando otra aplicación, la llamada fallará con MQRC_SUBSCRIPTION_IN_USE.

Para añadir esta opción a una suscripción existente, la llamada MQSUB (utilizando MQSO_ALTER) debe provenir del mismo ID de usuario que la suscripción original.

Si una llamada MQSUB hace referencia a una suscripción existente con MQSO_ANY_USERID establecido y el ID de usuario difiere de la suscripción original, la llamada sólo será satisfactoria si el nuevo ID de usuario tiene autorización para suscribirse al tema. Una vez completado con éxito, las futuras publicaciones para este suscriptor se colocan en la cola de suscriptores con el nuevo ID de usuario establecido en el mensaje de publicación.

No especifique tanto MQSO_ANY_USERID como MQSO_FIXED_USERID. Si no se especifica ninguno, el valor predeterminado es MQSO_FIXED_USERID.

MQSO_FIXED_USERID

Cuando se especifica MQSO_FIXED_USERID, la suscripción solo puede ser modificada o reanudada por el último ID de usuario que modificó la suscripción. Si la suscripción no se ha modificado, es el ID de usuario que creó la suscripción.

Si un verbo MQSUB hace referencia a una suscripción existente con MQSO_ANY_USERID establecido y modifica la suscripción utilizando MQSO_ALTER para utilizar la opción MQSO_FIXED_USERID, el ID de usuario de la suscripción queda fijado en este nuevo ID de usuario. La llamada sólo es satisfactoria si el nuevo ID de usuario tiene autoridad para suscribirse al tema.

Si un ID de usuario distinto al registrado como propietario de una suscripción intenta reanudar o modificar una suscripción MQSO_FIXED_USERID, la llamada falla con MQRC_IDENTITY_MISMATCH. El ID de usuario propietario de una suscripción se puede ver mediante el mandato DISPLAY SBSTATUS.

No especifique tanto MQSO_ANY_USERID como MQSO_FIXED_USERID. Si no se especifica ninguno, el valor predeterminado es MQSO_FIXED_USERID.

Opciones de publicación

Las siguientes opciones controlan la forma en que se envían las publicaciones a este suscriptor. Si se modifica una suscripción existente utilizando la opción MQSO_ALTER, se pueden cambiar estas opciones de publicación.

MQSO_NO_PROPIO_PUBLICACIONES

Indica al agente que la aplicación no desea ver ninguna de sus propias publicaciones. Las publicaciones se consideran originadas en la misma aplicación si los identificadores de conexión son los mismos. Al regresar de una llamada MQSUB utilizando MQSO_RESUME, esta opción se establece si procede.

MQSO_NUEVAS_PUBLICACIONES_SOLO

No se enviarán publicaciones actualmente retenidas, cuando se cree esta suscripción, solo nuevas publicaciones. Esta opción solo se aplica cuando se especifica MQSO_CREATE. Cualquier cambio posterior en una suscripción no altera el flujo de publicaciones, por lo que cualquier publicación conservada sobre un tema ya se habrá enviado al suscriptor como nueva publicación.

Si se especifica esta opción sin MQSO_CREATE, la llamada falla con MQRC_OPTIONS_ERROR. Al regresar de una llamada MQSUB utilizando MQSO_RESUME, esta opción no se establece aunque la suscripción se haya creado utilizando esta opción.

Si no se utiliza esta opción, los mensajes retenidos anteriormente se envían a la cola de destino proporcionada. Si esta acción falla debido a un error, ya sea MQRC_RETAINED_MSG_Q_ERROR o MQRC_RETAINED_NOT_DELIVERED, la creación de la suscripción falla.

MQSO_PUBLICACIONES_A_PEDIDO

Al seleccionar esta opción, se indica que el suscriptor solicitará información específicamente cuando sea necesario. El gestor de colas no envía mensajes no solicitados al suscriptor. La publicación retenida (o posiblemente varias publicaciones si se especifica un comodín en el tema) se envía al suscriptor cada vez que se realiza una llamada MQSUBRQ utilizando el identificador Hsub de una llamada MQSUB anterior. No se envían publicaciones como resultado de la llamada MQSUB utilizando esta opción. Al regresar de una llamada MQSUB utilizando MQSO_RESUME, esta opción se establece si procede.

Esta opción no es válida en combinación con un valor de « SubLevel » superior a 1.

Opciones de lectura anticipada

Las siguientes opciones controlan si los mensajes no persistentes se envían a una aplicación antes de que esta los solicite.

MQSO_LEER_POR_ADELANTE_COMO_Q_DEF

Si la llamada MQSUB utiliza un identificador administrado, el atributo de lectura anticipada predeterminado de la cola modelo asociada al tema suscrito determina si los mensajes se envían a la aplicación antes de que esta los solicite.

Éste es el valor predeterminado.

MQSO_NO_LEER_POR_ADELANTE

Si la llamada MQSUB utiliza un identificador administrado, los mensajes no se envían a la aplicación antes de que esta los solicite.

MQSO_LEER_POR_ADELANTE

Si la llamada MQSUB utiliza un identificador administrado, es posible que los mensajes se envíen a la aplicación antes de que esta los solicite.

Nota: Las siguientes notas se aplican a las opciones de lectura anticipada:

1. Solo se puede especificar una de estas opciones. Si se especifican tanto MQOO_READ_AHEAD como MQOO_NO_READ_AHEAD, se devuelve el código de motivo MQRC_OPTIONS_ERROR. Estas opciones solo son aplicables si se especifica MQSO_MANAGED.
2. No son aplicables para MQSUB cuando se pasa una cola que se ha abierto previamente. Es posible que la lectura anticipada no esté habilitada cuando se solicite. Las opciones MQGET utilizadas en la primera llamada MQGET pueden impedir que se habilite la lectura anticipada. Además, la lectura anticipada se desactiva cuando el cliente se conecta a un gestor de colas que no admite la lectura anticipada. Si la aplicación no se está ejecutando como cliente IBM MQ, estas opciones se ignoran.

Opciones comodín

Las siguientes opciones controlan cómo se interpretan los comodines en la cadena proporcionada en el campo « ObjectString » (Comodines de búsqueda) del MQSD. Solo puede especificar una de estas opciones. Si se modifica una suscripción existente utilizando la opción MQSO_ALTER, estas opciones comodín no se pueden cambiar. Al regresar de una llamada MQSUB utilizando MQSO_RESUME, se establece la opción comodín adecuada.

MQSO_CARACTERE_COMODIN

Los comodines solo funcionan con los caracteres que se encuentran dentro de la cadena del tema.

El comportamiento definido por MQSO_WILDCARD_CHAR se muestra en la siguiente tabla.

Tabla 557. Cómo se interpretan los comodines	
Carácter especial	Comportamiento
Barra inclinada (/)	Sin importancia, solo otro personaje
Asterisco (*)	Comodín, cero o más caracteres
Signo de interrogación (?)	Comodín, 1 carácter
Signo de porcentaje (%)	Caracteres de escape para permitir el uso de los caracteres (*), (?) o (%) en una cadena sin que se interpreten como caracteres especiales, por ejemplo, (%*), (%?) o (%%).

Por ejemplo, publicar sobre el siguiente tema:

```
/level0/level1/level2/level3/level4
```

empareja a los suscriptores utilizando los siguientes temas:

```
*  
/*  
/ level0/level1/level2/level3/*  
/ level0/level1/*/level3/level4  
/ level0/level1/le?el2/level3/level4
```

Nota: Este uso de comodines proporciona exactamente el significado que se indica en IBM MQ V6 y WebSphere MB V6 cuando se utilizan mensajes con formato MQRFH1 para publicar/suscribirse. Se recomienda no utilizarlo en aplicaciones de nueva creación, sino solo en aquellas que ya se ejecutaban con esa versión y que no se han modificado para utilizar el comportamiento predeterminado de comodines, tal y como se describe en MQSO_WILDCARD_TOPIC.

MQSO_WILDCARD_TEMA

Los comodines solo funcionan en elementos temáticos dentro de la cadena temática. Este es el comportamiento predeterminado si no se elige ninguno.

El comportamiento requerido por MQSO_WILDCARD_TOPIC se muestra en la siguiente tabla:

Tabla 558. Cómo se interpretan los comodines	
Carácter especial	Comportamiento
(/)	Separador de nivel de tema
signo de almohadilla (#)	Comodín: nivel de tema múltiple
Signo más (+)	Comodín: nivel de tema único

Notas:

Los signos (+) y (#) no se tratan como comodines si se mezclan con otros caracteres (incluidos ellos mismos) dentro de un nivel temático. En la siguiente cadena, los caracteres (#) y (+) se tratan como caracteres normales.

```
level0/level1/#+/level3/level#
```

Por ejemplo, publicar sobre el siguiente tema:

```
/level0/level1/level2/level3/level4
```

empareja a los suscriptores utilizando los siguientes temas:

```
#
/#
/ level0/level1/level2/level3/#
/ level0/level1/+level3/level4
```

Otras opciones

Las siguientes opciones controlan la forma en que se emite la llamada a la API, en lugar de la suscripción. Al regresar de una llamada MQSUB utilizando MQSO_RESUME, estas opciones no cambian. Consulte [“AlternateUserId \(MQCHAR12\)”](#) en la página 573 para obtener más detalles.

MQSO_ALTERNATE_USER_AUTHORITY

El campo AlternateUserId contiene un identificador de usuario para utilizarlo para validar esta llamada MQSUB. La llamada solo puede tener éxito si este AlternateUserId está autorizado para abrir el objeto con las opciones de acceso especificadas, independientemente de si el identificador de usuario bajo el que se ejecuta la aplicación está autorizado para hacerlo.

MQSO_SET_CORREL_ID

La suscripción consiste en utilizar el identificador de correlación proporcionado en el *SubCorrelId* campo. Si no se especifica esta opción, el gestor de colas crea automáticamente un identificador de correlación en el momento de la suscripción y lo devuelve a la aplicación en el *SubCorrelId* campo. Para obtener más información, consulte [SubCorrelId](#) para obtener más información.

Esta opción no se puede combinar con MQSO_MANAGED.

MQSO_SET_IDENTITY_CONTEXT

La suscripción consiste en utilizar el token de contabilidad y los datos de identidad de la aplicación proporcionados en los campos *PubAccountingToken* *PubApplIdentityData* y.

Si se especifica esta opción, se realiza la misma comprobación de autorización que si se accediera a la cola de destino mediante una llamada MQOPEN con MQOO_SET_IDENTITY_CONTEXT, excepto en el caso en que se utilice también la opción MQSO_MANAGED, en cuyo caso no hay comprobación de autorización en la cola de destino.

Si no se especifica esta opción, las publicaciones enviadas a este suscriptor tienen información de contexto predeterminada asociada a ellos, de la manera siguiente:

Tabla 559. Información contextual predeterminada para las publicaciones enviadas a este suscriptor	
Campo de MQMD	Valor utilizado
<i>UserIdentifier</i>	El ID de usuario asociado a la suscripción en el momento en que se realizó la suscripción.
<i>AccountingToken</i>	Determinado a partir del entorno si es posible; establecido en MQACT_NONE si no lo es.
<i>ApplIdentityData</i>	Establecer como en blanco

Esta opción sólo es válida con MQSO_CREATE y MQSO_ALTER. Si se utiliza con MQSO_RESUME, los campos *PubAccountingToken* *PubApplIdentityData* y se ignoran, por lo que esta opción no tiene ningún efecto.

Si se modifica una suscripción sin utilizar esta opción, cuando anteriormente la suscripción proporcionaba información de contexto de identidad, se genera información de contexto predeterminada para la suscripción modificada.

Si una suscripción que permite que distintos ID de usuario la utilicen con la opción MQSO_ANY_USERID, se reanuda con un ID de usuario diferente, se genera contexto de identidad predeterminado para el nuevo ID de usuario que es propietario ahora de la suscripción y las publicaciones posteriores se entregan conteniendo el nuevo contexto de identidad.

MQSO_FAIL_IF QUIESCING

La llamada MQSUB falla si el gestor de colas se encuentra en estado de inactividad. En z/OS, para una aplicación CICS IMS o, esta opción también fuerza el fallo de la llamada MQSUB si la conexión se encuentra en estado de inactividad.

ObjectName (MQCHAR48)

Este es el nombre del objeto de tema tal y como se define en el gestor de colas local.

El nombre puede contener los siguientes caracteres:

- Caracteres alfabéticos mayúsculos (de la A a la Z)
- Letras minúsculas (de la a a la z)
- Dígitos numéricos (del 0 al 9)
- Punto (.), barra inclinada (/), guión bajo (_), porcentaje (%)

El nombre no debe contener espacios en blanco al principio ni en el interior, pero puede contener espacios en blanco al final. Utilice un carácter nulo para indicar el final de los datos significativos en el nombre; el carácter nulo y cualquier carácter que le siga se tratan como espacios en blanco. Las siguientes restricciones se aplican en los entornos indicados:

- En los sistemas que utilizan EBCDIC Katakana, no se pueden utilizar caracteres en minúscula.
- En z/OS:
 - Evite los nombres que comiencen o terminen con un guión bajo, ya que no pueden ser procesados por los paneles de operaciones y control.
 - El carácter por ciento tiene un significado especial para RACF. Si RACF se utiliza como gestor de seguridad externo, los nombres no deben contener el símbolo %. Si lo hacen, esos nombres no se incluyen en ninguna comprobación de seguridad cuando se RACF utilizan perfiles genéricos.
- En IBM i, los nombres que contengan caracteres en minúscula, barra inclinada o porcentaje deben escribirse entre comillas cuando se especifiquen en comandos. Estas comillas no deben especificarse para nombres que aparecen como campos en estructuras o como parámetros en llamadas.

El *ObjectName* se utiliza para formar el nombre completo del tema.

El nombre completo del tema se puede construir a partir de dos campos diferentes: *ObjectName* y *ObjectString*. Para obtener más información sobre cómo se utilizan estos dos campos, consulte [Combinación de cadenas de temas](#).

Si no se encuentra el objeto identificado por el *ObjectName* campo, la llamada falla con el código de motivo MQRC_UNKNOWN_OBJECT_NAME, incluso si hay una cadena especificada en *ObjectString*.

Al regresar de una llamada MQSUB utilizando la opción MQSO_RESUME, este campo no se modifica.

La longitud de este campo viene dada por MQ_TOPIC_NAME_LENGTH. El valor inicial de este campo es la cadena nula en C y 48 caracteres en blanco en otros lenguajes de programación.

Si se modifica una suscripción existente utilizando la opción MQSO_ALTER, no se puede cambiar el nombre del objeto de tema suscrito. Este campo y el *ObjectString* campo pueden omitirse. Si se proporcionan, deben resolver al mismo nombre completo del tema. Si no lo hacen, la llamada falla con MQRC_TOPIC_NOT_ALTERABLE.

AlternateUserId (MQCHAR12)

Si se especifica MQSO_ALTERNATE_USER_AUTHORITY, este campo contiene un identificador de usuario alternativo que se utiliza para comprobar la autorización de la suscripción y para la salida a la cola de destino (especificada en el **Hobj** parámetro de la llamada MQSUB), en lugar del identificador de usuario con el que se está ejecutando actualmente la aplicación.

Si tiene éxito, el identificador de usuario especificado en este campo se registra como el identificador de usuario propietario de la suscripción en lugar del identificador de usuario con el que se está ejecutando actualmente la aplicación.

Si se especifica MQSO_ALTERNATE_USER_AUTHORITY y este campo está completamente en blanco hasta el primer carácter nulo o el final del campo, la suscripción solo podrá realizarse correctamente si no se necesita autorización de usuario para suscribirse a este tema con las opciones especificadas o la cola de destino para la salida.

Si no se especifica MQSO_ALTERNATE_USER_AUTHORITY, este campo se ignora.

En los entornos indicados existen las siguientes diferencias:

- En z/OS, solo se utilizan los primeros 8 caracteres de AlternateUserId para comprobar la autorización de la suscripción. Sin embargo, el identificador de usuario actual debe estar autorizado para especificar este identificador de usuario alternativo concreto; para esta comprobación se utilizan los 12 caracteres del identificador de usuario alternativo. El identificador de usuario solo debe contener caracteres permitidos por el administrador de seguridad externo.

Al regresar de una llamada MQSUB utilizando MQSO_RESUME, este campo permanece sin cambios.

Este es un campo de entrada. La longitud de este campo viene determinada por MQ_USER_ID_LENGTH. El valor inicial de este campo es la cadena nula en C y 12 caracteres en blanco en otros lenguajes de programación.

AlternateSecurityId (MQBYTE40)

Este es un identificador de seguridad que se transfiere con el AlternateUserId al servicio de autorizaciones para permitir que se realicen las comprobaciones de autorización correspondientes.

AlternateSecurityId sólo se utiliza si se especifica MQSO_ALTERNATE_USER_AUTHORITY y el campo AlternateUserId no está completamente en blanco hasta el primer carácter nulo o el final del campo.

Al regresar de una llamada MQSUB utilizando MQSO_RESUME, este campo permanece sin cambios.

Consulte la descripción de [“AlternateSecurityId \(MQBYTE40\)”](#) en la [página 467](#) en el tipo de datos MQOD para obtener más información.

SubExpiry (MQLONG)

Es el tiempo expresado en décimas de segundo tras el cual caduca la suscripción. Una vez transcurrido este intervalo, no habrá más publicaciones que coincidan con esta suscripción. Tan pronto como expira una suscripción, las publicaciones dejan de enviarse a la cola. Sin embargo, las publicaciones que ya están allí no se ven afectadas de ninguna manera. *SubExpiry* no tiene ningún efecto sobre la expiración de la publicación.

Se reconoce el siguiente valor especial:

MQEI_UNLIMITED

La suscripción tiene una duración ilimitada.

Si se modifica una suscripción existente utilizando la opción MQSO_ALTER, se puede cambiar la fecha de vencimiento de la suscripción.

Al regresar de una llamada MQSUB utilizando la opción MQSO_RESUME, este campo se establece en la fecha de vencimiento original de la suscripción y no en el tiempo de vencimiento restante.

ObjectString (MQCHARV)

Este es el nombre largo del objeto que se va a utilizar.

El *ObjectString* se utiliza para formar el nombre completo del tema.

El nombre completo del tema se puede construir a partir de dos campos diferentes: *ObjectName* y *ObjectString*. Para obtener más información sobre cómo se utilizan estos dos campos, consulte [Combinación de cadenas de temas](#).

La longitud máxima de *ObjectString* es 10240.

Si no *ObjectString* se especifica correctamente, según la descripción de cómo utilizar la estructura [MQCHARV](#), o si supera la longitud máxima, la llamada falla con el código de motivo MQRC_OBJECT_STRING_ERROR.

Este es un campo de entrada. Los valores iniciales de los campos de esta estructura son los mismos que los de la estructura [MQCHARV](#).

Si hay comodines en la *ObjectString* interpretación de esos comodines se puede controlar utilizando las opciones de comodines especificadas en el campo Opciones del MQSD.

Al regresar de una llamada MQSUB utilizando la opción MQSO_RESUME, este campo no se modifica. El nombre completo del tema utilizado se devuelve en el *ResObjectString* campo si se proporciona un búfer.

Si se modifica una suscripción existente utilizando la opción MQSO_ALTER, no se puede cambiar el nombre largo del objeto de tema suscrito. Este campo y el *ObjectName* campo pueden omitirse. Si se proporcionan, deben resolver al mismo nombre completo del tema o la llamada fallará con MQRC_TOPIC_NOT_ALTERABLE.

SubName (MQCHARV)

Esto especifica el nombre de la suscripción. Este campo solo es obligatorio si *Options* especifica la opción MQSO_DURABLE, pero si se proporciona, el gestor de colas también lo utilizará para MQSO_NON_DURABLE.

Si se especifica, *SubName* debe ser único dentro del gestor de colas, ya que es el método utilizado para identificar la suscripción.

La longitud máxima de *SubName* es 10240.

Este campo tiene dos funciones. Para una suscripción MQSO_DURABLE, utilice este campo para identificar una suscripción, de modo que pueda reanudarla después de haberla creado si ha cerrado el identificador de la suscripción (utilizando la opción MQCO_KEEP_SUB) o se ha desconectado del gestor de colas. Esto se hace utilizando la llamada MQSUB con la opción MQSO_RESUME. También se muestra en la vista administrativa de suscripciones en el campo SUBID en DISPLAY SBSTATUS.

Si *SubName* se especifica incorrectamente, según la descripción de cómo utilizar la estructura [MQCHARV](#), se omite cuando es necesario (es decir, *SubName.VSLength* es cero), o si supera la longitud máxima, la llamada falla con el código de motivo MQRC_SUB_NAME_ERROR.

Este es un campo de entrada. Los valores iniciales de los campos de esta estructura son los mismos que los de la estructura [MQCHARV](#).

Si se modifica una suscripción existente utilizando la opción MQSO_ALTER, no se puede cambiar el nombre de la suscripción, ya que es el campo identificador que se utiliza para encontrar la suscripción a la que se hace referencia. No se modifica en la salida de una llamada MQSUB con la opción MQSO_RESUME.

SubUserData (MQCHARV)

Esto especifica los datos de usuario de la suscripción. Los datos proporcionados en la suscripción en este campo se incluirán como la propiedad de mensaje MQSubUserData de cada publicación que se envíe a esta suscripción.

La longitud máxima de *SubUserData* es 10240.

Si *SubUserData* se especifica incorrectamente, según la descripción de cómo utilizar la estructura [MQCHARV](#), o si supera la longitud máxima, la llamada falla con el código de motivo MQRC_SUB_USER_DATA_ERROR.

Este es un campo de entrada. Los valores iniciales de los campos de esta estructura son los mismos que los de la estructura MQCHARV.

Si se modifica una suscripción existente utilizando la opción MQSO_ALTER, se pueden cambiar los datos de usuario de la suscripción.

Este campo de longitud variable se devuelve en la salida de una llamada MQSUB utilizando la opción MQSO_RESUME, si se proporciona un búfer y hay una longitud de búfer positiva en *VSBuflen*. Si no se proporciona ningún búfer en la llamada, solo se devuelve la longitud de la fecha de suscripción del usuario en el *VSLength* campo del MQCHARV. Si el búfer proporcionado es más pequeño que el espacio necesario para devolver el campo, solo se *VSBuflen* devuelven bytes en el búfer proporcionado.

SubCorrelId (MQBYTE24)

Este campo contiene un identificador de correlación común a todas las publicaciones que coinciden con esta suscripción.



Atención: un identificador de correlación sólo se puede pasar entre gestores de colas en un clúster de publicación/suscripción, no en una jerarquía.

Todas las publicaciones enviadas para coincidir con esta suscripción contienen este identificador de correlación en el descriptor del mensaje. Si varias suscripciones obtienen sus publicaciones de la misma cola, el uso de MQGET por identificador de correlación permite obtener solo las publicaciones de una suscripción específica. Este identificador de correlación puede ser generado por el gestor de colas o por el usuario.

Si no se especifica la opción MQSO_SET_CORREL_ID, el identificador de correlación lo genera el gestor de colas y este campo es un campo de salida que contiene el identificador de correlación que se establecerá en cada mensaje publicado para esta suscripción. El identificador de correlación generado consta de un identificador de producto de 4 bytes (AMQX o CSQM en ASCII o EBCDIC) seguido de una implementación específica del producto de una cadena única.

Si se especifica la opción MQSO_SET_CORREL_ID, el identificador de correlación lo genera el usuario y este campo es un campo de entrada que contiene el identificador de correlación que se establecerá en cada publicación para esta suscripción. En este caso, si el campo contiene MQCI_NONE, el identificador de correlación que se establece en cada mensaje publicado para esta suscripción es el identificador de correlación creado por la inserción original del mensaje.

Si la opción MQSO_GROUP_SUB se ha especificado y el identificador de correlación especificado es el mismo que una suscripción agrupada existente que utiliza la misma cola y una serie de tema que se solapa, sólo la suscripción más significativa del grupo se proporciona con una copia de la publicación.

La longitud de este campo viene dada por MQ_CORREL_ID_LENGTH. El valor inicial de este campo es MQCI_NONE.

Si está modificando una suscripción existente mediante la opción MQSO_ALTER y este campo es un campo de entrada, entonces se puede cambiar el identificador de correlación de la suscripción, a menos que la suscripción sea una suscripción agrupada, es decir, que se haya creado mediante la opción MQSO_GROUP_SUB, en cuyo caso no se puede cambiar el identificador de correlación de la suscripción.

Al regresar de una llamada MQSUB utilizando MQSO_RESUME, este campo se establece en el identificador de correlación actual para la suscripción.

PubPriority (MQLONG)

Este es el valor que aparecerá en el *Priority* campo del descriptor de mensajes (MQMD) de todos los mensajes de publicación que coincidan con esta suscripción. Para obtener más información sobre el *Priority* campo en el MQMD, consulte [“Priority \(MQLONG\)” en la página 422](#).

El valor debe ser mayor o igual que cero, donde cero es la prioridad más baja. También se pueden utilizar los valores especiales siguientes:

MQPRI_PRIORITY_AS_Q_DEF

Cuando se proporciona una cola de suscripción en el *Hobj* campo de la llamada MQSUB y no es un identificador gestionado, la prioridad del mensaje se toma del **DefPriority** atributo de esta cola. Si la cola es una cola de clúster o hay más de una definición en la ruta de resolución del nombre de la cola, la prioridad se determina cuando el mensaje de publicación se coloca en la cola, tal y como se describe para “[Priority \(MQLONG\)](#)” en la página 422.

Si la llamada MQSUB utiliza un identificador gestionado, la prioridad del mensaje se toma del **DefPriority** atributo de la cola modelo asociada al tema suscrito.

MQPRI_PRIORIDAD_COMO_PUBLICADO

La prioridad del mensaje es la prioridad de la publicación original. Este es el valor inicial del campo.

Si se modifica una suscripción existente utilizando la opción MQSO_ALTER, se puede cambiar la *Priority* de cualquier mensaje de publicación futuro.

Al regresar de una llamada MQSUB utilizando MQSO_RESUME, este campo se establece en la prioridad actual que se utiliza para la suscripción.

PubAccountingToken (MQBYTE32)

Este es el valor que aparecerá en el *AccountingToken* campo del descriptor de mensajes (MQMD) de todos los mensajes de publicación que coincidan con esta suscripción. *AccountingToken* forma parte del contexto de identidad del mensaje. Para obtener más información sobre el contexto del mensaje, consulte [Contexto de mensaje](#). Para obtener más información sobre el *AccountingToken* campo en el MQMD, consulte “[AccountingToken \(MQBYTE32\)](#)” en la página 430

Puede utilizar el siguiente valor especial para el *PubAccountingToken* campo:

MQACT_NONE

No se especifica ningún token contable.

El valor es un cero binario para la longitud del campo.

Para el lenguaje de programación C, también se define la constante MQACT_NONE_ARRAY, que tiene el mismo valor que MQACT_NONE, pero es una matriz de caracteres en lugar de una cadena.

Si no se especifica la opción MQSO_SET_IDENTITY_CONTEXT, el token de contabilidad es generado por el gestor de colas como información de contexto predeterminada y este campo es un campo de salida que contiene el *AccountingToken* que se establecerá en cada mensaje publicado para esta suscripción.

Si se especifica la opción MQSO_SET_IDENTITY_CONTEXT, el token de contabilidad lo genera el usuario y este campo es un campo de entrada que contiene el *AccountingToken* que se debe establecer en cada publicación para esta suscripción.

La longitud de este campo viene dada por MQ_ACCOUNTING_TOKEN_LENGTH. El valor inicial de este campo es MQACT_NONE.

Si se modifica una suscripción existente utilizando la opción MQSO_ALTER, se puede cambiar el valor de *AccountingToken* en cualquier mensaje de publicación futuro.

Al regresar de una llamada MQSUB utilizando MQSO_RESUME, este campo se establece en el actual *AccountingToken* que se utiliza para la suscripción.

PubApplIdentityData (MQCHAR32)

Este es el valor que se encuentra en el *ApplIdentityData* campo del descriptor de mensajes (MQMD) de todos los mensajes de publicación que coinciden con esta suscripción. *ApplIdentityData* forma parte del contexto de identidad del mensaje. Para obtener más información sobre el contexto del mensaje, consulte [Contexto de mensaje](#). Para obtener más información sobre el *ApplIdentityData* campo en el MQMD, consulte “[ApplIdentityData \(MQCHAR32\)](#)” en la página 432

Si no se especifica la opción MQSO_SET_IDENTITY_CONTEXT, el *ApplIdentityData* que se establece en cada mensaje publicado para esta suscripción es un espacio en blanco, como información de contexto predeterminada.

Si se especifica la opción MQSO_SET_IDENTITY_CONTEXT, el *PubApplIdentityData* lo genera el usuario y este campo es un campo de entrada que contiene el *ApplIdentityData* que se establecerá en cada publicación para esta suscripción.

La longitud de este campo viene dada por MQ_APPL_IDENTITY_DATA_LENGTH. El valor inicial de este campo es la cadena nula en C y 32 caracteres en blanco en otros lenguajes de programación.

Si se modifica una suscripción existente utilizando la opción MQSO_ALTER, se puede cambiar la *ApplIdentityData* de cualquier mensaje de publicación futuro.

Al regresar de una llamada MQSUB utilizando MQSO_RESUME, este campo se establece en el actual *ApplIdentityData* que se utiliza para la suscripción.

SelectionString (MQCHARV)

Esta es la cadena utilizada para proporcionar los criterios de selección que se utilizan al suscribirse a los mensajes de un tema.

Este campo de longitud variable se devolverá en la salida de una llamada MQSUB utilizando la opción MQSO_RESUME, si se proporciona un búfer y también hay una longitud de búfer positiva en VSBufSize. Si no se proporciona ningún búfer en la llamada, solo se devolverá la longitud de la cadena de selección en el campo VSLength de MQCHARV. Si el búfer proporcionado es inferior al espacio necesario para devolver el campo, solo se devuelven VSBufSize bytes en el búfer.

Si *SelectionString* se especifica incorrectamente, según la descripción de cómo utilizar la “MQCHARV - Cadena de longitud variable” en la página 243 estructura, o si supera la longitud máxima, la llamada falla con el código de motivo MQRC_SELECTION_STRING_ERROR.

SelectionString El uso se describe en [Selectores](#).

SubLevel (MQLONG)

Éste es el nivel asociado a la suscripción. Las publicaciones solo se envían a esta suscripción si se encuentra entre las suscripciones con el valor más alto de SubLevel menor o igual al valor de PubLevel utilizado en el momento de la publicación. Sin embargo, si una publicación se ha conservado, ya no está disponible para los suscriptores de niveles superiores, ya que se vuelve a publicar en PubLevel 1.

El valor debe estar comprendido entre cero y 9. Cero sería el nivel más bajo.

El valor inicial de este campo es 1.

Para obtener más información, consulte [Interceptación de publicaciones](#).

Si se modifica una suscripción existente utilizando la opción MQSO_ALTER, no se podrá cambiar el campo « SubLevel ».

No se permite combinar un valor de « SubLevel » superior a 1 con la opción MQSO_PUBLICATIONS_ON_REQUEST.

Al regresar de una llamada MQSUB utilizando MQSO_RESUME, este campo se establece en el nivel actual que se utiliza para la suscripción.

ResObjectString (MQCHARV)

Este es el nombre largo del objeto después de que el gestor de colas resuelve el nombre proporcionado en *ObjectName*.

Si se proporciona el nombre largo del objeto en *ObjectString* y no se proporciona nada en *ObjectName*, entonces el valor devuelto en este campo es el mismo que el proporcionado en *ObjectString*.

Si se omite este campo (es decir, si `ResObjectString.VSBufSize` es cero), entonces no se devuelve *ResObjectString* el, pero se devuelve la longitud en `ResObjectString.VSLength`. Si la longitud es inferior a la longitud total `ResObjectString`, se trunca y devuelve tantos caracteres de la derecha como quepan en la longitud proporcionada.

Si *ResObjectString* se especifica incorrectamente, según la descripción de cómo utilizar la estructura `MQCHARV`, o si supera la longitud máxima, la llamada falla con el código de motivo `MQRC_RES_OBJECT_STRING_ERROR`.

MQSMPO - Establecer opciones de propiedades de mensajes

La **MQSMPO** estructura permite a las aplicaciones especificar opciones que controlan cómo se establecen las propiedades de los mensajes. La estructura es un parámetro de entrada en la **MQSETMP** llamada.

Disponibilidad

Todos IBM MQ los sistemas y IBM MQ clientes.

Conjunto de caracteres y codificación

Los datos en **MQSMPO** deben estar en el juego de caracteres de la aplicación y la codificación de la aplicación (**MQENC_NATIVE**).

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 560. Campos en MQSMPO		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>StrucId</u> (identificador de estructura)	MQSMPO_STRUC_ID	'SMPO'
<u>Versión</u> (número de versión de la estructura)	MQSMPO_VERSION_1	1
<u>Opciones</u> (opciones)	MQSMPO_NINGUNO	0
<u>ValueEncoding</u> (codificación del valor de la propiedad)	MQENC_NATIVE	Depende del entorno
<u>ValueCCSID</u> (conjunto de caracteres del valor de la propiedad)	MQCCSI_APPL	-3
Notas: 1. El valor «Cadena nula o espacios en blanco» denota la cadena nula en C y los caracteres en blanco en otros lenguajes de programación. 2. En el lenguaje de programación C, la variable macro <code>MQSMPO_DEFAULT</code> contiene los valores que se enumeran en la tabla. Se puede utilizar de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQSMPO MySMPO = {MQSMPO_DEFAULT};</pre>		

Declaraciones lingüísticas

Declaración C para MQSMPO

```
typedef struct tagMQSMPO MQSMPO;
struct tagMQSMPO {
    MQCHAR4    StrucId;          /* Structure identifier */
    MQLONG     Version;         /* Structure version number */
};
```



```

MQLONG Options;          /* Options that control the action of MQSETMP */
MQLONG ValueEncoding;    /* Encoding of Value */
MQLONG ValueCCSID;       /* Character set identifier of Value */
};

```

COBOL declaración para MQSMPO

```

** MQSMPO structure
10 MQSMPO.
** Structure identifier
15 MQSMPO-STRUCID PIC X(4).
** Structure version number
15 MQSMPO-VERSION PIC S9(9) BINARY.
** Options that control the action of MQSETMP
15 MQSMPO-OPTIONS PIC S9(9) BINARY.
** Encoding of VALUE
15 MQSMPO-VALUEENCODING PIC S9(9) BINARY.
** Character set identifier of VALUE
15 MQSMPO-VALUECCSID PIC S9(9) BINARY.

```

PL/I declaración para MQSMPO

```

dcl
1 MQSMPO based,
3 StrucId      char(4),          /* Structure identifier */
3 Version      fixed bin(31),    /* Structure version number */
3 Options      fixed bin(31),    /* Options that control the action of MQSETMP */
3 ValueEncoding fixed bin(31),    /* Encoding of Value */
3 ValueCCSID    fixed bin(31),    /* Character set identifier of Value */

```

High Level Assembler declaración para MQSMPO

```

MQSMPO          DSECT
MQSMPO_STRUCID  DS CL4 Structure identifier
MQSMPO_VERSION  DS F   Structure version number
MQSMPO_OPTIONS  DS F   Options that control the action of
*               MQSETMP
MQSMPO_VALUEENCODING DS F Encoding of VALUE
MQSMPO_VALUECCSID DS F Character set identifier of VALUE
MQSMPO_LENGTH   EQU *-MQSMPO
MQSMPO_AREA     DS CL(MQSMPO_LENGTH)

```

Detalles de campo para MQSMPO

La **MQSMPO** estructura Opciones de configuración de mensajes permite a las aplicaciones especificar opciones que controlan cómo se configuran las propiedades de los mensajes. La estructura es un parámetro de entrada en la **MQSETMP** llamada.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 561. Campos en MQSMPO		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
StrucId (identificador de estructura)	MQSMPO_STRUC_ID	' SMP0 '
Versión (número de versión de la estructura)	MQSMPO_VERSION_1	1
Opciones (opciones)	MQSMPO_NINGUNO	0
ValueEncoding (codificación del valor de la propiedad)	MQENC_NATIVE	Depende del entorno
ValueCCSID (conjunto de caracteres del valor de la propiedad)	MQCCSI_APPL	-3

Tabla 561. Campos en MQSMPO (continuación)

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
Notas: 1. El valor «Cadena nula o espacios en blanco» denota la cadena nula en C y los caracteres en blanco en otros lenguajes de programación. 2. En el lenguaje de programación C, la variable macroMQSMPO_DEFAULT contiene los valores que se enumeran en la tabla. Se puede utilizar de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQSMPO MySMPO = {MQSMPO_DEFAULT};</pre>		

StrucId (MQCHAR4)

Este es el identificador de estructura de la estructura de opciones de propiedad del mensaje establecido. Siempre es un campo de entrada. Su valor es MQSMPO_STRUC_ID.

El valor debe ser:

MQSMPO_STRUC_ID

Identificador de la estructura de opciones de la propiedad del mensaje establecido.

Para el lenguaje de programación C, también se **MQSMPO_STRUC_ID_ARRAY** define la constante. Esto tiene el mismo valor que **MQSMPO_STRUC_ID**, pero es una matriz de caracteres en lugar de una cadena.

Versión (MQLONG)

Este es el número de versión de la estructura; el valor debe ser:

MQSMPO_VERSION_1

Version-1 Establecer la estructura de opciones de propiedades del mensaje.

La siguiente constante especifica el número de versión de la versión actual:

MQSMPO_VERSIÓN_ACTUAL

Estructura actual de las opciones de configuración de la propiedad del mensaje.

Este campo siempre es un campo de entrada. El valor inicial de este campo es **MQSMPO_VERSION_1**.

Opciones (MQLONG)

Opciones de ubicación

Las siguientes opciones se refieren a la ubicación relativa de la propiedad en comparación con el cursor de la propiedad:

MQSMPO_SET_FIRST

Establece el valor de la primera propiedad que coincide con el nombre especificado o, si no existe, añade una nueva propiedad después de todas las demás propiedades con una jerarquía coincidente.

MQSMPO_ESTABLECER_PROPIEDAD_BAJO_CURSOR

Establece el valor de la propiedad señalada por el cursor de propiedad. La propiedad señalada por el cursor de propiedad es la última que se consultó utilizando la opción MQIMPO_INQ_FIRST o MQIMPO_INQ_NEXT.

El cursor de propiedad se restablece cuando se reutiliza el identificador de mensaje en una llamada MQGET, o cuando se especifica el identificador de mensaje en el *MsgHandle* campo de la estructura MQGMO o MQPMO en una llamada MQPUT.

Si se utiliza esta opción cuando aún no se ha establecido el cursor de propiedad o si se ha eliminado el puntero de propiedad al cursor de propiedad, la llamada falla con el código de finalización MQCC_FAILED y el código de motivo MQRC_PROPERTY_NOT_AVAILABLE.

MQSMPO_SET_PROP_BEFORE_CURSOR

Establece una nueva propiedad antes de la propiedad señalada por el cursor de propiedad. La propiedad señalada por el cursor de propiedad es la última que se consultó utilizando la opción MQIMPO_INQ_FIRST o MQIMPO_INQ_NEXT.

El cursor de propiedad se restablece cuando se reutiliza el identificador de mensaje en una llamada MQGET, o cuando se especifica el identificador de mensaje en el *MsgHandle* campo de la estructura MQGMO o MQPMO en una llamada MQPUT.

Si se utiliza esta opción cuando aún no se ha establecido el cursor de propiedad o si se ha eliminado el puntero de propiedad al cursor de propiedad, la llamada falla con el código de finalización MQCC_FAILED y el código de motivo MQRC_PROPERTY_NOT_AVAILABLE.

MQSMPO_SET_PROP_DESPUÉS_DEL_CURSOR

Establece una nueva propiedad después de la propiedad señalada por el cursor de propiedad. La propiedad señalada por el cursor de propiedad es la última que se consultó utilizando la opción MQIMPO_INQ_FIRST o MQIMPO_INQ_NEXT.

El cursor de propiedad se restablece cuando se reutiliza el identificador de mensaje en una llamada MQGET, o cuando se especifica el identificador de mensaje en el *MsgHandle* campo de la estructura MQGMO o MQPMO en una llamada MQPUT.

Si se utiliza esta opción cuando aún no se ha establecido el cursor de propiedad o si se ha eliminado el puntero de propiedad al cursor de propiedad, la llamada falla con el código de finalización MQCC_FAILED y el código de motivo MQRC_PROPERTY_NOT_AVAILABLE.

MQSMPO_AÑADIR_PROPIEDAD

Hace que se añada una nueva propiedad después de todas las demás propiedades con una jerarquía coincidente. Si existe al menos una propiedad que coincida con el nombre especificado, se añade una nueva propiedad al final, después de la lista de propiedades.

Esta opción permite crear una lista de propiedades con el mismo nombre.

Si no necesita ninguna de las opciones descritas, utilice la siguiente opción:

MQSMPO_NINGUNO

No se ha especificado ninguna opción.

Este campo siempre es un campo de entrada. El valor inicial de este campo es MQSMPO_SET_FIRST.

ValueEncoding (MQLONG)

La codificación del valor de la propiedad que se establecerá si el valor es numérico.

Este campo siempre es un campo de entrada. El valor inicial de este campo es MQENC_NATIVE.

ValueCCSID (MQLONG)

El conjunto de caracteres del valor de la propiedad que se va a establecer si el valor es una cadena de caracteres.

Este campo siempre es un campo de entrada. El valor inicial de este campo es MQCCSI_APPL.

MQSRO - Opciones de solicitud de suscripción

La estructura MQSRO permite a la aplicación especificar opciones que controlan cómo se realiza una solicitud de suscripción. La estructura es un parámetro de entrada/salida en la llamada MQSUBRQ.

Disponibilidad

La estructura MQSRO está disponible en las siguientes plataformas:

-  AIX
-  IBM i
-  Linux
-  Windows
-  z/OS

y para IBM MQ MQI clients conectados a estos sistemas.

Versión

La versión actual de MQSRO es MQSRO_VERSION_1.

Conjunto de caracteres y codificación

Los datos en MQSRO deben estar en el juego de caracteres proporcionado por el atributo del **CodedCharSetId** gestor de colas y la codificación del gestor de colas local proporcionada por MQENC_NATIVE. Sin embargo, si la aplicación se ejecuta como un cliente MQ MQI, la estructura debe estar en el conjunto de caracteres y la codificación del cliente.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>StrucId</u> (identificador de estructura)	MQSRO_STRUC_ID	'SRO~'
<u>Versión</u> (número de versión de la estructura)	MQSRO_VERSION_1	1
<u>Opciones</u> (opciones)	MQSRO_NINGUNO	0
<u>NumPubs</u> (número de publicaciones)	Ninguna	0
Notas: 1. El símbolo ~ representa un único carácter en blanco. 2. En el lenguaje de programación C, la variable macroMQSRO_DEFAULT contiene los valores que se enumeran en la tabla. Se puede utilizar de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQSRO MySRO = {MQSRO_DEFAULT};</pre>		

Declaraciones lingüísticas

Declaración C para MQSRO

```
typedef struct tagMQSRO MQSRO;
struct tagMQSRO {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQLONG     Options;          /* Options that control the action of MQSUBRQ */
    MQLONG     NumPubs;          /* Number of publications sent */
    /* Ver:1 */
};
```

COBOL declaración para MQSRO

```
** MQSRO structure
10 MQSRO.
** Structure identifier
15 MQSRO-STRUCID          PIC X(4).
** Structure version number
15 MQSRO-VERSION          PIC S9(9) BINARY.
** Options that control the action of MQSUBRQ
15 MQSRO-OPTIONS          PIC S9(9) BINARY.
** Number of publications sent
15 MQSRO-NUMPUBS          PIC S9(9) BINARY.
```

PL/I declaración para MQSRO

```
dcl
1 MQSRO based,
3 StrucId          char(4),          /* Structure identifier */
3 Version          fixed bin(31), /* Structure version number */
3 Options          fixed bin(31), /* Options that control the action of MQSUBRQ */
3 NumPubs          fixed bin(31); /* Number of publications sent */
```

High Level Assembler declaración para MQSRO

```
MQSRO          DSECT
MQSRO_STRUCID  DS    CL4    Structure identifier
MQSRO_VERSION  DS    F      Structure version number
MQSRO_OPTIONS  DS    F      Options that control the action of MQSUBRQ
MQSRO_NUMPUBS  DS    F      Number of publications sent
*
MQSRO_LENGTH   EQU    *-MQSRO
ORG            MQSRO
MQSRO_AREA     DS      CL(MQSRO_LENGTH)
```

Detalles de campo para MQSRO

La estructura de opciones de solicitud de suscripción MQSRO permite a la aplicación especificar opciones que controlan cómo se realiza una solicitud de suscripción. La estructura es un parámetro de entrada/salida en la llamada MQSUBRQ.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
StrucId (identificador de estructura)	MQSRO_STRUC_ID	'SRO~'
Versión (número de versión de la estructura)	MQSRO_VERSION_1	1
Opciones (opciones)	MQSRO_NINGUNO	0
NumPubs (número de publicaciones)	Ninguna	0
Notas: 1. El símbolo ~ representa un único carácter en blanco. 2. En el lenguaje de programación C, la variable macroMQSRO_DEFAULT contiene los valores que se enumeran en la tabla. Se puede utilizar de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQSRO MySRO = {MQSRO_DEFAULT};</pre>		

StrucId (MQCHAR4)

Este es el identificador de estructura de la estructura de opciones de solicitud de suscripción. Siempre es un campo de entrada. Su valor es MQSRO_STRUC_ID.

El valor debe ser:

MQSRO_STRUC_ID

Identificador de la estructura de opciones de solicitud de suscripción.

Para el lenguaje de programación C, también se define la constante MQSRO_STRUC_ID_ARRAY. Tiene el mismo valor que MQSRO_STRUC_ID, pero es una matriz de caracteres en lugar de una cadena.

Versión (MQLONG)

Este es el número de versión de la estructura; el valor debe ser:

MQSRO_VERSION_1

Version-1 Estructura de opciones de solicitud de suscripción.

La siguiente constante especifica el número de versión de la versión actual:

MQSRO_VERSIÓN_ACTUAL

Versión actual de la estructura de opciones de solicitud de suscripción.

Este campo siempre es un campo de entrada. El valor inicial de este campo es MQSRO_VERSION_1.

Opciones (MQLONG)

Se debe especificar una de las siguientes opciones. Solo se puede especificar una opción.

MQSRO_FALLO_SI_EN_REPOSO

La llamada MQSUBRQ falla si el gestor de colas se encuentra en estado de inactividad. En z/OS, para una aplicación CICS IMS o, esta opción también fuerza el fallo de la llamada MQSUBRQ si la conexión se encuentra en estado de inactividad.

Opción predeterminada: Si no se requiere la opción descrita anteriormente, se debe utilizar la siguiente opción:

MQSRO_NINGUNO

Utilice este valor para indicar que no se han especificado otras opciones; todas las opciones toman sus valores predeterminados.

MQSRO_NONE ayuda a la documentación del programa. Aunque no está previsto que esta opción se utilice con ninguna otra, dado que su valor es cero, este uso no puede detectarse.

NumPubs (MQLONG)

Este es un campo de salida, devuelto a la aplicación para indicar el número de publicaciones enviadas a la cola de suscripción como resultado de esta llamada. Aunque se ha enviado este número de publicaciones como resultado de esta llamada, no hay garantía de que la aplicación pueda obtener tantos mensajes, especialmente si se trata de mensajes no persistentes.

Puede haber más de una publicación si el tema suscrito contenía un comodín. Si no había comodines en la cadena del tema cuando se creó la suscripción representada por *Hsub*, como máximo se envía una publicación como resultado de esta llamada.

MQSTS - Estructura de informes de estado

La estructura MQSTS es un parámetro de salida del comando MQSTAT. El comando MQSTAT se utiliza para recuperar información sobre el estado. Esta información se devuelve en una estructura MQSTS.

Conjunto de caracteres y codificación

Los datos de caracteres en MQSTS se encuentran en el juego de caracteres del gestor de colas local; esto viene dado por el atributo del *CodedCharSetId* gestor de colas. Los datos numéricos en MQSTS están en la codificación nativa de la máquina; esto viene dado por *Encoding*.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 562. Campos en MQSTS		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>StrucId</u> (identificador de estructura)	MQSTS_STRUC_ID	'STAT-'
<u>Versión</u> (número de versión de la estructura)	MQSTS_VERSION_1	1
<u>CompCode</u> (código de finalización del primer error)	MQCC_OK	0
<u>Motivo</u> (código de motivo del primer error)	MQRC_NONE	0
<u>PutSuccessCount</u> (número de llamadas asíncronas exitosas)	Ninguna	0
<u>PutWarningCount</u> (número de llamadas asíncronas put que generaron advertencias)	Ninguna	0
<u>PutFailureCount</u> (número de llamadas asíncronas fallidas)	Ninguna	0
<u>ObjectType</u> (tipo de objeto defectuoso)	MQOT_Q	1
<u>ObjectName</u> (nombre del objeto defectuoso)	Ninguna	Cadena nula o espacios en blanco
<u>ObjectQMgrName</u> (nombre del gestor de colas propietario del objeto defectuoso)	Ninguna	Cadena nula o espacios en blanco
<u>ResolvedObjectName</u> (nombre resuelto de la cola de destino)	Ninguna	Cadena nula o espacios en blanco
<u>ResolvedQMgrName</u> (nombre resuelto del gestor de colas de destino)	Ninguna	Cadena nula o espacios en blanco
Nota: Los campos restantes se ignoran si la versión es inferior a MQSTS_VERSION_2.		
<u>ObjectString</u> (nombre largo del objeto defectuoso)	MQCHARV_DEFAULT	{NULL,0,0,0,-3}
<u>SubName</u> (nombre de la suscripción fallida)	MQCHARV_DEFAULT	{NULL,0,0,0,-3}
<u>OpenOptions</u> (opciones abiertas asociadas con el fallo)	Ninguna	0
<u>SubOpciones</u> (opciones de suscripción asociadas con el fallo)	Ninguna	0

Tabla 562. Campos en MQSTS (continuación)

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
Notas: 1. El símbolo ~ representa un único carácter en blanco. 2. El valor «Cadena nula o espacios en blanco» denota la cadena nula en C y los caracteres en blanco en otros lenguajes de programación. 3. En el lenguaje de programación C, la variable macro MQSTS_DEFAULT contiene los valores que se enumeran en la tabla. Se puede utilizar de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQSTS MySTS = {MQSTS_DEFAULT};</pre>		

Declaraciones lingüísticas

Declaración C para MQSTS

```
typedef struct tagMQSTS MQSTS;
struct tagMQSTS {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQLONG     CompCode;         /* Completion Code of first error */
    MQLONG     Reason;           /* Reason Code of first error */
    MQLONG     PutSuccessCount;   /* Number of Async calls succeeded */
    MQLONG     PutWarningCount;   /* Number of Async calls had warnings */
    MQLONG     PutFailureCount;   /* Number of Async calls had failures */
    MQLONG     ObjectType;        /* Failing object type */
    MQCHAR48    ObjectName;       /* Failing object name */
    MQCHAR48    ObjectQMgrName;   /* Failing object queue manager name */
    MQCHAR48    ResolvedObjectName; /* Resolved name of destination queue */
    MQCHAR48    ResolvedQMgrName; /* Resolved name of destination qmgr */
/* Ver:1 */
    MQCHARV     ObjectString;     /* Failing object long name */
    MQCHARV     SubName;          /* Failing subscription name */
    MQLONG     OpenOptions;       /* Failing open options */
    MQLONG     SubOptions;        /* Failing subscription options */
/* Ver:2 */
};
```

COBOL declaración para MQSTS

```
** MQSTS structure
10  MQSTS.
** Structure identifier
15  MQSTS-STRUCID PIC X(4).
** Structure version number
15  MQSTS-VERSION PIC S9(9) BINARY.
** Completion Code of first error
15  MQSTS-COMPCODE PIC S9(9) BINARY.
** Reason Code of first error
15  MQSTS-REASON PIC S9(9) BINARY.
** Number of Async put calls succeeded
15  MQSTS-PUTSUCCESSCOUNT PIC S9(9) BINARY.
** Number of Async put calls had warnings
15  MQSTS-PUTWARNINGCOUNT PIC S9(9) BINARY.
** Number of Async put calls had failures
15  MQSTS-PUTFAILURECOUNT PIC S9(9) BINARY.
** Failing object type
15  MQSTS-OBJECTTYPE PIC S9(9) BINARY.
** Failing object name
15  MQSTS-OBJECTNAME PIC X(48).
** Failing object queue manager
15  MQSTS-OBJECTQMGRNAME PIC X(48).
** Resolved name of destination queue
15  MQSTS-RESOLVEDOBJECTNAME PIC X(48).
** Resolved name of destination qmgr
15  MQSTS-RESOLVEDQMGRNAME PIC X(48).
** Ver:1 **
```

```

** Failing object long name
15 MQSTS-OBJECTSTRING.
** Address of variable length string
20 MQSTS-OBJECTSTRING-VSPTR POINTER.
** Offset of variable length string
20 MQSTS-OBJECTSTRING-VSOFFSET PIC S9(9) BINARY.
** Size of buffer
20 MQSTS-OBJECTSTRING-VSBUFSIZE PIC S9(9) BINARY.
** Length of variable length string
20 MQSTS-OBJECTSTRING-VSLENGTH PIC S9(9) BINARY.
** CCSID of variable length string
20 MQSTS-OBJECTSTRING-VSCCSID PIC S9(9) BINARY.
** Failing subscription name
15 MQSTS-SUBNAME.
** Address of variable length string
20 MQSTS-SUBNAME-VSPTR POINTER.
** Offset of variable length string
20 MQSTS-SUBNAME-VSOFFSET PIC S9(9) BINARY.
** Size of buffer
20 MQSTS-SUBNAME-VSBUFSIZE PIC S9(9) BINARY.
** Length of variable length string
20 MQSTS-SUBNAME-VSLENGTH PIC S9(9) BINARY.
** CCSID of variable length string
20 MQSTS-SUBNAME-VSCCSID PIC S9(9) BINARY.
** Failing open options
15 MQSTS-OPENOPTIONS PIC S9(9) BINARY.
** Failing subscription options
15 MQSTS-SUBOPTIONS PIC S9(9) BINARY.
** Ver:2 **

```

PL/I declaración para MQSTS

```

dcl
1 MQSTS based,
3 StructId          char(4),          /* Structure identifier */
3 Version           fixed bin(31),    /* Structure version number */
3 CompCode          fixed bin(31),    /* Completion code */
3 Reason            fixed bin(31),    /* Reason code */
3 PutSuccessCount   fixed bin(31),    /* Put success count */
3 PutWarningCount   fixed bin(31),    /* Put warning count */
3 PutFailureCount   fixed bin(31),    /* Put failure count */
3 ObjectType        fixed bin(31),    /* Object type */
3 ObjectName        char(48),         /* Object name */
3 ObjectQmgrName    char(48),         /* Object queue manager */
3 ResolvedObjectName char(48),        /* Resolved Object name */
3 ResolvedQmgrName  char(48);        /* Resolved Object queue manager */
/* Ver:1 */
3 ObjectString,     /* Failing object long name */
5 VSPtr pointer,    /* Address of variable length string */
5 VSOffset fixed bin(31), /* Offset of variable length string */
5 VSBufSize fixed bin(31), /* Size of buffer */
5 VSLength fixed bin(31), /* Length of variable length string */
5 VSCCSID fixed bin(31); /* CCSID of variable length string */
3 SubName,          /* Failing subscription name */
5 VSPtr pointer,    /* Address of variable length string */
5 VSOffset fixed bin(31), /* Offset of variable length string */
5 VSBufSize fixed bin(31), /* Size of buffer */
5 VSLength fixed bin(31), /* Length of variable length string */
5 VSCCSID fixed bin(31); /* CCSID of variable length string */
3 OpenOptions fixed bin(31); /* Failing open options */
3 SubOptions fixed bin(31); /* Failing subscription options */
/* Ver:2 */

```

High Level Assembler declaración para MQSTS

MQSTS	DSECT		
MQSTS_STRUCID	DS	CL4	Structure identifier
MQSTS_VERSION	DS	F	Structure version number
MQSTS_COMPCODE	DS	F	Completion code
MQSTS_REASON	DS	F	Reason code
MQSTS_PUTSUCCESSCOUNT	DS	F	Success count
MQSTS_PUTWARNINGCOUNT	DS	F	Warning count
MQSTS_PUTFAILURECOUNT	DS	F	Failure count
MQSTS_OBJTYPE	DS	F	Object type
MQSTS_OBJNAME	DS	CL48	Object name
MQSTS_OBJQMGR	DS	CL48	Object queue manager
MQSTS_ROBJNAME	DS	CL48	Resolved object name
MQSTS_ROBJQMGR	DS	CL48	Resolved object queue manager
MQSTS_OBJECTSTRING	DS	0F	Force fullword alignment


```

MQSTS_OBJECTSTRING_VSPTR    DS    A    Address of variable length string
MQSTS_OBJECTSTRING_VSOFFSET DS    F    Offset of variable length string
MQSTS_OBJECTSTRING_VSBUFFSIZE DS    F    Size of buffer
MQSTS_OBJECTSTRING_VSLENGTH DS    F    Length of variable length string
MQSTS_OBJECTSTRING_VSCCSID   DS    F    CCSID of variable length string
MQSTS_OBJECTSTRING_LENGTH    EQU    *-MQSTS_OBJECTSTRING
                                ORG    MQSTS_OBJECTSTRING
MQSTS_OBJECTSTRING_AREA      DS    CL(MQSTS_OBJECTSTRING_LENGTH)
*
MQSTS_SUBNAME                DS    0F Force fullword alignment
MQSTS_SUBNAME_VSPTR          DS    A    Address of variable length string
MQSTS_SUBNAME_VSOFFSET       DS    F    Offset of variable length string
MQSTS_SUBNAME_VSBUFFSIZE     DS    F    Size of buffer
MQSTS_SUBNAME_VSLENGTH       DS    F    Length of variable length string
MQSTS_SUBNAME_VSCCSID        DS    F    CCSID of variable length string
MQSTS_SUBNAME_LENGTH         EQ     *-MQSTS_SUBNAME
                                ORG    MQSTS_SUBNAME
MQSTS_SUBNAME_AREA           DS    CL(MQSTS_SUBNAME_LENGTH)
*
MQSTS_OPENOPTIONS            DS    F    Failing open options
MQSTS_SUBOPTIONS             DS    F    Failing subscription option
MQSTS_LENGTH                 EQU    *-MQSTS
                                ORG    MQSTS
MQSTS_AREA                   DS    CL(MQSTS_LENGTH)

```

Referencia relacionada

“MQSTAT: recuperar información de estado” en la página 786

Utilice la llamada MQSTAT para recuperar información sobre el estado. El tipo de información de estado devuelta viene determinado por el valor Tipo especificado en la llamada.

Detalles de campo para MQSTS

La estructura de informes de estado MQSTS es un parámetro de salida del comando MQSTAT. El comando MQSTAT se utiliza para recuperar información sobre el estado. Esta información se devuelve en una estructura MQSTS.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 563. Campos en MQSTS		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>StrucId</u> (identificador de estructura)	MQSTS_STRUC_ID	'STAT-'
<u>Versión</u> (número de versión de la estructura)	MQSTS_VERSION_1	1
<u>CompCode</u> (código de finalización del primer error)	MQCC_OK	0
<u>Motivo</u> (código de motivo del primer error)	MQRC_NONE	0
<u>PutSuccessCount</u> (número de llamadas asíncronas exitosas)	Ninguna	0
<u>PutWarningCount</u> (número de llamadas asíncronas put que generaron advertencias)	Ninguna	0
<u>PutFailureCount</u> (número de llamadas asíncronas fallidas)	Ninguna	0
<u>ObjectType</u> (tipo de objeto defectuoso)	MQOT_Q	1
<u>ObjectName</u> (nombre del objeto defectuoso)	Ninguna	Cadena nula o espacios en blanco
<u>ObjectQMgrName</u> (nombre del gestor de colas propietario del objeto defectuoso)	Ninguna	Cadena nula o espacios en blanco

Tabla 563. Campos en MQSTS (continuación)

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>ResolvedObjectName</u> (nombre resuelto de la cola de destino)	Ninguna	Cadena nula o espacios en blanco
<u>ResolvedQMgrName</u> (nombre resuelto del gestor de colas de destino)	Ninguna	Cadena nula o espacios en blanco
Nota: Los campos restantes se ignoran si la versión es inferior a MQSTS_VERSION_2.		
<u>ObjectString</u> (nombre largo del objeto defectuoso)	MQCHARV_DEFAULT	{NULL,0,0,0,-3}
<u>SubName</u> (nombre de la suscripción fallida)	MQCHARV_DEFAULT	{NULL,0,0,0,-3}
<u>OpenOptions</u> (opciones abiertas asociadas con el fallo)	Ninguna	0
<u>SubOpciones</u> (opciones de suscripción asociadas con el fallo)	Ninguna	0
Notas: 1. El símbolo ~ representa un único carácter en blanco. 2. El valor «Cadena nula o espacios en blanco» denota la cadena nula en C y los caracteres en blanco en otros lenguajes de programación. 3. En el lenguaje de programación C, la variable macro MQSTS_DEFAULT contiene los valores que se enumeran en la tabla. Se puede utilizar de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQSTS MySTS = {MQSTS_DEFAULT};</pre>		

StrucId (MQCHAR4)

Este es el identificador de estructura de la estructura de notificación de estado. Siempre es un campo de entrada. Su valor es MQSTS_STRUC_ID.

El valor debe ser:

MQSTS_STRUC_ID

Identificador de la estructura de notificación de estado.

Para el lenguaje de programación C, también se MQSTS_STRUC_ID_ARRAY define la constante. Esto tiene el mismo valor que MQSTS_STRUC_ID, pero es una matriz de caracteres en lugar de una cadena.

Versión (MQLONG)

El número de versión de la estructura.

El valor debe ser uno de los siguientes:

MQSTS_VERSION_1

Estructura de informes de estado de la versión 1.

MQSTS_VERSION_2

Estructura de informes de estado de la versión 2.

La siguiente constante especifica el número de versión de la versión actual:

MQSTS_VERSIÓN_ACTUAL

Versión actual de la estructura de informes de estado. La versión actual es MQSTS_VERSION_2.

Version siempre es un campo de entrada. Su valor inicial es MQSTS_VERSION_1.

CompCode (MQLONG)

El código de finalización de la operación sobre la que se informa.

La interpretación de CompCode depende del valor del parámetro MQSTATType .

MQSTAT_TYPE_ASYNC_ERROR

Este es el código de finalización resultante de una operación asíncrona previa de inserción en el objeto especificado en ObjectName.

MQSTAT_TYPE_RECONEXIÓN

Si la conexión se está reconectando o no se ha podido reconectar, este es el código de finalización que ha provocado que la conexión comience a reconectarse.

Si la conexión está actualmente conectada, el valor es MQCC_OK.

MQSTAT_TYPE_ERROR_RECONEXIÓN

Si la conexión no se ha podido restablecer, este es el código de finalización que ha provocado el fallo de la reconexión.

Si la conexión está actualmente conectada o se está reconectando, el valor es MQCC_OK.

CompCode siempre es un campo de salida. Su valor inicial es MQCC_OK.

Razón (MQLONG)

El código de motivo de la operación sobre la que se informa.

La interpretación de Reason depende del valor del parámetro MQSTATType .

MQSTAT_TYPE_ASYNC_ERROR

Este es el código de motivo resultante de una operación asíncrona previa de inserción en el objeto especificado en ObjectName.

MQSTAT_TYPE_RECONEXIÓN

Si la conexión se está reconectando o no se ha podido reconectar, este es el código de motivo que ha provocado que la reconexión comience a reconectarse.

Si la conexión está actualmente conectada, el valor es MQRC_NONE.

MQSTAT_TYPE_ERROR_RECONEXIÓN

Si la conexión no se ha podido restablecer, este es el código de motivo que ha provocado el fallo en el restablecimiento de la conexión.

Si la conexión está actualmente conectada o se está reconectando, el valor es MQRC_NONE.

Reason es un campo de salida. Su valor inicial es MQRC_NONE.

PutSuccessCount (MQLONG)

El número de operaciones de inserción asíncronas que se han realizado correctamente.

El valor de PutSuccessCount depende del valor del parámetro MQSTATType .

MQSTAT_TYPE_ASYNC_ERROR

El número de operaciones de inserción asíncronas en el objeto nombrado en la MQSTS estructura que se completaron con MQCC_OK.

MQSTAT_TYPE_RECONEXIÓN

Cero.

MQSTAT_TYPE_ERROR_RECONEXIÓN

Cero.

PutSuccessCount es un campo de salida. Su valor inicial es cero.

PutWarningCount (MQLONG)

El número de operaciones de inserción asíncronas que finalizaron con una advertencia.

El valor de PutWarningCount depende del valor del parámetro **MQSTATType**.

MQSTAT_TYPE_ASYNC_ERROR

El número de operaciones de inserción asíncronas en el objeto nombrado en la MQSTS estructura que se completaron con MQCC_WARNING.

MQSTAT_TYPE_RECONEXIÓN

Cero.

MQSTAT_TYPE_ERROR_RECONEXIÓN

Cero.

PutWarningCount es un campo de salida. Su valor inicial es cero.

PutFailureCount (MQLONG)

El número de operaciones de inserción asíncronas que han fallado.

El valor de PutFailureCount depende del valor del parámetro **MQSTATType**.

MQSTAT_TYPE_ASYNC_ERROR

El número de operaciones de inserción asíncronas en el objeto nombrado en la MQSTS estructura que se completaron con MQCC_FAILED.

MQSTAT_TYPE_RECONEXIÓN

Cero.

MQSTAT_TYPE_ERROR_RECONEXIÓN

Cero.

PutFailureCount es un campo de salida. Su valor inicial es cero.

ObjectType (MQLONG)

El tipo del objeto mencionado en sobre *ObjectName* el que se informa.

Los valores posibles de ObjectType se enumeran en [“MQOT_* \(Tipos de objeto y tipos de objeto extendidos\)”](#) en la página 103.

ObjectType es un campo de salida. Su valor inicial es MQOT_Q.

ObjectName (MQCHAR48)

El nombre del objeto sobre el que se informa.

La interpretación de ObjectName depende del valor del parámetro **MQSTATType**.

MQSTAT_TYPE_ASYNC_ERROR

Este es el nombre de la cola o tema utilizado en la operación de inserción, cuyo fallo se notifica en los campos *CompCode Reason* y de la MQSTS estructura.

MQSTAT_TYPE_RECONEXIÓN

Si la conexión se está reconectando, este es el nombre del gestor de colas asociado a la conexión.

MQSTAT_TYPE_ERROR_RECONEXIÓN

Si la conexión no se ha podido restablecer, este es el nombre del objeto que ha provocado el fallo en el restablecimiento de la conexión. El motivo del fallo se indica en los campos *CompCode Reason* y de la MQSTS estructura.

ObjectName es un campo de salida. Su valor inicial es la cadena nula en C y 48 caracteres en blanco en otros lenguajes de programación.

ObjectQMgrName (MQCHAR48)

El nombre del gestor de colas sobre el que se está informando.

La interpretación de ObjectQMgrName depende del valor del parámetro MQSTATType .

MQSTAT_TYPE_ASYNC_ERROR

Este es el nombre del gestor de colas en el que se define el *ObjectName* objeto. Un nombre que está completamente en blanco hasta el primer carácter nulo o el final del campo indica el gestor de colas al que está conectada la aplicación (el gestor de colas local).

MQSTAT_TYPE_RECONEXIÓN



El **ObjectQMgrName** campo contiene el nombre del gestor de colas al que se solicita la reconexión, o está en blanco si no se ha especificado ningún gestor de colas. Si es posible, el cliente intenta volver a conectarse a un gestor de colas con ese nombre.



En blanco.

MQSTAT_TYPE_ERROR_RECONEXIÓN

Si la conexión no se ha podido restablecer, este es el nombre del objeto que ha provocado el fallo en el restablecimiento de la conexión. El motivo del fallo se indica en los campos *CompCode Reason* y de la MQSTS estructura.

ObjectQMgrName es un campo de salida. Su valor es la cadena nula en C y 48 caracteres en blanco en otros lenguajes de programación.

ResolvedObjectName (MQCHAR48)

El nombre del objeto nombrado en *ObjectName* después de que el administrador de colas local resuelva el nombre.

La interpretación de ResolvedObjectName depende del valor del parámetro MQSTATType .

MQSTAT_TYPE_ASYNC_ERROR

ResolvedObjectName es el nombre del objeto nombrado en *ObjectName* después de que el administrador de colas local resuelva el nombre. El nombre devuelto es el nombre de un objeto que existe en el gestor de colas identificado por *ResolvedQMgrName*.

MQSTAT_TYPE_RECONEXIÓN

En blanco.

MQSTAT_TYPE_ERROR_RECONEXIÓN

En blanco.

ResolvedObjectName es un campo de salida. Su valor inicial es la cadena nula en C y 48 caracteres en blanco en otros lenguajes de programación.

ResolvedQMgrName (MQCHAR48)

El nombre del gestor de colas de destino después de que el gestor de colas local resuelva el nombre.

La interpretación de `ResolvedQMgrName` depende del valor del parámetro `MQSTATType`.

MQSTAT_TYPE_ASYNC_ERROR

`ResolvedQMgrName` es el nombre del gestor de colas de destino después de que el gestor de colas local resuelva el nombre. El nombre devuelto es el nombre del gestor de colas propietario del objeto identificado por *ResolvedObjectName*. *ResolvedQMgrName* podría ser el nombre del administrador de colas local.

MQSTAT_TYPE_RECONEXIÓN

En blanco.

MQSTAT_TYPE_ERROR_RECONEXIÓN

En blanco.

`ResolvedQMgrName` siempre es un campo de salida. Su valor inicial es la cadena nula en C y 48 caracteres en blanco en otros lenguajes de programación.

ObjectString (MQCHARV)

Nombre largo del objeto defectuoso sobre el que se informa. Presente solo en la versión 2 de MQSTS o superior.

La interpretación de `ObjectString` depende del valor del parámetro `MQSTATType`.

MQSTAT_TYPE_ASYNC_ERROR

Este es el nombre completo del objeto de la cola o tema utilizado en la MQPUT operación que ha fallado.

MQSTAT_TYPE_RECONEXIÓN

Serie de caracteres de longitud cero

MQSTAT_TYPE_ERROR_RECONEXIÓN

Este es el nombre completo del objeto que provocó el fallo de la reconexión.

`ObjectString` es un campo de salida. Su valor inicial es una cadena de longitud cero.

SubName (MQCHARV)

El nombre de la suscripción fallida. Presente solo en la versión 2 de MQSTS o superior.

La interpretación de `SubName` depende del valor del parámetro `MQSTATType`.

MQSTAT_TYPE_ASYNC_ERROR

Cadena de longitud cero.

MQSTAT_TYPE_RECONEXIÓN

Cadena de longitud cero.

MQSTAT_TYPE_ERROR_RECONEXIÓN

El nombre de la suscripción que provocó el fallo de la reconexión. Si no hay ningún nombre de suscripción disponible, o si el error no está relacionado con una suscripción, se trata de una cadena de longitud cero.

`SubName` es un campo de salida. Su valor inicial es una cadena de longitud cero.

OpenOptions (MQLONG)

Se `OpenOptions` utiliza para abrir el objeto sobre el que se informa. Presente solo en la versión 2 de MQSTS o superior.

El valor de `OpenOptions` depende del valor del parámetro `MQSTATType`.

MQSTAT_TYPE_ASYNC_ERROR

Cero.

MQSTAT_TYPE_RECONEXIÓN

Cero.

MQSTAT_TYPE_ERROR_RECONEXIÓN

El `OpenOptions` utilizado cuando se produjo el fallo. El motivo del fallo se indica en los campos *CompCode Reason* y de la MQSTS estructura.

`OpenOptions` es un campo de salida. Su valor inicial es cero.

SubOptions (MQLONG)

Se `SubOptions` utilizaba para abrir la suscripción caducada. Presente solo en la versión 2 de MQSTS o superior.

La interpretación de `SubOptions` depende del valor del parámetro `MQSTATType`.

MQSTAT_TYPE_ASYNC_ERROR

Cero.

MQSTAT_TYPE_RECONEXIÓN

Cero.

MQSTAT_TYPE_ERROR_RECONEXIÓN

El `SubOptions` utilizado cuando se produjo el fallo. Si el error no está relacionado con la suscripción a un tema, el valor devuelto es cero.

`SubOptions` es un campo de salida. Su valor inicial es cero.

MQTM - Mensaje de activación

La estructura MQTM describe los datos del mensaje de activación que envía el gestor de colas a una aplicación de supervisión de activadores cuando se produce un evento de activación para una cola. Esta estructura forma parte de la Interfaz del IBM MQ monitor de activadores (TMI), que es una de las IBM MQ interfaces del marco.

Nombre de formato

MQFMT_TRIGGER.

Conjunto de caracteres y codificación

Los datos de caracteres en MQTM se encuentran en el juego de caracteres del gestor de colas que genera el MQTM. Los datos numéricos en MQTM están en la codificación de máquina del gestor de colas que genera el MQTM.

El conjunto de caracteres y la codificación del MQTM vienen dados por los campos *CodedCharSetId Encoding* y en:

- El MQMD (si la estructura MQTM se encuentra al inicio de los datos del mensaje), o
- La estructura de encabezado que precede a la estructura MQTM (en todos los demás casos).

Utilización

Es posible que una aplicación de monitorización de desencadenantes necesite pasar parte o toda la información del mensaje de desencadenante a la aplicación que inicia la aplicación de monitorización de desencadenantes. La información que podría necesitar la aplicación iniciada incluye *QName*, *TriggerData*, y *UserData*. La aplicación de monitorización de disparadores puede pasar la estructura MQTM directamente a la aplicación iniciada, o pasar una estructura e MQTMC2, dependiendo de lo que permita el entorno y resulte conveniente para la aplicación iniciada. Para obtener información sobre MQTMC2, consulte [“MQTMC2 - Mensaje de activación 2 \(formato de caracteres\)”](#) en la página 603.

- ▶ **z/OS** En z/OS, para una aplicación MQAT_CICS que se inicia utilizando la transacción CKTI, toda la estructura del mensaje de activación MQTM se pone a disposición de la transacción iniciada; la información se puede recuperar utilizando el comando EXEC CICS RETRIEVE.
- ▶ **IBM i** En IBM i, la aplicación de supervisión de desencadenantes proporcionada con IBM MQ pasa una estructura MQTMC2 a la aplicación iniciada.

Para obtener información sobre el uso de desencadenadores, consulte [Inicio IBM MQ de aplicaciones mediante desencadenadores](#).

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 564. Campos en MQTM para MQTM		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>StrucId</u> (identificador de estructura)	MQTM_STRUC_ID	' TM↵↵ '
<u>Versión</u> (número de versión de la estructura)	MQTM_VERSION_1	1
<u>QName</u> (nombre de la cola activada)	Ninguna	Cadena nula o espacios en blanco
<u>ProcessName</u> (nombre del objeto del proceso)	Ninguna	Cadena nula o espacios en blanco
<u>TriggerData</u> (datos desencadenantes)	Ninguna	Cadena nula o espacios en blanco
<u>ApplType</u> (tipo de aplicación)	Ninguna	0
<u>ApplId</u> (identificador de la aplicación)	Ninguna	Cadena nula o espacios en blanco
<u>EnvData</u> (datos medioambientales)	Ninguna	Cadena nula o espacios en blanco
<u>UserData</u> (datos del usuario)	Ninguna	Cadena nula o espacios en blanco

Tabla 564. Campos en MQTM para MQTM (continuación)

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
Notas: 1. El símbolo ~ representa un único carácter en blanco. 2. El valor «Cadena nula o espacios en blanco» denota la cadena nula en C y los caracteres en blanco en otros lenguajes de programación. 3. En el lenguaje de programación C, la variable macroMQTM_DEFAULT contiene los valores que se enumeran en la tabla. Úselo de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQTM MyTM = {MQTM_DEFAULT};</pre>		

Declaraciones lingüísticas

Declaración C para MQTM

```
typedef struct tagMQTM MQTM;
struct tagMQTM {
    MQCHAR4    StrucId;        /* Structure identifier */
    MQLONG     Version;        /* Structure version number */
    MQCHAR48    QName;         /* Name of triggered queue */
    MQCHAR48    ProcessName;   /* Name of process object */
    MQCHAR64    TriggerData;   /* Trigger data */
    MQLONG     ApplType;       /* Application type */
    MQCHAR256   ApplId;        /* Application identifier */
    MQCHAR128   EnvData;       /* Environment data */
    MQCHAR128   UserData;      /* User data */
};
```

COBOL declaración para MQTM

```
** MQTM structure
10 MQTM.
** Structure identifier
15 MQTM-STRUCID PIC X(4).
** Structure version number
15 MQTM-VERSION PIC S9(9) BINARY.
** Name of triggered queue
15 MQTM-QNAME PIC X(48).
** Name of process object
15 MQTM-PROCESSNAME PIC X(48).
** Trigger data
15 MQTM-TRIGGERDATA PIC X(64).
** Application type
15 MQTM-APPLTYPE PIC S9(9) BINARY.
** Application identifier
15 MQTM-APPLID PIC X(256).
** Environment data
15 MQTM-ENVDATA PIC X(128).
** User data
15 MQTM-USERDATA PIC X(128).
```

PL/I declaración para MQTM

```
dcl
1 MQTM based,
3 StrucId char(4), /* Structure identifier */
3 Version fixed bin(31), /* Structure version number */
3 QName char(48), /* Name of triggered queue */
3 ProcessName char(48), /* Name of process object */
3 TriggerData char(64), /* Trigger data */
3 ApplType fixed bin(31), /* Application type */
3 ApplId char(256), /* Application identifier */
3 EnvData char(128), /* Environment data */
3 UserData char(128); /* User data */
```

High Level Assembler declaración para MQTM

```

MQTM          DSECT
MQTM_STRUCID  DS    CL4      Structure identifier
MQTM_VERSION  DS    F        Structure version number
MQTM_QNAME    DS    CL48     Name of triggered queue
MQTM_PROCESSNAME DS    CL48   Name of process object
MQTM_TRIGGERDATA DS    CL64   Trigger data
MQTM_APPLTYPE DS    F        Application type
MQTM_APPLID   DS    CL256    Application identifier
MQTM_ENVDATA  DS    CL128    Environment data
MQTM_USERDATA DS    CL128    User data
*
MQTM_LENGTH   EQU    *-MQTM
              ORG    MQTM
MQTM_AREA     DS      CL(MQTM_LENGTH)

```

Declaración de Visual Basic para MQTM

```

Type MQTM
  StrucId      As String*4    'Structure identifier'
  Version      As Long        'Structure version number'
  QName        As String*48   'Name of triggered queue'
  ProcessName  As String*48   'Name of process object'
  TriggerData  As String*64   'Trigger data'
  ApplType     As Long        'Application type'
  ApplId       As String*256  'Application identifier'
  EnvData      As String*128  'Environment data'
  UserData     As String*128  'User data'
End Type

```

MQMD para un mensaje de activación

Tabla 565. Configuración de los campos en el MQMD de un mensaje de activación generado por el gestor de colas

Campo de MQMD	Valor utilizado
<i>StrucId</i>	MQMD_STRUC_ID
<i>Version</i>	MQMD_VERSION_1
<i>Report</i>	MQRO_NONE
<i>MsgType</i>	MQMT_DATAGRAM
<i>Expiry</i>	MQEI_UNLIMITED
<i>Feedback</i>	MQFB_NONE
<i>Encoding</i>	MQENC_NATIVE
<i>CodedCharSetId</i>	Atributo CodedCharSetId del gestor de colas
<i>Format</i>	MQFMT_TRIGGER
<i>Priority</i>	Atributo DefPriority de la cola de inicio
<i>Persistence</i>	MQPER_NOT_PERSISTENT
<i>MsgId</i>	Un valor único
<i>CorrelId</i>	MQCI_NONE
<i>BackoutCount</i>	0
<i>ReplyToQ</i>	Espacios en blanco
<i>ReplyToQMGr</i>	Nombre de gestor de colas
<i>UserIdentifier</i>	Espacios en blanco
<i>AccountingToken</i>	MQACT_NONE

Tabla 565. Configuración de los campos en el MQMD de un mensaje de activación generado por el gestor de colas (continuación)

Campo de MQMD	Valor utilizado
<i>ApplIdentityData</i>	Espacios en blanco
<i>PutApplType</i>	MQAT_QMGR, o según corresponda para el agente del canal de mensajes
<i>PutApplName</i>	Primeros 28 bytes del nombre del gestor de colas
<i>PutDate</i>	Fecha en la que se envía el mensaje de activación
<i>PutTime</i>	Momento en el que se envía el mensaje de activación
<i>ApplOriginData</i>	Espacios en blanco

Se recomienda que una aplicación que genere un mensaje de activación establezca valores similares, excepto en los siguientes casos:

- El *Priority* campo se puede establecer en MQPRI_PRIORITY_AS_Q_DEF (el gestor de colas cambiará esto a la prioridad predeterminada para la cola de inicio cuando se envíe el mensaje).
- El *ReplyToQMGr* campo se puede dejar en blanco (el gestor de colas lo cambiará por el nombre del gestor de colas local cuando se envíe el mensaje).
- Configure los campos de contexto según sea adecuado para la aplicación.

Detalles de campo para MQTM

La estructura del mensaje de activación MQTM describe los datos del mensaje de activación que envía el gestor de colas a una aplicación de supervisión de activaciones cuando se produce un evento de activación para una cola.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 566. Campos en MQTM para MQTM		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>StrucId</u> (identificador de estructura)	MQTM_STRUC_ID	'TM--'
<u>Versión</u> (número de versión de la estructura)	MQTM_VERSION_1	1
<u>QName</u> (nombre de la cola activada)	Ninguna	Cadena nula o espacios en blanco
<u>ProcessName</u> (nombre del objeto del proceso)	Ninguna	Cadena nula o espacios en blanco
<u>TriggerData</u> (datos desencadenantes)	Ninguna	Cadena nula o espacios en blanco
<u>ApplType</u> (tipo de aplicación)	Ninguna	0
<u>ApplId</u> (identificador de la aplicación)	Ninguna	Cadena nula o espacios en blanco
<u>EnvData</u> (datos medioambientales)	Ninguna	Cadena nula o espacios en blanco
<u>UserData</u> (datos del usuario)	Ninguna	Cadena nula o espacios en blanco

Tabla 566. Campos en MQTM para MQTM (continuación)

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
Notas: 1. El símbolo ~ representa un único carácter en blanco. 2. El valor «Cadena nula o espacios en blanco» denota la cadena nula en C y los caracteres en blanco en otros lenguajes de programación. 3. En el lenguaje de programación C, la variable macroMQTM_DEFAULT contiene los valores que se enumeran en la tabla. Úselo de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQTM MyTM = {MQTM_DEFAULT};</pre>		

StrucId (MQCHAR4)

Este es el identificador de estructura de la estructura del mensaje desencadenante. Siempre es un campo de entrada. Su valor es MQTM_STRUC_ID.

El valor debe ser:

MQTM_STRUC_ID

Identificador de la estructura del mensaje desencadenante.

Para el lenguaje de programación C, también se define la constante MQTM_STRUC_ID_ARRAY. Tiene el mismo valor que MQTM_STRUC_ID, pero es una matriz de caracteres en lugar de una cadena.

Versión (MQLONG)

Este es el número de versión de la estructura. El valor debe ser:

MQTM_VERSION_1

Número de versión para la estructura del mensaje de activación.

La siguiente constante especifica el número de versión de la versión actual:

MQTM_VERSIÓN_ACTUAL

Versión actual de la estructura del mensaje de activación.

El valor inicial de este campo es MQTM_VERSION_1.

QName (MQCHAR48)

Este es el nombre de la cola para la que se produjo un evento desencadenante, y lo utiliza la aplicación iniciada por la aplicación de supervisión de desencadenantes. El gestor de colas inicializa este campo con el valor del **QName** atributo de la cola activada; consulte para obtener [“Atributos para colas” en la página 841](#) más detalles sobre este atributo.

Los nombres que son más cortos que la longitud definida del campo se rellenan a la derecha con espacios en blanco; no terminan prematuramente con un carácter nulo.

La longitud de este campo viene determinada por MQ_Q_NAME_LENGTH. El valor inicial de este campo es la cadena nula en C y 48 caracteres en blanco en otros lenguajes de programación.

ProcessName (MQCHAR48)

Este es el nombre del objeto del proceso del gestor de colas especificado para la cola activada, y puede ser utilizado por la aplicación de supervisión de activadores que recibe el mensaje de activación. El gestor de colas inicializa este campo con el valor del **ProcessName** atributo de la cola identificada por el **QName** campo; consulte para obtener [“Atributos para colas” en la página 841](#) más detalles sobre este atributo.

Los nombres que son más cortos que la longitud definida del campo siempre se rellenan a la derecha con espacios en blanco; no terminan prematuramente con un carácter nulo.

La longitud de este campo viene determinada por MQ_PROCESS_NAME_LENGTH. El valor inicial de este campo es la cadena nula en C y 48 caracteres en blanco en otros lenguajes de programación.

TriggerData (MQCHAR64)

Se trata de datos de formato libre para su uso por parte de la aplicación de monitorización de disparadores que recibe el mensaje de disparo. El gestor de colas inicializa este campo con el valor del **TriggerData** atributo de la cola identificada por el *QName* campo; consulte [“Atributos para colas” en la página 841](#) para obtener más información sobre este atributo. El contenido de estos datos no tiene importancia para el gestor de colas.

En z/OS, para una CICS aplicación que comenzó a utilizar la transacción CKTI, esta información no se utiliza.

La longitud de este campo viene dada por MQ_TRIGGER_DATA_LENGTH. El valor inicial de este campo es la cadena nula en C y 64 caracteres en blanco en otros lenguajes de programación.

ApplType (MQLONG)

Esto identifica la naturaleza del programa que se va a iniciar y es utilizado por la aplicación de monitorización de disparadores que recibe el mensaje de disparo. El gestor de colas inicializa este campo con el valor del **ApplType** atributo del objeto de proceso identificado por el *ProcessName* campo; consulte [“Atributos de las definiciones de proceso” en la página 878](#) para obtener más detalles sobre este atributo. El contenido de estos datos no tiene importancia para el gestor de colas.

ApplType puede tener uno de los siguientes valores estándar. También se pueden utilizar tipos definidos por el usuario, pero deben restringirse a valores comprendidos entre MQAT_USER_FIRST y MQAT_USER_LAST:

MQAT_AIX

AIX aplicación (mismo valor que MQAT_UNIX).

MQAT_LOTE

aplicación por lotes

MQAT_CORREDOR

Aplicación de intermediario

MQAT_CICS

CICS transacción.

MQAT_CICS_PUENTE

CICS bridge .

MQAT_CICS_VSE

CICS/VSE transacción.

MQAT_DOS

IBM MQ MQI client aplicación en PC DOS.

MQAT_IMS

IMS .

MQAT_IMS_PUENTE

IMS aplicación puente.

MQAT_JAVA

Java .

MQAT_MVS

MVS o aplicación TSO (mismo valor que MQAT_ZOS).

MQAT_NOTAS_AGENTE

Lotus Notes Solicitud de agente.

MQAT_OS390

OS/390 aplicación (mismo valor que MQAT_ZOS).

MQAT_OS400

IBM i .

MQAT_RRS_LOTE

Solicitud por lotes de RRS.

MQAT_UNIX

UNIX .

MQAT_DESCONOCIDO

Aplicación de tipo desconocido.

MQAT_USUARIO

Tipo de aplicación definido por el usuario.

MQAT_VOS

Aplicación Stratus VOS.

MQAT_VENTANAS

Aplicación Windows de 16 bits.

MQAT_WINDOWS_NT

Aplicación Windows de 32 bits.

MQAT_WLM

z/OS Aplicación de gestión de la carga de trabajo.

MQAT_XCF

XCF.

MQAT_ZOS

z/OS .

MQAT_USUARIO_PRIMERO

Valor más bajo para el tipo de aplicación definido por el usuario.

MQAT_USUARIO_ÚLTIMO

Valor más alto para el tipo de aplicación definido por el usuario.

El valor inicial de este campo es 0.

AppId (MQCHAR256)

Se trata de una cadena de caracteres que identifica la aplicación que se va a iniciar y que utiliza la aplicación de supervisión de activadores que recibe el mensaje de activación. El gestor de colas inicializa este campo con el valor del **AppId** atributo del objeto de proceso identificado por el *ProcessName* campo; consulte [“Atributos de las definiciones de proceso” en la página 878](#) para obtener más detalles sobre este atributo. El contenido de estos datos no tiene importancia para el gestor de colas.

El significado de *AppId* lo determina la aplicación de monitorización de desencadenantes. El monitor de activación proporcionado por IBM MQ requiere *AppId* que sea el nombre de un programa ejecutable. Las siguientes notas se aplican a los entornos indicados:

- En z/OS, *AppId* es:
 - Un CICS identificador de transacción, para aplicaciones iniciadas utilizando la transacción CKTI del monitor CICS de disparadores
 - Un IMS identificador de transacción, para aplicaciones iniciadas utilizando el monitor IMS de disparadores CSQQTRMN
- En Windows los sistemas, el nombre del programa puede ir precedido de una unidad y una ruta de directorio.
- En IBM i, el nombre del programa puede ir precedido del nombre de una biblioteca y del carácter /.
- En AIX and Linux, el nombre del programa puede ir precedido de una ruta de directorio.

La longitud de este campo viene dada por MQ_PROCESS_APPL_ID_LENGTH. El valor inicial de este campo es la cadena nula en C y 256 caracteres en blanco en otros lenguajes de programación.

EnvData (MQCHAR128)

Se trata de una cadena de caracteres que contiene información relacionada con el entorno de la aplicación que se va a iniciar y que es utilizada por la aplicación de supervisión de activadores que recibe el mensaje de activación. El gestor de colas inicializa este campo con el valor del **EnvData** atributo del objeto de proceso identificado por el *ProcessName* campo; consulte [“Atributos de las definiciones de proceso”](#) en la página 878 para obtener más detalles sobre este atributo. El contenido de estos datos no tiene importancia para el gestor de colas.

En z/OS, para una CICS aplicación iniciada mediante la transacción CKTI, o una IMS aplicación que se iniciará mediante la transacción CSQQTRMN, esta información no se utiliza.

La longitud de este campo viene dada por MQ_PROCESS_ENV_DATA_LENGTH. El valor inicial de este campo es la cadena nula en C y 128 caracteres en blanco en otros lenguajes de programación.

UserData (MQCHAR128)

Se trata de una cadena de caracteres que contiene información del usuario relevante para la aplicación que se va a iniciar y que es utilizada por la aplicación de supervisión de activadores que recibe el mensaje de activación. El gestor de colas inicializa este campo con el valor del **UserData** atributo del objeto de proceso identificado por el *ProcessName* campo; consulte [“Atributos de las definiciones de proceso”](#) en la página 878 para obtener más detalles sobre este atributo. El contenido de estos datos no tiene importancia para el gestor de colas.

Para Microsoft Windows, la cadena de caracteres no debe contener comillas dobles si la definición del proceso se va a pasar a **runmqtrm**.

La longitud de este campo viene dada por MQ_PROCESS_USER_DATA_LENGTH. El valor inicial de este campo es la cadena nula en C y 128 caracteres en blanco en otros lenguajes de programación.

MQTMC2 - Mensaje de activación 2 (formato de caracteres)

Cuando una aplicación de monitor de desencadenadores recupera un mensaje de desencadenador (MQTM) de una cola de inicio, es posible que el monitor de desencadenadores tenga que pasar parte o toda la información del mensaje de desencadenador a la aplicación que inicia el monitor de desencadenadores.

La información que podría necesitar la aplicación iniciada incluye *QName*, *TriggerData*, y *UserData*. La aplicación del monitor de activación puede pasar la estructura MQTM directamente a la aplicación iniciada, o pasar una estructura MQTMC2 e en su lugar, dependiendo de lo que permita el entorno y resulte conveniente para la aplicación iniciada.

Esta estructura forma parte de la Interfaz del IBM MQ monitor de activadores (TMI), que es una de las IBM MQ interfaces del marco.

Conjunto de caracteres y codificación

Los datos de caracteres en MQTMC2 se encuentran en el juego de caracteres del gestor de colas local; esto viene dado por el atributo del **CodedCharSetId** gestor de colas.

Utilización

La estructura « MQTMC2 » es muy similar al formato de la estructura MQTM. La diferencia es que los campos que no son de caracteres en MQTM se cambian en MQTMC2 a campos de caracteres de la misma longitud, y el nombre del gestor de colas se añade al final de la estructura.

- **z/OS** En z/OS, para una aplicación MQAT_IMS que se inicia utilizando la aplicación CSQQTRMN, se pone a disposición de la aplicación iniciada una estructura MQTMC2.

- **IBM i** En IBM i, la aplicación de monitorización de desencadenantes proporcionada con IBM MQ pasa una estructura MQTMC2 a la aplicación iniciada.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 567. Campos en MQTMC2		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>StrucId</u> (identificador de estructura)	MQTMC_STRUC_ID	'TMC↵'
<u>Versión</u> (número de versión de la estructura)	MQTMC_VERSION_2	'↵↵↵2'
<u>QName</u> (nombre de la cola activada)	Ninguna	Cadena nula o espacios en blanco
<u>ProcessName</u> (nombre del objeto del proceso)	Ninguna	Cadena nula o espacios en blanco
<u>TriggerData</u> (datos desencadenantes)	Ninguna	Cadena nula o espacios en blanco
<u>ApplType</u> (tipo de aplicación)	Ninguna	Espacios en blanco
<u>ApplId</u> (identificador de la aplicación)	Ninguna	Cadena nula o espacios en blanco
<u>EnvData</u> (datos medioambientales)	Ninguna	Cadena nula o espacios en blanco
<u>UserData</u> (datos del usuario)	Ninguna	Cadena nula o espacios en blanco
<u>QMgrName</u> (nombre del administrador de la cola)	Ninguna	Cadena nula o espacios en blanco
Notas: 1. El símbolo ↵ representa un único carácter en blanco. 2. El valor «Cadena nula o espacios en blanco» denota la cadena nula en C y los caracteres en blanco en otros lenguajes de programación. 3. En el lenguaje de programación C, la variable macroMQTMC2_DEFAULT contiene los valores enumerados anteriormente. Úselo de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQTMC2 MyTMC = {MQTMC2_DEFAULT};</pre>		

Declaraciones lingüísticas

Declaración C para MQTMC2

```
typedef struct tagMQTMC2 MQTMC2;
struct tagMQTMC2 {
    MQCHAR4    StrucId;        /* Structure identifier */
    MQCHAR4    Version;        /* Structure version number */
    MQCHAR48   QName;          /* Name of triggered queue */
    MQCHAR48   ProcessName;    /* Name of process object */
    MQCHAR64   TriggerData;    /* Trigger data */
    MQCHAR4    ApplType;       /* Application type */
    MQCHAR256  ApplId;         /* Application identifier */
    MQCHAR128  EnvData;        /* Environment data */
    MQCHAR128  UserData;       /* User data */
}
```



```
MQCHAR48 QMgrName;    /* Queue manager name */
};
```

COBOL Declaración para MQTMC2

```
** MQTMC2 structure
10 MQTMC2.
** Structure identifier
15 MQTMC2-STRUCID PIC X(4).
** Structure version number
15 MQTMC2-VERSION PIC X(4).
** Name of triggered queue
15 MQTMC2-QNAME PIC X(48).
** Name of process object
15 MQTMC2-PROCESSNAME PIC X(48).
** Trigger data
15 MQTMC2-TRIGGERDATA PIC X(64).
** Application type
15 MQTMC2-APPLTYPE PIC X(4).
** Application identifier
15 MQTMC2-APPLID PIC X(256).
** Environment data
15 MQTMC2-ENVDATA PIC X(128).
** User data
15 MQTMC2-USERSDATA PIC X(128).
** Queue manager name
15 MQTMC2-QMGRNAME PIC X(48).
```

PL/I Declaración para MQTMC2

```
dcl
1 MQTMC2 based,
3 StrucId char(4), /* Structure identifier */
3 Version char(4), /* Structure version number */
3 QName char(48), /* Name of triggered queue */
3 ProcessName char(48), /* Name of process object */
3 TriggerData char(64), /* Trigger data */
3 ApplType char(4), /* Application type */
3 ApplId char(256), /* Application identifier */
3 EnvData char(128), /* Environment data */
3 UserData char(128), /* User data */
3 QMgrName char(48); /* Queue manager name */
```

High Level Assembler Declaración para MQTMC2

```
MQTMC2          DSECT
MQTMC2_STRUCID   DS CL4    Structure identifier
MQTMC2_VERSION   DS CL4    Structure version number
MQTMC2_QNAME     DS CL48   Name of triggered queue
MQTMC2_PROCESSNAME DS CL48 Name of process object
MQTMC2_TRIGGERDATA DS CL64 Trigger data
MQTMC2_APPLTYPE  DS CL4    Application type
MQTMC2_APPLID    DS CL256  Application identifier
MQTMC2_ENVDATA   DS CL128  Environment data
MQTMC2_USERSDATA DS CL128  User data
MQTMC2_QMGRNAME  DS CL48   Queue manager name
*
MQTMC2_LENGTH    EQU *-MQTMC2
ORG MQTMC2
MQTMC2_AREA      DS CL(MQTMC2_LENGTH)
```

Declaración de Visual Basic para MQTMC2

```
Type MQTMC2
StrucId As String*4 'Structure identifier'
Version As String*4 'Structure version number'
QName As String*48 'Name of triggered queue'
ProcessName As String*48 'Name of process object'
TriggerData As String*64 'Trigger data'
ApplType As String*4 'Application type'
ApplId As String*256 'Application identifier'
EnvData As String*128 'Environment data'
UserData As String*128 'User data'
QMgrName As String*48 'Queue manager name'
End Type
```

Detalles del campo para MQTMC2

Cuando una aplicación de monitor de desencadenadores recupera un mensaje de desencadenador (MQTM) de una cola de inicio, es posible que el monitor de desencadenadores tenga que pasar parte o toda la información del mensaje de desencadenador a la aplicación que inicia el monitor de desencadenadores.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 568. Campos en MQTMC2		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>StrucId</u> (identificador de estructura)	MQTMC_STRUC_ID	'TMC¬'
<u>Versión</u> (número de versión de la estructura)	MQTMC_VERSION_2	'¬¬¬2'
<u>QName</u> (nombre de la cola activada)	Ninguna	Cadena nula o espacios en blanco
<u>ProcessName</u> (nombre del objeto del proceso)	Ninguna	Cadena nula o espacios en blanco
<u>TriggerData</u> (datos desencadenantes)	Ninguna	Cadena nula o espacios en blanco
<u>ApplType</u> (tipo de aplicación)	Ninguna	Espacios en blanco
<u>ApplId</u> (identificador de la aplicación)	Ninguna	Cadena nula o espacios en blanco
<u>EnvData</u> (datos medioambientales)	Ninguna	Cadena nula o espacios en blanco
<u>UserData</u> (datos del usuario)	Ninguna	Cadena nula o espacios en blanco
<u>QMgrName</u> (nombre del administrador de la cola)	Ninguna	Cadena nula o espacios en blanco
Notas: 1. El símbolo ¬ representa un único carácter en blanco. 2. El valor «Cadena nula o espacios en blanco» denota la cadena nula en C y los caracteres en blanco en otros lenguajes de programación. 3. En el lenguaje de programación C, la variable macroMQTMC2_DEFAULT contiene los valores enumerados anteriormente. Úselo de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQTMC2 MyTMC = {MQTMC2_DEFAULT};</pre>		

StrucId (MQCHAR4)

Este es el identificador de estructura de la estructura del mensaje de activación 2 (formato de caracteres). Siempre es un campo de entrada. Su valor es MQTMC2_STRUC_ID.

El valor debe ser:

MQTMC2_STRUC_ID

Identificador de la estructura del mensaje desencadenante (formato de caracteres).

Para el lenguaje de programación C, también se define la constante MQTMC2_STRUC_ID_ARRAY. Esto tiene el mismo valor que MQTMC2_STRUC_ID, pero es una matriz de caracteres en lugar de una cadena.

Versión (MQCHAR4)

Número de versión de la estructura.

El valor debe ser:

MQTM_VERSION_2

Estructura del mensaje de activación de la versión 2 (formato de caracteres).

Para el lenguaje de programación C, también se define la constante `MQTMC\_VERSION\_2\_ARRAY`; esta tiene el mismo valor que `MQTMC\_VERSION\_2`, pero es una matriz de caracteres en lugar de una cadena.

La siguiente constante especifica el número de versión de la versión actual:

MQTM_VERSION_ACTUAL

Estructura de la versión actual del mensaje de activación (formato de caracteres).

QName (MQCHAR48)

Nombre de la cola activada.

Véase el [*QName*](#) campo en la estructura MQTM.

ProcessName (MQCHAR48)

Nombre del objeto de proceso.

Véase el [*ProcessName*](#) campo en la estructura MQTM.

TriggerData (MQCHAR64)

Datos del desencadenante.

Véase el [*TriggerData*](#) campo en la estructura MQTM.

ApplType (MQCHAR4)

Tipo de aplicación.

Este campo siempre contiene espacios en blanco, independientemente del valor del [*ApplType*](#) campo en la estructura MQTM del mensaje de activación original.

ApplId (MQCHAR256)

Identificador de la aplicación.

Véase el [*ApplId*](#) campo en la estructura MQTM.

EnvData (MQCHAR128)

Datos medioambientales.

Véase el [*EnvData*](#) campo en la estructura MQTM.

UserData (MQCHAR128)

Datos de usuario.

Véase el [*UserData*](#) campo en la estructura MQTM.

QMgrName (MQCHAR48)

Nombre del gestor de colas.

Este es el nombre del gestor de colas en el que se produjo el evento desencadenante.

MQWIH - Cabecera de información de trabajo

Si un mensaje debe ser procesado por el z/OS gestor de carga de trabajo (WLM), el mensaje debe comenzar con una estructura MQWIH. Esta estructura describe la información que debe estar presente al inicio de un mensaje que va a ser gestionado por WLM.

Disponibilidad

Todos IBM MQ los sistemas, además de IBM MQ los clientes conectados a estos sistemas.

Nombre de formato

MQFMT_WORK_INFO_HEADER.

Conjunto de caracteres y codificación

Los campos de la estructura MQWIH están en el juego de caracteres y la codificación indicados por los campos *CodedCharSetId Encoding* y de la estructura de cabecera que precede a MQWIH, o por los campos de la estructura MQMD si MQWIH se encuentra al principio de los datos del mensaje de la aplicación.

El juego de caracteres debe ser uno que tenga caracteres de un solo byte para los caracteres que son válidos en los nombres de las colas.

Utilización

Para cualquier IBM MQ plataforma compatible, puede crear y transmitir un mensaje que incluya la estructura MQWIH, pero solo un IBM MQ for z/OS gestor de colas puede interactuar con WLM. Por lo tanto, para que el mensaje llegue a WLM desde un gestor de colas externo, la red de gestores de colas debe incluir al menos un z/OS gestor de colas a través del cual se pueda enrutar el mensaje.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 569. Campos en MQWIH		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
StrucId (identificador de estructura)	MQWIH_STRUC_ID	'WIH~'
Versión (número de versión de la estructura)	MQWIH_VERSION_1	1
StrucLength (longitud de la estructura MQWIH)	MQWIH_LENGTH_1	120
Codificación (codificación numérica de los datos que siguen a MQWIH)	Ninguna	0
CodedCharSetId (identificador del conjunto de caracteres de los datos que siguen a MQWIH)	MQCCSI_INDETERMINADO	0
Formato (nombre del formato de los datos que siguen a MQWIH)	MQFMT_NONE	Espacios en blanco
Banderas (banderas)	MQWIH_NINGUNO	0

Tabla 569. Campos en MQWIH (continuación)

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>ServiceName</u> (nombre del servicio)	Ninguna	Espacios en blanco
<u>ServiceStep</u> (nombre del paso del servicio)	Ninguna	Espacios en blanco
<u>MsgToken</u> (token de mensaje)	MQMTOK_NONE	Nulos
<u>Reservado</u> (reservado)	Ninguna	Espacios en blanco

Notas:

1. El símbolo ~ representa un único carácter en blanco.
2. En el lenguaje de programación C, la variable macroMQWIH_DEFAULT contiene los valores que se enumeran en la tabla. Úselo de la siguiente manera para proporcionar valores iniciales para los campos de la estructura:

```
MQWIH MyWIH = {MQWIH_DEFAULT};
```

Declaraciones lingüísticas

Declaración C para MQWIH

```
typedef struct tagMQWIH MQWIH;
struct tagMQWIH {
    MQCHAR4    StrucId;        /* Structure identifier */
    MQLONG     Version;        /* Structure version number */
    MQLONG     StrucLength;    /* Length of MQWIH structure */
    MQLONG     Encoding;       /* Numeric encoding of data that follows
                               MQWIH */
    MQLONG     CodedCharSetId; /* Character-set identifier of data that
                               follows MQWIH */
    MQCHAR8     Format;         /* Format name of data that follows
                               MQWIH */
    MQLONG     Flags;          /* Flags */
    MQCHAR32    ServiceName;    /* Service name */
    MQCHAR8     ServiceStep;    /* Service step name */
    MQBYTE16    MsgToken;       /* Message token */
    MQCHAR32    Reserved;       /* Reserved */
};
```

COBOL declaración para MQWIH

```
**  MQWIH structure
10 MQWIH.
**    Structure identifier
15 MQWIH-STRUCID      PIC X(4).
**    Structure version number
15 MQWIH-VERSION      PIC S9(9) BINARY.
**    Length of MQWIH structure
15 MQWIH-STRUCLength  PIC S9(9) BINARY.
**    Numeric encoding of data that follows MQWIH
15 MQWIH-ENCODING     PIC S9(9) BINARY.
**    Character-set identifier of data that follows MQWIH
15 MQWIH-CODEDCHARSETID PIC S9(9) BINARY.
**    Format name of data that follows MQWIH
15 MQWIH-FORMAT       PIC X(8).
**    Flags
15 MQWIH-FLAGS        PIC S9(9) BINARY.
**    Service name
15 MQWIH-SERVICENAME  PIC X(32).
**    Service step name
15 MQWIH-SERVICESTEP  PIC X(8).
**    Message token
15 MQWIH-MSGTOKEN     PIC X(16).
**    Reserved
15 MQWIH-RESERVED     PIC X(32).
```

PL/I declaración para MQWIH

```
dcl
  1 MQWIH based,
  3 StrucId          char(4),          /* Structure identifier */
  3 Version          fixed bin(31), /* Structure version number */
  3 StrucLength      fixed bin(31), /* Length of MQWIH structure */
  3 Encoding         fixed bin(31), /* Numeric encoding of data that
                                     follows MQWIH */
  3 CodedCharSetId   fixed bin(31), /* Character-set identifier of data
                                     that follows MQWIH */
  3 Format           char(8),          /* Format name of data that follows
                                     MQWIH */
  3 Flags            fixed bin(31), /* Flags */
  3 ServiceName      char(32),        /* Service name */
  3 ServiceStep      char(8),        /* Service step name */
  3 MsgToken         char(16),        /* Message token */
  3 Reserved         char(32);        /* Reserved */
```

High Level Assembler declaración para MQWIH

```
MQWIH          DSECT
MQWIH_STRUCID   DS    CL4    Structure identifier
MQWIH_VERSION   DS    F      Structure version number
MQWIH_STRUCLNGTH DS    F      Length of MQWIH structure
MQWIH_ENCODING  DS    F      Numeric encoding of data that follows
*
MQWIH_CODEDCHARSETID DS    F  Character-set identifier of data that
*                               follows MQWIH
MQWIH_FORMAT    DS    CL8    Format name of data that follows MQWIH
MQWIH_FLAGS     DS    F      Flags
MQWIH_SERVICENAME DS    CL32  Service name
MQWIH_SERVICESTEP DS    CL8   Service step name
MQWIH_MSGTOKEN  DS    XL16   Message token
MQWIH_RESERVED  DS    CL32   Reserved
*
MQWIH_LENGTH    EQU    *-MQWIH
                ORG    MQWIH
MQWIH_AREA      DS      CL(MQWIH_LENGTH)
```

Declaración de Visual Basic para MQWIH

```
Type MQWIH
  StrucId      As String*4 'Structure identifier'
  Version      As Long     'Structure version number'
  StrucLength  As Long     'Length of MQWIH structure'
  Encoding     As Long     'Numeric encoding of data that follows'
                  'MQWIH'
  CodedCharSetId As Long   'Character-set identifier of data that'
                  'follows MQWIH'
  Format       As String*8 'Format name of data that follows MQWIH'
  Flags        As Long     'Flags'
  ServiceName  As String*32 'Service name'
  ServiceStep  As String*8  'Service step name'
  MsgToken     As MQBYTE16 'Message token'
  Reserved     As String*32 'Reserved'
End Type
```

Detalles de campo para MQWIH

Si un mensaje va a ser procesado por el z/OS gestor de carga de trabajo (WLM), el mensaje debe comenzar con una estructura de encabezado de información de trabajo MQWIH. Esta estructura describe la información que debe estar presente al inicio de un mensaje que va a ser gestionado por WLM.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 570. Campos en MQWIH

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>StrucId</u> (identificador de estructura)	MQWIH_STRUC_ID	'WIH↵'
<u>Versión</u> (número de versión de la estructura)	MQWIH_VERSION_1	1
<u>StrucLength</u> (longitud de la estructura MQWIH)	MQWIH_LENGTH_1	120
<u>Codificación</u> (codificación numérica de los datos que siguen a MQWIH)	Ninguna	0
<u>CodedCharSetId</u> (identificador del conjunto de caracteres de los datos que siguen a MQWIH)	MQCCSI_INDETERMINADO	0
<u>Formato</u> (nombre del formato de los datos que siguen a MQWIH)	MQFMT_NONE	Espacios en blanco
<u>Banderas</u> (banderas)	MQWIH_NINGUNO	0
<u>ServiceName</u> (nombre del servicio)	Ninguna	Espacios en blanco
<u>ServiceStep</u> (nombre del paso del servicio)	Ninguna	Espacios en blanco
<u>MsgToken</u> (token de mensaje)	MQMTOK_NONE	Nulos
<u>Reservado</u> (reservado)	Ninguna	Espacios en blanco
Notas: 1. El símbolo ↵ representa un único carácter en blanco. 2. En el lenguaje de programación C, la variable macroMQWIH_DEFAULT contiene los valores que se enumeran en la tabla. Úselo de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQWIH MyWIH = {MQWIH_DEFAULT};</pre>		

StrucId (MQCHAR4)

Este es el identificador de estructura de la estructura de encabezado de información del trabajo. Siempre es un campo de entrada. Su valor es MQWIH_STRUC_ID.

El valor debe ser:

MQWIH_STRUC_ID

Identificador de la estructura del encabezado de la información del trabajo.

Para el lenguaje de programación C, también se define la constante MQWIH_STRUC_ID_ARRAY. Tiene el mismo valor que MQWIH_STRUC_ID, pero es una matriz de caracteres en lugar de una cadena.

Versión (MQLONG)

Este es el número de versión de la estructura. El valor debe ser:

MQWIH_VERSION_1

Version-1 Estructura del encabezado de la información del trabajo.

La siguiente constante especifica el número de versión de la versión actual:

MQWIH_VERSIÓN_ACTUAL

Versión actual de la estructura del encabezado de la información del trabajo.

El valor inicial de este campo es MQWIH_VERSION_1.

StrucLength (MQLONG)

Esta es la longitud de la estructura MQWIH. El valor debe ser:

MQWIH_LENGTH_1

Longitud de la estructura del encabezado de información de trabajo de version-1.

La siguiente constante especifica la longitud de la versión actual:

MQWIH_LONGITUD_ACTUAL

Longitud de la estructura actual del encabezado de la información del trabajo.

El valor inicial de este campo es MQWIH_LENGTH_1.

Encoding (MQLONG)

Esto especifica la codificación numérica de los datos que siguen a la estructura MQWIH; no se aplica a los datos numéricos de la propia estructura MQWIH.

En la llamada MQPUT o MQPUT1, la aplicación debe establecer este campo con el valor adecuado para los datos.

El valor inicial de este campo es 0.

CodedCharSetId (MQLONG)

Especifica el identificador del conjunto de caracteres de los datos que siguen a la estructura MQWIH; no se aplica a los datos de caracteres de la propia estructura MQWIH.

En la llamada MQPUT o MQPUT1, la aplicación debe establecer este campo con el valor adecuado para los datos. Puede utilizar el siguiente valor especial:

MQCCSI_HEREDAR

Los datos de caracteres en los datos *que siguen* esta estructura están en el mismo conjunto de caracteres que esta estructura.

El gestor de colas cambia este valor en la estructura enviada en el mensaje al identificador real del juego de caracteres de la estructura. Si no se produce ningún error, la llamada MQGET no devuelve el valor MQCCSI_INHERIT.

MQCCSI_INHERIT no se puede utilizar si el valor del *PutApplType* campo en MQMD es MQAT_BROKER.

El valor inicial de este campo es MQCCSI_UNDEFINED.

Format (MQCHAR8)

Especifica el nombre del formato de los datos que siguen a la estructura MQWIH.

En la llamada MQPUT o MQPUT1, la aplicación debe establecer este campo con el valor adecuado para los datos. Las reglas para codificar este campo son las mismas que para el *Format* campo en MQMD.

La longitud de este campo viene dada por MQ_FORMAT_LENGTH. El valor inicial de este campo es MQFMT_NONE.

Flags (MQLONG)

El valor debe ser:

MQWIH_NINGUNO

Sin banderas.

El valor inicial de este campo es MQWIH_NONE.

ServiceName (MQCHAR32)

Este es el nombre del servicio que procesará el mensaje.

La longitud de este campo viene dada por MQ_SERVICE_NAME_LENGTH. El valor inicial de este campo es 32 caracteres en blanco.

ServiceStep (MQCHAR8)

Este es el nombre del paso al *ServiceName* que se refiere el mensaje.

La longitud de este campo viene dada por MQ_SERVICE_STEP_LENGTH. El valor inicial de este campo es 8 caracteres en blanco.

MsgToken (MQBYTE16)

Se trata de un token de mensaje que identifica de forma única el mensaje.

Para las llamadas MQPUT y MQPUT1, este campo se ignora. La longitud de este campo viene dada por MQ_MSG_TOKEN_LENGTH. El valor inicial de este campo es MQMTOK_NONE.

Reservado (MQCHAR32)

Este es un campo reservado; debe dejarse en blanco.

MQXP - Bloque de parámetros de salida

La estructura MQXP se utiliza como parámetro de entrada/salida para la salida de cruce de API. Para obtener más información sobre esta salida, consulte [La salida que cruza la API](#).

Conjunto de caracteres y codificación

Los datos de caracteres en MQXP están en el juego de caracteres del gestor de colas local; esto viene dado por el atributo del **CodedCharSetId** gestor de colas. Los datos numéricos en MQXP están en la codificación nativa de la máquina; esto viene dado por MQENC_NATIVE.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 571. Campos en MQXP	
Nombre y descripción del campo	Nombre de la constante
<u>StrucId</u> (identificador de estructura)	MQXP_STRUC_ID
<u>Versión</u> (número de versión de la estructura)	MQXP_VERSION_1
<u>ExitId</u> (identificador de salida)	MQXT_API_CROSSING_EXIT
<u>ExitReason</u> (motivo de la invocación de la salida)	Ninguna
<u>ExitResponse</u> (respuesta desde la salida)	Ninguna
<u>ExitCommand</u> (Código de llamada API)	Ninguna
<u>ExitParmCount</u> (número de parámetros)	Ninguna
<u>Reservado</u> (reservado)	Ninguna

Tabla 571. Campos en MQXP (continuación)

Nombre y descripción del campo	Nombre de la constante
ExitUserArea (área de usuario)	Ninguna

Declaraciones lingüísticas

Declaración C para MQXP

```
typedef struct tagMQXP MQXP;
struct tagMQXP {
    MQCHAR4    StrucId;        /* Structure identifier */
    MQLONG     Version;        /* Structure version number */
    MQLONG     ExitId;         /* Exit identifier */
    MQLONG     ExitReason;     /* Reason for invocation of exit */
    MQLONG     ExitResponse;   /* Response from exit */
    MQLONG     ExitCommand;    /* API call code */
    MQLONG     ExitParmCount;  /* Parameter count */
    MQLONG     Reserved;       /* Reserved */
    MQBYTE16   ExitUserArea;   /* User area */
};
```

COBOL declaración para MQXP

```
**  MQXP structure
10 MQXP.
**  Structure identifier
15 MQXP-STRUCID      PIC X(4).
**  Structure version number
15 MQXP-VERSION     PIC S9(9) BINARY.
**  Exit identifier
15 MQXP-EXITID      PIC S9(9) BINARY.
**  Reason for invocation of exit
15 MQXP-EXITREASON  PIC S9(9) BINARY.
**  Response from exit
15 MQXP-EXITRESPONSE PIC S9(9) BINARY.
**  API call code
15 MQXP-EXITCOMMAND PIC S9(9) BINARY.
**  Parameter count
15 MQXP-EXITPARMCOUNT PIC S9(9) BINARY.
**  Reserved
15 MQXP-RESERVED    PIC S9(9) BINARY.
**  User area
15 MQXP-EXITUSERAREA PIC X(16).
```

PL/I declaración para MQXP

```
dcl
1 MQXP based,
3 StrucId      char(4),        /* Structure identifier */
3 Version      fixed bin(31), /* Structure version number */
3 ExitId       fixed bin(31), /* Exit identifier */
3 ExitReason    fixed bin(31), /* Reason for invocation of exit */
3 ExitResponse  fixed bin(31), /* Response from exit */
3 ExitCommand   fixed bin(31), /* API call code */
3 ExitParmCount fixed bin(31), /* Parameter count */
3 Reserved      fixed bin(31), /* Reserved */
3 ExitUserArea  char(16);     /* User area */
```

High Level Assembler declaración para MQXP

```
MQXP          DSECT
MQXP_STRUCID  DS    CL4    Structure identifier
MQXP_VERSION  DS    F      Structure version number
MQXP_EXITID   DS    F      Exit identifier
MQXP_EXITREASON DS    F    Reason for invocation of exit
MQXP_EXITRESPONSE DS    F  Response from exit
MQXP_EXITCOMMAND DS    F   API call code
MQXP_EXITPARMCOUNT DS    F  Parameter count
MQXP_RESERVED DS    F      Reserved
MQXP_EXITUSERAREA DS    XL16 User area
*
MQXP_LENGTH   EQU    *-MQXP
```

Detalles de campo para MQXP

La estructura del bloque de parámetros de salida MQXP se utiliza como parámetro de entrada/salida para la salida de cruce API.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 572. Campos en MQXP	
Nombre y descripción del campo	Nombre de la constante
<u>StrucId</u> (identificador de estructura)	MQXP_STRUC_ID
<u>Versión</u> (número de versión de la estructura)	MQXP_VERSION_1
<u>ExitId</u> (identificador de salida)	MQXT_API_CROSSING_EXIT
<u>ExitReason</u> (motivo de la invocación de la salida)	Ninguna
<u>ExitResponse</u> (respuesta desde la salida)	Ninguna
<u>ExitCommand</u> (Código de llamada API)	Ninguna
<u>ExitParmCount</u> (número de parámetros)	Ninguna
<u>Reservado</u> (reservado)	Ninguna
<u>ExitUserArea</u> (área de usuario)	Ninguna

StrucId (MQCHAR4)

Este es el identificador de estructura de la estructura del parámetro de salida. Siempre es un campo de entrada. Su valor es MQXP_STRUC_ID.

El valor debe ser:

MQXP_STRUC_ID

Identificador de la estructura del parámetro de salida.

Para el lenguaje de programación C, también se define la constante MQXP_STRUC_ID_ARRAY. Tiene el mismo valor que MQXP_STRUC_ID, pero es una matriz de caracteres en lugar de una cadena.

Versión (MQLONG)

Este es el número de versión de la estructura. El valor debe ser:

MQXP_VERSION_1

Número de versión para la estructura del bloque de parámetros de salida.

Nota: Cuando se introduce una nueva versión de esta estructura, no se modifica el diseño de la parte existente. Por lo tanto, la salida debe comprobar que el número de versión sea igual o superior a la versión más baja que contenga los campos que la salida necesita utilizar.

Este es un campo de entrada a la salida.

ExitId (MQLONG)

Esto se establece al entrar en la rutina de salida e indica el tipo de salida:

MQXT_API_CROSSING_EXIT

Salida con cruce de API para CICS.

Este es un campo de entrada a la salida.

ExitReason (MQLONG)

Esto se establece al entrar en la rutina de salida. Para la salida de cruce de API, indica si la rutina se llama antes o después de la ejecución de la llamada a la API:

MQXR_ANTES

Antes de la ejecución de la API.

MQXR_DESPUÉS

Después de la ejecución de la API.

Este es un campo de entrada a la salida.

ExitResponse (MQLONG)

El valor lo establece la salida para comunicarse con la persona que llama. Los valores siguientes están definidos:

MQXCC_OK

Salida completada con éxito.

MQXCC_SUPPRIMIR_FUNCIÓN

Función de supresión.

Cuando este valor se establece mediante una salida que cruza la API llamada *antes de* la llamada a la API, la llamada a la API no se realiza. El *CompCode* para la llamada se establece en MQCC_FAILED, el *Reason* se establece en MQRC_SUPPRESSED_BY_EXIT y todos los demás parámetros permanecen tal y como los dejó la salida.

Cuando este valor es establecido por una salida que cruza la API llamada *después de* la llamada a la API, el gestor de colas lo ignora.

MQXCC_OMITIR_FUNCIÓN

Función de salto.

Cuando este valor se establece mediante una salida que cruza la API llamada *antes de* la llamada a la API, la llamada a la API no se realiza; los parámetros *CompCode* y *Reason* y todos los demás parámetros permanecen tal y como los dejó la salida.

Cuando este valor es establecido por una salida que cruza la API llamada *después de* la llamada a la API, el gestor de colas lo ignora.

Este es un campo de salida de la salida.

ExitCommand (MQLONG)

Este campo se establece al entrar en la rutina de salida. Identifica la llamada a la API que provocó la invocación de la salida:

MQXC_CALLBACK

La llamada CALLBACK.

MQXC_MQBACK

La llamada MQBACK.

MQXC_MQCB

La llamada de MQCB.

MQXC_MQCLOSE

La llamada MQCLOSE.

MQXC_MQCMIT

La llamada MQCMIT.

MQXC_MQCTL

La llamada MQCTL.

MQXC_MQGET

La llamada MQGET.

MQXC_MQINQ

La llamada MQINQ.

MQXC_MQOPEN

La llamada MQOPEN.

MQXC_MQPUT

La llamada MQPUT.

MQXC_MQPUT1

La llamada « MQPUT1 ».

MQXC_MQSET

La llamada MQSET.

MQXC_MQSTAT

La llamada MQSTAT.

MQXC_MQSUB

La llamada MQSUB.

MQXC_MQSUBRQ

La llamada MQSUBRQ.

Este es un campo de entrada a la salida.

ExitParmCount (MQLONG)

Este campo se establece al entrar en la rutina de salida. Contiene el número de parámetros que toma la llamada MQ.

Tabla 573. Número de parámetros para cada llamada MQ

Nombre de llamada	Número de parámetros
MQBACK	3
MQCLOSE	5
MQCMIT	3
MQGET	9
MQINQ	10
MQOPEN	6
MQPUT	8
MQPUT1	8
MQSET	10

Este es un campo de entrada a la salida.

Reservado (MQLONG)

Este es un campo reservado. Su valor no es significativo para la salida.

ExitUserArea (MQBYTE16)

Este es un campo que está disponible para que lo utilice la salida. Se inicializa con un cero binario para la longitud del campo antes de la primera invocación de la salida para la tarea y, a partir de entonces, cualquier cambio realizado en este campo por la salida se conserva en todas las invocaciones de la salida. Se define el siguiente valor:

MQXUA_NONE

Sin información sobre el usuario.

El valor es un cero binario para la longitud del campo.

Para el lenguaje de programación C, también se define la constante MQXUA_NONE_ARRAY, que tiene el mismo valor que MQXUA_NONE, pero es una matriz de caracteres en lugar de una cadena.

La longitud de este campo viene dada por MQ_EXIT_USER_AREA_LENGTH. Este es un campo de entrada/salida hacia la salida.

MQXQH - Cabecera de cola de transmisión

La estructura MQXQH describe la información que se antepone a los datos del mensaje de aplicación de los mensajes cuando se encuentran en colas de transmisión. Una cola de transmisión es un tipo especial de cola local que almacena temporalmente mensajes destinados a colas remotas (es decir, destinados a colas que no pertenecen al gestor de colas local). Una cola de transmisión se indica mediante el **Usage** atributo queue con el valor MQUS_TRANSMISSION.

Nombre de formato

MQFMT_XMIT_Q_HEADER

Conjunto de caracteres y codificación

Los datos en MQXQH deben estar en el juego de caracteres proporcionado por el atributo del **CodedCharSetId** gestor de colas y la codificación del gestor de colas local proporcionada por MQENC_NATIVE.

Establezca el juego de caracteres y la codificación del MQXQH en los campos *CodedCharSetId* *Encoding* y en:

- El MQMD independiente (si la estructura MQXQH se encuentra al principio de los datos del mensaje), o
- La estructura de encabezado que precede a la estructura MQXQH (en todos los demás casos).

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 574. Campos en MQXQH para MQXQH		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>StrucId</u> (identificador de estructura)	MQXQH_STRUC_ID	'XQH→'
<u>Versión</u> (número de versión de la estructura)	MQXQH_VERSION_1	1
<u>RemoteQName</u> (nombre de la cola de destino)	Ninguna	Cadena nula o espacios en blanco
<u>RemoteQMgrName</u> (nombre del gestor de colas de destino)	Ninguna	Cadena nula o espacios en blanco

Tabla 574. Campos en MQXQH para MQXQH (continuación)

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>MsgDesc</u> (descriptor del mensaje original)	Mismos nombres y valores que MQMD; véase Tabla 518 en la página 395	-
Notas: 1. El símbolo ~ representa un único carácter en blanco. 2. El valor «Cadena nula o espacios en blanco» denota la cadena nula en C y los caracteres en blanco en otros lenguajes de programación. 3. En el lenguaje de programación C, la variable macroMQXQH_DEFAULT contiene los valores que se enumeran en la tabla. Úselo de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQXQH MyXQH = {MQXQH_DEFAULT};</pre>		

Declaraciones lingüísticas

Declaración C para MQXQH

```
typedef struct tagMQXQH MQXQH;
struct tagMQXQH {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQCHAR48    RemoteQName;     /* Name of destination queue */
    MQCHAR48    RemoteQMGrName;  /* Name of destination queue manager */
    MQMD1      MsgDesc;         /* Original message descriptor */
};
```

COBOL declaración para MQXQH

```
** MQXQH structure
10 MQXQH.
** Structure identifier
15 MQXQH-STRUCID PIC X(4).
** Structure version number
15 MQXQH-VERSION PIC S9(9) BINARY.
** Name of destination queue
15 MQXQH-REMOTEQNAME PIC X(48).
** Name of destination queue manager
15 MQXQH-REMOTEQMGRNAME PIC X(48).
** Original message descriptor
15 MQXQH-MSGDESC.
** Structure identifier
20 MQXQH-MSGDESC-STRUCID PIC X(4).
** Structure version number
20 MQXQH-MSGDESC-VERSION PIC S9(9) BINARY.
** Report options
20 MQXQH-MSGDESC-REPORT PIC S9(9) BINARY.
** Message type
20 MQXQH-MSGDESC-MSGTYPE PIC S9(9) BINARY.
** Expiry time
20 MQXQH-MSGDESC-EXPIRY PIC S9(9) BINARY.
** Feedback or reason code
20 MQXQH-MSGDESC-FEEDBACK PIC S9(9) BINARY.
** Numeric encoding of message data
20 MQXQH-MSGDESC-ENCODING PIC S9(9) BINARY.
** Character set identifier of message data
20 MQXQH-MSGDESC-CODEDCHARSETID PIC S9(9) BINARY.
** Format name of message data
20 MQXQH-MSGDESC-FORMAT PIC X(8).
** Message priority
20 MQXQH-MSGDESC-PRIORITY PIC S9(9) BINARY.
** Message persistence
20 MQXQH-MSGDESC-PERSISTENCE PIC S9(9) BINARY.
```

```

**      Message identifier
20 MQXQH-MSGDESC-MSGID      PIC X(24).
**      Correlation identifier
20 MQXQH-MSGDESC-CORRELID   PIC X(24).
**      Backout counter
20 MQXQH-MSGDESC-BACKOUTCOUNT PIC S9(9) BINARY.
**      Name of reply-to queue
20 MQXQH-MSGDESC-REPLYTOQ    PIC X(48).
**      Name of reply queue manager
20 MQXQH-MSGDESC-REPLYTOQMGR PIC X(48).
**      User identifier
20 MQXQH-MSGDESC-USERIDENTIFIER PIC X(12).
**      Accounting token
20 MQXQH-MSGDESC-ACCOUNTINGTOKEN PIC X(32).
**      Application data relating to identity
20 MQXQH-MSGDESC-APPLIDENTITYDATA PIC X(32).
**      Type of application that put the message
20 MQXQH-MSGDESC-PUTAPPLTYPE  PIC S9(9) BINARY.
**      Name of application that put the message
20 MQXQH-MSGDESC-PUTAPPLNAME  PIC X(28).
**      Date when message was put
20 MQXQH-MSGDESC-PUTDATE      PIC X(8).
**      Time when message was put
20 MQXQH-MSGDESC-PUTTIME      PIC X(8).
**      Application data relating to origin
20 MQXQH-MSGDESC-APPLORIGINDATA PIC X(4).

```

PL/I declaración para MQXQH

```

dcl
  1 MQXQH based,
    3 StrucId      char(4),          /* Structure identifier */
    3 Version      fixed bin(31),    /* Structure version number */
    3 RemoteQName  char(48),         /* Name of destination queue */
    3 RemoteQMgrName char(48),       /* Name of destination queue
                                     manager */
    3 MsgDesc,
      5 StrucId      char(4),          /* Original message descriptor */
      5 Version      fixed bin(31),    /* Structure identifier */
      5 Version      fixed bin(31),    /* Structure version number */
      5 Report       fixed bin(31),    /* Report options */
      5 MsgType      fixed bin(31),    /* Message type */
      5 Expiry       fixed bin(31),    /* Expiry time */
      5 Feedback     fixed bin(31),    /* Feedback or reason code */
      5 Encoding     fixed bin(31),    /* Numeric encoding of message
                                     data */
      5 CodedCharSetId fixed bin(31), /* Character set identifier of
                                     message data */
      5 Format        char(8),          /* Format name of message data */
      5 Priority      fixed bin(31),    /* Message priority */
      5 Persistence  fixed bin(31),    /* Message persistence */
      5 MsgId        char(24),         /* Message identifier */
      5 CorrelId     char(24),         /* Correlation identifier */
      5 BackoutCount fixed bin(31),    /* Backout counter */
      5 ReplyToQ     char(48),         /* Name of reply-to queue */
      5 ReplyToQMgr  char(48),         /* Name of reply queue manager */
      5 UserIdentifier char(12),       /* User identifier */
      5 AccountingToken char(32),      /* Accounting token */
      5 ApplIdentityData char(32),     /* Application data relating to
                                     identity */
      5 PutApplType  fixed bin(31),    /* Type of application that put the
                                     message */
      5 PutApplName  char(28),         /* Name of application that put the
                                     message */
      5 PutDate      char(8),          /* Date when message was put */
      5 PutTime      char(8),          /* Time when message was put */
      5 ApplOriginData char(4);        /* Application data relating to
                                     origin */

```

High Level Assembler declaración para MQXQH

MQXQH	DSECT	
MQXQH_STRUCID	DS	CL4 Structure identifier
MQXQH_VERSION	DS	F Structure version number
MQXQH_REMOTEQNAME	DS	CL48 Name of destination queue
MQXQH_REMOTEQMGRNAME	DS	CL48 Name of destination queue manager
* MQXQH_MSGDESC	DS	0F Force fullword alignment
MQXQH_MSGDESC_STRUCID	DS	CL4 Structure identifier
MQXQH_MSGDESC_VERSION	DS	F Structure version number

MQXQH_MSGDESC_REPORT	DS	F	Report options
MQXQH_MSGDESC_MSGTYPE	DS	F	Message type
MQXQH_MSGDESC_EXPIRY	DS	F	Expiry time
MQXQH_MSGDESC_FEEDBACK	DS	F	Feedback or reason code
MQXQH_MSGDESC_ENCODING	DS	F	Numeric encoding of message data
*			
MQXQH_MSGDESC_CODEDCHARSETID	DS	F	Character set identifier of message data
*			
MQXQH_MSGDESC_FORMAT	DS	CL8	Format name of message data
MQXQH_MSGDESC_PRIORITY	DS	F	Message priority
MQXQH_MSGDESC_PERSISTENCE	DS	F	Message persistence
MQXQH_MSGDESC_MSGID	DS	XL24	Message identifier
MQXQH_MSGDESC_CORRELID	DS	XL24	Correlation identifier
MQXQH_MSGDESC_BACKOUTCOUNT	DS	F	Backout counter
MQXQH_MSGDESC_REPLYTOQ	DS	CL48	Name of reply-to queue
MQXQH_MSGDESC_REPLYTOQMGR	DS	CL48	Name of reply queue manager
MQXQH_MSGDESC_USERIDENTIFIER	DS	CL12	User identifier
MQXQH_MSGDESC_ACCOUNTINGTOKEN	DS	XL32	Accounting token
MQXQH_MSGDESC_APPLIDENTITYDATA	DS	CL32	Application data relating to identity
*			
MQXQH_MSGDESC_PUTAPPLTYPE	DS	F	Type of application that put the message
*			
MQXQH_MSGDESC_PUTAPPLNAME	DS	CL28	Name of application that put the message
*			
MQXQH_MSGDESC_PUTDATE	DS	CL8	Date when message was put
MQXQH_MSGDESC_PUTTIME	DS	CL8	Time when message was put
MQXQH_MSGDESC_APPLORIGINDATA	DS	CL4	Application data relating to origin
*			
MQXQH_MSGDESC_LENGTH	EQU	*-MQXQH_MSGDESC	
	ORG	MQXQH_MSGDESC	
MQXQH_MSGDESC_AREA	DS	CL(MQXQH_MSGDESC_LENGTH)	
*			
MQXQH_LENGTH	EQU	*-MQXQH	
	ORG	MQXQH	
MQXQH_AREA	DS	CL(MQXQH_LENGTH)	

Declaración de Visual Basic para MQXQH

```

Type MQXQH
    StructId      As String*4 'Structure identifier'
    Version       As Long     'Structure version number'
    RemoteQName   As String*48 'Name of destination queue'
    RemoteQMGrName As String*48 'Name of destination queue manager'
    MsgDesc       As MQMD1    'Original message descriptor'
End Type

```

Campos en el descriptor de mensaje independiente

Un mensaje que se encuentra en una cola de transmisión tiene *dos* descriptors de mensaje:

- Un descriptor de mensaje se almacena por separado de los datos del mensaje; se denomina *descriptor de mensaje independiente* y lo genera el gestor de colas cuando el mensaje se coloca en la cola de transmisión. Algunos de los campos del descriptor de mensajes independiente se copian del descriptor de mensajes proporcionado por la aplicación en la llamada MQPUT o MQPUT1.

El descriptor de mensaje independiente es el que se devuelve a la aplicación en el **MsgDesc** parámetro de la llamada MQGET cuando el mensaje se elimina de la cola de transmisión.

- Se almacena un segundo descriptor de mensaje dentro de la estructura MQXQH como parte de los datos del mensaje; este se denomina *descriptor de mensaje incrustado* y es una copia del descriptor de mensaje proporcionado por la aplicación en la llamada MQPUT o MQPUT1 (con pequeñas variaciones).

El descriptor de mensaje incrustado es siempre un MQMD (version-1). Si el mensaje enviado por la aplicación tiene valores no predeterminados para uno o más de los campos version-2 en el MQMD, una estructura MQMDE sigue al MQXQH y, a su vez, es seguida por los datos del mensaje de la aplicación (si los hay). El MQMDE es:

- Generado por el gestor de colas (si la aplicación utiliza un MQMD version-2 para colocar el mensaje), o
- Ya presente al inicio de los datos del mensaje de la aplicación (si la aplicación utiliza un MQMD version-1 para enviar el mensaje).

El descriptor de mensaje incrustado es el que se devuelve a la aplicación en el **MsgDesc** parámetro de la llamada MQGET cuando el mensaje se elimina de la cola de destino final.

Los campos del descriptor de mensajes separado son configurados por el gestor de colas tal y como se muestra. Si el gestor de colas no admite el MQMD « version-2 », se utiliza un MQMD « version-1 » sin pérdida de funcionalidad.

Tabla 575. Valores utilizados para los campos en el MQMD separado

Campo en MQMD separado	Valor utilizado
<i>StrucId</i>	MQMD_STRUC_ID
<i>Version</i>	MQMD_VERSION_2
<i>Report</i>	Copiado del descriptor de mensaje incrustado, pero con los bits identificados por MQRO_ACCEPT_UNSUP_IF_XMIT_MASK establecidos en cero. (Esto evita que se genere un mensaje de informe COA o COD cuando se coloca o se elimina un mensaje de una cola de transmisión)
<i>MsgType</i>	Copiado del descriptor de mensaje incrustado.
<i>Expiry</i>	Copiado del descriptor de mensaje incrustado.
<i>Feedback</i>	Copiado del descriptor de mensaje incrustado.
<i>Encoding</i>	MQENC_NATIVE (véase la nota)
<i>CodedCharSetId</i>	Atributo CodedCharSetId del gestor de colas.
<i>Format</i>	MQFMT_XMIT_Q_HEADER
<i>Priority</i>	Copiado del descriptor de mensaje incrustado.
<i>Persistence</i>	Copiado del descriptor de mensaje incrustado.
<i>MsgId</i>	El gestor de colas genera un nuevo valor. Este identificador de mensaje es diferente del <i>MsgId</i> que el gestor de colas puede haber generado para el descriptor de mensaje incrustado descrito anteriormente.
<i>CorrelId</i>	El <i>MsgId</i> del descriptor de mensaje incrustado. Los mensajes enviados a SYSTEM.CLUSTER.TRANSMIT.QUEUE están reservados para <i>CorrelId</i> uso interno.
<i>BackoutCount</i>	0
<i>ReplyToQ</i>	Copiado del descriptor de mensaje incrustado.
<i>ReplyToQMGr</i>	Copiado del descriptor de mensaje incrustado.
<i>UserIdentifier</i>	Copiado del descriptor de mensaje incrustado.
<i>AccountingToken</i>	Copiado del descriptor de mensaje incrustado. Los mensajes enviados a SYSTEM.CLUSTER.TRANSMIT.QUEUE están reservados para <i>AccountingToken</i> uso interno.
<i>ApplIdentityData</i>	Copiado del descriptor de mensaje incrustado.
<i>PutApplType</i>	MQAT_QMGR
<i>PutApplName</i>	Los primeros 28 bytes del nombre del gestor de colas.
<i>PutDate</i>	Fecha en la que el mensaje se colocó en la cola de transmisión.
<i>PutTime</i>	Hora en la que el mensaje se colocó en la cola de transmisión.
<i>ApplOriginData</i>	Espacios en blanco
<i>GroupId</i>	MQGI_NONE
<i>MsgSeqNumber</i>	1

Tabla 575. Valores utilizados para los campos en el MQMD separado (continuación)

Campo en MQMD separado	Valor utilizado
<i>Offset</i>	0
<i>MsgFlags</i>	MQMF_NINGUNO
<i>OriginalLength</i>	MQOL_INDETERMINADO

- En Windows, el valor de MQENC_NATIVE para Micro Focus COBOL difiere del valor para C. El valor del *Encoding* campo en el descriptor de mensaje independiente es siempre el valor de C en estos entornos; este valor es 546 en decimal. Además, los campos enteros de la estructura MQXQH están en la codificación que corresponde a este valor (la codificación Intel nativa).

Campos del descriptor de mensaje incrustado

Los campos del descriptor de mensaje incrustado tienen los mismos valores que los del **MsgDesc** parámetro de la llamada MQPUT o MQPUT1, excepto en los siguientes casos:

- El *Version* campo siempre tiene el valor « MQMD_VERSION_1 ».
- Si el *Priority* campo tiene el valor MQPRI_PRIORITY_AS_Q_DEF, se sustituye por el valor del atributo **DefPriority** de la cola.
- Si el *Persistence* campo tiene el valor MQPER_PERSISTENCE_AS_Q_DEF, se sustituye por el valor del atributo **DefPersistence** de la cola.
- Si el *MsgId* campo tiene el valor MQMI_NONE, o se ha especificado la opción MQPMO_NEW_MSG_ID, o el mensaje es un mensaje de lista de distribución, *MsgId* se sustituye por un nuevo identificador de mensaje generado por el gestor de colas.

Cuando un mensaje de lista de distribución se divide en mensajes de lista de distribución más pequeños colocados en diferentes colas de transmisión, el *MsgId* campo de cada uno de los nuevos descriptores de mensajes incrustados es el mismo que el del mensaje de lista de distribución original.

- Si se especificó la opción MQPMO_NEW_CORREL_ID, *CorrelId* se sustituye por un nuevo identificador de correlación generado por el gestor de colas.
- Los campos de contexto se establecen según lo indicado por las opciones MQPMO_*_CONTEXT especificadas en el **PutMsgOpts** parámetro; los campos de contexto son:
 - *AccountingToken*
 - *ApplIdentityData*
 - *ApplOriginData*
 - *PutApplName*
 - *PutApplType*
 - *PutDate*
 - *PutTime*
 - *UserIdentifier*
- Los campos « version-2 » (si estaban presentes) se eliminan del MQMD y se trasladan a una estructura MQMDE, si uno o varios de los campos « version-2 » tienen un valor distinto al predeterminado.

Transferir mensajes a colas remotas

Cuando una aplicación coloca un mensaje en una cola remota (ya sea especificando directamente el nombre de la cola remota o utilizando una definición local de la cola remota), el gestor de colas local:

- Crea una estructura MQXQH que contiene el descriptor de mensaje incrustado
- Añade un MQMDE si es necesario y aún no existe
- Añade los datos del mensaje de la aplicación

- Coloca el mensaje en una cola de transmisión adecuada

Colocar mensajes directamente en las colas de transmisión

Una aplicación también puede colocar un mensaje directamente en una cola de transmisión. En este caso, la aplicación debe anteponer los datos del mensaje de la aplicación con una estructura MQXQH e inicializar los campos con los valores adecuados. Además, el *Format* campo del **MsgDesc** parámetro de la llamada MQPUT o MQPUT1 debe tener el valor MQFMT_XMIT_Q_HEADER.

Los datos de caracteres de la estructura MQXQH creada por la aplicación deben estar en el juego de caracteres del gestor de colas local (definido por el **CodedCharSetId** atributo queue manager), y los datos enteros deben estar en la codificación nativa de la máquina. Además, los datos de caracteres en la estructura MQXQH deben rellenarse con espacios en blanco hasta alcanzar la longitud definida del campo; los datos no deben terminar prematuramente utilizando un carácter nulo, ya que el gestor de colas no convierte el carácter nulo y los caracteres siguientes en espacios en blanco en la estructura MQXQH.

Sin embargo, el gestor de colas no comprueba que exista una estructura MQXQH ni que se hayan especificado valores válidos para los campos.

Las aplicaciones no deben enviar sus mensajes directamente a SYSTEM.CLUSTER.TRANSMIT.QUEUE.

Obtener mensajes de las colas de transmisión

Las aplicaciones que reciben mensajes de una cola de transmisión deben procesar la información de la estructura MQXQH de forma adecuada. La presencia de la estructura MQXQH al principio de los datos del mensaje de aplicación se indica mediante el valor MQFMT_XMIT_Q_HEADER que se devuelve en el *Format* campo del **MsgDesc** parámetro de la llamada MQGET. Los valores devueltos en los campos *CodedCharSetId Encoding* y del **MsgDesc** parámetro indican el conjunto de caracteres y la codificación de los datos de caracteres e enteros en la estructura MQXQH. El conjunto de caracteres y la codificación de los datos del mensaje de la aplicación se definen mediante los campos *CodedCharSetId Encoding* y del descriptor de mensajes incrustado.

Detalles de campo para MQXQH

La estructura del encabezado de la cola de transmisión MQXQH describe la información que se antepone a los datos del mensaje de la aplicación cuando estos se encuentran en colas de transmisión.

Campos

Nota: En la siguiente tabla, los campos se agrupan por uso, en lugar de por orden alfabético.

Tabla 576. Campos en MQXQH para MQXQH		
Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
<u>StrucId</u> (identificador de estructura)	MQXQH_STRUC_ID	'XQH'
<u>Versión</u> (número de versión de la estructura)	MQXQH_VERSION_1	1
<u>RemoteQName</u> (nombre de la cola de destino)	Ninguna	Cadena nula o espacios en blanco
<u>RemoteQMGrName</u> (nombre del gestor de colas de destino)	Ninguna	Cadena nula o espacios en blanco
<u>MsgDesc</u> (descriptor del mensaje original)	Mismos nombres y valores que MQMD; véase Tabla 518 en la página 395	-

Tabla 576. Campos en MQXQH para MQXQH (continuación)

Nombre y descripción del campo	Nombre de la constante	Valor inicial (si lo hay) de la constante
Notas: 1. El símbolo ~ representa un único carácter en blanco. 2. El valor «Cadena nula o espacios en blanco» denota la cadena nula en C y los caracteres en blanco en otros lenguajes de programación. 3. En el lenguaje de programación C, la variable macro MQXQH_DEFAULT contiene los valores que se enumeran en la tabla. Úselo de la siguiente manera para proporcionar valores iniciales para los campos de la estructura: <pre>MQXQH MyXQH = {MQXQH_DEFAULT};</pre>		

StrucId (MQCHAR4)

Este es el identificador de estructura de la estructura de encabezado de la cola de transmisión. Siempre es un campo de entrada. Su valor es MQXQH_STRUC_ID.

El valor debe ser:

MQXQH_STRUC_ID

Identificador de la estructura del encabezado de la cola de transmisión.

Para el lenguaje de programación C, también se define la constante MQXQH_STRUC_ID_ARRAY. Tiene el mismo valor que MQXQH_STRUC_ID, pero es una matriz de caracteres en lugar de una cadena.

Versión (MQLONG)

Este es el número de versión de la estructura. El valor debe ser:

MQXQH_VERSION_1

Número de versión para la estructura del encabezado de la cola de transmisión.

La siguiente constante especifica el número de versión de la versión actual:

MQXQH_VERSIÓN_ACTUAL

Versión actual de la estructura del encabezado de la cola de transmisión.

El valor inicial de este campo es MQXQH_VERSION_1.

RemoteQName (MQCHAR48)

Este es el nombre de la cola de mensajes que es el destino final aparente del mensaje (esto podría no ser el destino final si, por ejemplo, esta cola se define en *RemoteQMGrName* como una definición local de otra cola remota).

Si el mensaje es un mensaje de lista de distribución (es decir, el *Format* campo del descriptor de mensaje incrustado es MQFMT_DIST_HEADER), *RemoteQName* está en blanco.

La longitud de este campo viene determinada por MQ_Q_NAME_LENGTH. El valor inicial de este campo es la cadena nula en C y 48 caracteres en blanco en otros lenguajes de programación.

RemoteQMGrName (MQCHAR48)

Este es el nombre del administrador de colas o del grupo de uso compartido de colas propietario de la cola que es el destino final aparente del mensaje.

Si el mensaje es un mensaje de lista de distribución, *RemoteQMGrName* está en blanco.

La longitud de este campo viene determinada por MQ_Q_MGR_NAME_LENGTH. El valor inicial de este campo es la cadena nula en C y 48 caracteres en blanco en otros lenguajes de programación.

MsgDesc (MQMD1)

Este es el descriptor de mensaje incrustado, y es una copia casi exacta del descriptor de mensaje MQMD que se especificó como parámetro **MsgDesc** en la llamada MQPUT o MQPUT1 cuando el mensaje se colocó originalmente en la cola remota.

Nota: Este es un MQMD de version-1.

Los valores iniciales de los campos de esta estructura son los mismos que los de la estructura MQMD.

Llamadas de funciones

Información sobre todas las llamadas de « MQI » que se pueden realizar. Se proporcionan descripciones, sintaxis, información sobre parámetros, notas de uso e invocaciones de lenguaje para cada uno de los diferentes idiomas posibles para cada una de las diferentes llamadas.

  Hay ayuda en línea disponible para estas llamadas en las Linux plataformas AIX y, en forma de man páginas.

Nota: Las llamadas asociadas con la conversión de datos, MQXCNV y MQ_DATA_CONV_EXIT, se describen en [“salida de conversión de datos” en la página 915](#).

Uso de las llamadas en el lenguaje C

Los parámetros que son solo de entrada y de tipo MQHCONN, MQHOBJ, MQHMSG o MQLONG se pasan por valor. Para todos los demás parámetros, la dirección del parámetro se pasa por valor.

No es necesario especificar todos los parámetros que se pasan por referencia cada vez que se invoca una función. Cuando no necesite un parámetro concreto, especifique un puntero nulo como parámetro en la invocación de la función, en lugar de la dirección de los datos del parámetro. Los parámetros para los cuales esto es posible se identifican en las descripciones de llamada.

No se devuelve ningún parámetro como valor de la llamada; en terminología C, esto significa que todas las llamadas devuelven void.

Declaración del parámetro Buffer

Las **MQPUT1** llamadas **MQGET**, **MQPUT**, y tienen cada una un parámetro con un tipo de datos indefinido: el **Buffer** parámetro. Utilice este parámetro para enviar y recibir los datos de los mensajes de la aplicación.

Los parámetros de este tipo se muestran en los ejemplos de C como matrices de MQBYTE. Puede declarar los parámetros de esta manera, pero normalmente es más conveniente declararlos como la estructura concreta que describe la disposición de los datos en el mensaje. El prototipo de la función declara el parámetro como un puntero a void, de modo que se puede especificar la dirección de cualquier tipo de datos como parámetro en la invocación de la llamada. El puntero a void es un puntero a datos de formato indefinido. Se define como:

```
typedef void *PMQVOID;
```

Referencia relacionada

 [Ejemplos de resultados del CEDF a partir de convocatorias de « MQI »](#)

Descripciones de llamadas

En esta sección se describen las llamadas a « MQI ».

- [“MQBACK - Deshacer cambios” en la página 629](#)
- [“MQBEGIN - Comenzar unidad de trabajo” en la página 633](#)

- [“MQBUFMH - Convertir almacenamiento intermedio en descriptor de contexto de mensaje” en la página 636](#)
- [“MQCB - Gestionar devolución de llamada” en la página 639](#)
- [“MQCB_FUNCTION - Función de devolución de llamada” en la página 649](#)
- [“MQCLOSE - Cerrar objeto” en la página 651](#)
- [“MQCMIT - Confirmar cambios” en la página 659](#)
- [“MQCONN - Conectar gestor de colas” en la página 663](#)
- [“MQCONNX - Conectar gestor de colas \(ampliado\)” en la página 670](#)
- [“MQCRTMH: crear identificador de mensaje” en la página 676](#)
- [“MQCTL - Control de devoluciones de llamada” en la página 679](#)
- [“MQDISC - Desconectar el gestor de colas” en la página 686](#)
- [“MQDLTMH - Eliminar identificador de mensaje” en la página 690](#)
- [“MQDLTMP - Eliminar propiedad del mensaje” en la página 692](#)
- [“Mensaje MQGET - get” en la página 694](#)
- [“MQINQ - Consultar atributos de objeto” en la página 707](#)
- [“MQINQMP - Consultar propiedad del mensaje” en la página 724](#)
- [“MQMHBUF: convierte el identificador de mensaje en un búfer” en la página 730](#)
- [“MQOPEN - Abrir objeto” en la página 734](#)
- [“MQPUT - Enviar mensaje” en la página 752](#)
- [“MQPUT1 - Pon un mensaje” en la página 766](#)
- [“MQSET - Establecer atributos de objeto” en la página 776](#)
- [“MQSETMP: establecer la propiedad del mensaje” en la página 782](#)
- [“MQSTAT: recuperar información de estado” en la página 786](#)
- [“MQMHBUF: convierte el identificador de mensaje en un búfer” en la página 730](#)
- [“MQSUB - Registrar suscripción” en la página 790](#)
- [“MQSUBRQ - Solicitud de suscripción” en la página 797](#)

La ayuda en línea en las plataformas UNIX , en forma de páginas de *man* , está disponible para estas llamadas.

Nota: Las llamadas asociadas con la conversión de datos, MQXCNVC y MQ_DATA_CONV_EXIT, se encuentran en [“salida de conversión de datos” en la página 915](#).

Convenciones utilizadas en las descripciones de llamada

Para cada llamada, esta colección de temas proporciona una descripción de los parámetros y el uso de la llamada en un formato que es independiente del lenguaje de programación. Esto va seguido de invocaciones típicas de la llamada, y declaraciones típicas de sus parámetros, en cada uno de los lenguajes de programación soportados.

Importante: Al codificar llamadas de API de IBM MQ , debe asegurarse de que se proporcionan todos los parámetros relevantes (tal como se describe en las secciones siguientes). No hacerlo puede producir resultados imprevisibles.

La descripción de cada llamada contiene las secciones siguientes:

Nombre de llamada

El nombre de la llamada, seguido de una breve descripción de la finalidad de la llamada.

Parámetros

Para cada parámetro, el nombre va seguido de su tipo de datos entre paréntesis () y uno de los siguientes:

entrada

Proporcione información en el parámetro cuando realice la llamada.

salida

El gestor de colas devuelve información en el parámetro cuando la llamada se completa o falla.

entrada/salida

Proporcione información en el parámetro cuando realice la llamada y el gestor de colas cambie la información cuando la llamada se complete o falle.

Por ejemplo:

Compcode (MQLONG)-salida

En algunos casos, el tipo de datos es una estructura. En todos los casos, hay más información sobre el tipo de datos o la estructura en [“Tipos de datos elementales”](#) en la [página 177](#).

Los dos últimos parámetros de cada llamada son un código de terminación y un código de razón. El código de terminación indica si la llamada se ha completado correctamente, parcialmente o no. En el código de razón se proporciona más información sobre el éxito parcial o el fracaso de la llamada. Para obtener más información sobre cada código de terminación y razón, consulte [“Códigos de retorno”](#) en la [página 881](#).

Notas de uso

Información adicional sobre la llamada, describiendo cómo utilizarla y cualquier restricción sobre su uso.

Invocación de lenguaje ensamblador

Invocación típica de la llamada, y declaración de sus parámetros, en lenguaje ensamblador.

Invocación en C

Invocación típica de la llamada, y declaración de sus parámetros, en C.

Invocación en COBOL

Típica invocación de la llamada, y declaración de sus parámetros, en COBOL.

Invocación en PL/I

Típica invocación de la llamada, y declaración de sus parámetros, en PL/I.

Todos los parámetros se pasan por referencia.

Invocación en Visual Basic

Invocación típica de la llamada, y declaración de sus parámetros, en Visual Basic.

Otras convenciones de notación son:

Constantes

Los nombres de constantes se muestran en mayúsculas; por ejemplo, MQOO_OUTPUT. A continuación se muestra un conjunto de constantes que tienen el mismo prefijo: MQIA_*. Consulte [“Constantes”](#) en la [página 7](#) para ver el valor de una constante.

Matrices

En algunas llamadas, los parámetros son matrices de series de caracteres que no tienen tamaños fijos. En las descripciones de estos parámetros, una *n* minúscula representa una constante numérica. Cuando codifique la declaración para ese parámetro, sustituya el *n* por el valor numérico que necesite.

Utilización de las llamadas en el lenguaje C

Los parámetros que son *sólo de entrada* y de tipo MQHCONN, MQHOBJ, MQHMSG o MQLONG se pasan por valor. Para todos los demás parámetros, la *dirección* del parámetro se pasa por valor.

No es necesario que especifique todos los parámetros que se pasan por dirección cada vez que invoca una función. Cuando no necesite un parámetro determinado, especifique un puntero nulo como parámetro en la invocación de función, en lugar de la dirección de los datos de parámetro. Los parámetros para los cuales esto es posible se identifican en las descripciones de llamada.

No se devuelve ningún parámetro como valor de la llamada; en terminología C, esto significa que todas las llamadas devuelven void.

Declaración del parámetro Buffer

Las llamadas 'MQGET', 'MQPUT', y 'MQPUT1' tienen cada una un parámetro que tiene un tipo de dato indefinido: el parámetro *Buffer*. Utilice este parámetro para enviar y recibir los datos de los mensajes de la aplicación.

Los parámetros de este tipo se muestran en los ejemplos de C como matrices de MQBYTE. Puede declarar los parámetros de esta forma, pero suele ser más conveniente declararlos como la estructura particular que describe la disposición de los datos en el mensaje. El prototipo de la función declara el parámetro como un puntero a hueco, de forma que se puede especificar la dirección de cualquier tipo de dato como parámetro en la invocación de la llamada.

Pointer-to-void es un puntero a datos de formato indefinido. Se define como:

```
typedef void *PMQVOID;
```

MQBACK - Deshacer cambios

La llamada MQBACK indica al gestor de colas que se deben revertir todas las operaciones de recepción y envío de mensajes que se hayan producido desde el último punto de sincronización.

Los mensajes incluidos como parte de una unidad de trabajo se eliminan; los mensajes recuperados como parte de una unidad de trabajo se restablecen en la cola.

- En z/OS, esta llamada solo la utilizan los programas por lotes (incluidos los IMS programas DL/I por lotes).

Sintaxis

MQBACK (*Hconn*, *Código de compensación*, *Motivo*)

Parámetros

Hconn

Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* fue devuelto por una llamada anterior a MQCONN o MQCONNX.

CompCode

Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_WARNING

Aviso (finalización parcial).

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_WARNING:

MQRC_RESULTADO_PENDIENTE

(2124, X'84C') El resultado de la operación de retroceso está pendiente.

Si *CompCode* es MQCC_FAILED:

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

MQRC_API_EXIT_ERROR

(2374, X'946') La salida de la API ha fallado.

MQRC_ASID_MISMATCH

(2157, X'86D') Los ASID primario e inicial varían.

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') La llamada MQI ha entrado antes de completar la llamada anterior.

MQRC_CF_STRUC_IN_USE

(2346, X'92A') La estructura de recurso de acoplamiento se está utilizando.

MQRC_CONNECTION_BROKEN

(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

MQRC_ERROR_DE_ENTORNO

(2012, X'7DC') Llamada no válida en el entorno.

MQRC_HCONN_ERROR

(2018, X'7E2') El manejador de conexión no es válido.

MQRC_OBJECT_DAMAGED

(2101, X'835') El objeto se ha dañado.

MQRC_RESULTADO_MIXTO

(2123, X'84B') El resultado de la operación de confirmación o retroceso es mixto.

MQRC_Q_MGR_STOPPING

(2162, X'872') El gestor de colas está concluyendo.

MQRC_RESOURCE_PROBLEM

(2102, X'836') No hay disponibles suficientes recursos del sistema.

MQRC_STORAGE_MEDIUM_FULL

(2192, X'890') El soporte de almacenamiento externo está lleno.

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_UNEXPECTED_ERROR

(2195, X'893') Se ha producido un error inesperado.

Para obtener información detallada sobre estos códigos, consulte [Mensajes y códigos de motivo](#).

Notas de uso

1. Solo puede utilizar esta llamada cuando el propio gestor de colas coordina la unidad de trabajo. Este puede ser:
 - Una unidad de trabajo local, donde los cambios solo afectan a los recursos de un MQ.
 - Una unidad de trabajo global, en la que los cambios pueden afectar a recursos que pertenecen a otros gestores de recursos, además de afectar a los recursos de MQ.Para obtener más información sobre las unidades de trabajo locales y globales, consulte [“MQBEGIN - Comenzar unidad de trabajo”](#) en la página 633.
2. En entornos en los que el gestor de colas no coordina la unidad de trabajo, utilice la llamada de retroceso adecuada en lugar de MQBACK. El entorno también podría admitir una salida implícita provocada por la terminación anómala de la aplicación.
 - En z/OS, utilice las siguientes llamadas:
 - Los programas por lotes (incluidos los IMS programas DL/I por lotes) pueden utilizar la llamada MQBACK si la unidad de trabajo solo afecta a los recursos de MQ. Sin embargo, si la unidad de trabajo afecta tanto a recursos de MQ como a recursos que pertenecen a otros gestores

de recursos (por ejemplo, Db2), utilice la llamada SRRBACK proporcionada por el Servicio de z/OS Recursos Recuperables (RRS). La llamada SRRBACK devuelve los cambios realizados en los recursos que pertenecen a los gestores de recursos que se han habilitado para la coordinación RRS.

- CICS Las aplicaciones deben utilizar el EXEC CICS SYNCPOINT ROLLBACK comando para revertir la unidad de trabajo. No utilice la llamada MQBACK para CICS aplicaciones.
 - IMS Las aplicaciones (excepto los programas DL/I por lotes) deben utilizar IMS llamadas como ROLB para revertir la unidad de trabajo. No utilice la llamada MQBACK para IMS aplicaciones (excepto programas DL/I por lotes).
 - En IBM i, utilice esta llamada para las unidades de trabajo locales coordinadas por el gestor de colas. Esto significa que no debe existir una definición de compromiso a nivel de trabajo, es decir, no debe haberse emitido el STRCMTCTL comando con el **CMTSCOPE (*JOB)** parámetro para el trabajo.
3. Si una aplicación finaliza con cambios no confirmados en una unidad de trabajo, la disposición de dichos cambios depende de si la aplicación finaliza de forma normal o anómala. Consulte las notas de uso en [“MQDISC - Desconectar el gestor de colas”](#) en la página 686 para obtener más detalles.
4. Cuando una aplicación coloca u obtiene mensajes en grupos o segmentos de mensajes lógicos, el gestor de colas conserva la información relativa al grupo de mensajes y al mensaje lógico de las últimas llamadas MQPUT y MQGET realizadas con éxito. Esta información está asociada al identificador de cola e incluye datos tales como:
- Los valores de los campos *GroupId*, *MsgSeqNumber*, *MsgFlags Offset*, y en MQMD.
 - Si el mensaje forma parte de una unidad de trabajo.
 - Para la llamada MQPUT: si el mensaje es persistente o no persistente.

El gestor de colas mantiene *tres* conjuntos de información sobre grupos y segmentos, uno para cada uno de los siguientes:

- La última llamada MQPUT exitosa (esto puede formar parte de una unidad de trabajo).
 - La última llamada MQGET exitosa que eliminó un mensaje de la cola (esto puede formar parte de una unidad de trabajo).
 - La última llamada MQGET exitosa que exploró un mensaje en la cola (esto no puede formar parte de una unidad de trabajo).
5. La información asociada a la llamada MQGET se restaura al valor que tenía antes de la primera llamada MQGET correcta para ese identificador de cola en la unidad de trabajo actual.

Las colas que fueron actualizadas por la aplicación después de que se iniciara la unidad de trabajo, pero fuera del ámbito de la unidad de trabajo, no recuperan su información de grupo y segmento si se revierte la unidad de trabajo.

Al restaurar la información del grupo y del segmento a su valor anterior cuando se revierte una unidad de trabajo, la aplicación puede distribuir un grupo de mensajes grande o un mensaje lógico grande compuesto por muchos segmentos entre varias unidades de trabajo, y reiniciar en el punto correcto del grupo de mensajes o del mensaje lógico si falla una de las unidades de trabajo.

El uso de varias unidades de trabajo puede resultar ventajoso si el gestor de colas local solo dispone de un almacenamiento de colas limitado. Sin embargo, la aplicación debe conservar suficiente información para poder reiniciar el envío o la recepción de mensajes en el punto correcto si se produce un fallo del sistema.

Para obtener más información sobre cómo reiniciar en el punto correcto después de un fallo del sistema, consulte la opción MQPMO_LOGICAL_ORDER descrita en [“MQPMO - Opciones de mensaje de envío”](#) en la página 477 y la opción MQGMO_LOGICAL_ORDER descrita en [“MQGMO - Opciones para obtener mensajes”](#) en la página 331.

Las notas de uso restantes solo se aplican cuando el gestor de colas coordina las unidades de trabajo.

6. Una unidad de trabajo tiene el mismo alcance que un identificador de conexión. Todas las llamadas a MQ que afecten a una unidad de trabajo concreta deben realizarse utilizando el mismo identificador

de conexión. Las llamadas realizadas utilizando un identificador de conexión diferente (por ejemplo, llamadas realizadas por otra aplicación) afectan a una unidad de trabajo diferente. Consulte el **Hconn** parámetro descrito en “MQCONN - Conectar gestor de colas” en la página 663 para obtener información sobre el ámbito de los identificadores de conexión.

7. Solo los mensajes que se colocaron o recuperaron como parte de la unidad de trabajo actual se ven afectados por esta llamada.
8. Una aplicación de larga duración que emite llamadas MQGET, MQPUT o MQPUT1 dentro de una unidad de trabajo, pero que nunca emite una llamada de confirmación o retroceso, puede llenar las colas con mensajes que no están disponibles para otras aplicaciones. Para protegerse contra esta posibilidad, el administrador debe establecer el atributo del **MaxUncommittedMsgs** gestor de colas en un valor lo suficientemente bajo como para evitar que las aplicaciones descontroladas llenen las colas, pero lo suficientemente alto como para permitir que las aplicaciones de mensajería esperadas funcionen correctamente.

Invocación en C

```
MQBACK (Hconn, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQHCONN  Hconn;      /* Connection handle */
MQLONG   CompCode;   /* Completion code */
MQLONG   Reason;     /* Reason code qualifying CompCode */
```

Invocación en COBOL

```
CALL 'MQBACK' USING HCONN, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
** Connection handle
01 HCONN      PIC S9(9) BINARY.
** Completion code
01 COMPCODE   PIC S9(9) BINARY.
** Reason code qualifying COMPCODE
01 REASON     PIC S9(9) BINARY.
```

Invocación en PL/I

```
call MQBACK (Hconn, CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```
dcl Hconn      fixed bin(31); /* Connection handle */
dcl CompCode   fixed bin(31); /* Completion code */
dcl Reason     fixed bin(31); /* Reason code qualifying CompCode */
```

Invocación en ensamblador de alto nivel

```
CALL MQBACK,(HCONN,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

```
HCONN      DS F Connection handle
COMPCODE   DS F Completion code
REASON     DS F Reason code qualifying COMPCODE
```

Invocación en Visual Basic

```
MQBACK Hconn, CompCode, Reason
```

Declare los parámetros como se indica a continuación:

```
Dim Hconn As Long 'Connection handle'  
Dim CompCode As Long 'Completion code'  
Dim Reason As Long 'Reason code qualifying CompCode'
```

MQBEGIN - Comenzar unidad de trabajo

La llamada MQBEGIN inicia una unidad de trabajo coordinada por el gestor de colas, en la que pueden intervenir gestores de recursos externos.

Sintaxis

MQBEGIN (*Hconn*, *BeginOptions*, *Compcode*, *Reason*)

Parámetros

Hconn

Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* fue devuelto por una llamada anterior a MQCONN o MQCONNX.

Hconn Debe ser un identificador de conexión no compartido. Si se especifica un identificador de conexión compartida, la llamada falla con el código de motivo MQRC_HCONN_ERROR. Consulte la descripción de las opciones MQCNO_HANDLE_SHARE_* en [“MQCNO - Opciones de conexión”](#) en la [página 269](#) para obtener más información sobre los identificadores compartidos y no compartidos.

BeginOptions

Tipo: MQBO - entrada/salida

Estas son opciones que controlan la acción de MQBEGIN, tal y como se describe en [“MQBO - Opciones de inicio”](#) en la [página 225](#).

Si no se requieren opciones, los programas escritos en C o en ensambladore S/390 e pueden especificar una dirección de parámetro nula, en lugar de especificar la dirección de una estructura MQBO.

CompCode

Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_WARNING

Aviso (finalización parcial).

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_WARNING:

MQRC_SIN_PARTICIPANTES_EXTERNOS

(2121, X'849') No hay gestores de recursos participantes registrados.

MQRC_PARTICIPANTE_NO_DISPONIBLE

(2122, X'84A') Gestor de recursos participante no disponible.

Si *CompCode* es MQCC_FAILED:

MQRC_API_EXIT_ERROR

(2374, X'946') La salida de la API ha fallado.

MQRC_BO_ERROR

(2134, X'856') Estructura de opciones de inicio no válida.

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') La llamada MQI ha entrado antes de completar la llamada anterior.

MQRC_CONNECTION_BROKEN

(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

MQRC_ERROR_DE_ENTORNO

(2012, X'7DC') Llamada no válida en el entorno.

MQRC_HCONN_ERROR

(2018, X'7E2') El manejador de conexión no es válido.

MQRC_OPTIONS_ERROR

(2046, X'7FE') Las opciones no son válidas o no son coherentes.

MQRC_Q_MGR_STOPPING

(2162, X'872') El gestor de colas está concluyendo.

MQRC_RESOURCE_PROBLEM

(2102, X'836') No hay disponibles suficientes recursos del sistema.

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_UNEXPECTED_ERROR

(2195, X'893') Se ha producido un error inesperado.

MQRC_UOW_EN_PROCESO

(2128, X'850') Unidad de trabajo ya iniciada.

Para obtener más información sobre estos códigos, consulte [Mensajes y códigos de motivo](#).

Notas de uso

1. Utilice la llamada MQBEGIN para iniciar una unidad de trabajo coordinada por el gestor de colas y que puede implicar cambios en recursos propiedad de otros gestores de recursos. El gestor de colas admite tres tipos de unidades de trabajo:
 - **Unidad de trabajo local coordinada por el gestor de colas** : Unidad de trabajo en la que el gestor de colas es el único gestor de recursos que participa, por lo que actúa como coordinador de la unidad de trabajo.
 - Para iniciar este tipo de unidad de trabajo, especifique la opción MQPMO_SYNCPOINT o MQGMO_SYNCPOINT en la primera llamada a MQPUT, MQPUT1 o MQGET de la unidad de trabajo.
 - Para confirmar o cancelar este tipo de unidad de trabajo, utilice la llamada MQCMIT o MQBACK.
 - **Unidad de trabajo global coordinada por el gestor de colas** : Unidad de trabajo en la que el gestor de colas actúa como coordinador de la unidad de trabajo, tanto para los recursos de MQ como para los recursos que pertenecen a otros gestores de recursos. Esos administradores de recursos cooperan con el administrador de colas para garantizar que todos los cambios en los recursos de la unidad de trabajo se confirmen o se deshagan juntos.
 - Para iniciar este tipo de unidad de trabajo, utilice la llamada MQBEGIN.
 - Para confirmar o cancelar este tipo de unidad de trabajo, utilice las llamadas MQCMIT y MQBACK.

- **Unidad de trabajo global coordinada externamente** : unidad de trabajo en la que el gestor de colas es un participante, pero no actúa como coordinador de la unidad de trabajo. En su lugar, hay un coordinador externo de unidades de trabajo con el que coopera el gestor de colas.
 - Para iniciar este tipo de unidad de trabajo, utilice la llamada correspondiente proporcionada por el coordinador externo de la unidad de trabajo.

Si se utiliza la llamada MQBEGIN para intentar iniciar la unidad de trabajo, la llamada falla con el código de motivo MQRC_ENVIRONMENT_ERROR.
 - Para confirmar o deshacer este tipo de unidad de trabajo, utilice las llamadas de confirmación y deshacer proporcionadas por el coordinador externo de unidades de trabajo.

Si utiliza la llamada MQCMIT o MQBACK para confirmar o deshacer la unidad de trabajo, la llamada fallará con el código de motivo MQRC_ENVIRONMENT_ERROR.
- 2. Si la aplicación finaliza con cambios no confirmados en una unidad de trabajo, la disposición de dichos cambios depende de si la aplicación finaliza de forma normal o anómala. Consulte las notas de uso en [“MQDISC - Desconectar el gestor de colas”](#) en la página 686 para obtener más detalles.
- 3. Una aplicación solo puede participar en una unidad de trabajo a la vez. La llamada MQBEGIN falla con el código de motivo MQRC_UOW_IN_PROGRESS si ya existe una unidad de trabajo para la aplicación, independientemente del tipo de unidad de trabajo que sea.
- 4. La llamada MQBEGIN no es válida en un entorno de cliente de MQ MQI. El intento de utilizar la llamada falla con el código de motivo MQRC_ENVIRONMENT_ERROR.
- 5. Cuando el gestor de colas actúa como coordinador de unidades de trabajo para unidades de trabajo globales, los gestores de recursos que pueden participar en la unidad de trabajo se definen en el archivo de configuración del gestor de colas.
- 6. En IBM i, se admiten los tres tipos de unidades de trabajo de la siguiente manera:
 - **La unidad de trabajo local coordinada por el gestor de colas** solo se puede utilizar cuando no existe una definición de compromiso a nivel de trabajo, es decir, no se debe haber emitido el STRCMTCTL comando con el **CMTSCOPE(*JOB)** parámetro para el trabajo.
 - No se admite **la unidad de trabajo global coordinada por el gestor de colas**.
 - **La unidad de trabajo global coordinada externamente** solo se puede utilizar cuando existe una definición de compromiso a nivel de trabajo, es decir, se debe haber emitido el STRCMTCTL comando con el **CMTSCOPE(*JOB)** parámetro para el trabajo. Si se ha hecho esto, las IBM iCOMMIT operaciones ROLLBACK y se aplican a los recursos de MQ, así como a los recursos que pertenecen a otros administradores de recursos participantes.

Invocación en C

```
MQBEGIN (Hconn, &BeginOptions, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQHCONN  Hconn;           /* Connection handle */
MQBO     BeginOptions;    /* Options that control the action of MQBEGIN */
MQLONG   CompCode;        /* Completion code */
MQLONG   Reason;          /* Reason code qualifying CompCode */
```

Invocación en COBOL

```
CALL 'MQBEGIN' USING HCONN, BEGINOPTIONS, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
** Connection handle
01 HCONN          PIC S9(9) BINARY.
** Options that control the action of MQBEGIN
01 BEGINOPTIONS.
   COPY CMQBOV.
```

```

** Completion code
01 COMPCODE      PIC S9(9) BINARY.
** Reason code qualifying COMPCODE
01 REASON        PIC S9(9) BINARY.

```

Invocación en PL/I

```
call MQBEGIN (Hconn, BeginOptions, CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```

dcl Hconn          fixed bin(31); /* Connection handle */
dcl BeginOptions   like MQBO;    /* Options that control the action of
                                   MQBEGIN */
dcl CompCode       fixed bin(31); /* Completion code */
dcl Reason         fixed bin(31); /* Reason code qualifying CompCode */

```

Invocación en Visual Basic

```
MQBEGIN Hconn, BeginOptions, CompCode, Reason
```

Declare los parámetros como se indica a continuación:

```

Dim Hconn          As Long 'Connection handle'
Dim BeginOptions   As MQBO 'Options that control the action of MQBEGIN'
Dim CompCode       As Long 'Completion code'
Dim Reason         As Long 'Reason code qualifying CompCode'

```

MQBUFMH - Convertir almacenamiento intermedio en descriptor de contexto de mensaje

La llamada a la función MQBUFMH convierte un búfer en un identificador de mensaje y es la inversa de la llamada MQMHBUF.

Esta llamada toma un descriptor de mensaje y las propiedades de un MQRFH2 e en el búfer y las pone a disposición a través de un identificador de mensaje. Las propiedades « MQRFH2 » de los datos del mensaje se eliminan de forma opcional. Los *Format* campos *Encoding*, *CodedCharSetId* y del descriptor del mensaje se actualizan, si es necesario, para describir correctamente el contenido del búfer después de que se hayan eliminado las propiedades.

Sintaxis

```
MQBUFMH ( Hconn, Hmsg, BufMsgHOpts, MsgDesc, BufferLength, Buffer, DataLength, Compcode, Reason )
```

Parámetros

Hconn

Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de **Hconn** debe coincidir con el identificador de conexión que se utilizó para crear el identificador de mensaje especificado en el **Hmsg** parámetro.

Si el identificador de mensaje se creó utilizando MQHC_UNASSOCIATED_HCONN, se debe establecer una conexión válida en el subproceso que convierte un búfer en un identificador de mensaje. Si no se establece una conexión válida, la llamada falla con MQRC_CONNECTION_BROKEN.

Hmsg

Tipo: MQHMSG - entrada

Este es el identificador del mensaje para el que se requiere un búfer. El valor fue devuelto por una llamada MQCRTMH anterior.

BufMsgHOpts

Tipo: MQBMHO - entrada

La estructura MQBMHO permite a las aplicaciones especificar opciones que controlan cómo se producen los identificadores de mensajes a partir de los búferes.

Consulte [“MQBMHO: Opciones de búfer a controlador de mensajes”](#) en la página 219 para obtener los detalles.

MsgDesc

Tipo: MQMD - entrada/salida

La *MsgDesc* estructura contiene las propiedades del descriptor del mensaje y describe el contenido del área del búfer.

Al finalizar la llamada, las propiedades se eliminan opcionalmente del área de búfer y, en este caso, el descriptor de mensajes se actualiza para describir correctamente el área de búfer.

Los datos de esta estructura deben estar en el juego de caracteres y la codificación de la aplicación.

BufferLength

Tipo: MQLONG - entrada

BufferLength es la longitud del área de búfer, en bytes.

Un *BufferLength* de cero bytes es válido e indica que el área del búfer no contiene datos.

Almacenamiento intermedio

Tipo: MQBYTEExBufferLength - entrada/salida

Buffer define el área que contiene el búfer de mensajes. Para la mayoría de los datos, debe alinear el búfer en un límite de 4 bytes.

Si **Buffer** contiene datos alfanuméricos o numéricos, establezca los campos *CodedCharSetId* *Encoding* y del **MsgDesc** parámetro con los valores adecuados para los datos; esto permite convertir los datos, si es necesario.

Si se encuentran propiedades en el búfer de mensajes, se eliminan opcionalmente; posteriormente estarán disponibles desde el identificador de mensajes al regresar de la llamada.

En el lenguaje de programación C, el parámetro se declara como un puntero a void, lo que significa que se puede especificar como parámetro la dirección de cualquier tipo de datos.

Si el **BufferLength** parámetro es cero, no **Buffer** se hace referencia a; en este caso, la dirección del parámetro pasada por programas escritos en C o en ensambladore System/390 e puede ser nula.

DataLength

Tipo: MQLONG - salida

La longitud, en bytes, del búfer del que se podrían eliminar las propiedades.

CompCode

Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'089C') Adaptador no disponible.

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

MQRC_ASID_MISMATCH

(2157, X'86D') Los ASID primario e inicial varían.

MQRC_BMHO_ERROR

(2489, X'09B9') La estructura de opciones del búfer al controlador de mensajes no es válida.

MQRC_BUFFER_ERROR

(2004, X'07D4') Parámetro de búfer no válido.

MQRC_BUFFER_LENGTH_ERROR

(2005, X'07D5') Parámetro de longitud del búfer no válido.

MQRC_CALL_IN_PROGRESS

(2219, X'08AB') MQI : se ha iniciado una llamada antes de que finalizara la anterior.

MQRC_CONNECTION_BROKEN

(2009, X'07D9') Se ha perdido la conexión con el gestor de colas.

MQRC_HMSG_ERROR

(2460, X'099C') Identificador de mensaje no válido.

MQRC_MD_ERROR

(2026, X'07EA') Descriptor de mensaje no válido.

MQRC_MSG_HANDLE_EN_USO

(2499, X'09C3') El identificador del mensaje ya está en uso.

MQRC_OPTIONS_ERROR

(2046, X'07FE') Opciones no válidas o incoherentes.

MQRC_RFH_ERROR

(2334, X'091E') Estructura MQRFH2 no válida.

MQRC_RFH_FORMATO_ERROR

(2421, X'0975') No se pudo analizar una carpeta MQRFH2 que contiene propiedades.

MQRC_UNEXPECTED_ERROR

(2195, X'893') Se ha producido un error inesperado.

Para obtener información detallada sobre estos códigos, consulte [Mensajes y códigos de motivo](#).

Notas de uso

Las llamadas MQBUFMH no pueden ser interceptadas por salidas API: un búfer se convierte en un identificador de mensaje en el espacio de la aplicación; la llamada no llega al gestor de colas.

Invocación en C

```
MQBUFMH (Hconn, Hmsg, &BufMsgHOpts, &MsgDesc, BufferLength, Buffer,  
         &DataLength, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQHCONN Hconn;          /* Connection handle */  
MQHMSG  Hmsg;           /* Message handle */  
MQBMHO  BufMsgHOpts;    /* Options that control the action of MQBUFMH */  
MQMD    MsgDesc;        /* Message descriptor */  
MQLONG  BufferLength;    /* Length in bytes of the Buffer area */  
MQBYTE  Buffer[n];       /* Area to contain the message buffer */  
MQLONG  DataLength;     /* Length of the output buffer */  
MQLONG  CompCode;       /* Completion code */  
MQLONG  Reason;         /* Reason code qualifying CompCode */
```

Invocación en COBOL

```
CALL 'MQBUFMH' USING HCONN, HMSG, BUFMSGHOPTS, MSGDESC, BUFFERLENGTH,  
                     BUFFER, DATALENGTH, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
** Connection handle  
01 HCONN          PIC S9(9) BINARY.  
** Message handle  
01 HMSG           PIC S9(18) BINARY.  
** Options that control the action of MQBUFMH  
01 BUFMSGHOPTS.  
   COPY CMQBMHOV.  
** Message descriptor  
01 MSGDESC.  
   COPY CMQMD.  
** Length in bytes of the Buffer area  
01 BUFFERLENGTH  PIC S9(9) BINARY.  
** Area to contain the message buffer  
01 BUFFER         PIC X(n).  
** Length of the output buffer  
01 DATALENGTH   PIC S9(9) BINARY.  
** Completion code  
01 COMPCODE      PIC S9(9) BINARY.  
** Reason code qualifying COMPCODE  
01 REASON        PIC S9(9) BINARY.
```

Invocación en PL/I

```
call MQBUFMH (Hconn, Hmsg, BufMsgHOpts, MsgDesc, BufferLength, Buffer,  
DataLength, CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```
dcl Hconn          fixed bin(31); /* Connection handle */  
dcl Hmsg           fixed bin(63); /* Message handle */  
dcl BufMsgHOpts    like MQBMHO;  /* Options that control the action of  
                                MQBUFMH */  
dcl MsgDesc        like MQMD;     /* Message descriptor */  
dcl BufferLength    fixed bin(31); /* Length in bytes of the Buffer area */  
dcl Buffer          char(n);       /* Area to contain the message buffer */  
dcl DataLength     fixed bin(31); /* Length of the output buffer */  
dcl CompCode       fixed bin(31); /* Completion code */  
dcl Reason         fixed bin(31); /* Reason code qualifying CompCode */
```

Invocación en ensamblador de alto nivel

```
CALL MQBUFMH, (HCONN,HMSG,BUFMSGHOPTS,MSGDESC,BUFFERLENGTH,BUFFER,  
              DATALENGTH,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

HCONN	DS	F	Connection handle
HMSG	DS	D	Message handle
BUFMSGHOPTS	CMQBMHOA	,	Options that control the action of MQBUFMH
MSGDESC	CMQMDA	,	Message descriptor
BUFFERLENGTH	DS	F	Length in bytes of the BUFFER area
BUFFER	DS	CL(n)	Area to contain the properties
DATALENGTH	DS	F	Length of the output buffer
COMPCODE	DS	F	Completion code
REASON	DS	F	Reason code qualifying COMPCODE

MQCB - Gestionar devolución de llamada

La llamada MQCB registra una devolución de llamada para el identificador de objeto especificado y controla la activación y los cambios en la devolución de llamada.

Una devolución de llamada es un fragmento de código (especificado como el nombre de una función que se puede vincular dinámicamente o como puntero de función) que se invoca cuando IBM MQ se producen determinados eventos.

Para utilizar MQCB y MQCTL en un cliente, debe estar conectado a un servidor en el que el parámetro **SHARECNV** negociado del canal haya acordado un valor distinto de cero.

Los tipos de devolución de llamada que se pueden definir son:

Consumidor de mensajes

La función de devolución de llamada del consumidor de mensajes se invoca cuando hay un mensaje disponible en un identificador de objeto que cumple los criterios de selección especificados.

Solo se puede registrar una función de devolución de llamada para cada identificador de objeto. Si se va a leer una sola cola con múltiples criterios de selección, entonces la cola debe abrirse varias veces y debe registrarse una función de consumo en cada controlador.

Manejador de sucesos

El controlador de eventos se invoca para condiciones que afectan a todo el entorno de devolución de llamada.

La función se invoca cuando se produce una condición de evento, por ejemplo, cuando se detiene o se suspende un gestor de colas o una conexión.

La función no se invoca para condiciones específicas de un único consumidor de mensajes, por ejemplo, MQRC_GET_INHIBITED; sin embargo, se invoca si una función de devolución de llamada no finaliza normalmente.

Sintaxis

MQCB (*Hconn*, *Operación*, *Descripción de devolución de llamada*, *Hobj*, *Descripción de mensaje*, *Obtener opciones de mensaje*, *Código de compilación*, *Motivo*)

Parámetros

Hconn

Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* fue devuelto por una llamada anterior a MQCONN o MQCONNX.

En z/OS para CICS aplicaciones, puede especificar el siguiente valor especial para *MQHC_DEF_HCONN* utilizar el identificador de conexión asociado a esta unidad de ejecución.

Operación

Tipo: MQLONG - entrada

La operación que se está procesando en la devolución de llamada definida para el identificador de objeto especificado. Debe especificar una de las siguientes opciones. Para especificar más de una opción, suma los valores (sin añadir la misma constante más de una vez) o combínalos mediante la operación OR bit a bit (si el lenguaje de programación admite operaciones bit a bit).

MQOP_REGISTRAR

Defina la función de devolución de llamada para el identificador de objeto especificado. Esta operación define la función que se va a llamar y los criterios de selección que se van a utilizar.

Si ya se ha definido una función de devolución de llamada para el identificador del objeto, se sustituye la definición. Si se detecta un error al reemplazar la devolución de llamada, la función se da de baja.

Si se registra una devolución de llamada en la misma función de devolución de llamada en la que se había dado de baja anteriormente, esto se trata como una operación de sustitución; no se invocan las llamadas iniciales ni finales.

Puede utilizar MQOP_REGISTER con MQOP_SUSPEND o MQOP_RESUME.

MQOP_DEREGISTRAR

Detiene el consumo de mensajes para el identificador del objeto y elimina el identificador de los que son elegibles para una devolución de llamada.

Una devolución de llamada se da de baja automáticamente si se cierra el identificador asociado.

Si se llama a MQOP_DEREGISTER desde un consumidor y la devolución de llamada tiene definida una llamada de parada, esta se invoca al regresar del consumidor.

Si esta operación se emite contra un *Hobj* sin consumidor registrado, la llamada devuelve MQRC_CALLBACK_NOT_REGISTERED.

MQOP_SUSPENDER

Suspende el consumo de mensajes para el identificador del objeto.

Si esta operación se aplica a un controlador de eventos, el controlador de eventos no recibe eventos mientras está suspendido, y los eventos perdidos mientras se encuentra en estado suspendido no se proporcionan a la operación cuando se reanuda.

Mientras está suspendida, la función de consumo sigue recibiendo las devoluciones de llamada del tipo de control.

MQOP_RESUMIR

Reanuda el consumo de mensajes para el identificador del objeto.

Si esta operación se aplica a un controlador de eventos, el controlador de eventos no recibe eventos mientras está suspendido, y los eventos perdidos mientras se encuentra en estado suspendido no se proporcionan a la operación cuando se reanuda.

CallbackDesc

Tipo: MQCBD - entrada

Esta es una estructura que identifica la función de devolución de llamada que está registrando la aplicación y las opciones utilizadas al registrarla.

Consulte [MQCBD](#) para obtener más información sobre la estructura.

El descriptor de devolución de llamada solo es necesario para la opción MQOP_REGISTER; si el descriptor no es necesario, la dirección del parámetro pasada puede ser nula.

Hobj

Tipo: MQHOBJ - entrada

Este identificador representa el acceso que se ha establecido al objeto desde el que se va a consumir un mensaje. Se trata de un identificador que ha sido devuelto por una llamada anterior a [MQOPEN](#) o [MQSUB](#) (en el **Hobj** parámetro).

Hobj No es necesario al definir una rutina de controlador de eventos (MQCBT_EVENT_HANDLER) y debe especificarse como MQHO_NONE.

Si *Hobj* ha sido devuelto desde una llamada MQOPEN, la cola debe haberse abierto con una o más de las siguientes opciones:

- MQOO_INPUT_SHARED
- MQOO_INPUT_EXCLUSIVE
- MQOO_INPUT_AS_Q_DEF
- MQOO_BROWSE

MsgDesc

Tipo: MQMD - entrada

Esta estructura describe los atributos del mensaje necesario y los del mensaje recuperado.

El **MsgDesc** parámetro define los atributos de los mensajes requeridos por el consumidor y la versión del MQMD que se pasará al consumidor del mensaje.

Los *MsgId*, *CorrelId*, *GroupId*, *MsgSeqNumber*, y *Offset* del MQMD se utilizan para la selección de mensajes, dependiendo de las opciones especificadas en el **GetMsgOpts** parámetro.

Se utilizan *Encoding CodedCharSetId* y para la conversión de mensajes si se especifica la opción MQGMO_CONVERT.

Consulte [MQMD](#) para obtener más detalles.

MsgDesc Se utiliza para MQOP_REGISTER y si necesita valores distintos de los predeterminados para cualquier campo. *MsgDesc* no se utiliza para un controlador de eventos.

Si el descriptor no es necesario, la dirección del parámetro pasada puede ser nula.

Tenga en cuenta que si hay varios consumidores registrados en la misma cola con selectores que se superponen, el consumidor elegido para cada mensaje es indefinido.

GetMsgOpts

Tipo: MQGMO - entrada

El **GetMsgOpts** parámetro controla cómo el consumidor de mensajes obtiene los mensajes. Todas las opciones de este parámetro tienen los significados descritos en [“MQGMO - Opciones para obtener mensajes”](#) en la página 331, cuando se utilizan en una llamada MQGET, excepto:

MQGMO_SET_SIGNAL

Esta opción no está permitida.

MQGMO_BROWSE_FIRST, MQGMO_BROWSE_NEXT, MQGMO_MARK_*

El orden de los mensajes entregados a un consumidor que navega viene dictado por las combinaciones de estas opciones. Las combinaciones significativas son:

MQGMO_BROWSE_FIRST

El primer mensaje de la cola se entrega repetidamente al consumidor. Esto resulta útil cuando el consumidor consume de forma destructiva el mensaje en la devolución de llamada. Utilice esta opción con precaución.

MQGMO_BROWSE_NEXT

El consumidor recibe cada mensaje de la cola, desde la posición actual del cursor hasta llegar al final de la cola.

MQGMO_BROWSE_FIRST + MQGMO_BROWSE_NEXT

El cursor se restablece al inicio de la cola. A continuación, se envía cada mensaje al consumidor hasta que el cursor llega al final de la cola.

MQGMO_BROWSE_FIRST + MQGMO_MARK_*

Comenzando por el principio de la cola, se le da al consumidor el primer mensaje no marcado de la cola, que luego se marca para este consumidor. Esta combinación garantiza que el consumidor pueda recibir nuevos mensajes añadidos detrás del punto actual del cursor.

MQGMO_BROWSE_NEXT + MQGMO_MARK_*

A partir de la posición del cursor, se le proporciona al consumidor el siguiente mensaje no marcado de la cola, que luego se marca para este consumidor. Utilice esta combinación con precaución, ya que los mensajes se pueden añadir a la cola detrás de la posición actual del cursor.

MQGMO_BROWSE_FIRST + MQGMO_BROWSE_NEXT + MQGMO_MARK_*

Esta combinación no está permitida. Si se utiliza, la llamada devuelve MQRC_OPTIONS_ERROR.

MQGMO_NO_WAIT, MQGMO_WAIT y WaitInterval

Estas opciones controlan cómo se invoca al consumidor.

MQGMO_NO_WAIT

El consumidor nunca recibe una llamada con MQRC_NO_MSG_AVAILABLE. El consumidor solo recibe llamadas para mensajes y eventos.

MQGMO_WAIT con un valor nulo e WaitInterval

El código MQRC_NO_MSG_AVAILABLE se transmite al consumidor cuando no hay mensajes disponibles y el consumidor se ha iniciado o se le ha entregado al menos un mensaje desde el último código de motivo «sin mensajes».

Esto evita que el consumidor realice sondeos en un bucle ocupado cuando se especifica un intervalo de espera cero.

MQGMO_WAIT y un valor positivo de WaitInterval

Se llama al consumidor tras el intervalo de espera especificado con el código de motivo MQRC_NO_MSG_AVAILABLE. Esta llamada se realiza independientemente de si se han entregado mensajes al consumidor. Esto permite al usuario realizar procesamiento de tipo heartbeat o por lotes.

MQGMO_WAIT y WaitInterval de MQWI_UNLIMITED

Esto especifica una espera infinita antes de devolver MQRC_NO_MSG_AVAILABLE. El consumidor nunca recibe una llamada con MQRC_NO_MSG_AVAILABLE.

GetMsgOpts Se utiliza solo para MQOP_REGISTER y si necesita valores distintos de los predeterminados para cualquier campo. *GetMsgOpts* no se utiliza para un controlador de eventos.

Si no son necesarios, la *GetMsgOpts* dirección del parámetro pasada puede ser nula. El uso de este parámetro es equivalente a especificar MQGMO_DEFAULT junto con MQGMO_FAIL_IF QUIESCING.

Si se proporciona un identificador de propiedades de mensaje en la estructura MQGMO, se proporciona una copia en la estructura MQGMO que se pasa a la devolución de llamada del consumidor. Al regresar de la llamada MQCB, la aplicación puede eliminar el identificador de propiedades del mensaje.

CompCode

Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_WARNING

Aviso (finalización parcial).

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

Los códigos de motivo de la siguiente lista son los que el gestor de colas puede devolver para el **Reason** parámetro.

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'89C') El adaptador no está disponible.

nMQRC_ADAPTER_CONV_LOAD_ERROR

(2133, X'855') No se han podido cargar módulos de servicio de conversión de datos.

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

MQRC_API_EXIT_ERROR

(2374, X'946') La salida de la API ha fallado.

MQRC_API_EXIT_LOAD_ERROR

(2183, X'887') No se ha podido cargar la salida de la API.

MQRC_ASID_MISMATCH

(2157, X'86D') Los ASID primario e inicial varían.

MQRC_BUFFER_LENGTH_ERROR

(2005, X'7D5') El parámetro de longitud de almacenamiento intermedio no es válido.

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') La llamada MQI ha entrado antes de completar la llamada anterior.

MQRC_CALLBACK_LINK_ERROR

(2487, X'9B7') Campo de tipo de devolución de llamada incorrecto.

MQRC_CALLBACK_NO_REGISTRADO

(2448, X'990') No se puede dar de baja, suspender o reanudar porque no hay ninguna devolución de llamada registrada.

MQRC_CALLBACK_ROUTINE_ERROR

(2486, X'9B6') Se debe *CallbackFunctionCallbackName* especificar o, pero no ambos.

MQRC_CALLBACK_TYPE_ERROR

(2483, X'9B3') Campo de tipo de devolución de llamada incorrecto.

MQRC_CBD OPCIONES_ERROR

(2484, X'9B4') Campo de opciones MQCBD incorrecto.

MQRC_CICS_WAIT_FAILED

(2140, X'85C') Solicitud de espera rechazada por CICS.

MQRC_CONNECTION_BROKEN

(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

MQRC_CONNECTION_NOT_AUTHORIZED

(2217, X'8A9') No tiene autorización para la conexión.

MQRC_CONNECTION QUIESCING

(2202, X'89A') Conexión en fase de inmovilización.

MQRC_CONNECTION_STOPPING

(2203, X'89B') La conexión está concluyendo.

MQRC_CORREL_ID_ERROR

(2207, X'89F') Error de identificador de correlación.

MQRC_DATA_LENGTH_ERROR

(2010, X'7DA') El parámetro longitud de datos no es válido.

MQRC_ERROR_DE_ENTORNO

(2012, X'7DC') Llamada no válida en el entorno.

MQRC_FUNCTION_NOT_SUPPORTED

(2298, X'8FA') La función solicitada no está disponible en el entorno actual.

MQRC_GET_INHIBITED

(2016, X'7E0') Se han inhibido las obtenciones para la cola.

MQRC_GLOBAL_UOW_CONFLICT

(2351, X'92F') Conflicto de unidades de trabajo global.

MQRC_GMO_ERROR

(2186, X'88A') No es válida la estructura de las opciones de obtención de mensaje.

MQRC_HANDLE_IN_USE_FOR_UOW

(2353, X'931') Descriptor de contexto que se utiliza para la unidad de trabajo global.

MQRC_HCONN_ERROR

(2018, X'7E2') El manejador de conexión no es válido.

MQRC_HOBJ_ERROR

(2019, X'7E3') El manejador de objeto no es válido.

MQRC_INCONSISTENT_BROWSE

(2259, X'8D3') La especificación de examinar es incoherente.

MQRC_INCONSISTENT_UOW

(2245, X'8C5') Especificación incoherente de unidad de trabajo.

MQRC_INVALID_MSG_UNDER_CURSOR

(2246, X'8C6') El mensaje bajo cursor no es válido para recuperación.

MQRC_LOCAL_UOW_CONFLICT

(2352, X'930') La unidad de trabajo global entra en conflicto con la unidad de trabajo local.

MQRC_MATCH_OPTIONS_ERROR

(2247, X'8C7') Las opciones de coincidencia no son válidas.

MQRC_MAX_LONGITUD_DEL_MENSAJE_ERROR

(2485, X'9B4') Campo *MaxMsgLength* incorrecto.

MQRC_MD_ERROR

(2026, X'7EA') El descriptor de mensaje no es válido.

MQRC_MODULE_ENTRY_NOT_FOUND

(2497, X'9C1') No se ha encontrado el punto de entrada de la función especificada en el módulo.

MQRC_MÓDULO_INVÁLIDO

(2496, X'9C0') Se ha encontrado el módulo, pero es del tipo incorrecto; no es de 32 bits, 64 bits ni una biblioteca de enlaces dinámicos válida.

MQRC_MÓDULO_NO_ENCONTRADO

(2495, X'9BF') Módulo no encontrado en la ruta de búsqueda o sin autorización para cargarse.

MQRC_MSG_SEQ_NUMBER_ERROR

(2250, X'8CA') El número de secuencia de mensaje no es válido.

MQRC_MSG_TOKEN_ERROR

(2331, X'91B') El uso del símbolo de mensaje no es válido.

MQRC_NO_MSG_AVAILABLE

(2033, X'7F1') No hay ningún mensaje disponible.

MQRC_NO_MSG_UNDER_CURSOR

(2034, X'7F2') El cursor para examinar no está situado en el mensaje.

MQRC_NOT_OPEN_FOR_BROWSE

(2036, X'7F4') La cola no se ha abierto para examen.

MQRC_NOT_OPEN_FOR_INPUT

(2037, X'7F5') La cola no se ha abierto para entrada.

MQRC_OBJECT_CHANGED

(2041, X'7F9') La definición de objeto ha cambiado desde que se ha abierto.

MQRC_OBJECT_DAMAGED

(2101, X'835') El objeto se ha dañado.

ERROR_DE_OPERACIÓN_MQRC

(2206, X'89E') Código de operación incorrecto en la llamada a la API.

MQRC_OPTIONS_ERROR

(2046, X'7FE') Las opciones no son válidas o no son coherentes.

MQRC_PAGESET_ERROR

(2193, X'891') Error al acceder al conjunto de datos del conjunto de páginas.

MQRC_Q_DELETED

(2052, X'804') La cola se ha suprimido.

MQRC_Q_INDEX_TYPE_ERROR

(2394, X'95A') La cola tiene un tipo de índice incorrecto.

MQRC_Q_MGR_NAME_ERROR

(2058, X'80A') El nombre del gestor de colas no es válido o no se conoce.

MQRC_Q_MGR_NOT_AVAILABLE

(2059, X'80B') El gestor de colas no está disponible para efectuar la conexión.

MQRC_Q_MGR QUIESCING

(2161, X'871') El gestor de colas se está desactivando temporalmente.

MQRC_Q_MGR_STOPPING

(2162, X'872') El gestor de colas está concluyendo.

MQRC_RESOURCE_PROBLEM

(2102, X'836') No hay disponibles suficientes recursos del sistema.

MQRC_SIGNAL_OUTSTANDING

(2069, X'815') Símbolo pendiente para este descriptor de contexto.

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_SUPPRESSED_BY_EXIT

(2109, X'83D') El programa de salida ha suprimido la llamada.

MQRC_SYNCPOINT_LIMIT_REACHED

(2024, X'7E8') No pueden manejarse más mensajes dentro de la unidad de trabajo actual.

MQRC_SYNCPOINT_NOT_AVAILABLE

(2072, X'818') El punto de sincronización de soporte no está disponible.

MQRC_UNEXPECTED_ERROR

(2195, X'893') Se ha producido un error inesperado.

MQRC_UOW_ENLISTMENT_ERROR

(2354, X'932') Ha fallado el listado en la unidad de trabajo global.

MQRC_UOW_MIX_NOT_SUPPORTED

(2355, X'933') No se soporta la mezcla de llamadas de unidad de trabajo.

MQRC_UOW_NOT_AVAILABLE

(2255, X'8CF') La unidad de trabajo no está disponible para ser utilizada por el gestor de colas.

MQRC_WAIT_INTERVAL_ERROR

(2090, X'82A') El intervalo de espera de MQGMO no es válido.

MQRC_WRONG_GMO_VERSION

(2256, X'8D0') Se ha suministrado una versión equivocada de MQGMO.

MQRC_WRONG_MD_VERSION

(2257, X'8D1') La versión del MQMD suministrado es incorrecta.

Para obtener información detallada sobre estos códigos, consulte [Mensajes y códigos de motivo](#).

Notas de uso

1. MQCB se utiliza para definir la acción que se invocará para cada mensaje, que coincida con los criterios especificados, disponible en la cola. Cuando se procesa la acción, el mensaje se elimina de la cola y se pasa al consumidor de mensajes definido, o se proporciona un token de mensaje, que se utiliza para recuperar el mensaje.
2. MQCB se puede utilizar para definir rutinas de devolución de llamada antes de iniciar el consumo con MQCTL o se puede utilizar desde dentro de una rutina de devolución de llamada.
3. Para utilizar MQCB desde fuera de una rutina de devolución de llamada, primero debe suspender el consumo de mensajes mediante MQCTL y reanudarlo posteriormente.
4. MQCB no es compatible con el IMS adaptador.

Secuencia de devolución de llamada del consumidor de mensajes

Puede configurar un consumidor para que invoque una devolución de llamada en puntos clave durante el ciclo de vida del consumidor. Por ejemplo:

- cuando el consumidor se registra por primera vez,
- cuando se inicia la conexión,

- cuando se interrumpe la conexión y
- cuando el consumidor se da de baja, ya sea de forma explícita o implícita mediante un MQCLOSE.

Tabla 577. Definiciones de los verbos MQCTL	
Verbo	Significado
MQCTL(INICIO)	Llamada MQCTL utilizando la operación MQOP_START
MQCTL (STOP)	Llamada MQCTL utilizando la operación MQOP_STOP
MQCTL(WAIT)	Llamada MQCTL utilizando la operación MQOP_START_WAIT

Esto permite al consumidor mantener el estado asociado con el consumidor. Cuando una aplicación solicita una devolución de llamada, las reglas para la invocación del consumidor son las siguientes:

REGISTRAR

Siempre es el primer tipo de invocación de la devolución de llamada.

Siempre se invoca en el mismo subproceso, como la invocación MQCB(REGISTER).

START

Siempre se llama de forma sincrónica con el verbo MQCTL(START).

- Todas las devoluciones de llamada START se completan antes de que devuelva el verbo MQCTL(START).

Está en el mismo subproceso que la entrega del mensaje si se solicita THREAD_AFFINITY.

La llamada con start no está garantizada si, por ejemplo, una devolución de llamada anterior emite MQCTL(STOP) durante MQCTL(START).

STOP

Después de esta llamada, no se entregarán más mensajes ni eventos hasta que se reinicie la conexión.

Se garantiza una parada (STOP) si la aplicación fue llamada previamente para un inicio (START), un mensaje o un evento.

DEREGISTER

Siempre es el último tipo de invocación de la devolución de llamada.

Asegúrese de que su aplicación realice la inicialización y limpieza basadas en subprocesos en las devoluciones de llamada START y STOP. Puede realizar la inicialización y limpieza sin hilos con las devoluciones de llamada REGISTER y DEREGISTER.

No haga ninguna suposición sobre la vida útil y la disponibilidad del hilo que no sea lo que se indica. Por ejemplo, no confíes en que un subproceso permanezca activo más allá de la última llamada a DEREGISTER. Del mismo modo, cuando hayas optado por no utilizar THREAD_AFFINITY, no des por sentado que el subproceso existe cada vez que se inicia la conexión.

Si su aplicación tiene requisitos específicos en cuanto a las características del subproceso, siempre puede crear un subproceso acorde con ellos y, a continuación, utilizar MQCTL(WAIT). Esto tiene el efecto de «donar» el hilo a IBM MQ para la entrega asíncrona de mensajes.

Uso de la conexión del consumidor de mensajes

Puede configurar un consumidor para que invoque una devolución de llamada en puntos clave durante el ciclo de vida del consumidor. Por ejemplo:

- cuando el consumidor se registra por primera vez,
- cuando se inicia la conexión,
- cuando se interrumpe la conexión y
- cuando el consumidor se da de baja, ya sea de forma explícita o implícita mediante un MQCLOSE.

Tabla 578. Definiciones de los verbos MQCTL	
Verbo	Significado
MQCTL(INICIO)	Llamada MQCTL utilizando la operación MQOP_START
MQCTL (STOP)	Llamada MQCTL utilizando la operación MQOP_STOP
MQCTL(WAIT)	Llamada MQCTL utilizando la operación MQOP_START_WAIT

Esto permite al consumidor mantener el estado asociado con el consumidor. Cuando una aplicación solicita una devolución de llamada, las reglas para la invocación del consumidor son las siguientes:

REGISTRAR

Siempre es el primer tipo de invocación de la devolución de llamada.

Siempre se invoca en el mismo subproceso, como la invocación MQCB(REGISTER).

START

Siempre se llama de forma sincrónica con el verbo MQCTL(START).

- Todas las devoluciones de llamada START se completan antes de que devuelva el verbo MQCTL(START).

Está en el mismo subproceso que la entrega del mensaje si se solicita THREAD_AFFINITY.

La llamada con start no está garantizada si, por ejemplo, una devolución de llamada anterior emite MQCTL(STOP) durante MQCTL(START).

STOP

Después de esta llamada, no se entregarán más mensajes ni eventos hasta que se reinicie la conexión.

Se garantiza una parada (STOP) si la aplicación fue llamada previamente para un inicio (START), un mensaje o un evento.

DEREGISTER

Siempre es el último tipo de invocación de la devolución de llamada.

Asegúrese de que su aplicación realice la inicialización y limpieza basadas en subprocesos en las devoluciones de llamada START y STOP. Puede realizar la inicialización y limpieza sin hilos con las devoluciones de llamada REGISTER y DEREGISTER.

No haga ninguna suposición sobre la vida útil y la disponibilidad del hilo que no sea lo que se indica. Por ejemplo, no confíes en que un subproceso permanezca activo más allá de la última llamada a DEREGISTER. Del mismo modo, cuando hayas optado por no utilizar THREAD_AFFINITY, no des por sentado que el subproceso existe cada vez que se inicia la conexión.

Si su aplicación tiene requisitos específicos en cuanto a las características del subproceso, siempre puede crear un subproceso acorde con ellos y, a continuación, utilizar MQCTL(WAIT). Esto tiene el efecto de «donar» el hilo a IBM MQ para la entrega asíncrona de mensajes.

Invocación en C

```
MQCB (Hconn, Operation, CallbackDesc, Hobj, MsgDesc,
      GetMsgOpts, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQHCONN  Hconn;          /* Connection handle */
MQLONG   Operation;      /* Operation being processed */
MQCBD    CallbackDesc;   /* Callback descriptor */
MQHOBJS  Hobj;           /* Object handle */
MQMD     MsgDesc         /* Message descriptor attributes */
MQGMO    GetMsgOpts      /* Message options */
MQLONG   CompCode;       /* Completion code */
MQLONG   Reason;         /* Reason code qualifying CompCode */
```

Invocación en COBOL

```
CALL 'MQCB' USING HCONN, OPERATION, CBDESC, HOBJ, MSGDESC,  
GETMSGOPTS, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
** Connection handle  
01 HCONN PIC S9(9) BINARY.  
** Operation  
01 OPERATION PIC S9(9) BINARY.  
** Callback Descriptor  
01 CBDESC.  
COPY CMQCBDV.  
01 HOBJ PIC S9(9) BINARY.  
** Message Descriptor  
01 MSGDESC.  
COPY CMQMDV.  
** Get Message Options  
01 GETMSGOPTS.  
COPY CMQGMV.  
** Completion code  
01 COMPCODE PIC S9(9) BINARY.  
** Reason code qualifying COMPCODE  
01 REASON PIC S9(9) BINARY.
```

Invocación en PL/I

```
call MQCB(Hconn, Operation, CallbackDesc, Hobj, MsgDesc, GetMsgOpts,  
CompCode, Reason)
```

Declare los parámetros como se indica a continuación:

```
dcl Hconn          fixed bin(31); /* Connection handle */  
dcl Operation      fixed bin(31); /* Operation */  
dcl CallbackDesc   like MQCBD;    /* Callback Descriptor */  
dcl Hobj           fixed bin(31); /* Object Handle */  
dcl MsgDesc        like MQMD;     /* Message Descriptor */  
dcl GetMsgOpts     like MQGMO;    /* Get Message Options */  
dcl CompCode       fixed bin(31); /* Completion code */  
dcl Reason         fixed bin(31); /* Reason code qualifying CompCode */
```

MQCB_FUNCTION - Función de devolución de llamada

La llamada a la función MQCB_FUNCTION es la función de devolución de llamada para el manejo de eventos y el consumo asíncrono de mensajes.

La definición de la llamada MQCB_FUNCTION se proporciona únicamente para describir los parámetros que se pasan a la función de devolución de llamada. El gestor de colas no proporciona ningún punto de entrada denominado MQCB_FUNCTION.

La especificación de la función real que se va a llamar es una entrada para la llamada [MQCB](#) y se pasa a través de la estructura [MQCBD](#).

Sintaxis

MQCB_FUNCTION (*Hconn*, *MsgDesc*, *GetMsgOpts*, *Buffer*, *Context*)

Parámetros

Hconn

Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* fue devuelto por una llamada anterior a MQCONN o MQCONNEX. En z/OS para CICS aplicaciones, se puede omitir la llamada MQCONN y especificar el siguiente valor para Hconn:

MQHC_DEF_HCONN

Manejador de conexión predeterminado.

MsgDesc

Tipo: MQMD - entrada

Esta estructura describe los atributos del mensaje recuperado.

Consulte [“MQMD - Descriptor de mensaje”](#) en la página 389 para obtener los detalles.

La versión de MQMD aprobada es la misma versión que se aprobó en la convocatoria de MQCB que definió la función de consumidor.

La dirección del MQMD se pasa como caracteres nulos si se utilizó un MQGMO versión 4 para solicitar que se devolviera un identificador de mensaje en lugar de un MQMD.

Este es un campo de entrada para la función de consumo de mensajes; no es relevante para una función de controlador de eventos.

GetMsgOpts

Tipo: MQGMO - entrada

Opciones utilizadas para controlar las acciones del consumidor de mensajes. Este parámetro también contiene información adicional sobre el mensaje devuelto.

Consulte [MQGMO](#) para obtener más detalles.

La versión de MQGMO aprobada es la última versión compatible.

Este es un campo de entrada para la función de consumo de mensajes; no es relevante para una función de controlador de eventos.

Almacenamiento intermedio

Tipo: MQBYTEExBufferLength - entrada

Esta es el área que contiene los datos del mensaje.

Si no hay ningún mensaje disponible para esta llamada, o si el mensaje no contiene datos, la dirección del *Buffer* se pasa como nulos.

Este es un campo de entrada para la función de consumo de mensajes; no es relevante para una función de controlador de eventos.

Contexto

Tipo: MQCBC - entrada/salida

Esta estructura proporciona información contextual a las funciones de devolución de llamada.

Consulte [“MQCBC - Contexto de devolución de llamada”](#) en la página 227 para obtener los detalles.

Notas de uso

1. Tenga en cuenta que si sus rutinas de devolución de llamada utilizan servicios que podrían retrasar o bloquear el subproceso, por ejemplo, MQGET con espera, podrían retrasar el envío de otras devoluciones de llamada.
2. No se establece automáticamente una unidad de trabajo independiente para cada invocación de una rutina de devolución de llamada, por lo que las rutinas pueden emitir una llamada de confirmación o aplazar la confirmación hasta que se haya procesado un lote lógico de trabajo. Cuando se confirma el lote de trabajo, se confirman los mensajes de todas las funciones de devolución de llamada que se han invocado desde el último punto de sincronización.
3. Los programas invocados por CICS LINK o CICS START recuperan parámetros utilizando CICS servicios a través de objetos con nombre conocidos como contenedores de canales. Los nombres de los contenedores son los mismos que los nombres de los parámetros. Para obtener más información, consulte la documentación de CICS.

4. Las rutinas de devolución de llamada pueden emitir una llamada MQDISC, pero no para su propia conexión. Por ejemplo, si una rutina de devolución de llamada ha creado una conexión, también puede desconectarla.
5. En general, una rutina de devolución de llamada no debe depender de que se invoque desde el mismo hilo cada vez. Si es necesario, utilice MQCTLO_THREAD_AFFINITY cuando se inicie la conexión.
6. Cuando una rutina de devolución de llamada recibe un código de motivo distinto de cero, debe tomar las medidas adecuadas.
7. MQCB_FUNCTION no es compatible con el IMS adaptador.

MQCLOSE - Cerrar objeto

La llamada MQCLOSE renuncia al acceso a un objeto y es la inversa de las llamadas MQOPEN y MQSUB.

Sintaxis

MQCLOSE (*Hconn*, *Hobj*, *Opciones*, *CódigoComp*, *Motivo*)

Parámetros

Hconn

Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* fue devuelto por una llamada anterior a MQCONN o MQCONNX.

En z/OS para CICS aplicaciones, puede omitir la llamada MQCONN y especificar el siguiente valor para *Hconn*:

MQHC_DEF_HCONN

Manejador de conexión predeterminado.

Hobj

Tipo: MQHOBJ - entrada/salida

Este identificador representa el objeto que se está cerrando. El objeto puede ser de cualquier tipo. El valor de *Hobj* fue devuelto por una llamada MQOPEN anterior.

Una vez completada con éxito la llamada, el administrador de colas establece este parámetro en un valor que no es un identificador válido para el entorno. Este valor es:

MQHO_OBJETO_INUTILIZABLE

Manipulación de objeto no utilizable.

En z/OS, *Hobj* se establece en un valor que no está definido.

Opciones

Tipo: MQLONG - entrada

Este parámetro controla cómo se cierra el objeto.

Solo las colas dinámicas permanentes y las suscripciones pueden cerrarse de más de una forma, ya que deben conservarse o eliminarse; se trata de colas con el **DefinitionType** atributo que tiene el valor MQQDT_PERMANENT_DYNAMIC (véase el **DefinitionType** atributo descrito en [“Atributos para colas” en la página 841](#)). Las opciones cercanas se resumen en este tema.

Las suscripciones duraderas se pueden mantener o eliminar; se crean utilizando la llamada MQSUB con la opción MQSO_DURABLE.

Al cerrar el identificador de un destino gestionado (es decir, el **Hobj** parámetro devuelto en una llamada MQSUB que utilizó la opción MQSO_MANAGED), el gestor de colas limpia todas las publicaciones que no se hayan recuperado cuando también se haya eliminado la suscripción asociada. La suscripción se elimina utilizando la opción MQCO_REMOVE_SUB en el **Hsub** parámetro devuelto en una llamada MQSUB. Nota: MQCO_REMOVE_SUB es el comportamiento predeterminado en MQCLOSE para una suscripción no duradera.

Al cerrar un identificador a un destino no administrado, usted es responsable de limpiar la cola donde se envían las publicaciones. Cierre la suscripción utilizando primero MQCO_REMOVE_SUB y, a continuación, procese los mensajes de la cola hasta que no quede ninguno.

Debe especificar solo una opción de entre las siguientes:

Opciones de colas dinámicas: estas opciones controlan cómo se cierran las colas dinámicas permanentes.

MQCO_DELETE

La cola se elimina si se da alguna de las siguientes condiciones:

- Es una cola dinámica permanente, creada por una llamada MQOPEN anterior, y no hay mensajes en la cola ni solicitudes de obtención o inserción pendientes para la cola (ya sea para la tarea actual o cualquier otra tarea).
- Es la cola dinámica temporal creada por la llamada MQOPEN que devolvió *Hobj*. En este caso, se eliminan todos los mensajes de la cola.

En todos los demás casos, incluido el caso en el que se *Hobj* devolvió el en una llamada MQSUB, la llamada falla con el código de motivo MQRC_OPTION_NOT_VALID_FOR_TYPE y el objeto no se elimina.

En z/OS, si la cola es una cola dinámica que se ha eliminado lógicamente y este es el último identificador para ella, la cola se elimina físicamente. Para conocer detalles, consulte [“Notas de uso”](#) en la página 656.

MQCO_DELETE_PURGE

La cola se elimina y se borran todos los mensajes que contiene si se da alguna de las siguientes condiciones:

- Es una cola dinámica permanente, creada por una llamada MQOPEN anterior, y no hay solicitudes de obtención o inserción pendientes sin confirmar para la cola (ni para la tarea actual ni para ninguna otra tarea).
- Es la cola dinámica temporal creada por la llamada MQOPEN que devolvió *Hobj*.

En todos los demás casos, incluido el caso en el que se *Hobj* devolvió el en una llamada MQSUB, la llamada falla con el código de motivo MQRC_OPTION_NOT_VALID_FOR_TYPE y el objeto no se elimina.

Tabla 579. Opciones de cierre para diferentes tipos de objetos			
Tipo de objeto o cola	MQCO_NONE	MQCO_DELETE	MQCO_DELETE_PURGE
Objeto distinto de una cola	Retenido	No válido	No válido
Cola predefinida	Retenido	No válido	No válido
cola dinámica permanente	Retenido	Se elimina si está vacío y no hay actualizaciones pendientes	Mensajes eliminados; cola eliminada si no hay actualizaciones pendientes
Cola dinámica temporal (llamada emitida por el creador de la cola)	Suprimida	Suprimida	Suprimida
Cola dinámica temporal (llamada no emitida por el creador de la cola)	Retenido	No válido	No válido
Lista de distribución	Retenido	No válido	No válido
Destino de suscripción gestionado	Retenido	No válido	No válido

Tabla 579. Opciones de cierre para diferentes tipos de objetos (continuación)			
Tipo de objeto o cola	MQCO_NONE	MQCO_DELETE	MQCO_DELETE_PURGE
Lista de distribución (se ha eliminado la suscripción)	Mensajes eliminados; cola eliminada	No válido	No válido

Opciones de cierre de suscripción: estas opciones controlan si las suscripciones duraderas se eliminan cuando se cierra el identificador y si se limpian las publicaciones que aún esperan ser leídas por la aplicación. Estas opciones solo son válidas para su uso con un identificador de objeto devuelto en el **Hsub** parámetro de una llamada MQSUB.

MQCO_MANTENER_SUB

La gestión de la suscripción está cerrada, pero la suscripción realizada se mantiene. Las publicaciones se siguen enviando a la dirección especificada en la suscripción. Esta opción solo es válida si la suscripción se realizó con la opción MQSO_DURABLE.

MQCO_KEEP_SUB es el valor predeterminado si la suscripción es duradera

MQCO_REMOVE_SUB

Se elimina la suscripción y se cierra el identificador de la suscripción.

El **Hobj** parámetro de la llamada MQSUB no se invalida al cerrar el **Hsub** parámetro y puede seguir utilizándose para MQGET o MQCB para recibir las publicaciones restantes. Cuando el **Hobj** parámetro de la llamada MQSUB también se cierra, si se trataba de un destino gestionado, se eliminan las publicaciones no recuperadas.

MQCO_REMOVE_SUB es el valor predeterminado si la suscripción no es duradera.

La finalización satisfactoria de MQCO_REMOVE_SUB no significa que la acción haya finalizado. Para comprobar que esta llamada se ha completado, consulte el paso DELETE SUB en [Comprobación de que los comandos asíncronos para redes distribuidas han finalizado](#).

Estas opciones de cancelación de la suscripción se resumen en las siguientes tablas.

Tabla 580. Opciones para cerrar una suscripción duradera pero conservar la suscripción	
Tarea	Opción de cancelación de suscripción
Mantener las publicaciones en un identificador MQOPENed	MQCO_MANTENER_SUB
Eliminar publicaciones en un identificador MQOPENed	Acción no permitida
Mantener las publicaciones en un identificador MQSO_MANAGED	MQCO_MANTENER_SUB
Eliminar publicaciones en un identificador MQSO_MANAGED	Acción no permitida

Para cancelar la suscripción, ya sea cerrando un identificador de suscripción duradero y cancelando la suscripción o cerrando un identificador de suscripción no duradero, utilice las siguientes opciones de cierre de suscripción:

Tabla 581. Opciones para darse de baja	
Tarea	Opción de cancelación de suscripción
Mantener las publicaciones en un identificador MQOPENed	MQCO_REMOVE_SUB
Eliminar publicaciones en un identificador MQOPENed	Acción no permitida
Mantener las publicaciones en un identificador MQSO_MANAGED	MQCO_REMOVE_SUB

Opciones de lectura anticipada : Las siguientes opciones controlan lo que ocurre con los mensajes no persistentes que se han enviado al cliente antes de que una aplicación los haya solicitado y que aún no han sido consumidos por la aplicación. Estos mensajes se almacenan en el búfer de lectura anticipada del cliente a la espera de que la aplicación los solicite y pueden descartarse o consumirse de la cola antes de que se complete MQCLOSE.

MQCO_IMMEDIATO

El objeto se cierra inmediatamente y cualquier mensaje que se haya enviado al cliente antes de que una aplicación lo solicite se descarta y no está disponible para ser consumido por ninguna aplicación. Éste es el valor predeterminado.

MQCO QUIESCE

Se realiza una solicitud para cerrar el objeto, pero si algún mensaje que se haya enviado al cliente antes de que una aplicación lo solicite sigue residiendo en el búfer de lectura anticipada del cliente, la llamada MQCLOSE devuelve una advertencia de MQRC_READ_AHEAD_MSGS y el identificador del objeto sigue siendo válido.

La aplicación puede seguir utilizando el identificador del objeto para recuperar mensajes hasta que no haya más disponibles y, a continuación, volver a cerrar el objeto. Ya no se envían más mensajes al cliente antes de que una aplicación los solicite, la lectura anticipada ahora está desactivada.

Se recomienda a las aplicaciones que utilicen MQCO QUIESCE en lugar de intentar llegar a un punto en el que no haya más mensajes en el búfer de lectura anticipada del cliente, ya que podría llegar un mensaje entre la última llamada a MQGET y la siguiente MQCLOSE, que se descartaría si se utilizara MQCO_IMMEDIATE.

Si se emite un MQCLOSE con MQCO QUIESCE desde una función de devolución de llamada asíncrona, se aplica el mismo comportamiento de lectura anticipada de mensajes. Si se devuelve la advertencia MQRC_READ_AHEAD_MSGS, la función de devolución de llamada se invoca al menos una vez más. Cuando el último mensaje restante que se leyó por adelantado se ha pasado a la función de devolución de llamada, el campo MQCBC ConsumerFlags se establece en MQCBCF_READA_BUFFER_EMPTY.

Opción predeterminada: Si no necesita ninguna de las opciones descritas anteriormente, puede utilizar la siguiente opción:

MQCO_NONE

No se requiere ningún procesamiento de cierre opcional.

Esto debe especificarse para:

- Objetos distintos de las colas
- Colas predefinidas
- Colas dinámicas temporales (pero solo en aquellos casos en los que *Hobj* no es el identificador devuelto por la llamada MQOPEN que creó la cola).
- Listas de distribución

En todos los casos anteriores, el objeto se conserva y no se elimina.

Si se especifica esta opción para una cola dinámica temporal:

- La cola se elimina, si fue creada por la llamada MQOPEN que devolvió *Hobj* ; cualquier mensaje que se encuentre en la cola se purga.
- En todos los demás casos, la cola (y cualquier mensaje que contenga) se conserva.

Si se especifica esta opción para una cola dinámica permanente, la cola se conserva y no se elimina.

En z/OS, si la cola es una cola dinámica que se ha eliminado lógicamente y este es el último identificador para ella, la cola se elimina físicamente. Para conocer detalles, consulte [“Notas de uso” en la página 656](#).

CompCode

Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_WARNING

Aviso (finalización parcial).

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

Los códigos de motivo enumerados son los que el gestor de colas puede devolver para el **Reason** parámetro.

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_WARNING:

MQRC_INCOMPLETE_GROUP

(2241, X'8C1') El grupo de mensajes no está completo.

MQRC_INCOMPLETE_MSG

(2242, X'8C2') El mensaje lógico no está completo.

MQRC_LEER_POR_ADELANTE_MENSAJES

(nnnn, X'xxx') El cliente ha leído mensajes anticipados que aún no han sido consumidos por la aplicación.

Si *CompCode* es MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'89C') El adaptador no está disponible.

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

MQRC_API_EXIT_ERROR

(2374, X'946') La salida de la API ha fallado.

MQRC_API_EXIT_LOAD_ERROR

(2183, X'887') No se ha podido cargar la salida de la API.

MQRC_ASID_MISMATCH

(2157, X'86D') Los ASID primario e inicial varían.

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') La llamada MQI ha entrado antes de completar la llamada anterior.

MQRC_CF_NO_DISPONIBLE

(2345, X'929') Instalación de acoplamiento no disponible.

MQRC_CF_STRUC_FAILED

(2373, X'945') La estructura de recurso de acoplamiento ha fallado.

MQRC_CF_STRUC_IN_USE

(2346, X'92A') La estructura de recurso de acoplamiento se está utilizando.

MQRC_CICS_WAIT_FAILED

(2140, X'85C') Solicitud de espera rechazada por CICS.

MQRC_CONNECTION_BROKEN

(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

MQRC_CONNECTION_NOT_AUTHORIZED

(2217, X'8A9') No tiene autorización para la conexión.

MQRC_CONNECTION_STOPPING

(2203, X'89B') La conexión está concluyendo.

MQRC_DB2_NOT_AVAILABLE

(2342, X'926') Db2 subsistema no disponible.

MQRC_HCONN_ERROR

(2018, X'7E2') El manejador de conexión no es válido.

MQRC_HOBJ_ERROR

(2019, X'7E3') El manejador de objeto no es válido.

MQRC_NOT_AUTHORIZED

(2035, X'7F3') Acceso no autorizado.

MQRC_OBJECT_DAMAGED

(2101, X'835') El objeto se ha dañado.

MQRC_OPCIÓN_NO_VÁLIDA_PARA_EL_TIPO

(2045, X'7FD') En una llamada MQOPEN o MQCLOSE: opción no válida para el tipo de objeto.

MQRC_OPTIONS_ERROR

(2046, X'7FE') Las opciones no son válidas o no son coherentes.

MQRC_PAGESET_ERROR

(2193, X'891') Error al acceder al conjunto de datos del conjunto de páginas.

MQRC_Q_MGR_NAME_ERROR

(2058, X'80A') El nombre del gestor de colas no es válido o no se conoce.

MQRC_Q_MGR_NOT_AVAILABLE

(2059, X'80B') El gestor de colas no está disponible para efectuar la conexión.

MQRC_Q_MGR_STOPPING

(2162, X'872') El gestor de colas está concluyendo.

MQRC_Q_NOT_EMPTY

(2055, X'807') La cola contiene uno o más mensajes o solicitudes de inserción o obtención no confirmadas.

MQRC_RESOURCE_PROBLEM

(2102, X'836') No hay disponibles suficientes recursos del sistema.

MQRC_ERROR_DE_SEGURIDAD

(2063, X'80F') Se ha producido un error de seguridad.

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_SUPPRESSED_BY_EXIT

(2109, X'83D') El programa de salida ha suprimido la llamada.

MQRC_UNEXPECTED_ERROR

(2195, X'893') Se ha producido un error inesperado.

Para obtener información detallada sobre estos códigos, consulte [Mensajes y códigos de motivo](#).

Notas de uso

1. Cuando una aplicación emite la llamada MQDISC, o finaliza de forma normal o anómala, cualquier objeto que haya sido abierto por la aplicación y siga abierto se cierra automáticamente con la opción MQCO_NONE.
2. Los siguientes puntos se aplican si el objeto que se cierra es una *cola* :
 - Si las operaciones en la cola se realizan como parte de una unidad de trabajo, la cola se puede cerrar antes o después de que se produzca el punto de sincronización sin que ello afecte al resultado del punto de sincronización. Si se activa la cola, realizar una reversión antes de cerrarla puede

provocar que se envíe un mensaje de activación. Para obtener más información sobre los mensajes de activación, consulte [Propiedades de los mensajes de activación](#).

- Si la cola se abrió con la opción MQOO_BROWSE, se destruye el cursor de exploración. Si la cola se vuelve a abrir con la opción MQOO_BROWSE, se crea un nuevo cursor de exploración (véase [MQOO_BROWSE](#)).
 - Si un mensaje está bloqueado actualmente para este identificador en el momento de la llamada MQCLOSE, se libera el bloqueo (véase [MQGMO_LOCK](#)).
 - Si hay una solicitud MQGET con la opción MQGMO_SET_SIGNAL pendiente contra el identificador de cola que se está cerrando z/OS, la solicitud se cancela (consulte [MQGMO_SET_SIGNAL](#)). Las solicitudes de señal para la misma cola pero presentadas contra diferentes identificadores (*Hobj*) no se ven afectadas (a menos que se esté eliminando una cola dinámica, en cuyo caso también se cancelan).
3. Los siguientes puntos se aplican si el objeto que se cierra es una *cola dinámica* (ya sea permanente o temporal):
- Para una cola dinámica, puede especificar las opciones MQCO_DELETE y MQCO_DELETE_PURGE independientemente de las opciones especificadas en la llamada MQOPEN correspondiente.
 - Cuando se elimina una cola dinámica, se cancelan todas las llamadas MQGET con la opción MQGMO_WAIT pendientes en la cola y se devuelve el código de motivo MQRC_Q_DELETED. Véase [MQGMO_WAIT](#).

Aunque las aplicaciones no pueden acceder a una cola eliminada, la cola no se elimina del sistema y los recursos asociados no se liberan hasta que se hayan cerrado todos los identificadores que hacen referencia a la cola y todas las unidades de trabajo que afectan a la cola se hayan confirmado o revertido.

En z/OS, una cola que se ha eliminado lógicamente pero que aún no se ha eliminado del sistema impide la creación de una nueva cola con el mismo nombre que la cola eliminada; en este caso, la llamada MQOPEN falla con el código de motivo MQRC_NAME_IN_USE. Además, dicha cola se puede seguir mostrando mediante comandos MQSC, aunque las aplicaciones no puedan acceder a ella.

- Cuando se elimina una cola dinámica permanente, si el *Hobj* identificador especificado en la llamada MQCLOSE no es el que devolvió la llamada MQOPEN que creó la cola, se comprueba que el identificador de usuario utilizado para validar la llamada MQOPEN esté autorizado para eliminar la cola. Si se especificó la opción MQOO_ALTERNATE_USER_AUTHORITY en la llamada MQOPEN, el identificador de usuario que se comprueba es el *AlternateUserId*.

Esta comprobación no se realiza si:

- El identificador especificado es el devuelto por la llamada MQOPEN que creó la cola.
 - La cola que se está eliminando es una cola dinámica temporal.
- Cuando se cierra una cola dinámica temporal, si el *Hobj* identificador especificado en la llamada MQCLOSE es el que devolvió la llamada MQOPEN que creó la cola, esta se elimina. Esto ocurre independientemente de las opciones de cierre especificadas en la llamada MQCLOSE. Si hay mensajes en la cola, se descartan; no se generan mensajes de informe.

Si hay unidades de trabajo no comprometidas que afectan a la cola, la cola y sus mensajes se siguen eliminando, pero las unidades de trabajo no fallan. Sin embargo, como se ha descrito anteriormente, los recursos asociados a las unidades de trabajo no se liberan hasta que cada una de las unidades de trabajo se ha confirmado o cancelado.

4. Los siguientes puntos se aplican si el objeto que se cierra es una *lista de distribución* :

- La única opción válida para cerrar una lista de distribución es MQCO_NONE; la llamada falla con el código de motivo MQRC_OPTIONS_ERROR o MQRC_OPTION_NOT_VALID_FOR_TYPE si se especifican otras opciones.
- Cuando se cierra una lista de distribución, no se devuelven los códigos de finalización individuales ni los códigos de motivo para las colas de la lista; solo están disponibles los parámetros **CompCode Reason** y de la llamada con fines de diagnóstico.

Si se produce un fallo al cerrar una de las colas, el gestor de colas continúa con el procesamiento e intenta cerrar las colas restantes de la lista de distribución. Los parámetros **CompCode Reason** y de la llamada se configuran para devolver información que describe el fallo. Es posible que el código de finalización sea MQCC_FAILED, aunque la mayoría de las colas se hayan cerrado correctamente. No se identifica la cola en la que se produjo el error.

Si se produce un fallo en más de una cola, no se define qué fallo se notifica en los parámetros **CompCode Reason** y.

Invocación en C

```
MQCLOSE (Hconn, &Hobj, Options, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQHCONN  Hconn;      /* Connection handle */
MQHOBJ    Hobj;      /* Object handle */
MQLONG    Options;   /* Options that control the action of MQCLOSE */
MQLONG    CompCode;  /* Completion code */
MQLONG    Reason;    /* Reason code qualifying CompCode */
```

Invocación en COBOL

```
CALL 'MQCLOSE' USING HCONN, HOBJ, OPTIONS, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
** Connection handle
01 HCONN      PIC S9(9) BINARY.
** Object handle
01 HOBJ       PIC S9(9) BINARY.
** Options that control the action of MQCLOSE
01 OPTIONS    PIC S9(9) BINARY.
** Completion code
01 COMPCODE   PIC S9(9) BINARY.
** Reason code qualifying COMPCODE
01 REASON     PIC S9(9) BINARY.
```

Invocación en PL/I

```
call MQCLOSE (Hconn, Hobj, Options, CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```
dcl Hconn      fixed bin(31); /* Connection handle */
dcl Hobj       fixed bin(31); /* Object handle */
dcl Options    fixed bin(31); /* Options that control the action of
                               MQCLOSE */
dcl CompCode   fixed bin(31); /* Completion code */
dcl Reason     fixed bin(31); /* Reason code qualifying CompCode */
```

Invocación en ensamblador de alto nivel

```
CALL MQCLOSE,(HCONN,HOBJ,OPTIONS,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

```
HCONN      DS F Connection handle
HOBJ       DS F Object handle
OPTIONS    DS F Options that control the action of MQCLOSE
COMPCODE   DS F Completion code
REASON     DS F Reason code qualifying COMPCODE
```

Invocación en Visual Basic

```
MQCLOSE Hconn, Hobj, Options, CompCode, Reason
```

Declare los parámetros como se indica a continuación:

```
Dim Hconn As Long 'Connection handle'  
Dim Hobj As Long 'Object handle'  
Dim Options As Long 'Options that control the action of MQCLOSE'  
Dim CompCode As Long 'Completion code'  
Dim Reason As Long 'Reason code qualifying CompCode'
```

MQCMIT - Confirmar cambios

La llamada MQCMIT indica al gestor de colas que la aplicación ha alcanzado un punto de sincronización y que todas las operaciones de obtención y envío de mensajes que se hayan producido desde el último punto de sincronización deben hacerse permanentes.

Los mensajes incluidos como parte de una unidad de trabajo se ponen a disposición de otras aplicaciones; los mensajes recuperados como parte de una unidad de trabajo se eliminan.

-  En z/OS, la llamada solo la usan los programas por lotes (incluidos los programas IMS DL/I por lotes).

Sintaxis

MQCMIT (*Hconn*, *CódigoComp*, *Motivo*)

Parámetros

Hconn

Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* fue devuelto por una llamada anterior a MQCONN o MQCONNX.

CompCode

Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_WARNING

Aviso (finalización parcial).

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

Los códigos de motivo enumerados son los que el gestor de colas puede devolver para el **Reason** parámetro.

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_WARNING:

MQRC_BACKED_OUT

(2003, X'7D3') Unidad de trabajo restituida.

MQRC_RESULTADO_PENDIENTE

(2124, X'84C') El resultado de la operación de confirmación está pendiente.

Si *CompCode* es MQCC_FAILED:

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

MQRC_API_EXIT_ERROR

(2374, X'946') La salida de la API ha fallado.

MQRC_ASID_MISMATCH

(2157, X'86D') Los ASID primario e inicial varían.

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') La llamada MQI ha entrado antes de completar la llamada anterior.

MQRC_CALL_INTERRUPTED

(2549, X'9F5') MQPUT o MQCMIT se interrumpieron y el proceso de reconexión no puede restablecer un resultado definitivo.

MQRC_CF_STRUC_IN_USE

(2346, X'92A') La estructura de recurso de acoplamiento se está utilizando.

MQRC_CONNECTION_BROKEN

(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

MQRC_ERROR_DE_ENTORNO

(2012, X'7DC') Llamada no válida en el entorno.

MQRC_HCONN_ERROR

(2018, X'7E2') El manejador de conexión no es válido.

MQRC_OBJECT_DAMAGED

(2101, X'835') El objeto se ha dañado.

MQRC_RESULTADO_MIXTO

(2123, X'84B') El resultado de la operación de confirmación o retroceso es mixto.

MQRC_Q_MGR_STOPPING

(2162, X'872') El gestor de colas está concluyendo.

MQRC_RECONNECT_FAILED

(2548, X'9F4') Después de volver a conectarse, se produjo un error al restablecer los identificadores para una conexión reconectable.

MQRC_RESOURCE_PROBLEM

(2102, X'836') No hay disponibles suficientes recursos del sistema.

MQRC_STORAGE_MEDIUM_FULL

(2192, X'890') El soporte de almacenamiento externo está lleno.

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_UNEXPECTED_ERROR

(2195, X'893') Se ha producido un error inesperado.

Para obtener información detallada sobre estos códigos, consulte [Mensajes y códigos de motivo](#).

Notas de uso

1. Utilice esta llamada solo cuando el propio gestor de colas coordine la unidad de trabajo. Este puede ser:
 - Una unidad de trabajo local, donde los cambios afectan únicamente a IBM MQ los recursos.
 - Una unidad de trabajo global, donde los cambios pueden afectar a recursos que pertenecen a otros gestores de recursos, además de afectar a IBM MQ los propios recursos.

Para obtener más información sobre las unidades de trabajo locales y globales, consulte [“MQBEGIN - Comenzar unidad de trabajo”](#) en la página 633.

2. En entornos en los que el gestor de colas no coordina la unidad de trabajo, se debe utilizar la llamada de confirmación adecuada en lugar de MQCMIT. El entorno también podría admitir una confirmación implícita causada por el cierre normal de la aplicación.
 - En z/OS, utilice las siguientes llamadas:
 - Los programas por lotes (incluidos los IMS programas DL/I por lotes) pueden utilizar la llamada MQCMIT si la unidad de trabajo solo IBM MQ afecta a los recursos. Sin embargo, si la unidad de trabajo afecta tanto IBM MQ a recursos propios como a recursos que pertenecen a otros gestores de recursos (por ejemplo, Db2), utilice la llamada SRRCMIT proporcionada por el Servicio de z/OS Recursos Recuperables (RRS). La llamada SRRCMIT confirma los cambios en los recursos que pertenecen a los gestores de recursos que se han habilitado para la coordinación RRS.
 - CICS Las aplicaciones deben utilizar el EXEC CICS SYNCPOINT comando para confirmar explícitamente la unidad de trabajo. Por otra parte, finalizar la transacción da lugar a una confirmación implícita de la unidad de trabajo. La llamada MQCMIT no se puede utilizar para CICS aplicaciones.
 - IMS Las aplicaciones (excepto los programas DL/I por lotes) deben utilizar IMS llamadas como GU y CHPK para confirmar la unidad de trabajo. La llamada MQCMIT no se puede utilizar para IMS aplicaciones (excepto programas DL/I por lotes).
 - En IBM i, utilice esta llamada para las unidades de trabajo locales coordinadas por el gestor de colas. Esto significa que no debe existir una definición de compromiso a nivel de trabajo, es decir, no debe haberse emitido el STRCMTCTL comando con el **CMTSCOPE(*JOB)** parámetro para el trabajo.
3. Si una aplicación finaliza con cambios no confirmados en una unidad de trabajo, la disposición de dichos cambios depende de si la aplicación finaliza de forma normal o anómala. Consulte [las notas de uso de MQDISC](#) para obtener más detalles.
4. Cuando una aplicación coloca u obtiene mensajes en grupos o segmentos de mensajes lógicos, el gestor de colas conserva la información relativa al grupo de mensajes y al mensaje lógico de las últimas llamadas MQPUT y MQGET realizadas con éxito. Esta información está asociada al identificador de cola e incluye datos tales como:
 - Los valores de los campos *GroupId*, *MsgSeqNumber*, *MsgFlags Offset*, y en MQMD.
 - Si el mensaje forma parte de una unidad de trabajo.
 - Para la llamada MQPUT: si el mensaje es persistente o no persistente.

Cuando se confirma una unidad de trabajo, el gestor de colas conserva la información del grupo y del segmento, y la aplicación puede seguir enviando o recibiendo mensajes en el grupo de mensajes actual o en el mensaje lógico.

Conservar la información del grupo y del segmento cuando se confirma una unidad de trabajo permite a la aplicación distribuir un grupo de mensajes grande o un mensaje lógico grande compuesto por muchos segmentos entre varias unidades de trabajo. El uso de varias unidades de trabajo resulta ventajoso si el gestor de colas local solo dispone de un almacenamiento de colas limitado. Sin embargo, la aplicación debe conservar suficiente información para reiniciar el envío o la recepción de mensajes en el punto correcto si se produce un fallo del sistema. Para obtener más información sobre cómo reiniciar en el punto correcto después de un fallo del sistema, consulte [MQPMO_LOGICAL_ORDER](#) y [MQGMO_LOGICAL_ORDER](#).

Las notas de uso restantes solo se aplican cuando el gestor de colas coordina las unidades de trabajo:

5. Una unidad de trabajo tiene el mismo alcance que un identificador de conexión; todas IBM MQ las llamadas que afectan a una unidad de trabajo concreta deben realizarse utilizando el mismo identificador de conexión. Las llamadas realizadas utilizando un identificador de conexión diferente (por ejemplo, llamadas realizadas por otra aplicación) afectan a una unidad de trabajo diferente. Consulte el **Hconn** parámetro descrito en MQCONN para obtener información sobre el ámbito de los identificadores de conexión.

6. Solo los mensajes que se colocaron o recuperaron como parte de la unidad de trabajo actual se ven afectados por esta llamada.
7. Una aplicación de larga duración que emite llamadas MQGET, MQPUT o MQPUT1 dentro de una unidad de trabajo, pero que nunca emite una llamada de confirmación o de retroceso, puede llenar las colas con mensajes que no están disponibles para otras aplicaciones. Para evitarlo, el administrador debe establecer el atributo del **MaxUncommittedMsgs** gestor de colas en un valor lo suficientemente bajo como para impedir que las aplicaciones descontroladas llenen las colas, pero lo suficientemente alto como para permitir que las aplicaciones de mensajería esperadas funcionen correctamente.
8.  En AIX, Linux, and Windows los sistemas, si el **Reason** parámetro es MQRC_CONNECTION_BROKEN (con un *CompCode* valor de MQCC_FAILED) o MQRC_UNEXPECTED_ERROR, es posible que la unidad de trabajo se haya confirmado correctamente.
9. Cuando una aplicación ha incluido la inserción de un mensaje persistente dentro de la unidad de trabajo, y la llamada MQPUT o MQPUT1 ha utilizado la opción MQPMO_ASYNC_RESPONSE, y la inserción ha fallado, entonces MQCMIT devuelve MQCC_FAILED, MQRC_BACKED_OUT. Consulte la descripción de MQPMO_ASYNC_RESPONSE en [Detalles del campo para MQPMO](#).

En un escenario similar, para un mensaje no persistente dentro de la unidad de trabajo, MQCMIT devuelve MQCC_OK, MQRC_NONE.

En el caso de que haya varias operaciones de envío (put) y se haya utilizado MQPMO_ASYNC_RESPONSE, si falla el envío de cualquier mensaje persistente, la llamada MQCMIT devolverá MQCC_FAILED, MQRC_BACKED_OUT.

Invocación en C

```
MQCMIT (Hconn, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQHCONN  Hconn;      /* Connection handle */
MQLONG   CompCode;   /* Completion code */
MQLONG   Reason;     /* Reason code qualifying CompCode */
```

Invocación en COBOL

```
CALL 'MQCMIT' USING HCONN, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
** Connection handle
01 HCONN      PIC S9(9) BINARY.
** Completion code
01 COMPCODE   PIC S9(9) BINARY.
** Reason code qualifying COMPCODE
01 REASON     PIC S9(9) BINARY.
```

Invocación en PL/I

```
call MQCMIT (Hconn, CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```
dcl Hconn      fixed bin(31); /* Connection handle */
dcl CompCode   fixed bin(31); /* Completion code */
dcl Reason     fixed bin(31); /* Reason code qualifying CompCode */
```

Invocación en ensamblador de alto nivel

```
CALL MQCMIT,(HCONN,COMP CODE,REASON)
```

Declare los parámetros como se indica a continuación:

HCONN	DS	F	Connection handle
COMP CODE	DS	F	Completion code
REASON	DS	F	Reason code qualifying COMP CODE

Invocación en Visual Basic

```
MQCMIT Hconn, CompCode, Reason
```

Declare los parámetros como se indica a continuación:

```
Dim Hconn As Long 'Connection handle'  
Dim CompCode As Long 'Completion code'  
Dim Reason As Long 'Reason code qualifying CompCode'
```

MQCONN - Conectar gestor de colas

La llamada MQCONN conecta un programa de aplicación con un gestor de colas.

Proporciona un identificador de conexión del gestor de colas, que la aplicación utiliza en las llamadas posteriores de puesta en cola de mensajes.

- En z/OS, CICS las aplicaciones no tienen que emitir esta llamada. Estas aplicaciones se conectan automáticamente al gestor de colas al que está conectado el CICS sistema. Sin embargo, las llamadas MQCONN y MQDISC siguen siendo aceptadas desde CICS aplicaciones.
- En IBM i, las aplicaciones deben utilizar la llamada MQCONN o MQCONNX para conectarse al gestor de colas, y la llamada MQDISC para desconectarse del gestor de colas.

No se puede establecer una conexión de cliente en una instalación solo de servidor, ni una conexión local en una instalación solo de cliente.

Sintaxis

MQCONN (*Nombre del gestor de calidad, Hconn, Código de compensación, Motivo*)

Parámetros

QMgrName

Tipo: MQCHAR48 - entrada

Este es el nombre del gestor de colas al que la aplicación desea conectarse. El nombre puede contener los siguientes caracteres:

- Caracteres alfabéticos mayúsculos (de la A a la Z)
- Letras minúsculas (de la a a la z)
- Dígitos numéricos (del 0 al 9)
- Punto (.), barra inclinada (/), guión bajo (_), porcentaje (%)

El nombre no debe contener espacios en blanco al principio ni en el interior, pero puede contener espacios en blanco al final. Se puede utilizar un carácter nulo para indicar el final de los datos significativos en el nombre; el carácter nulo y cualquier carácter que le siga se tratan como espacios en blanco. Las siguientes restricciones se aplican en los entornos indicados:

- En los sistemas que utilizan EBCDIC Katakana, no se pueden utilizar caracteres en minúscula.
- En z/OS, los nombres que comienzan o terminan con un guión bajo no pueden ser procesados por las operaciones y los paneles de control. Por este motivo, evite este tipo de nombres.

- En IBM i, incluya entre comillas los nombres que contengan caracteres en minúscula, barra inclinada o porcentaje cuando se especifiquen en los comandos. No especifique estas comillas en el **QMGrName** parámetro.

Si el nombre está compuesto únicamente por espacios en blanco, se utiliza el nombre del gestor de colas *predeterminado*. Sin embargo, tenga en cuenta el uso de nombres de gestores de colas en blanco descrito en la sección sobre IBM MQ MQI client aplicaciones.

El nombre especificado para *QMGrName* debe ser el nombre de un gestor de colas *conectable* o, si se utilizan grupos de gestores de colas, el nombre del grupo de gestores de colas.

En z/OS, los gestores de colas a los que es posible conectarse vienen determinados por el entorno:

- Para CICS, solo puede utilizar el gestor de colas al que está conectado el CICS sistema. El **QMGrName** parámetro debe especificarse igualmente, pero su valor se ignora; los caracteres en blanco son una opción adecuada.
- Para IMS, solo se pueden conectar los gestores de colas que figuran en la tabla de definición del subsistema (CSQQDEFV) y en la tabla SSM de IMS (véase la nota de uso [6](#)).
- Para z/OS batch y TSO, solo se pueden conectar los gestores de colas que residen en el mismo sistema que la aplicación (véase la nota de uso [6](#)).

Grupos para compartir colas : En sistemas en los que existen varios gestores de colas y están configurados para formar un grupo de intercambio de colas, se puede especificar el nombre del grupo de intercambio de colas en *QMGrName* lugar del nombre de un gestor de colas. Esto permite a la aplicación conectarse a *cualquier* gestor de colas que esté disponible en el grupo de intercambio de colas y que se encuentre en la misma z/OS imagen que la aplicación. El sistema también se puede configurar para que, al utilizar un espacio en blanco, *QMGrName* se conecte al grupo de intercambio de colas en lugar de al gestor de colas predeterminado.

Si *QMGrName* especifica el nombre del grupo de uso compartido de colas, pero también hay un gestor de colas con ese nombre en el sistema, se establece la conexión con este último en lugar del primero. Solo si falla esa conexión, se intentará conectarse a uno de los gestores de colas del grupo de uso compartido de colas.

Si la conexión se establece correctamente, puede utilizar el identificador devuelto por la llamada MQCONN o MQCONNX para acceder a *todos* los recursos (tanto compartidos como no compartidos) que pertenecen al gestor de colas al que se ha establecido la conexión. El acceso a estos recursos está sujeto a los controles de autorización habituales.

Si la aplicación emite dos llamadas MQCONN o MQCONNX para establecer conexiones simultáneas, y una o ambas llamadas especifican el nombre del grupo de intercambio de colas, la segunda llamada devuelve el código de finalización MQCC_WARNING y el código de motivo MQRC_ALREADY_CONNECTED cuando se conecta al mismo gestor de colas que la primera llamada.

Los grupos de compartición de colas solo se admiten en z/OS. La conexión a un grupo de uso compartido de colas solo es compatible en entornos por lotes, lotes RRS y TSO. CICS Para CICS, solo puede utilizar el grupo de uso compartido de colas al que está conectado el CICS sistema. Aún debe especificar el **QMGrName** parámetro, pero su valor se ignora; los caracteres en blanco son una opción adecuada.



Atención: IMS No se puede conectar a un grupo de uso compartido de colas.

IBM MQ MQI client Aplicaciones : Para IBM MQ MQI client las aplicaciones, se intenta establecer una conexión para cada definición de canal de conexión de cliente con el nombre del gestor de colas especificado, hasta que se establece una conexión con éxito. Sin embargo, el gestor de colas debe tener el mismo nombre que el nombre especificado. Si se especifica un nombre completamente en blanco, se prueba cada canal de conexión de cliente con un nombre de gestor de colas completamente en blanco hasta que uno tenga éxito; en este caso, no se comprueba el nombre real del gestor de colas.

IBM MQ Las aplicaciones cliente no son compatibles con z/OS, pero z/OS puede actuar como un IBM MQ servidor, al que pueden conectarse IBM MQ las aplicaciones cliente.

IBM MQ MQI client Grupos de administradores de colas : si el nombre especificado comienza con un asterisco (*), es posible que el administrador de colas al que se establece la conexión tenga un nombre diferente al especificado por la aplicación. El nombre especificado (sin el asterisco) define un *grupo* de gestores de colas que son aptos para la conexión. La implementación selecciona uno del grupo probando cada uno por turno hasta encontrar uno con el que se pueda establecer una conexión. El orden en el que se intentan las conexiones está influenciado por el peso del canal del cliente y los valores de afinidad de conexión de los canales candidatos. Si ninguno de los gestores de colas del grupo está disponible para la conexión, la llamada falla. Cada gestor de colas se prueba una sola vez. Si solo se especifica un asterisco para el nombre, se utiliza un grupo de gestores de colas predeterminado definido por la implementación.

Los grupos de gestores de colas solo son compatibles con aplicaciones que se ejecutan en un entorno de cliente de MQ; la llamada falla si una aplicación que no es cliente especifica un nombre de gestor de colas que comienza con un asterisco. Un grupo se define proporcionando varias definiciones de canales de conexión de clientes con el mismo nombre de gestor de colas (el nombre especificado sin el asterisco), para comunicarse con cada uno de los gestores de colas del grupo. El grupo predeterminado se define proporcionando una o más definiciones de canales de conexión de cliente, cada una con un nombre de gestor de colas en blanco (especificar un nombre completamente en blanco tiene el mismo efecto que especificar un solo asterisco como nombre para una aplicación cliente).

Después de conectarse a un gestor de colas de un grupo, una aplicación puede especificar espacios en blanco de la forma habitual en los campos de nombre del gestor de colas de los descriptores de mensajes y objetos para indicar el nombre del gestor de colas al que se ha conectado la aplicación (el *gestor de colas local*). Si la aplicación necesita conocer este nombre, utilice la llamada MQINQ para consultar el atributo del **QMGrName** gestor de colas.

Anteponer un asterisco al nombre de la conexión implica que la aplicación no depende de la conexión a un gestor de colas concreto del grupo. Las aplicaciones adecuadas son:

- Aplicaciones que envían mensajes pero no reciben mensajes.
- Aplicaciones que envían mensajes de solicitud y luego obtienen los mensajes de respuesta de una cola *dinámica temporal*.

Las aplicaciones inadecuadas son aquellas que necesitan obtener mensajes de una cola concreta en un gestor de colas concreto; dichas aplicaciones no deben anteponer un asterisco al nombre.

Si se especifica un asterisco, la longitud máxima del resto del nombre es de 47 caracteres.

La longitud de este parámetro viene dada por MQ_Q_MGR_NAME_LENGTH.

Hconn

Tipo: MQHCONN - salida

Este manejador representa la conexión con el gestor de colas. Especifíquelo en todas las llamadas posteriores de cola de mensajes emitidas por la aplicación. Deja de ser válido cuando se emite la llamada MQDISC o cuando finaliza la unidad de procesamiento que define el ámbito del identificador.

IBM MQ Ahora suministra la biblioteca mqm con paquetes de cliente, así como paquetes de servidor. Esto significa que, cuando se realiza una llamada « MQI » que se encuentra en la biblioteca mqm, se comprueba el tipo de conexión para ver si se trata de una conexión de cliente o de servidor, y a continuación se realiza la llamada subyacente correcta. Por lo tanto, una salida que se pasa a *Hconn* ahora se puede vincular a la biblioteca mqm, pero se utiliza en una instalación de cliente.

Ámbito de aplicación : El ámbito del identificador devuelto depende de la llamada utilizada para conectarse al gestor de colas (MQCONN o MQCONNX). Si la llamada utilizada es MQCONNX, el ámbito del identificador también depende de la opción MQCNO_HANDLE_SHARE_* especificada en el *Options* campo de la estructura MQCNO.

- Si la llamada es MQCONN, o se especifica la opción MQCNO_HANDLE_SHARE_NONE, el identificador devuelto es un identificador *no compartido*.

El ámbito de un identificador no compartido es la unidad más pequeña de procesamiento paralelo admitida por la plataforma en la que se ejecuta la aplicación (consulte para obtener [Tabla 582](#) en

la página 666 más información); el identificador no es válido fuera de la unidad de procesamiento paralelo desde la que se realizó la llamada.

- Si se especifica la opción MQCNO_HANDLE_SHARE_BLOCK o MQCNO_HANDLE_SHARE_NO_BLOCK, el identificador devuelto es un identificador *compartido*.

El ámbito de un identificador compartido es el proceso propietario del subproceso desde el que se realizó la llamada; el identificador se puede utilizar desde cualquier subproceso que pertenezca a ese proceso. No todas las plataformas admiten subforos.

- Si la llamada MQCONN o MQCONNX falla con un código de finalización igual a MQCC_FAILED, el valor Hconn queda sin definir.

Tabla 582. Ámbito de aplicación de los identificadores no compartidos en diversas plataformas	
Plataforma	Ámbito de aplicación del identificador no compartido
z/OS	• CICS: la CICS tarea • IMS: la tarea, hasta el siguiente punto de sincronización (excluyendo las subtareas de la tarea) • z/OS lote y TSO: la tarea (excluyendo las subtareas de la tarea)
IBM i	Trabajo
AIX and Linux	Hebra
Aplicaciones Windows de 32 bits	Hebra
Aplicaciones Windows de 64 bits	Hebra

En z/OS para CICS aplicaciones, el valor devuelto es:

MQHC_DEF_HCONN

Manejador de conexión predeterminado.

CompCode

Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_WARNING

Aviso (finalización parcial).

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_WARNING:

MQRC_YA_CONECTADO

(2002, X'7D2') Aplicación ya conectada.

MQRC_CLUSTER_SALIDA_ERROR_CARGA

(2267, X'8DB') No se puede cargar la salida de la carga de trabajo del clúster.

MQRC_SSL_YA_INICIALIZADO

(2391, X'957') SSL ya inicializado.

Si *CompCode* es MQCC_FAILED:

ERROR DE CARGA DE CONEXIÓN DEL ADAPTADOR MQRC

(2129, X'851') No se puede cargar el módulo de conexión del adaptador.

MQRC_ADAPTER_DEFS_ERROR

(2131, X'853') Módulo de definición del subsistema adaptador no válido.

MQRC_ADAPTER_DEFS_LOAD_ERROR

(2132, X'854') No se puede cargar el módulo de definición del subsistema del adaptador.

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'89C') El adaptador no está disponible.

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

MQRC_ADAPTADOR_ALMACENAMIENTO_ESCASIDAD

(2127, X'84F') Almacenamiento insuficiente para el adaptador.

MQRC_OTRO_GESTOR_DE_COLAS_CONECTADO

(2103, X'837') Ya hay otro gestor de colas conectado.

MQRC_API_EXIT_ERROR

(2374, X'946') La salida de la API ha fallado.

MQRC_API_SALIDA_ERROR_INICIAL

(2375, X'947') Error en la inicialización de salida de la API.

MQRC_API_EXIT_TERM_ERROR

(2376, X'948') Error al finalizar la salida de la API.

MQRC_ASID_MISMATCH

(2157, X'86D') Los ASID primario e inicial varían.

MQRC_BUFFER_LENGTH_ERROR

(2005, X'7D5') El parámetro de longitud de almacenamiento intermedio no es válido.

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') La llamada MQI ha entrado antes de completar la llamada anterior.

MQRC_CONN_ID_EN_USO

(2160, X'870') Identificador de conexión ya en uso.

MQRC_CONNECTION_BROKEN

(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

ERROR_DE_CONEXIÓN_MQRC

(2273, X'8E1') Error al procesar la llamada MQCONN.

MQRC_CONEXIÓN_NO_DISPONIBLE

(2568, X' A08 ') Se produce en una llamada MQCONN o MQCONNX cuando el gestor de colas no puede proporcionar una conexión del tipo solicitado en la instalación actual. Una conexión con el cliente no se puede realizar en una instalación solo de servidor. No se puede realizar una conexión local en una instalación solo de cliente.

MQRC_CONNECTION QUIESCING

(2202, X'89A') Conexión en fase de inmovilización.

MQRC_CONNECTION_STOPPING

(2203, X'89B') La conexión está concluyendo.

ERROR_DE_HARDWARE_CRIPTOGRÁFICO_MQRC

(2382, X'94E') Error de configuración del hardware criptográfico.

MQRC_DUPLICADO_RECUPERACIÓN_COORDENADAS

(2163, X'873') Existe un coordinador de recuperación.

MQRC_ERROR_DE_ENTORNO

(2012, X'7DC') Llamada no válida en el entorno.

Además, en la llamada MQCONN, pasar el “MQCSP - Parámetros de seguridad” en la página 291 bloque de control desde una aplicación CICS o IMS.

MQRC_HCONN_ERROR

(2018, X'7E2') El manejador de conexión no es válido.

MQRC_HOST_NOT_AVAILABLE

(2538, X'9EA') Se ha emitido una llamada MQCONN desde un cliente para conectarse a un gestor de colas, pero el intento de asignar una conversación al sistema remoto ha fallado.

MQRC_INSTALACIÓN_INCOMPATIBILIDAD

(2583, X' A17 ') No coincide la instalación del gestor de colas con la biblioteca seleccionada.

ERROR_DEL_REPOSITORIO_DE_CLAVES_MQRC

(2381, X'94D') Repositorio de claves no válido.

MQRC_LÍMITE_MÁXIMO_DE_CONEXIONES_ALCANZADO

(2025, X'7E9') Se ha alcanzado el número máximo de conexiones.

MQRC_NOT_AUTHORIZED

(2035, X'7F3') Acceso no autorizado.

MQRC_ABIERTO_FALLIDO

(2137, X'859') No se ha podido abrir el objeto.

MQRC_Q_MGR_NAME_ERROR

(2058, X'80A') El nombre del gestor de colas no es válido o no se conoce.

MQRC_Q_MGR_NOT_AVAILABLE

(2059, X'80B') El gestor de colas no está disponible para efectuar la conexión.

MQRC_Q_MGR QUIESCING

(2161, X'871') El gestor de colas se está desactivando temporalmente.

MQRC_Q_MGR_STOPPING

(2162, X'872') El gestor de colas está concluyendo.

MQRC_RESOURCE_PROBLEM

(2102, X'836') No hay disponibles suficientes recursos del sistema.

MQRC_ERROR_DE_SEGURIDAD

(2063, X'80F') Se ha producido un error de seguridad.

MQRC_SSL_INITIALIZATION_ERROR

(2393, X'959') Error de inicialización de SSL.

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_UNEXPECTED_ERROR

(2195, X'893') Se ha producido un error inesperado.

Para obtener información detallada sobre estos códigos, consulte [Mensajes y códigos de motivo](#).

Notas de uso

1. El gestor de colas al que se conecta mediante la llamada MQCONN se denomina *gestor de colas local*.
2. Las colas que pertenecen al gestor de colas local se muestran en la aplicación como colas locales. Es posible enviar mensajes a estas colas y recibir mensajes de ellas.

Las colas compartidas que pertenecen al grupo de intercambio de colas al que pertenece el gestor de colas local aparecen en la aplicación como colas locales. Es posible enviar mensajes a estas colas y recibir mensajes de ellas.

Las colas que pertenecen a gestores de colas remotos aparecen como colas remotas. Es posible colocar mensajes en estas colas, pero no obtener mensajes de ellas.

3. Si el gestor de colas falla mientras se está ejecutando una aplicación, esta debe volver a emitir la llamada MQCONN para obtener un nuevo identificador de conexión que se utilizará en las llamadas

IBM MQ posteriores. La aplicación puede emitir la llamada MQCONN periódicamente hasta que la llamada tenga éxito.

Si una aplicación no está segura de si está conectada al gestor de colas, puede emitir de forma segura una llamada MQCONN para obtener un identificador de conexión. Si la aplicación ya está conectada, el identificador devuelto es el mismo que el devuelto por la llamada MQCONN anterior, pero con el código de finalización MQCC_WARNING y el código de motivo MQRC_ALREADY_CONNECTED.

4. Cuando la aplicación haya terminado de utilizar IBM MQ las llamadas, deberá utilizar la llamada MQDISC para desconectarse del gestor de colas.
5. Si la llamada MQCONN falla con un código de finalización igual a MQCC_FAILED, el valor Hconn no está definido.
6. En z/OS:

- Los lotes, TSO y IMS las aplicaciones deben emitir la llamada MQCONN para utilizar las demás IBM MQ llamadas. Estas aplicaciones pueden conectarse a más de un gestor de colas simultáneamente.

Si el gestor de colas falla, la aplicación debe volver a realizar la llamada después de que el gestor de colas se haya reiniciado para obtener un nuevo identificador de conexión.

Aunque IMS las aplicaciones pueden emitir la llamada MQCONN repetidamente, incluso cuando ya están conectadas, esto no es recomendable para los programas de procesamiento de mensajes en línea (MPP).

- CICS Las aplicaciones no tienen que emitir la llamada MQCONN para utilizar las otras IBM MQ llamadas, pero pueden hacerlo si lo desean; se aceptan tanto la llamada MQCONN como la llamada MQDISC. Sin embargo, no es posible conectarse a más de un gestor de colas simultáneamente.

Si el gestor de colas falla, estas aplicaciones se vuelven a conectar automáticamente cuando el gestor de colas se reinicia, por lo que no es necesario emitir la llamada MQCONN.

7. En z/OS, para definir los gestores de colas disponibles:

- Para aplicaciones por lotes, los programadores del sistema pueden utilizar la macro CSQBDEF para crear un módulo (CSQBDEFV) que defina el nombre predeterminado del gestor de colas o el nombre del grupo de uso compartido de colas.
- Para IMS las aplicaciones, los programadores del sistema pueden utilizar la macro CSQQDEFX para crear un módulo (CSQQDEFV) que define los nombres de los gestores de colas disponibles y especifica el gestor de colas predeterminado.

Además, cada gestor de colas debe definirse en la IMS región de control y en cada región dependiente que acceda a ese gestor de colas. Para ello, debe crear un miembro del subsistema en la biblioteca IMS.PROCLIB e identificar el miembro del subsistema en las regiones IMS correspondientes. Si una aplicación intenta conectarse a un gestor de colas que no está definido en el miembro del subsistema de su IMS región, la aplicación se interrumpe.

 Para obtener más información sobre el uso de estas macros, consulte [Macros destinadas al uso de los clientes](#).

8. En IBM i, los programas que finalizan de forma anómala no se desconectan automáticamente del gestor de colas. Escriba aplicaciones que permitan la posibilidad de que la llamada MQCONN o MQCONNX devuelva el código de finalización MQCC_WARNING y el código de motivo MQRC_ALREADY_CONNECTED. Utilice el identificador de conexión devuelto en esta situación como de costumbre.

Invocación en C

```
MQCONN (QMgrName, &Hconn, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQCHAR48 QMgrName; /* Name of queue manager */
MQHCONN Hconn; /* Connection handle */
```

```

MQLONG    CompCode;    /* Completion code */
MQLONG    Reason;      /* Reason code qualifying CompCode */

```

Invocación en COBOL

```
CALL 'MQCONN' USING QMGRNAME, HCONN, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```

** Name of queue manager
01 QMGRNAME PIC X(48).
** Connection handle
01 HCONN     PIC S9(9) BINARY.
** Completion code
01 COMPCODE  PIC S9(9) BINARY.
** Reason code qualifying COMPCODE
01 REASON    PIC S9(9) BINARY.

```

Invocación en PL/I

```
call MQCONN (QMgrName, Hconn, CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```

dcl QMgrName char(48);      /* Name of queue manager */
dcl Hconn     fixed bin(31); /* Connection handle */
dcl CompCode  fixed bin(31); /* Completion code */
dcl Reason    fixed bin(31); /* Reason code qualifying CompCode */

```

Invocación en ensamblador de alto nivel

```
CALL MQCONN,(QMGRNAME,HCONN,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

```

QMGRNAME DS CL48 Name of queue manager
HCONN    DS F   Connection handle
COMPCODE DS F   Completion code
REASON   DS F   Reason code qualifying COMPCODE

```

Invocación en Visual Basic

```
MQCONN QMgrName, Hconn, CompCode, Reason
```

Declare los parámetros como se indica a continuación:

```

Dim QMgrName As String*48 'Name of queue manager'
Dim Hconn    As Long      'Connection handle'
Dim CompCode As Long      'Completion code'
Dim Reason   As Long      'Reason code qualifying CompCode'

```

MQCONN - Conectar gestor de colas (ampliado)

La llamada MQCONN conecta un programa de aplicación con un gestor de colas. Proporciona un identificador de conexión del gestor de colas, que es utilizado por la aplicación en llamadas IBM MQ posteriores.

La llamada MQCONN es similar a la llamada MQCONN, salvo que MQCONN permite especificar opciones para controlar el funcionamiento de la llamada.

- Esta llamada es compatible con todos IBM MQ los sistemas y IBM MQ clientes conectados a estos sistemas.

No se puede establecer una conexión de cliente en una instalación solo de servidor, ni una conexión local en una instalación solo de cliente.

Sintaxis

MQCONN (*Nombre del gestor de colas, Opciones de conexión, Conexión, Código de comparación, Motivo*)

Parámetros

QMgrName

Tipo: MQCHAR48 - entrada

Consulte el **QMgrName** parámetro descrito en [“MQCONN - Conectar gestor de colas”](#) en la página 663 para obtener más detalles.

ConnectOpts

Tipo: MQCNO - entrada/salida

Consulte [“MQCNO - Opciones de conexión”](#) en la página 269 para obtener los detalles.

Hconn

Tipo: MQHCONN - salida

Este manejador representa la conexión con el gestor de colas. Especifíquelo en todas las llamadas posteriores de cola de mensajes emitidas por la aplicación. Deja de ser válido cuando se emite la llamada MQDISC o cuando finaliza la unidad de procesamiento que define el ámbito del identificador.

IBM MQ Ahora suministra la biblioteca mqm con paquetes de cliente, así como paquetes de servidor. Esto significa que, cuando se realiza una llamada « MQI » que se encuentra en la biblioteca mqm, se comprueba el tipo de conexión para ver si se trata de una conexión de cliente o de servidor, y a continuación se realiza la llamada subyacente correcta. Por lo tanto, una salida que se pasa a *Hconn* ahora se puede vincular a la biblioteca mqm, pero se utiliza en una instalación de cliente.

Ámbito de aplicación : El ámbito del identificador devuelto depende de la llamada utilizada para conectarse al gestor de colas (MQCONN o MQCONNX). Si la llamada utilizada es MQCONNX, el ámbito del identificador también depende de la opción MQCNO_HANDLE_SHARE_* especificada en el *Options* campo de la estructura MQCNO.

- Si la llamada es MQCONN, o se especifica la opción MQCNO_HANDLE_SHARE_NONE, el identificador devuelto es un identificador *no compartido*.

El ámbito de un identificador no compartido es la unidad más pequeña de procesamiento paralelo admitida por la plataforma en la que se ejecuta la aplicación (consulte para obtener [Tabla 583 en la página 672](#) más información); el identificador no es válido fuera de la unidad de procesamiento paralelo desde la que se realizó la llamada.

- Si se especifica la opción MQCNO_HANDLE_SHARE_BLOCK o MQCNO_HANDLE_SHARE_NO_BLOCK, el identificador devuelto es un identificador *compartido*.

El ámbito de un identificador compartido es el proceso propietario del subproceso desde el que se realizó la llamada; el identificador se puede utilizar desde cualquier subproceso que pertenezca a ese proceso. No todas las plataformas admiten subforos.

- Si la llamada MQCONN o MQCONNX falla con un código de finalización igual a MQCC_FAILED, el valor Hconn queda sin definir.

Tabla 583. Ámbito de aplicación de los identificadores no compartidos en diversas plataformas	
Plataforma	Ámbito de aplicación del identificador no compartido
z/OS	• CICS: la CICS tarea • IMS: la tarea, hasta el siguiente punto de sincronización (excluyendo las subtareas de la tarea) • z/OS lote y TSO: la tarea (excluyendo las subtareas de la tarea)
IBM i	Trabajo
AIX and Linux	Hebra
Aplicaciones Windows de 32 bits	Hebra
Aplicaciones Windows de 64 bits	Hebra

En z/OS para CICS aplicaciones, el valor devuelto es:

MQHC_DEF_HCONN

Manejador de conexión predeterminado.

CompCode

Tipo: MQLONG - salida

Consulte el **CompCode** parámetro descrito en [“MQCONN - Conectar gestor de colas”](#) en la página 663 para obtener más detalles.

Razón

Tipo: MQLONG - salida

Las llamadas MQCONN y MQCONNX pueden devolver los siguientes códigos. Para obtener una lista de códigos adicionales que puede devolver la llamada MQCONNX, consulte los siguientes códigos.

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_WARNING:

MQRC_YA_CONECTADO

(2002, X'7D2') Aplicación ya conectada.

MQRC_CLUSTER_SALIDA_ERROR_CARGA

(2267, X'8DB') No se puede cargar la salida de la carga de trabajo del clúster.

MQRC_SSL_YA_INICIALIZADO

(2391, X'957') SSL ya inicializado.

Si *CompCode* es MQCC_FAILED:

ERROR DE CARGA DE CONEXIÓN DEL ADAPTADOR MQRC

(2129, X'851') No se puede cargar el módulo de conexión del adaptador.

MQRC_ADAPTER_DEFS_ERROR

(2131, X'853') Módulo de definición del subsistema adaptador no válido.

MQRC_ADAPTER_DEFS_LOAD_ERROR

(2132, X'854') No se puede cargar el módulo de definición del subsistema del adaptador.

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'89C') El adaptador no está disponible.

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

MQRC_ADAPTADOR_ALMACENAMIENTO_ESCASIDAD

(2127, X'84F') Almacenamiento insuficiente para el adaptador.

MQRC_OTRO_GESTOR_DE_COLAS_CONECTADO

(2103, X'837') Ya hay otro gestor de colas conectado.

MQRC_API_EXIT_ERROR

(2374, X'946') La salida de la API ha fallado.

MQRC_API_SALIDA_ERROR_INICIAL

(2375, X'947') Error en la inicialización de salida de la API.

MQRC_API_EXIT_TERM_ERROR

(2376, X'948') Error al finalizar la salida de la API.

MQRC_ASID_MISMATCH

(2157, X'86D') Los ASID primario e inicial varían.

MQRC_BUFFER_LENGTH_ERROR

(2005, X'7D5') El parámetro de longitud de almacenamiento intermedio no es válido.

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') La llamada MQI ha entrado antes de completar la llamada anterior.

MQRC_CONN_ID_EN_USO

(2160, X'870') Identificador de conexión ya en uso.

MQRC_CONNECTION_BROKEN

(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

ERROR_DE_CONEXIÓN_MQRC

(2273, X'8E1') Error al procesar la llamada MQCONN.

MQRC_CONEXIÓN_NO_DISPONIBLE

(2568, X' A08 ') Se produce en una llamada MQCONN o MQCONNx cuando el gestor de colas no puede proporcionar una conexión del tipo solicitado en la instalación actual. Una conexión con el cliente no se puede realizar en una instalación solo de servidor. No se puede realizar una conexión local en una instalación solo de cliente.

MQRC_CONNECTION QUIESCING

(2202, X'89A') Conexión en fase de inmovilización.

MQRC_CONNECTION_STOPPING

(2203, X'89B') La conexión está concluyendo.

ERROR_DE_HARDWARE_CRIPTOGRÁFICO_MQRC

(2382, X'94E') Error de configuración del hardware criptográfico.

MQRC_DUPLICADO_RECUPERACIÓN_COORDENADAS

(2163, X'873') Existe un coordinador de recuperación.

MQRC_ERROR_DE_ENTORNO

(2012, X'7DC') Llamada no válida en el entorno.

Además, en la llamada MQCONNx, pasar el “MQCSP - Parámetros de seguridad” en la [página 291](#) bloque de control desde una aplicación CICS o IMS.

MQRC_HCONN_ERROR

(2018, X'7E2') El manejador de conexión no es válido.

MQRC_HOST_NOT_AVAILABLE

(2538, X'9EA') Se ha emitido una llamada MQCONN desde un cliente para conectarse a un gestor de colas, pero el intento de asignar una conversación al sistema remoto ha fallado.

MQRC_INSTALACIÓN_INCOMPATIBILIDAD

(2583, X' A17 ') No coincide la instalación del gestor de colas con la biblioteca seleccionada.

ERROR_DEL_REPOSITORIO_DE_CLAVES_MQRC

(2381, X'94D') Repositorio de claves no válido.

MQRC_LÍMITE_MÁXIMO_DE_CONEXIONES_ALCANZADO

(2025, X'7E9') Se ha alcanzado el número máximo de conexiones.

MQRC_NOT_AUTHORIZED

(2035, X'7F3') Acceso no autorizado.

MQRC_ABIERTO_FALLIDO

(2137, X'859') No se ha podido abrir el objeto.

MQRC_Q_MGR_NAME_ERROR

(2058, X'80A') El nombre del gestor de colas no es válido o no se conoce.

MQRC_Q_MGR_NOT_AVAILABLE

(2059, X'80B') El gestor de colas no está disponible para efectuar la conexión.

MQRC_Q_MGR QUIESCING

(2161, X'871') El gestor de colas se está desactivando temporalmente.

MQRC_Q_MGR_STOPPING

(2162, X'872') El gestor de colas está concluyendo.

MQRC_RESOURCE_PROBLEM

(2102, X'836') No hay disponibles suficientes recursos del sistema.

MQRC_ERROR_DE_SEGURIDAD

(2063, X'80F') Se ha producido un error de seguridad.

MQRC_SSL_INITIALIZATION_ERROR

(2393, X'959') Error de inicialización de SSL.

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_UNEXPECTED_ERROR

(2195, X'893') Se ha producido un error inesperado.

La llamada MQCONN puede devolver los siguientes códigos de motivo adicionales:

Si *CompCode* es MQCC_FAILED:

MQRC_ERROR_AIR

(2385, X'951') Registro de información de autenticación no válido.

MQRC_AUTH_INFO_CONN_NAME_ERROR

(2387, X'953') Nombre de conexión de la información de autenticación no válido.

ERROR_EN_EL_NÚMERO_DE_REGISTROS_DE INFORMACIÓN_DE AUTENTICACIÓN_MQRC

(2383, X'94F') Recuento de registros de información de autenticación no válido.

MQRC_AUTH_INFO_REC_ERROR

(2384, X'950') Campos del registro de información de autenticación no válidos.

MQRC_AUTH_INFO_TYPE_ERROR

(2386, X'952') Tipo de información de autenticación no válido.

MQRC_CD_ERROR

(2277, X'8E5') Definición de canal no válida.

ERROR_DE_CONEXIÓN_DEL_CLIENTE_MQRC

(2278, X'8E6') Campos de conexión del cliente no válidos.

MQRC_CNO_ERROR

(2139, X'85B') Estructura de opciones de conexión no válida.

MQRC_CONN_TAG_EN_USO

(2271, X'8DF') Etiqueta de conexión en uso.

MQRC_CONN_TAG_NO_UTILIZABLE

(2350, X'92E') Etiqueta de conexión no utilizable.

MQRC_CSP_ERROR

(2595, X' A23 ') La estructura MQCSP no es válida.

MQRC_FUNCTION_NOT_SUPPORTED

(2298, X'8FA') La función solicitada no está disponible en el entorno actual.

ERROR_CON_LA_CONTRASEÑA_MQRC_LDAP

(2390, X'956') Contraseña LDAP no válida.

ERROR_DE_NOMBRE_DE_USUARIO_LDAP_MQRC

(2388, X'954') LDAP campos de nombre de usuario no válidos.

MQRC_LDAP_NOMBRE_USUARIO_LONGITUD_ERROR

(2389, X'955') LDAP Longitud del nombre de usuario no válida.

MQRC_OPTIONS_ERROR

(2046, X'7FE') Las opciones no son válidas o no son coherentes.

MQRC_SCO_ERROR

(2380, X'94C') Estructura de opciones de configuración SSL no válida.

ERROR_DE_CONFIGURACIÓN_MQRC_SSL

(2392, X'958') Error de configuración de SSL.

MQRC_TOKEN_TIMESTAMP_NO_VÁLIDO

(2064, X'810') El token de autenticación aún no es válido o ha caducado.

Para obtener información detallada sobre estos códigos, consulte [Mensajes y códigos de motivo](#).

Notas de uso

Para el lenguaje de programación Visual Basic, se aplica lo siguiente:

- El **ConnectOpts** parámetro se declara como de tipo MQCNO. Si la aplicación se ejecuta como un IBM MQ MQI clienty desea especificar los parámetros del canal de conexión del cliente, declare el **ConnectOpts** parámetro como de tipo Any, de modo que la aplicación pueda especificar una estructura MQCNOCD en la llamada en lugar de una estructura MQCNO. Sin embargo, esto significa que no se puede comprobar el **ConnectOpts** parámetro para garantizar que sea del tipo de datos correcto.

Invocación en C

```
MQCONN (QMgrName, &ConnectOpts, &Hconn, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQCHAR48  QMgrName;      /* Name of queue manager */
MQCNO      ConnectOpts;   /* Options that control the action of MQCONN */
MQHCONN     Hconn;        /* Connection handle */
MQLONG      CompCode;     /* Completion code */
MQLONG      Reason;       /* Reason code qualifying CompCode */
```

Invocación en COBOL

```
CALL 'MQCONN' USING QMGRNAME, CONNECTOPTS, HCONN, COMPCODE,
REASON.
```

Declare los parámetros como se indica a continuación:

```
** Name of queue manager
01 QMGRNAME      PIC X(48).
** Options that control the action of MQCONN
01 CONNECTOPTS.
   COPY CMQCNV.
** Connection handle
01 HCONN         PIC S9(9) BINARY.
** Completion code
01 COMPCODE      PIC S9(9) BINARY.
** Reason code qualifying COMPCODE
01 REASON        PIC S9(9) BINARY.
```

Invocación en PL/I

```
call MQCONN (QMgrName, ConnectOpts, Hconn, CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```
decl QMgrName      char(48);      /* Name of queue manager */
decl ConnectOpts   like MQCNO;    /* Options that control the action of
                                   MQCONN */
decl Hconn         fixed bin(31);  /* Connection handle */
decl CompCode      fixed bin(31);  /* Completion code */
decl Reason        fixed bin(31);  /* Reason code qualifying CompCode */
```

Invocación en ensamblador de alto nivel

```
CALL MQCONN, (QMGRNAME,CONNECTOPTS,HCONN,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

QMGRNAME	DS	CL48	Name of queue manager
CONNECTOPTS	CMQCN0A	,	Options that control the action of MQCONN
HCONN	DS	F	Connection handle
COMPCODE	DS	F	Completion code
REASON	DS	F	Reason code qualifying COMPCODE

Invocación en Visual Basic

```
MQCONN QMgrName, ConnectOpts, Hconn, CompCode, Reason
```

Declare los parámetros como se indica a continuación:

```
Dim QMgrName As String*48 'Name of queue manager'
Dim ConnectOpts As MQCNO 'Options that control the action of
                          'MQCONN'
Dim Hconn As Long 'Connection handle'
Dim CompCode As Long 'Completion code'
Dim Reason As Long 'Reason code qualifying CompCode'
```

MQCRTMH: crear identificador de mensaje

La llamada MQCRTMH devuelve un identificador de mensaje.

Una aplicación puede utilizar la llamada MQCRTMH en llamadas posteriores de cola de mensajes:

- Utilice la llamada [MQSETMP](#) para establecer una propiedad del identificador del mensaje.
- Utilice la llamada [MQINQMP](#) para consultar el valor de una propiedad del identificador del mensaje.
- Utilice la llamada [MQDLTMP](#) para eliminar una propiedad del identificador del mensaje.

El identificador de mensaje se puede utilizar en las llamadas MQPUT y MQPUT1 para asociar las propiedades del identificador de mensaje con las del mensaje que se está enviando. Del mismo modo, al especificar un identificador de mensaje en la llamada MQGET, se puede acceder a las propiedades del mensaje que se está recuperando utilizando el identificador de mensaje cuando finaliza la llamada MQGET.

Utilice [MQDLTMH](#) para eliminar el identificador del mensaje.

Sintaxis

```
MQCRTMH ( Hconn, CrtMsgHOpts, Hmsg, CompCode, Motivo )
```


Parámetros

Hconn

Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* fue devuelto por una llamada anterior a MQCONN o MQCONNX. Si la conexión con el gestor de colas deja de ser válida y no hay ninguna IBM MQ llamada operando en el identificador del mensaje, se llama implícitamente a [MQDLTMH](#) para eliminar el mensaje.

Como alternativa, puede especificar el siguiente valor:

MQHC_NO_ASOCIADO_HCONN

El identificador de conexión no representa una conexión con ningún gestor de colas en particular.

Cuando se utiliza este valor, el identificador del mensaje debe eliminarse con una llamada explícita a [MQDLTMH](#) para liberar cualquier almacenamiento que se le haya asignado; IBM MQ nunca elimina implícitamente el identificador del mensaje.

Debe haber al menos una conexión válida establecida con un gestor de colas en el subproceso que crea el identificador de mensaje; de lo contrario, la llamada fallará con el error MQRC_HCONN_ERROR.

En un entorno con múltiples instalaciones en un único sistema, el valor MQHC_UNASSOCIATED_HCONN solo se puede utilizar con la primera instalación cargada en el proceso. El código de motivo MQRC_HMSG_NOT_AVAILABLE se devuelve si el identificador del mensaje se proporciona a una instalación diferente.

En z/OS para CICS aplicaciones, se puede omitir la llamada MQCONN y se puede especificar el siguiente valor para *Hconn*:

MQHC_DEF_HCONN

Manejador de conexión predeterminado

CrtMsgHOpts

Tipo: MQCMHO - entrada

Las opciones que controlan la acción de MQCRTMH. Consulte [MQCMHO](#) para obtener más detalles.

Hmsg

Tipo: MQHMSG - salida

En la salida se devuelve un identificador de mensaje que se puede utilizar para establecer, consultar y eliminar propiedades del identificador de mensaje. Inicialmente, el identificador del mensaje no contiene ninguna propiedad.

Un identificador de mensaje también tiene un descriptor de mensaje asociado. Inicialmente contiene los valores predeterminados. Los valores de los campos descriptores de mensajes asociados se pueden establecer y consultar utilizando las llamadas MQSETMP y MQINQMP. La llamada MQDLTMP restablece un campo del descriptor de mensajes a su valor predeterminado.

Si el parámetro *Hconn* se especifica con el valor MQHC_UNASSOCIATED_HCONN, el identificador de mensaje devuelto se puede utilizar en llamadas MQGET, MQPUT o MQPUT1 con cualquier conexión dentro de la unidad de procesamiento, pero solo puede ser utilizado por una IBM MQ llamada a la vez. Si el identificador está en uso cuando una segunda IBM MQ llamada intenta utilizar el mismo identificador de mensaje, la segunda IBM MQ llamada falla con el código de motivo MQRC_MSG_HANDLE_IN_USE.

Si el parámetro *Hconn* no es MQHC_UNASSOCIATED_HCONN, el identificador de mensaje devuelto solo se puede utilizar en la conexión especificada.

Se debe utilizar el mismo valor del parámetro *Hconn* en las siguientes llamadas a `MQI` en las que se utilice este identificador de mensaje:

- MQDLTMH
- MQSETMP

- MQINQMP
- MQDLTMP
- MQMHBUFF
- MQBUFMH

El identificador de mensaje devuelto deja de ser válido cuando se emite la llamada MQDLTMH para el identificador de mensaje o cuando finaliza la unidad de procesamiento que define el ámbito del identificador. MQDLTMH se invoca implícitamente si se proporciona una conexión específica al crear el identificador de mensaje y la conexión con el gestor de colas deja de ser válida, por ejemplo, si se invoca MQDBC.

CompCode

Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'089C') Adaptador no disponible.

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

MQRC_ASID_MISMATCH

(2157, X'86D') Los ASID primario e inicial varían.

MQRC_CALL_IN_PROGRESS

(2219, X'08AB') Se ha iniciado una llamada al MQI antes de que finalizara la llamada anterior.

MQRC_CMHO_ERROR

(2461, X'099D') Estructura de opciones de control de mensajes no válida.

MQRC_CONNECTION_BROKEN

(2273, X'7D9') Se ha perdido la conexión con el gestor de colas.

MQRC_HANDLE_NO_DISPONIBLE

(2017, X'07E1') No hay más asas disponibles.

MQRC_HCONN_ERROR

(2018, X'7E2') El manejador de conexión no es válido.

MQRC_HMSG_ERROR

(2460, X'099C') El puntero del identificador del mensaje no es válido.

MQRC_OPTIONS_ERROR

(2046, X'07FE') Opciones no válidas o incoherentes.

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_UNEXPECTED_ERROR

(2195, X'893') Se ha producido un error inesperado.

Para obtener información detallada sobre estos códigos, consulte [Mensajes y códigos de motivo](#).

C

```
MQCRTMH (Hconn, &CrtMsgHOpts, &Hmsg, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQHCONN  Hconn;          /* Connection handle */
MQCMHO   CrtMsgHOpts;    /* Options that control the action of MQCRTMH */
MQHMSG   Hmsg;           /* Message handle */
MQLONG   CompCode;       /* Completion code */
MQLONG   Reason;         /* Reason code qualifying CompCode */
```

COBOL

```
CALL 'MQCRTMH' USING HCONN, CRTMSGHOPTS, HMSG, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
** Connection handle
01 HCONN      PIC S9(9) BINARY.
** Options that control the action of MQCRTMH
01 CRTMSGHOPTS.
   COPY CMQCMHOV.
** Message handle
01 HMSG       PIC S9(18) BINARY.
** Completion code
01 COMPCODE   PIC S9(9) BINARY.
** Reason code qualifying COMPCODE
01 REASON     PIC S9(9) BINARY.
```

PL/I

```
call MQCRTMH (Hconn, CrtMsgHOpts, Hmsg, CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```
dcl Hconn      fixed bin(31); /* Connection handle */
dcl CrtMsgHOpts like MQCMHO;  /* Options that control the action of MQCRTMH */
dcl Hmsg       fixed bin(63); /* Message handle */
dcl CompCode   fixed bin(31); /* Completion code */
dcl Reason     fixed bin(31); /* Reason code qualifying CompCode */
```

High Level Assembler

```
CALL MQCRTMH,(HCONN,CRTMSGHOPTS,HMSG,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

```
HCONN      DS      F      Connection handle
CRTMSGHOPTS CMQCMHOA ,      Options that control the action of MQCRTMH
HMSG       DS      D      Message handle
COMPCODE   DS      F      Completion code
REASON     DS      F      Reason code qualifying COMPCODE
```

MQCTL - Control de devoluciones de llamada

La llamada MQCTL realiza acciones de control sobre las devoluciones de llamada y los identificadores de objetos abiertos para una conexión.

Sintaxis

MQCTL (*Hconn*, *Operación*, *ControlOpts*, *CompCode*, *Motivo*)

Parámetros

Hconn

Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* fue devuelto por una llamada anterior a MQCONN o MQCONNX.

En z/OS para CICS aplicaciones, se puede omitir la llamada MQCONN y se puede especificar el siguiente valor especial para *Hconn*:

MQHC_DEF_HCONN

Manejador de conexión predeterminado.

Operación

Tipo: MQLONG - entrada

La operación que se está procesando en la devolución de llamada definida para el identificador de objeto especificado. Debe especificar una, y solo una, de las siguientes opciones:

MQOP_INICIO

Inicie el consumo de mensajes para todas las funciones de consumo de mensajes definidas para el identificador de conexión especificado.

Las devoluciones de llamada se ejecutan en un subproceso iniciado por el sistema, que es diferente de cualquiera de los subprocesos de la aplicación.

Esta operación cede el control del identificador de conexión proporcionado al sistema. Las únicas llamadas a `MQI` que puede realizar un hilo distinto del hilo consumidor son:

- MQCTL con la operación MQOP_STOP
- MQCTL con la operación MQOP_SUSPEND
- MQDISC: realiza MQCTL con la operación MQOP_STOP antes de desconectar HConn.

Se devuelve MQRC_HCONN_ASYNC_ACTIVE si se emite una llamada IBM MQ a la API mientras se inicia el controlador de conexión y la llamada no se origina en una función de consumidor de mensajes.

Si un consumidor de mensajes detiene la conexión durante MQCBCT_START_CALL, la llamada MQCTL devuelve un código de motivo de error MQRC_CONNECTION_STOPPED.

Esto se puede emitir en una función de consumo. Para la misma conexión que la rutina de devolución de llamada, su único propósito es cancelar una operación MQOP_STOP emitida previamente.

Esta opción no es compatible en los siguientes entornos: CICS en z/OS o si la aplicación está vinculada a una biblioteca IBM MQ sin subprocesos.

MQOP_INICIO_ESPERA

Inicie el consumo de mensajes para todas las funciones de consumo de mensajes definidas para el identificador de conexión especificado.

Los consumidores de mensajes se ejecutan en el mismo subproceso y el control no se devuelve al llamante de MQCTL hasta que:

- Liberado mediante el uso de las operaciones MQCTL MQOP_STOP o MQOP_SUSPEND, o
- Todas las rutinas de consumo han sido dadas de baja o suspendidas.

Si todos los consumidores se dan de baja o se suspenden, se emite una operación MQOP_STOP implícita.

Esta opción no se puede utilizar desde una rutina de devolución de llamada, ni para el identificador de conexión actual ni para ningún otro identificador de conexión. Si se intenta la llamada, devuelve MQRC_ENVIRONMENT_ERROR.

Si, en cualquier momento durante una operación MQOP_START_WAIT, no hay consumidores registrados y no suspendidos, la llamada falla con un código de motivo MQRC_NO_CALLBACKS_ACTIVE.

Si, durante una operación MQOP_START_WAIT, se suspende la conexión, la llamada MQCTL devuelve un código de motivo de advertencia MQRC_CONNECTION_SUSPENDED; la conexión permanece «iniciada».

La aplicación puede optar por emitir MQOP_STOP o MQOP_RESUME. En este caso, la operación MQOP_RESUME se bloquea.

Esta opción no es compatible con un cliente de un solo subproceso.

MQOP_STOP

Detenga el consumo de mensajes y espere a que todos los consumidores completen sus operaciones antes de que esta opción finalice. Esta operación libera el identificador de conexión.

Si se emite desde una rutina de devolución de llamada, esta opción no surtirá efecto hasta que la rutina finalice. No se vuelven a llamar rutinas de consumo de mensajes después de que hayan finalizado las rutinas de consumo de mensajes ya leídos y después de que se hayan realizado las llamadas de detención (si se solicitan) a las rutinas de devolución de llamada.

Si se emite fuera de una rutina de devolución de llamada, el control no vuelve al llamante hasta que las rutinas de consumo de los mensajes ya leídos hayan finalizado y después de que se hayan realizado las llamadas de detención (si se solicitan) a las devoluciones de llamada. Sin embargo, las devoluciones de llamada siguen registradas.

Esta función no tiene ningún efecto sobre los mensajes de lectura anticipada. Debe asegurarse de que los consumidores ejecuten MQCLOSE(MQCO_QUIESCE) desde la función de devolución de llamada para determinar si hay más mensajes disponibles para su entrega.

MQOP_SUSPENDER

Pausar el consumo de mensajes. Esta operación libera el identificador de conexión.

Esto no tiene ningún efecto en la lectura anticipada de mensajes para la aplicación. Si tiene intención de dejar de consumir mensajes durante un periodo prolongado, considere cerrar la cola y volver a abrirla cuando continúe el consumo.

Si se emite desde una rutina de devolución de llamada, no surtirá efecto hasta que la rutina finalice. No se llamarán más rutinas de consumo de mensajes después de que finalice la rutina actual.

Si se emite fuera de una devolución de llamada, el control no vuelve al llamante hasta que la rutina de consumo actual haya finalizado y no se llamen más.

MQOP_RESUMIR

Reanuda el consumo de mensajes.

Esta opción se emite normalmente desde el subproceso principal de la aplicación, pero también se puede utilizar desde una rutina de devolución de llamada para cancelar una solicitud de suspensión anterior emitida en la misma rutina.

Si se utiliza MQOP_RESUME para reanudar un MQOP_START_WAIT, la operación se bloquea.

ControlOpts

Tipo: MQCTLO - entrada

Opciones que controlan la acción de MQCTL

Consulte [MQCTLO](#) para obtener más información sobre la estructura.

CompCode

Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_WARNING

Aviso (finalización parcial).

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_ADAPTER_CONV_LOAD_ERROR

(2133, X'855') No se han podido cargar módulos de servicio de conversión de datos.

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'89C') El adaptador no está disponible.

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

MQRC_API_EXIT_ERROR

(2374, X'946') La salida de la API ha fallado.

MQRC_API_EXIT_LOAD_ERROR

(2183, X'887') No se ha podido cargar la salida de la API.

MQRC_ASID_MISMATCH

(2157, X'86D') Los ASID primario e inicial varían.

MQRC_BUFFER_LENGTH_ERROR

(2005, X'7D5') El parámetro de longitud de almacenamiento intermedio no es válido.

MQRC_CALLBACK_LINK_ERROR

(2487, X'9B7') No se puede llamar a la rutina de devolución de llamada

MQRC_CALLBACK_NOT_REGISTRADO

(2448, X'990') No se puede dar de baja, suspender o reanudar porque no hay ninguna devolución de llamada registrada

MQRC_CALLBACK_ROUTINE_ERROR

(2486, X'9B6') Se han especificado tanto CallbackFunction como CallbackName en una llamada MQOP_REGISTER.

O bien se ha especificado CallbackFunction o CallbackName, pero no coincide con la función de devolución de llamada registrada actualmente.

MQRC_CALLBACK_TYPE_ERROR

(2483, X'9B3') Campo CallBackType incorrecto.

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') La llamada MQI ha entrado antes de completar la llamada anterior.

MQRC_CBD_ERROR

(2444, X'98C') El bloque de opciones es incorrecto.

MQRC_CBD OPCIONES_ERROR

(2484, X'9B4') Campo de opciones MQCBD incorrecto.

MQRC_CICS_WAIT_FAILED

(2140, X'85C') Solicitud de espera rechazada por CICS.

MQRC_CONNECTION_BROKEN

(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

MQRC_CONNECTION_NOT_AUTHORIZED

(2217, X'8A9') No tiene autorización para la conexión.

MQRC_CONNECTION_QUIESCING

(2202, X'89A') Conexión en fase de inmovilización.

MQRC_CONNECTION_STOPPING

(2203, X'89B') La conexión está concluyendo.

MQRC_CORREL_ID_ERROR

(2207, X'89F') Error de identificador de correlación.

MQRC_ERROR_DE_ENTORNO

(2012, X'7DC') Llamada no válida en el entorno.

MQRC_FUNCTION_NOT_SUPPORTED

(2298, X'8FA') La función solicitada no está disponible en el entorno actual.

MQRC_GET_INHIBITED

(2016, X'7E0') Se han inhibido las obtenciones para la cola.

MQRC_GLOBAL_UOW_CONFLICT

(2351, X'92F') Conflicto de unidades de trabajo global.

MQRC_GMO_ERROR

(2186, X'88A') No es válida la estructura de las opciones de obtención de mensaje.

MQRC_HANDLE_IN_USE_FOR_UOW

(2353, X'931') Descriptor de contexto que se utiliza para la unidad de trabajo global.

MQRC_HCONN_ERROR

(2018, X'7E2') El manejador de conexión no es válido.

MQRC_HOBJ_ERROR

(2019, X'7E3') El manejador de objeto no es válido.

MQRC_INCONSISTENT_BROWSE

(2259, X'8D3') La especificación de examinar es incoherente.

MQRC_INCONSISTENT_UOW

(2245, X'8C5') Especificación incoherente de unidad de trabajo.

MQRC_INVALID_MSG_UNDER_CURSOR

(2246, X'8C6') El mensaje bajo cursor no es válido para recuperación.

MQRC_LOCAL_UOW_CONFLICT

(2352, X'930') La unidad de trabajo global entra en conflicto con la unidad de trabajo local.

MQRC_MATCH_OPTIONS_ERROR

(2247, X'8C7') Las opciones de coincidencia no son válidas.

MQRC_MAX_MSG_LENGTH_ERROR

(2485, X'9B5') Campo « MaxMsgLength » incorrecto

MQRC_MD_ERROR

(2026, X'7EA') El descriptor de mensaje no es válido.

MQRC_MODULE_ENTRY_NOT_FOUND

(2497, X'9C1') No se ha encontrado el punto de entrada de la función especificada en el módulo.

MQRC_MÓDULO_INVÁLIDO

(2496, X'9C0') Se ha encontrado el módulo, pero es del tipo incorrecto (32 bit/64 bits) o no es un archivo DLL válido.

MQRC_MÓDULO_NO_ENCONTRADO

(2495, X'9BF') Módulo no encontrado en la ruta de búsqueda o sin autorización para cargarse.

MQRC_MSG_ID_ERROR

(2206, X'89E') Error de identificador de mensaje.

MQRC_MSG_SEQ_NUMBER_ERROR

(2250, X'8CA') El número de secuencia de mensaje no es válido.

MQRC_MSG_TOKEN_ERROR

(2331, X'91B') El uso del símbolo de mensaje no es válido.

MQRC_NOT_OPEN_FOR_BROWSE

(2036, X'7F4') La cola no se ha abierto para examen.

MQRC_NOT_OPEN_FOR_INPUT

(2037, X'7F5') La cola no se ha abierto para entrada.

MQRC_OBJECT_CHANGED

(2041, X'7F9') La definición de objeto ha cambiado desde que se ha abierto.

MQRC_OBJECT_DAMAGED

(2101, X'835') El objeto se ha dañado.

ERROR_DE_OPERACIÓN_MQRC

(2488, X'9B8') Código de operación incorrecto en la llamada a la API

MQRC_OPTIONS_ERROR

(2046, X'7FE') Las opciones no son válidas o no son coherentes.

MQRC_PAGESET_ERROR

(2193, X'891') Error al acceder al conjunto de datos del conjunto de páginas.

MQRC_Q_DELETED

(2052, X'804') La cola se ha suprimido.

MQRC_Q_INDEX_TYPE_ERROR

(2394, X'95A') La cola tiene un tipo de índice incorrecto.

MQRC_Q_MGR_NAME_ERROR

(2058, X'80A') El nombre del gestor de colas no es válido o no se conoce.

MQRC_Q_MGR_NOT_AVAILABLE

(2059, X'80B') El gestor de colas no está disponible para efectuar la conexión.

MQRC_Q_MGR QUIESCING

(2161, X'871') El gestor de colas se está desactivando temporalmente.

MQRC_Q_MGR_STOPPING

(2162, X'872') El gestor de colas está concluyendo.

MQRC_RESOURCE_PROBLEM

(2102, X'836') No hay disponibles suficientes recursos del sistema.

MQRC_SIGNAL_OUTSTANDING

(2069, X'815') Símbolo pendiente para este descriptor de contexto.

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_SUPPRESSED_BY_EXIT

(2109, X'83D') El programa de salida ha suprimido la llamada.

MQRC_SYNCPOINT_NOT_AVAILABLE

(2072, X'818') No hay soporte para Syncpoint.

MQRC_UNEXPECTED_ERROR

(2195, X'893') Se ha producido un error inesperado.

MQRC_UOW_ENLISTMENT_ERROR

(2354, X'932') Ha fallado el listado en la unidad de trabajo global.

MQRC_UOW_MIX_NOT_SUPPORTED

(2355, X'933') No se soporta la mezcla de llamadas de unidad de trabajo.

MQRC_UOW_NOT_AVAILABLE

(2255, X'8CF') La unidad de trabajo no está disponible para ser utilizada por el gestor de colas.

MQRC_WAIT_INTERVAL_ERROR

(2090, X'82A') El intervalo de espera de MQGMO no es válido.

MQRC_WRONG_GMO_VERSION

(2256, X'8D0') Se ha suministrado una versión equivocada de MQGMO.

MQRC_WRONG_MD_VERSION

(2257, X'8D1') La versión del MQMD suministrado es incorrecta.

Para obtener información detallada sobre estos códigos, consulte [Mensajes y códigos de motivo](#).

Notas de uso

1. Las rutinas de devolución de llamada deben comprobar las respuestas de todos los servicios que invocan y, si la rutina detecta una condición que no se puede resolver, debe emitir un comando MQCB MQOP_DEREGISTER para evitar llamadas repetidas a la rutina de devolución de llamada.
2. Si utiliza la lectura asíncrona en una aplicación en la que el gestor de transacciones de XA gestiona transacciones globales, incluidas las actualizaciones de IBM MQ, debe tener en cuenta los siguientes aspectos adicionales:

- a. No es válido llamar a MQCTL(MQOP_START) para un **HConn**, después de que se haya creado, tras llamar a **xa_open**.

El motivo es que el objeto **HConn** se ha vinculado a un contexto de `XA`, por lo que no es posible acceder a él desde el hilo o hilos independientes que utiliza el mecanismo de consumo asíncrono.

- b. Si llama a MQCTL(MQOP_START) en ese escenario, la llamada falla con el código de motivo MQRC_ASYNC_XA_CONFLICT (2350).

- c. Es válido llamar a MQCTL(MQOP_START_WAIT) para un **HConn**, después de que se haya creado, tras llamar a **xa_open**.

El motivo es que este método de iniciar el mecanismo de consumo asíncrono hace que todas las devoluciones de llamada posteriores para el **HConn** se ejecuten en el subproceso donde se realiza la llamada MQCTL. Por lo tanto, el vínculo entre el **HConn** y el hilo no se pierde.

3.  En z/OS, cuando la operación es MQOP_START:

- Los programas que utilizan rutinas de devolución de llamada asíncronas deben estar autorizados para utilizar z/OS UNIX System Services (z/OS UNIX).
- Language Environment Los programas (LE) que utilizan rutinas de devolución de llamada asíncronas deben utilizar la opción de tiempo de ejecución LE « POSIX » (ON).
- Los programas que no sean LE y que utilicen rutinas de devolución de llamada asíncronas no deben utilizar la interfaz z/OS UNIX pthread_create (servicio invocable BPX1PTC).

4.  MQCTL no es compatible con el IMS adaptador.

Nota: En CICS, MQOP_START no es compatible. En su lugar, utilice la llamada a la función MQOP_START_WAIT.

Invocación en C

```
MQCTL (Hconn, Operation, &ControlOpts, &CompCode, &Reason)
```

Declare los parámetros como se indica a continuación:

```
MQHCONN  Hconn;          /* Connection handle */
MQLONG   Operation;      /* Operation being processed */
MQCTLO   ControlOpts     /* Options that control the action of MQCTL */
MQLONG   CompCode;       /* Completion code */
MQLONG   Reason;         /* Reason code qualifying CompCode */
```

Invocación en COBOL

```
CALL 'MQCTL' USING HCONN, OPERATION, CTLOPTS, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
** Connection handle
01 HCONN    PIC S9(9) BINARY.
** Operation
01 OPERATION PIC S9(9) BINARY.
```

```

** Control Options
01 CTLOPTS.
   COPY CMQCTLOV.
** Completion code
01 COMPCODE PIC S9(9) BINARY.
** Reason code qualifying COMPCODE
01 REASON   PIC S9(9) BINARY.

```

Invocación en PL/I

```
call MQCTL(Hconn, Operation, CtlOpts, CompCode, Reason)
```

Declare los parámetros como se indica a continuación:

```

dcl Hconn      fixed bin(31); /* Connection handle */
dcl Operation  fixed bin(31); /* Operation */
dcl CtlOpts like MQCTLO;      /* Options that control the action of MQCTL */
dcl CompCode   fixed bin(31); /* Completion code */
dcl Reason     fixed bin(31); /* Reason code qualifying CompCode */

```

MQDISC - Desconectar el gestor de colas

La llamada MQDISC rompe la conexión entre el gestor de colas y el programa de aplicación, y es la inversa de la llamada MQCONN o MQCONNX.

- En z/OS todas las aplicaciones que utilizan consumo de mensajes asíncronos, gestión de eventos o devolución de llamada, el subproceso de control principal debe emitir una llamada MQDISC antes de finalizar. Consulte [Consumo asíncrono de IBM MQ mensajes](#) para obtener más detalles.
- En z/OS, CICS las aplicaciones no necesitan emitir esta llamada para desconectarse del gestor de colas.

Si una CICS aplicación realiza esta llamada, no tendrá ningún efecto a menos que se haya realizado previamente una llamada MQCONNX, especificando uno de los siguientes:

```

MQCNO_SERIALIZE_CONN_TAG_Q_MGR
MQCNO_SERIALIZE_CONN_TAG_QSG
MQCNO_RESTRICT_CONN_TAG_Q_MGR o
MQCNO_RESTRICT_CONN_TAG_QSG

```

opciones, en cuyo caso se cierran todos los manejadores de objetos actualmente abiertos.

Sintaxis

MQDISC (*Hconn*, *CódigoComp*, *Motivo*)

Parámetros

Hconn

Tipo: MQHCONN - entrada/salida

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* fue devuelto por una llamada anterior a MQCONN o MQCONNX.

En z/OS para CICS aplicaciones, puede omitir la llamada MQCONN y especificar el siguiente valor para *Hconn*:

MQHC_DEF_HCONN

Manejador de conexión predeterminado.

Una vez completada con éxito la llamada, el gestor de colas establece *Hconn* en un valor que no es un identificador válido para el entorno. Este valor es:

MQHC_CONEXIÓN_INUTILIZABLE

Mango de conexión inutilizable.

En z/OS, *Hconn* se establece en un valor que no está definido.

CompCode

Tipo: MQLONG - salida

El código de finalización; es uno de los siguientes códigos:

MQCC_OK

Realización satisfactoria.

MQCC_WARNING

Aviso (finalización parcial).

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_WARNING:

MQRC_BACKED_OUT

(2003, X'7D3') Unidad de trabajo restituida.

MQRC_CONN_TAG_NO_LIBERADO

(2344, X'928') Etiqueta de conexión no liberada.

MQRC_RESULTADO_PENDIENTE

(2124, X'84C') El resultado de la operación de confirmación está pendiente.

Si *CompCode* es MQCC_FAILED:

ERROR DE CARGA DEL DISCO DEL ADAPTADOR MQRC

(2138, X'85A') No se puede cargar el módulo de desconexión del adaptador.

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'89C') El adaptador no está disponible.

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

MQRC_API_EXIT_ERROR

(2374, X'946') La salida de la API ha fallado.

MQRC_API_SALIDA_ERROR_INICIAL

(2375, X'947') Error en la inicialización de salida de la API.

MQRC_API_EXIT_TERM_ERROR

(2376, X'948') Error al finalizar la salida de la API.

MQRC_ASID_MISMATCH

(2157, X'86D') Los ASID primario e inicial varían.

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') La llamada MQI ha entrado antes de completar la llamada anterior.

MQRC_CONNECTION_BROKEN

(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

MQRC_CONNECTION_STOPPING

(2203, X'89B') La conexión está concluyendo.

MQRC_HCONN_ERROR

(2018, X'7E2') El manejador de conexión no es válido.

MQRC_RESULTADO_MIXTO

(2123, X'84B') El resultado de la operación de confirmación o retroceso es mixto.

MQRC_PAGESET_ERROR

(2193, X'891') Error al acceder al conjunto de datos del conjunto de páginas.

MQRC_Q_MGR_NAME_ERROR

(2058, X'80A') El nombre del gestor de colas no es válido o no se conoce.

MQRC_Q_MGR_NOT_AVAILABLE

(2059, X'80B') El gestor de colas no está disponible para efectuar la conexión.

MQRC_Q_MGR_STOPPING

(2162, X'872') El gestor de colas está concluyendo.

MQRC_RESOURCE_PROBLEM

(2102, X'836') No hay disponibles suficientes recursos del sistema.

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_UNEXPECTED_ERROR

(2195, X'893') Se ha producido un error inesperado.

Para obtener información detallada sobre estos códigos, consulte [Mensajes y códigos de motivo](#).

Notas de uso

1. Si se emite una llamada MQDISC cuando la conexión aún tiene objetos abiertos en esa conexión, el gestor de colas cierra esos objetos, con las opciones de cierre establecidas en MQCO_NONE.
2. Si la aplicación finaliza con cambios no confirmados en una unidad de trabajo, la disposición de dichos cambios depende de cómo finalice la aplicación:

- a. Si la aplicación emite la llamada MQDISC antes de finalizar:

- Para una unidad de trabajo coordinada por el gestor de colas, el gestor de colas emite la llamada MQCMIT en nombre de la aplicación. La unidad de trabajo se confirma si es posible y se cancela si no lo es.
- En el caso de una unidad de trabajo coordinada externamente, no se produce ningún cambio en el estado de la unidad de trabajo; sin embargo, el gestor de colas suele indicar que la unidad de trabajo debe confirmarse cuando así lo solicita el coordinador de la unidad de trabajo.

En z/OS, CICS, IMS (excepto los programas de lotes DL/1) y las aplicaciones RRS son así.

- b. Si la aplicación finaliza normalmente pero sin emitir la llamada MQDISC, la acción que se lleva a cabo depende del entorno:

- En z/OS, excepto para las aplicaciones MQJavaJMS o MQ, se producen las acciones descritas en la nota 2a.
- En todos los demás casos, se producen las acciones descritas en la nota 2c.

Debido a las diferencias entre entornos, asegúrese de que las aplicaciones que desea portar confirmen o deshagan la unidad de trabajo antes de finalizar.

- c. Si la aplicación finaliza *de forma anómala* sin emitir la llamada MQDISC, la unidad de trabajo se revierte.

3. En z/OS, se aplican los siguientes puntos:

- CICS Las aplicaciones no tienen que emitir la llamada MQDISC para desconectarse del gestor de colas, ya que el CICS propio sistema se conecta al gestor de colas y la llamada MQDISC no tiene ningún efecto sobre esta conexión.
- CICS, IMS (excepto los programas de procesamiento por lotes DL/1) y las aplicaciones RRS utilizan unidades de trabajo coordinadas por un coordinador externo de unidades de trabajo. Como resultado, la llamada MQDISC no afecta al estado de la unidad de trabajo (si la hay) que existe cuando se emite la llamada.

Sin embargo, la llamada MQDISC *indica* el fin del uso de la etiqueta de conexión *ConnTag* que estaba asociada a la conexión mediante una llamada MQCONN anterior emitida por la aplicación. Si hay una unidad de trabajo activa que hace referencia a la etiqueta de conexión cuando se emite la llamada MQDISC, la llamada se completa con el código de finalización MQCC_WARNING y el código

de motivo MQRC_CONN_TAG_NOT_RELEASED. La etiqueta de conexión no estará disponible para su reutilización hasta que el coordinador externo de la unidad de trabajo haya resuelto la unidad de trabajo.

Nota: En CICS, MQOP_START no es compatible. En su lugar, utilice la llamada a la función MQOP_START_WAIT.

Invocación en C

```
MQDISC (&Hconn, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQHCONN  Hconn;      /* Connection handle */
MQQLONG   CompCode;   /* Completion code */
MQQLONG   Reason;     /* Reason code qualifying CompCode */
```

Invocación en COBOL

```
CALL 'MQDISC' USING HCONN, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
** Connection handle
01 HCONN      PIC S9(9) BINARY.
** Completion code
01 COMPCODE   PIC S9(9) BINARY.
** Reason code qualifying COMPCODE
01 REASON     PIC S9(9) BINARY.
```

Invocación en PL/I

```
call MQDISC (Hconn, CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```
dcl Hconn      fixed bin(31); /* Connection handle */
dcl CompCode   fixed bin(31); /* Completion code */
dcl Reason     fixed bin(31); /* Reason code qualifying CompCode */
```

System/390 invitación del ensamblador

```
CALL MQDISC,(HCONN,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

```
HCONN      DS F Connection handle
COMPCODE   DS F Completion code
REASON     DS F Reason code qualifying COMPCODE
```

Invocación en Visual Basic

```
MQDISC Hconn, CompCode, Reason
```

Declare los parámetros como se indica a continuación:

```
Dim Hconn As Long 'Connection handle'
Dim CompCode As Long 'Completion code'
Dim Reason As Long 'Reason code qualifying CompCode'
```

MQDLTMH - Eliminar identificador de mensaje

La llamada MQDLTMH elimina un identificador de mensaje y es la inversa de la llamada MQCRTMH.

Sintaxis

MQDLTMH (*Hconn, Hmsg, DltMsgHOpts, CompCode, Reason*)

Parámetros

Hconn

Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas.

El valor debe coincidir con el identificador de conexión que se utilizó para crear el identificador de mensaje especificado en el **Hmsg** parámetro.

Si el identificador de mensaje se creó utilizando MQHC_UNASSOCIATED_HCONN, se debe establecer una conexión válida en el subproceso que elimina el identificador de mensaje; de lo contrario, la llamada fallará con MQRC_CONNECTION_BROKEN.

Hmsg

Tipo: MQHMSG - entrada/salida

Este es el identificador del mensaje que se va a eliminar. El valor fue devuelto por una llamada MQCRTMH anterior.

Una vez completada correctamente la llamada, el identificador se establece en un valor no válido para el entorno. Este valor es:

MQHM_INUTILIZABLE_HMSG

Identificador de mensaje no utilizable.

El identificador de mensaje no se puede eliminar si hay otra IBM MQ llamada en curso a la que se le haya pasado el mismo identificador de mensaje.

DltMsgHOpts

Tipo: MQDMHO - entrada

Consulte [MQDMHO](#) para obtener más detalles.

CompCode

Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'089C') Adaptador no disponible.

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

MQRC_ASID_MISMATCH

(2157, X'86D') Los ASID primario e inicial varían.

MQRC_CALL_IN_PROGRESS

(2219, X'08AB') Se ha iniciado una llamada al MQI antes de que finalizara la llamada anterior.

MQRC_CONNECTION_BROKEN

(2009, X'07D9') Se ha perdido la conexión con el gestor de colas.

MQRC_DMHO_ERROR

(2462, X'099E') Estructura de opciones de control de mensajes eliminada no válida.

MQRC_HMSG_ERROR

(2460, X'099C') El puntero del identificador del mensaje no es válido.

MQRC_MSG_HANDLE_EN_USO

(2499, X'09C3') El identificador del mensaje ya está en uso.

MQRC_OPTIONS_ERROR

(2046, X'07FE') Opciones no válidas o incoherentes.

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_UNEXPECTED_ERROR

(2195, X'893') Se ha producido un error inesperado.

Para obtener información detallada sobre estos códigos, consulte [Mensajes y códigos de motivo](#).

Invocación en C

```
MQDLTMH (Hconn, &Hmsg, &DltMsgHOpts, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQHCONN  Hconn;          /* Connection handle */
MQHMSG    Hmsg;          /* Message handle */
MQDMHO    DltMsgHOpts;  /* Options that control the action of MQDLTMH */
MQLONG    CompCode;      /* Completion code */
MQLONG    Reason;        /* Reason code qualifying CompCode */
```

Invocación en COBOL

```
CALL 'MQDLTMH' USING HCONN, HMSG, DLTMSGHOPTS, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
** Connection handle
01  HCONN    PIC S9(9) BINARY.

** Options that control the action of MQDLTMH
01  DLTMSGHOPTS.
COPY CMQDMHOL.

** Completion code
01  COMPCODE  PIC S9(9) BINARY.

** Reason code qualifying COMPCODE
01  REASON    PIC S9(9) BINARY.
```

Invocación en PL/I

```
call MQDLTMH (Hconn, Hmsg, DltMsgHOpts, CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```
dcl Hconn          /* Connection handle */
dcl Hmsg           /* Message handle */
```

```

dcl DltMsgHopts like MQDMHO; /* Options that control the action of MQDLTMH */
dcl CompCode /* Completion code */
dcl Reason /* Reason code qualifying CompCode */

```

Invocación en ensamblador de alto nivel

```
CALL MQDLTMH, (HCONN,HMSG,DLTMSGHOPTS,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

HCONN	DS	F	Connection handle
HMSG	DS	D	Message handle
DLTMSGHOPTS	CMQDMHOA	,	Options that control the action of MQDLTMH
COMPCODE	DS	F	Completion code
REASON	DS	F	Reason code qualifying COMPCODE

MQDLTMP - Eliminar propiedad del mensaje

La llamada MQDLTMP elimina una propiedad de un identificador de mensaje y es la inversa de la llamada MQSETMP.

Sintaxis

MQDLTMP (*Hconn*, *Hmsg*, *DltPropOpts*, *Nombre*, *CódigoComp*, *Motivo*)

Parámetros

Hconn

Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor debe coincidir con el identificador de conexión que se utilizó para crear el identificador de mensaje especificado en el **Hmsg** parámetro.

Si el identificador de mensaje se creó utilizando MQHC_UNASSOCIATED_HCONN, se debe establecer una conexión válida en el subproceso que elimina el identificador de mensaje; de lo contrario, la llamada fallará con MQRC_CONNECTION_BROKEN.

Hmsg

Tipo: MQHMSG - entrada

Este es el identificador del mensaje que contiene la propiedad que se va a eliminar. El valor fue devuelto por una llamada MQCRTMH anterior.

DltPropOpts

Tipo: MQDMPO - entrada

Consulte el tipo de datos [MQDMPO](#) para obtener más detalles.

Nombre

Tipo: MQCHARV - entrada

El nombre de la propiedad que se va a eliminar. Consulte [Nombres de propiedades](#) para obtener más información sobre los nombres de propiedades.

No se permiten comodines en el nombre de la propiedad.

CompCode

Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_WARNING

Aviso (finalización parcial).

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_WARNING:

MQRC_PROPIEDAD_NO_DISPONIBLE

(2471, X'09A7') Propiedad no disponible.

MQRC_RFH_FORMATO_ERROR

(2421, X'0975') No se pudo analizar una carpeta MQRFH2 que contiene propiedades.

Si *CompCode* es MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'089C') Adaptador no disponible.

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'0852') No se puede cargar el módulo de servicio del adaptador.

MQRC_ASID_MISMATCH

(2157, X'086D') Los ASID primarios y domésticos son diferentes.

MQRC_CALL_IN_PROGRESS

(2219, X'08AB') MQI : se ha iniciado una llamada antes de que finalizara la anterior.

MQRC_CONNECTION_BROKEN

(2009, X'07D9') Se ha perdido la conexión con el gestor de colas.

MQRC_DMPO_ERROR

(2481, X'09B1') Estructura de opciones de propiedad del mensaje eliminado no válida.

MQRC_HMSG_ERROR

(2460, X'099C') Identificador de mensaje no válido.

MQRC_MSG_HANDLE_EN_USO

(2499, X'09C3') El identificador del mensaje ya está en uso.

MQRC_OPTIONS_ERROR

(2046, X'07FE') Opciones no válidas o incoherentes.

ERROR_NOMBRE_PROPIEDAD_MQRC

(2442, X'098A') Nombre de propiedad no válido.

MQRC_SOURCE_CCSID_ERROR

(2111, X'083F') El identificador del conjunto de caracteres codificados del nombre de la propiedad no es válido.

MQRC_UNEXPECTED_ERROR

(2195, X'0893') Se ha producido un error inesperado.

Para obtener información detallada sobre estos códigos, consulte [Mensajes y códigos de motivo](#).

Invocación en C

```
MQDLTMP (Hconn, Hmsg, &DltPropOpts, &Name, &CompCode, &Reason)
```

Declare los parámetros como se indica a continuación:

```
MQHCONN Hconn;    /* Connection handle */
MQHMSG Hmsg;      /* Message handle */
```

```

MQDMPO DltPropOpts; /* Options that control the action of MQDLTMP */
MQCHARV Name; /* Property name */
MQLONG CompCode; /* Completion code */
MQLONG Reason; /* Reason code qualifying CompCode */

```

Invocación en COBOL

```
CALL 'MQDLTMP' USING HCONN, HMSG, DLTPROPOPTS, NAME, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```

** Connection handle
01 HCONN PIC S9(9) BINARY.
** Message handle
01 HMSG PIC S9(18) BINARY.
** Options that control the action of MQDLTMP
01 DLTPROPOPTS.
COPY CMQDMPOV.
** Property name
01 NAME.
COPY CMQCHRVV.
** Completion code
01 COMPCODE PIC S9(9) BINARY.
** Reason code qualifying COMPCODE
01 REASON PIC S9(9) BINARY.

```

Invocación en PL/I

```
call MQDLTMP (Hconn, Hmsg, DltPropOpts, Name, CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```

dcl Hconn          fixed bin(31); /* Connection handle */
dcl Hmsg           fixed bin(63); /* Message handle */
dcl DltPropOpts    like MQDMPO; /* Options that control the action of MQDLTMP */
dcl Name           like MQCHARV; /* Property name */
dcl CompCode       fixed bin(31); /* Completion code */
dcl Reason         fixed bin(31); /* Reason code qualifying CompCode */

```

Invocación en ensamblador de alto nivel

```
CALL MQDLTMP, (HCONN,HMSG,DLTPROPOPTS,NAME,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

HCONN	DS	F	Connection handle
HMSG	DS	D	Message handle
DLTPROPOPTS	CMQDMPOA	,	Options that control the action of MQDLTMP
NAME	CMQCHRVA	,	Property name
COMPCODE	DS	F	Completion code
REASON	DS	F	Reason code qualifying COMPCODE

Mensaje MQGET - get

La llamada MQGET recupera un mensaje de una cola local que se ha abierto utilizando la llamada MQOPEN.

Sintaxis

MQGET (*Hconn, Hobj, MsgDesc, GetMsgOpts, BufferLength, Buffer, DataLength, CompCode, Reason*)

Parámetros

Hconn

Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* fue devuelto por una llamada anterior a MQCONN o MQCONNX.

En z/OS para CICS aplicaciones, se puede omitir la llamada MQCONN y especificar el siguiente valor para *Hconn*:

MQHC_DEF_HCONN

Manejador de conexión predeterminado.

Hobj

Tipo: MQHOBJ - entrada

Este manejador representa la cola de la que debe recuperarse el mensaje. El valor de *Hobj* fue devuelto por una llamada MQOPEN anterior. La cola debe haberse abierto con una o varias de las opciones siguientes (consulte [“MQOPEN - Abrir objeto”](#) en la página 734 para obtener más detalles):

- MQOO_INPUT_SHARED
- MQOO_INPUT_EXCLUSIVE
- MQOO_INPUT_AS_Q_DEF
- MQOO_BROWSE

MsgDesc

Tipo: MQMD - entrada/salida

Esta estructura describe los atributos del mensaje necesario y los del mensaje recuperado. Consulte [“MQMD - Descriptor de mensaje”](#) en la página 389 para obtener los detalles.

Si *BufferLength* es menor que la longitud del mensaje, *MsgDesc* se rellena mediante el gestor de colas, independientemente de si se ha especificado MQGMO_ACCEPT_TRUNCATED_MSG en el **GetMsgOpts** parámetro (véase el campo MQGMO - Opciones).

Si la aplicación proporciona un MQMD version-1, el mensaje devuelto tiene un MQMDE prefijado a los datos del mensaje de la aplicación, pero solo si uno o más de los campos del MQMDE tienen un valor no predeterminado. Si todos los campos de MQMDE tienen valores predeterminados, se omitirá MQMDE. Un nombre de formato de MQFMT_MD_EXTENSION en el campo *Formato* en MQMD indica que hay un MQMDE.

La aplicación no tiene que proporcionar una estructura MQMD si se suministra un manejador de mensajes válido en el campo *MsgHandle*. Si no se proporciona ningún valor en este campo, el descriptor del mensaje se tomará del descriptor asociado a los manejadores de mensajes.

Si la aplicación proporciona un manejador de mensajes en lugar de una estructura MQMD, y especifica MQGMO_PROPERTIES_FORCE_MQRFH2, la llamada fallará con el código de razón MQRC_MD_ERROR. La llamada también falla, con el código de motivo MQRC_MD_ERROR, si la aplicación no proporciona una estructura MQMD y especifica MQGMO_PROPERTIES_AS_Q_DEF, y el **PropertyControl** atributo de cola es MQPROP_FORCE_MQRFH2.

Si se especifican opciones de coincidencia y se utiliza el descriptor de mensajes asociado con el manejador de mensajes, los campos de entrada que se utilizan para la comparación proceden del manejador de mensajes.

GetMsgOpts

Tipo: MQGMO - entrada/salida

Consulte [“MQGMO - Opciones para obtener mensajes”](#) en la página 331 para obtener los detalles.

BufferLength

Tipo: MQLONG - entrada

Esta es la longitud en bytes del área *Buffer* . Especifique cero para los mensajes que no tienen datos o si se va a eliminar el mensaje de la cola y a descartar los datos (en este caso debe especificar MQGMO_ACCEPT_TRUNCATED_MSG).

Nota: La longitud del mensaje más largo que se puede leer de la cola viene dada por el **MaxMsgLength** atributo queue; véase [“Atributos para colas”](#) en la página 841.

Almacenamiento intermedio

Tipo: MQBYTEExBufferLength - salida

Se trata del área que contiene los datos del mensaje. Alinee el almacenamiento intermedio en un límite adecuado según la naturaleza de los datos del mensaje. La alineación de 4 bytes es adecuada para la mayoría de los mensajes (incluidos los mensajes que contienen IBM MQ estructuras de encabezado), pero algunos mensajes pueden requerir una alineación más estricta. Por ejemplo, un ejemplo que contiene un entero binario de 64 bits podría requerir una alineación de 8 bytes.

Si *BufferLength* es menor que la longitud del mensaje, se traslada a toda la parte del mensaje **Buffer** que sea posible. Esto ocurre tanto si se especifica MQGMO_ACCEPT_TRUNCATED_MSG en el **GetMsgOpts** parámetro (consulte el campo MQGMO - Opciones para obtener más información).

El conjunto de caracteres y la codificación de los datos en **Buffer** vienen dados por los campos *CodedCharSetId* *Encoding* y devuelta en el **MsgDesc** parámetro. Si estos valores son distintos de los valores que necesita el destinatario, éste deberá convertir los datos de mensajes de la aplicación en el juego de caracteres y en la codificación necesarios. Se puede utilizar la opción MQGMO_CONVERT (con una salida escrita por el usuario, si es necesario) para convertir los datos de mensaje; consulte “MQGMO - Opciones para obtener mensajes” en la página 331 para obtener detalles de esta opción.

Nota: Todos los demás parámetros de la llamada MQGET están en el juego de caracteres y la codificación del gestor de colas local (indicados por el atributo del **CodedCharSetId** gestor de colas y MQENC_NATIVE).

Si falla la llamada, el contenido del almacenamiento también se podría haber modificado.

En el lenguaje de programación C, el parámetro se declara como pointer-to-void y se puede especificar cualquier tipo de datos como parámetro.

Si el **BufferLength** parámetro es cero, no *Buffer* se hace referencia a; en este caso, la dirección del parámetro pasada por programas escritos en C o en ensambladore System/390 e puede ser nula.

DataLength

Tipo: MQLONG - salida

Indica la longitud en bytes de los datos de la aplicación *en el mensaje*. Si este valor es mayor que *BufferLength*, solo se *BufferLength* devuelven bytes en el **Buffer** parámetro (es decir, el mensaje se trunca). Si el valor es cero, el mensaje no contendrá datos de aplicación.

Si *BufferLength* es menor que la longitud del mensaje, *DataLength* sigue siendo completado por el gestor de colas, independientemente de si se especifica MQGMO_ACCEPT_TRUNCATED_MSG en el **GetMsgOpts** parámetro (consulte el campo MQGMO - Opciones para obtener más información). Esto permite a la aplicación determinar el tamaño del almacenamiento intermedio necesario para introducir los datos del mensaje y después reemitir la llamada con un almacenamiento intermedio que tenga el tamaño adecuado.

Sin embargo, si se especifica la opción MQGMO_CONVERT y los datos del mensaje convertido son demasiado largos para caber en *Buffer*, el valor devuelto para *DataLength* es:

- La longitud de los datos *no convertidos*, para formatos definidos por el gestor de colas.

En este caso, si la naturaleza de los datos hace que se expandan durante la conversión, la aplicación debe asignar un búfer mayor que el valor devuelto por el gestor de colas para *DataLength*.

- El valor devuelto por la salida de conversión de datos, para formatos definidos por la aplicación.

CompCode

Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_WARNING

Aviso (finalización parcial).

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

Los códigos de motivo enumerados son los que el gestor de colas puede devolver para el **Reason** parámetro. Si la aplicación especifica la opción MQGMO_CONVERT y se invoca una salida escrita por el usuario para convertir algunos o todos los datos del mensaje, la salida decide qué valor se devuelve para el **Reason** parámetro. Como resultado, es posible tener valores distintos de los que se documentan.

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_WARNING:

MQRC_CONVERTED_MSG_TOO_BIG

(2120, X'848') Los datos convertidos son demasiado grandes para el almacenamiento intermedio.

MQRC_CONVERTED_STRING_TOO_BIG

(2190, X'88E') La serie convertida es demasiado grande para el campo.

MQRC_DBCS_ERROR

(2150, X'866') La serie DBCS no es válida.

MQRC_FORMAT_ERROR

(2110, X'83E') El formato de mensaje no válido.

MQRC_INCOMPLETE_GROUP

(2241, X'8C1') El grupo de mensajes no está completo.

MQRC_INCOMPLETE_MSG

(2242, X'8C2') El mensaje lógico no está completo.

MQRC_INCONSISTENT_CCSDS

(2243, X'8C3') Los segmentos del mensaje tienen distintos CCSID.

MQRC_INCONSISTENT_ENCODINGS

(2244, X'8C4') Los segmentos del mensaje tienen distintas codificaciones.

MQRC_INCONSISTENT_UOW

(2245, X'8C5') Especificación incoherente de unidad de trabajo.

MQRC_MSG_TOKEN_ERROR

(2331, X'91B') Uso no válido del token de mensaje.

MQRC_NO_MSG_LOCKED

(2209, X'8A1') No hay ningún mensaje bloqueado.

MQRC_NOT_CONVERTED

(2119, X'847') Los datos del mensaje no se han convertido.

MQRC_OPTIONS_CHANGED

(nnnn, X'xxx') Se han cambiado las opciones que tenían que ser coherentes.

MQRC_PARTIALLY_CONVERTED

(2272, X'8E0') Los datos del mensaje se han convertido parcialmente.

MQRC_SIGNAL_REQUEST_ACCEPTED

(2070, X'816') No se ha devuelto ningún mensaje (pero se acepta la solicitud de señal).

MQRC_SOURCE_BUFFER_ERROR

(2145, X'861') El parámetro de almacenamiento intermedio de origen no es válido.

MQRC_SOURCE_CCSDS_ERROR

(2111, X'83F') El identificador de juego de caracteres codificado origen no es válido.

MQRC_SOURCE_DECIMAL_ENC_ERROR

(2113, X'841') Codificación de decimal empaquetado en el mensaje no reconocida.

MQRC_SOURCE_FLOAT_ENC_ERROR

(2114, X'842') Codificación de coma flotante en el mensaje no reconocida.

MQRC_SOURCE_INTEGER_ENC_ERROR

(2112, X'840') Codificación de entero de origen no reconocida.

MQRC_SOURCE_LENGTH_ERROR

(2143, X'85F') Parámetro de longitud de origen no válido.

MQRC_TARGET_BUFFER_ERROR

(2146, X'862') Parámetro de almacenamiento intermedio de destino no válido.

MQRC_TARGET_CCSDID_ERROR

(2115, X'843') El identificador de juego de caracteres codificado de destino no es válido.

MQRC_TARGET_DECIMAL_ENC_ERROR

(2117, X'845') Codificación de decimal empaquetado especificada por receptor no reconocida.

MQRC_TARGET_FLOAT_ENC_ERROR

(2118, X'846') Codificación de coma flotante especificada por receptor no reconocida.

MQRC_TARGET_INTEGER_ENC_ERROR

(2116, X'844') Codificación de entero de destino no reconocida.

MQRC_TRUNCATED_MSG_ACCEPTED

(2079, X'81F') Se ha devuelto un mensaje truncado (el proceso se ha completado).

MQRC_TRUNCATED_MSG_FAILED

(2080, X'820') Se ha devuelto un mensaje truncado (el proceso no se ha completado).

Si *CompCode* es MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'89C') El adaptador no está disponible.

nMQRC_ADAPTER_CONV_LOAD_ERROR

(2133, X'855') No se han podido cargar módulos de servicio de conversión de datos.

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

MQRC_API_EXIT_ERROR

(2374, X'946') La salida de la API ha fallado.

MQRC_API_EXIT_LOAD_ERROR

(2183, X'887') No se ha podido cargar la salida de la API.

MQRC_ASID_MISMATCH

(2157, X'86D') Los ASID primario e inicial varían.

MQRC_BACKED_OUT

(2003, X'7D3') Unidad de trabajo restituida.

MQRC_BUFFER_ERROR

(2004, X'7D4') El parámetro almacenamiento intermedio no es válido.

MQRC_BUFFER_LENGTH_ERROR

(2005, X'7D5') El parámetro de longitud de almacenamiento intermedio no es válido.

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') La llamada MQI ha entrado antes de completar la llamada anterior.

MQRC_CF_NO_DISPONIBLE

(2345, X'929') Instalación de acoplamiento no disponible.

MQRC_CF_STRUC_FAILED

(2373, X'945') La estructura de recurso de acoplamiento ha fallado.

MQRC_CF_STRUC_IN_USE

(2346, X'92A') La estructura de recurso de acoplamiento se está utilizando.

MQRC_CF_STRUC_LIST_HDR_IN_USE

(2347, X'92B') La cabecera de lista de la estructura de recurso de acoplamiento se está utilizando.

MQRC_CICS_WAIT_FAILED

(2140, X'85C') Solicitud de espera rechazada por CICS.

MQRC_CONNECTION_BROKEN

(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

MQRC_CONNECTION_NOT_AUTHORIZED

(2217, X'8A9') No tiene autorización para la conexión.

MQRC_CONNECTION QUIESCING

(2202, X'89A') Conexión en fase de inmovilización.

MQRC_CONNECTION_STOPPING

(2203, X'89B') La conexión está concluyendo.

MQRC_CORREL_ID_ERROR

(2207, X'89F') Error de identificador de correlación.

MQRC_DATA_LENGTH_ERROR

(2010, X'7DA') El parámetro longitud de datos no es válido.

MQRC_DB2_NOT_AVAILABLE

(2342, X'926') Db2 subsistema no disponible.

MQRC_GET_INHIBITED

(2016, X'7E0') Se han inhibido las obtenciones para la cola.

MQRC_GLOBAL_UOW_CONFLICT

(2351, X'92F') Conflicto de unidades de trabajo global.

MQRC_GMO_ERROR

(2186, X'88A') No es válida la estructura de las opciones de obtención de mensaje.

MQRC_HANDLE_IN_USE_FOR_UOW

(2353, X'931') Descriptor de contexto que se utiliza para la unidad de trabajo global.

MQRC_HCONN_ERROR

(2018, X'7E2') El manejador de conexión no es válido.

MQRC_HOBJ_ERROR

(2019, X'7E3') El manejador de objeto no es válido.

MQRC_INCONSISTENT_BROWSE

(2259, X'8D3') La especificación de examinar es incoherente.

MQRC_INCONSISTENT_UOW

(2245, X'8C5') Especificación incoherente de unidad de trabajo.

MQRC_INVALID_MSG_UNDER_CURSOR

(2246, X'8C6') El mensaje bajo cursor no es válido para recuperación.

MQRC_LOCAL_UOW_CONFLICT

(2352, X'930') La unidad de trabajo global entra en conflicto con la unidad de trabajo local.

MQRC_MATCH_OPTIONS_ERROR

(2247, X'8C7') Las opciones de coincidencia no son válidas.

MQRC_MD_ERROR

(2026, X'7EA') El descriptor de mensaje no es válido.

MQRC_MSG_ID_ERROR

(2206, X'89E') Error de identificador de mensaje.

MQRC_MSG_SEQ_NUMBER_ERROR

(2250, X'8CA') El número de secuencia de mensaje no es válido.

MQRC_MSG_TOKEN_ERROR

(2331, X'91B') El uso del símbolo de mensaje no es válido.

MQRC_NO_MSG_AVAILABLE

(2033, X'7F1') No hay ningún mensaje disponible.

MQRC_NO_MSG_UNDER_CURSOR

(2034, X'7F2') El cursor para examinar no está situado en el mensaje.

MQRC_NOT_OPEN_FOR_BROWSE

(2036, X'7F4') La cola no se ha abierto para examen.

MQRC_NOT_OPEN_FOR_INPUT

(2037, X'7F5') La cola no se ha abierto para entrada.

MQRC_OBJECT_CHANGED

(2041, X'7F9') La definición de objeto ha cambiado desde que se ha abierto.

MQRC_OBJECT_DAMAGED

(2101, X'835') El objeto se ha dañado.

MQRC_OPTIONS_ERROR

(2046, X'7FE') Las opciones no son válidas o no son coherentes.

MQRC_PAGESET_ERROR

(2193, X'891') Error al acceder al conjunto de datos del conjunto de páginas.

MQRC_Q_DELETED

(2052, X'804') La cola se ha suprimido.

MQRC_Q_INDEX_TYPE_ERROR

(2394, X'95A') La cola tiene un tipo de índice incorrecto.

MQRC_Q_MGR_NAME_ERROR

(2058, X'80A') El nombre del gestor de colas no es válido o no se conoce.

MQRC_Q_MGR_NOT_AVAILABLE

(2059, X'80B') El gestor de colas no está disponible para efectuar la conexión.

MQRC_Q_MGR QUIESCING

(2161, X'871') El gestor de colas se está desactivando temporalmente.

MQRC_Q_MGR_STOPPING

(2162, X'872') El gestor de colas está concluyendo.

MQRC_RESOURCE_PROBLEM

(2102, X'836') No hay disponibles suficientes recursos del sistema.

MQRC_SECOND_MARK_NOT_ALLOWED

(2062, X'80E') Ya hay un mensaje marcado.

MQRC_SIGNAL_OUTSTANDING

(2069, X'815') Símbolo pendiente para este descriptor de contexto.

MQRC_SIGNAL1_ERROR

(2099, X'833') Campo de señal no válido.

MQRC_STORAGE_MEDIUM_FULL

(2192, X'890') El soporte de almacenamiento externo está lleno.

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_SUPPRESSED_BY_EXIT

(2109, X'83D') El programa de salida ha suprimido la llamada.

MQRC_SYNCPOINT_LIMIT_REACHED

(2024, X'7E8') No pueden manejarse más mensajes dentro de la unidad de trabajo actual.

MQRC_SYNCPOINT_NOT_AVAILABLE

(2072, X'818') El punto de sincronización de soporte no está disponible.

MQRC_UNEXPECTED_ERROR

(2195, X'893') Se ha producido un error inesperado.

MQRC_UOW_ENLISTMENT_ERROR

(2354, X'932') Ha fallado el listado en la unidad de trabajo global.

MQRC_UOW_MIX_NOT_SUPPORTED

(2355, X'933') No se soporta la mezcla de llamadas de unidad de trabajo.

MQRC_UOW_NOT_AVAILABLE

(2255, X'8CF') La unidad de trabajo no está disponible para ser utilizada por el gestor de colas.

MQRC_WAIT_INTERVAL_ERROR

(2090, X'82A') El intervalo de espera de MQGMO no es válido.

MQRC_WRONG_GMO_VERSION

(2256, X'8D0') Se ha suministrado una versión equivocada de MQGMO.

MQRC_WRONG_MD_VERSION

(2257, X'8D1') La versión del MQMD suministrado es incorrecta.

Para obtener información detallada sobre estos códigos, consulte [Mensajes y códigos de motivo](#).

Notas de uso

1. El mensaje recuperado se suprime normalmente de la cola. Esta supresión puede producirse como parte de la propia llamada MQGET o como parte de un punto de sincronización.

Las opciones de examen son: MQGMO_BROWSE_FIRST, MQGMO_BROWSE_NEXT y MQGMO_BROWSE_MSG_UNDER_CURSOR.

2. Si se especifica la opción MQGMO_LOCK con una de las opciones de examen, el mensaje examinado se bloquea para que sea visible sólo para este manejador.

Si se especifica la opción MQGMO_UNLOCK, se desbloquea un mensaje anteriormente bloqueado. En este caso no se recupera ningún mensaje, y los parámetros **MsgDesc**, **BufferLength**, **BufferDataLength** y no se comprueban ni se modifican.

3. Para las aplicaciones que emiten una llamada MQGET, se puede perder el mensaje recuperado si la aplicación termina de norma anormal o si se perdiera la conexión mientras se procesa la llamada. Este problema surge porque el sustituto que se ejecuta en la misma plataforma que el gestor de colas que emite la llamada MQGET en nombre de la aplicación no puede detectar la pérdida de la aplicación hasta que el sustituto está a punto de devolver el mensaje a la aplicación, después de que el mensaje haya sido eliminado de la cola. Este problema puede ocurrir tanto para mensajes persistentes como para mensajes no persistentes.

Para eliminar el riesgo de perder mensajes de esta forma, recupere siempre los mensajes en unidades de trabajo. Es decir, especificando la opción MQGMO_SYNCPOINT en la llamada MQGET y utilizando las llamadas MQCMIT o MQBACK para confirmar o para restituir la unidad de trabajo cuando finaliza el proceso del mensaje. Si se especifica MQGMO_SYNCPOINT y el cliente finaliza de forma anómala o la conexión se pierde, el sustituto restituye la unidad de trabajo en el gestor de colas y el mensaje se restablece en la cola. Para obtener más información sobre los puntos de sincronización, consulte [Consideraciones sobre los puntos de sincronización en IBM MQ aplicaciones](#).

Esta situación puede darse tanto con IBM MQ clientes como con aplicaciones que se ejecutan en la misma plataforma que el gestor de colas.

4. Si una aplicación pone una secuencia de mensajes en una determinada cola dentro de una única unidad de trabajo y, a continuación, confirma esa unidad de trabajo de forma satisfactoria, los mensajes están disponibles para la recuperación tal como se indica a continuación:
 - Si la cola es una *cola no compartida* (es decir, una cola local), todos los mensajes dentro de la unidad de trabajo estarán disponibles al mismo tiempo.
 - Si la cola es *compartida*, los mensajes dentro de la unidad de trabajo estarán disponibles en el orden en que se colocaron, pero no todos al mismo tiempo. Cuando el sistema está muy cargado, es posible que el primer mensaje de la unidad de trabajo se recupere correctamente, pero que la llamada MQGET para el segundo o posteriores mensajes en la unidad de trabajo fallen con MQRC_NO_MSG_AVAILABLE. Si esto ocurre, la aplicación debe esperar un poco y luego volverá a intentar la operación.
5. Si una aplicación coloca una secuencia de mensajes en la misma cola sin utilizar grupos de mensajes, el orden de dichos mensajes se conserva si se cumplen determinadas condiciones. Consulte [Notas de uso de MQPUT](#) para obtener detalles. Si las condiciones se cumplen, los mensajes se presentan a la aplicación receptora en el orden en que fueron enviados, si:
 - Sólo un receptor está obteniendo mensajes de la cola.

Si hay dos o más aplicaciones que obtienen mensajes de la cola, debe ponerse de acuerdo con el remitente sobre el mecanismo que se utilizará para identificar los mensajes que pertenecen a una secuencia. Por ejemplo, el remitente podría establecer todos los `CorrelId` campos de los mensajes de una secuencia con un valor exclusivo de esa secuencia de mensajes.

- El receptor no cambia deliberadamente el orden de recuperación, por ejemplo, especificando un o `MsgIdCorrelId` concreto.

Si la aplicación emisora pone los mensajes como un grupo de mensajes, los mensajes se presentan a la aplicación receptora en el orden correcto si la aplicación receptora especifica la opción `MQGMO_LOGICAL_ORDER` en la llamada `MQGET`. Para obtener más información sobre los grupos de mensajes, consulte:

- [MQMD - Campo MsgFlags](#)
- [MQPMO_LOGICAL_ORDER](#)
- [MQGMO_LOGICAL_ORDER](#)

Si el usuario está obteniendo mensajes de un grupo bajo el punto de sincronización, deben garantizar que se haya procesado todo el grupo antes de intentar finalizar la transacción.

6. Las aplicaciones deben comprobar el código de respuesta `MQFB_QUIT` en el `Feedback` campo del **MsgDesc** parámetro y finalizar si encuentran este valor. Consulte [MQMD - Campo Feedback](#) para obtener más información.
7. Si la cola identificada por `Hobj` se abrió con la opción `MQOO_SAVE_ALL_CONTEXT` y el código de finalización de la llamada `MQGET` es `MQCC_OK` o `MQCC_WARNING`, el contexto asociado al identificador de cola `Hobj` se establece en el contexto del mensaje que se ha recuperado (a menos que se haya establecido la opción `MQGMO_BROWSE_FIRST`, `MQGMO_BROWSE_NEXT` o `MQGMO_BROWSE_MSG_UNDER_CURSOR`, en cuyo caso el contexto se marca como no disponible).

Puede utilizar el contexto guardado en una llamada `MQPUT` o `MQPUT1` posterior especificando las opciones `MQPMO_PASS_IDENTITY_CONTEXT` o `MQPMO_PASS_ALL_CONTEXT`. Esto permite que el contexto del mensaje recibido se transfiera en su totalidad o en parte a otro mensaje (por ejemplo, cuando el mensaje se reenvía a otra cola). Para obtener más información sobre el contexto del mensaje, consulte [Contexto de mensaje](#).

8. Si se incluye la opción `MQGMO_CONVERT` en el **GetMsgOpts** parámetro, los datos del mensaje de aplicación se convierten a la representación solicitada por la aplicación receptora, antes de que los datos se coloquen en el **Buffer** parámetro:
 - El `Format` campo de la información de control del mensaje identifica la estructura de los datos de la aplicación, y los campos `CodedCharSetId` `Encoding` y de la información de control del mensaje especifican su identificador de juego de caracteres y su codificación.
 - La aplicación que emite la llamada `MQGET` especifica en los campos `CodedCharSetId` `Encoding` y del **MsgDesc** parámetro el identificador del juego de caracteres y la codificación a la que se deben convertir los datos del mensaje de la aplicación.

Cuando es necesario convertir los datos del mensaje, la conversión la realiza el propio gestor de colas o una salida escrita por el usuario, dependiendo del valor del `Format` campo en la información de control del mensaje:

- Los siguientes nombres de formato son formatos que convierte el gestor de colas; estos formatos se denominan formatos "incorporados":
 - `MQFMT_ADMIN`
 - `MQFMT_CICS` (z/OS solo)
 - `MQFMT_COMMAND_1`
 - `MQFMT_COMMAND_2`
 - `MQFMT_DEAD_LETTER_HEADER`
 - `MQFMT_DIST_HEADER`
 - `MQFMT_EVENT` versión 1

- MQFMT_EVENT versión 2 (z/OS solo)
 - MQFMT_IMS
 - MQFMT_IMS_VAR_STRING
 - MQFMT_MD_EXTENSION
 - MQFMT_PCF
 - MQFMT_REF_MSG_HEADER
 - MQFMT_RF_HEADER
 - MQFMT_RF_HEADER_2
 - MQFMT_STRING
 - MQFMT_TRIGGER
 - MQFMT_WORK_INFO_HEADER (z/OS solo)
 - MQFMT_XMIT_Q_HEADER
- El nombre de formato MQFMT_NONE es un valor especial que indica que la naturaleza de los datos del mensaje no está definida. Como consecuencia, el gestor de colas no intenta la conversión cuando se recupera el mensaje de la cola.

Nota: Si se especifica MQGMO_CONVERT en la llamada MQGET para un mensaje que tiene un nombre de formato MQFMT_NONE, y el juego de caracteres o la codificación del mensaje difiere de la especificada en el **MsgDesc** parámetro, el mensaje se devuelve en el **Buffer** parámetro (suponiendo que no haya otros errores), pero la llamada se completa con el código de finalización MQCC_WARNING y el código de motivo MQRC_FORMAT_ERROR.

Puede utilizar MQFMT_NONE cuando la naturaleza de los datos del mensaje significa que no requieren conversión o cuando las aplicaciones emisoras o receptoras han acordado entre ellas la forma de envío de los datos del mensaje.

- Los demás nombres de formato pasan el mensaje a una salida escrita por el usuario para su conversión. La salida tiene el mismo nombre que el formato, aparte de las adiciones específicas del entorno. Los nombres de formato especificados por el usuario no deben comenzar con las letras IBM MQ.

Consulte [“salida de conversión de datos” en la página 915](#) para obtener más información sobre la salida de conversión de datos.

Los datos de usuario del mensaje se pueden convertir entre cualquier juego de caracteres y codificación admitidos. Sin embargo, tenga en cuenta que, si el mensaje contiene una o más IBM MQ estructuras de encabezado, el mensaje no se puede convertir desde o hacia un juego de caracteres que tenga caracteres de doble byte o multibyte para cualquiera de los caracteres que son válidos en los nombres de cola. Si se intenta, se devuelven los códigos de razón MQRC_SOURCE_CCSID_ERROR o MQRC_TARGET_CCSID_ERROR y el mensaje sin convertir. El conjunto de caracteres Unicode UTF-16 es un ejemplo de este tipo de conjunto de caracteres.

En la devolución de la llamada MQGET, el código de razón siguiente indica que el mensaje se ha convertido correctamente:

- MQRC_NONE

El siguiente código de motivo indica que el mensaje podría haberse convertido correctamente; la aplicación debe comprobar los campos CodedCharSetId Encoding y del **MsgDesc** parámetro para averiguarlo:

- MQRC_TRUNCATED_MSG_ACCEPTED

Los demás códigos de razón indican que el mensaje no se ha convertido.

Nota: La interpretación de este código de motivo es válida para las conversiones realizadas por una salida escrita por el usuario solo si la salida se ajusta a las directrices de procesamiento descritas en [“salida de conversión de datos” en la página 915](#).

9. Si utilice la interfaz orientada a objetos para obtener mensajes, puede optar por no especificar un almacenamiento intermedio para guardar los datos de mensajes de una llamada MQGET. Cuando se recibe un mensaje utilizando una aplicación orientada a objetos sin restringir el tamaño del búfer de mensajes recibidos, la aplicación no falla con MQRC_CONVERTED_MSG_TOO_BIG y recibe el mensaje convertido. Esto se cumple en los entornos siguientes:

- .NET, incluidas las aplicaciones totalmente gestionadas
- C++
- Java (IBM MQ classes for Java)

Nota: Para todos los clientes, si el valor de `sharingConversations` es cero y el búfer es demasiado pequeño para recibir el mensaje convertido, se devuelve el mensaje sin convertir, con el código de motivo MQRC_CONVERTED_MSG_TOO_BIG. Para obtener más información `sharingConversations`, consulte [Uso de conversaciones compartidas en una aplicación cliente](#).

10. Para los formatos incorporados, el gestor de colas puede realizar la *conversión predeterminada* de series de caracteres en el mensaje cuando se especifica la opción MQGMO_CONVERT. La conversión predeterminada permite que, para convertir datos de series de caracteres, el gestor de colas utilice un juego de caracteres predeterminado especificado por la instalación que se aproxima al juego de conjunto de caracteres real. Como consecuencia, la llamada MQGET puede realizarse con éxito con el código de terminación MQCC_OK, en vez de terminar con MQCC_WARNING y los códigos de razón MQRC_SOURCE_CCSD_ERROR o MQRC_TARGET_CCSD_ERROR.

Nota: El resultado de utilizar un juego de caracteres aproximado para convertir datos de series de caracteres es que es posible que algunos se conviertan de forma incorrecta. Para evitarlo, en la serie utilice caracteres que sean comunes para el juego de caracteres actual y el juego de caracteres predeterminado.

La conversión predeterminada se aplica a los datos de los mensajes de la aplicación y a los campos de tipo carácter de las estructuras MQMD y MQMDE:

- La conversión predeterminada de los datos del mensaje de la aplicación solo se produce cuando se cumplen todas las siguientes condiciones:
 - La aplicación especifica MQGMO_CONVERT.
 - El mensaje contiene datos que deben convertirse de o a un juego de caracteres que no se admite.
 - La conversión predeterminada no estaba activada cuando se ha instalado o reiniciado el gestor de colas.
- La conversión predeterminada de los campos de caracteres de las estructuras MQMD y MQMDE se produce según convenga, si dicha conversión predeterminada está habilitada para el gestor de colas. La conversión se realiza aunque la aplicación no especifique la opción MQGMO_CONVERT en la llamada MQGET.

11. Para el lenguaje de programación Visual Basic, se aplican los puntos siguientes:

- Si el tamaño del **Buffer** parámetro es menor que la longitud especificada por el **BufferLength** parámetro, la llamada falla con el código de motivo MQRC_STORAGE_NOT_AVAILABLE.
- El **Buffer** parámetro se declara como de tipo `String`. Si los datos que se van a recuperar de la cola no son de tipo `String` utilice el llamada MQGETAny en lugar de MQGET.

La llamada MQGETAny tiene los mismos parámetros que la llamada MQGET, excepto que el **Buffer** parámetro se declara como de tipo `Any`, lo que permite recuperar cualquier tipo de datos. Sin embargo, esto significa que `Buffer` no se puede comprobar para garantizar que tenga al menos `BufferLength` bytes de tamaño.

12. No todas las opciones MQGET se admiten si la lectura anticipada está habilitada. En la tabla siguiente se indica qué opciones están permitidas y si se pueden modificar entre llamadas MQGET.

Tabla 584. Opciones MQGET permitidas cuando la lectura anticipada está habilitada			
	Permitidas cuando la lectura anticipada está habilitada y se pueden modificar entre llamadas MQGET	Permitido cuando la lectura anticipada está habilitada, pero no se puede modificar entre llamadas MQGET ^a	Opciones de MQGET que no están permitidas cuando la lectura anticipada está habilitada ^b
Valores MD de MQGET	MsgId ^c CorrelId ^c	Encoding CodedCharSetId	
Opciones MQGET MQGMO	MQGMO_WAIT MQGMO_NO_WAIT MQGMO_FAIL_IF_QUIESCING MQGMO_BROWSE_FIRST ^d MQGMO_BROWSE_NEXT ^d MQGMO_BROWSE_MESSAGE _BAJO_CURSOR ^d	MQGMO_SYNCPOINT_IF_PERSISTENT MQGMO_NO_SYNCPOINT MQGMO_ACCEPT_TRUNCATED_MSG MQGMO_CONVERT MQGMO_LOGICAL_ORDER MQGMO_COMPLETE_MSG MQGMO_ALL_MSGS_AVAILABLE MQGMO_ALL_SEGMENTS_AVAILABLE MQGMO_MARK_BROWSE_HANDLE MQGMO_MARK_BROWSE_CO_OP MQGMO_UNMARK_BROWSE_CO_OP MQGMO_UNMARK_BROWSE_HANDLE MQGMO_UNMARKED_BROWSE_MSG MQGMO_PROPERTIES_FORCE_MQRFH2 MQGMO_NO_PROPERTIES MQGMO_PROPERTIES_IN_HANDLE MQGMO_PROPERTIES_COMPATIBILITY	MQGMO_SET_SIGNAL MQGMO_SYNCPOINT MQGMO_MARK_SKIP _BACKOUT MQGMO_MSG_UNDER _CURSOR ^d MQGMO_LOCK MQGMO_UNLOCK
Valores MQGMO		MsgHandle	

- a. Si estas opciones se modifican entre llamadas MQGET se devuelve el código de razón MQRC_OPTIONS_CHANGED.
 - b. Si estas opciones se especifican en la primera llamada MQGET, la lectura anticipada está inhabilitada. Si estas opciones se especifican en una llamada MQGET posterior, se devuelve el código de razón MQRC_OPTIONS_ERROR.
 - c. Las aplicaciones cliente han de tener presente que si los valores MsgId y CorrelId se modifican entre llamadas MQGET, es posible que los mensajes con los valores anteriores ya se hayan enviado al cliente y permanezcan en el almacenamiento intermedio de lectura anticipada del cliente hasta que se consuman (o se depuren automáticamente).
 - d. La primera llamada MQGET determina si los mensajes se han de examinar u obtener de una cola cuando la lectura anticipada está habilitada. Si la aplicación intenta utilizar una combinación de opciones de examen y de obtención, se devuelve el código de razón MQRC_OPTIONS_CHANGED.
 - e. MQGMO_MSG_UNDER_CURSOR no es posible con la lectura anticipada. Los mensajes se pueden examinar u obtener cuando la lectura anticipada está habilitada, pero no se puede hacer ambas cosas a la vez.
13. Las aplicaciones pueden obtener mensajes sin confirmar de forma destructiva sólo si los mensajes se han colocado en la misma unidad de trabajo local desde la que se obtienen. Las aplicaciones no pueden obtener mensajes sin confirmar de forma no destructiva.
 14. Los mensajes en un cursor para examinar se pueden recuperar en una unidad de trabajo. No es posible recuperar de esta forma un mensaje sin confirmar.

Invocación en C

```
MQGET (Hconn, Hobj, &MsgDesc, &GetMsgOpts, BufferLength, Buffer,
      &DataLength, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQHCONN  Hconn;          /* Connection handle */
MQHOBJ   Hobj;           /* Object handle */
MQMD     MsgDesc;        /* Message descriptor */
MQGMO    GetMsgOpts;     /* Options that control the action of MQGET */
MQLONG   BufferLength;    /* Length in bytes of the Buffer area */
MQBYTE   Buffer[n];       /* Area to contain the message data */
MQLONG   DataLength;     /* Length of the message */
MQLONG   CompCode;       /* Completion code */
MQLONG   Reason;         /* Reason code qualifying CompCode */
```

Invocación en COBOL

```
CALL 'MQGET' USING HCONN, HOBJ, MSGDESC, GETMSGOPTS, BUFFERLENGTH,  
BUFFER, DATALENGTH, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
** Connection handle  
01 HCONN          PIC S9(9) BINARY.  
** Object handle  
01 HOBJ           PIC S9(9) BINARY.  
** Message descriptor  
01 MSGDESC.  
  COPY CMQMDV.  
** Options that control the action of MQGET  
01 GETMSGOPTS.  
  COPY CMQGMV.  
** Length in bytes of the BUFFER area  
01 BUFFERLENGTH PIC S9(9) BINARY.  
** Area to contain the message data  
01 BUFFER        PIC X(n).  
** Length of the message  
01 DATALENGTH   PIC S9(9) BINARY.  
** Completion code  
01 COMPCODE      PIC S9(9) BINARY.  
** Reason code qualifying COMPCODE  
01 REASON        PIC S9(9) BINARY.
```

Invocación en PL/I

```
call MQGET (Hconn, Hobj, MsgDesc, GetMsgOpts, BufferLength, Buffer,  
            DataLength, CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```
dc1 Hconn          fixed bin(31); /* Connection handle */  
dc1 Hobj           fixed bin(31); /* Object handle */  
dc1 MsgDesc        like MQMD;    /* Message descriptor */  
dc1 GetMsgOpts     like MQGMO;    /* Options that control the action of  
                                MQGET */  
dc1 BufferLength    fixed bin(31); /* Length in bytes of the Buffer  
                                area */  
dc1 Buffer          char(n);      /* Area to contain the message data */  
dc1 DataLength     fixed bin(31); /* Length of the message */  
dc1 CompCode       fixed bin(31); /* Completion code */  
dc1 Reason         fixed bin(31); /* Reason code qualifying CompCode */
```

Invocación en ensamblador de alto nivel

```
CALL MQGET, (HCONN,HOBJ,MSGDESC,GETMSGOPTS,BUFFERLENGTH,  
            BUFFER,DATALENGTH,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

HCONN	DS	F	Connection handle
HOBJ	DS	F	Object handle
MSGDESC	CMQMDA	,	Message descriptor
GETMSGOPTS	CMQGMOA	,	Options that control the action of MQGET
BUFFERLENGTH	DS	F	Length in bytes of the BUFFER area
BUFFER	DS	CL(n)	Area to contain the message data
DATALENGTH	DS	F	Length of the message
COMPCODE	DS	F	Completion code
REASON	DS	F	Reason code qualifying COMPCODE

Invocación en Visual Basic

```
MQGET Hconn, Hobj, MsgDesc, GetMsgOpts, BufferLength, Buffer,  
DataLength, CompCode, Reason
```

Declare los parámetros como se indica a continuación:

Dim Hconn	As Long	'Connection handle'
Dim Hobj	As Long	'Object handle'
Dim MsgDesc	As MQMD	'Message descriptor'
Dim GetMsgOpts	As MQGMO	'Options that control the action of MQGET'
Dim BufferLength	As Long	'Length in bytes of the Buffer area'
Dim Buffer	As String	'Area to contain the message data'
Dim DataLength	As Long	'Length of the message'
Dim CompCode	As Long	'Completion code'
Dim Reason	As Long	'Reason code qualifying CompCode'

MQINQ - Consultar atributos de objeto

La MQINQ llamada devuelve una matriz de números enteros y un conjunto de cadenas de caracteres que contienen los atributos de un objeto.

Los siguientes tipos de objetos son válidos:

- Gestor de colas
- Cola
- Lista de nombres
- Definición de proceso

Sintaxis

MQINQ (*Hconn*, *Hobj*, *SelectorCount*, *Selectors*, *IntAttrCount*, *IntAttrs*, *CharAttrLength*, *CharAttrs*, *CompCode*, *Reason*)

Parámetros

Hconn

Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* fue devuelto por una llamada MQCONN anterior a MQCONN o.

En z/OS para CICS aplicaciones, se puede omitir la MQCONN llamada y especificar el siguiente valor para *Hconn*:

MQHC_DEF_HCONN

Manejador de conexión predeterminado.

Hobj

Tipo: MQHOBJ - entrada

Este identificador representa el objeto (de cualquier tipo) con los atributos requeridos. El identificador debe ser devuelto por una llamada MQOPEN anterior que haya especificado la opción MQOO_INQUIRE.

SelectorCount

Tipo: MQLONG - entrada

Este es el recuento de selectores que se proporcionan en la *Selectors* matriz. Es el número de atributos que se van a devolver. Cero es un valor válido. El número máximo permitido es 256.

Selectores

Tipo: MQLONG x *SelectorCount* - entrada

Se trata de una matriz de **SelectorCount** selectores de atributos; cada selector identifica un atributo (entero o carácter) con un valor que es obligatorio.

Cada selector debe ser válido para el tipo de objeto que *Hobj* representa; de lo contrario, la llamada falla con el código de finalización MQCC_FAILED y el código de motivo MQRC_SELECTOR_ERROR.

En el caso especial de las colas:

- Si el selector no es válido para colas de ningún tipo, la llamada falla con el código de finalización MQCC_FAILED y el código de motivo MQRC_SELECTOR_ERROR.
- Si el selector solo se aplica a colas de tipos distintos al tipo del objeto, la llamada se realiza correctamente con el código de finalización MQCC_WARNING y el código de motivo MQRC_SELECTOR_NOT_FOR_TYPE.
- Si la cola consultada es una cola de clúster, los selectores válidos dependen de cómo se haya resuelto la cola; consulte “Notas de uso” en la página 721 para obtener más detalles.

Los selectores se pueden especificar en cualquier orden. Los valores de los atributos que corresponden a selectores de atributos enteros (MQIA_* selectors) se devuelven en *IntAttrs* en el mismo orden en que aparecen estos selectores en *Selectors*. Los valores de los atributos que corresponden a los selectores de atributos de caracteres (MQCA_* selectores) se devuelven en *CharAttrs* en el mismo orden en que aparecen dichos selectores. MQIA_* Los selectores pueden intercalarse con los MQCA_* selectores; solo es importante el orden relativo dentro de cada tipo.

Nota:

1. Los selectores de atributos enteros y de caracteres se asignan dentro de dos rangos diferentes; los MQIA_* selectores residen dentro del rango MQIA_FIRST hasta MQIA_LAST, y los MQCA_* selectores dentro del rango MQCA_FIRST hasta MQCA_LAST.

Para cada rango, las constantes MQIA_LAST_USED y MQCA_LAST_USED definen el valor más alto que acepta el gestor de colas.
2. Si todos los MQIA_* selectores aparecen primero, se pueden utilizar los mismos números de elemento para abordar los elementos correspondientes en las matrices *Selectors IntAttrs* y.
3. Si el **SelectorCount** parámetro es cero, no *Selectors* se hace referencia a. En este caso, la dirección del parámetro pasada por programas escritos en C o en ensambladore S/390 e podría ser nula.

Los atributos que se pueden consultar se enumeran en las siguientes tablas. Para los MQCA_* selectores, la constante que define la longitud en bytes de la cadena resultante en *CharAttrs* se proporciona entre paréntesis.

Las tablas siguientes enumeran los selectores, por objeto, en orden alfabético, de la siguiente manera:

- MQINQTabla 585 en la página 708 Selektores de atributos para colas
- MQINQTabla 586 en la página 711 Selektores de atributos para listas de nombres
- MQINQTabla 587 en la página 711 Selektores de atributos para definiciones de procesos
- MQINQTabla 588 en la página 712 Selektores de atributos para el gestor de colas

Tabla 585. MQINQ Selektores de atributos para colas		
Selector	Longitud del campo	Descripción
MQCA_ALTERATION_DATE	MQ_DATE_LENGTH	Fecha de la última modificación
MQCA_ALTERATION_TIME	MQ_TIME_LENGTH	Fecha de la última modificación
MQCA_BACKOUT_REQ_Q_NAME	MQ_Q_NAME_LENGTH	Nombre de la cola de espera por cancelación excesiva
MQCA_BASE_Q_NAME	MQ_Q_NAME_LENGTH	Nombre de la cola a la que apunta el alias
 MQCA_CF_STRUC_NAME	MQ_CF_STRUC_NAME_LENGTH	Nombre de estructura de recurso de acoplamiento
MQCA_CLUS_CHL_NAME	MQ_CHANNEL_NAME_LENGTH	Nombre del canal emisor del clúster que utiliza esta cola como cola de transmisión.

Tabla 585. MQINQ Selektors de atributos para colas (continuación)		
Selector	Longitud del campo	Descripción
MQCA_CLUSTER_NAME	MQ_CLUSTER_NAME_LENGTH	Nombre del clúster
MQCA_CLUSTER_NAMELIST	MQ_NAMELIST_NAME_LENGTH	Lista de nombres de clúster
MQCA_CREATION_DATE	MQ_CREATION_DATE_LENGTH	Fecha de creación de la cola
MQCA_CREATION_TIME	MQ_CREATION_TIME_LENGTH	Tiempo de creación de la cola
MQCA_CUSTOM	MQ_CUSTOM_LENGTH	El atributo personalizado para nuevas funciones
MQCA_INITIATION_Q_NAME	MQ_Q_NAME_LENGTH	Nombre cola iniciación
MQCA_PROCESS_NAME	MQ_PROCESS_NAME_LENGTH	Nombre de la definición del proceso
MQCA_Q_DESC	MQ_Q_DESC_LENGTH	Descripción de la cola
MQCA_Q_NAME	MQ_Q_NAME_LENGTH	Nombre de cola
MQCA_REMOTE_Q_MGR_NAME	MQ_Q_MGR_NAME_LENGTH	Nombre del gestor de colas remoto
MQCA_REMOTE_Q_NAME	MQ_Q_NAME_LENGTH	Nombre de la cola remota tal y como se conoce en el gestor de colas remotas
 MQCA_STORAGE_CLASS	MQ_STORAGE_CLASS_LENGTH	Nombre de la clase de almacenamiento
MQCA_TRIGGER_DATA	MQ_TRIGGER_DATA_LENGTH	Datos desencadenantes
MQCA_XMIT_Q_NAME	MQ_Q_NAME_LENGTH	Nombre de cola de transmisión
 MQIA_ACCOUNTING_Q	MQLONG	Controla la recopilación de datos contables para la cola
MQIA_BACKOUT_THRESHOLD	MQLONG	Umbral de restituciones
MQIA_CLWL_Q_PRIORITY	MQLONG	Prioridad de la cola
MQIA_CLWL_Q_RANK	MQLONG	Rango de la cola
MQIA_CLWL_USEQ	MQLONG	Utilizar colas remotas
MQIA_CURRENT_Q_DEPTH	MQLONG	Número de mensajes en la cola
MQIA_DEF_BIND	MQLONG	Enlace predeterminado
MQIA_DEF_INPUT_OPEN_OPTION	MQLONG	Opción predeterminada abierta para entrada
MQIA_DEF_PERSISTENCE	MQLONG	Persistencia de mensajes predeterminada
MQIA_DEF_PRIORITY	MQLONG	Prioridad de mensajes predeterminada
MQIA_DEFINITION_TYPE	MQLONG	Tipo de definición de cola
 MQIA_DIST_LISTS	MQLONG	Soporte de lista de distribución

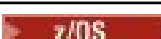
Tabla 585. MQINQ Selektors de atributos para colas (continuación)		
Selector	Longitud del campo	Descripción
MQIA_HARDEN_GET_BACKOUT	MQLONG	Si se debe endurecer el recuento de retrocesos
 MQIA_INDEX_TYPE	MQLONG	Tipo de índice mantenido para la cola
MQIA_INHIBIT_GET	MQLONG	Si se permiten las operaciones get
MQIA_INHIBIT_PUT	MQLONG	Si se permiten las operaciones de puesta
MQIA_MAX_MSG_LENGTH	MQLONG	Longitud máxima de mensaje
MQIA_MAX_Q_DEPTH	MQLONG	Número máximo de mensajes permitidos en la cola
MQIA_MSG_DELIVERY_SEQUENCE	MQLONG	Si la prioridad del mensaje es relevante
MQIA_NPM_CLASS	MQLONG	Nivel de fiabilidad para mensajes no persistentes
MQIA_OPEN_INPUT_COUNT	MQLONG	Número de MQOPEN llamadas que tienen la cola abierta para entrada
MQIA_OPEN_OUTPUT_COUNT	MQLONG	Número de MQOPEN llamadas que tienen la cola abierta para salida
MQIA_PROPERTY_CONTROL	MQLONG	Atributo de control de propiedad
 MQIA_Q_DEPTH_HIGH_EVENT	MQLONG	Atributo de control para eventos de profundidad de cola alta
 MQIA_Q_DEPTH_HIGH_LIMIT	MQLONG	Límite máximo para la profundidad de la cola
 MQIA_Q_DEPTH_LOW_EVENT	MQLONG	Atributo de control para eventos de profundidad de cola baja
 MQIA_Q_DEPTH_LOW_LIMIT	MQLONG	Límite inferior para la profundidad de la cola
 MQIA_Q_DEPTH_MAX_EVENT	MQLONG	Atributo de control para eventos máximos de profundidad de cola
 MQIA_Q_SERVICE_INTERVAL	MQLONG	Límite para el intervalo de servicio de cola
 MQIA_Q_SERVICE_INTERVAL_EVENT	MQLONG	Atributo de control para eventos de intervalo de servicio de cola
MQIA_Q_TYPE	MQLONG	Tipo de cola
 MQIA_QSG_DISP	MQLONG	Disposición de grupo de uso compartido de colas
MQIA_RETENTION_INTERVAL	MQLONG	Intervalo de retención de la cola
 MQIA_SCOPE	MQLONG	Ámbito de definición de la cola

Tabla 585. MQINQ Selektors de atributos para colas (continuación)		
Selector	Longitud del campo	Descripción
MQIA_SHAREABILITY	MQLONG	Si se puede compartir la cola para la entrada
 MQIA_STATISTICS_Q	MQLONG	Controla la recopilación de datos estadísticos para la cola
MQIA_TRIGGER_CONTROL	MQLONG	Activar control
MQIA_TRIGGER_DEPTH	MQLONG	Profundidad de desencadenante
MQIA_TRIGGER_MSG_PRIORITY	MQLONG	Prioridad de mensaje de umbral para desencadenantes
MQIA_TRIGGER_TYPE	MQLONG	Tipo de desencadenante
MQIA_USAGE	MQLONG	Utilización

Tabla 586. MQINQ Selektors de atributos para listas de nombres		
Selector	Longitud del campo	Descripción
MQCA_ALTERATION_DATE	MQ_DATE_LENGTH	Fecha de la última modificación
MQCA_ALTERATION_TIME	MQ_TIME_LENGTH	Fecha de la última modificación
MQCA_NAMELIST_DESC	MQ_NAMELIST_DESC_LENGTH	Descripción de la lista de nombres
MQCA_NAMELIST_NAME	MQ_NAMELIST_NAME_LENGTH	Nombre del objeto de la lista de nombres
 MQIA_NAMELIST_TYPE	MQLONG	Tipo de lista de nombres
MQCA_NAMES	MQ_Q_NAME_LENGTH <i>x Number of names in the list</i>	Nombres en la lista de nombres
MQIA_NAME_COUNT	MQLONG	Número de nombres en la lista de nombres
 MQIA_QSG_DISP	MQLONG	Disposición de grupo de uso compartido de colas

Tabla 587. MQINQ Selektors de atributos para definiciones de procesos		
Selector	Longitud del campo	Descripción
MQCA_ALTERATION_DATE	MQ_DATE_LENGTH	Fecha de la última modificación
MQCA_ALTERATION_TIME	MQ_TIME_LENGTH	Fecha de la última modificación
MQCA_APPL_ID	MQ_PROCESS_APPL_ID_LENGTH	Identificador de aplicación
MQCA_ENV_DATA	MQ_PROCESS_ENV_DATA_LENGTH	Datos de entorno
MQCA_PROCESS_DESC	MQ_PROCESS_DESC_LENGTH	Descripción de la definición del proceso
MQCA_PROCESS_NAME	MQ_PROCESS_NAME_LENGTH	Nombre de la definición del proceso

Tabla 587. MQINQ Selektors de atributos para definiciones de procesos (continuación)		
Selector	Longitud del campo	Descripción
MQCA_USER_DATA	MQ_PROCESS_USER_DATA_LENGTH	Datos de usuario
MQIA_APPL_TYPE	MQLONG	Tipo de aplicación
 MQIA_QSG_DISP	MQLONG	Disposición de grupo de uso compartido de colas

Tabla 588. MQINQ Selektors de atributos para el gestor de colas		
Selector	Longitud del campo	Descripción
MQCA_ALTERATION_DATE	MQ_DATE_LENGTH	Fecha de la última modificación
MQCA_ALTERATION_TIME	MQ_TIME_LENGTH	Fecha de la última modificación
MQCA_CHANNEL_AUTO_DEF_EXIT	MQ_EXIT_NAME_LENGTH	Nombre de salida de definición automática de canal
MQCA_CHINIT_SERVICE_PARM		Reservado para uso de IBM
MQCA_CLUSTER_WORKLOAD_DATA	MQ_EXIT_DATA_LENGTH	Datos pasados a la salida de la carga de trabajo del clúster
MQCA_CLUSTER_WORKLOAD_EXIT	MQ_EXIT_NAME_LENGTH	Nombre de la salida de carga de trabajo del clúster
MQCA_COMMAND_INPUT_Q_NAME	MQ_Q_NAME_LENGTH	Nombre de la cola de entrada de comandos del sistema
MQCA_CUSTOM	MQ_CUSTOM_LENGTH	El atributo personalizado para nuevas funciones
MQCA_DEAD_LETTER_Q_NAME	MQ_Q_NAME_LENGTH	Nombre de la cola de mensajes no entregados
MQCA_DEF_XMIT_Q_NAME	MQ_Q_NAME_LENGTH	Nombre predeterminado de la cola de transmisión
 MQCA_DNS_GROUP	MQ_DNS_GROUP_NAME_LENGTH	Nombre del grupo para el oyente TCP que gestiona las transmisiones entrantes para que el grupo de uso compartido de colas se una. El nombre se aplica cuando se utiliza Workload Manager Dynamic Domain Name Services.
 MQCA_IGQ_USER_ID	MQ_USER_ID_LENGTH	Identificador de usuario en cola dentro del grupo
MQCA_INITIAL_KEY	MQ_INITIAL_KEY_LENGTH	Clave inicial para el sistema de protección con contraseña Devuelve ***** si no está en blanco, o en blanco si se está utilizando la clave inicial predeterminada.
 MQCA_INSTALLATION_DESC	MQ_INSTALLATION_DESC_LENGTH	Descripción de la instalación asociada

Tabla 588. MQINQ Selektors de atributos para el gestor de colas (continuación)

Selector	Longitud del campo	Descripción
 MQCA_INSTALLATION_NAME	MQ_INSTALLATION_NAME_LENGTH	Nombre de la instalación asociada al gestor de colas
 MQCA_INSTALLATION_PATH	MQ_INSTALLATION_PATH_LENGTH	Ruta donde está instalado IBM MQ el asociado
 MQCA_LU_GROUP_NAME	MQ_LU_NAME_LENGTH	Nombre genérico de LU para el oyente de LU 6.2 que gestiona las transmisiones entrantes para que las utilice el grupo de uso compartido de colas
 MQCA_LU_NAME	MQ_LU_NAME_LENGTH	Nombre de la LU que se utilizará para las transmisiones salientes de la LU 6.2. Establezca este nombre en la misma LU que utiliza el oyente para las transmisiones entrantes
 MQCA_LU62_ARM_SUFFIX	MQ_ARM_SUFFIX_LENGTH	Sufijo del SYS1.PARMLIB miembro APPCPM <i>xx</i> , que nomina el LUADD para este iniciador de canal
MQCA_PARENT	MQ_Q_MGR_NAME_LENGTH	Nombre de un gestor de colas conectado jerárquicamente que se designa como padre de este gestor de colas
MQCA_Q_MGR_DESC	MQ_Q_MGR_DESC_LENGTH	Descripción del gestor de colas
MQCA_Q_MGR_IDENTIFIER	MQ_Q_MGR_IDENTIFIER_LENGTH	Identificador del gestor de colas (H)
MQCA_Q_MGR_NAME	MQ_Q_MGR_NAME_LENGTH	Nombre del administrador de colas local
 MQCA_QSG_NAME	MQ_QSG_NAME_LENGTH	Nombre del grupo de compartición de colas
MQCA_REPOSITORY_NAME	MQ_CLUSTER_NAME_LENGTH	Nombre del clúster para el que el gestor de colas proporciona servicios de repositorio
MQCA_REPOSITORY_NAMELIST	MQ_NAMELIST_NAME_LENGTH	Nombre del objeto de la lista de nombres que contiene los nombres de los clústeres para los que el gestor de colas proporciona servicios de repositorio
MQCA_SSL_KEY_REPO_PASSWORD	MQ_SSL_ENCRYPT_KEY_REPO_PWD_LEN	Contraseña del repositorio de claves Devuelve ***** si no está en blanco, o en blanco si no está configurado. Cifrado cuando se configura antes del almacenamiento.
 MQCA_TCP_NAME	MQ_TCP_NAME_LENGTH	Nombre del sistema « TCP/IP » que está utilizando
 MQIA_ACCOUNTING_CONN_OVERRIDE	MQLONG	Anular la configuración de contabilidad

Tabla 588. MQINQ Selektors de atributos para el gestor de colas (continuación)

Selector	Longitud del campo	Descripción
► Multi MQIA_ACCOUNTING_INTERVAL	MQLONG	Con qué frecuencia se deben llevar los registros contables intermedios
► Multi MQIA_ACCOUNTING_MQI	MQLONG	Controla la recopilación de información contable para los datos de MQI
► Multi MQIA_ACCOUNTING_Q	MQLONG	Controla la recopilación de información contable para las colas
► z/OS MQIA_ACTIVE_CHANNELS	MQLONG	Número máximo de canales que pueden estar activos en cualquier momento
► z/OS MQIA_ADOPTNEWMCA_CHECK	MQLONG	Elementos que se comprueban para determinar si se adopta un MCA. La comprobación se realiza cuando se detecta un nuevo canal entrante que tiene el mismo nombre que un MCA que ya está activo.
► Multi MQIA_ADOPTNEWMCA_INTERVAL	MQLONG	Tiempo, en segundos, que el nuevo canal espera a que finalice el canal huérfano
► z/OS MQIA_ADOPTNEWMCA_TYPE	MQLONG	Si se debe reiniciar automáticamente una instancia huérfana de un MCA de un tipo de canal concreto cuando se detecta una nueva solicitud de canal entrante que coincide con los AdoptNewMCACheck parámetros
► Multi MQIA_AUTHORITY_EVENT	MQLONG	Atributo de control para eventos de autoridad
► z/OS MQIA_BRIDGE_EVENT	MQLONG	Atributo de control para eventos de puente de IMS
► Multi MQIA_CHANNEL_AUTO_DEF	MQLONG	Atributo de control para la definición automática de canales
► Multi MQIA_CHANNEL_AUTO_DEF_EVENT	MQLONG	Atributo de control para eventos de definición automática de canales
MQIA_CHANNEL_EVENT	MQLONG	Atributo de control para eventos de canal
► z/OS MQIA_CHINIT_ADAPTERS	MQLONG	Número de subtareas del adaptador que se utilizarán para procesar IBM MQ las llamadas
► z/OS MQIA_CHINIT_DISPATCHERS	MQLONG	Número de despachadores que se utilizarán para el iniciador del canal
► z/OS MQIA_CHINIT_TRACE_AUTOSTART	MQLONG	Si se debe iniciar automáticamente el seguimiento del iniciador del canal

Tabla 588. MQINQ Selektors de atributos para el gestor de colas (continuación)

Selector	Longitud del campo	Descripción
► z/OS MQIA_CHINIT_TRACE_TABLE_SIZE	MQLONG	Tamaño del espacio de datos de rastreo (en MB) del iniciador del canal
MQIA_CLUSTER_WORKLOAD_LENGTH	MQLONG	Duración de la carga de trabajo del clúster.
MQIA_CLWL_MRU_CHANNELS	MQLONG	Número de canales utilizados más recientemente para el equilibrio de la carga de trabajo del clúster
MQIA_CLWL_USEQ	MQLONG	Utilizar colas remotas
MQIA_CODED_CHAR_SET_ID	MQLONG	Identificador de juego de caracteres codificado
MQIA_COMMAND_EVENT	MQLONG	Atributo de control para eventos de comando
MQIA_COMMAND_LEVEL	MQLONG	Nivel de comando compatible con el gestor de colas
► Multi MQIA_CONFIGURATION_EVENT	MQLONG	Atributo de control para eventos de configuración
MQIA_DEF_CLUSTER_XMIT_Q_TYPE	MQLONG	Tipo de cola de transmisión predeterminada que se debe utilizar para canales de clúster emisor.
► Multi MQIA_DIST_LISTS	MQLONG	Soporte de lista de distribución
► z/OS MQIA_DNS_WLM	MQLONG	Si el listener TCP que gestiona las transmisiones entrantes para el grupo de uso compartido de colas se registra en Workload Manager para los servicios de nombres de dominio dinámicos
► z/OS MQIA_EXPIRY_INTERVAL	MQLONG	Intervalo entre exploraciones de mensajes caducados
► z/OS MQIA_GROUP_UR	MQLONG	Atributo de control que determina si las unidades de recuperación GROUP están habilitadas para este gestor de colas. La unidad de disposición de recuperación GROUP solo está disponible si el gestor de colas es miembro de un grupo de uso compartido de colas
► z/OS MQIA_IGQ_PUT_AUTHORITY	MQLONG	Autoridad para poner colas dentro del grupo
► Multi MQIA_INHIBIT_EVENT	MQLONG	Atributo de control para inhibir eventos
► z/OS MQIA_INTRA_GROUP_QUEUING	MQLONG	Soporte para colas dentro del grupo

Tabla 588. MQINQ Selektors de atributos para el gestor de colas (continuación)

Selector	Longitud del campo	Descripción
 MQIA_LISTENER_TIMER	MQLONG	Intervalo de tiempo (en segundos) entre IBM MQ los intentos de reiniciar el listener si falla la operación « APPC » o « TCP/IP ».
 MQIA_LOCAL_EVENT	MQLONG	Atributo de control para eventos locales
 MQIA_LOGGER_EVENT	MQLONG	Atributo de control para inhibir eventos
 MQIA_LU62_CHANNELS	MQLONG	Número máximo de canales que pueden estar activos, o clientes que pueden estar conectados, utilizando el protocolo de transmisión LU 6.2
MQIA_MSG_MARK_BROWSE_INTERVAL	MQLONG	Intervalo de tiempo (en milisegundos) tras el cual el gestor de colas puede eliminar automáticamente una marca de los mensajes de exploración.  Atención: No debe establecer este valor por debajo del valor predeterminado de 5000.
 MQIA_MAX_CHANNELS	MQLONG	Número máximo de canales que pueden estar activos (incluidos los canales de conexión al servidor con clientes conectados)
MQIA_MAX_HANDLES	MQLONG	Número máximo de asas
MQIA_MAX_MSG_LENGTH	MQLONG	Longitud máxima de mensaje
MQIA_MAX_PRIORITY	MQLONG	Prioridad máxima
MQIA_MAX_UNCOMMITTED_MSGS	MQLONG	Número máximo de mensajes no confirmados dentro de una unidad de trabajo
 MQIA_OUTBOUND_PORT_MAX	MQLONG	Con MQIA_OUTBOUND_PORT_MIN, define el rango de números de puerto que se utilizarán al vincular los canales salientes
 MQIA_OUTBOUND_PORT_MIN	MQLONG	Con MQIA_OUTBOUND_PORT_MAX, define el rango de números de puerto que se utilizarán al vincular los canales salientes
 MQIA_PERFORMANCE_EVENT	MQLONG	Atributo de control para eventos de rendimiento
MQIA_PLATFORM	MQLONG	Plataforma en la que reside el gestor de colas
MQIA_PROT_POLICY_CAPABILITY	MQLONG	Indica si las capacidades de seguridad de Advanced Message Security están disponibles para un gestor de colas.

Tabla 588. MQINQ Selektors de atributos para el gestor de colas (continuación)

Selector	Longitud del campo	Descripción
MQIA_PUBSUB_MAXMSG_RETRY_COUNT	MQLONG	El número de intentos de reprocesar un mensaje de comando fallido en el punto de sincronización
MQIA_PUBSUB_MODE	MQLONG	Si el motor de publicación/suscripción y la interfaz de publicación/suscripción en cola están en funcionamiento. Las aplicaciones para publicar o suscribirse utilizando la interfaz de programación de aplicaciones requieren el motor de publicación/suscripción. Las colas supervisadas por la interfaz de publicación/suscripción en cola requieren que dicha interfaz esté en funcionamiento.
MQIA_PUBSUB_NP_MSG	MQLONG	Si se debe descartar (o conservar) un mensaje de entrada no entregado
MQIA_PUBSUB_NP_RESP	MQLONG	Controla el comportamiento de los mensajes de respuesta no entregados
MQIA_PUBSUB_SYNC_PT	MQLONG	Si solo se procesan los mensajes persistentes (o todos) bajo el punto de sincronización
 MQIA_QMGR_CFCONLOS	MQLONG	Especifica la acción que se debe realizar cuando el gestor de colas pierde la conectividad con la estructura de administración o con cualquier estructura CF con CFCONLOS establecido en ASQMGR
 MQIA_RECEIVE_TIMEOUT	MQLONG	¿Cuánto tiempo espera, aproximadamente, un canal « TCP/IP » para recibir datos, incluidos los «heartbeats», de su socio antes de volver al estado inactivo? El valor es numérico, calificado por MQIA_RECEIVE_TIMEOUT_TYPE.
 MQIA_RECEIVE_TIMEOUT_MIN	MQLONG	Tiempo mínimo que un canal « TCP/IP » espera para recibir datos, incluidos los latidos, de su interlocutor, antes de volver al estado inactivo
 MQIA_RECEIVE_TIMEOUT_TYPE	MQLONG	¿Cuánto tiempo espera, aproximadamente, un canal « TCP/IP » para recibir datos, incluidos los «heartbeats», de su socio antes de volver al estado inactivo? MQIA_RECEIVE_TIMEOUT_TYPE es el calificador aplicado a MQIA_RECEIVE_TIMEOUT.

Tabla 588. MQINQ Selektors de atributos para el gestor de colas (continuación)

Selector	Longitud del campo	Descripción
► Multi MQIA_REMOTE_EVENT	MQLONG	Atributo de control para eventos remotos
► z/OS MQIA_SECURITY_CASE	MQLONG	Caso de perfiles de seguridad
MQIA_SSL_EVENT	MQLONG	Atributo de control para eventos de canal
MQIA_SSL_FIPS_REQUIRED	MQLONG	Utilice únicamente FIPS algoritmos certificados para la criptografía
MQIA_SSL_RESET_COUNT	MQLONG	Recuento de restablecimiento de claves TLS
► Multi MQIA_START_STOP_EVENT	MQLONG	Atributo de control para eventos de inicio y parada
MQIA_STATISTICS_AUTO_CLUSTER	MQLONG	Controla la recopilación de información estadística de supervisión para los canales emisores del clúster
MQIA_STATISTICS_CHANNEL	MQLONG	Controla la recopilación de datos estadísticos para los canales
► Multi MQIA_STATISTICS_INTERVAL	MQLONG	Con qué frecuencia se deben escribir los datos de supervisión de estadísticas
► Multi MQIA_STATISTICS_MQI	MQLONG	Controla la recopilación de información estadística de supervisión para el gestor de colas
► Multi MQIA_STATISTICS_Q	MQLONG	Controla la recopilación de datos estadísticos para las colas
MQIA_SYNCPOINT	MQLONG	disponibilidad del punto de sincronización
► z/OS MQIA_TCP_CHANNELS	MQLONG	Número máximo de canales activos o de clientes conectados utilizando el protocolo de transmisión « TCP/IP »
► z/OS MQIA_TCP_KEEP_ALIVE	MQLONG	Si se debe utilizar la función KEEPALIVE de TCP para comprobar que el otro extremo de la conexión sigue estando disponible
► z/OS MQIA_TCP_STACK_TYPE	MQLONG	Si el iniciador del canal solo puede utilizar el espacio de direcciones TCP/IP especificado en TCPNAME, o si, opcionalmente, puede vincularse a cualquier dirección TCP/IP seleccionada
► z/OS MQIA_TRACE_ROUTE_RECORDING	MQLONG	Controla el registro de la información de seguimiento de ruta
MQIA_TREE_LIFE_TIME	MQLONG	Vida útil de los temas no administrativos no utilizados

Tabla 588. MQINQ Selektors de atributos para el gestor de colas (continuación)

Selector	Longitud del campo	Descripción
MQIA_TRIGGER_INTERVAL	MQLONG	Activar intervalo

IntAttrCount

Tipo: MQLONG - entrada

Este es el número de elementos de la *IntAttr*s matriz. Cero es un valor válido.

Si *IntAttrCount* es al menos el número de MQIA_* selectores en el **Selectors** parámetro, se devuelven todos los atributos enteros solicitados.

IntAttr

Tipo: MQLONG x *IntAttrCount* - salida

Se trata de una matriz de valores de *IntAttrCount* atributos enteros.

Los valores de los atributos enteros se devuelven en el mismo orden que los MQIA_* selectores del **Selectors** parámetro. Si la matriz contiene más elementos que el número de MQIA_* selectores, los elementos sobrantes no se modifican.

Si *Hobj* representa una cola, pero un selector de atributos no se aplica a ese tipo de cola, se devuelve MQIAV_NOT_APPLICABLE el valor específico. Se devuelve para el elemento correspondiente en la *IntAttr*s matriz.

Si el parámetro **IntAttrCountSelectorCount** o es cero, no *IntAttr*s se hace referencia a. En este caso, la dirección del parámetro pasada por programas escritos en C o en ensambladore S/390 e podría ser nula.

CharAttrLength

Tipo: MQLONG - entrada

Esta es la longitud en bytes del **CharAttr**s parámetro.

CharAttrLength debe ser al menos la suma de las longitudes de los atributos de caracteres solicitados (véase [Selectores](#)). Cero es un valor válido.

CharAttr

Tipo: MQCHAR x *CharAttrLength* - salida

Este es el búfer en el que se devuelven los atributos de los caracteres, concatenados entre sí. La longitud del búfer se proporciona en el parámetro **CharAttrLength**.

Los atributos de los caracteres se devuelven en el mismo orden que los MQCA_* selectores en el **Selectors** parámetro. La longitud de cada cadena de atributos es fija para cada atributo (véase [Selectores](#)), y el valor que contiene se rellena a la derecha con espacios en blanco si es necesario. Puede proporcionar un búfer más grande que el necesario para contener todos los atributos de caracteres solicitados y el relleno. Los bytes más allá del último valor de atributo devuelto no se modifican.

Si *Hobj* representa una cola, pero un selector de atributos no se aplica a ese tipo de cola, se devuelve una cadena de caracteres compuesta íntegramente por asteriscos (*). El asterisco se devuelve como el valor de ese atributo en *CharAttr*s.

Si el parámetro *CharAttrLengthSelectorCount* o es cero, no *CharAttr*s se hace referencia a. En este caso, la dirección del parámetro pasada por programas escritos en C o en ensambladore S/390 e podría ser nula.

CompCode

Tipo: MQLONG - salida

El código de finalización:

MQCC_OK

Realización satisfactoria.

MQCC_WARNING

Aviso (finalización parcial).

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000 ') No hay motivos para informar.

Si *CompCode* es MQCC_WARNING:

MQRC_CHAR_ATTRS_TOO_SHORT

(2008, X'7D8 ') No hay suficiente espacio para los atributos de los caracteres.

MQRC_INT_ATTR_COUNT_TOO_SMALL

(2022, X'7E6 ') No hay suficiente espacio para los atributos enteros.

MQRC_SELECTOR_NOT_FOR_TYPE

(2068, X'814 ') Selector no aplicable al tipo de cola.

Si *CompCode* es MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'89C ') Adaptador no disponible.

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852 ') No se puede cargar el módulo de servicio del adaptador.

MQRC_API_EXIT_ERROR

(2374, X'946 ') Error al salir de la API.

MQRC_API_EXIT_LOAD_ERROR

(2183, X'887 ') No se puede cargar la salida de la API.

MQRC_ASID_MISMATCH

(2157, X'86D ') Los ASID primarios y domésticos son diferentes.

MQRC_CALL_IN_PROGRESS

(2219, X'8AB ') Se ha iniciado una llamada al número MQI antes de que finalizara la llamada anterior.

MQRC_CF_STRUC_FAILED

(2373, X'945 ') Fallo en la estructura de la instalación de acoplamiento.

MQRC_CF_STRUC_IN_USE

(2346, X'92A ') Estructura de la instalación de acoplamiento en uso.

MQRC_CHAR_ATTR_LENGTH_ERROR

(2006, X'7D6 ') Longitud de los atributos de los caracteres no válida.

MQRC_CHAR_ATTRS_ERROR

(2007, X'7D7 ') Cadena de atributos de carácter no válida.

MQRC_CICS_WAIT_FAILED

(2140, X'85C ') Solicitud de espera rechazada por CICS.

MQRC_CONNECTION_BROKEN

(2009, X'7D9 ') Se ha perdido la conexión con el gestor de colas.

MQRC_CONNECTION_NOT_AUTHORIZED

(2217, X'8A9 ') No autorizado para la conexión.

MQRC_CONNECTION_STOPPING

(2203, X'89B ') Conexión cerrándose.

MQRC_HCONN_ERROR

(2018, X'7E2 ') Identificador de conexión no válido.

MQRC_HOBJ_ERROR

(2019, X'7E3') Identificador de objeto no válido.

MQRC_INT_ATTR_COUNT_ERROR

(2021, X'7E5') Recuento de atributos enteros no válido.

MQRC_INT_ATTRS_ARRAY_ERROR

(2023, X'7E7') Matriz de atributos enteros no válida.

MQRC_NOT_OPEN_FOR_INQUIRE

(2038, X'7F6') La cola no está abierta para consultas.

MQRC_OBJECT_CHANGED

(2041, X'7F9') La definición del objeto ha cambiado desde que se abrió.

MQRC_OBJECT_DAMAGED

(2101, X'835') Objeto dañado.

MQRC_PAGESET_ERROR

(2193, X'891') Error al acceder al conjunto de datos de la página.

MQRC_Q_DELETED

(2052, X'804') Cola eliminada.

MQRC_Q_MGR_NAME_ERROR

(2058, X'80A') El nombre del administrador de colas no es válido o se desconoce.

MQRC_Q_MGR_NOT_AVAILABLE

(2059, X'80B') El gestor de colas no está disponible para la conexión.

MQRC_Q_MGR_STOPPING

(2162, X'872') El gestor de colas se está apagando.

MQRC_RESOURCE_PROBLEM

(2102, X'836') Recursos del sistema insuficientes disponibles.

MQRC_SELECTOR_COUNT_ERROR

(2065, X'811') Recuento de selectores no válido.

MQRC_SELECTOR_ERROR

(2067, X'813') Selector de atributo no válido.

MQRC_SELECTOR_LIMIT_EXCEEDED

(2066, X'812') Número de selectores demasiado grande.

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') Espacio de almacenamiento insuficiente.

MQRC_SUPPRESSED_BY_EXIT

(2109, X'83D') Llamada suprimida por el programa de salida.

MQRC_UNEXPECTED_ERROR

(2195, X'893') Se ha producido un error inesperado.

Para obtener información detallada sobre estos códigos, consulte [Mensajes y códigos de motivo](#).

Notas de uso

1. Los valores devueltos son una instantánea de los atributos seleccionados. No hay garantía de que los atributos sigan siendo los mismos antes de que la aplicación pueda actuar sobre los valores devueltos.
2. Cuando se abre una cola de modelos, se crea una cola local dinámica. Se crea una cola local dinámica incluso si se abre la cola del modelo para consultar sus atributos.

Los atributos de la cola dinámica son, en gran medida, los mismos que los atributos de la cola modelo en el momento en que se crea la cola dinámica. Si luego utiliza la MQINQ llamada en esta cola, el gestor de colas devuelve los atributos de la cola dinámica, y no los atributos de la cola modelo. Consulte [Tabla 597 en la página 844](#) para obtener más información sobre qué atributos de la cola modelo hereda la cola dinámica.

3. Si el objeto consultado es una cola alias, los valores de atributo devueltos por la MQINQ llamada son los atributos de la cola alias. No son los atributos de la cola base o del tema al que resuelve el alias.
4. Si el objeto consultado es una cola de clúster, los atributos que se pueden consultar dependen de cómo se abra la cola:

- Puede abrir una cola de clúster para consultar más una o varias de las operaciones de entrada, exploración o configuración. Para ello, debe existir una instancia local de la cola del clúster para que la apertura se realice correctamente. En este caso, los atributos que se pueden consultar son los atributos válidos para las colas locales.

Si la cola del clúster está abierta para consultas sin entrada, exploración o configuración especificadas, la llamada devuelve el código de finalización MQCC_WARNING y el código de motivo MQRC_SELECTOR_NOT_FOR_TYPE (2068) si se intenta consultar atributos que solo son válidos para colas locales y no para colas de clúster.

- Puede abrir una cola de clúster para realizar consultas pasando el nombre del gestor de colas base del gestor de colas conectado.

Para ello, debe existir una instancia local de la cola del clúster para que la apertura se realice correctamente. Si no se pasa el gestor de colas base, la llamada devuelve el código de finalización MQCC_WARNING y el código de motivo MQRC_SELECTOR_NOT_FOR_TYPE (2068) si se intenta consultar atributos que solo son válidos para colas locales y no para colas de clúster.

- Si la cola del clúster se abre solo para consulta, o para consulta y salida, solo se pueden consultar los atributos enumerados. El **QType** atributo tiene el valor MQQT_CLUSTER en este caso:

- MQCA_Q_DESC
- MQCA_Q_NAME
- MQIA_DEF_BIND
- MQIA_DEF_PERSISTENCE
- MQIA_DEF_PRIORITY
- MQIA_INHIBIT_PUT
- MQIA_Q_TYPE

Puede abrir la cola del clúster sin ningún enlace fijo. Puede abrirlo con MQOO_BIND_NOT_FIXED especificado en la MQOPEN llamada. Como alternativa, especifique MQOO_BIND_AS_Q_DEFy establezca el **DefBind** atributo de la cola en MQBND_BIND_NOT_FIXED. Si se abre una cola de clúster sin enlace fijo, las MQINQ llamadas sucesivas a la cola pueden consultar diferentes instancias de la cola de clúster. Sin embargo, es habitual que todas las instancias tengan los mismos valores de atributos.

- Se puede definir un objeto de cola alias para un clúster. Dado que TARGTYPE y TARGET no son atributos de clúster, el proceso que realiza un MQOPEN proceso en la cola de alias no conoce el objeto al que resuelve el alias.

Durante la fase inicial MQOPEN, la cola alias se resuelve en un gestor de colas y una cola en el clúster. La resolución de nombres se lleva a cabo de nuevo en el gestor de colas remoto y es aquí donde se resuelve el TARGTYPE de la cola alias.

Si la cola de alias se resuelve en un alias de tema, la publicación de los mensajes enviados a la cola de alias se lleva a cabo en este gestor de colas remoto.

Ver colas de clúster

5. Es posible que desee consultar varios atributos y, a continuación, establecer algunos de ellos mediante la MQSET llamada. Para programar consultas y ajustes de manera eficiente, coloque los atributos que se van a ajustar al principio de las matrices del selector. Si lo hace, se pueden utilizar las mismas matrices con recuentos reducidos para MQSET.
6. Si se produce más de una de las situaciones de advertencia (véase el **CompCode** parámetro), el código de motivo devuelto es el primero de la siguiente lista que sea aplicable:
 - a. MQRC_SELECTOR_NOT_FOR_TYPE

b. MQRC_INT_ATTR_COUNT_TOO_SMALL

c. MQRC_CHAR_ATTRS_TOO_SHORT

7. El siguiente tema contiene información sobre los atributos de los objetos:

- [“Atributos para colas” en la página 841](#)
- [“Atributos de las listas de nombres” en la página 876](#)
- [“Atributos de las definiciones de proceso” en la página 878](#)
- [“Atributos para el gestor de colas” en la página 801](#)

Invocación en C

```
MQINQ (Hconn, Hobj, SelectorCount, Selectors, IntAttrCount, IntAttrs,  
       CharAttrLength, CharAttrs, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQHCONN  Hconn;           /* Connection handle */  
MQHOBJ   Hobj;            /* Object handle */  
MQLONG   SelectorCount;   /* Count of selectors */  
MQLONG   Selectors[n];    /* Array of attribute selectors */  
MQLONG   IntAttrCount;    /* Count of integer attributes */  
MQLONG   IntAttrs[n];     /* Array of integer attributes */  
MQLONG   CharAttrLength;  /* Length of character attributes buffer */  
MQCHAR   CharAttrs[n];    /* Character attributes */  
MQLONG   CompCode;        /* Completion code */  
MQLONG   Reason;          /* Reason code qualifying CompCode */
```

Invocación en COBOL

```
CALL 'MQINQ' USING HCONN, HOBJ, SELECTORCOUNT, SELECTORS-TABLE,  
                  INTATTRCOUNT, INTATTRS-TABLE, CHARATTRLENGTH,  
                  CHARATTRS, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
** Connection handle  
01 HCONN          PIC S9(9) BINARY.  
** Object handle  
01 HOBJ           PIC S9(9) BINARY.  
** Count of selectors  
01 SELECTORCOUNT PIC S9(9) BINARY.  
** Array of attribute selectors  
01 SELECTORS-TABLE.  
02 SELECTORS      PIC S9(9) BINARY OCCURS n TIMES.  
** Count of integer attributes  
01 INTATTRCOUNT PIC S9(9) BINARY.  
** Array of integer attributes  
01 INTATTRS-TABLE.  
02 INTATTRS       PIC S9(9) BINARY OCCURS n TIMES.  
** Length of character attributes buffer  
01 CHARATTRLENGTH PIC S9(9) BINARY.  
** Character attributes  
01 CHARATTRS      PIC X(n).  
** Completion code  
01 COMPCODE        PIC S9(9) BINARY.  
** Reason code qualifying COMPCODE  
01 REASON          PIC S9(9) BINARY.
```

Invocación en PL/I

```
call MQINQ (Hconn, Hobj, SelectorCount, Selectors, IntAttrCount,  
            IntAttrs, CharAttrLength, CharAttrs, CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```

dcl Hconn          fixed bin(31); /* Connection handle */
dcl Hobj           fixed bin(31); /* Object handle */
dcl SelectorCount  fixed bin(31); /* Count of selectors */
dcl Selectors(n)   fixed bin(31); /* Array of attribute selectors */
dcl IntAttrCount   fixed bin(31); /* Count of integer attributes */
dcl IntAttrs(n)    fixed bin(31); /* Array of integer attributes */
dcl CharAttrLength fixed bin(31); /* Length of character attributes
                                buffer */
dcl CharAttrs      char(n);       /* Character attributes */
dcl CompCode       fixed bin(31); /* Completion code */
dcl Reason         fixed bin(31); /* Reason code qualifying
                                CompCode */

```

Invocación en ensamblador de alto nivel

```

CALL MQINQ, (HCONN, HOBJ, SELECTORCOUNT, SELECTORS, INTATTRCOUNT, X
            INTATTRS, CHARATTRLENGTH, CHARATTRS, COMPCODE, REASON)

```

Declare los parámetros como se indica a continuación:

HCONN	DS	F	Connection handle
HOBJ	DS	F	Object handle
SELECTORCOUNT	DS	F	Count of selectors
SELECTORS	DS	(n)F	Array of attribute selectors
INTATTRCOUNT	DS	F	Count of integer attributes
INTATTRS	DS	(n)F	Array of integer attributes
CHARATTRLENGTH	DS	F	Length of character attributes buffer
CHARATTRS	DS	CL(n)	Character attributes
COMPCODE	DS	F	Completion code
REASON	DS	F	Reason code qualifying COMPCODE

Invocación en Visual Basic

```

MQINQ Hconn, Hobj, SelectorCount, Selectors, IntAttrCount, IntAttrs,
CharAttrLength, CharAttrs, CompCode, Reason

```

Declare los parámetros como se indica a continuación:

Dim Hconn	As Long	'Connection handle'
Dim Hobj	As Long	'Object handle'
Dim SelectorCount	As Long	'Count of selectors'
Dim Selectors	As Long	'Array of attribute selectors'
Dim IntAttrCount	As Long	'Count of integer attributes'
Dim IntAttrs	As Long	'Array of integer attributes'
Dim CharAttrLength	As Long	'Length of character attributes buffer'
Dim CharAttrs	As String	'Character attributes'
Dim CompCode	As Long	'Completion code'
Dim Reason	As Long	'Reason code qualifying CompCode'

MQINQMP - Consultar propiedad del mensaje

La llamada MQINQMP devuelve el valor de una propiedad de un mensaje.

Sintaxis

MQINQMP (*Hconn*, *Hmsg*, *InqPropOpts*, *Nombre*, *PropDesc*, *Tipo*, *LongitudDelValor*, *Valor*, *LongitudDeLosDatos*, *CódigoDeCompilación*, *Motivo*)

Parámetros

Hconn

Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* debe coincidir con el identificador de conexión que se utilizó para crear el identificador de mensaje especificado en el **Hmsg** parámetro.

Si el identificador del mensaje se creó utilizando MQHC_UNASSOCIATED_HCONN, se debe establecer una conexión válida en el subproceso que consulta una propiedad del identificador del mensaje; de lo contrario, la llamada fallará con MQRC_CONNECTION_BROKEN.

Hmsg

Tipo: MQHMSG - entrada

Este es el identificador del mensaje que se va a consultar. El valor fue devuelto por una llamada MQCRTMH anterior.

InqPropOpts

Tipo: MQIMPO - entrada/salida

Consulte el tipo de datos [MQIMPO](#) para obtener más detalles.

Nombre

Tipo: MQCHARV - entrada/salida

El nombre de la propiedad sobre la que desea realizar la consulta.

Si no se encuentra ninguna propiedad con este nombre, la llamada falla con el motivo MQRC_PROPERTY_NOT_AVAILABLE.

Puede utilizar el carácter comodín signo de porcentaje (%) al final del nombre de la propiedad. El comodín coincide con cero o más caracteres, incluido el carácter punto (.). Esto permite a una aplicación consultar el valor de muchas propiedades. Llame a MQINQMP con la opción MQIMPO_INQ_FIRST para obtener la primera propiedad coincidente y vuelva a llamarlo con la opción MQIMPO_INQ_NEXT para obtener la siguiente propiedad coincidente. Cuando no hay más propiedades coincidentes disponibles, la llamada falla con MQRC_PROPERTY_NOT_AVAILABLE. Si el *ReturnedName* campo de la estructura InqPropOpts se inicializa con una dirección o desplazamiento para el nombre devuelto de la propiedad, esto se completa al regresar de MQINQMP con el nombre de la propiedad que se ha coincidido. Si el *VSBuFSIZE* campo de la estructura InqPropOpts es menor que la longitud del nombre *ReturnedName* de la propiedad devuelta, el código de finalización se establece en MQCC_FAILED con el motivo MQRC_PROPERTY_NAME_TOO_BIG.

Las propiedades que tienen sinónimos conocidos se devuelven de la siguiente manera:

1. Propiedades con el prefijo «mqps» se devuelven como el nombre IBM MQ de la propiedad. Por ejemplo, "MQTopicString" es el nombre devuelto en lugar de « mqps.Top »
2. Propiedades con el prefijo «jms» o «mcd» se devuelven como el nombre del JMS campo del encabezado, por ejemplo, «JMSExpiration» es el nombre devuelto en lugar de « jms.Exp ».
3. Propiedades con el prefijo «usr» se devuelven sin ese prefijo, por ejemplo, se devuelve «Color» en lugar de « usr.Color ».

Las propiedades con sinónimos solo se devuelven una vez.

En el lenguaje de programación C, se definen las siguientes variables macro para consultar todas las propiedades y, a continuación, todas las propiedades que comienzan por «usr.»:

MQPROP_CONSULTAR_TODO

Consultar todas las propiedades del mensaje.

MQPROP_INQUIRE_ALL se puede utilizar de la siguiente manera:

```
MQCHARV Name = {MQPROP_INQUIRE_ALL};
```

MQPROP_CONSULTAR_TODOS_USUARIOS

Consulte todas las propiedades del mensaje que comienzan por «usr.». El nombre devuelto se devuelve sin «usr» .

Si se especifica MQIMP_INQ_NEXT, pero el nombre ha cambiado desde la llamada anterior o se trata de la primera llamada, entonces se implica MQIMPO_INQ_FIRST.

Consulte [Nombres de propiedades](#) y [Restricciones de nombres de propiedades](#) para obtener más información sobre el uso de nombres de propiedades.

PropDesc

Tipo: MQPD - salida

Esta estructura se utiliza para definir los atributos de una propiedad, incluyendo lo que ocurre si la propiedad no es compatible, a qué contexto de mensajes pertenece la propiedad y en qué mensajes debe copiarse la propiedad. Consulte [MQPD](#) para obtener más detalles sobre esta estructura.

Tipo

Tipo: MQLONG - entrada/salida

Al regresar de la llamada MQINQMP, este parámetro se establece en el tipo de datos *Valor*. El tipo de datos puede ser cualquiera de los siguientes:

MQTYPE_BOOLEAN

Un valor booleano.

MQTYPE_BYTE_STRING

una cadena de bytes.

MQTYPE_INT8

Un entero con signo de 8 bits.

MQTYPE_INT16

Un entero con signo de 16 bits.

MQTYPE_INT32

Un entero con signo de 32 dígitos.

MQTYPE_INT64

Un entero con signo de 64 bits.

MQTYPE_FLOAT32

Un número de coma flotante de 32 bits.

MQTYPE_FLOAT64

Un número de coma flotante de 64 bits.

MQTYPE_STRING

Una cadena de caracteres.

MQTYPE_NULL

La propiedad existe, pero tiene un valor nulo.

Si no se reconoce el tipo de datos del valor de la propiedad, se devuelve MQTYPE_STRING y se coloca una representación de cadena del valor en el área *Valor*. Una representación de cadena del tipo de datos se puede encontrar en el *TypeString* campo del *InqPropOpts* parámetro. Se devuelve un código de finalización de advertencia con el motivo MQRC_PROP_TYPE_NOT_SUPPORTED.

Además, si se especifica la opción MQIMPO_CONVERT_TYPE, se solicita la conversión del valor de la propiedad. Utilice *Tipo* como entrada para especificar el tipo de datos en el que desea que se devuelva la propiedad. Consulte la descripción de la opción [MQIMPO_CONVERT_TYPE](#) de la estructura [MQIMPO](#) para obtener más información sobre la conversión de tipos de datos.

Si no solicita la conversión de tipos, puede utilizar el siguiente valor en la entrada:

MQTYPE_AS_SET

El valor de la propiedad se devuelve sin convertir su tipo de datos.

ValueLength

Tipo: MQLONG - entrada

La longitud en bytes del área Valor. Especifique cero para las propiedades para las que no necesita que se devuelva el valor. Podrían ser propiedades diseñadas por una aplicación para tener un valor nulo o una cadena vacía. Especifique también cero si se ha especificado la opción [MQIMPO_QUERY_LENGTH](#); en este caso, no se devuelve ningún valor.

Valor

Tipo: MQBYTEX *ValueLength* - salida

Esta es el área que contiene el valor de la propiedad solicitada. El búfer debe alinearse en un límite adecuado para el valor que se devuelve. Si no se hace así, puede producirse un error al acceder posteriormente al valor.

Si *ValueLength* es menor que la longitud del valor de la propiedad, se transfiere a *Value* la mayor parte posible del valor de la propiedad y la llamada falla con el código de finalización MQCC_FAILED y el motivo MQRC_PROPERTY_VALUE_TOO_BIG.

El conjunto de caracteres de los datos en *Valor* viene dado por el campo ReturnedCCSID en el parámetro InqPropOpts. La codificación de los datos en *Value* viene dada por el campo « ReturnedEncoding » (Valor de la clave) del parámetro « InqPropOpts » (Valor de la clave).

En el lenguaje de programación C, el parámetro se declara como un puntero a void; se puede especificar como parámetro la dirección de cualquier tipo de datos.

Si el *ValueLength* parámetro es cero, no se hace referencia a *Value* y su valor pasado por programas escritos en C o ensamblador System/390 puede ser nulo.

DataLength

Tipo: MQLONG - salida

Esta es la longitud en bytes del valor real de la propiedad tal y como se devuelve en el área *Valor*.

Si *DataLength* es menor que la longitud del valor de la propiedad, *DataLength* se sigue rellenando al regresar de la llamada MQINQMP. Esto permite a la aplicación determinar el tamaño del búfer necesario para alojar el valor de la propiedad y, a continuación, volver a emitir la llamada con un búfer del tamaño adecuado.

También se pueden devolver los siguientes valores.

Si el parámetro *Type* se establece en MQTYPE_STRING o MQTYPE_BYTE_STRING:

MQVL_CADENA_VACÍA

La propiedad existe, pero no contiene caracteres ni bytes.

CompCode

Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_WARNING

Aviso (finalización parcial).

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_WARNING:

MQRC_PROP_NAME_NOT_CONVERTED

(2492, X'09BC') Nombre de la propiedad devuelta no convertido.

MQRC_PROP_VALOR_NO_CONVERTIDO

(2466, X'09A2') Valor de la propiedad no convertido.

MQRC_PROP_TYPE_NOT_SUPPORTED

(2467, X'09A3') El tipo de datos de la propiedad no es compatible.

MQRC_RFH_FORMATO_ERROR

(2421, X'0975') No se pudo analizar una carpeta MQRFH2 que contiene propiedades.

Si *CompCode* es MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'089C') Adaptador no disponible.

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'0852') No se puede cargar el módulo de servicio del adaptador.

MQRC_ASID_MISMATCH

(2157, X'086D') Los ASID primarios y domésticos son diferentes.

MQRC_BUFFER_ERROR

(2004, X'07D4') Parámetro de valor no válido.

MQRC_BUFFER_LENGTH_ERROR

(2005, X'07D5') El parámetro de longitud de valor no es válido.

MQRC_CALL_IN_PROGRESS

(2219, X'08AB') Se ha iniciado una llamada al MQI antes de que finalizara la llamada anterior.

MQRC_CONNECTION_BROKEN

(2009, X'07D9') Se ha perdido la conexión con el gestor de colas.

MQRC_DATA_LENGTH_ERROR

(2010, X'07DA') Parámetro de longitud de datos no válido.

MQRC_IMPO_ERROR

(2464, X'09A0') Estructura de opciones de propiedad del mensaje de consulta no válida.

MQRC_HMSG_ERROR

(2460, X'099C') Identificador de mensaje no válido.

MQRC_MSG_HANDLE_EN_USO

(2499, X'09C3') El identificador del mensaje ya está en uso.

MQRC_OPTIONS_ERROR

(2046, X'07F8') Opciones no válidas o incoherentes.

MQRC_PD_ERROR

(2482, X'09B2') Estructura del descriptor de propiedad no válida.

MQRC_PROP_CONV_NO_COMPATIBLE

(2470, X'09A6') No se admite la conversión del tipo de datos real al tipo de datos solicitado.

ERROR_NOMBRE_PROPIEDAD_MQRC

(2442, X'098A') Nombre de propiedad no válido.

MQRC_NOMBRE_DE_LA_PROPIEDAD_DEMASIADO_LARGO

(2465, X'09A1') El nombre de la propiedad es demasiado largo para el búfer de nombres devuelto.

MQRC_PROPIEDAD_NO_DISPONIBLE

(2471, X'09A7') Propiedad no disponible.

MQRC_VALOR_DE_PROPIEDAD_DEMASIADO_GRANDE

(2469, X'09A5') El valor de la propiedad es demasiado grande para el área Valor.

ERROR_FORMATO_NÚMERO_PROPIEDAD_MQRC

(2472, X'09A8') Error de formato numérico encontrado en los datos del valor.

MQRC_TIPO_DE_PROPIEDAD_ERROR

(2473, X'09A9') Tipo de propiedad solicitado no válido.

MQRC_SOURCE_CCSID_ERROR

(2111, X'083F') El identificador del conjunto de caracteres codificados del nombre de la propiedad no es válido.

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'0871') Espacio de almacenamiento insuficiente.

MQRC_UNEXPECTED_ERROR

(2195, X'0893') Se ha producido un error inesperado.

Para obtener información detallada sobre estos códigos, consulte [Mensajes y códigos de motivo](#).

Invocación en C

```
MQINQMP (Hconn, Hmsg, &InqPropOpts, &Name, &PropDesc, &Type,  
ValueLength, Value, &DataLength, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQHCONN Hconn;          /* Connection handle */  
MQHMSG Hmsg;             /* Message handle */  
MQIMPO InqPropOpts;     /* Options that control the action of MQINQMP */  
MQCHARV Name;           /* Property name */  
MQPD PropDesc;          /* Property descriptor */  
MQLONG Type;            /* Property data type */  
MQLONG ValueLength;     /* Length in bytes of the Value area */  
MQBYTE Value[n];        /* Area to contain the property value */  
MQLONG DataLength;      /* Length of the property value */  
MQLONG CompCode;        /* Completion code */  
MQLONG Reason;          /* Reason code qualifying CompCode */
```

Invocación en COBOL

```
CALL 'MQINQMP' USING HCONN, HMSG, INQMSGOPTS, NAME, PROPDSC, TYPE,  
VALUELENGTH, VALUE, DATALENGTH, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
** Connection handle  
01 HCONN          PIC S9(9) BINARY.  
** Message handle  
01 HMSG           PIC S9(18) BINARY.  
** Options that control the action of MQINQMP  
01 INQMSGOPTS.  
   COPY CMQIMPOV.  
** Property name  
01 NAME.  
   COPY CMQCHRVV.  
** Property descriptor  
01 PROPDSC.  
   COPY CMQPDV.  
** Property data type  
01 TYPE           PIC S9(9) BINARY.  
** Length in bytes of the VALUE area  
01 VALUELENGTH PIC S9(9) BINARY.  
** Area to contain the property value  
01 VALUE          PIC X(n).  
** Length of the property value  
01 DATALENGTH PIC S9(9) BINARY.  
** Completion code  
01 COMPCODE       PIC S9(9) BINARY.  
** Reason code qualifying COMPCODE  
01 REASON        PIC S9(9) BINARY.
```

Invocación en PL/I

```
call MQINQMP (Hconn, Hmsg, InqPropOpts, Name, PropDesc, Type,  
ValueLength, Value, DataLength, CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```
dcl Hconn          fixed bin(31); /* Connection handle */  
dcl Hmsg           fixed bin(63); /* Message handle */  
dcl InqPropOpts    like MQIMPO;   /* Options that control the action of MQINQMP */  
dcl Name           like MQCHARV;  /* Property name */  
dcl PropDesc       like MQPD;     /* Property descriptor */  
dcl Type           fixed bin (31); /* Property data type */  
dcl ValueLength    fixed bin (31); /* Length in bytes of the Value area */  
dcl Value          char (n);      /* Area to contain the property value */  
dcl DataLength     fixed bin (31); /* Length of the property value */
```

```

dcl CompCode      fixed bin (31); /* Completion code */
dcl Reason        fixed bin (31); /* Reason code qualifying CompCode */

```

Invocación en ensamblador de alto nivel

```

CALL MQINQMP, (HCONN,HMSG,INQMSGOPTS,NAME,PROPDSC,TYPE,
VALUELENGTH,VALUE,DATALength,COMPCODE,REASON)

```

Declare los parámetros como se indica a continuación:

HCONN	DS	F	Connection handle
HMSG	DS	D	Message handle
INQMSGOPTS	CMQIMPOA	,	Options that control the action of MQINQMP
NAME	CMQCHRVA	,	Property name
PROPDSC	CMQPDA	,	Property descriptor
TYPE	DS	F	Property data type
VALUELENGTH	DS	F	Length in bytes of the VALUE area
VALUE	DS	CL(n)	Area to contain the property value
DATALength	DS	F	Length of the property value
COMPCODE	DS	F	Completion code
REASON	DS	F	Reason code qualifying COMPCODE

MQMHBUF: convierte el identificador de mensaje en un búfer

La llamada MQMHBUF convierte un identificador de mensaje en un búfer y es la inversa de la llamada MQBUFMH.

Sintaxis

MQMHBUF (*Hconn*, *Hmsg*, *MsgHBufOpts*, *Nombre*, *MsgDesc*, *LongitudBúfer*, *Búfer*, *LongitudDatos*, *CódigoComp*, *Motivo*)

Parámetros

Hconn

Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* debe coincidir con el identificador de conexión que se utilizó para crear el identificador de mensaje especificado en el **Hmsg** parámetro.

Si el identificador de mensaje se creó utilizando MQHC_UNASSOCIATED_HCONN, se debe establecer una conexión válida en el subproceso que elimina el identificador de mensaje. Si no se establece una conexión válida, la llamada falla con MQRC_CONNECTION_BROKEN.

Hmsg

Tipo: MQHMSG - entrada

Este es el identificador del mensaje para el que se requiere un búfer. El valor fue devuelto por una llamada MQCRTMH anterior.

MsgHBufOpts

Tipo: MQMHBO - entrada

La estructura MQMHBO permite a las aplicaciones especificar opciones que controlan cómo se producen los búferes a partir de los identificadores de mensajes.

Consulte “MQMHBO - Opciones del controlador de mensajes al búfer” en la [página 451](#) para obtener los detalles.

Nombre

Tipo: MQCHARV - entrada

El nombre de la propiedad o propiedades que se van a incluir en el búfer.

Si no se encuentra ninguna propiedad que coincida con el nombre, la llamada falla con MQRC_PROPERTY_NOT_AVAILABLE.

Puede utilizar un comodín para introducir más de una propiedad en el búfer. Para ello, utilice el carácter comodín «%» al final del nombre de la propiedad. Este comodín coincide con cero o más caracteres, incluido el «.» ?.

En el lenguaje de programación C, se definen las siguientes variables macro para consultar todas las propiedades y todas las propiedades que comienzan por «usr»:

MQPROP_CONSULTAR_TODO

Coloca todas las propiedades del mensaje en el búfer

MQPROP_CONSULTAR_TODOS_USUARIOS

Pon todas las propiedades del mensaje que empiezan con los caracteres «usr.» en el búfer.

Consulte [Nombres de propiedades y Restricciones de nombres de propiedades](#) para obtener más información sobre el uso de nombres de propiedades.

MsgDesc

Tipo: MQMD - entrada/salida

La *MsgDesc* estructura describe el contenido del área de búfer.

En la salida, los campos *Encoding*, *CodedCharSetId* *Format* y se configuran para describir correctamente la codificación, el identificador del juego de caracteres y el formato de los datos en el área del búfer tal y como los ha escrito la llamada.

Los datos de esta estructura están en el juego de caracteres y la codificación de la aplicación.

BufferLength

Tipo: MQLONG - entrada

BufferLength es la longitud del área de búfer, en bytes.

Almacenamiento intermedio

Tipo: MQBYTEExBufferLength - salida

Buffer define el área que contiene las propiedades del mensaje. Debe alinear el búfer en un límite de 4 bytes.

Si *BufferLength* es menor que la longitud necesaria para almacenar las propiedades en *Buffer*, MQMHBUFF falla con MQRC_PROPERTY_VALUE_TOO_BIG.

El contenido del búfer puede cambiar incluso si la llamada falla.

DataLength

Tipo: MQLONG - salida

DataLength es la longitud, en bytes, de las propiedades devueltas en el búfer. Si el valor es cero, ninguna propiedad coincide con el valor dado en *Name* y la llamada falla con el código de motivo MQRC_PROPERTY_NOT_AVAILABLE.

Si *BufferLength* es menor que la longitud necesaria para almacenar las propiedades en el búfer, la llamada MQMHBUFF falla con MQRC_PROPERTY_VALUE_TOO_BIG, pero aún así se introduce un valor en *DataLength*. Esto permite a la aplicación determinar el tamaño del búfer necesario para acomodar las propiedades y, a continuación, volver a emitir la llamada con el *BufferLength*.

CompCode

Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

El código de razón que califica a *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'089C') Adaptador no disponible.

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

MQRC_ASID_MISMATCH

(2157, X'86D') Los ASID primario e inicial varían.

MQRC_MHBO_ERROR

(2501, X'095C') El identificador de mensaje para la estructura de opciones del búfer no es válido.

MQRC_BUFFER_ERROR

(2004, X'07D4') Parámetro de búfer no válido.

MQRC_BUFFER_LENGTH_ERROR

(2005, X'07D5') Parámetro de longitud del búfer no válido.

MQRC_CALL_IN_PROGRESS

(2219, X'08AB') MQI : se ha iniciado una llamada antes de que finalizara la anterior.

MQRC_CONNECTION_BROKEN

(2009, X'07D9') Se ha perdido la conexión con el gestor de colas.

MQRC_DATA_LENGTH_ERROR

(2010, X'07DA') Parámetro de longitud de datos no válido.

MQRC_HMSG_ERROR

(2460, X'099C') Identificador de mensaje no válido.

MQRC_MD_ERROR

(2026, X'07EA') Descriptor de mensaje no válido.

MQRC_MSG_HANDLE_EN_USO

(2499, X'09C3') El identificador del mensaje ya está en uso.

MQRC_OPTIONS_ERROR

(2046, X'07FE') Opciones no válidas o incoherentes.

ERROR_NOMBRE_PROPIEDAD_MQRC

(2442, X'098A') El nombre de la propiedad no es válido.

MQRC_PROPIEDAD_NO_DISPONIBLE

(2471, X'09A7') Propiedad no disponible.

MQRC_VALOR_DE_PROPIEDAD_DEMASIADO_GRANDE

(2469, X'09A5') El valor de BufferLength es demasiado pequeño para contener las propiedades especificadas.

MQRC_UNEXPECTED_ERROR

(2195, X'893') Se ha producido un error inesperado.

Para obtener información detallada sobre estos códigos, consulte [Mensajes y códigos de motivo](#).

Invocación en C

```
MQMHBUF (Hconn, Hmsg, &MsgHBufOpts, &Name, &MsgDesc, BufferLength, Buffer,  
         &DataLength, &CompCode, &Reason);
```


Declare los parámetros como se indica a continuación:

```
MQHCONN Hconn;          /* Connection handle */
MQHMSG Hmsg;             /* Message handle */
MQMHBO MsgHBufOpts;      /* Options that control the action of MQMHBUF */
MQCHARV Name;           /* Property name */
MQMD MsgDesc;           /* Message descriptor */
MQLONG BufferLength;     /* Length in bytes of the Buffer area */
MQBYTE Buffer[n];        /* Area to contain the properties */
MQLONG DataLength;      /* Length of the properties */
MQLONG CompCode;        /* Completion code */
MQLONG Reason;          /* Reason code qualifying CompCode */
```

Notas de uso

MQMHBUF convierte un identificador de mensaje en un búfer.

Puede utilizarlo con una salida API MQGET para acceder a determinadas propiedades, utilizando las API de propiedades de mensajes, y luego pasarlas en un búfer a una aplicación diseñada para utilizar encabezados MQRFH2 en lugar de identificadores de mensajes.

Esta llamada es la inversa de la llamada MQBUFMH, que se puede utilizar para analizar las propiedades de un mensaje desde un búfer a un identificador de mensaje.

Invocación en COBOL

```
CALL 'MQMHBUF' USING HCONN, HMSG, MSGHBUFOPTS, NAME, MSGDESC,
                     BUFFERLENGTH, BUFFER, DATALENGTH, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
** Connection handle
01 HCONN          PIC S9(9) BINARY.
** Message handle
01 HMSG           PIC S9(18) BINARY.
** Options that control the action of MQMHBUF
01 MSGHBUFOPTS.
   COPY CMQMHBV.
** Property name
01 NAME
   COPY CMQCHRVV.
** Message descriptor
01 MSGDESC
   COPY CMQMDV.
** Length in bytes of the Buffer area */
01 BUFFERLENGTH  PIC S9(9) BINARY.
** Area to contain the properties
01 BUFFER        PIC X(n).
** Length of the properties
01 DATALENGTH   PIC S9(9) BINARY.
** Completion code
01 COMPCODE      PIC S9(9) BINARY.
** Reason code qualifying COMPCODE
01 REASON        PIC S9(9) BINARY.
```

Invocación en PL/I

```
call MQMHBUF (Hconn, Hmsg, MsgHBufOpts, Name, MsgDesc, BufferLength, Buffer,
DataLength, CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```
dcl Hconn          fixed bin(31); /* Connection handle */
dcl Hmsg           fixed bin(63); /* Message handle */
dcl MsgHBufOpts    like MQMHBO;  /* Options that control the action of MQMHBUF */
dcl Name           like MQCHARV; /* Property name */
dcl MsgDesc        like MQMD;    /* Message descriptor */
dcl BufferLength    fixed bin(31); /* Length in bytes of the Buffer area */
dcl Buffer          char(n);      /* Area to contain the properties */
dcl DataLength     fixed bin(31); /* Length of the properties */
```

```

dcl CompCode      fixed bin(31); /* Completion code */
dcl Reason        fixed bin(31); /* Reason code qualifying CompCode */

```

Invocación en ensamblador de alto nivel

```

CALL MQMHBUFF, (HCONN,HMSG,MSGHBUFOPTS,NAME,MSGDESC,BUFFERLENGTH,
               BUFFER,DATALength,COMPCode,Reason)

```

Declare los parámetros como se indica a continuación:

HCONN	DS	F	Connection handle
HMSG	DS	D	Message handle
MSGHBUFOPTS	CMQMHB0A	,	Options that control the action of MQMHBUFF
NAME	CMQCHRVA	,	Property name
MSGDESC	CMQMDA	,	Message descriptor
BUFFERLENGTH	DS	F	Length in bytes of the BUFFER area
BUFFER	DS	CL(n)	Area to contain the properties
DATALength	DS	F	Length of the properties
COMPCode	DS	F	Completion code
Reason	DS	F	Reason code qualifying COMPCode

MQOPEN - Abrir objeto

La llamada MQOPEN establece el acceso a un objeto.

Los siguientes tipos de objetos son válidos:

- Cola (incluidas las listas de distribución)
- Lista de nombres
- Definición de proceso
- Gestor de colas
- Tema

Sintaxis

MQOPEN (*Hconn*, *ObjDesc*, *Opciones*, *Hobj*, *CódigoComp*, *Motivo*)

Parámetros

Hconn

Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de Hconn fue devuelto por una llamada anterior a MQCONN o MQCONNEX.

 En z/OS para CICS aplicaciones, se puede omitir la llamada MQCONN y especificar el siguiente valor para *Hconn*:

MQHC_DEF_HCONN

Manejador de conexión predeterminado.

ObjDesc

Tipo: MQOD - entrada/salida

Esta es una estructura que identifica el objeto que se va a abrir; consulte [“MQOD - Descriptor de objeto”](#) en la página 454 para obtener más detalles.

Si el ObjectName campo del **ObjDesc** parámetro es el nombre de una cola de modelos, una cola local dinámica se crea con los atributos de la cola modelo; esto ocurre independientemente de las opciones que especifique en el **Options** parámetro. Las operaciones posteriores que utilizan el *Hobj* devuelto por la llamada MQOPEN se realizan en la nueva cola dinámica, y no en la cola modelo. Esto es válido incluso para las llamadas MQINQ y MQSET. El nombre de la cola de modelos en el **ObjDesc** parámetro se sustituye por el nombre de la cola dinámica creada. El tipo de cola dinámica

viene determinado por el valor del **DefinitionType** atributo de la cola modelo (véase “Atributos para colas” en la página 841). Para obtener información sobre las opciones de cierre aplicables a las colas dinámicas, consulte la descripción de la llamada MQCLOSE.

Opciones

Tipo: MQLONG - entrada

Debe especificar al menos una de las siguientes opciones:

- MQOO_BROWSE
- MQOO_INPUT_* (solo uno de estos)
- MQOO_INQUIRE
- MQOO_OUTPUT
- MQOO_SET
- MQOO_BIND_* (solo uno de estos)

Consulte la siguiente tabla para obtener más información sobre estas opciones; se pueden especificar otras opciones según sea necesario. Para especificar más de una opción, suma los valores (sin añadir la misma constante más de una vez) o combínalos mediante la operación OR bit a bit (si el lenguaje de programación admite operaciones bit a bit). Las combinaciones que no son válidas se indican; todas las demás combinaciones son válidas. Solo se permiten las opciones que son aplicables ObjDesc al tipo de objeto especificado por.

Tabla 589. Opciones MQOPEN válidas para colas y temas						
Opción	Alias ¹	Local y modelo	Remoto	Cúmulo no local	Lista de distribución	Tema
<u>MQOO_INPUT_AS_Q_DEF</u>	Sí	Sí	No	No	No	No
<u>MQOO_INPUT_SHARED</u>	Sí	Sí	No	No	No	No
<u>MQOO_INPUT_EXCLUSIVE</u>	Sí	Sí	No	No	No	No
<u>MQOO_OUTPUT</u>	Sí	Sí	Sí	Sí	Sí	Sí
<u>MQOO_BROWSE</u>	Sí	Sí	No	No	No	No
<u>MQOO_CO_OP</u>	Sí	Sí	No	No	No	No
<u>MQOO_INQUIRE</u>	Sí	Sí	<u>2</u>	Sí	No	No
<u>MQOO_SET</u>	Sí	Sí	<u>2</u>	No	No	No
<u>MQOO_BIND_ON_OPEN</u> ³	Sí	Sí	Sí	Sí	Sí	No
<u>MQOO_BIND_NOT_FIXED</u> ³	Sí	Sí	Sí	Sí	Sí	No
<u>MQOO_BIND_ON_GROUP</u> ³	Sí	Sí	Sí	Sí	Sí	No
<u>MQOO_BIND_AS_Q_DEF</u> ³	Sí	Sí	Sí	Sí	Sí	No
<u>MQOO_SAVE_ALL_CONTEXT</u>	Sí	Sí	No	No	No	No
<u>MQOO_PASS_IDENTIDAD_CONTEXTO</u>	Sí	Sí	Sí	Sí	Sí	<u>4</u>
<u>MQOO_PASS_ALL_CONTEXT</u>	Sí	Sí	Sí	Sí	Sí	Sí
<u>MQOO_SET_IDENTITY_CONTEXT</u>	Sí	Sí	Sí	Sí	Sí	<u>4</u>
<u>MQOO_SET_ALL_CONTEXT</u>	Sí	Sí	Sí	Sí	Sí	Sí
<u>MQOO_NO_LEER_POR_ADELANTE</u>	Sí	Sí	No	No	No	No
<u>MQOO_LEER_POR_ADELANTE</u>	Sí	Sí	No	No	No	No
<u>MQOO_LEER_POR_ADELANTE_COMO_Q_DEF</u>	Sí	Sí	No	No	No	No
<u>MQOO_ALTERNATE_USER_AUTHORITY</u>	Sí	Sí	Sí	Sí	Sí	Sí
<u>MQOO_FAIL_IF QUIESCING</u>	Sí	Sí	Sí	Sí	Sí	Sí
<u>MQOO_RESOLVER_LOCAL_Q</u>	Sí	Sí	Sí	Sí	No	No

Tabla 589. Opciones MQOPEN válidas para colas y temas (continuación)

Opción	Alias ¹	Local y modelo	Remoto	Cúmulo no local	Lista de distribución	Tema
MQOO_RESOLVER_TEMA_LOCAL	No	No	No	No	No	Sí
MQOO_NO_MULTICAST	No	No	No	No	No	Sí

Notas:

1. La validez de las opciones para los alias depende de la validez de la opción para la cola a la que resuelve el alias.
2. Esta opción solo es válida para la definición local de una cola remota.
3. Esta opción se puede especificar para cualquier tipo de cola, pero se ignora si la cola no es una cola de clúster. Sin embargo, el **DefBind** atributo de cola anula la cola base incluso cuando la cola alias no se encuentra en un clúster.
4. Estos atributos se pueden utilizar con un tema, pero solo afectan al contexto establecido para el mensaje retenido, no a los campos de contexto enviados a ningún suscriptor.

Opciones de acceso : Las siguientes opciones controlan el tipo de operaciones que se pueden realizar en el objeto:

MQOO_INPUT_AS_Q_DEF

Abrir la cola para obtener mensajes utilizando el valor predeterminado definido en la cola.

La cola se abre para su uso con llamadas MQGET posteriores. El tipo de acceso es compartido o exclusivo, dependiendo del valor del atributo **DefInputOpenOption** queue; consulte [“Atributos para colas”](#) en la página 841 para obtener más detalles.

Esta opción solo es válida para colas locales, alias y modelos; no es válida para colas remotas, listas de distribución y objetos que no sean colas.

MQOO_INPUT_SHARED

Abrir cola para obtener mensajes con acceso compartido.

La cola se abre para su uso con llamadas MQGET posteriores. La llamada puede tener éxito si la cola está abierta actualmente por esta u otra aplicación con MQOO_INPUT_SHARED, pero falla con el código de motivo MQRC_OBJECT_IN_USE si la cola está abierta actualmente con MQOO_INPUT_EXCLUSIVE.

Esta opción solo es válida para colas locales, alias y modelos; no es válida para colas remotas, listas de distribución y objetos que no sean colas.

MQOO_INPUT_EXCLUSIVE

Abrir cola para obtener mensajes con acceso exclusivo.

La cola se abre para su uso con llamadas MQGET posteriores. La llamada falla con el código de motivo MQRC_OBJECT_IN_USE si la cola está actualmente abierta por esta u otra aplicación para la entrada de cualquier tipo (MQOO_INPUT_SHARED o MQOO_INPUT_EXCLUSIVE).

Esta opción solo es válida para colas locales, alias y modelos; no es válida para colas remotas, listas de distribución y objetos que no sean colas.

MQOO_OUTPUT

Abre la cola para insertar mensajes, o un tema o cadena de temas para publicar mensajes.

La cola o el tema se abre para su uso con llamadas MQPUT posteriores.

Una llamada MQOPEN con esta opción puede tener éxito incluso si el **InhibitPut** atributo de cola está establecido en MQQA_PUT_INHIBITED (aunque las llamadas MQPUT posteriores fallarán mientras el atributo esté establecido en este valor).

Esta opción es válida para todos los tipos de colas, incluidas las listas de distribución y los temas.

Las siguientes notas se aplican a estas opciones:

- Solo se puede especificar una de estas opciones.
- Una llamada MQOPEN con una de estas opciones puede tener éxito incluso si el **InhibitGet** atributo de cola está establecido en MQQA_GET_INHIBITED (aunque las llamadas MQGET posteriores fallarán mientras el atributo esté establecido en este valor).
- Si la cola se define como no compartible (es decir, el **Shareability** atributo de cola tiene el valor MQQA_NOT_SHAREABLE), los intentos de abrir la cola para acceso compartido se tratan como intentos de abrir la cola con acceso exclusivo.
- Si se abre una cola alias con una de estas opciones, la comprobación de uso exclusivo (o de si otra aplicación tiene uso exclusivo) se realiza con respecto a la cola base a la que resuelve el alias.
- Estas opciones no son válidas si **ObjectQMgrName** es el nombre de un alias del gestor de colas; esto es así incluso si el valor del **RemoteQMgrName** atributo en la definición local de una cola remota utilizada para el alias del gestor de colas es el nombre del gestor de colas local.

MQOO_BROWSE

Abre la cola para ver los mensajes.

La cola se abre para su uso con llamadas MQGET posteriores con una de las siguientes opciones:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_NEXT
- MQGMO_BROWSE_MSG_UNDER_CURSOR

Esto está permitido incluso si la cola está actualmente abierta para MQOO_INPUT_EXCLUSIVE. Una llamada MQOPEN con la opción MQOO_BROWSE establece un cursor de exploración y lo coloca lógicamente antes del primer mensaje de la cola; consulte [MQGMO - Campo de opciones](#) para obtener más información.

Esta opción solo es válida para colas locales, alias y modelos; no es válida para colas remotas, listas de distribución y objetos que no sean colas. Tampoco es válido si **ObjectQMgrName** es el nombre de un alias del gestor de colas; esto es así incluso si el valor del **RemoteQMgrName** atributo en la definición local de una cola remota utilizada para el alias del gestor de colas es el nombre del gestor de colas local.

MQOO_CO_OP

Abrir como miembro colaborador del conjunto de manijas.

Esta opción solo es válida con la opción MQOO_BROWSE. Si se especifica sin MQOO_BROWSE, MQOPEN devuelve MQRC_OPTIONS_ERROR.

El identificador devuelto se considera miembro de un conjunto cooperativo de identificadores para llamadas MQGET posteriores con una de las siguientes opciones:

- MQGMO_MARK_BROWSE_CO_OP
- MQGMO_UNMARKED_BROWSE_MSG
- MQGMO_UNMARK_BROWSE_CO_OP

Esta opción solo es válida para colas locales, alias y modelos; no es válida para colas remotas, listas de distribución y objetos que no sean colas.

MQOO_INQUIRE

Abrir objeto para consultar atributos.

La cola, la lista de nombres, la definición del proceso o el gestor de colas se abren para su uso con llamadas MQINQ posteriores.

Esta opción es válida para todos los tipos de objetos, excepto las listas de distribución. No es válido si **ObjectQMgrName** es el nombre de un alias del gestor de colas; esto es así incluso si el valor del **RemoteQMgrName** atributo en la definición local de una cola remota utilizada para el alias del gestor de colas es el nombre del gestor de colas local.

MQOO_SET

Abrir cola para establecer atributos.

La cola se abre para su uso con llamadas MQSET posteriores.

Esta opción es válida para todos los tipos de colas, excepto las listas de distribución. No es válido si `ObjectQMgrName` es el nombre de una definición local de una cola remota; esto es cierto incluso si el valor del **RemoteQMgrName** atributo en la definición local de una cola remota utilizada para el aliasing del gestor de colas es el nombre del gestor de colas local.

Opciones de enlace : Las siguientes opciones se aplican cuando el objeto que se abre es una cola de clúster; estas opciones controlan el enlace del identificador de cola a una instancia de la cola de clúster:

MQOO_BIND_ON_OPEN

El administrador de colas local vincula el identificador de la cola a una instancia de la cola de destino cuando se abre la cola. Como resultado, todos los mensajes enviados utilizando este identificador se envían a la misma instancia de la cola de destino y por la misma ruta.

Esta opción solo es válida para colas y solo afecta a colas de clúster. Si se especifica en una cola que no sea de clúster, la opción se pasará por alto.

MQOO_BIND_NO_SE_HA_CORREGIDO

Esto impide que el administrador de colas local vincule el identificador de cola a una instancia de la cola de destino. Como resultado, las sucesivas llamadas a MQPUT que utilizan este identificador envían los mensajes a diferentes instancias de la cola de destino, o a la misma instancia pero por rutas diferentes. También permite que la instancia seleccionada sea modificada posteriormente por el gestor de colas local, por un gestor de colas remoto o por un agente de canal de mensajes (MCA), según las condiciones de la red.

Nota: Las aplicaciones cliente y servidor que necesitan intercambiar una serie de mensajes para completar una transacción no deben utilizar MQOO_BIND_NOT_FIXED (o MQOO_BIND_AS_Q_DEF cuando `DefBind` tiene el valor MQBND_BIND_NOT_FIXED), ya que los mensajes sucesivos de la serie podrían enviarse a diferentes instancias de la aplicación servidor.

Si se especifica MQOO_BROWSE o una de las opciones MQOO_INPUT_* para una cola de clúster, el gestor de colas se ve obligado a seleccionar la instancia local de la cola de clúster. Como resultado, la vinculación del identificador de cola es fija, incluso si se especifica MQOO_BIND_NOT_FIXED.

Si se especifica MQOO_INQUIRE con MQOO_BIND_NOT_FIXED, las llamadas MQINQ sucesivas que utilicen ese identificador podrían consultar diferentes instancias de la cola del clúster, aunque normalmente todas las instancias tienen los mismos valores de atributo.

MQOO_BIND_NOT_FIXED solo es válido para colas y solo afecta a las colas de clúster. Si se especifica en una cola que no sea de clúster, la opción se pasará por alto.

MQOO_BIND_ON_GROUP

Permite a una aplicación solicitar que un grupo de mensajes se asigne en su totalidad a la misma instancia de destino.

Esta opción solo es válida para colas y solo afecta a colas de clúster. Si se especifica en una cola que no sea de clúster, la opción se pasará por alto.

MQOO_BIND_AS_Q_DEF

El gestor de colas local vincula el identificador de cola de la forma definida por el **DefBind** atributo de cola. El valor de este atributo es MQBND_BIND_ON_OPEN, MQBND_BIND_NOT_FIXED o MQBND_BIND_ON_GROUP.

MQOO_BIND_AS_Q_DEF es el valor predeterminado cuando no se especifica MQOO_BIND_ON_OPEN, MQOO_BIND_NOT_FIXED o MQOO_BIND_ON_GROUP.

MQOO_BIND_AS_Q_DEF ayuda a la documentación del programa. No está previsto que esta opción se utilice con ninguna de las otras dos opciones de enlace, pero dado que su valor es cero, dicho uso no puede detectarse.

Opciones de contexto : Las siguientes opciones controlan el procesamiento del contexto del mensaje:

MQOO_SAVE_ALL_CONTEXT

La información contextual está asociada a este identificador de cola. Esta información se establece a partir del contexto de cualquier mensaje recuperado utilizando este identificador. Para obtener más información sobre el contexto de los mensajes, consulte [Contexto de los mensajes y Control de la información de contexto](#).

Esta información contextual puede pasarse a un mensaje que luego se coloca en una cola mediante las llamadas MQPUT o MQPUT1. Consulte las opciones MQPMO_PASS_IDENTITY_CONTEXT y MQPMO_PASS_ALL_CONTEXT descritas en “MQPMO - Opciones de mensaje de envío” en la página 477.

Hasta que no se haya recuperado correctamente un mensaje, no se puede pasar el contexto a un mensaje que se está colocando en una cola.

Un mensaje recuperado utilizando una de las opciones de exploración MQGMO_BROWSE_* no tiene guardada su información de contexto (aunque los campos de contexto del **MsgDesc** parámetro se establecen después de una exploración).

Esta opción solo es válida para colas locales, alias y modelos; no es válida para colas remotas, listas de distribución y objetos que no sean colas. Se debe especificar una de las opciones MQOO_INPUT_*.

MQOO_PASS_IDENTIDAD_CONTEXTO

Esto permite especificar la opción MQPMO_PASS_IDENTITY_CONTEXT en el **PutMsgOpts** parámetro cuando se coloca un mensaje en una cola; esto proporciona al mensaje la información del contexto de identidad de una cola de entrada que se abrió con la opción MQOO_SAVE_ALL_CONTEXT. Para obtener más información sobre el contexto de los mensajes, consulte [Contexto de los mensajes y Control de la información de contexto](#).

Se debe especificar la opción MQOO_OUTPUT.

Esta opción es válida para todos los tipos de colas, incluidas las listas de distribución.

MQOO_PASS_ALL_CONTEXT

Esto permite especificar la opción MQPMO_PASS_ALL_CONTEXT en el **PutMsgOpts** parámetro cuando se coloca un mensaje en una cola; esto proporciona al mensaje la información de identidad y contexto de origen de una cola de entrada que se abrió con la opción MQOO_SAVE_ALL_CONTEXT. Para obtener más información sobre el contexto de los mensajes, consulte [Contexto de los mensajes y Control de la información de contexto](#).

Esta opción implica MQOO_PASS_IDENTITY_CONTEXT, por lo que no es necesario especificarla. Se debe especificar la opción MQOO_OUTPUT.

Esta opción es válida para todos los tipos de colas, incluidas las listas de distribución.

MQOO_SET_IDENTITY_CONTEXT

Esto permite especificar la opción MQPMO_SET_IDENTITY_CONTEXT en el **PutMsgOpts** parámetro cuando se coloca un mensaje en una cola; esto proporciona al mensaje la información del contexto de identidad contenida en el **MsgDesc** parámetro especificado en la llamada MQPUT o MQPUT1. Para obtener más información sobre el contexto de los mensajes, consulte [Contexto de los mensajes y Control de la información de contexto](#).

Esta opción implica MQOO_PASS_IDENTITY_CONTEXT, por lo que no es necesario especificarla. Se debe especificar la opción MQOO_OUTPUT.

Esta opción es válida para todos los tipos de colas, incluidas las listas de distribución.

MQOO_SET_ALL_CONTEXT

Esto permite especificar la opción MQPMO_SET_ALL_CONTEXT en el **PutMsgOpts** parámetro cuando se coloca un mensaje en una cola; esto proporciona al mensaje la información de identidad y contexto de origen contenida en el **MsgDesc** parámetro especificado en la llamada MQPUT o MQPUT1. Para obtener más información sobre el contexto de los mensajes, consulte [Contexto de los mensajes y Control de la información de contexto](#).

Esta opción implica las siguientes opciones, que por lo tanto no es necesario especificar:

- MQOO_PASS_IDENTIDAD_CONTEXTO
- MQOO_PASS_ALL_CONTEXT
- MQOO_SET_IDENTITY_CONTEXT

Se debe especificar la opción MQOO_OUTPUT.

Esta opción es válida para todos los tipos de colas, incluidas las listas de distribución.

Opciones para leer más adelante :

Cuando se llama a MQOPEN con MQOO_READ_AHEAD, el cliente de IBM MQ sólo permite la lectura anticipada si se cumplen ciertas condiciones. Estas condiciones incluyen:

- La aplicación cliente debe compilarse y enlazarse a las bibliotecas de cliente MQI de IBM MQ con hebras.
- El canal de cliente debe utilizar el protocolo TCP/IP
- El canal debe tener un valor SharingConversations (SHARECNV) distinto de cero tanto en las definiciones de canal de cliente y servidor.

Las siguientes opciones controlan si los mensajes no persistentes se envían al cliente antes de que una aplicación los solicite. Las siguientes notas se aplican a las opciones de lectura anticipada:

- Solo se puede especificar una de estas opciones.
- Estas opciones solo son válidas para colas locales, alias y de modelo. No son válidos para colas remotas, listas de distribución, temas o gestores de colas.
- Estas opciones solo son aplicables cuando también se especifican MQOO_BROWSE, MQOO_INPUT_SHARED y MQOO_INPUT_EXCLUSIVE, aunque no es un error especificar estas opciones con MQOO_INQUIRE o MQOO_SET.
- Si la aplicación no se está ejecutando como cliente IBM MQ , estas opciones se ignoran.

MQOO_NO_LEER_POR_ADELANTE

Los mensajes no persistentes no se envían al cliente antes de que una aplicación los solicite.

MQOO_LEER_POR_ADELANTE

Los mensajes no persistentes se envían al cliente antes de que una aplicación los solicite.

MQOO_LEER_POR_ADELANTE_COMO_Q_DEF

El comportamiento de lectura anticipada viene determinado por el atributo de lectura anticipada predeterminado de la cola que se abre. Éste es el valor predeterminado.

Otras opciones : Las siguientes opciones controlan la comprobación de autorizaciones, lo que ocurre cuando el gestor de colas está inactivo, si se debe resolver el nombre de la cola local y la multidifusión:

MQOO_ALTERNATE_USER_AUTHORITY

El *AlternateUserId* campo del **ObjDesc** parámetro contiene un identificador de usuario que se utilizará para validar esta llamada MQOPEN. La llamada solo puede tener éxito si está *AlternateUserId* autorizado para abrir el objeto con las opciones de acceso especificadas, independientemente de si el identificador de usuario bajo el que se ejecuta la aplicación está autorizado para hacerlo. Sin embargo, esto no se aplica a las opciones de contexto especificadas, que siempre se comparan con el identificador de usuario con el que se ejecuta la aplicación.

Esta opción es válida para todos los tipos de objetos.

MQOO_FAIL_IF QUIESCING

La llamada MQOPEN falla si el gestor de colas se encuentra en estado de inactividad.

 En z/OS, para una aplicación CICS IMS o, esta opción también fuerza el fallo de la llamada MQOPEN si la conexión se encuentra en estado de inactividad.

Esta opción es válida para todos los tipos de objetos.

Para obtener información sobre los canales de clientes, consulte [IBM MQ MQI clients](#).

MQOO_RESOLVE_LOCAL_Q

Rellene el campo « ResolvedQName » (Nombre de la cola local) de la estructura MQOD con el nombre de la cola local que se ha abierto. Del mismo modo, el campo « ResolvedQMgrName » se rellena con el nombre del gestor de colas local que aloja la cola local. Si la estructura MQOD es inferior a la versión 3, MQOO_RESOLVE_LOCAL_Q se ignora sin que se devuelva ningún error.

La cola local siempre se devuelve cuando se abre una cola local, alias o modelo, pero este no es el caso cuando, por ejemplo, se abre una cola remota o una cola de clúster no local sin la opción MQOO_RESOLVE_LOCAL_Q; los campos « ResolvedQName » y « ResolvedQMgrName » se rellenan con « RemoteQName » y « RemoteQMgrName » que se encuentran en la definición de la cola remota, o de forma similar con la cola de clúster remota elegida.

Si se especifica MQOO_RESOLVE_LOCAL_Q al abrir, por ejemplo, una cola remota, ResolvedQName es la cola de transmisión en la que se colocan los mensajes. El campo « ResolvedQMgrName » se rellena con el nombre del gestor de colas local que aloja la cola de transmisión.

Si está autorizado para navegar, introducir o generar resultados en una cola, dispone de la autoridad necesaria para especificar este indicador en la llamada MQOPEN. No se necesita ninguna autorización especial.

Esta opción solo es válida para colas y gestores de colas.

MQOO_RESOLVE_TEMA_LOCAL

Rellene el campo « ResolvedQName » (Tema de la sesión) de la estructura MQOD con el nombre del tema administrativo abierto.

MQOO_NO_MULTICAST

Los mensajes de publicación no se envían mediante multidifusión.

Esta opción solo es válida con la opción MQOO_OUTPUT. Si se especifica sin MQOO_OUTPUT, MQOPEN devuelve MQRC_OPTIONS_ERROR.

Esta opción solo es válida para un tema.

Hobj

Tipo: MQHOBJ - salida

Este identificador representa el acceso que se ha establecido al objeto. Debe especificarse en las llamadas IBM MQ posteriores que operan sobre el objeto. Deja de ser válido cuando se emite la llamada MQCLOSE o cuando finaliza la unidad de procesamiento que define el ámbito del identificador.

El ámbito del identificador de objeto devuelto es el mismo que el ámbito del identificador de conexión especificado en la llamada. Consulte [MQCONN - Parámetro Hconn](#) para obtener información sobre el ámbito de los identificadores.

CompCode

Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_WARNING

Aviso (finalización parcial).

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

El código de razón que califica a *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_WARNING:

MQRC_MÚLTIPLES_RAZONES

(2136, X'858') Se han devuelto varios códigos de motivo.

Si *CompCode* es MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'89C') El adaptador no está disponible.

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

MQRC_ALIAS_BASE_Q_TIPO_ERROR

(2001, X'7D1') La cola base alias no es un tipo válido.

MQRC_API_EXIT_ERROR

(2374, X'946') La salida de la API ha fallado.

MQRC_API_EXIT_LOAD_ERROR

(2183, X'887') No se ha podido cargar la salida de la API.

MQRC_ASID_MISMATCH

(2157, X'86D') Los ASID primario e inicial varían.

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') La llamada MQI ha entrado antes de completar la llamada anterior.

MQRC_CF_NO_DISPONIBLE

(2345, X'929') Instalación de acoplamiento no disponible.

MQRC_CF_STRUC_AUTH_FAILED

(2348, X'92C') Error en la comprobación de la autorización de la estructura de la instalación de acoplamiento.

MQRC_CF_STRUC_ERROR

(2349, X'92D') Estructura de la instalación de acoplamiento no válida.

MQRC_CF_STRUC_FAILED

(2373, X'945') La estructura de recurso de acoplamiento ha fallado.

MQRC_CF_STRUC_IN_USE

(2346, X'92A') La estructura de recurso de acoplamiento se está utilizando.

MQRC_CF_STRUC_LIST_HDR_IN_USE

(2347, X'92B') La cabecera de lista de la estructura de recurso de acoplamiento se está utilizando.

MQRC_CICS_WAIT_FAILED

(2140, X'85C') Solicitud de espera rechazada por CICS.

ERROR DE SALIDA DEL CLÚSTER MQRC

(2266, X'8DA') Error al salir de la carga de trabajo del clúster.

MQRC_CLUSTER_PUT_INHIBITED

(2268, X'8DC') Se inhibieron las llamadas de puesta para todas las colas del clúster.

MQRC_CLUSTER_RESOLUTION_ERROR

(2189, X'88D') La resolución de nombres de clúster ha fallado.

ERROR_RECURSO_CLÚSTER_MQRC

(2269, X'8DD') Error de recursos del clúster.

MQRC_CONNECTION_BROKEN

(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

MQRC_CONNECTION_NOT_AUTHORIZED

(2217, X'8A9') No tiene autorización para la conexión.

MQRC_CONNECTION QUIESCING

(2202, X'89A') Conexión en fase de inmovilización.

MQRC_CONNECTION_STOPPING

(2203, X'89B') La conexión está concluyendo.

MQRC_DB2_NOT_AVAILABLE

(2342, X'926') Db2 subsistema no disponible.

MQRC_DEF_XMIT_Q_TYPE_ERROR

(2198, X'896') Cola de transmisión predeterminada no local.

MQRC_DEF_XMIT_Q_USO_ERROR

(2199, X'897') Error de uso de la cola de transmisión predeterminada.

MQRC_DYNAMIC_Q_NAME_ERROR

(2011, X'7DB') Nombre de la cola dinámica no válido.

MQRC_HANDLE_NO_DISPONIBLE

(2017, X'7E1') No hay más asas disponibles.

MQRC_HCONN_ERROR

(2018, X'7E2') El manejador de conexión no es válido.

MQRC_HOBJ_ERROR

(2019, X'7E3') El manejador de objeto no es válido.

MQRC_MÚLTIPLES_RAZONES

(2136, X'858') Se han devuelto varios códigos de motivo.

MQRC_NOMBRE_EN_USO

(2201, X'899') Nombre en uso.

MQRC_NOMBRE_NO_VÁLIDO_PARA_TIPO

(2194, X'892') Nombre de objeto no válido para el tipo de objeto.

MQRC_NOT_AUTHORIZED

(2035, X'7F3') Acceso no autorizado.

MQRC_OBJETO_YA_EXISTE

(2100, X'834') El objeto existe.

MQRC_OBJECT_DAMAGED

(2101, X'835') El objeto se ha dañado.

MQRC_OBJECT_IN_USE

(2042, X'7FA') Objeto ya abierto con opciones conflictivas.

MQRC_OBJETO_NIVEL_INCOMPATIBLE

(2360, X'938') Nivel de objeto no compatible.

MQRC_OBJETO_NOMBRE_ERROR

(2152, X'868') Nombre del objeto no válido.

MQRC_OBJETO_NO_ÚNICO

(2343, X'927') Objeto no único.

MQRC_OBJETO_Q_MGR_NOMBRE_ERROR

(2153, X'869') Nombre del gestor de colas de objetos no válido.

ERROR_DE_REGISTROS_DE_OBJETOS_MQRC

(2155, X'86B') Registros de objetos no válidos.

MQRC_OBJECT_STRING_ERROR

(2441, X'0989') Campo de cadena de objeto no válido

MQRC_TIPO_OBJETO_ERROR

(2043, X'7FB') Tipo de objeto no válido.

MQRC_OD_ERROR

(2044, X'7FC') Estructura del descriptor de objeto no válida.

MQRC_OPCIÓN_NO_VÁLIDA_PARA_EL_TIPO

(2045, X'7FD') Opción no válida para el tipo de objeto.

MQRC_OPTIONS_ERROR

(2046, X'7FE') Las opciones no son válidas o no son coherentes.

MQRC_PAGESET_ERROR

(2193, X'891') Error al acceder al conjunto de datos del conjunto de páginas.

MQRC_PÁGINAS_COMPLETAS

(2192, X'890') El soporte de almacenamiento externo está lleno.

MQRC_Q_DELETED

(2052, X'804') La cola se ha suprimido.

MQRC_Q_MGR_NAME_ERROR

(2058, X'80A') El nombre del gestor de colas no es válido o no se conoce.

MQRC_Q_MGR_NOT_AVAILABLE

(2059, X'80B') El gestor de colas no está disponible para efectuar la conexión.

MQRC_Q_MGR QUIESCING

(2161, X'871') El gestor de colas se está desactivando temporalmente.

MQRC_Q_MGR_STOPPING

(2162, X'872') El gestor de colas está concluyendo.

MQRC_Q_TIPO_ERROR

(2057, X'809') Tipo de cola no válido.

MQRC_RECS_PRESENT_ERROR

(2154, X'86A') Número de registros presentes no válido.

MQRC_REMOTO_Q_NOMBRE_ERROR

(2184, X'888') Nombre de cola remota no válido.

MQRC_RESOURCE_PROBLEM

(2102, X'836') No hay disponibles suficientes recursos del sistema.

ERROR_EN_LOS_REGISTROS_DE_RESPUESTA_MQRC

(2156, X'86C') Registros de respuesta no válidos.

MQRC_ERROR_DE_SEGURIDAD

(2063, X'80F') Se ha producido un error de seguridad.

MQRC_SELECTOR_SYNTAX_ERROR

2459 (X'099B') Se ha emitido una llamada MQOPEN, MQPUT1 o MQSUB, pero se ha especificado una cadena de selección que contiene un error de sintaxis.

MQRC_DETENIDO_POR_SALIDA_DEL_CLÚSTER

(2188, X'88C') Llamada rechazada por salida de carga de trabajo del clúster.

MQRC_STORAGE_MEDIUM_FULL

(2192, X'890') El soporte de almacenamiento externo está lleno.

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_SUPPRESSED_BY_EXIT

(2109, X'83D') El programa de salida ha suprimido la llamada.

MQRC_UNEXPECTED_ERROR

(2195, X'893') Se ha producido un error inesperado.

MQRC_ALIAS_DESCONOCIDO_BASE_Q

(2082, X'822') Cola base de alias desconocida.

MQRC_DESCONOCIDO_DEF_XMIT_Q

(2197, X'895') Cola de transmisión predeterminada desconocida.

MQRC_UNKNOWN_OBJECT_NAME

(2085, X'825') Nombre del objeto desconocido.

MQRC_OBJETO_DESCONOCIDO_Q_MGR

(2086, X'826') Administrador de colas de objetos desconocido.

MQRC_DESCONOCIDO_REMOTO_Q_MGR

(2087, X'827') Administrador de colas remotas desconocido.

MQRC_DESCONOCIDO_XMIT_Q

(2196, X'894') Cola de transmisión desconocida.

MQRC_NIVEL_CF_INCORRECTO

(2366, X'93E') La estructura de la instalación de acoplamiento tiene un nivel incorrecto.

MQRC_XMIT_Q_TYPE_ERROR

(2091, X'82B') Cola de transmisión no local.

MQRC_XMIT_Q_ERROR_DE_USO

(2092, X'82C') Cola de transmisión con uso incorrecto.

Para obtener información detallada sobre estos códigos, consulte [Mensajes y códigos de motivo](#).

Notas generales de uso

1. El objeto abierto es uno de los siguientes:

- Una cola para:
 - Obtener o explorar mensajes (utilizando la llamada MQGET)
 - Enviar mensajes (utilizando la llamada MQPUT)
 - Consultar los atributos de la cola (utilizando la llamada MQINQ)
 - Establecer los atributos de la cola (utilizando la llamada MQSET)

Si la cola nombrada es una cola modelo, se crea una cola local dinámica. Consulte el **ObjDesc** parámetro descrito en “MQOPEN - Abrir objeto” en la página 734.

Una lista de distribución es un tipo especial de objeto de cola que contiene una lista de colas. Se puede abrir para introducir mensajes, pero no para obtenerlos o explorarlos, ni para consultar o establecer atributos. Véase la nota de uso 8 para más detalles.

Una cola que tiene QSGDISP (GROUP) es un tipo especial de definición de cola que no se puede utilizar con las llamadas MQOPEN o MQPUT1.

- Una lista de nombres para consultar los nombres de las colas de la lista (utilizando la llamada MQINQ).
 - Una definición de proceso para consultar los atributos del proceso (utilizando la llamada MQINQ).
 - El gestor de colas para consultar los atributos del gestor de colas local (utilizando la llamada MQINQ).
 - Un tema para publicar un mensaje (utilizando la llamada MQPUT)
2. Una aplicación puede abrir el mismo objeto más de una vez. Se devuelve un identificador de objeto diferente para cada apertura. Cada identificador que se devuelve se puede utilizar para las funciones para las que se realizó la apertura correspondiente.
3. Si el objeto que se abre es una cola distinta de una cola de clúster, toda la resolución de nombres dentro del gestor de colas local se lleva a cabo en el momento de la llamada MQOPEN. Puede incluir:
- Resolución del nombre de una definición local de una cola remota al nombre del gestor de colas remotas, y el nombre por el que se conoce la cola en el gestor de colas remotas
 - Resolución del nombre del gestor de colas remoto al nombre de una cola de transmisión local
 -  Solo z/OS en, resolución del nombre del gestor de colas remoto al nombre de la cola de transmisión compartida utilizada por el agente IGQ (solo se aplica si los gestores de colas locales y remotos pertenecen al mismo grupo de uso compartido de colas)
 - Resolución de alias al nombre de una cola base o un objeto de tema.

Sin embargo, tenga en cuenta que las llamadas MQINQ o MQSET posteriores para el identificador se refieren únicamente al nombre que se ha abierto, y no al objeto resultante tras la resolución del nombre. Por ejemplo, si el objeto abierto es un alias, los atributos devueltos por la llamada MQINQ son los atributos del alias, no los atributos de la cola base o del objeto de tema al que resuelve el alias.

Si el objeto que se abre es una cola de clúster, la resolución de nombres puede producirse en el momento de la llamada MQOPEN o aplazarse hasta más tarde. El punto en el que se produce la resolución se controla mediante las opciones MQOO_BIND_* especificadas en la llamada MQOPEN:

- MQOO_BIND_ON_OPEN
- MQOO_BIND_NO_SE_HA_CORREGIDO
- MQOO_BIND_AS_Q_DEF
- MQOO_BIND_ON_GROUP

Consulte [Resolución de nombres](#) para obtener más información sobre la resolución de nombres para colas de clúster.

4. Una llamada MQOPEN con la opción MQOO_BROWSE establece un cursor de exploración, para su uso con llamadas MQGET que especifican el identificador del objeto y una de las opciones de exploración. Esto permite escanear la cola sin alterar su contenido. Un mensaje que se ha encontrado al navegar se puede eliminar de la cola utilizando la opción MQGMO_MSG_UNDER_CURSOR.

Se pueden activar varios cursores de exploración para una sola aplicación mediante la emisión de varias solicitudes MQOPEN para la misma cola.

5. Las aplicaciones iniciadas por un monitor de activadores reciben el nombre de la cola asociada a la aplicación cuando esta se inicia. Este nombre de cola se puede especificar en el **ObjDesc** parámetro para abrir la cola. Para conocer detalles, consulte [“MQTMC2 - Mensaje de activación 2 \(formato de caracteres\)”](#) en la página 603.

Opciones de lectura anticipada

Cuando se llama a MQOPEN con MQOO_READ_AHEAD, el cliente de IBM MQ sólo permite la lectura anticipada si se cumplen ciertas condiciones. Estas condiciones incluyen:

- La aplicación cliente debe compilarse y enlazarse a las bibliotecas de cliente MQI de IBM MQ con hebras.
- El canal de cliente debe utilizar el protocolo TCP/IP
- El canal debe tener un valor SharingConversations (SHARECNV) distinto de cero tanto en las definiciones de canal de cliente y servidor.

Las siguientes notas se aplican al uso de las opciones de lectura anticipada.

1. Las opciones de lectura anticipada solo son aplicables cuando se especifica una, y solo una, de las opciones MQOO_BROWSE, MQOO_INPUT_SHARED y MQOO_INPUT_EXCLUSIVE. No se genera un error si se especifican opciones de lectura anticipada con las opciones MQOO_INQUIRE o MQOO_SET.
2. La lectura anticipada no se habilita cuando se solicita si las opciones utilizadas en la primera llamada MQGET no son compatibles con la lectura anticipada. Además, la lectura anticipada se desactiva cuando el cliente se conecta a un gestor de colas que no admite la lectura anticipada.
3. Si la aplicación no se está ejecutando como un IBM MQ cliente, se ignoran las opciones de lectura anticipada.

Colas de clúster

Las siguientes notas se aplican al uso de colas de clúster.

1. Cuando se abre una cola de clúster por primera vez y el gestor de colas local no es un gestor de colas de repositorio completo, el gestor de colas local obtiene información sobre la cola de clúster de un gestor de colas de repositorio completo. Cuando la red está ocupada, el gestor de colas local puede tardar varios segundos en recibir la información necesaria del gestor de colas del repositorio. Como resultado, la aplicación que emite la llamada MQOPEN podría tener que esperar hasta 10 segundos antes de que el control vuelva de la llamada MQOPEN. Si el gestor de colas local no recibe la información necesaria sobre la cola del clúster en este tiempo, la llamada falla con el código de motivo MQRC_CLUSTER_RESOLUTION_ERROR.

2. Cuando se abre una cola de clúster y hay varias instancias de la cola en el clúster, la instancia que se abre depende de las opciones especificadas en la llamada MQOPEN:

- Si las opciones especificadas incluyen alguna de las siguientes:
 - MQOO_BROWSE
 - MQOO_INPUT_AS_Q_DEF
 - MQOO_INPUT_EXCLUSIVE
 - MQOO_INPUT_SHARED
 - MQOO_SET

La instancia de la cola del clúster abierta debe ser la instancia local. Si no hay ninguna instancia local de la cola, la llamada MQOPEN falla.

- Si las opciones especificadas no incluyen ninguna de las opciones descritas anteriormente, pero incluyen una o ambas de las siguientes:
 - MQOO_INQUIRE
 - MQOO_OUTPUT

La instancia abierta es la instancia local, si existe, y una instancia remota en caso contrario (si se utilizan los valores predeterminados de CLWLUSEQ). Sin embargo, la instancia elegida por el gestor de colas puede ser modificada por una salida de carga de trabajo del clúster (si la hay).

3. Si hay una suscripción para la cola, pero no está reconocida por un repositorio completo, el objeto no está presente en el clúster y la llamada falla con el código de motivo MQRC_OBJECT_NAME.

Para obtener más información sobre las colas de clúster, consulte [Colas de clúster](#).

Listas de distribución

Las siguientes notas se aplican al uso de listas de distribución.

Las listas de distribución son compatibles con los siguientes entornos:

-  AIX
-  IBM i
-  Linux
-  Windows

y para IBM MQ MQI clients conectados a estos sistemas.

1. Los campos de la estructura MQOD deben configurarse de la siguiente manera al abrir una lista de distribución:

- Version debe ser MQOD_VERSION_2 o superior.
- ObjectType debe ser MQOT_Q.
- ObjectName Debe estar en blanco o contener una cadena nula.
- ObjectQMgrName Debe estar en blanco o contener una cadena nula.
- RecsPresent Tiene que ser mayor que cero.
- Uno de ObjectRecOffset y ObjectRecPtr debe ser cero y el otro distinto de cero.
- No más de uno de ResponseRecOffset y ResponseRecPtr puede ser distinto de cero.
- Debe haber registros RecsPresent de objetos, dirigidos por ObjectRecOffset o ObjectRecPtr. Los registros de objetos deben configurarse con los nombres de las colas de destino que se van a abrir.

- Si uno de `ResponseRecOffset` y `ResponseRecPtr` es distinto de cero, debe haber `RecsPresent` registros de respuesta presentes. Los establece el gestor de colas si la llamada se completa con el código de motivo `MQRC_MULTIPLE_REASONS`.

version-2 MQOD también se puede utilizar para abrir una sola cola que no esté en una lista de distribución, asegurándose de que `RecsPresent` sea cero.

2. Solo las siguientes opciones abiertas son válidas en el **Options** parámetro:

- `MQOO_OUTPUT`
- `MQOO_PASS_*_CONTEXT`
- `MQOO_SET_*_CONTEXT`
- `MQOO_ALTERNATE_USER_AUTHORITY`
- `MQOO_FAIL_IF_QUIESCING`

3. Las colas de destino de la lista de distribución pueden ser colas locales, alias o remotas, pero no pueden ser colas modelo. Si se especifica una cola modelo, dicha cola no se abre, con el código de motivo `MQRC_Q_TYPE_ERROR`. Sin embargo, esto no impide que otras colas de la lista se abran correctamente.

4. Los parámetros del código de finalización y del código de motivo se establecen de la siguiente manera:

- Si todas las operaciones abiertas para las colas de la lista de distribución tienen éxito o fracasan de la misma manera, los parámetros del código de finalización y del código de motivo se establecen para describir el resultado común. Los registros de respuesta MQRR (si los proporciona la aplicación) no se establecen en este caso.

Por ejemplo, si todas las aperturas tienen éxito, el código de finalización se establece en `MQCC_OK` y el código de motivo se establece en `MQRC_NONE`; si todas las aperturas fallan porque ninguna de las colas existe, los parámetros se establecen en `MQCC_FAILED` y `MQRC_UNKNOWN_OBJECT_NAME`.

- Si las operaciones abiertas para las colas de la lista de distribución no tienen todas el mismo resultado, ya sea éxito o fracaso:
 - El parámetro del código de finalización se establece en `MQCC_WARNING` si al menos una apertura se ha realizado correctamente, y en `MQCC_FAILED` si todas han fallado.
 - El parámetro del código de motivo se establece en `MQRC_MULTIPLE_REASONS`.
 - Los registros de respuesta (si los proporciona la aplicación) se establecen en los códigos de finalización individuales y los códigos de motivo para las colas de la lista de distribución.

5. Cuando se ha abierto correctamente una lista de distribución, el identificador `Hobj` devuelto por la llamada se puede utilizar en llamadas MQPUT posteriores para colocar mensajes en las colas de la lista de distribución, y en una llamada MQCLOSE para renunciar al acceso a la lista de distribución. La única opción válida para cerrar una lista de distribución es `MQCO_NONE`.

La llamada `MQPUT1` también se puede utilizar para enviar un mensaje a una lista de distribución; la estructura MQOD que define las colas de la lista se especifica como parámetro en esa llamada.

6. Cada destino abierto correctamente en la lista de distribución cuenta como un identificador independiente al comprobar si la aplicación ha superado el número máximo permitido de identificadores (consulte el atributo del **MaxHandles** gestor de colas). Esto es así incluso cuando dos o más destinos de la lista de distribución se resuelven en la misma cola física. Si la llamada MQOPEN o MQPUT1 para una lista de distribución provocara que el número de identificadores en uso por la aplicación superara `MaxHandles`, la llamada fallaría con el código de motivo `MQRC_HANDLE_NOT_AVAILABLE`.

7. Cada destino que se abre correctamente tiene el valor de su **OpenOutputCount** atributo incrementado en uno. Si dos o más destinos de la lista de distribución se resuelven en la misma cola física, el **OpenOutputCount** atributo de esa cola se incrementa en el número de destinos de la lista de distribución que se resuelven en esa cola.

8. Cualquier cambio en las definiciones de la cola que hubiera provocado que un identificador dejara de ser válido si las colas se hubieran abierto individualmente (por ejemplo, un cambio en la ruta de

resolución), no provoca que el identificador de la lista de distribución deje de ser válido. Sin embargo, sí provoca un error en esa cola concreta cuando se utiliza el identificador de la lista de distribución en una llamada MQPUT posterior.

9. Una lista de distribución solo puede contener un destino.

Colas remotas

Las siguientes notas se aplican al uso de colas remotas.

Se puede especificar una cola remota de dos maneras en el **ObjDesc** parámetro de esta llamada.

- Especificando el nombre `ObjectName` de una definición local de la cola remota. En este caso, `ObjectQMgrName` se refiere al gestor de colas local y puede especificarse como espacios en blanco o (en el lenguaje de programación C) una cadena nula.

La validación de seguridad realizada por el gestor de colas local verifica que el usuario está autorizado para abrir la definición local de la cola remota.

- Especificando `ObjectName` el nombre de la cola remota tal y como lo conoce el gestor de colas remotas. En este caso, `ObjectQMgrName` es el nombre del gestor de colas remoto.

La validación de seguridad realizada por el gestor de colas local verifica que el usuario está autorizado para enviar mensajes a la cola de transmisión resultante del proceso de resolución de nombres.

En cualquier caso:

- El gestor de colas local no envía ningún mensaje al gestor de colas remoto para comprobar que el usuario está autorizado a colocar mensajes en la cola.
- Cuando un mensaje llega al gestor de colas remoto, este puede rechazarlo porque el usuario que lo ha originado no está autorizado.

Consulte los campos `ObjectName` `ObjectQMgrName` y descritos en [“MQOD - Descriptor de objeto”](#) en la [página 454](#) para obtener más información.

Objetos

Seguridad

Las siguientes notas se refieren a los aspectos de seguridad relacionados con el uso de MQOPEN.

El gestor de colas realiza comprobaciones de seguridad cuando se emite una llamada MQOPEN, con el fin de verificar que el identificador de usuario bajo el que se ejecuta la aplicación tiene el nivel de autoridad adecuado antes de permitir el acceso. La comprobación de autoridad se realiza sobre el nombre del objeto que se está abriendo, y no sobre el nombre o nombres resultantes tras la resolución de un nombre.

Si el objeto que se abre es una cola alias que apunta a un objeto tema, el gestor de colas realiza una comprobación de seguridad del nombre de la cola alias antes de realizar una comprobación de seguridad del tema, como si se hubiera utilizado directamente el objeto tema.

Si el objeto que se abre es un objeto de tema, ya sea solo `ObjectName` con o utilizando `ObjectString` (con o sin una base `ObjectName`), el gestor de colas realiza la comprobación de seguridad utilizando la cadena de tema resultante, tomada del objeto de tema especificado en `ObjectName`, y, si es necesario, concatenándola con la proporcionada en `ObjectString`, y luego buscando el objeto de tema más cercano en o por encima de ese punto en el árbol de temas para realizar la comprobación de seguridad. Puede que este no sea el mismo objeto de tema que se especificó en `ObjectName`.

Si el objeto que se abre es una cola modelo, el gestor de colas realiza una comprobación de seguridad completa tanto del nombre de la cola modelo como del nombre de la cola dinámica que se crea. Si la cola dinámica resultante se abre explícitamente, se realiza una comprobación de seguridad de recursos adicional con respecto al nombre de la cola dinámica.

z/OS En z/OS, el gestor de colas realiza comprobaciones de seguridad solo si la seguridad está habilitada. Para obtener más información sobre la comprobación de seguridad, consulte [Configuración de la seguridad en z/OS](#).

Atributos

Las siguientes notas se refieren a los atributos.

Los atributos de un objeto pueden cambiar mientras una aplicación tiene el objeto abierto. En muchos casos, la aplicación no se da cuenta de esto, pero para ciertos atributos, el gestor de colas marca el identificador como no válido. Estos atributos son:

- Cualquier atributo que afecte a la resolución del nombre del objeto. Esto se aplica independientemente de las opciones abiertas utilizadas e incluye lo siguiente:
 - Un cambio en el **BaseQName** atributo de una cola alias que está abierta.
 - Un cambio en el **TargetType** atributo de una cola alias que está abierta.
 - Un cambio en los atributos **RemoteQName** de **RemoteQMGrName** la cola o, para cualquier identificador que esté abierto para esta cola, o para una cola que se resuelva a través de esta definición como un alias del gestor de colas.
 - Cualquier cambio que provoque que un identificador actualmente abierto para una cola remota se resuelva en una cola de transmisión diferente, o que no se resuelva en ninguna. Por ejemplo, esto puede incluir:
 - Un cambio en el **XmitQName** atributo de la definición local de una cola remota, independientemente de si la definición se utiliza para una cola o para un alias del gestor de colas.
 - **z/OS** Solo z/OS en caso de que se produzca un cambio en el valor del atributo del gestor **IntraGroupQueuing** de colas o un cambio en la definición de la cola de transmisión compartida (SYSTEM.QSG.TRANSMIT.QUEUE) utilizada por el agente IGQ.

Hay una excepción a esto: la creación de una nueva cola de transmisión. Un identificador que se habría resuelto en esta cola si hubiera estado presente cuando se abrió el identificador, pero que en su lugar se resolvió en la cola de transmisión predeterminada, no se invalida.

- Un cambio en el atributo del **DefXmitQName** gestor de colas. En este caso, todos los identificadores abiertos que se resolvieron en la cola mencionada anteriormente (que se resolvió en ella solo porque era la cola de transmisión predeterminada) se marcan como no válidos. Los identificadores que se han resuelto en esta cola por otros motivos no se ven afectados.
- El atributo de **Shareability** cola, si hay dos o más identificadores que actualmente proporcionan acceso MQOO_INPUT_SHARED para esta cola, o para una cola que se resuelve en esta cola. Si es así, todos los manejadores que están abiertos para esta cola, o para una cola que se resuelve en esta cola, se marcan como no válidos, independientemente de las opciones abiertas.

z/OS En z/OS, los identificadores descritos anteriormente se marcan como no válidos si uno o más identificadores están proporcionando actualmente acceso MQOO_INPUT_SHARED o MQOO_INPUT_EXCLUSIVE a la cola.

- El **Usage** atributo de cola, para todos los identificadores que están abiertos para esta cola, o para una cola que se resuelve en esta cola, independientemente de las opciones abiertas.

Cuando un identificador se marca como no válido, todas las llamadas posteriores (excepto MQCLOSE) que utilicen este identificador fallarán con el código de motivo MQRC_OBJECT_CHANGED. La aplicación debe emitir una llamada MQCLOSE (utilizando el identificador original) y, a continuación, volver a abrir la cola. Las actualizaciones no confirmadas realizadas con el identificador antiguo de llamadas anteriores correctas aún pueden confirmarse o revertirse, según lo requiera la lógica de la aplicación.

Si cambiar un atributo provoca que esto suceda, utilice una versión especial forzada de la llamada.

Invocación en C

```
MQOPEN (Hconn, &ObjDesc, Options, &Hobj, &CompCode,
        &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQHCONN  Hconn;      /* Connection handle */
MQOD      ObjDesc;    /* Object descriptor */
MQLONG    Options;    /* Options that control the action of MQOPEN */
MQHOBJ    Hobj;       /* Object handle */
MQLONG    CompCode;   /* Completion code */
MQLONG    Reason;     /* Reason code qualifying CompCode */
```

Invocación en COBOL

```
CALL 'MQOPEN' USING HCONN, OBJDESC, OPTIONS, HOBJ, COMPCODE, REASON
```

Declare los parámetros como se indica a continuación:

```
** Connection handle
01 HCONN      PIC S9(9) BINARY.
** Object descriptor
01 OBJDESC.
   COPY CMQODV.
** Options that control the action of MQOPEN
01 OPTIONS    PIC S9(9) BINARY.
** Object handle
01 HOBJ       PIC S9(9) BINARY.
** Completion code
01 COMPCODE   PIC S9(9) BINARY.
** Reason code qualifying COMPCODE
01 REASON     PIC S9(9) BINARY.
```

Invocación en PL/I

```
call MQOPEN (Hconn, ObjDesc, Options, Hobj, CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```
dcl Hconn      fixed bin(31); /* Connection handle */
dcl ObjDesc    like MQOD;     /* Object descriptor */
dcl Options    fixed bin(31); /* Options that control the action of
                               MQOPEN */
dcl Hobj       fixed bin(31); /* Object handle */
dcl CompCode   fixed bin(31); /* Completion code */
dcl Reason     fixed bin(31); /* Reason code qualifying CompCode */
```

Invocación en ensamblador de alto nivel

```
CALL MQOPEN,(HCONN,OBJDESC,OPTIONS,HOBJ,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

```
HCONN      DS      F  Connection handle
OBJDESC     CMQODA   ,  Object descriptor
OPTIONS     DS      F  Options that control the action of MQOPEN
HOBJ        DS      F  Object handle
COMPCODE    DS      F  Completion code
REASON      DS      F  Reason code qualifying COMPCODE
```

Invocación en Visual Basic



```
MQOPEN Hconn, ObjDesc, Options, Hobj, CompCode, Reason
```

Declare los parámetros como se indica a continuación:

```
Dim Hconn      As Long 'Connection handle'
Dim ObjDesc    As MQOD 'Object descriptor'
Dim Options    As Long 'Options that control the action of MQOPEN'
Dim Hobj       As Long 'Object handle'
Dim CompCode   As Long 'Completion code'
Dim Reason     As Long 'Reason code qualifying CompCode'
```

MQPUT - Enviar mensaje

La llamada MQPUT coloca un mensaje en una cola o lista de distribución, o en un tema. La cola, la lista de distribución o el tema deben estar ya abiertos.

Sintaxis

MQPUT (*Hconn, Hobj, MsgDesc, PutMsgOpts, BufferLength, Buffer, CompCode, Reason*)

Parámetros

Hconn

Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de Hconn fue devuelto por una llamada anterior a MQCONN o MQCONNX.

 En z/OS para CICS aplicaciones, se puede omitir la llamada MQCONN y especificar el siguiente valor para Hconn:

MQHC_DEF_HCONN

Manejador de conexión predeterminado.

Hobj

Tipo: MQHOBJ - entrada

Este identificador representa la cola a la que se añade el mensaje o el tema en el que se publica el mensaje. El valor de Hobj fue devuelto por una llamada MQOPEN anterior que especificaba la opción MQOO_OUTPUT.

MsgDesc

Tipo: MQMD - entrada/salida

Esta estructura describe los atributos del mensaje que se envía y recibe información sobre el mensaje una vez completada la solicitud put. Consulte [“MQMD - Descriptor de mensaje”](#) en la página 389 para obtener detalles.

Si la aplicación proporciona un MQMD version-1, los datos del mensaje pueden ir precedidos de una estructura MQMDE para especificar los valores de los campos que existen en el MQMD version-2 pero no en el version-1. El campo *Formato* del MQMD debe establecerse en MQFMT_MD_EXTENSION para indicar que hay un MQMDE presente. Consulte [“MQMDE - Extensión del descriptor de mensajes”](#) en la página 443 para obtener más detalles.

La aplicación no necesita proporcionar una estructura MQMD si se proporciona un identificador de mensaje válido en los campos OriginalMsgHandle NewMsgHandle o de la estructura MQPMO. Si no se proporciona nada en uno de estos campos, el descriptor del mensaje se toma del descriptor asociado con los identificadores de mensaje.

Si utiliza o tiene previsto utilizar salidas API, le recomendamos que proporcione explícitamente una estructura MQMD y no utilice los descriptores de mensajes asociados a los identificadores de mensajes. Esto se debe a que la salida API asociada con la llamada MQPUT o MQPUT1 no puede determinar qué valores MQMD utiliza el gestor de colas para completar la solicitud MQPUT o MQPUT1.

PutMsgOpts

Tipo: MQPMO - entrada/salida

Consulte “MQPMO - Opciones de mensaje de envío” en la página 477 para obtener los detalles.

BufferLength

Tipo: MQLONG - entrada

La longitud del mensaje en `Buffer`. El valor cero es válido e indica que el mensaje no contiene datos de aplicación. El límite superior para `BufferLength` depende de varios factores:

- Si el destino es una cola local o se resuelve en una cola local, el límite superior depende de si:
 - El gestor de colas local admite la segmentación.
 - La aplicación de envío especifica el indicador que permite al gestor de colas segmentar el mensaje. Esta bandera es `MQMF_SEGMENTATION_ALLOWED` y se puede especificar en un MQMD de tipo « version-2 » (MQMD de tipo 1) o en un MQMDE utilizado con un MQMD de tipo « version-1 » (MQMD de tipo 2).

Si se cumplen ambas condiciones, `BufferLength` no puede superar 999 999 999 menos el valor del `Offset` campo en MQMD. Por lo tanto, el mensaje lógico más largo que se puede introducir es de 999 999 999 bytes (cuando `Offset` es cero). Sin embargo, las restricciones de recursos impuestas por el sistema operativo o el entorno en el que se ejecuta la aplicación pueden dar lugar a un límite inferior.

Si no se cumple una o ambas condiciones anteriores, `BufferLength` no puede superar el menor de los atributos **MaxMsgLength** de la cola y el **MaxMsgLength** atributo del gestor de colas.

- Si el destino es una cola remota o se resuelve en una cola remota, se aplican las condiciones para las colas locales, pero en cada gestor de colas por el que debe pasar el mensaje para llegar a la cola de destino; en particular:
 1. La cola de transmisión local utilizada para almacenar temporalmente el mensaje en el gestor de colas local
 2. Colas de transmisión intermedias (si las hay) utilizadas para almacenar el mensaje en los gestores de colas en la ruta entre los gestores de colas locales y de destino
 3. La cola de destino en el gestor de colas de destino

Por lo tanto, el mensaje más largo que se puede enviar está determinado por la más restrictiva de estas colas y gestores de colas.

Cuando un mensaje se encuentra en una cola de transmisión, la información adicional reside con los datos del mensaje, lo que reduce la cantidad de datos de la aplicación que se pueden transportar. En esta situación, reste `MQ_MSG_HEADER_LENGTH` bytes de los `MaxMsgLength` valores de las colas de transmisión al determinar el límite para `BufferLength`.

Nota: Solo el incumplimiento de la condición 1 puede diagnosticarse de forma sincrónica (con el código de motivo `MQRC_MSG_TOO_BIG_FOR_Q` o `MQRC_MSG_TOO_BIG_FOR_Q_MGR`) cuando se envía el mensaje. Si no se cumplen las condiciones 2 o 3, el mensaje se redirige a una cola de mensajes no entregados (dead-letter), ya sea en un gestor de colas intermedio o en el gestor de colas de destino. Si esto ocurre, se genera un mensaje de informe si el remitente lo ha solicitado.

Almacenamiento intermedio

Tipo: MQBYTEBufferLength - entrada

Este es un búfer que contiene los datos de la aplicación que se van a enviar. El búfer debe alinearse en un límite adecuado a la naturaleza de los datos del mensaje. La alineación de 4 bytes es adecuada para la mayoría de los mensajes (incluidos los mensajes que contienen IBM MQ estructuras de encabezado), pero algunos mensajes pueden requerir una alineación más estricta. Por ejemplo, un mensaje que contenga un entero binario de 64 bits podría requerir una alineación de 8 bytes.

Si `Buffer` contiene datos numéricos o caracteres, establezca los campos `CodedCharSetId` `Encoding` y del **MsgDesc** parámetro con los valores adecuados para los datos; esto permite al

destinatario del mensaje convertir los datos (si es necesario) al juego de caracteres y la codificación utilizados por el destinatario.

Nota: Todos los demás parámetros de la llamada MQPUT deben estar en el juego de caracteres y la codificación del gestor de colas local (indicados por el atributo del **CodedCharSetId** gestor de colas y MQENC_NATIVE).

En el lenguaje de programación C, el parámetro se declara como un puntero a void; se puede especificar como parámetro la dirección de cualquier tipo de datos.

Si el **BufferLength** parámetro es cero, no **Buffer** se hace referencia a; en este caso, la dirección del parámetro pasada por programas escritos en C o en ensambladore System/390 e puede ser nula.

CompCode

Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_WARNING

Aviso (finalización parcial).

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

El código de motivo que califica CompCode.

Si CompCode es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si CompCode es MQCC_WARNING:

MQRC_INCOMPLETE_GROUP

(2241, X'8C1') El grupo de mensajes no está completo.

MQRC_INCOMPLETE_MSG

(2242, X'8C2') El mensaje lógico no está completo.

MQRC_INCONSISTENT_PERSISTENCE

(2185, X'889') Especificación de persistencia inconsistente.

MQRC_INCONSISTENT_UOW

(2245, X'8C5') Especificación incoherente de unidad de trabajo.

MQRC_MÚLTIPLES_RAZONES

(2136, X'858') Se han devuelto varios códigos de motivo.

MQRC_PRIORIDAD_SUPERA_MÁXIMO

(2049, X'801') La prioridad del mensaje excede el valor máximo admitido.

MQRC_OPCIÓN_DE_INFORME_DESCONOCIDA

(2104, X'838') No se reconocen las opciones de informe en el descriptor del mensaje.

Si CompCode es MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'89C') El adaptador no está disponible.

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

MQRC_ALIAS_TARGET_TYPE_CHANGED

(2480, X'09B0') El tipo de destino de la suscripción ha cambiado de objeto de cola a objeto de tema.

MQRC_API_EXIT_ERROR

(2374, X'946') La salida de la API ha fallado.

MQRC_API_EXIT_LOAD_ERROR

(2183, X'887') No se ha podido cargar la salida de la API.

MQRC_ASID_MISMATCH

(2157, X'86D') Los ASID primario e inicial varían.

MQRC_BACKED_OUT

(2003, X'7D3') Unidad de trabajo restituida.

MQRC_BUFFER_ERROR

(2004, X'7D4') El parámetro almacenamiento intermedio no es válido.

MQRC_BUFFER_LENGTH_ERROR

(2005, X'7D5') El parámetro de longitud de almacenamiento intermedio no es válido.

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') La llamada MQI ha entrado antes de completar la llamada anterior.

MQRC_CALL_INTERRUPTED

(2549, X'9F5') MQPUT o MQCMIT se interrumpieron y el proceso de reconexión no puede restablecer un resultado definitivo.

MQRC_CF_NO_DISPONIBLE

(2345, X'929') Instalación de acoplamiento no disponible.

MQRC_CF_STRUC_FAILED

(2373, X'945') La estructura de recurso de acoplamiento ha fallado.

MQRC_CF_STRUC_IN_USE

(2346, X'92A') La estructura de recurso de acoplamiento se está utilizando.

MQRC_CFGR_ERROR

(2416, X'970') La estructura del parámetro del grupo PCF MQCFGR en los datos del mensaje no es válida.

MQRC_CFH_ERROR

(2235, X'8BB') Estructura de encabezado PCF no válida.

MQRC_CFIF_ERROR

(2414, X'96E') La estructura del parámetro del filtro entero PCF en los datos del mensaje no es válida.

MQRC_CFIL_ERROR

(2236, X'8BC') Estructura de parámetro de lista de enteros PCF o estructura de parámetro de lista de enteros PCIF*64 no válida.

MQRC_CFIN_ERROR

(2237, X'8BD') Estructura de parámetro entero PCF o estructura de parámetro entero PCIF*64 no válida.

MQRC_CFSF_ERROR

(2415, X'96F') La estructura del parámetro del filtro de cadena PCF en los datos del mensaje no es válida.

MQRC_CFSL_ERROR

(2238, X'8BE') Estructura del parámetro de la lista de cadenas PCF no válida.

MQRC_CFST_ERROR

(2239, X'8BF') Estructura del parámetro de cadena PCF no válida.

MQRC_CICS_WAIT_FAILED

(2140, X'85C') Solicitud de espera rechazada por CICS.

ERROR DE SALIDA DEL CLÚSTER MQRC

(2266, X'8DA') Error al salir de la carga de trabajo del clúster.

MQRC_CLUSTER_RESOLUTION_ERROR

(2189, X'88D') La resolución de nombres de clúster ha fallado.

ERROR_RECURSO_CLÚSTER_MQRC

(2269, X'8DD') Error de recursos del clúster.

MQRC_CÓDIGO_NO_VÁLIDO_PARA_XCF_Q

(2106, X'83A') La opción de informe COD no es válida para la cola XCF.

MQRC_CONNECTION_BROKEN

(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

MQRC_CONNECTION_NOT_AUTHORIZED

(2217, X'8A9') No tiene autorización para la conexión.

MQRC_CONNECTION QUIESCING

(2202, X'89A') Conexión en fase de inmovilización.

MQRC_CONNECTION_STOPPING

(2203, X'89B') La conexión está concluyendo.

ERROR_DE CONTENIDO_MQRC

2554 (X'09FA') No se pudo analizar el contenido del mensaje para determinar si el mensaje debía entregarse a un suscriptor con un selector de mensajes ampliado.

MQRC_CONTEXTO_ERROR_MANEJO

(2097, X'831') El identificador de cola al que se hace referencia no guarda el contexto.

MQRC_CONTEXTO_NO_DISPONIBLE

(2098, X'832') No hay contexto disponible para el identificador de cola al que se hace referencia.

MQRC_DATA_LENGTH_ERROR

(2010, X'7DA') El parámetro longitud de datos no es válido.

MQRC_DH_ERROR

(2135, X'857') Estructura del encabezado de distribución no válida.

MQRC_DLH_ERROR

(2141, X'85D') Estructura del encabezado de carta muerta no válida.

MQRC_EPH_ERROR

(2420, X'974') Estructura PCF incrustada no válida.

ERROR_CADUCIDAD_MQRC

(2013, X'7DD') Fecha de caducidad no válida.

MQRC_ERROR_DE_RESPUESTA

(2014, X'7DE') Código de retroalimentación no válido.

MQRC_GLOBAL_UOW_CONFLICT

(2351, X'92F') Conflicto de unidades de trabajo global.

MQRC_GROUP_ID_ERROR

(2258, X'8D2') Identificador de grupo no válido.

MQRC_HANDLE_IN_USE_FOR_UOW

(2353, X'931') Descriptor de contexto que se utiliza para la unidad de trabajo global.

MQRC_HCONN_ERROR

(2018, X'7E2') El manejador de conexión no es válido.

MQRC_ERROR_ENCABEZADO

(2142, X'85E') MQ Estructura del encabezado no válida.

MQRC_HOBJ_ERROR

(2019, X'7E3') El manejador de objeto no es válido.

MQRC_IIH_ERROR

(2148, X'864') Estructura de IMS encabezado de información no válida.

MQRC_INCOMPLETE_GROUP

(2241, X'8C1') El grupo de mensajes no está completo.

MQRC_INCOMPLETE_MSG

(2242, X'8C2') El mensaje lógico no está completo.

MQRC_INCONSISTENT_PERSISTENCE

(2185, X'889') Especificación de persistencia inconsistente.

MQRC_INCONSISTENT_UOW

(2245, X'8C5') Especificación incoherente de unidad de trabajo.

MQRC_LOCAL_UOW_CONFLICT

(2352, X'930') La unidad de trabajo global entra en conflicto con la unidad de trabajo local.

MQRC_MD_ERROR

(2026, X'7EA') El descriptor de mensaje no es válido.

MQRC_MDE_ERROR

(2248, X'8C8') Extensión del descriptor de mensaje no válida.

MQRC_RESPUESTA_FALTANTE_A_P

(2027, X'7EB') Falta la cola de respuestas o se ha utilizado MQPMO_SUPPRESS_REPLYTO

MQRC_FALTA_WIH

(2332, X'91C') Los datos del mensaje no comienzan con MQWIH.

MQRC_MSG_FLAGS_ERROR

(2249, X'8C9') Indicadores de mensaje no válidos.

MQRC_MSG_SEQ_NUMBER_ERROR

(2250, X'8CA') El número de secuencia de mensaje no es válido.

MQRC_MSG_DEMASIADO_GRANDE_PARA_Q

(2030, X'7EE') La longitud del mensaje supera el máximo permitido para la cola.

MQRC_MSG_DEMASIADO_GRANDE_PARA_Q_MGR

(2031, X'7EF') La longitud del mensaje supera el máximo permitido por el gestor de colas.

MQRC_MSG_TYPE_ERROR

(2029, X'7ED') Tipo de mensaje no válido en el descriptor de mensajes.

MQRC_MÚLTIPLES_RAZONES

(2136, X'858') Se han devuelto varios códigos de motivo.

MQRC_NO_DESTINOS_DISPONIBLES

(2270, X'8DE') No hay colas de destino disponibles.

MQRC_NO_ABIERTO_PARA_SALIDA

(2039, X'7F7') La cola no está abierta para la salida.

MQRC_NO_ABIERTO_PARA_PASAR_TODO

(2093, X'82D') La cola no está abierta para pasar todo el contexto.

MQRC_NO_ABIERTO_PARA_PASAR_IDENTIFICACIÓN

(2094, X'82E') La cola no está abierta para el contexto de identidad de pase.

MQRC_NO_ABIERTO_PARA_CONFIGURAR_TODO

(2095, X'82F') La cola no está abierta para establecer todo el contexto.

MQRC_NO_ABIERTO_PARA_ESTABLECER_IDENTIFICADOR

(2096, X'830') Cola no abierta para el contexto de identidad establecido.

MQRC_OBJECT_CHANGED

(2041, X'7F9') La definición de objeto ha cambiado desde que se ha abierto.

MQRC_OBJECT_DAMAGED

(2101, X'835') El objeto se ha dañado.

ERROR_DESVIACIÓN_MQRC

(2251, X'8CB') Desplazamiento del segmento de mensaje no válido.

MQRC_ABIERTO_FALLIDO

(2137, X'859') No se ha podido abrir el objeto.

MQRC_OPTIONS_ERROR

(2046, X'7FE') Las opciones no son válidas o no son coherentes.

ERROR_DE_LONGITUD_ORIGINAL_MQRC

(2252, X'8CC') Longitud original no válida.

MQRC_PAGESET_ERROR

(2193, X'891') Error al acceder al conjunto de datos del conjunto de páginas.

MQRC_PÁGINAS_COMPLETAS

(2192, X'890') El soporte de almacenamiento externo está lleno.

MQRC_PCF_ERROR

(2149, X'865') Estructuras PCF no válidas.

MQRC_PERSISTENCIA_ERROR

(2047, X'7FF') Persistencia no válida.

MQRC_PERSISTENTE_NO_PERMITIDO

(2048, X'800') La cola no admite mensajes persistentes.

MQRC_PMO_ERROR

(2173, X'87D') Estructura de opciones de mensaje de envío no válida.

MQRC_PMO_REGISTRO_FLAGS_ERROR

(2158, X'86E') Los indicadores de registro de mensajes no son válidos.

MQRC_ERROR_PRIORIDAD

(2050, X'802') Prioridad del mensaje no válida.

MQRC_FALLO_PUBLICACIÓN

(2502, X'9C6') La publicación no se ha entregado a ninguno de los suscriptores.

MQRC_PUT_INHIBITED

(2051, X'803') Inhibición de llamadas de puesta para la cola, para la cola a la que se resuelve esta cola o para el tema.

MQRC_PUT_MSG_RECORDS_ERROR

(2159, X'86F') Los registros de mensajes no son válidos.

MQRC_PUT_NO_RETENIDO

(2479, X'09AF') No se pudo conservar la publicación

MQRC_Q_DELETED

(2052, X'804') La cola se ha suprimido.

MQRC_Q_COMPLETO

(2053, X'805') La cola ya contiene el número máximo de mensajes.

MQRC_Q_MGR_NAME_ERROR

(2058, X'80A') El nombre del gestor de colas no es válido o no se conoce.

MQRC_Q_MGR_NOT_AVAILABLE

(2059, X'80B') El gestor de colas no está disponible para efectuar la conexión.

MQRC_Q_MGR QUIESCING

(2161, X'871') El gestor de colas se está desactivando temporalmente.

MQRC_Q_MGR_STOPPING

(2162, X'872') El gestor de colas está concluyendo.

MQRC_Q ESPACIO_NO_DISPONIBLE

(2056, X'808') No hay espacio disponible en el disco para la cola.

MQRC_RECONNECT_FAILED

(2548, X'9F4') Después de volver a conectarse, se produjo un error al restablecer los identificadores para una conexión reconectable.

MQRC_RECS_PRESENT_ERROR

(2154, X'86A') Número de registros presentes no válido.

MQRC_REPORT_OPTIONS_ERROR

(2061, X'80D') Las opciones de informe en el descriptor de mensajes no son válidas.

MQRC_RESOURCE_PROBLEM

(2102, X'836') No hay disponibles suficientes recursos del sistema.

ERROR EN LOS REGISTROS DE RESPUESTA MQRC

(2156, X'86C') Registros de respuesta no válidos.

MQRC_RFH_ERROR

(2334, X'91E') Estructura MQRFH o MQRFH2 no válida.

MQRC_RMH_ERROR

(2220, X'8AC') Estructura del encabezado del mensaje de referencia no válida.

MQRC_LONGITUD_DEL_SEGMENTO_CERO

(2253, X'8CD') La longitud de los datos en el segmento del mensaje es cero.

MQRC_SEGMENTOS_NO_COMPATIBLES

(2365, X'93D') Segmentos no compatibles.

MQRC_SELECTION_NOT_AVAILABLE

2551 (X'09F7') Existe un posible suscriptor para la publicación, pero el gestor de colas no puede comprobar si debe enviar la publicación al suscriptor.

MQRC_DETENIDO_POR_SALIDA_DEL_CLÚSTER

(2188, X'88C') Llamada rechazada por salida de carga de trabajo del clúster.

MQRC_CLASE_ALMACENAMIENTO_ERROR

(2105, X'839') Error de clase de almacenamiento.

MQRC_STORAGE_MEDIUM_FULL

(2192, X'890') El soporte de almacenamiento externo está lleno.

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_SUPPRESSED_BY_EXIT

(2109, X'83D') El programa de salida ha suprimido la llamada.

MQRC_SYNCPOINT_LIMIT_REACHED

(2024, X'7E8') No pueden manejarse más mensajes dentro de la unidad de trabajo actual.

MQRC_SYNCPOINT_NOT_AVAILABLE

(2072, X'818') No hay soporte para Syncpoint.

MQRC_TM_ERROR

(2265, X'8D9') Estructura del mensaje de activación no válida.

MQRC_TMC_ERROR

(2191, X'88F') Estructura del mensaje de activación de caracteres no válida.

MQRC_UNEXPECTED_ERROR

(2195, X'893') Se ha producido un error inesperado.

MQRC_UOW_ENLISTMENT_ERROR

(2354, X'932') Ha fallado el listado en la unidad de trabajo global.

MQRC_UOW_MIX_NOT_SUPPORTED

(2355, X'933') No se soporta la mezcla de llamadas de unidad de trabajo.

MQRC_UOW_NOT_AVAILABLE

(2255, X'8CF') La unidad de trabajo no está disponible para ser utilizada por el gestor de colas.

MQRC_WIH_ERROR

(2333, X'91D') Estructura MQWIH no válida.

MQRC_WRONG_MD_VERSION

(2257, X'8D1') La versión del MQMD suministrado es incorrecta.

MQRC_XQH_ERROR

(2260, X'8D4') Estructura de encabezado de la cola de transmisión no válida.

Para obtener información detallada sobre estos códigos, consulte [Mensajes y códigos de motivo](#).

Notas sobre el uso del tema

1. Las siguientes notas se aplican al uso de los temas:

- a. Cuando se utiliza MQPUT para publicar mensajes en un tema, en el que uno o más suscriptores de ese tema no pueden recibir la publicación debido a un problema con su cola de suscriptores (por

ejemplo, porque está llena), el código de motivo devuelto a la llamada MQPUT y el comportamiento de entrega dependen de la configuración de los atributos PMSGDLV o NMSGDLV en el TOPIC. Tenga en cuenta que la entrega de una publicación a la cola de cartas muertas cuando se especifica MQRO_DEAD_LETTER_Q, o el descarte del mensaje cuando se especifica MQRO_DISCARD_MSG, se considera una entrega correcta del mensaje. Si no se entrega ninguna de las publicaciones, MQPUT devuelve MQRC_PUBLICATION_FAILURE. Esto puede ocurrir en los siguientes casos:

- Se publica un mensaje en un TEMA con PMSGDLV o NMSGDLV (dependiendo de la persistencia del mensaje) establecido en ALL y cualquier suscripción (duradera o no) tiene una cola que no puede recibir la publicación.
- Se publica un mensaje en un TEMA con PMSGDLV o NMSGDLV (dependiendo de la persistencia del mensaje) establecido en ALLDUR y una suscripción duradera tiene una cola que no puede recibir la publicación.

El MQPUT puede devolver MQRC_NONE aunque las publicaciones no se hayan podido entregar a algunos suscriptores en los siguientes casos:

- Se publica un mensaje en un TEMA con PMSGDLV o NMSGDLV (dependiendo de la persistencia del mensaje) establecido en ALLAVAIL y cualquier suscripción, duradera o no, tiene una cola que no puede recibir la publicación.
- Se publica un mensaje en un TEMA con PMSGDLV o NMSGDLV (dependiendo de la persistencia del mensaje) establecido en ALLDUR y una suscripción no duradera tiene una cola que no puede recibir la publicación.

Puede utilizar el atributo de tema USEDLC para determinar si se utiliza la cola de mensajes no entregados cuando los mensajes de publicación no se pueden entregar a su cola de suscriptores correcta. Para obtener más información sobre el uso de USEDLC, consulte [DEFINE TOPIC](#).

- b. Si no hay suscriptores al tema que se está utilizando, el mensaje publicado no se envía a ninguna cola y se descarta. No importa si el mensaje es persistente o no persistente, o si tiene una caducidad ilimitada o tiene un tiempo de caducidad, se descarta si no hay suscriptores. La excepción a esto es si el mensaje debe conservarse, en cuyo caso, aunque no se envía a las colas de ningún suscriptor, se almacena en el tema para ser entregado a cualquier nueva suscripción o a cualquier suscriptor que solicite publicaciones conservadas utilizando MQSUBRQ.

MQPUT y MQPUT1

Puede utilizar tanto las llamadas MQPUT como MQPUT1 para colocar mensajes en una cola; la llamada que se utilice dependerá de las circunstancias

- Utilice la llamada MQPUT para colocar varios mensajes en la misma cola.

Primero se emite una llamada MQOPEN que especifica la opción MQOO_OUTPUT, seguida de una o más solicitudes MQPUT para añadir mensajes a la cola; finalmente, la cola se cierra con una llamada MQCLOSE. Esto ofrece un mejor rendimiento que el uso repetido de la llamada MQPUT1.

- Utilice la llamada `MQPUT1` para colocar un solo mensaje en una cola.

Esta llamada encapsula las llamadas MQOPEN, MQPUT y MQCLOSE en una sola llamada, minimizando el número de llamadas que deben emitirse.

Colas de destino

Las siguientes notas se aplican al uso de colas de destino:

1. Si una aplicación coloca una secuencia de mensajes en la misma cola sin utilizar grupos de mensajes, El orden de esos mensajes se mantiene si se cumplen las condiciones detalladas. Algunas condiciones se aplican tanto a las colas de destino locales como a las remotas; otras condiciones solo se aplican a las colas de destino remotas.

Condiciones que se aplican a las colas de destino locales y remotas

- Todas las llamadas MQPUT se encuentran dentro de la misma unidad de trabajo, o ninguna de ellas se encuentra dentro de una unidad de trabajo.

Tenga en cuenta que cuando los mensajes se colocan en una cola concreta dentro de una sola unidad de trabajo, los mensajes de otras aplicaciones pueden intercalarse con la secuencia de mensajes de la cola.

- Todas las llamadas MQPUT se realizan utilizando el mismo identificador de objeto *Hobj*.

En algunos entornos, la secuencia de mensajes también se conserva cuando se utilizan diferentes identificadores de objetos, si las llamadas se realizan desde la misma aplicación. El significado de «*misma aplicación*» viene determinado por el entorno:

-  En z/OS, la aplicación es:
 - Para CICS, la CICS tarea
 - Para IMS, la tarea
 - Para z/OS el lote, la tarea
-  En IBM i, la aplicación es el trabajo.
-  En AIX, Linux, and Windows, la aplicación es el hilo.

- Los mensajes tienen todos la misma prioridad.
- Los mensajes no se colocan en una cola de clúster con MQOO_BIND_NOT_FIXED especificado (o con MQOO_BIND_AS_Q_DEF en vigor cuando el atributo de cola DefBind tiene el valor MQBND_BIND_NOT_FIXED).

Condiciones adicionales que se aplican a las colas de destino remotas

- Solo hay un camino desde el gestor de colas de envío hasta el gestor de colas de destino.
Si algunos mensajes de la secuencia pueden seguir una ruta diferente (por ejemplo, debido a una reconfiguración, al equilibrio del tráfico o a la selección de la ruta en función del tamaño del mensaje), no se puede garantizar el orden de los mensajes en el gestor de colas de destino.
- Los mensajes no se colocan temporalmente en colas de mensajes no entregados en los gestores de colas de envío, intermedios o de destino.

Si uno o más mensajes se colocan temporalmente en una cola de mensajes no entregados (por ejemplo, porque una cola de transmisión o la cola de destino están temporalmente llenas), los mensajes pueden llegar a la cola de destino fuera de secuencia.

- Los mensajes son todos persistentes o todos no persistentes.

Si un canal en la ruta entre los gestores de colas de envío y destino tiene su

NonPersistentMsgSpeed atributo establecido en MQNPMMS_FAST, los mensajes no persistentes pueden adelantarse a los mensajes persistentes, lo que da como resultado que no se conserve el orden de los mensajes persistentes en relación con los mensajes no persistentes. Sin embargo, se conserva el orden de los mensajes persistentes entre sí y de los mensajes no persistentes entre sí.

Si no se cumplen estas condiciones, puede utilizar grupos de mensajes para conservar el orden de los mensajes, pero esto requiere que tanto la aplicación emisora como la receptora utilicen la función de agrupación de mensajes. Para obtener más información sobre los grupos de mensajes, consulte:

- [MQMD - Campo MsgFlags](#)
- [MQPMO_LOGICAL_ORDER](#)
- [MQGMO_LOGICAL_ORDER](#)

Listas de distribución

Las siguientes notas se aplican al uso de listas de distribución.

Las listas de distribución son compatibles con los siguientes entornos:

-  AIX
-  IBM i
-  Linux
-  Windows

y para IBM MQ MQI clients conectados a estos sistemas.

1. Puede enviar mensajes a una lista de distribución utilizando un MQPMO de tipo « version-1 » (enviar a una lista de distribución) o « version-2 » (enviar a una lista de distribución). Si utiliza un MQPMO version-1 (o un MQPMO version-2 con RecsPresent igual a cero), la aplicación no puede proporcionar registros de mensajes enviados ni registros de respuesta. No se pueden identificar las colas que presentan errores si el mensaje se envía correctamente a algunas colas de la lista de distribución y a otras no.

Si la aplicación proporciona registros de mensajes enviados o registros de respuesta, establezca el Version campo en MQPMO_VERSION_2.

También puede utilizar un MQPMO version-2 para enviar mensajes a una sola cola que no esté en una lista de distribución, asegurándose de que RecsPresent sea cero.

2. Los parámetros del código de finalización y del código de motivo se establecen de la siguiente manera:

- Si todas las entradas en las colas de la lista de distribución tienen éxito o fracasan de la misma manera, los parámetros de código de finalización y código de motivo se establecen para describir el resultado común. Los registros de respuesta MQRR (si los proporciona la aplicación) no se establecen en este caso.

Por ejemplo, si todas las operaciones de inserción se realizan correctamente, el código de finalización y el código de motivo se establecen en MQCC_OK y MQRC_NONE; si todas las operaciones de inserción fallan porque todas las colas están inhibidas para las inserciones, los parámetros se establecen en MQCC_FAILED y MQRC_PUT_INHIBITED.

- Si las entradas en las colas de la lista de distribución no tienen todas el mismo resultado, ya sea éxito o fracaso:

- El parámetro del código de finalización se establece en MQCC_WARNING si al menos una operación se ha realizado correctamente, y en MQCC_FAILED si todas han fallado.
- El parámetro del código de motivo se establece en MQRC_MULTIPLE_REASONS.
- Los registros de respuesta (si los proporciona la aplicación) se establecen en los códigos de finalización individuales y los códigos de motivo para las colas de la lista de distribución.

Si la inserción en un destino falla porque la apertura para ese destino ha fallado, los campos del registro de respuesta se establecen en MQCC_FAILED y MQRC_OPEN_FAILED; ese destino se incluye en InvalidDestCount.

3. Si un destino de la lista de distribución se resuelve en una cola local, el mensaje se coloca en esa cola en forma normal (es decir, no como un mensaje de lista de distribución). Si más de un destino se resuelve en la misma cola local, se coloca un mensaje en la cola para cada uno de esos destinos.

Si un destino de la lista de distribución se resuelve en una cola remota, se coloca un mensaje en la cola de transmisión adecuada. Cuando varios destinos se resuelven en la misma cola de transmisión, se puede colocar en la cola de transmisión un único mensaje de lista de distribución que contenga dichos destinos, incluso si estos no eran adyacentes en la lista de destinos proporcionada por la aplicación. Sin embargo, esto solo se puede hacer si la cola de transmisión admite mensajes de listas de distribución (consulte [DistLists](#)).

Si la cola de transmisión no admite listas de distribución, se coloca una copia del mensaje en formato normal en la cola de transmisión para cada destino que utilice dicha cola de transmisión.

Si una lista de distribución con los datos del mensaje de la aplicación es demasiado grande para una cola de transmisión, el mensaje de la lista de distribución se divide en mensajes de lista de distribución más pequeños, cada uno con menos destinos. Si los datos del mensaje de la aplicación

apenas caben en la cola, los mensajes de la lista de distribución no se pueden utilizar en absoluto, y el gestor de colas genera una copia del mensaje en formato normal para cada destino que utiliza esa cola de transmisión.

Si diferentes destinos tienen diferentes prioridades o persistencias de mensajes (esto puede ocurrir cuando la aplicación especifica MQPRI_PRIORITY_AS_Q_DEF o MQPER_PERSISTENCE_AS_Q_DEF), los mensajes no se mantienen en el mismo mensaje de la lista de distribución. En su lugar, el gestor de colas genera tantos mensajes de lista de distribución como sean necesarios para dar cabida a los diferentes valores de prioridad y persistencia.

4. Una inserción en una lista de distribución puede dar lugar a:

- Un único mensaje de lista de distribución, o
- Varios mensajes de listas de distribución más pequeños, o
- Una mezcla de mensajes de listas de distribución y mensajes normales, o
- Solo mensajes normales.

Lo que ocurra de lo anterior depende de si:

- Los destinos de la lista son locales, remotos o una combinación de ambos.
- Los destinos tienen la misma prioridad y persistencia de mensajes.
- Las colas de transmisión pueden contener mensajes de listas de distribución.
- Las longitudes máximas de los mensajes de las colas de transmisión son lo suficientemente grandes como para dar cabida al mensaje en forma de lista de distribución.

Sin embargo, independientemente de cuál de las situaciones anteriores se produzca, cada mensaje *físico* resultante (es decir, cada mensaje normal o mensaje de lista de distribución resultante de la inserción) cuenta como *un* solo mensaje cuando:

- Comprobación de si la aplicación ha superado el número máximo permitido de mensajes en una unidad de trabajo (véase el atributo del **MaxUncommittedMsgs** gestor de colas).
- Comprobación de si se cumplen las condiciones de activación.
- Incrementar la profundidad de las colas y comprobar si se superaría la profundidad máxima de las colas.

5. Cualquier cambio en las definiciones de la cola que hubiera provocado que un identificador dejara de ser válido si las colas se hubieran abierto individualmente (por ejemplo, un cambio en la ruta de resolución), no provoca que el identificador de la lista de distribución deje de ser válido. Sin embargo, sí provoca un error en esa cola concreta cuando se utiliza el identificador de la lista de distribución en una llamada MQPUT posterior.

Cabeceras

Si se coloca un mensaje con una o más IBM MQ estructuras de encabezado al principio de los datos del mensaje de la aplicación, el gestor de colas realiza ciertas comprobaciones en las estructuras de encabezado para verificar que son válidas. Si el gestor de colas detecta un error, la llamada falla con un código de motivo adecuado. Las comprobaciones realizadas varían en función de las estructuras particulares presentes:

- Las comprobaciones solo se realizan si se utiliza un MQMD de version-2 o posterior en la llamada MQPUT o MQPUT1. No se realizan comprobaciones si se utiliza un MQMD version-1, incluso si hay un MQMDE al principio de los datos del mensaje.
- Las estructuras que no son compatibles con el gestor de colas local y las estructuras que siguen al primer MQDLH del mensaje no se validan.
- Las estructuras MQDH y MQMDE están totalmente validadas por el gestor de colas.
- Otras estructuras son validadas parcialmente por el gestor de colas (no se comprueban todos los campos).

Las comprobaciones generales realizadas por el gestor de colas incluyen lo siguiente:

- El **StrucId** campo debe ser válido.
- El **Version** campo debe ser válido.
- El **StrucLength** campo debe especificar un valor lo suficientemente grande como para incluir la estructura más cualquier dato de longitud variable que forme parte de la estructura.
- El **CodedCharSetId** campo no debe ser cero ni un valor negativo que no sea válido (MQCCSI_DEFAULT, MQCCSI_EMBEDDED, MQCCSI_Q_MGR y MQCCSI_UNDEFINED no son válidos en la mayoría de las estructuras IBM MQ de encabezado).
- El **BufferLength** parámetro de la llamada debe especificar un valor lo suficientemente grande como para incluir la estructura (la estructura no debe extenderse más allá del final del mensaje).

Además de las comprobaciones generales de las estructuras, deben cumplirse las siguientes condiciones:

- La suma de las longitudes de las estructuras en un mensaje PCF debe ser igual a la longitud especificada por el **BufferLength** parámetro en la llamada MQPUT o MQPUT1. Un mensaje PCF es un mensaje que tiene un nombre de formato MQFMT_ADMIN, MQFMT_EVENT o MQFMT_PCF.
- Una IBM MQ estructura no debe truncarse, excepto en las siguientes situaciones en las que se permiten las estructuras truncadas:
 - Mensajes que son mensajes de informe.
 - Mensajes PCF.
 - Mensajes que contienen una estructura MQDLH. (Las estructuras que siguen al primer MQDLH pueden truncarse; las estructuras que preceden al MQDLH no pueden truncarse)
- Una IBM MQ estructura no debe dividirse en dos o más segmentos; la estructura debe estar contenida íntegramente en un solo segmento.

Almacenamiento intermedio

Para el lenguaje de programación Visual Basic, se aplican los puntos siguientes:

- Si el tamaño del **Buffer** parámetro es menor que la longitud especificada por el **BufferLength** parámetro, la llamada falla con el código de motivo MQRC_BUFFER_LENGTH_ERROR.
- El **Buffer** parámetro se declara como de tipo String. Si los datos que se van a colocar en la cola no son de tipo String, utilice el llamada MQPUTAny en lugar de MQPUT.

La llamada MQPUTAny tiene los mismos parámetros que la llamada MQPUT, excepto que el **Buffer** parámetro se declara como de tipo Any, lo que permite colocar cualquier tipo de datos en la cola. Sin embargo, esto significa que Buffer no se puede comprobar para garantizar que tenga al menos BufferLength bytes de tamaño.

Invocación en C

```
MQPUT (Hconn, Hobj, &MsgDesc, &PutMsgOpts, BufferLength, Buffer,
      &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQHCONN  Hconn;          /* Connection handle */
MQHOBJ   Hobj;           /* Object handle */
MQMD     MsgDesc;        /* Message descriptor */
MQPMO    PutMsgOpts;     /* Options that control the action of MQPUT */
MQLONG   BufferLength;   /* Length of the message in Buffer */
MQBYTE   Buffer[n];      /* Message data */
MQLONG   CompCode;       /* Completion code */
MQLONG   Reason;         /* Reason code qualifying CompCode */
```

Invocación en COBOL

```
CALL 'MQPUT' USING HCONN, HOBJ, MSGDESC, PUTMSGOPTS, BUFFERLENGTH,
                  BUFFER, COMPCODE, REASON.
```


Declare los parámetros como se indica a continuación:

```
** Connection handle
01 HCONN          PIC S9(9) BINARY.
** Object handle
01 HOBJ          PIC S9(9) BINARY.
** Message descriptor
01 MSGDESC.
COPY CMQMDV.
** Options that control the action of MQPUT
01 PUTMSGOPTS.
COPY CMQPMOV.
** Length of the message in BUFFER
01 BUFFERLENGTH PIC S9(9) BINARY.
** Message data
01 BUFFER        PIC X(n).
** Completion code
01 COMPCODE      PIC S9(9) BINARY.
** Reason code qualifying COMPCODE
01 REASON        PIC S9(9) BINARY.
```

Invocación en PL/I

```
call MQPUT (Hconn, Hobj, MsgDesc, PutMsgOpts, BufferLength, Buffer,
            CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```
dcl Hconn          fixed bin(31); /* Connection handle */
dcl Hobj           fixed bin(31); /* Object handle */
dcl MsgDesc        like MQMD;    /* Message descriptor */
dcl PutMsgOpts     like MQPMO;    /* Options that control the action of
                                MQPUT */
dcl BufferLength    fixed bin(31); /* Length of the message in Buffer */
dcl Buffer          char(n);      /* Message data */
dcl CompCode       fixed bin(31); /* Completion code */
dcl Reason         fixed bin(31); /* Reason code qualifying CompCode */
```

Invocación en ensamblador de alto nivel

```
CALL MQPUT, (HCONN,HOBJ,MSGDESC,PUTMSGOPTS,BUFFERLENGTH, X
            BUFFER,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

HCONN	DS	F	Connection handle
HOBJ	DS	F	Object handle
MSGDESC	CMQMDA	,	Message descriptor
PUTMSGOPTS	CMQPMOA	,	Options that control the action of MQPUT
BUFFERLENGTH	DS	F	Length of the message in BUFFER
BUFFER	DS	CL(n)	Message data
COMPCODE	DS	F	Completion code
REASON	DS	F	Reason code qualifying COMPCODE

Invocación en Visual Basic

Windows

```
MQPUT Hconn, Hobj, MsgDesc, PutMsgOpts, BufferLength, Buffer, CompCode,
      Reason
```

Declare los parámetros como se indica a continuación:

Dim Hconn	As Long	'Connection handle'
Dim Hobj	As Long	'Object handle'
Dim MsgDesc	As MQMD	'Message descriptor'
Dim PutMsgOpts	As MQPMO	'Options that control the action of MQPUT'
Dim BufferLength	As Long	'Length of the message in Buffer'
Dim Buffer	As String	'Message data'

Dim CompCode	As Long	'Completion code'
Dim Reason	As Long	'Reason code qualifying CompCode'

MQPUT1 - Pon un mensaje

La llamada `MQPUT1` coloca un mensaje en una cola, una lista de distribución o un tema.

No es necesario que la cola, la lista de distribución o el tema estén abiertos.

Sintaxis

MQPUT1 (*Hconn*, *ObjDesc*, *MsgDesc*, *PutMsgOpts*, *BufferLength*, *Buffer*, *CompCode*, *Reason*)

Parámetros

Hconn

Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* fue devuelto por una llamada anterior a MQCONN o MQCONNEX.

 En z/OS para CICS aplicaciones, se puede omitir la llamada MQCONN y especificar el siguiente valor para *Hconn*:

MQHC_DEF_HCONN

Manejador de conexión predeterminado.

ObjDesc

Tipo: MQOD - entrada/salida

Esta es una estructura que identifica la cola a la que se añade el mensaje o el tema en el que se publica el mensaje. Consulte [“MQOD - Descriptor de objeto”](#) en la página 454 para obtener detalles.

Si la estructura es una cola, el usuario debe estar autorizado para abrir la cola para la salida. La cola no debe ser una cola modelo.

MsgDesc

Tipo: MQMD - entrada/salida

Esta estructura describe los atributos del mensaje que se envía y recibe información de respuesta una vez completada la solicitud put. Consulte [“MQMD - Descriptor de mensaje”](#) en la página 389 para obtener detalles.

Si la aplicación proporciona un MQMD version-1, los datos del mensaje pueden ir precedidos de una estructura MQMDE para especificar los valores de los campos que existen en el MQMD version-2 pero no en el version-1. Establezca el `Format` campo en MQMD en MQFMT_MD_EXTENSION para indicar que hay un MQMDE presente. Consulte [“MQMDE - Extensión del descriptor de mensajes”](#) en la página 443 para obtener más detalles.

La aplicación no necesita proporcionar una estructura MQMD si se suministra un identificador de mensaje válido en el `MsgHandle` campo de la estructura MQGMO o en los campos `OriginalMsgHandle` `NewMsgHandle` o de la estructura MQPMO. Si no se proporciona nada en uno de estos campos, el descriptor del mensaje se toma del descriptor asociado con los identificadores de mensaje.

PutMsgOpts

Tipo: MQPMO - entrada/salida

Consulte [“MQPMO - Opciones de mensaje de envío”](#) en la página 477 para obtener los detalles.

BufferLength

Tipo: MQLONG - entrada

La longitud del mensaje en `Buffer`. El valor cero es válido e indica que el mensaje no contiene datos de aplicación. El límite superior depende de varios factores; véase “MQPUT - Enviar mensaje” en la página 752 para obtener una descripción del **BufferLength** parámetro.

Almacenamiento intermedio

Tipo: `MQBYTEExBufferLength` - entrada

Se trata de un búfer que contiene los datos del mensaje de la aplicación que se va a enviar. Alinee el almacenamiento intermedio en un límite adecuado según la naturaleza de los datos del mensaje. La alineación de 4 bytes es adecuada para la mayoría de los mensajes (incluidos los mensajes que contienen IBM MQ estructuras de encabezado), pero algunos mensajes pueden requerir una alineación más estricta. Por ejemplo, un mensaje que contenga un entero binario de 64 bits podría requerir una alineación de 8 bytes.

Si `Buffer` contiene datos numéricos o caracteres, establezca los campos `CodedCharSetId` `Encoding` y del **MsgDesc** parámetro con los valores adecuados para los datos; esto permite al destinatario del mensaje convertir los datos (si es necesario) al juego de caracteres y la codificación utilizados por el destinatario.

Nota: Todos los demás parámetros de la llamada `MQPUT1` deben estar en el juego de caracteres y la codificación del gestor de colas local (especificados por el **CodedCharSetId** atributo `queue manager` y `MQENC_NATIVE`).

En el lenguaje de programación C, el parámetro se declara como un puntero a void; se puede especificar como parámetro la dirección de cualquier tipo de datos.

Si el **BufferLength** parámetro es cero, no `Buffer` se hace referencia a; en este caso, la dirección del parámetro pasada por programas escritos en C o en ensambladore System/390 e puede ser nula.

CompCode

Tipo: `MQLONG` - salida

El código de terminación; es uno de los siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_WARNING

Aviso (finalización parcial).

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: `MQLONG` - salida

El código de motivo que califica `CompCode`.

Si `CompCode` es `MQCC_OK`:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si `CompCode` es `MQCC_WARNING`:

MQRC_MÚLTIPLES_RAZONES

(2136, X'858') Se han devuelto varios códigos de motivo.

MQRC_INCOMPLETE_GROUP

(2241, X'8C1') El grupo de mensajes no está completo.

MQRC_INCOMPLETE_MSG

(2242, X'8C2') El mensaje lógico no está completo.

MQRC_PRIORIDAD_SUPERA_MÁXIMO

(2049, X'801') La prioridad del mensaje excede el valor máximo admitido.

MQRC_OPCIÓN_DE_INFORME_DESCONOCIDA

(2104, X'838') Opciones de informe en el descriptor de mensajes no reconocidas.

Si CompCode es MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'89C') El adaptador no está disponible.

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

MQRC_ALIAS_BASE_Q_TIPO_ERROR

(2001, X'7D1') La cola base alias no es un tipo válido.

MQRC_API_EXIT_ERROR

(2374, X'946') La salida de la API ha fallado.

MQRC_API_EXIT_LOAD_ERROR

(2183, X'887') No se ha podido cargar la salida de la API.

MQRC_ASID_MISMATCH

(2157, X'86D') Los ASID primario e inicial varían.

MQRC_BACKED_OUT

(2003, X'7D3') Unidad de trabajo restituida.

MQRC_BUFFER_ERROR

(2004, X'7D4') El parámetro almacenamiento intermedio no es válido.

MQRC_BUFFER_LENGTH_ERROR

(2005, X'7D5') El parámetro de longitud de almacenamiento intermedio no es válido.

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') La llamada MQI ha entrado antes de completar la llamada anterior.

MQRC_CF_NO_DISPONIBLE

(2345, X'929') Instalación de acoplamiento no disponible.

MQRC_CF_STRUC_AUTH_FAILED

(2348, X'92C') Error en la comprobación de la autorización de la estructura de la instalación de acoplamiento.

MQRC_CF_STRUC_ERROR

(2349, X'92D') Estructura de la instalación de acoplamiento no válida.

MQRC_CF_STRUC_FAILED

(2373, X'945') La estructura de recurso de acoplamiento ha fallado.

MQRC_CF_STRUC_IN_USE

(2346, X'92A') La estructura de recurso de acoplamiento se está utilizando.

MQRC_CF_STRUC_LIST_HDR_IN_USE

(2347, X'92B') La cabecera de lista de la estructura de recurso de acoplamiento se está utilizando.

MQRC_CFGR_ERROR

(2416, X'970') La estructura del parámetro del grupo PCF MQCFGR en los datos del mensaje no es válida.

MQRC_CFH_ERROR

(2235, X'8BB') Estructura de encabezado PCF no válida.

MQRC_CFIF_ERROR

(2414, X'96E') La estructura del parámetro del filtro entero PCF en los datos del mensaje no es válida.

MQRC_CFIL_ERROR

(2236, X'8BC') Estructura de parámetro de lista de enteros PCF o estructura de parámetro de lista de enteros PCIF*64 no válida.

MQRC_CFIN_ERROR

(2237, X'8BD') Estructura de parámetro entero PCF o estructura de parámetro entero PCIF*64 no válida.

MQRC_CFSF_ERROR

(2415, X'96F') La estructura del parámetro del filtro de cadena PCF en los datos del mensaje no es válida.

MQRC_CFSL_ERROR

(2238, X'8BE') Estructura del parámetro de la lista de cadenas PCF no válida.

MQRC_CFST_ERROR

(2239, X'8BF') Estructura del parámetro de cadena PCF no válida.

MQRC_CICS_WAIT_FAILED

(2140, X'85C') Solicitud de espera rechazada por CICS.

ERROR DE SALIDA DEL CLÚSTER MQRC

(2266, X'8DA') Error al salir de la carga de trabajo del clúster.

MQRC_CLUSTER_RESOLUTION_ERROR

(2189, X'88D') La resolución de nombres de clúster ha fallado.

ERROR_RECURSO_CLÚSTER_MQRC

(2269, X'8DD') Error de recursos del clúster.

MQRC_CÓDIGO_NO_VÁLIDO_PARA_XCF_Q

(2106, X'83A') La opción de informe COD no es válida para la cola XCF.

MQRC_CONNECTION_BROKEN

(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

MQRC_CONNECTION_NOT_AUTHORIZED

(2217, X'8A9') No tiene autorización para la conexión.

MQRC_CONNECTION QUIESCING

(2202, X'89A') Conexión en fase de inmovilización.

MQRC_CONNECTION_STOPPING

(2203, X'89B') La conexión está concluyendo.

ERROR_DE CONTENIDO_MQRC

2554 (X'09FA') No se pudo analizar el contenido del mensaje para determinar si el mensaje se puede entregar a un suscriptor con un selector de mensajes ampliado.

MQRC_CONTEXTO_ERROR_MANEJO

(2097, X'831') El identificador de cola al que se hace referencia no guarda el contexto.

MQRC_CONTEXTO_NO_DISPONIBLE

(2098, X'832') No hay contexto disponible para el identificador de cola al que se hace referencia.

MQRC_DATA_LENGTH_ERROR

(2010, X'7DA') El parámetro longitud de datos no es válido.

MQRC_DB2_NOT_AVAILABLE

(2342, X'926') Db2 subsistema no disponible.

MQRC_DEF_XMIT_Q_TYPE_ERROR

(2198, X'896') Cola de transmisión predeterminada no local.

MQRC_DEF_XMIT_Q_USO_ERROR

(2199, X'897') Error de uso de la cola de transmisión predeterminada.

MQRC_DH_ERROR

(2135, X'857') Estructura del encabezado de distribución no válida.

MQRC_DLH_ERROR

(2141, X'85D') Estructura del encabezado de carta muerta no válida.

MQRC_EPH_ERROR

(2420, X'974') Estructura PCF incrustada no válida.

ERROR_CADUCIDAD_MQRC

(2013, X'7DD') Fecha de caducidad no válida.

MQRC_ERROR_DE_RESPUESTA

(2014, X'7DE') Código de retroalimentación no válido.

MQRC_GLOBAL_UOW_CONFLICT

(2351, X'92F') Conflicto de unidades de trabajo global.

MQRC_GROUP_ID_ERROR

(2258, X'8D2') Identificador de grupo no válido.

MQRC_HANDLE_IN_USE_FOR_UOW

(2353, X'931') Descriptor de contexto que se utiliza para la unidad de trabajo global.

MQRC_HANDLE_NO_DISPONIBLE

(2017, X'7E1') No hay más asas disponibles.

MQRC_HCONN_ERROR

(2018, X'7E2') El manejador de conexión no es válido.

MQRC_ERROR_ENCABEZADO

(2142, X'85E') IBM MQ Estructura del encabezado no válida.

MQRC_IIH_ERROR

(2148, X'864') Estructura de IMS encabezado de información no válida.

MQRC_LOCAL_UOW_CONFLICT

(2352, X'930') La unidad de trabajo global entra en conflicto con la unidad de trabajo local.

MQRC_MD_ERROR

(2026, X'7EA') El descriptor de mensaje no es válido.

MQRC_MDE_ERROR

(2248, X'8C8') Extensión del descriptor de mensaje no válida.

MQRC_RESPUESTA_FALTANTE_A_P

(2027, X'7EB') Falta la cola de respuestas.

MQRC_FALTA_WIH

(2332, X'91C') Los datos del mensaje no comienzan con MQWIH.

MQRC_MSG_FLAGS_ERROR

(2249, X'8C9') Indicadores de mensaje no válidos.

MQRC_MSG_SEQ_NUMBER_ERROR

(2250, X'8CA') El número de secuencia de mensaje no es válido.

MQRC_MSG_DEMASIADO_GRANDE_PARA_Q

(2030, X'7EE') La longitud del mensaje supera el máximo permitido para la cola.

MQRC_MSG_DEMASIADO_GRANDE_PARA_Q_MGR

(2031, X'7EF') La longitud del mensaje supera el máximo permitido por el gestor de colas.

MQRC_MSG_TYPE_ERROR

(2029, X'7ED') Tipo de mensaje no válido en el descriptor de mensajes.

MQRC_MÚLTIPLES_RAZONES

(2136, X'858') Se han devuelto varios códigos de motivo.

MQRC_NO_DESTINOS_DISPONIBLES

(2270, X'8DE') No hay colas de destino disponibles.

MQRC_NOT_AUTHORIZED

(2035, X'7F3') Acceso no autorizado.

MQRC_OBJECT_DAMAGED

(2101, X'835') El objeto se ha dañado.

MQRC_OBJECT_IN_USE

(2042, X'7FA') Objeto ya abierto con opciones conflictivas.

MQRC_OBJETO_NIVEL_INCOMPATIBLE

(2360, X'938') Nivel de objeto no compatible.

MQRC_OBJETO_NOMBRE_ERROR

(2152, X'868') Nombre del objeto no válido.

MQRC_OBJETO_NO_ÚNICO

(2343, X'927') Objeto no único.

MQRC_OBJETO_Q_MGR_NOMBRE_ERROR

(2153, X'869') Nombre del gestor de colas de objetos no válido.

ERROR_DE_REGISTROS_DE_OBJETOS_MQRC

(2155, X'86B') Registros de objetos no válidos.

MQRC_TIPO_OBJETO_ERROR

(2043, X'7FB') Tipo de objeto no válido.

MQRC_OD_ERROR

(2044, X'7FC') Estructura del descriptor de objeto no válida.

ERROR_DESVIACIÓN_MQRC

(2251, X'8CB') Desplazamiento del segmento de mensaje no válido.

MQRC_OPTIONS_ERROR

(2046, X'7FE') Las opciones no son válidas o no son coherentes.

ERROR_DE_LONGITUD_ORIGINAL_MQRC

(2252, X'8CC') Longitud original no válida.

MQRC_PAGESET_ERROR

(2193, X'891') Error al acceder al conjunto de datos del conjunto de páginas.

MQRC_PÁGINAS_COMPLETAS

(2192, X'890') El soporte de almacenamiento externo está lleno.

MQRC_PCF_ERROR

(2149, X'865') Estructuras PCF no válidas.

MQRC_PERSISTENCIA_ERROR

(2047, X'7FF') Persistencia no válida.

MQRC_PERSISTENTE_NO_PERMITIDO

(2048, X'800') La cola no admite mensajes persistentes.

MQRC_PMO_ERROR

(2173, X'87D') Estructura de opciones de mensaje de envío no válida.

MQRC_PMO_REGISTRO_FLAGS_ERROR

(2158, X'86E') Los indicadores de registro de mensajes no son válidos.

MQRC_ERROR_PRIORIDAD

(2050, X'802') Prioridad del mensaje no válida.

MQRC_FALLO_PUBLICACIÓN

(2502, X'9C6') La publicación no se ha entregado a ninguno de los suscriptores.

MQRC_PUT_INHIBITED

(2051, X'803') Se inhibieron las llamadas de puesta en cola.

MQRC_PUT_MSG_RECORDS_ERROR

(2159, X'86F') Los registros de mensajes no son válidos.

MQRC_Q_DELETED

(2052, X'804') La cola se ha suprimido.

MQRC_Q_COMPLETO

(2053, X'805') La cola ya contiene el número máximo de mensajes.

MQRC_Q_MGR_NAME_ERROR

(2058, X'80A') El nombre del gestor de colas no es válido o no se conoce.

MQRC_Q_MGR_NOT_AVAILABLE

(2059, X'80B') El gestor de colas no está disponible para efectuar la conexión.

MQRC_Q_MGR QUIESCING

(2161, X'871') El gestor de colas se está desactivando temporalmente.

MQRC_Q_MGR_STOPPING

(2162, X'872') El gestor de colas está concluyendo.

MQRC_Q ESPACIO_NO_DISPONIBLE

(2056, X'808') No hay espacio disponible en el disco para la cola.

MQRC_Q_TIPO_ERROR

(2057, X'809') Tipo de cola no válido.

MQRC_RECS_PRESENT_ERROR

(2154, X'86A') Número de registros presentes no válido.

MQRC_REMOTO_Q_NOMBRE_ERROR

(2184, X'888') Nombre de cola remota no válido.

MQRC_REPORT_OPTIONS_ERROR

(2061, X'80D') Las opciones de informe en el descriptor de mensajes no son válidas.

MQRC_RESOURCE_PROBLEM

(2102, X'836') No hay disponibles suficientes recursos del sistema.

ERROR_EN_LOS_REGISTROS_DE_RESPUESTA_MQRC

(2156, X'86C') Registros de respuesta no válidos.

MQRC_RFH_ERROR

(2334, X'91E') Estructura MQRFH o MQRFH2 no válida.

MQRC_RMH_ERROR

(2220, X'8AC') Estructura del encabezado del mensaje de referencia no válida.

MQRC_ERROR_DE_SEGURIDAD

(2063, X'80F') Se ha producido un error de seguridad.

MQRC_LONGITUD_DEL_SEGMENTO_CERO

(2253, X'8CD') La longitud de los datos en el segmento del mensaje es cero.

MQRC_SELECTION_NOT_AVAILABLE

2551 (X'09F7') Existe un posible suscriptor para la publicación, pero el gestor de colas no puede comprobar si debe enviar la publicación al suscriptor.

MQRC_DETENIDO_POR_SALIDA_DEL_CLÚSTER

(2188, X'88C') Llamada rechazada por salida de carga de trabajo del clúster.

MQRC_CLASE_ALMACENAMIENTO_ERROR

(2105, X'839') Error de clase de almacenamiento.

MQRC_STORAGE_MEDIUM_FULL

(2192, X'890') El soporte de almacenamiento externo está lleno.

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_SUPPRESSED_BY_EXIT

(2109, X'83D') El programa de salida ha suprimido la llamada.

MQRC_SYNCPOINT_LIMIT_REACHED

(2024, X'7E8') No pueden manejarse más mensajes dentro de la unidad de trabajo actual.

MQRC_SYNCPOINT_NOT_AVAILABLE

(2072, X'818') No hay soporte para Syncpoint.

MQRC_TM_ERROR

(2265, X'8D9') Estructura del mensaje de activación no válida.

MQRC_TMC_ERROR

(2191, X'88F') Estructura del mensaje de activación de caracteres no válida.

MQRC_UNEXPECTED_ERROR

(2195, X'893') Se ha producido un error inesperado.

MQRC_ALIAS_DESCONOCIDO_BASE_Q

(2082, X'822') Cola base de alias desconocida.

MQRC_DESCONOCIDO_DEF_XMIT_Q

(2197, X'895') Cola de transmisión predeterminada desconocida.

MQRC_UNKNOWN_OBJECT_NAME

(2085, X'825') Nombre del objeto desconocido.

MQRC_OBJETO_DESCONOCIDO_Q_MGR

(2086, X'826') Administrador de colas de objetos desconocido.

MQRC_DESCONOCIDO_REMOTO_Q_MGR

(2087, X'827') Administrador de colas remotas desconocido.

MQRC_DESCONOCIDO_XMIT_Q

(2196, X'894') Cola de transmisión desconocida.

MQRC_UOW_ENLISTMENT_ERROR

(2354, X'932') Ha fallado el listado en la unidad de trabajo global.

MQRC_UOW_MIX_NOT_SUPPORTED

(2355, X'933') No se soporta la mezcla de llamadas de unidad de trabajo.

MQRC_UOW_NOT_AVAILABLE

(2255, X'8CF') La unidad de trabajo no está disponible para ser utilizada por el gestor de colas.

MQRC_WIH_ERROR

(2333, X'91D') Estructura MQWIH no válida.

MQRC_NIVEL_CF_INCORRECTO

(2366, X'93E') La estructura de la instalación de acoplamiento tiene un nivel incorrecto.

MQRC_WRONG_MD_VERSION

(2257, X'8D1') La versión del MQMD suministrado es incorrecta.

MQRC_XMIT_Q_TYPE_ERROR

(2091, X'82B') Cola de transmisión no local.

MQRC_XMIT_Q_ERROR_DE_USO

(2092, X'82C') Cola de transmisión con uso incorrecto.

MQRC_XQH_ERROR

(2260, X'8D4') Estructura de encabezado de la cola de transmisión no válida.

Para obtener información detallada sobre estos códigos, consulte [Mensajes y códigos de motivo](#).

Notas de uso

1. Las llamadas MQPUT y MQPUT1 pueden utilizarse para colocar mensajes en una cola; la llamada que se utilice dependerá de las circunstancias:
 - Utilice la llamada MQPUT para colocar varios mensajes en la *misma* cola.
Primero se emite una llamada MQOPEN que especifica la opción MQOO_OUTPUT, seguida de una o más solicitudes MQPUT para añadir mensajes a la cola; finalmente, la cola se cierra con una llamada MQCLOSE. Esto ofrece un mejor rendimiento que el uso repetido de la llamada MQPUT1.
 - Utilice la llamada `MQPUT1` para colocar *un* solo mensaje en una cola.
Esta llamada encapsula las llamadas MQOPEN, MQPUT y MQCLOSE en una sola llamada, minimizando el número de llamadas que deben emitirse.
2. Si una aplicación coloca una secuencia de mensajes en la misma cola sin utilizar grupos de mensajes, el orden de dichos mensajes se conserva si se cumplen determinadas condiciones. Sin embargo, en la mayoría de los entornos, la llamada `MQPUT1` no cumple estas condiciones, por lo que no conserva el orden de los mensajes. En estos entornos se debe utilizar la llamada MQPUT. Consulte [Notas de uso de MQPUT](#) para obtener detalles.
3. La llamada MQPUT1 se puede utilizar para enviar mensajes a listas de distribución. Para obtener información general al respecto, consulte las notas de uso de las llamadas MQOPEN y MQPUT.

Las listas de distribución son compatibles con los siguientes entornos:

-  AIX
-  IBM i
-  Linux

-  Windows

y para IBM MQ los clientes conectados a estos sistemas.

Las siguientes diferencias se aplican al utilizar la llamada MQPUT1 :

- a. Si la aplicación proporciona registros de respuesta MQRR, estos deben proporcionarse utilizando la estructura MQOD; no pueden proporcionarse utilizando la estructura MQPMO.
- b. El código de motivo MQRC_OPEN_FAILED nunca es devuelto por MQPUT1 en los registros de respuesta; si una cola no se puede abrir, el registro de respuesta para esa cola contiene el código de motivo resultante de la operación de apertura.

Si una operación abierta para una cola tiene éxito con un código de finalización MQCC_WARNING, el código de finalización y el código de motivo del registro de respuesta para esa cola se sustituyen por los códigos de finalización y motivo resultantes de la operación de inserción.

Al igual que con las llamadas MQOPEN y MQPUT, el gestor de colas establece los registros de respuesta (si se proporcionan) solo cuando el resultado de la llamada no es el mismo para todas las colas de la lista de distribución; esto se indica mediante la finalización de la llamada con el código de motivo MQRC_MULTIPLE_REASONS.

4. Si se utiliza la llamada `MQPUT1` para colocar un mensaje en una cola de clúster, la llamada se comporta como si se hubiera especificado `MQOO\_BIND\_NOT\_FIXED` en la llamada `MQOPEN`.
5. Si se coloca un mensaje con una o más IBM MQ estructuras de encabezado al principio de los datos del mensaje de la aplicación, el gestor de colas realiza ciertas comprobaciones en las estructuras de encabezado para verificar que son válidas. Para obtener más información al respecto, consulte las notas de uso de la llamada MQPUT.
6. Si se produce más de una de las situaciones de advertencia (véase el **CompCode** parámetro), el código de motivo devuelto es el primero de la siguiente lista que sea aplicable:
 - a. MQRC_MÚLTIPLES_RAZONES
 - b. MQRC_INCOMPLETE_MSG
 - c. MQRC_INCOMPLETE_GROUP
 - d. MQRC_PRIORITY_EXCEEDS_MAXIMUM o MQRC_UNKNOWN_REPORT_OPTION
7. Para el lenguaje de programación Visual Basic, se aplican los puntos siguientes:
 - Si el tamaño del **Buffer** parámetro es menor que la longitud especificada por el **BufferLength** parámetro, la llamada falla con el código de motivo MQRC_BUFFER_LENGTH_ERROR.
 - El **Buffer** parámetro se declara como de tipo String. Si los datos que se van a colocar en la cola no son de tipo String, utilice elMQPUT1Any llamar en lugar de MQPUT1.

La llamada `MQPUT1Any` tiene los mismos parámetros que la llamada `MQPUT1`, excepto que el **Buffer** parámetro `` se declara como de tipo `Any`, lo que permite colocar cualquier tipo de datos en la cola. Sin embargo, esto significa que **Buffer** no se puede comprobar para garantizar que tenga al menos **BufferLength** bytes de tamaño.

8. Cuando se emite una llamada MQPUT1 con MQPMO_SYNCPOINT, el comportamiento predeterminado cambia, de modo que la operación put se completa de forma asíncrona. Esto puede crear cambio de comportamiento en algunas aplicaciones que se basan en la devolución de determinados campos de las estructuras MQOD y MQMD y que ahora contienen valores no definidos. Una aplicación puede especificar MQPMO_SYNC_RESPONSE para garantizar que la operación de inserción se realice de forma sincrónica y que se completen todos los valores de campo adecuados.

Invocación en C

```
MQPUT1 (Hconn, &ObjDesc, &MsgDesc, &PutMsgOpts,
        BufferLength, Buffer, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```

MQHCONN  Hconn;          /* Connection handle */
MQOD      ObjDesc;       /* Object descriptor */
MQMD      MsgDesc;       /* Message descriptor */
MQPMO     PutMsgOpts;    /* Options that control the action of MQPUT1 */
MQLONG    BufferLength;   /* Length of the message in Buffer */
MQBYTE    Buffer[n];      /* Message data */
MQLONG    CompCode;      /* Completion code */
MQLONG    Reason;        /* Reason code qualifying CompCode */

```

Invocación en COBOL

```

CALL 'MQPUT1' USING HCONN, OBJDESC, MSGDESC, PUTMSGOPTS,
                   BUFFERLENGTH, BUFFER, COMPCODE, REASON.

```

Declare los parámetros como se indica a continuación:

```

** Connection handle
01 HCONN          PIC S9(9) BINARY.
** Object descriptor
01 OBJDESC.
   COPY CMQODV.
** Message descriptor
01 MSGDESC.
   COPY CMQMDV.
** Options that control the action of MQPUT1
01 PUTMSGOPTS.
   COPY CMQPMOV.
** Length of the message in BUFFER
01 BUFFERLENGTH  PIC S9(9) BINARY.
** Message data
01 BUFFER        PIC X(n).
** Completion code
01 COMPCODE      PIC S9(9) BINARY.
** Reason code qualifying COMPCODE
01 REASON        PIC S9(9) BINARY.

```

Invocación en PL/I

```

call MQPUT1 (Hconn, ObjDesc, MsgDesc, PutMsgOpts, BufferLength, Buffer,
             CompCode, Reason);

```

Declare los parámetros como se indica a continuación:

```

dcl Hconn          fixed bin(31); /* Connection handle */
dcl ObjDesc        like MQOD;     /* Object descriptor */
dcl MsgDesc        like MQMD;     /* Message descriptor */
dcl PutMsgOpts     like MQPMO;    /* Options that control the action of
                                   MQPUT1 */
dcl BufferLength    fixed bin(31); /* Length of the message in Buffer */
dcl Buffer          char(n);       /* Message data */
dcl CompCode       fixed bin(31); /* Completion code */
dcl Reason         fixed bin(31); /* Reason code qualifying CompCode */

```

Invocación en ensamblador de alto nivel

```

CALL MQPUT1,(HCONN,OBJDESC,MSGDESC,PUTMSGOPTS,BUFFERLENGTH,  X
             BUFFER,COMPCODE,REASON)

```

Declare los parámetros como se indica a continuación:

```

HCONN      DS      F      Connection handle
OBJDESC    CMQODA   ,      Object descriptor
MSGDESC    CMQMDA   ,      Message descriptor
PUTMSGOPTS CMQPMOA  ,      Options that control the action of MQPUT1
BUFFERLENGTH DS     F      Length of the message in BUFFER
BUFFER     DS      CL(n)  Message data
COMPCODE   DS      F      Completion code
REASON     DS      F      Reason code qualifying COMPCODE

```

Invocación en Visual Basic

Windows

```
MQPUT1 Hconn, ObjDesc, MsgDesc, PutMsgOpts, BufferLength, Buffer,  
CompCode, Reason
```

Declare los parámetros como se indica a continuación:

```
Dim Hconn      As Long   'Connection handle'  
Dim ObjDesc    As MQOD   'Object descriptor'  
Dim MsgDesc    As MQMD   'Message descriptor'  
Dim PutMsgOpts As MQPMO   'Options that control the action of MQPUT1'  
Dim BufferLength As Long  'Length of the message in Buffer'  
Dim Buffer      As String 'Message data'  
Dim CompCode   As Long   'Completion code'  
Dim Reason     As Long   'Reason code qualifying CompCode'
```

MQSET - Establecer atributos de objeto

Utilice la llamada MQSET para cambiar los atributos de un objeto representado por un identificador. El objeto debe ser una cola.

Sintaxis

MQSET (*Hconn, Hobj, SelectorCount, Selectors, IntAttrCount, IntAttrs, CharAttrLength, CharAttrs, Compcode, Reason*)

Parámetros

Hconn

Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de Hconn fue devuelto por una llamada anterior a MQCONN o MQCONNX.

 En z/OS para CICS aplicaciones, se puede omitir la llamada MQCONN y especificar el siguiente valor para Hconn:

MQHC_DEF_HCONN

Manejador de conexión predeterminado.

Hobj

Tipo: MQHOBJ - entrada

Este identificador representa el objeto de cola con los atributos que se van a establecer. El identificador fue devuelto por una llamada MQOPEN anterior que especificaba la opción MQOO_SET.

SelectorCount

Tipo: MQLONG - entrada

Este es el recuento de selectores que se proporcionan en la Selectors matriz. Es el número de atributos que se van a establecer. Cero es un valor válido. El número máximo permitido es 256.

Selectores

Tipo: MQLONGxSelectorCount - entrada

Se trata de una matriz de **SelectorCount** selectores de atributos; cada selector identifica un atributo (entero o carácter) con un valor que se va a establecer.

Cada selector debe ser válido para el tipo de cola que Hobj representa. Solo se permiten determinados valores MQIA_* y MQCA_*, tal y como se indica más adelante.

Los selectores se pueden especificar en cualquier orden. Los valores de los atributos que corresponden a selectores de atributos enteros (selectores MQIA_*) deben especificarse en IntAttrs en el mismo orden en que aparecen estos selectores en Selectors. Los valores de los

atributos que corresponden a los selectores de atributos de caracteres (selectores MQCA_*) deben especificarse en CharAttr en el mismo orden en que aparecen dichos selectores. Los selectores MQIA_* pueden intercalarse con los selectores MQCA_*; solo es importante el orden relativo dentro de cada tipo.

Puede especificar el mismo selector más de una vez; si lo hace, el último valor especificado para un selector concreto es el que surte efecto.

Nota:

1. Los selectores de atributos enteros y de caracteres se asignan dentro de dos rangos diferentes; los selectores MQIA_* residen dentro del rango MQIA_FIRST a MQIA_LAST, y los selectores MQCA_* dentro del rango MQCA_FIRST a MQCA_LAST.

Para cada rango, las constantes MQIA_LAST_USED y MQCA_LAST_USED definen el valor más alto que acepta el gestor de colas.

2. Si todos los selectores MQIA_* aparecen primero, se pueden utilizar los mismos números de elemento para direccionar los elementos correspondientes en las matrices Selectors IntAttr y.
3. Si el **SelectorCount** parámetro es cero, no Selectors se hace referencia a; en este caso, la dirección del parámetro pasada por programas escritos en C o en ensambladore System/390 e podría ser nula.

Los atributos que se pueden configurar se enumeran en la siguiente tabla. No se pueden establecer otros atributos mediante esta llamada. Para los selectores de atributos MQCA_*, la constante que define la longitud en bytes de la cadena que se requiere en CharAttr se proporciona entre paréntesis.

Tabla 590. Seletores de atributos MQSET para colas		
Selector	Descripción	Nota
MQCA_TRIGGER_DATA	Datos de activación (MQ_TRIGGER_DATA_LENGTH).	
LISTAS_DE_DISTANCIAS_MQIA	Compatibilidad con listas de distribución.	1
MQIA_INHIBIR_OBTENER	Si se permiten las operaciones get.	
MQIA_INHIBIR_PUT	Si se permiten las operaciones de inserción.	
MQIA_TRIGGER_CONTROL	Control de desencadenante.	
MQIA_TRIGGER_DEPTH	Profundidad del desencadenante.	
MQIA_TRIGGER_MSG_PRIORITY	Umbral de prioridad del mensaje para activaciones.	
MQIA_TIPO_DISPARADOR	Tipo de desencadenante.	

Nota:

1. Solo compatible con las siguientes plataformas:

-  AIX
-  IBM i
-  Linux
-  Windows

y para IBM MQ MQI clients conectados a estos sistemas.

IntAttrCount

Tipo: MQLONG - entrada

Este es el número de elementos de la **IntAttr** matriz, y debe ser al menos igual al número de selectores MQIA_* del **Selectors** parámetro. Cero es un valor válido si no hay ninguno.

IntAttr

Tipo: MQLONGxIntAttrCount - entrada

Se trata de una matriz de valores de IntAttrCount atributos enteros. Estos valores de atributo deben estar en el mismo orden que los selectores MQIA_* en la **Selectors** matriz.

Si el parámetro **IntAttrCountSelectorCount** o es cero, no **IntAttr** se hace referencia a; en este caso, la dirección del parámetro pasada por programas escritos en C o en ensambladore System/390 es podría ser nula.

CharAttrLength

Tipo: MQLONG - entrada

Esta es la longitud en bytes del **CharAttr** parámetro, y debe ser al menos la suma de las longitudes de los atributos de caracteres especificados en la **Selectors** matriz. Cero es un valor válido si no hay selectores MQCA_* en **Selectors**.

CharAttr

Tipo: MQCHAR x CharAttrLength - entrada

Este es el búfer que contiene los valores de los atributos de los caracteres, concatenados entre sí. La longitud del búfer se proporciona en el parámetro **CharAttrLength**.

Los atributos de los caracteres deben especificarse en el mismo orden que los selectores MQCA_* en la **Selectors** matriz. La longitud de cada atributo de carácter es fija (véase **Selectores**). Si el valor que se va a establecer para un atributo contiene menos caracteres no en blanco que la longitud definida del atributo, rellene el valor a **CharAttr** la derecha con espacios en blanco para que el valor del atributo coincida con la longitud definida del atributo.

Si el parámetro **CharAttrLengthSelectorCount** o es cero, no **CharAttr** se hace referencia a; en este caso, la dirección del parámetro pasada por programas escritos en C o en ensambladore System/390 es podría ser nula.

CompCode

Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

El código de motivo que califica **CompCode**.

Si **CompCode** es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si **CompCode** es MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'89C') El adaptador no está disponible.

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

MQRC_API_EXIT_ERROR

(2374, X'946') La salida de la API ha fallado.

MQRC_API_EXIT_LOAD_ERROR

(2183, X'887') No se ha podido cargar la salida de la API.

MQRC_ASID_MISMATCH

(2157, X'86D') Los ASID primario e inicial varían.

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') La llamada MQI ha entrado antes de completar la llamada anterior.

MQRC_CF_NO_DISPONIBLE

(2345, X'929') Instalación de acoplamiento no disponible.

MQRC_CF_STRUC_FAILED

(2373, X'945') La estructura de recurso de acoplamiento ha fallado.

MQRC_CF_STRUC_IN_USE

(2346, X'92A') La estructura de recurso de acoplamiento se está utilizando.

MQRC_CF_STRUC_LIST_HDR_IN_USE

(2347, X'92B') La cabecera de lista de la estructura de recurso de acoplamiento se está utilizando.

MQRC_CHAR_ATTR_LENGTH_ERROR

(2006, X'7D6') Longitud de los atributos de los caracteres no válida.

MQRC_CHAR_ATTRS_ERROR

(2007, X'7D7') Cadena de atributos de carácter no válida.

MQRC_CICS_WAIT_FAILED

(2140, X'85C') Solicitud de espera rechazada por CICS.

MQRC_CONNECTION_BROKEN

(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

MQRC_CONNECTION_NOT_AUTHORIZED

(2217, X'8A9') No tiene autorización para la conexión.

MQRC_CONNECTION_STOPPING

(2203, X'89B') La conexión está concluyendo.

MQRC_DB2_NOT_AVAILABLE

(2342, X'926') Db2 subsistema no disponible.

MQRC_HCONN_ERROR

(2018, X'7E2') El manejador de conexión no es válido.

MQRC_HOBJ_ERROR

(2019, X'7E3') El manejador de objeto no es válido.

MQRC_INHIBIR_VALOR_ERROR

(2020, X'7E4') El valor del atributo de cola inhibit-get o inhibit-put no es válido.

MQRC_INT_ATTR_COUNT_ERROR

(2021, X'7E5') Recuento de atributos enteros no válido.

MQRC_INT_ATTRS_ARRAY_ERROR

(2023, X'7E7') Matriz de atributos enteros no válida.

MQRC_NO_ABIERTO_PARA_CONFIGURAR

(2040, X'7F8') La cola no está abierta para el conjunto.

MQRC_OBJECT_CHANGED

(2041, X'7F9') La definición de objeto ha cambiado desde que se ha abierto.

MQRC_OBJECT_DAMAGED

(2101, X'835') El objeto se ha dañado.

MQRC_PAGESET_ERROR

(2193, X'891') Error al acceder al conjunto de datos del conjunto de páginas.

MQRC_Q_DELETED

(2052, X'804') La cola se ha suprimido.

MQRC_Q_MGR_NAME_ERROR

(2058, X'80A') El nombre del gestor de colas no es válido o no se conoce.

MQRC_Q_MGR_NOT_AVAILABLE

(2059, X'80B') El gestor de colas no está disponible para efectuar la conexión.

MQRC_Q_MGR_STOPPING

(2162, X'872') El gestor de colas está concluyendo.

MQRC_RESOURCE_PROBLEM

(2102, X'836') No hay disponibles suficientes recursos del sistema.

ERROR_CON_EL_NÚMERO_DE_SELECCIONES_MQRC

(2065, X'811') Recuento de selectores no válido.

MQRC_SELECTOR_ERROR

(2067, X'813') Selector de atributos no válido.

MQRC_SELECTOR_LÍMITE_EXCEDIDO

(2066, X'812') Número de selectores demasiado grande.

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_SUPPRESSED_BY_EXIT

(2109, X'83D') El programa de salida ha suprimido la llamada.

ERROR_DE_CONTROL_DE_DISPARO_MQRC

(2075, X'81B') El valor del atributo de control de activación no es válido.

MQRC_TRIGGER_ERROR_DE_PROFUNDIDAD

(2076, X'81C') El valor del atributo trigger-depth no es válido.

MQRC_TRIGGER_MSG_PRIORITY_ERR

(2077, X'81D') El valor del atributo trigger-message-priority no es válido.

MQRC_TRIGGER_TYPE_ERROR

(2078, X'81E') El valor del atributo de tipo trigger no es válido.

MQRC_UNEXPECTED_ERROR

(2195, X'893') Se ha producido un error inesperado.

Para obtener información detallada sobre estos códigos, consulte [Mensajes y códigos de motivo](#).

Notas de uso

1. Mediante esta llamada, la aplicación puede especificar una matriz de atributos enteros, una colección de cadenas de atributos de caracteres, o ambos. Si no se producen errores, todos los atributos especificados se establecen simultáneamente. Si se produce un error (por ejemplo, si un selector no es válido o se intenta establecer un atributo con un valor que no es válido), la llamada falla y no se establece ningún atributo.
2. Los valores de los atributos se pueden determinar utilizando la llamada MQINQ; consulte [“MQINQ - Consultar atributos de objeto”](#) en la [página 707](#) para obtener más detalles.

Nota: No todos los atributos cuyos valores se pueden consultar mediante la llamada MQINQ pueden cambiar sus valores mediante la llamada MQSET. Por ejemplo, con esta llamada no se pueden establecer atributos de objetos de proceso ni de gestores de colas.

3. Los cambios en los atributos se conservan tras los reinicios del gestor de colas (excepto las modificaciones en las colas dinámicas temporales, que no se conservan tras los reinicios del gestor de colas).
4. No se pueden cambiar los atributos de una cola de modelos utilizando la llamada MQSET. Sin embargo, si abre una cola de modelos utilizando la llamada MQOPEN con la opción MQOO_SET, puede utilizar la llamada MQSET para establecer los atributos de la cola local dinámica creada por la llamada MQOPEN.
5. Si el objeto que se está configurando es una cola de clúster, debe existir una instancia local de la cola de clúster para que la apertura se realice correctamente.

Para obtener más información sobre los atributos de los objetos, consulte:

- “Atributos para colas” en la página 841
- “Atributos de las listas de nombres” en la página 876
- “Atributos de las definiciones de proceso” en la página 878
- “Atributos para el gestor de colas” en la página 801

Invocación en C

```
MQSET (Hconn, Hobj, SelectorCount, Selectors, IntAttrCount, IntAttrs,
CharAttrLength, CharAttrs, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQHCONN  Hconn;           /* Connection handle */
MQHOBJS  Hobj;            /* Object handle */
MQLONG   SelectorCount;   /* Count of selectors */
MQLONG   Selectors[n];    /* Array of attribute selectors */
MQLONG   IntAttrCount;    /* Count of integer attributes */
MQLONG   IntAttrs[n];     /* Array of integer attributes */
MQLONG   CharAttrLength;  /* Length of character attributes buffer */
MQCHAR   CharAttrs[n];    /* Character attributes */
MQLONG   CompCode;        /* Completion code */
MQLONG   Reason;          /* Reason code qualifying CompCode */
```

Invocación en COBOL

```
CALL 'MQSET' USING HCONN, HOBJ, SELECTORCOUNT, SELECTORS-TABLE,
INTATTRCOUNT, INTATTRS-TABLE, CHARATTRLENGTH,
CHARATTRS, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
** Connection handle
01 HCONN          PIC S9(9) BINARY.
** Object handle
01 HOBJ           PIC S9(9) BINARY.
** Count of selectors
01 SELECTORCOUNT PIC S9(9) BINARY.
** Array of attribute selectors
01 SELECTORS-TABLE.
02 SELECTORS      PIC S9(9) BINARY OCCURS n TIMES.
** Count of integer attributes
01 INTATTRCOUNT PIC S9(9) BINARY.
** Array of integer attributes
01 INTATTRS-TABLE.
02 INTATTRS      PIC S9(9) BINARY OCCURS n TIMES.
** Length of character attributes buffer
01 CHARATTRLENGTH PIC S9(9) BINARY.
** Character attributes
01 CHARATTRS      PIC X(n).
** Completion code
01 COMPCODE       PIC S9(9) BINARY.
** Reason code qualifying COMPCODE
01 REASON         PIC S9(9) BINARY.
```

Invocación en PL/I

```
call MQSET (Hconn, Hobj, SelectorCount, Selectors, IntAttrCount,
IntAttrs, CharAttrLength, CharAttrs, CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```
dcl Hconn          fixed bin(31); /* Connection handle */
dcl Hobj           fixed bin(31); /* Object handle */
dcl SelectorCount  fixed bin(31); /* Count of selectors */
dcl Selectors(n)   fixed bin(31); /* Array of attribute selectors */
dcl IntAttrCount   fixed bin(31); /* Count of integer attributes */
dcl IntAttrs(n)    fixed bin(31); /* Array of integer attributes */
dcl CharAttrLength fixed bin(31); /* Length of character attributes
```

```

                                buffer */
dcl CharAttrs      char(n);      /* Character attributes */
dcl CompCode       fixed bin(31); /* Completion code */
dcl Reason         fixed bin(31); /* Reason code qualifying
                                CompCode */

```

Invocación en ensamblador de alto nivel

```

CALL MQSET, (HCONN, HOBJ, SELECTORCOUNT, SELECTORS, INTATTRCOUNT, X
            INTATTRS, CHARATTRLENGTH, CHARATTRS, COMPCODE, REASON)

```

Declare los parámetros como se indica a continuación:

HCONN	DS	F	Connection handle
HOBJ	DS	F	Object handle
SELECTORCOUNT	DS	F	Count of selectors
SELECTORS	DS	(n)F	Array of attribute selectors
INTATTRCOUNT	DS	F	Count of integer attributes
INTATTRS	DS	(n)F	Array of integer attributes
CHARATTRLENGTH	DS	F	Length of character attributes buffer
CHARATTRS	DS	CL(n)	Character attributes
COMPCODE	DS	F	Completion code
REASON	DS	F	Reason code qualifying COMPCODE

Invocación en Visual Basic

```

MQSET Hconn, Hobj, SelectorCount, Selectors, IntAttrCount, IntAttrs,
      CharAttrLength, CharAttrs, CompCode, Reason

```

Declare los parámetros como se indica a continuación:

Dim Hconn	As Long	'Connection handle'
Dim Hobj	As Long	'Object handle'
Dim SelectorCount	As Long	'Count of selectors'
Dim Selectors	As Long	'Array of attribute selectors'
Dim IntAttrCount	As Long	'Count of integer attributes'
Dim IntAttrs	As Long	'Array of integer attributes'
Dim CharAttrLength	As Long	'Length of character attributes buffer'
Dim CharAttrs	As String	'Character attributes'
Dim CompCode	As Long	'Completion code'
Dim Reason	As Long	'Reason code qualifying CompCode'

MQSETMP: establecer la propiedad del mensaje

Utilice la llamada MQSETMP para establecer o modificar una propiedad de un identificador de mensaje.

Sintaxis

MQSETMP (*Hconn*, *Hmsg*, *SetPropOpts*, *Nombre*, *PropDesc*, *Tipo*, *LongitudDelValor*, *Valor*, *CódigoDeComposición*, *Motivo*)

Parámetros

Hconn

Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas.

El valor debe coincidir con el identificador de conexión que se utilizó para crear el identificador de mensaje especificado en el **Hmsg** parámetro. Si el identificador de mensaje se creó utilizando MQHC_UNASSOCIATED_HCONN, se debe establecer una conexión válida en el subproceso que configura una propiedad del identificador de mensaje; de lo contrario, la llamada fallará con el código de motivo MQRC_CONNECTION_BROKEN.

Hmsg

Tipo: MQHMSG - entrada

Este es el identificador del mensaje que se va a modificar. El valor fue devuelto por una llamada MQCRTMH anterior.

SetPropOpts

Tipo: MQSMPO - entrada

Controla cómo se configuran las propiedades de los mensajes.

Esta estructura permite a las aplicaciones especificar opciones que controlan cómo se establecen las propiedades de los mensajes. La estructura es un parámetro de entrada en la llamada MQSETMP. Para más información, consulte [MQSMPO](#).

Nombre

Tipo: MQCHARV- entrada

Este es el nombre de la propiedad que se va a configurar.

Consulte [Nombres de propiedades y Restricciones de nombres de propiedades](#) para obtener más información sobre el uso de nombres de propiedades.

PropDesc

Tipo: MQPD - entrada/salida

Esta estructura se utiliza para definir los atributos de una propiedad, incluyendo:

- ¿Qué ocurre si la propiedad no es compatible?
- ¿A qué contexto de mensaje pertenece la propiedad?
- ¿En qué mensajes se copia la propiedad a medida que fluye?

Consulte [MQPD](#) para obtener más información sobre esta estructura.

Tipo

Tipo: MQLONG - entrada

El tipo de datos de la propiedad que se está configurando. Puede ser uno de los siguientes:

MQTYPE_BOOLEAN

Un valor booleano. *ValueLength* debe ser 4.

MQTYPE_BYTE_STRING

Una cadena de bytes. *ValueLength* debe ser cero o mayor.

MQTYPE_INT8

Un entero con signo de 8 bits. *ValueLength* debe ser 1.

MQTYPE_INT16

Un entero con signo de 16 bits. *ValueLength* debe ser 2.

MQTYPE_INT32

Un entero con signo de 32 dígitos. *ValueLength* debe ser 4.

MQTYPE_INT64

Un entero con signo de 64 bits. *ValueLength* debe ser 8.

MQTYPE_FLOAT32

Un número de coma flotante de 32 bits. *ValueLength* debe ser 4.

Nota: este tipo no es compatible con aplicaciones que utilizan IBM COBOL para z/OS.

MQTYPE_FLOAT64

Un número de coma flotante de 64 bits. *ValueLength* debe ser 8.

Nota: este tipo no es compatible con aplicaciones que utilizan IBM COBOL para z/OS.

MQTYPE_STRING

Una cadena de caracteres. *ValueLength* debe ser cero o mayor, o el valor especial MQVL_NULL_TERMINATED.

MQTYPE_NULL

La propiedad existe, pero tiene un valor nulo. *ValueLength* debe ser cero.

ValueLength

Tipo: MQLONG - entrada

La longitud en bytes del valor de la propiedad en el parámetro *Value*. El valor cero solo es válido para valores nulos o para cadenas de caracteres o cadenas de bytes. Cero indica que la propiedad existe, pero que el valor no contiene caracteres ni bytes.

El valor debe ser mayor o igual a cero o al siguiente valor especial si el parámetro *Type* tiene MQTYPE_STRING establecido:

MQVL_TERMINADO_CON_NULL

El valor está delimitado por el primer valor nulo que se encuentra en la cadena. El valor nulo no se incluye como parte de la cadena. Este valor no es válido si no se ha establecido también MQTYPE_STRING.

Nota: El carácter nulo utilizado para terminar una cadena si se establece MQVL_NULL_TERMINATED es un nulo del conjunto de caracteres del valor.

Valor

Tipo: MQBYTEXValueLength - entrada

El valor de la propiedad que se va a establecer. El búfer debe alinearse en un límite adecuado a la naturaleza de los datos del valor.

En el lenguaje de programación C, el parámetro se declara como un puntero a void; se puede especificar como parámetro la dirección de cualquier tipo de datos.

Si *ValueLength* es cero, no se hace referencia al *valor*. En este caso, la dirección del parámetro pasada por programas escritos en C o en ensambladore System/390 e puede ser nula.

CompCode

Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

El código de razón que califica a *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_WARNING:

MQRC_RFH_FORMATO_ERROR

(2421, X'0975') No se pudo analizar una carpeta MQRFH2 que contiene propiedades.

Si *CompCode* es MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'089C') Adaptador no disponible.

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

MQRC_ASID_MISMATCH

(2157, X'86D') Los ASID primario e inicial varían.

MQRC_BUFFER_ERROR

(2004, X'07D4') Parámetro de valor no válido.

MQRC_BUFFER_LENGTH_ERROR

(2005, X'07D5') El parámetro de longitud de valor no es válido.

MQRC_CALL_IN_PROGRESS

(2219, X'08AB') Se ha iniciado una llamada al MQI antes de que finalizara la llamada anterior.

MQRC_HMSG_ERROR

(2460, X'099C') El puntero del identificador del mensaje no es válido.

MQRC_MSG_HANDLE_EN_USO

(2499, X'09C3') El identificador del mensaje ya está en uso.

MQRC_OPTIONS_ERROR

(2046, X'07FE') Opciones no válidas o incoherentes.

MQRC_PD_ERROR

(2482, X'09B2') Estructura del descriptor de propiedad no válida.

ERROR_NOMBRE_PROPIEDAD_MQRC

(2442, X'098A') Nombre de propiedad no válido.

MQRC_TIPO_DE_PROPIEDAD_ERROR

(2473, X'09A9') Tipo de datos de propiedad no válido.

ERROR_FORMATO_NÚMERO_PROPIEDAD_MQRC

(2472, X'09A8') Error de formato numérico encontrado en los datos del valor.

MQRC_SMPO_ERROR

(2463, X'099F') La estructura de opciones de propiedad del mensaje no es válida.

MQRC_SOURCE_CCSID_ERROR

(2111, X'083F') El identificador del conjunto de caracteres codificados del nombre de la propiedad no es válido.

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_UNEXPECTED_ERROR

(2195, X'893') Se ha producido un error inesperado.

Para obtener información detallada sobre estos códigos, consulte [Mensajes y códigos de motivo](#).

Invocación en C

```
MQSETMP (Hconn, Hmsg, &SetPropOpts, &Name, &PropDesc, Type,
ValueLength, &Value, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQHCONN  Hconn;          /* Connection handle */
MQHMSG    Hmsg;          /* Message handle */
MQSMPO    SetPropOpts;   /* Options that control the action of MQSETMP */
MQCHARV   Name;         /* Property name */
MQPD      PropDesc;      /* Property descriptor */
MQLONG    Type;         /* Property data type */
MQLONG    ValueLength;   /* Length of property value in Value */
MQBYTE    Value[n];      /* Property value */
MQLONG    CompCode;      /* Completion code */
MQLONG    Reason;        /* Reason code qualifying CompCode */
```

Invocación en COBOL

```
CALL 'MQSETMP' USING HCONN, HMSG, SETMSGOPTS, NAME, PROPDSC, TYPE,
                     VALUELENGTH, VALUE, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
**      Connection handle
01  HCONN      PIC S9(9) BINARY.
**      Message handle
```

```

01 HMSG      PIC S9(18) BINARY.
** Options that control the action of MQSETMP
01 SETMSGOPTS.
   COPY CMQSMPOV.
** Property name
01 NAME
   COPY CMQCHRVV.
** Property descriptor
01 PROPDESC.
   COPY CMQPDV.
** Property data type
01 TYPE      PIC S9(9) BINARY.
** Length of property value in VALUE
01 VALUELENGTH PIC S9(9) BINARY.
** Property value
01 VALUE     PIC X(n).
** Completion code
01 COMPCODE  PIC S9(9) BINARY.
** Reason code qualifying COMPCODE
01 REASON    PIC S9(9) BINARY.

```

Invocación en PL/I

```
call MQSETMP (Hconn, Hmsg, SetPropOpts, Name, PropDesc, Type, ValueLength,
             Value, CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```

dcl Hconn      fixed bin(31); /* Connection handle */
dcl Hmsg       fixed bin(63); /* Message handle */
dcl SetPropOpts like MQSMP0;  /* Options that control the action of MQSETMP */
dcl Name       like MQCHARV;  /* Property name */
dcl PropDesc   like MQPD;     /* Property descriptor */
dcl Type       fixed bin(31); /* Property data type */
dcl ValueLength fixed bin(31); /* Length of property value in Value */
dcl Value      char(n);       /* Property value */
dcl CompCode   fixed bin(31); /* Completion code */
dcl Reason     fixed bin(31); /* Reason code qualifying CompCode */

```

Invocación en ensamblador de alto nivel

```
CALL MQSETMP, (HCONN,HMSG,SETMSGHOPTS,NAME,PROPDESC,TYPE,VALUELENGTH,
              VALUE,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

HCONN	DS	F	Connection handle
HMSG	DS	D	Message handle
SETMSGOPTS	CMQSMPOA	,	Options that control the action of MQSETMP
NAME	CMQCHRVV	,	Property name
PROPDESC	CMQPDV	,	Property descriptor
TYPE	DS	F	Property data type
VALUELENGTH	DS	F	Length of property value in VALUE
VALUE	DS	CL(n)	Property value
COMPCODE	DS	F	Completion code
REASON	DS	F	Reason code qualifying COMPCODE

MQSTAT: recuperar información de estado

Utilice la llamada MQSTAT para recuperar información sobre el estado. El tipo de información de estado devuelta viene determinado por el valor Tipo especificado en la llamada.

Sintaxis

MQSTAT (*Hconn*, *Tipo*, *Estado*, *Código de comparación*, *Motivo*)

Parámetros

Hconn

Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* fue devuelto por una llamada anterior a MQCONN o MQCONNX.

En z/OS para CICS aplicaciones, se puede omitir la llamada MQCONN y especificar el siguiente valor para *Hconn*:

MQHC_DEF_HCONN

Manejador de conexión predeterminado.

Tipo

Tipo: MQLONG - entrada

Tipo de información de estado que se solicita. Los valores válidos son:

MQSTAT_TYPE_ASYNC_ERROR

Devuelve información sobre operaciones de inserción asíncronas anteriores.

MQSTAT_TYPE_RECONEXIÓN

Devuelve información sobre la reconexión. Si la conexión se está reconectando o no se ha podido reconectar, la información describe el fallo que ha provocado que la conexión comience a reconectarse.

Este valor solo es válido para conexiones de clientes. Para otros tipos de conexión, la llamada falla con el código de motivo **MQRC_ENVIRONMENT_ERROR**

MQSTAT_TYPE_ERROR_RECONEXIÓN

Devuelve información sobre un fallo anterior relacionado con la reconexión. Si la conexión no se ha podido restablecer, la información describe el fallo que ha provocado que la reconexión haya fallado.

Este valor solo es válido para conexiones de clientes. Para otros tipos de conexión, la llamada falla con el código de motivo **MQRC_ENVIRONMENT_ERROR**.

Estado

Tipo: MQSTS - entrada/salida

Estructura de la información de estado. Consulte [“MQSTS - Estructura de informes de estado” en la página 585](#) para obtener los detalles.

CompCode

Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

El código de razón que califica a *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_API_EXIT_ERROR

(2374, X'946') Error al salir de la API

MQRC_API_EXIT_LOAD_ERROR

(2183, X'887') No se ha podido cargar la salida de la API.

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') La llamada MQI ha entrado antes de completar la llamada anterior.

MQRC_CONNECTION_BROKEN

(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

MQRC_CONNECTION_STOPPING

(2203, X'89B') La conexión está concluyendo.

MQRC_FUNCTION_NOT_SUPPORTED

(2298, X'8FA') La función solicitada no está disponible en el entorno actual.

MQRC_HCONN_ERROR

(2018, X'7E2') El manejador de conexión no es válido.

MQRC_Q_MGR_STOPPING

(2162, X'872') - El gestor de colas se detiene

MQRC_RESOURCE_PROBLEM

(2102, X'836') No hay disponibles suficientes recursos del sistema.

MQRC_STAT_TYPE_ERROR

(2430, error « X'97E' » con tipo MQSTAT)

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_STS_ERROR

(2426, X'97A') Error con la estructura MQSTS

MQRC_UNEXPECTED_ERROR

(2195, X'893') Se ha producido un error inesperado.

Para obtener información detallada sobre estos códigos, consulte [Mensajes y códigos de motivo](#).

Notas de uso

1. Una llamada a MQSTAT especificando un tipo de MQSTAT_TYPE_ASYNC_ERROR devuelve información sobre operaciones asíncronas anteriores MQPUTMQPUT1 y. La MQSTS estructura devuelta al regresar de la MQSTAT llamada contiene la primera información de advertencia o error asíncrono registrada para esa conexión. Si tras el primer error o advertencia se producen otros, normalmente no alteran estos valores. Sin embargo, si se produce un error con un código de finalización MQCC_WARNING, se devuelve en su lugar un fallo MQCC_FAILED posterior con un código de finalización.
2. Si no se han producido errores desde que se estableció la conexión o desde la última llamada a MQSTAT, se devuelven un CompCode de MQCC_OKMQRC_NONE y un motivo de en la MQSTS estructura.
3. El recuento del número de llamadas asíncronas que se han procesado bajo el identificador de conexión se devuelve mediante tres campos contadores: PutSuccessCount, PutWarningCount y PutFailureCount. El gestor de colas incrementa estos contadores cada vez que se procesa correctamente una operación asíncrona, se produce una advertencia o se produce un error (tenga en cuenta que, a efectos contables, una inserción en una lista de distribución cuenta una vez por cada cola de destino, en lugar de una vez por cada lista de distribución). Un contador no se incrementa más allá del valor AMQ_LONG_MAX positivo máximo.
4. Una llamada exitosa a MQSTAT provoca que se restablezca cualquier información o recuento de errores anteriores.
5. El comportamiento de MQSTAT depende del valor del **MQSTAT Type** parámetro que proporcione.
6. **MQSTAT_TYPE_ASYNC_ERROR**
 - a. Una llamada a MQSTAT especificando un tipo de MQSTAT_TYPE_ASYNC_ERROR devuelve información sobre operaciones asíncronas anteriores MQPUTMQPUT1 y. La MQSTS estructura devuelta al regresar de la MQSTAT llamada contiene la primera información de advertencia o error asíncrono registrada para esa conexión. Si tras el primer error o advertencia se producen

otros, normalmente no alteran estos valores. Sin embargo, si se produce un error con un código de finalización MQCC_WARNING, se devuelve en su lugar un fallo MQCC_FAILED posterior con un código de finalización.

- b. Si no se han producido errores desde que se estableció la conexión o desde la última llamada a MQSTAT, se devuelven un CompCode de MQCC_OKMQRC_NONE y un motivo de en la MQSTS estructura.
- c. El recuento del número de llamadas asíncronas que se han procesado bajo el identificador de conexión se devuelve mediante tres campos contadores: PutSuccessCount, PutWarningCount y PutFailureCount. El gestor de colas incrementa estos contadores cada vez que se procesa correctamente una operación asíncrona, se produce una advertencia o se produce un error (tenga en cuenta que, a efectos contables, una inserción en una lista de distribución cuenta una vez por cada cola de destino, en lugar de una vez por cada lista de distribución). Un contador no se incrementa más allá del valor AMQ_LONG_MAXpositivo máximo.
- d. Una llamada exitosa a MQSTAT provoca que se restablezca cualquier información o recuento de errores anteriores.

MQSTAT_TYPE_RECONEXIÓN

Supongamos que llamas a MQSTAT con Type establecido en MQSTAT_TYPE_RECONNECTION dentro de un controlador de eventos durante la reconexión. Considera estos ejemplos.

El cliente está intentando volver a conectarse o no ha podido volver a conectarse.

CompCode en la MQSTS estructura es MQCC_FAILED y Reason podría ser MQRC_CONNECTION_BROKEN o MQRC_Q_MGR QUIESCING. ObjectType es MQOT_Q_MGR, ObjectName es el nombre del administrador de colas, y ObjectQMgrName está en blanco.

El cliente completó la reconexión correctamente o nunca se desconectó.

CompCode en la MQSTS estructura es MQCC_OK y el Reason es MQRC_NONE

Las llamadas posteriores MQSTAT devuelven los mismos resultados.

MQSTAT_TYPE_ERROR_RECONEXIÓN

Supongamos que realizas una llamada MQSTAT con Type el valor establecido en MQSTAT_TYPE_RECONNECTION_ERROR tras recibir MQRC_RECONNECT_FAILED una llamada de MQI. Considera estos ejemplos.

Se produjo un error de autorización al reabrir una cola durante la reconexión a un gestor de colas diferente.

CompCode en la MQSTS estructura es MQCC_FAILED y Reason es la razón por la que falló la reconexión, como MQRC_NOT_AUTHORIZED. ObjectType es el tipo de objeto que causó el problema, como MQOT_QUEUE, ObjectName es el nombre de la cola y ObjectQMgrName el nombre del gestor de colas propietario de la cola.

Se ha producido un error de conexión de socket durante la reconexión.

CompCode en la MQSTS estructura es MQCC_FAILED y Reason es la razón por la que falló la reconexión, como MQRC_HOST_NOT_AVAILABLE. ObjectType es MQOT_Q_MGR, ObjectName es el nombre del administrador de colas, y ObjectQMgrName está en blanco.

Las llamadas posteriores MQSTAT devuelven los mismos resultados.

Invocación en C

```
MQSTAT (Hconn, StatType, &Stat, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQHCONN Hconn;      /* Connection Handle */
MQLONG StatType;     /* Status type */
MQSTS Stat;          /* Status information structure */
MQLONG CompCode;     /* Completion code */
MQLONG Reason;       /* Reason code qualifying CompCode */
```

Invocación en COBOL

```
CALL 'MQSTAT' USING HCONN, STATTYPE, STAT, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
**      Connection handle
01      HCONN      PIC S9(9)      BINARY.
**      Status type
01      STATTYPE   PIC S9(9)      BINARY.
**      Status information
01      STAT.
      COPY CMQSTSV.
**      Completion code
01      COMPCODE   PIC S9(9)      BINARY.
**      Reason code qualifying COMPCODE
01      REASON     PIC S9(9)      BINARY.
```

Invocación en PL/I

```
call MQSTAT (Hconn, StatType, Stat, Compcode, Reason);
```

Declare los parámetros como se indica a continuación:

```
dcl Hconn      fixed bin(31); /* Connection handle */
dcl StatType   fixed bin(31); /* Status type */
dcl Stat       like MQSTS;    /* Status information structure */
dcl CompCode   fixed bin(31); /* Completion code */
dcl Reason     fixed bin(31); /* Reason code qualifying CompCode */
```

System/390 Invocación del ensamblador

```
CALL MQSTAT,(HCONN,STATTYPE,STAT,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

HCONN	DS	F	Connection handle
STATTYPE	DS	F	Status type
STAT	CMQSTSA,		Status information structure
COMPCODE	DS	F	Completion code
REASON	DS	F	Reason code qualifying COMPCODE

MQSUB - Registrar suscripción

Utilice la llamada MQSUB para registrar la suscripción de aplicaciones a un tema determinado.

Sintaxis

MQSUB (*Hconn*, *SubDesc*, *Hobj*, *Hsub*, *Compcode*, *Reason*)

Parámetros

Hconn

Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* fue devuelto por una llamada anterior a MQCONN o MQCONNX.

En z/OS para CICS aplicaciones, se puede omitir la llamada MQCONN y especificar el siguiente valor para *Hconn*:

MQHC_DEF_HCONN

Manejador de conexión predeterminado.

SubDesc

Tipo: MQSD - entrada/salida

Esta es una estructura que identifica el objeto en uso que está registrando la aplicación. Consulte [“MQSD - Descriptor de suscripción” en la página 558](#) para obtener más información.

Hobj

Tipo: MQHOBJ - entrada/salida

Este descriptor de contexto representa el acceso que se ha establecido para obtener los mensajes enviados a esta suscripción. Estos mensajes se pueden almacenar en una cola específica o el gestor de colas puede gestionar su almacenamiento sin utilizar ninguna cola específica.

Para utilizar una cola específica, debe asociarla con la suscripción al crear la suscripción. Puede hacerlo de dos maneras:

- Utilizando el mandato DEFINE SUB MQSC y proporcionando a ese mandato el nombre de un objeto de cola.
- Proporcionando este descriptor de contexto al llamar a MQSUB con MQSO_CREATE

Si se proporciona este descriptor de contexto como un parámetro de entrada en la llamada, debe ser un descriptor de contexto de objeto válido devuelto de una llamada MQOPEN anterior de una cola utilizando al menos una de las opciones siguientes:

- MQOO_INPUT_*
- MQOO_BROWSE
- MQOO_OUTPUT (si la cola es una cola remota)

Si este no es el caso, la llamada falla con MQRC_HOBJ_ERROR. No puede ser un descriptor de contexto de objeto a una cola de alias que se resuelva en un objeto de tema. Si es así, la llamada falla con MQRC_HOBJ_ERROR.

Si el gestor de colas va a gestionar el almacenamiento de los mensajes enviados a esta suscripción, se debe establecer al crear la suscripción, utilizando la opción MQSO_MANAGED. A continuación, el gestor de colas devuelve este descriptor de contexto como un parámetro de salida en la llamada. El descriptor de contexto devuelto se conoce como descriptor de contexto gestionado. Si se especifica MQHO_NONE pero no se especifica MQSO_MANAGED, la llamada falla con MQRC_HOBJ_ERROR.

Cuando el gestor de colas le devuelve un descriptor de contexto gestionado, puede utilizarlo en una llamada MQGET o MQCB con o sin opciones browse, en una llamada MQINQ, o en MQCLOSE. No puede utilizarlo en MQPUT, MQSUB, MQSET; si se intenta, fallará con MQRC_NOT_OPEN_FOR_OUTPUT, MQRC_HOBJ_ERROR o MQRC_NOT_OPEN_FOR_SET.

Si esta suscripción se reanuda utilizando la opción MQSO_RESUME en la estructura MQSD, el descriptor de contexto se puede devolver a la aplicación en este parámetro estableciendo MQSO_MANAGED en MQHO_NONE. Puede hacer esto independientemente de si la suscripción utiliza o no un descriptor de contexto gestionado, y puede ser útil para proporcionar suscripciones creadas utilizando DEFINE SUB con el descriptor de contexto a la cola de suscripciones definida en ese mandato. En el caso donde se está reanudando una suscripción creada administrativamente, la cola se abre con MQOO_INPUT_AS_Q_DEF y MQOO_BROWSE. Si necesita especificar otras opciones, la aplicación debe abrir la cola de suscripciones explícitamente y proporcionar el descriptor de contexto de objeto en la llamada. Si hay un problema al abrir la cola, la llamada falla con MQRC_INVALID_DESTINATION. Si se proporciona *Hobj*, debe ser equivalente a *Hobj* en la llamada MQSUB original. Esto significa que si se proporciona un descriptor de contexto de objeto devuelto de una llamada MQOPEN, el descriptor de contexto debe ser a la misma cola que se ha utilizado anteriormente. Si no es la misma cola, la llamada falla con MQRC_HOBJ_ERROR.

Si esta suscripción se está modificando utilizando la opción MQSO_ALTER en la estructura MQSD, se puede proporcionar un *Hobj* distinto. Las publicaciones entregadas a la cola e identificadas anteriormente mediante este parámetro permanecen en esa cola y es responsabilidad de la aplicación recuperar estos mensajes si ahora el parámetro **Hobj** representa una cola distinta.

Tabla 591. Utilización de <i>hobj</i> con varias opciones de suscripción		
Opciones	<i>Hobj</i>	Descripción
MQSO_CREATE + MQSO_MANAGED	Ignorado en la salida	Crea una suscripción con almacenamiento de mensajes gestionado por el gestor de colas
MQSO_CREATE	Un descriptor de contexto de objeto válido	Crea una suscripción que facilita una cola específica como destino de los mensajes.
MQSO_RESUME	MQHO_NONE	Reanuda una suscripción creada anteriormente independientemente de si está gestionada o no, y el gestor de colas devuelve el descriptor de contexto de objeto para que lo utilice la aplicación.
MQSO_RESUME	Un descriptor de contexto de objeto coincidente válido	Reanuda una suscripción creada anteriormente que utiliza una cola específica como destino de los mensajes y utiliza un descriptor de contexto de objeto con opciones de apertura específicas.
MQSO_ALTER + MQSO_MANAGED	MQHO_NONE	Modifica una suscripción existente que utilizaba anteriormente una cola específica, por lo que ahora es una suscripción gestionada. La clase de destino (gestionado o no) no se puede cambiar.
MQSO_ALTER	Un descriptor de contexto de objeto válido	Modifica una suscripción existente, gestionada o no, para que ahora utilice una cola específica. Cuando no se utiliza la opción MQSO_MANAGED, la cola proporcionada se puede cambiar, pero no se puede cambiar la clase de destino (gestionada o no).

Independientemente de si se ha proporcionado o devuelto, se debe especificar *Hobj* en las llamadas MQGET o MQCB subsiguientes que deseen recibir los mensajes de publicación enviados a esta suscripción.

El descriptor de contexto *Hobj* ya no es válido cuando se emite en él la llamada MQCLOSE, o cuando la unidad de proceso que define el ámbito del descriptor de contexto termina (hasta que la aplicación se desconecta). El ámbito del descriptor de contexto de objeto devuelto es el mismo que el del descriptor de conexión especificado en la llamada. Consulte [Hconn \(MQHCONN\) - salida](#) para obtener información sobre el ámbito del descriptor de contexto. Un MQCLOSE del descriptor de contexto *Hobj* no afecta al descriptor de contexto *Hsub*.

Hsub

Tipo: MQHOBJ - salida

Este descriptor de contexto representa la suscripción que se ha realizado. Se puede utilizar para otras dos operaciones:

- Se puede utilizar en una llamada MQSUBRQ subsiguiente para solicitar que las publicaciones se envíen cuando se haya utilizado la opción MQSO_PUBLICATIONS_ON_REQUEST al realizar la suscripción.

- Se puede utilizar en una llamada MQCLOSE subsiguiente para eliminar la suscripción que se ha realizado. El descriptor de contexto *Hsub* deja de ser válido cuando se emite la llamada MQCLOSE, o cuando la unidad de proceso que define el ámbito del descriptor de contexto termina. El ámbito del descriptor de contexto de objeto devuelto es el mismo que el del descriptor de conexión especificado en la llamada. Un MQCLOSE del descriptor de contexto *Hsub* no afecta al descriptor de contexto *Hobj*.

Este descriptor de contexto no se puede pasar a una llamada MQGET o MQCB. Debe utilizar el parámetro **Hobj**. No puede utilizar este descriptor de contexto en ninguna llamada IBM MQ que no sea MQCLOSE o MQSUBRQ. Si se pasa este descriptor de contexto a cualquier otra llamada IBM MQ, se genera MQRC_HOBY_ERROR.

CompCode

Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK

Realización satisfactoria

MQCC_WARNING

Aviso (terminación parcial)

MQCC_FAILED

Llamada fallida

Razón

Tipo: MQLONG - salida

El código de razón que califica a *CompCode*.

Si *CompCode* es MQCC_OK, el código de razón es el siguiente:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED, el código de razón es uno de los siguientes:

MQRC_CLUSTER_RESOLUTION_ERROR

(2189, X'88D') La resolución de nombres de clúster ha fallado.

MQRC_DURABILITY_NOT_ALLOWED

2436 (X'0984') Ha fallado una llamada MQSUB utilizando la opción MQSO_DURABLE.

MQRC_FUNCTION_NOT_SUPPORTED

2298 (X'08FA') La función solicitada no está disponible en el entorno actual.

MQRC_HOBY_ERROR

2019 (X'07E3') Descriptor de contexto de objeto Hobj no válido.

MQRC_IDENTITY_MISMATCH

2434 (X'0982') El nombre de suscripción coincide con una suscripción existente.

MQRC_NOT_AUTHORIZED

2035 (X'07F3') El usuario no está autorizado para realizar la operación.

MQRC_NO_SUBSCRIPTION

2428 (X'097C') El nombre de suscripción identificado no existe.

MQRC_OBJECT_STRING_ERROR

2441 (X'0989') El campo Objectstring no es válido.

MQRC_OPTIONS_ERROR

2046 (X'07FE') El parámetro o campo Options contiene opciones que no son válidas, o una combinación de opciones que no es válida.

MQRC_Q_MGR QUIESCING

2161 (X'0871') El gestor de colas se está poniendo en pausa.

MQRC_RECONNECT_Q_MGR_REQD

2555 (X'09FB'X) Se requiere la opción MQCNO_RECONNECT_Q_MGR.

MQRC_RETAINED_MSG_Q_ERROR

2525 (X'09DD') No se pueden recuperar las publicaciones retenidas que existen para la serie de tema suscrita.

MQRC_RETAINED_NOT_DELIVERED

2526 (X'09DE') Las publicaciones existentes para la serie de tema suscrita no se pueden entregar a la cola de destino de suscripción y no se pueden entregar a la cola de mensajes no entregados.

MQRC_SD_ERROR

2424 (X'0978') El descriptor de suscripción (MQSD) no es válido.

MQRC_SELECTION_NOT_AVAILABLE

2551 (X'09F7') La serie de selección no sigue la sintaxis del selector de IBM MQ y no hay ningún proveedor de selección de mensajes ampliados disponible.

MQRC_SELECTION_STRING_ERROR

2519 (X'09D7') La serie de selección se debe especificar tal como se describe en la documentación de la estructura MQCHARV.

MQRC_SELECTOR_SYNTAX_ERROR

2459 (X'099B') Se ha emitido una llamada MQOPEN, MQPUT1 o MQSUB pero la serie de selección que se ha especificado contenía un error de sintaxis.

MQRC_SUB_USER_DATA_ERROR

2431 (X'097F') El campo SubUserData no es válido.

MQRC_SUB_NAME_ERROR

2440 (X'0988') El campo SubName no es válido.

MQRC_SUB_ALREADY_EXISTS

2432 (X'0980') La suscripción ya existe.

MQRC_SUB_USER_DATA_ERROR

2431 (X'097F') El campo SubUserData no es válido.

MQRC_TOPIC_STRING_ERROR

2425 (X'0979') La serie de tema no es válida.

MQRC_UNKNOWN_OBJECT_NAME

2085 (X'0825') No se puede encontrar el objeto identificado en el campo ObjectName de MQSD.

MQRC_SUB_JOIN_NOT_ALTERABLE

29440 (X'7300') La modalidad de uso compartido de suscripciones no es compatible con la suscripción existente. Se puede devolver este error al intentar reanudar una suscripción compartida de JMS 2.0 en una aplicación que no sea JMS.

Para obtener información detallada sobre estos códigos, consulte [Mensajes y códigos de motivo](#).

Notas de uso

- La suscripción se compone de un tema, nombrado utilizando el nombre abreviado de un objeto de tema predefinido, el nombre completo de la serie de tema o formado por la concatenación de dos partes. Consulte la descripción de *ObjectName* y *ObjectString* en [“MQSD - Descriptor de suscripción”](#) en la página 558.
- El gestor de colas efectúa comprobaciones de seguridad cuando se emite una llamada MQSUB para verificar que el identificador de usuario con el que se está ejecutando la aplicación tiene el nivel de autorización adecuado antes de permitir el acceso. El objeto de tema adecuado se encuentra en la jerarquía de temas y se realiza una comprobación de autoridad en este objeto de tema para garantizar que se ha establecido la autoridad para suscripción. Si no se utiliza la opción MQSO_MANAGED, se realiza una comprobación de autoridad en la cola de destino para garantizar que se ha establecido la autoridad para salida. Si se utiliza la opción MQSO_MANAGED, no se realiza ninguna comprobación de autoridad en la cola gestionada para acceso de consulta o salida.
- Si no proporciona un Hobj como entrada, la llamada MQSUB asigna dos descriptores de contexto, un descriptor de contexto de objeto (Hobj) y un descriptor de contexto de suscripción (Hsub).

- El Hobj devuelto en la llamada MQSUB cuando se utiliza la opción MQSO_MANAGED se puede consultar para encontrar atributos como por ejemplo el umbral de retroceso o el nombre de recolocación excesiva en cola de retroceso. También puede consultar el nombre de la cola gestionada, pero no debe intentar abrir directamente esta cola.
 - Las suscripciones se pueden agrupar, lo que permite que se entregue una única publicación al grupo de suscripciones, incluso si más de una suscripción del grupo coincidía con la publicación. Las suscripciones se pueden agrupar utilizando la opción MQSO_GROUP_SUB, y para agrupar suscripciones estas deben
 - utilizar la misma cola con nombre (es decir, no utilizar la opción MQSO_MANAGED) en el mismo gestor de colas, representado por el parámetro Hobj en la llamada MQSUB
 - compartir el mismo SubCorrelId
 - estar en el mismo SubLevel
- Estos atributos definen el conjunto de suscripciones que se tienen en cuenta para formar parte de un grupo y que también son los atributos que no se pueden alterar si se agrupa una suscripción. Si se altera el SubLevel, se genera MQRC_SUBLEVEL_NOT_ALTERABLE, y si se altera cualquiera de los demás atributos (que se pueden cambiar si una suscripción no está agrupada) se genera MQRC_GROUPING_NOT_ALTERABLE.
- La finalización satisfactoria de la llamada MQSUB no significa que la acción se haya completado. Para comprobar que esta llamada se ha completado, consulte el paso [DEFINE SUB](#) en [Comprobación de que los mandatos asíncronos en redes distribuidas han finalizado](#).
 - Los campos del MQSD están rellenos cuando se devuelven de una llamada MQSUB que utiliza la opción MQSO_RESUME. El MQSD devuelto se puede pasar directamente en una llamada MQSUB que utiliza la opción MQSO_ALTER con los cambios que es necesario realizar a la suscripción aplicada al MQSD. Algunos campos tienen algunas consideraciones especiales, tal como se indica en la tabla.

Tabla 592. Consideraciones especiales de los campos del MQSD	
Nombre de campo en MQSD	Consideraciones especiales
Opciones de acceso o creación	Algunas de las opciones se pueden restablecer cuando se devuelve la llamada MQSUB. Si, a continuación, reutiliza el MQSD en una llamada MQSUB, la opción necesaria se debe establecer explícitamente.
Opciones de durabilidad, Opciones de destino, Opciones de registro & Opciones de comodín	Estas opciones se establecen según corresponda
Opciones de publicación	Estas opciones se establecen según corresponda, excepto MQSO_NEW_PUBLICATIONS_ONLY, que sólo es aplicable a MQSO_CREATE.
Otras opciones	Estas opciones no se modifican en la devolución de una llamada MQSUB. Controlan cómo se emite la llamada de API y no se almacenan con la suscripción. Deben establecerse de la forma que sea necesaria en cualquier llamada MQSUB subsiguiente que vuelva a utilizar MQSD.
ObjectName	Este campo de sólo entrada no se modifica en la devolución de una llamada MQSUB.
ObjectString	Este campo de sólo entrada no se modifica en la devolución de una llamada MQSUB. El nombre de tema completo utilizado en el campo <i>ResObjectString</i> , si se proporciona un almacenamiento intermedio.

Tabla 592. Consideraciones especiales de los campos del MQSD (continuación)	
Nombre de campo en MQSD	Consideraciones especiales
AlternateUserId y AlternateSecurityId	Estos campos de sólo entrada no se modifican en la devolución de una llamada MQSUB. Controlan cómo se emite la llamada de API y no se almacenan con la suscripción. Deben establecerse de la forma que sea necesaria en cualquier llamada MQSUB subsiguiente que vuelva a utilizar MQSD.
SubExpiry	En la devolución de una llamada MQSUB mediante la opción MQSO_RESUME, este campo se establece en la caducidad original de la suscripción y no en el tiempo restante para la caducidad. Si a continuación reutiliza el MQSD en una llamada MQSUB utilizando la opción MQSO_ALTER, restablezca la caducidad de la suscripción para reiniciar la cuenta regresiva.
SubName	Este campo es un campo de entrada en una llamada MQSUB y no se modifica en la salida.
SubUserData y SelectionString	Estos campos de longitud variable se devuelven en la salida de una llamada MQSUB utilizando la opción MQSO_RESUME, si se proporciona un almacenamiento intermedio, y también una longitud de almacenamiento intermedio positiva en <i>VSBufSize</i>. Si no se proporciona ningún almacenamiento intermedio, sólo se devuelve la longitud en el campo <i>VSLength</i> de MQCHARV. Si el almacenamiento intermedio proporcionado es inferior al espacio necesario para devolver el campo, sólo se devuelven <i>VSBufSize</i> bytes en el almacenamiento intermedio proporcionado. Si a continuación reutiliza el MQSD en una llamada MQSUB utilizando la opción MQSO_ALTER y no se proporciona ningún almacenamiento intermedio pero se proporciona <i>VSLength</i> distinto de cero, si esta longitud coincide con la longitud existente del campo no se realizará ninguna modificación en el campo.
SubCorrelId y PubAccountingToken	Si no utiliza MQSO_SET_CORREL_ID, el gestor de colas genera el <i>SubCorrelId</i>. Si no utiliza MQSO_SET_IDENTITY_CONTEXT, el gestor de colas genera <i>PubAccountingToken</i>. Estos campos se devuelven en el MQSD de una llamada MQSUB utilizando la opción MQSO_RESUME. Si el gestor de colas los genera, el valor generado se devuelve en una llamada MQSUB utilizando la opción MQSO_CREATE o MQSO_ALTER.
PubPriority, SubLevel & PubApplIdentityData	Estos campos se devuelven en el MQSD.
ResObjectString	Este campo de sólo salida se devuelve en el MQSD si se proporciona un almacenamiento intermedio.

Invocación en C

```
MQSUB (Hconn, &SubDesc, &Hobj, &Hsub, &CompCode, &Reason)
```

Declare los parámetros como se indica a continuación:

```
MQHCONN Hconn; /* Connection handle */
MQSD SubDesc; /* Subscription descriptor */
MQHOBJ Hobj; /* Object handle */
MQHOBJ Hsub; /* Subscription handle */
MQLONG CompCode; /* Completion code */
MQLONG Reason; /* Reason code qualifying CompCode */
```

Invocación en COBOL

```
CALL 'MQSUB' USING HCONN, SUBDESC, HOBJ, HSUB, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
** Connection handle
01 HCONN PIC S9(9) BINARY.
** Subscription descriptor
01 SUBDESC.
COPY CMQSDV.
** Object handle
01 HOBJ PIC S9(9) BINARY.
** Subscription handle
01 HSUB PIC S9(9) BINARY.
** Completion code
01 COMPCODE PIC S9(9) BINARY.
** Reason code qualifying COMPCODE
01 REASON PIC S9(9) BINARY.
```

Invocación en PL/I

```
call MQSUB (Hconn, SubDesc, Hobj, Hsub, CompCode, Reason)
```

Declare los parámetros como se indica a continuación:

```
dcl Hconn fixed bin(31); /* Connection handle */
dcl SubDesc like MQSD; /* Subscription descriptor */
dcl Hobj fixed bin(31); /* Object handle */
dcl Hsub fixed bin(31); /* Subscription handle */
dcl CompCode fixed bin(31); /* Completion code */
dcl Reason fixed bin(31); /* Reason code qualifying CompCode */
```

Invocación en ensamblador de alto nivel

```
CALL MQSUB, (HCONN, SUBDESC, HOBJ, HSUB, COMPCODE, REASON)
```

Declare los parámetros como se indica a continuación:

```
HCONN DS F Connection handle
SUBDESC CMQSDA , Subscription descriptor
HOBJ DS F Object handle
HSUB DS F Subscription handle
COMPCODE DS F Completion code
REASON DS F Reason code qualifying COMPCODE
```

MQSUBRQ - Solicitud de suscripción

Utilice la llamada MQSUBRQ para realizar una solicitud de la publicación retenida, cuando el suscriptor se haya registrado con MQSO_PUBLICATIONS_ON_REQUEST.

Sintaxis

MQSUBRQ (*Hconn*, *Hsub*, *Acción*, *SubRqOpts*, *Código de comparación*, *Motivo*)

Parámetros

Hconn

Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* fue devuelto por una llamada anterior a MQCONN o MQCONNX.

En z/OS para CICS aplicaciones, se puede omitir la llamada MQCONN y especificar el siguiente valor para *Hconn*:

MQHC_DEF_HCONN

Manejador de conexión predeterminado.

Hsub

Tipo: MQHOBJ - entrada

Este identificador representa la suscripción para la que se va a solicitar una actualización. El valor de *Hsub* se devolvió desde una llamada MQSUB anterior.

Acción

Tipo: MQLONG - entrada

Este parámetro controla la acción concreta que se solicita en la suscripción. Se debe especificar el siguiente valor:

MQSR_ACCIÓN_PUBLICACIÓN

Esta acción solicita que se envíe una publicación actualizada sobre el tema especificado. Solo se puede utilizar si el suscriptor especificó la opción MQSO_PUBLICATIONS_ON_REQUEST en la llamada MQSUB cuando realizó la suscripción. Si el gestor de colas tiene una publicación retenida para el tema, esta se envía al suscriptor. Si no es así, la llamada falla. Si se envía una solicitud a una publicación que se ha retenido, esto se indica mediante la propiedad del mensaje « MQIsRetained » (Solicitud retenida) de dicha publicación.

Dado que el tema de la suscripción existente representado por el parámetro Hsub puede contener comodines, es posible que el suscriptor reciba varias publicaciones retenidas.

SubRqOpts

Tipo: MQSRO - entrada/salida

Estas opciones controlan la acción de MQSUBRQ. Consulte [“MQSRO - Opciones de solicitud de suscripción”](#) en la [página 582](#) para obtener más detalles.

Si no se requieren opciones, los programas escritos en C o en ensambladore S/390 e pueden especificar una dirección de parámetro nula en lugar de especificar la dirección de una estructura MQSRO.

CompCode

Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK

Realización satisfactoria

MQCC_WARNING

Aviso (terminación parcial)

MQCC_FAILED

Llamada fallida

Razón

Tipo: MQLONG - salida

El código de razón que califica a *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_FUNCTION_NOT_SUPPORTED

2298 (X'08FA') La función solicitada no está disponible en el entorno actual.

MQRC_NO_MENSAJE_RETENIDO

2437 (X'0985') Actualmente no hay publicaciones conservadas almacenadas para este tema.

MQRC_OPTIONS_ERROR

2046 (X'07FE') El parámetro o campo Options contiene opciones que no son válidas, o una combinación de opciones que no es válida.

MQRC_Q_MGR QUIESCING

2161 (X'0871') El gestor de colas se está poniendo en pausa.

MQRC_SRO_ERROR

2438 (X'0986') En la llamada MQSUBRQ, las opciones de solicitud de suscripción MQSRO no son válidas.

MQRC_RETAINED_MSG_Q_ERROR

2525 (X'09DD') No se pueden recuperar las publicaciones retenidas que existen para la serie de tema suscrita.

MQRC_RETAINED_NOT_DELIVERED

2526 (X'09DE') Las publicaciones existentes para la serie de tema suscrita no se pueden entregar a la cola de destino de suscripción y no se pueden entregar a la cola de mensajes no entregados.

Para obtener información detallada sobre estos códigos, consulte [Mensajes y códigos de motivo](#).

Notas de uso

Las siguientes notas de uso se aplican al uso del código de acción MQSR_ACTION_PUBLICATION:

1. Si este verbo se completa correctamente, las publicaciones retenidas que coinciden con la suscripción especificada se han enviado a la suscripción y se pueden recibir utilizando MQGET o MQCB con el Hobj devuelto en el verbo MQSUB original que creó la suscripción.
2. Si el tema suscrito por el verbo MQSUB original que creó la suscripción contenía un comodín, se pueden enviar más de una publicación retenida. El número de publicaciones enviadas como resultado de esta convocatoria se registra en el campo « NumPubs » (Publicaciones enviadas) de la estructura « SubRqOpts » (Estadísticas de la convocatoria).
3. Si este verbo se completa con un código de motivo MQRC_NO_RETAINED_MSG, significa que no hay publicaciones retenidas actualmente para el tema especificado.#
4. Si este verbo se completa con un código de motivo MQRC_RETAINED_MSG_Q_ERROR o MQRC_RETAINED_NOT_DELIVERED, significa que actualmente hay publicaciones retenidas para el tema especificado, pero se ha producido un error que ha impedido su entrega.
5. La aplicación debe tener una suscripción vigente al tema antes de poder realizar esta llamada. Si la suscripción se realizó en una instancia anterior de la aplicación y no hay disponible un identificador válido para la suscripción, la aplicación debe llamar primero a MQSUB con la opción MQSO_RESUME para obtener un identificador que se utilizará en esta llamada.
6. Las publicaciones se envían al destino registrado para su uso con la suscripción actual de esta aplicación. Si las publicaciones deben enviarse a otro lugar, primero debe modificarse la suscripción utilizando la llamada MQSUB con la opción MQSO_ALTER.

Invocación en C

```
MQSUB (Hconn, Hsub, Action, &SubRqOpts, &CompCode, &Reason)
```

Declare los parámetros como se indica a continuación:

```
MQHCONN Hconn;      /* Connection handle */
MQHOBJ   Hsub;       /* Subscription handle */
MQLONG   Action;     /* Action requested by MQSUBRQ */
MQSRO    SubRqOpts; /* Options that control the action of MQSUBRQ */
MQLONG   CompCode;   /* Completion code */
MQLONG   Reason;     /* Reason code qualifying CompCode */
```

Invocación en COBOL

```
CALL 'MQSUBRQ' USING HCONN, HSUB, ACTION, SUBRQOPTS, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
** Connection handle
01 HCONN PIC S9(9) BINARY.
** Subscription handle
01 HSUB PIC S9(9) BINARY.
** Action requested by MQSUBRQ
01 ACTION PIC S9(9) BINARY.
** Options that control the action of MQSUBRQ
01 SUBRQOPTS.
COPY CMQSROV.
** Completion code
01 COMPCODE PIC S9(9) BINARY.
** Reason code qualifying COMPCODE
01 REASON PIC S9(9) BINARY.
```

Invocación en PL/I

```
call MQSUBRQ (Hconn, Hsub, Action, SubRqOpts, CompCode, Reason)
```

Declare los parámetros como se indica a continuación:

```
dcl Hconn fixed bin(31); /* Connection handle */
dcl Hsub fixed bin(31);  /* Subscription handle */
dcl Action fixed bin(31); /* Action requested by MQSUBRQ */
dcl SubRqOpts like MQSRO; /* Options that control the action of MQSUBRQ */
dcl CompCode fixed bin(31); /* Completion code */
dcl Reason fixed bin(31); /* Reason code qualifying CompCode */
```

Invocación en ensamblador de alto nivel

```
CALL MQSUBRQ,(HCONN, HSUB, ACTION, SUBRQOPTS,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

```
HCONN DS F Connection handle
HSUB DS F Subscription handle
ACTION DS F Action requested by MQSUBRQ
SUBRQOPTS CMQSROA , Options that control the action of MQSUBRQ
COMPCODE DS F Completion code
REASON DS F Reason code qualifying COMPCODE
```

objeto, atributos de

Esta colección de temas lista sólo los objetos IBM MQ que pueden ser objeto de una llamada a función MQINQ, y proporciona detalles de los atributos que se pueden consultar y los selectores que se van a utilizar.

Atributos para el gestor de colas

Algunos atributos del gestor de colas son fijos para implementaciones concretas; otros se pueden modificar mediante el comando **ALTER QMGRMQSC**.

Los atributos también se pueden mostrar utilizando el comando **DISPLAY QMGR**. La mayoría de los atributos del gestor de colas se pueden consultar abriendo un objeto especial MQOT_Q_MGR y utilizando la llamada MQINQ con el identificador devuelto.

La siguiente tabla resume los atributos específicos del gestor de colas. Los atributos se describen en orden alfabético.

Nota: Los nombres de los atributos que se muestran en esta sección son nombres descriptivos utilizados con la llamada MQINQ; los nombres son los mismos que para los comandos PCF. Cuando se utilizan comandos MQSC para definir, modificar o mostrar atributos, se utilizan nombres cortos alternativos. Para obtener más información, consulte [los comandos MQSC](#).

Tabla 593. Atributos para el gestor de colas	
Atributo	Descripción
AccountingConnOverride	Anular la configuración contable.
AccountingInterval	Con qué frecuencia se deben llevar los registros contables intermedios.
ActivityConnOverride	Anular la configuración de la actividad.
ActivityTrace	Controla la recopilación de registros de actividad de la aplicación IBM MQMQI.
AdoptNewMCACheck	Elementos comprobados para determinar si se adopta el nuevo MCA.
 AdoptNewMCAType	Si se debe reiniciar automáticamente una instancia huérfana de un MCA de un tipo de canal concreto.
AlterationDate	Fecha en la que se modificó por última vez la definición.
AlterationTime	Fecha en la que se modificó por última vez la definición.
AuthorityEvent	 Desde IBM MQ 10.0.0, controla si se generan eventos de autorización exitosos y no exitosos.
 AuthorityEventScope	Controla el alcance de los eventos de autorización cuando está habilitado.
 BridgeEvent	Atributo de control para eventos de puente.
ChannelAutoDef	Controla si se permite la definición automática de canales.
ChannelAutoDefEvent	Controla si se generan eventos de definición automática de canales.
ChannelAutoDefExit	Nombre de la salida de usuario para la definición automática de canales.
ChannelEvent	Atributo de control para eventos de canal.
ChannelInitiatorControl	Atributo de control para el iniciador del canal.
ChannelMonitoring	Datos de monitorización en línea para canales.
ChannelStatistics	Controla la recopilación de datos estadísticos para los canales.
 ChinitAdapters	Número de subtarefas del adaptador para procesar IBM MQ llamadas.
 ChinitDispatchers	Número de despachadores que se utilizarán para el iniciador del canal.
	Reservado para IBM uso.
 ChinitTraceAutoStart	Si el seguimiento del iniciador del canal debe iniciarse automáticamente.
 ChinitTraceTableSize	Tamaño del espacio de datos de rastreo del iniciador del canal.
ClusterSenderMonitoringDefault	Datos de supervisión en línea predeterminados para los canales emisores del clúster.
ClusterSenderStatistics	Controla la recopilación de información de supervisión de estadísticas para los canales emisores del clúster.
ClusterWorkloadData	Datos de usuario para la salida de la carga de trabajo del clúster.
ClusterWorkloadExit	Nombre de la salida de usuario para la gestión de la carga de trabajo del clúster.

Tabla 593. Atributos para el gestor de colas (continuación)

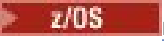
Atributo	Descripción
ClusterWorkloadLength	Longitud máxima de los datos del mensaje transmitidos a la salida de la carga de trabajo del clúster.
CLWLMRUCChannels	Número de canales utilizados más recientemente para equilibrar la carga de trabajo del clúster.
CLWLUseQ	La carga de trabajo del clúster utiliza una cola remota.
CodedCharSetId	Identificador de juego de caracteres codificados.
CommandEvent	Atributo de control para eventos de comando.
CommandInputQName atributo	Nombre de la cola de entrada de comandos.
CommandLevel	Nivel de comando.
CommandServerControl atributo	Atributo de control para el servidor de comandos.
Atributo de evento de configuración	Atributo de control para eventos de configuración.
DeadLetterQName	Nombre de la cola de mensajes no entregados.
DefClusterXmitQueueType	Tipo de cola de transmisión de clúster predeterminado.
DefXmitQName	Nombre predeterminado de la cola de transmisión.
DistLists	Compatibilidad con listas de distribución.
 DNSGroup	Nombre del grupo para el oyente « TCP » cuando se utiliza la compatibilidad con los servicios de nombres de dominio dinámicos de Workload Manager.
 DNSWLM	Si un oyente TCP se registra en Workload Manager para Dynamic Domain Name Services.
ExpiryInterval	Intervalo entre escaneos de mensajes caducados.
IGQPutAuthority	Autoridad para poner colas dentro del grupo.
IGQUserId	Identificador de usuario en cola dentro del grupo.
InhibitEvent	Controla si se generan eventos de inhibición (Inhibit Get e Inhibit Put).
InitialKey 1	Clave inicial para el sistema de protección con contraseña.
IPAddressVersion	Versión de la dirección Internet Protocol.
 IntraGroupQueuing	Soporte para colas dentro del grupo.
 ListenerTimer	Intervalo de tiempo entre los intentos de reiniciar el listener tras un fallo de APPC o TCP/IP.
LocalEvent	Controla si se generan eventos de error locales.
LoggerEvent	Controla si se generan eventos de registro.
 LUGroupName	Nombre genérico de LU para el listener de LU 6.2 que gestiona las transmisiones entrantes para el grupo de uso compartido de colas.
 Nombre de unidad	Nombre de la LU que se utilizará para las transmisiones de salida de la LU 6.2.
LU62ARMSuffix	Sufijo de un miembro de SYS1.PARMLIB APPCPMxx, que nombra LUADD para este iniciador de canal.
 LU62Channels	Número máximo de canales actuales o clientes conectados que utilizan LU 6.2.
MaxActiveChannels	Número máximo de canales que pueden estar activos en cualquier momento.
MaxChannels	Número máximo de canales actuales.
MaxHandles	Número máximo de asas.
MaxMsgLength	Longitud máxima del mensaje en bytes.
MaxPriority, atributo	Prioridad máxima.
MaxPropertiesLength	Longitud máxima de los datos de la propiedad en bytes.
MaxUncommittedMsgs	Número máximo de mensajes no confirmados dentro de una unidad de trabajo.
MQIAccounting	Controla la recopilación de información contable para los datos de MQI.
MQIStatistics	Controla la recopilación de información estadística de supervisión para el gestor de colas.
MsgMarkBrowseInterval	Intervalo tras el cual el gestor de colas puede eliminar la marca de los mensajes consultados.

Tabla 593. Atributos para el gestor de colas (continuación)

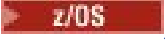
Atributo	Descripción
 OutboundPortMin	Con <i>OutboundPortMin</i> , define el rango de números de puerto que se utilizará al vincular canales salientes.
 OutboundPortMax	Con <i>OutboundPortMax</i> , define el rango de números de puerto que se utilizará al vincular canales salientes.
PerformanceEvent	Controla si se generan eventos relacionados con el rendimiento.
Plataforma	Plataforma en la que se ejecuta el gestor de colas.
PubSubNPIInputMsg	Si se debe descartar (o conservar) un mensaje de entrada no entregado.
PubSubNPResponse	Controla el comportamiento de los mensajes no entregados.
PubSubMaxMsgRetryCount	El número de reintentos al procesar (bajo punto de sincronismo) un mensaje de mandato anómalo.
PubSubSyncPoint	Si solo deben procesarse los mensajes persistentes (o todos) bajo el punto de sincronización.
PubSubMode	Si la interfaz de publicación/suscripción en cola está en funcionamiento.
QMgrDesc	Descripción del gestor de colas.
QMgrIdentifier	Identificador único generado internamente del gestor de colas.
QMgrName	Nombre del gestor de colas.
QSGName	Nombre del grupo de uso compartido de colas.
QueueAccounting	Controla la recopilación de información contable para las colas.
QueueMonitoring	Datos de supervisión en línea para colas.
QueueStatistics	Controla la recopilación de datos estadísticos para las colas.
 ReceiveTimeout	Cuánto tiempo espera un canal de « TCP/IP » a recibir datos antes de volver al estado inactivo.
 ReceiveTimeoutMin	Clasificadorio para ReceiveTimeout .
 ReceiveTimeoutType	Tiempo mínimo que un canal de « TCP/IP » espera a recibir datos antes de volver al estado inactivo.
RemoteEvent	Controla si se generan eventos de error remotos.
RepositoryName	Nombre del clúster para el que este gestor de colas proporciona servicios de repositorio.
RepositoryNamelist	Nombre del objeto de la lista de nombres que contiene los nombres de los clústeres para los que este gestor de colas proporciona servicios de repositorio.
ScyCase	Caso de perfiles de seguridad
SharedQMgrName	Nombre del gestor de colas compartidas
“SPLCAP” en la página 837	Advanced Message Security (AMS) protección para un gestor de colas activada o desactivada.
Lista de nombres SSLCRL 1	Nombre del objeto de la lista de nombres que contiene los nombres de los objetos de información de autenticación.
SSLCryptoHardware 1	Cadena de configuración de hardware criptográfico.
SSLEvent	Atributo de control para eventos TLS.
SSLFIPSRequired	Utilice únicamente FIPS algoritmos certificados para la criptografía.
SSLKeyRepository 1	Ubicación del repositorio de claves TLS.
SSLKeyRepositoryPassword 1	La contraseña para el repositorio de claves TLS.
SSLKeyResetCount	TLS Recuento de reinicio de teclas.
SSLTasks 1	Número de subtarefas del servidor para procesar llamadas de TLS.
StatisticsInterval	Con qué frecuencia se deben registrar los datos de supervisión estadística.
StartStopEvent	Controla si se generan eventos de inicio y fin.
SyncPoint	Disponibilidad de Syncpoint.
 Canales TCP	Número máximo de canales activos o de clientes conectados que utilizan el protocolo « TCP/IP ».
 TCPKeepAlive	Si se debe utilizar TCP KEEPALIVE para comprobar el otro extremo de la conexión.

Tabla 593. Atributos para el gestor de colas (continuación)

Atributo	Descripción
 Nombre TCP	Nombre del sistema « TCP/IP » que está utilizando.
 TCPStackType	Cómo puede el creador del canal utilizar las direcciones TCP/IP.
TraceRouteRecording atributo	Controla el registro de la información de seguimiento de ruta.
TriggerInterval	Intervalo entre el disparador y el mensaje.
Versión	Versión.
XrCapability	Especifica si se admiten los comandos de telemetría.
Notas: Este atributo no se puede consultar mediante la llamada MQINQ y no se describe en esta sección. Consulte el Administrador de colas de cambios para obtener más información sobre este atributo.	

Tareas relacionadas

Especificar que solo FIPS se utilicen certificados CipherSpecs en tiempo de ejecución en el cliente MQI

Referencia relacionada

[Federal Information Processing Standards \(FIPS\) para AIX, Linux, and Windows](#)

AccountingConnOverride (MQLONG)

Esto permite a las aplicaciones alterar temporalmente el valor de los valores ACCTMQI y ACCTQDATA en el atributo Qmgr.

El valor puede ser uno de los siguientes:

MQMON_INHABILITADO

Las aplicaciones no pueden alterar temporalmente el valor de los atributos ACCTMQI y ACCTQ Qmgr utilizando el campo Opciones de la estructura MQCNO en la llamada MQCONN. Éste es el valor predeterminado.

MQMON_HABILITADO

Las aplicaciones pueden alterar temporalmente los atributos ACCTQ y ACCTMQI Qmgr utilizando el campo Opciones de la estructura MQCNO.

Los cambios en este valor sólo son efectivos para las conexiones con el gestor de colas después del cambio en el atributo.

Este atributo sólo está soportado en las plataformas siguientes:

-  IBM i
-   AIX and Linux
-  Windows

Para determinar el valor de este atributo, utilice el selector MQIA_ACCOUNTING_CONN_OVERRIDE con la llamada MQINQ.

AccountingInterval (MQLONG)

Especifica cuánto tiempo debe transcurrir antes de que se graben los registros de contabilidad intermedios (en segundos).

El valor es un entero en el rango de 0 a 604800, con un valor predeterminado de 1800 (30 minutos). Especifique 0 para desactivar los registros intermedios.

Este atributo sólo está soportado en las plataformas siguientes:

-  IBM i
-   AIX and Linux

-  Linux
-  Windows

Para determinar el valor de este atributo, utilice el selector MQIA_ACCOUNTING_INTERVAL con la llamada MQINQ.

ActivityConnOverride (MQLONG)

Esto permite a las aplicaciones alterar temporalmente el valor de ACTVTRC en el atributo del gestor de colas.

El valor puede ser uno de los siguientes:

MQMON_INHABILITADO

Las aplicaciones no pueden alterar temporalmente el valor del atributo de gestor de colas ACTVTRC utilizando el campo Opciones de la estructura MQCNO en la llamada MQCONN. Éste es el valor predeterminado.

MQMON_HABILITADO

Las aplicaciones pueden alterar temporalmente el atributo de gestor de colas ACTVTRC utilizando el campo Opciones de la estructura MQCNO.

Los cambios en este valor sólo son efectivos para las conexiones con el gestor de colas después del cambio en el atributo.

Este atributo sólo está soportado en [Multiplataformas](#).

Para determinar el valor de este atributo, utilice el selector MQIA_ACTIVITY_CONN_OVERRIDE con la llamada MQINQ.

ActivityTrace (MQLONG)

Esto controla la recopilación de registros de IBM MQ actividad de la aplicación MQI.

El valor puede ser uno de los siguientes:

MQMON_ON

Recopila IBM MQ el registro de actividad de la aplicación MQI.

MQMON_OFF

No recopile IBM MQ el registro de actividad de la aplicación MQI. Éste es el valor predeterminado.

Si establece el atributo de gestor de colas ACTVCON0 en ENABLED, este valor puede alterarse temporalmente para conexiones individuales utilizando el campo Opciones de la estructura MQCNO.

Los cambios en este valor sólo son efectivos para las conexiones con el gestor de colas después del cambio en el atributo.

Este atributo sólo está soportado en [Multiplataformas](#).

Para determinar el valor de este atributo, utilice el selector MQIA_ACTIVITY_TRACE con la llamada MQINQ.

AdoptNewMCACheck (MQLONG)

Define los elementos que se deben comprobar para determinar si se debe adoptar un MCA cuando se detecta un nuevo canal de entrada que tiene el mismo nombre que un MCA que ya está activo

El valor puede ser uno de los siguientes:

MQADOPT_CHECK_Q_MGR_NAME

Compruebe el nombre del gestor de colas.

MQADOPT_CHECK_NET_ADDR

Compruebe la dirección de red.

MQADOPT_CHECK_ALL

Compruebe el nombre del gestor de colas y la dirección de red. Si es posible, realice esta comprobación para evitar que los canales se apaguen, de forma involuntaria o maliciosa. Éste es el valor predeterminado.

MQADOPT_CHECK_NONE

No compruebe ningún elemento.

Los cambios en este atributo entrarán en vigor la próxima vez que un canal intente adoptar un canal.

 Este atributo sólo está soportado en z/OS.

Para determinar el valor de este atributo, utilice el selector MQIA_ADOPTNEWMCA_CHECK con la llamada MQINQ.

AdoptNewMCAType (MQLONG)

Esto especifica si se debe reiniciar automáticamente una instancia huérfana de un MCA de un tipo de canal concreto cuando se detecta una nueva solicitud de canal entrante que coincide con el atributo AdoptNewMCACheck

Es uno de los siguientes valores:

MQADOPT_TYPE_NO

No es necesario adoptar instancias de canal huérfanas. Éste es el valor predeterminado.

MQADOPT_TYPE_ALL

Adopta todos los tipos de canales.

Este atributo solo es compatible z/OS con.

Para determinar el valor de este atributo, utilice el selector MQIA_ADOPTNEWMCA_TYPE con la llamada MQINQ.

AlterationDate (MQCHAR12)

Es la fecha en la que se modificó por última vez la definición. El formato de la fecha es YYYY-MM-DD, relleno con dos espacios en blanco finales para que la longitud sea de 12 bytes.

Para determinar el valor de este atributo, utilice el selector MQCA_ALTERATION_DATE con la llamada MQINQ. La longitud de este atributo la proporciona MQ_DATE_LENGTH.

AlterationTime (MQCHAR8)

Es la hora en que se modificó por última vez la definición. El formato de la hora es HH.MM.SS.

Para determinar el valor de este atributo, utilice el selector MQCA_ALTERATION_TIME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_TIME_LENGTH.

AuthorityEvent (MQLONG)

 Desde IBM MQ 10.0.0, controla si se generan eventos de autorización exitosos y no exitosos.

Es uno de los siguientes valores:

MQEVR_DISABLED

Notificación de eventos desactivada.

MQEVR_ENABLED

Informes de eventos habilitados.

Para obtener más información sobre los eventos, consulte [Supervisión de eventos](#).

Para determinar el valor de este atributo, utilice el selector MQIA_AUTHORITY_EVENT con la llamada MQINQ.

AuthorityEventScope (MQLONG)

Esto controla el alcance de los eventos de autorización cuando está habilitado. Es uno de los siguientes valores:

MQAUSC_FALLOS

Notificación de eventos de error habilitada solo para fallos (predeterminado).

MQAUSC_TODAS_LAS_CONEXIONES

Solo informes de errores y eventos de conexión prolongada.

MQAUSC_TODAS_LAS_COMPROBACIONES

Informes de errores y todos los eventos extendidos.

Para obtener más información sobre los eventos, consulte [Supervisión de eventos](#).

Para determinar el valor de este atributo, utilice el selector MQIA_AUTHOREV_SCOPE con la llamada MQINQ.

 BridgeEvent (MQLONG)

Esto especifica si se generan eventos de puent IMS.

El valor puede ser uno de los siguientes:

MQEVR_ENABLED

Genere eventos de puent IMS, de la siguiente manera:

MQRC_PUENTE_INICIADO
MQRC_PUENTE_DETENIDO

MQEVR_DISABLED

No generar eventos de puent IMS; este es el valor predeterminado.

Este atributo solo es compatible z/OS con.

Para determinar el valor de este atributo, utilice el selector MQIA_BRIDGE_EVENT con la llamada MQINQ.

ChannelAutoDef (MQLONG)

Este atributo controla la definición automática de canales de tipo MQCHT_RECEIVER y MQCHT_SVRCONN. La definición automática de canales MQCHT_CLUSSDR siempre está habilitada. El valor puede ser uno de los siguientes:

MQCHAD_DISABLED

Definición automática de canal inhabilitada.

MQCHAD_ENABLED

Definición automática de canal habilitada.

 Este atributo sólo está soportado en [Multiplataformas](#) .

Para determinar el valor de este atributo, utilice el selector MQIA_CHANNEL_AUTO_DEF con la llamada MQINQ.

ChannelAutoDefEvent (MQLONG)

Esto controla si se generan sucesos de definición automática de canal. Se aplica a canales de tipo MQCHT_RECEIVER, MQCHT_SVRCONN y MQCHT_CLUSSDR. El valor puede ser uno de los siguientes:

MQEVR_DISABLED

Informes de sucesos inhabilitados.

MQEVR_ENABLED

Informes de sucesos habilitados.

Para obtener más información sobre los sucesos, consulte [Supervisión de sucesos](#).

 Este atributo sólo está soportado en [Multiplataformas](#) .

Para determinar el valor de este atributo, utilice el selector MQIA_CHANNEL_AUTO_DEF_EVENT con la llamada MQINQ.

ChannelAutoDefExit (MQCHARn)

Es el nombre de la salida de usuario para la definición automática de canal. Si este nombre no está en blanco y *ChannelAutoDef* tiene el valor MQCHAD_ENABLED, se llama a la salida cada vez que el gestor de colas está a punto de crear una definición de canal. Esto se aplica a los canales de tipo MQCHT_RECEIVER, MQCHT_SVRCONN y MQCHT_CLUSSDR. A continuación, la salida puede realizar una de las acciones siguientes:

- Cree la definición de canal sin cambios.
- Modifique los atributos de la definición de canal que se crea.
- Suprimir la creación del canal por completo.

Nota: Tanto la longitud como el valor de este atributo son específicos del entorno. Consulte la introducción a la estructura MQCD en [“MQCD-Definición de canal”](#) en la [página 1502](#) para obtener detalles sobre el valor de este atributo en diversos entornos.

 En z/OS, este atributo sólo se aplica a los canales de clúster emisor y de clúster receptor.

Para determinar el valor de este atributo, utilice el selector MQCA_CHANNEL_AUTO_DEF_EXIT con la llamada MQINQ. La longitud de este atributo la proporciona MQ_EXIT_NAME_LENGTH.

ChannelEvent (MQLONG)

Especifica si se generan sucesos de canal.

Es uno de los valores siguientes:

MQEVR_EXCEPCIÓN

Solo genere los siguientes sucesos de canal:

- MQRC_CHANNEL_ACTIVATED
- MQRC_CHANNEL_CONV_ERROR
- MQRC_CHANNEL_NOT_ACTIVATED
- MQRC_CHANNEL_STOPPED con lo siguiente ReasonQualifiers:

MQRQ_CHANNEL_STOPPED_ERROR
MQRQ_CHANNEL_STOPPED_RETRY
MQRQ_CHANNEL_STOPPED_DISABLED

MQRC_CHANNEL_STOPPED_BY_USER

MQEVR_ENABLED

Generar todos los sucesos de canal. Es decir, además de los generados por EXCEPTION, genera los siguientes sucesos de canal:

- MQRC_CHANNEL_STARTED
- MQRC_CHANNEL_STOPPED con lo siguiente ReasonQualifier:

MQRQ_CHANNEL_STOPPED_OK

MQEVR_DISABLED

No genere sucesos de canal; este es el valor predeterminado.

Para determinar el valor de este atributo, utilice el selector MQIA_CHANNEL_EVENT con la llamada MQINQ.

ChannelInitiatorControl (MQLONG)

Especifica si el iniciador de canal debe iniciarse cuando se inicia el gestor de colas.

Es uno de los valores siguientes:

MANUAL DE CONTROL DE MQSVC

El iniciador de canal no se debe iniciar automáticamente.

MQSVC_CONTROL_Q_MGR

El iniciador de canal se iniciará automáticamente cuando se inicie el gestor de colas.

Para determinar el valor de este atributo, utilice el selector MQIA_CHINIT_CONTROL con la llamada MQINQ.

ChannelMonitoring (MQLONG)

Este atributo especifica datos de supervisión en línea para canales.

El valor puede ser uno de los siguientes:

MQMON_NONE

Inhabilite la recopilación de datos para la supervisión de canales para todos los canales independientemente del valor del atributo de canal MONCHL. Éste es el valor predeterminado.

MQMON_OFF

Desactive la recopilación de datos de supervisión para los canales que especifiquen QMGR en el atributo de canal MONCHL.

MQMON_LOW

Active la recopilación de datos de supervisión con una proporción baja de recopilación de datos para canales que especifican QMGR en el atributo de canal MONCHL.

MQMON_MEDIO

Active la recopilación de datos de supervisión con una proporción moderada de recopilación de datos para canales que especifican QMGR en el atributo de canal MONCHL.

MQMON_HIGH

Active la recopilación de datos de supervisión con una proporción alta de recopilación de datos para canales que especifican QMGR en el atributo de canal MONCHL.

 En z/OS estos sistemas, al habilitar este parámetro simplemente se activa la recopilación de datos estadísticos, independientemente del valor que se seleccione. Especificar LOW, MEDIUM, o HIGH no influye en los resultados.

Para determinar el valor de este atributo, utilice el selector MQIA_MONITORING_CHANNEL con la llamada MQINQ.

ChannelStatistics (MQLONG)

Esto controla la recopilación de datos estadísticos para los canales.

El valor puede ser uno de los siguientes:

MQMON_NONE

Inhabilite la recopilación de datos para las estadísticas de canal para todos los canales independientemente del valor del atributo de canal STATCHL. Éste es el valor predeterminado.

MQMON_OFF

Desactive la recopilación de datos de estadísticas para los canales que especifiquen QMGR en el atributo de canal STATCHL.

MQMON_LOW

Active la recopilación de datos de estadísticas con una proporción baja de recopilación de datos para canales que especifican QMGR en el atributo de canal STATCHL.

MQMON_MEDIO

Active la recopilación de datos de estadísticas con una proporción moderada de recopilación de datos para canales que especifican QMGR en el atributo de canal STATCHL.

MQMON_HIGH

Active la recopilación de datos de estadísticas con una proporción alta de recopilación de datos para canales que especifican QMGR en el atributo de canal STATCHL.

Para la mayoría de los sistemas, se recomienda utilizar MEDIUM. Sin embargo, para un canal que procesa un gran volumen de mensajes cada segundo, es posible que desee reducir el nivel de muestreo seleccionando LOW. Además, para un canal que sólo procesa unos pocos mensajes, y para el que la información más actual es importante, es posible que desee seleccionar HIGH.

► z/OS En z/OS estos sistemas, al habilitar este parámetro simplemente se activa la recopilación de datos estadísticos, independientemente del valor que se seleccione. Especificar LOW, MEDIUM, o HIGH no influye en los resultados. Este parámetro debe estar habilitado para poder recopilar los registros de contabilidad de canal.

Para determinar el valor de este atributo, utilice el selector MQIA_STATISTICS_CHANNEL con la llamada MQINQ.

► z/OS ChinitAdapters (MQLONG)

Este es el número de subtareas del adaptador que se utilizarán para procesar IBM MQ las llamadas. El valor debe estar entre 0 y 9999, con un valor predeterminado de 8.

La proporción entre adaptadores y distribuidores (el atributo ChinitDispatchers) debe ser de aproximadamente 8 a 5. Sin embargo, si solo tiene unos pocos canales, no es necesario que reduzca el valor de este parámetro con respecto al valor predeterminado. Puede utilizar los siguientes valores: para un sistema de prueba, 8 (predeterminado); para un sistema de producción, 20. Lo ideal sería disponer de 20 adaptadores, lo que proporciona un mayor paralelismo de IBM MQ llamadas. Es importante para los mensajes persistentes. Es posible que un número menor de adaptadores sea más adecuado para mensajes no persistentes.

Este atributo solo es compatible z/OS con.

Para determinar el valor de este atributo, utilice el selector MQIA_CHINIT_ADAPTERS con la llamada MQINQ.

► z/OS ChinitDispatchers (MQLONG)

Este es el número de despachadores que se utilizarán para el iniciador del canal. El valor debe estar entre 0 y 9999, con un valor predeterminado de 5.

Como pauta general, se debe contar con un despachador por cada 50 canales actuales. Sin embargo, si solo tiene unos pocos canales, no es necesario que reduzca el valor de este atributo desde el valor predeterminado. Si utilizas TCP/IP, el número máximo de operadores que se pueden utilizar para los canales TCP/IP es 100, aunque especifique aquí un valor superior. Puede utilizar los siguientes ajustes: sistemas de prueba, 5 (el valor predeterminado); sistemas de producción, 20 (se necesitan 20 distribuidores para gestionar hasta 1000 canales activos).

Este atributo solo es compatible z/OS con.

Para determinar el valor de este atributo, utilice el selector MQIA_CHINIT_DISPATCHERS con la llamada MQINQ.

► z/OS ChinitTraceAutoStart (MQLONG)

Esto especifica si se debe iniciar automáticamente el seguimiento del iniciador del canal.

El valor puede ser uno de los siguientes:

MQTRAXSTR_SÍ

Iniciar automáticamente el seguimiento del iniciador del canal. Éste es el valor predeterminado.

MQTRAXSTR_NO

No iniciar automáticamente el seguimiento del iniciador del canal.

Este atributo solo es compatible z/OS con.

Para determinar el valor de este atributo, utilice el selector MQIA_CHINIT_TRACE_AUTO_START con la llamada MQINQ.

ChinitTraceTableSize (MQLONG)

Este es el tamaño del espacio de datos de rastreo del iniciador del canal (en MB).

El valor debe estar comprendido entre 0 y 2048, con un valor predeterminado de 2.

Nota: Siempre que utilice grandes espacios z/OS de datos, asegúrese de que dispone de suficiente almacenamiento auxiliar en su sistema para soportar cualquier actividad de paginación z/OS relacionada. También tendrá que aumentar el tamaño de los conjuntos de datos SYS1.DUMP.

Este atributo solo es compatible z/OS con.

Para determinar el valor de este atributo, utilice el selector MQIA_CHINIT_TRACE_TABLE_SIZE con la llamada MQINQ.

ClusterSenderMonitoringDefault (MQLONG)

Especifica el valor que debe sustituirse por el atributo ChannelMonitoring de los canales emisores de clúster definidos automáticamente.

El valor puede ser uno de los siguientes:

MQMON_Q_MGR

La recopilación de datos de supervisión en línea se hereda del valor del atributo **ChannelMonitoring** del gestor de colas. Éste es el valor predeterminado.

MQMON_OFF

La supervisión del canal está inhabilitada

MQMON_LOW

A menos que *ChannelMonitoring* sea MQMON_NONE, la supervisión se habilita con una tasa baja de recopilación de datos con un efecto mínimo en el rendimiento del sistema. No es probable que los datos recopilados sean los más actuales.

MQMON_MEDIO

A menos que *ChannelMonitoring* sea MQMON_NONE, la supervisión se habilita con una tasa moderada de recopilación de datos con un efecto limitado en el rendimiento del sistema.

MQMON_HIGH

A menos que *ChannelMonitoring* sea MQMON_NONE, la supervisión se habilita con una alta tasa de recopilación de datos con un efecto probable en el rendimiento del sistema. Los datos recopilados son los más actuales disponibles.

Para determinar el valor de este atributo, utilice el selector MQIA_MONITORING_AUTO_CLUSSDR con la llamada MQINQ.

ClusterSenderStatistics (MQLONG)

Puesto que los canales emisores de clúster se pueden definir automáticamente a partir de la definición de CLUSRCVR en el repositorio, no puede modificar el valor del atributo STATCHL para estos canales emisores de clúster definidos automáticamente utilizando el canal ALTER. Para estos canales, la decisión de recopilar datos de supervisión en línea se basa en el valor de este atributo de gestor de colas.

El valor puede ser uno de los siguientes:

MQMON_Q_MGR

La recopilación de datos de estadísticas para canales emisores de clúster definidos automáticamente se basa en el valor del atributo de gestor de colas STATCHL. Éste es el valor predeterminado.

MQMON_OFF

Desactive la recopilación de datos de estadísticas para los canales emisores de clúster definidos automáticamente.

MQMON_LOW

Habilite la recopilación de datos de estadísticas para los canales emisores de clúster definidos automáticamente con una proporción baja de recopilación de datos.

MQMON_MEDIO

Habilite la recopilación de datos de estadísticas para los canales emisores de clúster definidos automáticamente con una proporción moderada de recopilación de datos.

MQMON_HIGH

Habilite la recopilación de datos de estadísticas para los canales de clúster emisor definidos automáticamente con una proporción alta de recopilación de datos.

Para la mayoría de los sistemas recomendamos MEDIO. Sin embargo, para un canal emisor de clúster definido automáticamente que procesa un gran volumen de mensajes cada segundo, es posible que desee reducir el nivel de muestreo seleccionando LOW. Además, para un canal que sólo procesa unos pocos mensajes, y para el que la información más actual es importante, es posible que desee seleccionar HIGH.

 En z/OS estos sistemas, al habilitar este parámetro simplemente se activa la recopilación de datos estadísticos, independientemente del valor que se seleccione. Especificar LOW, MEDIUM, o HIGH no influye en los resultados. Este parámetro debe estar habilitado para poder recopilar los registros de contabilidad de canal.

Para determinar el valor de este atributo, utilice el selector MQIA_STATISTICS_AUTO_CLUSSDR con la llamada MQINQ.

ClusterWorkloadData (MQCHAR32)

Es una serie de caracteres de 32 bytes definida por el usuario que se pasa a la salida de carga de trabajo del clúster cuando se llama. Si no hay datos para pasar a la salida, la serie está en blanco.

Para determinar el valor de este atributo, utilice el selector MQCA_CLUSTER_WORKLOAD_DATA con la llamada MQINQ.

ClusterWorkloadExit (MQCHARn)

Es el nombre de la salida de usuario para la gestión de carga de trabajo de clúster. Si este nombre no está en blanco, se llama a la salida cada vez que se transfiere un mensaje a una cola de clúster o se mueve de una cola de clúster emisor a otra. A continuación, la salida puede aceptar la instancia de cola seleccionada por el gestor de colas como destino del mensaje o seleccionar otra instancia de cola.

Nota: Tanto la longitud como el valor de este atributo son específicos del entorno.

Para determinar el valor de este atributo, utilice el selector MQCA_CLUSTER_WORKLOAD_EXIT con la llamada MQINQ. La longitud de este atributo la proporciona MQ_EXIT_NAME_LENGTH.

ClusterWorkloadLength (MQLONG)

Es la longitud máxima de los datos de mensaje que se pasan a la salida de carga de trabajo del clúster. La longitud real de los datos pasados a la salida es el mínimo de lo siguiente:

- Longitud del mensaje.
- El atributo **MaxMsgLength** del gestor de colas.
- El atributo **ClusterWorkloadLength**.

Para determinar el valor de este atributo, utilice el selector MQIA_CLUSTER_WORKLOAD_LENGTH con la llamada MQINQ.

Canales CLWLMRU (MQLONG)

Especifica el número máximo de canales de clúster utilizados más recientemente, que se deben tener en cuenta para que los utilice el algoritmo de elección de carga de trabajo de clúster.

Es un valor comprendido entre 1 y 999999999.

Para determinar el valor de este atributo, utilice el selector MQIA_CLWL_MRU_CHANNELS con la llamada MQINQ.

CLWLUseQ (MQLONG)

Especifica si se deben utilizar colas remotas para la carga de trabajo del clúster.

El valor puede ser uno de los siguientes:

MQCLWL_USEQ_ANY

Utilice colas locales y remotas.

MQCLWL_USEQ_LOCAL

No utilice colas remotas. Éste es el valor predeterminado.

Para determinar el valor de este atributo, utilice el selector MQIA_CLWL_USEQ con la llamada MQINQ.

CodedCharSetId (MQLONG)

Esto define el juego de caracteres que utiliza el gestor de colas para todos los campos de cadena de caracteres definidos en el directorio de objetos (MQI), como los nombres de los objetos y la fecha y hora de creación de las colas. El juego de caracteres debe ser uno que tenga caracteres de un solo byte para los caracteres que son válidos en los nombres de objeto. No se aplica a los datos de aplicación transportados en el mensaje. El valor depende del entorno:

- En z/OS, el valor se establece a partir de los parámetros del sistema cuando se inicia el gestor de colas; el valor predeterminado es 500.
- En Windows, el valor es el CODEPAGE primario del usuario que crea el gestor de colas.
- En IBM i, el valor es el que se establece en el entorno cuando se crea por primera vez el gestor de colas.
- En AIX and Linux, el valor es el valor predeterminado CODESET para el entorno local del usuario que crea el gestor de colas.

Para determinar el valor de este atributo, utilice el selector MQIA_CODED_CHAR_SET_ID con la llamada MQINQ.

CommandEvent (MQLONG)

Especifica si se generan sucesos de mandato, como se indica a continuación:

MQEVR_DISABLED

No generar sucesos de mandato. Este es el valor predeterminado.

MQEVR_ENABLED

Generar sucesos de mandato.

MQEVR_SIN_VISUALIZACIÓN

Los sucesos de mandato se generan para todos los mandatos satisfactorios distintos de MQINQ.

Para determinar el valor de este atributo, utilice el selector MQIA_COMMAND_EVENT con la llamada MQINQ.

CommandInputQName (MQCHAR48)

Es el nombre de la cola de entrada de mandatos definida en el gestor de colas local. Es una cola a la que los usuarios pueden enviar mandatos, si están autorizados a hacerlo. El nombre de la cola depende del entorno:

- En z/OS, el nombre de la cola es SYSTEM.COMMAND.INPUT; se le pueden enviar los mandatos MQSC y PCF. Consulte [Mandatos MQSC](#) para obtener detalles de los mandatos MQSC y [Definiciones de los formatos de mandatos programables](#) para obtener detalles de los mandatos PCF.
- En todos los demás entornos, el nombre de la cola es SYSTEM.ADMIN.COMMAND.QUEUEy solo se le pueden enviar mandatos PCF. Sin embargo, se puede enviar un mandato MQSC a esta cola si el mandato MQSC está entre un mandato PCF de tipo MQCMD_ESCAPE. Consulte [Escape](#) para obtener información sobre el mandato Escape.

Para determinar el valor de este atributo, utilice el selector MQCA_COMMAND_INPUT_Q_NAME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_Q_NAME_LENGTH.

CommandLevel (MQLONG)

Esto indica el nivel de comandos de control del sistema que admite el gestor de colas. Puede tener uno de los valores siguientes:

MQCMDL_LEVEL_800

Nivel 800 de comandos de control del sistema.

Este valor es devuelto por las siguientes versiones:

- IBM MQ for AIX 8.0
- IBM MQ for HP-UX 8.0
- IBM MQ for IBM i 8.0
- IBM MQ for Linux 8.0
- IBM MQ for Solaris 8.0
- IBM MQ for Windows 8.0
- IBM MQ for z/OS 8.0

MQCMDL_LEVEL_801

Nivel 801 de comandos de control del sistema.

Este valor es devuelto por las siguientes versiones:

- IBM MQ for AIX 8.0.0 Fix Pack 2
- IBM MQ for HP-UX 8.0.0 Fix Pack 2
- IBM MQ for IBM i 8.0.0 Fix Pack 2
- IBM MQ for Linux 8.0.0 Fix Pack 2
- IBM MQ for Solaris 8.0.0 Fix Pack 2

MQCMDL_LEVEL_802

Nivel 802 de comandos de control del sistema.

Este valor es devuelto por las siguientes versiones:

- IBM MQ for AIX 8.0.0 Fix Pack 3
- IBM MQ for HP-UX 8.0.0 Fix Pack 3
- IBM MQ for IBM i 8.0.0 Fix Pack 3
- IBM MQ for Linux 8.0.0 Fix Pack 3
- IBM MQ for Solaris 8.0.0 Fix Pack 3
- IBM MQ for Windows 8.0.0 Fix Pack 3
- IBM MQ for HPE NonStop 8.1.0

MQCMDL_LEVEL_900

Nivel 900 de comandos de control del sistema.

Este valor es devuelto por las siguientes versiones:

- IBM MQ for AIX 9.0
- IBM MQ for HP-UX 9.0
- IBM MQ for IBM i 9.0
- IBM MQ for Linux 9.0
- IBM MQ for Solaris 9.0
- IBM MQ for Windows 9.0
- IBM MQ for z/OS 9.0

MQCMDL_LEVEL_901

Nivel 901 de comandos de control del sistema.

Este valor es devuelto por las siguientes versiones:

- IBM MQ for Linux 9.0.1
- IBM MQ for Windows 9.0.1
- IBM MQ for z/OS 9.0.1

MQCMDL_LEVEL_902

Nivel 902 de comandos de control del sistema.

Este valor es devuelto por las siguientes versiones:

- IBM MQ for Linux 9.0.2
- IBM MQ for Windows 9.0.2
- IBM MQ for z/OS 9.0.2

MQCMDL_LEVEL_903

Nivel 903 de comandos de control del sistema.

Este valor es devuelto por las siguientes versiones:

- IBM MQ for Linux 9.0.3
- IBM MQ for Windows 9.0.3
- IBM MQ for z/OS 9.0.3

MQCMDL_LEVEL_904

Nivel 904 de comandos de control del sistema.

Este valor es devuelto por las siguientes versiones:

- IBM MQ for AIX 9.0.4
- IBM MQ for Linux 9.0.4
- IBM MQ for Windows 9.0.4
- IBM MQ for z/OS 9.0.4

MQCMDL_LEVEL_905

Nivel 905 de comandos de control del sistema.

Este valor es devuelto por las siguientes versiones:

- IBM MQ for AIX 9.0.5
- IBM MQ for Linux 9.0.5
- IBM MQ for Windows 9.0.5
- IBM MQ for z/OS 9.0.5

MQCMDL_LEVEL_910

Nivel 910 de comandos de control del sistema.

Este valor es devuelto por las siguientes versiones:

- IBM MQ for AIX 9.1
- IBM MQ for IBM i 9.1
- IBM MQ for Linux 9.1
- IBM MQ for Solaris 9.1
- IBM MQ for Windows 9.1
- IBM MQ for z/OS 9.1

MQCMDL_LEVEL_911

Nivel 911 de comandos de control del sistema.

Este valor es devuelto por las siguientes versiones:

- IBM MQ for AIX 9.1.1
- IBM MQ for Linux 9.1.1
- IBM MQ for Windows 9.1.1
- IBM MQ for z/OS 9.1.1

MQCMDL_LEVEL_912

Nivel 912 de comandos de control del sistema.

Este valor es devuelto por las siguientes versiones:

- IBM MQ for AIX 9.1.2
- IBM MQ for Linux 9.1.2
- IBM MQ for Windows 9.1.2
- IBM MQ for z/OS 9.1.2

MQCMDL_LEVEL_913

Nivel 913 de comandos de control del sistema.

Este valor es devuelto por las siguientes versiones:

- IBM MQ for AIX 9.1.3
- IBM MQ for Linux 9.1.3
- IBM MQ for Windows 9.1.3
- IBM MQ for z/OS 9.1.3

MQCMDL_LEVEL_914

Nivel 914 de comandos de control del sistema.

Este valor es devuelto por las siguientes versiones:

- IBM MQ for AIX 9.1.4
- IBM MQ for Linux 9.1.4
- IBM MQ for Windows 9.1.4
- IBM MQ for z/OS 9.1.4

MQCMDL_LEVEL_915

Nivel 915 de comandos de control del sistema.

Este valor es devuelto por las siguientes versiones:

- IBM MQ for AIX 9.1.5
- IBM MQ for Linux 9.1.5
- IBM MQ for Windows 9.1.5
- IBM MQ for z/OS 9.1.5

MQCMDL_LEVEL_910

Nivel 910 de comandos de control del sistema.

Este valor es devuelto por las siguientes versiones:

- IBM MQ for AIX 9.1
- IBM MQ for IBM i 9.1
- IBM MQ for Linux 9.1
- IBM MQ for Windows 9.1
- IBM MQ for z/OS 9.1

MQCMDL_LEVEL_920

Nivel 920 de comandos de control del sistema.

Este valor es devuelto por las siguientes versiones:

- IBM MQ for AIX 9.2
- IBM MQ for IBM i 9.2
- IBM MQ for Linux 9.2
- IBM MQ for Windows 9.2
- IBM MQ for z/OS 9.2

MQCMDL_LEVEL_921

Nivel 921 de comandos de control del sistema.

Este valor es devuelto por las siguientes versiones:

- IBM MQ for AIX 9.2.1
- IBM MQ for Linux 9.2.1
- IBM MQ for Windows 9.2.1
- IBM MQ for z/OS 9.2.1

MQCMDL_LEVEL_922

Nivel 922 de comandos de control del sistema.

Este valor es devuelto por las siguientes versiones:

- IBM MQ for AIX 9.2.2
- IBM MQ for Linux 9.2.2
- IBM MQ for Windows 9.2.2
- IBM MQ for z/OS 9.2.2

MQCMDL_LEVEL_923

Nivel 923 de comandos de control del sistema.

Este valor es devuelto por las siguientes versiones:

- IBM MQ for AIX 9.2.3
- IBM MQ for Linux 9.2.3
- IBM MQ for Windows 9.2.3
- IBM MQ for z/OS 9.2.3

MQCMDL_LEVEL_924

Nivel 924 de comandos de control del sistema.

Este valor es devuelto por las siguientes versiones:

- IBM MQ for AIX 9.2.4
- IBM MQ for Linux 9.2.4
- IBM MQ for Windows 9.2.4
- IBM MQ for z/OS 9.2.4

MQCMDL_LEVEL_925

Nivel 925 de comandos de control del sistema.

Este valor es devuelto por las siguientes versiones:

- IBM MQ for AIX 9.2.5
- IBM MQ for Linux 9.2.5
- IBM MQ for Windows 9.2.5
- IBM MQ for z/OS 9.2.5

MQCMDL_LEVEL_930

Nivel 930 de comandos de control del sistema.

Este valor es devuelto por las siguientes versiones:

- IBM MQ for AIX 9.3
- IBM MQ for IBM i 9.3
- IBM MQ for Linux 9.3
- IBM MQ for Windows 9.3
- IBM MQ for z/OS 9.3

MQCMDL_LEVEL_931

Nivel 931 de comandos de control del sistema.

Este valor es devuelto por las siguientes versiones:

- IBM MQ for AIX 9.3.1
- IBM MQ for Linux 9.3.1
- IBM MQ for Windows 9.3.1
- IBM MQ for z/OS 9.3.1

MQCMDL_LEVEL_932

Nivel 932 de comandos de control del sistema.

Este valor es devuelto por las siguientes versiones:

- IBM MQ for AIX 9.3.2
- IBM MQ for Linux 9.3.2
- IBM MQ for Windows 9.3.2
- IBM MQ for z/OS 9.3.2

MQCMDL_LEVEL_933

Nivel 933 de comandos de control del sistema.

Este valor es devuelto por las siguientes versiones:

- IBM MQ for AIX 9.3.3
- IBM MQ for Linux 9.3.3
- IBM MQ for Windows 9.3.3
- IBM MQ for z/OS 9.3.3

MQCMDL_LEVEL_934

Nivel 934 de comandos de control del sistema.

Este valor es devuelto por las siguientes versiones:

- IBM MQ for AIX 9.3.4
- IBM MQ for Linux 9.3.4
- IBM MQ for Windows 9.3.4
- IBM MQ for z/OS 9.3.4

MQCMDL_LEVEL_935

Nivel 935 de comandos de control del sistema.

Este valor es devuelto por las siguientes versiones:

- IBM MQ for AIX 9.3.5
- IBM MQ for Linux 9.3.5
- IBM MQ for Windows 9.3.5
- IBM MQ for z/OS 9.3.5

MQCMDL_LEVEL_940

Nivel 940 de comandos de control del sistema.

Este valor es devuelto por las siguientes versiones:

- IBM MQ for AIX 9.4.0
- IBM MQ for Linux 9.4.0
- IBM MQ for Windows 9.4.0
- IBM MQ for z/OS 9.4.0

El conjunto de comandos de control del sistema que corresponde a un valor concreto del **CommandLevel** atributo varía en función del valor del **Platform** atributo; ambos deben utilizarse para decidir qué comandos de control del sistema son compatibles.

Para determinar el valor de este atributo, utilice el MQIA_COMMAND_LEVEL selector con la MQINQ llamada.

CommandServerControl (MQLONG)

Especifica si el servidor de mandatos debe iniciarse cuando se inicia el gestor de colas.

El valor puede ser cualquiera de los valores siguientes:

MANUAL DE CONTROL DE MQSVC

El servidor de mandatos no se debe iniciar automáticamente.

MQSVC_CONTROL_Q_MGR

El servidor de mandatos se iniciará automáticamente cuando se inicie el gestor de colas.

Este atributo no está soportado en z/OS.

Para determinar el valor de este atributo, utilice el selector MQIA_CMD_SERVER_CONTROL con la llamada MQINQ.

ConfigurationEvent (MQLONG)

Controla si se generan sucesos de configuración.

Para determinar el valor de este atributo, utilice el selector MQIA_CONFIGURATION_EVENT con la llamada MQINQ.

El valor puede ser cualquiera de los valores siguientes:

MQEVN_DISABLED

Informes de sucesos inhabilitados.

MQEVN_ENABLED

Informes de sucesos habilitados.

Multi CurrentQFileSize (MQLONG)

El tamaño actual del archivo de cola en megabytes, redondeado al megabyte más cercano.

Tabla 594. Tipos de colas a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X			

El valor de este atributo de estado de la cola es el tamaño actual de la cola, redondeado al megabyte más cercano. Para una nueva cola con atributos predeterminados, el valor de **CurrentQFileSize** es 1.

El valor máximo de este atributo es 99 999 9999 MB y no hay ningún valor predeterminado para este atributo.

Multi CurrentMaxQFileSize (MQLONG)

El tamaño máximo actual al que puede crecer el archivo de cola, redondeado al megabyte más cercano, dado el tamaño de bloque actual en uso en una cola.

Tabla 595. Tipos de colas a los que se aplica este atributo

Local	Modelo	Alias	Remoto	Clúster
X	X			

El uso de este campo tiene dos funciones:

- Si establece **MaxQFileSize** en el valor predeterminado para el tamaño de bloque actual, **CurrentMaxQFileSize** muestra el valor real al que equivale el valor predeterminado.
- Si **CurrentMaxQFileSize** no coincide **MaxQFileSize** con, sabrá que la cola debe vaciarse para adoptar una granularidad mayor.

Nota: Consulte [Modificación de IBM MQ archivos de cola](#) para obtener más información sobre cómo cambiar el tamaño de los archivos de cola y el tamaño y la granularidad de los bloques.

El valor máximo de este atributo es 99 999 9999 MB y no hay ningún valor predeterminado. El valor es el valor máximo establecido actualmente; para una nueva cola con los atributos predeterminados, el valor de **CurrentMaxQFileSize** es 2 088 960 MB.

DeadLetterQName (MQCHAR48)

Este es el nombre de una cola definida en el gestor de colas local como la cola de mensajes no entregados (mensajes no entregados). Los mensajes se envían a esta cola si no pueden enrutarse a su destino correcto.

Por ejemplo, los mensajes se colocan en esta cola cuando:

- Llega un mensaje a un gestor de colas, destinado a una cola que aún no está definida en ese gestor de colas
- Llega un mensaje al gestor de colas, pero la cola a la que va destinado no puede recibirlo porque, posiblemente:
 - La cola está llena
 - Las solicitudes de envío se inhiben
 - El nodo emisor no tiene autoridad para colocar mensajes en la cola

Las aplicaciones también pueden colocar mensajes en la cola de mensajes no entregados.

Los mensajes de informe se tratan de la misma manera que los mensajes normales; si el mensaje de informe no se puede entregar a su cola de destino (normalmente la cola especificada por el *ReplyToQ* campo en el descriptor de mensajes del mensaje original), el mensaje de informe se coloca en la cola de mensajes no entregados (dead-letter).

Nota: Los mensajes que han superado su tiempo de caducidad (véase [MQMD - Campo de caducidad](#)) **no** se transfieren a esta cola cuando se descartan. Sin embargo, si la aplicación de envío lo solicita, se sigue generando un mensaje de informe de caducidad (MQRO_EXPIRATION) y se envía a la *ReplyToQ* cola.

Los mensajes no se colocan en la cola de mensajes no entregados (mensajes no entregados) cuando la aplicación que emitió la solicitud de inserción ha sido notificada de forma sincrónica del problema mediante el código de motivo devuelto por la llamada MQPUT o MQPUT1 (por ejemplo, un mensaje insertado en una cola local para la que se inhiben las solicitudes de inserción).

Los mensajes de la cola de mensajes no entregados (mensajes no entregados) a veces tienen los datos del mensaje de la aplicación precedidos por una estructura MQDLH. Esta estructura contiene información adicional que indica por qué el mensaje se colocó en la cola de mensajes no entregados (mensajes no entregados). Consulte [“MQDLH - Cabecera de mensajes no entregados”](#) en la página 311 para obtener más detalles sobre esta estructura.

Esta cola debe ser una cola local, con un **Usage** atributo MQUS_NORMAL.

Si un gestor de colas no admite una cola de mensajes no entregados (mensajes no entregados) o no se ha definido ninguna, el nombre estará en blanco. Todos los gestores IBM MQ de colas admiten una cola de mensajes no entregados (dead-letter), pero por defecto no está definida.

Si la cola de mensajes no entregados (mensajes no entregados) no está definida, está llena o no se puede utilizar por algún otro motivo, los mensajes que habrían sido transferidos a ella por un agente de canal de mensajes se conservan en la cola de transmisión.

Para determinar el valor de este atributo, utilice el selector MQCA_DEAD_LETTER_Q_NAME con la llamada MQINQ. La longitud de este atributo viene dada por MQ_Q_NAME_LENGTH.

DefClusterXmitQueueType (MQLONG)

El atributo « DefClusterXmitQueueType » controla qué cola de transmisión seleccionan por defecto los canales «cluster-sender» para recibir mensajes y enviarlos a los canales «cluster-receiver».

Los valores de **DefClusterXmitQueueType** son MQCLXQ_SCTQ o MQCLXQ_CHANNEL.

MQCLXQ_SCTQ

Todos los canales de clúster emisor envían mensajes de SYSTEM.CLUSTER.TRANSMIT.QUEUE. El correlID de los mensajes colocados en la cola de transmisión identifica el canal de clúster emisor al que va destinado el mensaje.

La variable SCTQ se establece al definir un gestor de colas.

MQCLXQ_CHANNEL

Cada canal de clúster emisor envía mensajes desde una cola de transmisión diferente.

Cada cola de transmisión se crea como una cola dinámica permanente de la cola modelo SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE.

Si el atributo del gestor de colas, DefClusterXmitQueueType, está establecido en CHANNEL, la configuración predeterminada cambia a los canales de clúster emisor que se están asociando con colas de transmisión de clúster individuales. SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUELas colas de transmisión son colas dinámicas permanentes creadas a partir de la cola modelo. Cada cola de transmisión está asociada a un canal de clúster emisor. Cuando un canal de clúster emisor presta servicio a una cola de transmisión de clúster, la cola de transmisión contiene mensajes únicamente para un gestor de colas de un clúster. Puede configurar clústeres de modo que cada gestor de colas de un clúster sólo contenga una cola de clúster. En este caso, el tráfico de mensajes de un gestor de colas a cada cola de clúster se transfiere por separado de los mensajes a otras colas.

Para consultar el valor, llame a MQINQo envíe un mandato Consultar gestor de colas (MQCMD_INQUIRE_Q_MGR) PCF, estableciendo el selector MQIA_DEF_CLUSTER_XMIT_Q_TYPE . Para cambiar el valor, envíe un mandato PCF Cambiar gestor de colas (MQCMD_CHANGE_Q_MGR), estableciendo el selector MQIA_DEF_CLUSTER_XMIT_Q_TYPE .

Referencia relacionada

[Cambiar gestor de colas](#)

[Consultar gestor de colas](#)

[“MQINQ - Consultar atributos de objeto” en la página 707](#)

La MQINQ llamada devuelve una matriz de números enteros y un conjunto de cadenas de caracteres que contienen los atributos de un objeto.

DefXmitQName (MQCHAR48)

Es el nombre de la cola de transmisión que se utiliza para la transmisión de mensajes a gestores de colas remotos, si no hay ninguna otra indicación de qué cola de transmisión utilizar.

Si no hay ninguna cola de transmisión predeterminada, el nombre está totalmente en blanco. El valor inicial de este atributo está en blanco.

Para determinar el valor de este atributo, utilice el selector MQCA_DEF_XMIT_Q_NAME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_Q_NAME_LENGTH.

DistLists (MQLONG)

Esto indica si el gestor de colas local da soporte a listas de distribución en las llamadas MQPUT y MQPUT1 . Es uno de los valores siguientes:

MQDL_COMPATIBLE

Listas de distribución soportadas.

MQDL_NO_COMPATIBLE

Listas de distribución no soportadas.

Para determinar el valor de este atributo, utilice el selector MQIA_DIST_LISTS con la llamada MQINQ.

DNSGroup (MQCHAR18)

Este parámetro ya no se utiliza. Vea [qué ha cambiado en IBM MQ 8.0](#).

Este atributo solo es compatible z/OS con.

Para determinar el valor de este atributo, utilice el selector MQCA_DNS_GROUP con la llamada MQINQ. La longitud de este atributo viene dada por MQ_DNS_GROUP_NAME_LENGTH.

DNSWLM (MQLONG)

Este parámetro ya no se utiliza. Vea [qué ha cambiado en IBM MQ 8.0](#).

El valor puede ser uno de los siguientes:

MQDNSWLM_SÍ

Este valor puede verse en un gestor de colas migrado desde una versión anterior. El valor se ignora.

MQDNSWLM_NO

Este es el único valor admitido por el gestor de colas.

Este atributo solo es compatible z/OS con.

Para determinar el valor de este atributo, utilice el selector MQIA_DNS_WLM con la llamada MQINQ.

ExpiryInterval (MQLONG)

Esto indica la frecuencia con la que el gestor de colas explora las colas en busca de mensajes caducados. Es un intervalo de tiempo en segundos comprendido entre 1 y 99 999 999, o el siguiente valor especial:

MQEXPI_OFF

El gestor de colas no explora las colas en busca de mensajes caducados.

Para determinar el valor de este atributo, utilice el selector MQIA_EXPIRY_INTERVAL con la llamada MQINQ.

 Este atributo sólo está soportado en z/OS.

IGQPutAuthority (MQLONG)

Este atributo sólo se aplica si el gestor de colas local es miembro de un grupo de compartición de colas. Indica el tipo de comprobación de autorización que se realiza cuando el agente de transferencia a colas dentro del grupo local (agente IGQ) elimina un mensaje de la cola de transmisión compartida y coloca el mensaje en una cola local. El valor puede ser uno de los siguientes:

MQIGQPA_PREDETERMINADO

El identificador de usuario que se comprueba para la autorización es el valor del campo *UserIdentifier* en el MQMD *separado* que está asociado con el mensaje cuando el mensaje está en la cola de transmisión compartida. Es el identificador de usuario del programa que ha colocado el mensaje en la cola de transmisión compartida, y normalmente es el mismo que el identificador de usuario bajo el que se ejecuta el gestor de colas remoto.

Si el perfil RESLEVEL indica que se debe comprobar más de un identificador de usuario, también se comprueba el identificador de usuario del agente de IGQ local (*IGQUserId*).

CONTEXTO_MQIGQPA

El identificador de usuario que se comprueba para la autorización es el valor del campo *UserIdentifier* en el MQMD *separado* que está asociado con el mensaje cuando el mensaje está

en la cola de transmisión compartida. Es el identificador de usuario del programa que ha colocado el mensaje en la cola de transmisión compartida, y normalmente es el mismo que el identificador de usuario bajo el que se ejecuta el gestor de colas remoto.

Si el perfil RESLEVEL indica que se debe comprobar más de un identificador de usuario, también se comprueban el identificador de usuario del agente de IGQ local (*IGQUserId*) y el valor del campo *UserIdentifier* en el MQMD incorporado. Este último identificador de usuario suele ser el identificador de usuario de la aplicación que ha originado el mensaje.

MQIGQPA_SOLO_IGQ

El identificador de usuario que se comprueba para la autorización es el identificador de usuario del agente de IGQ local (*IGQUserId*).

Si el perfil RESLEVEL indica que se debe comprobar más de un identificador de usuario, este identificador de usuario se utiliza para todas las comprobaciones.

MQIGQPA_ALTERNATIVA_O_IGQ

El identificador de usuario que se comprueba para la autorización es el identificador de usuario del agente de IGQ local (*IGQUserId*).

Si el perfil RESLEVEL indica que se debe comprobar más de un identificador de usuario, también se comprueba el valor del campo *UserIdentifier* en el MQMD incorporado. Este identificador de usuario suele ser el identificador de usuario de la aplicación que ha originado el mensaje.

Para determinar el valor de este atributo, utilice el selector MQIA_IGQ_PUT_AUTHORITY con la llamada MQINQ.

 Este atributo sólo está soportado en z/OS.

IGQUserId (MQLONG)

Este atributo sólo es aplicable si el gestor de colas local es miembro de un grupo de compartición de colas. Especifica el identificador de usuario que está asociado con el agente de transferencia a colas dentro del grupo local (agente IGQ). Este identificador es uno de los identificadores de usuario que se pueden comprobar para la autorización cuando el agente de IGQ coloca mensajes en colas locales. Los identificadores de usuario reales que se comprueban dependen del valor del atributo **IGQPutAuthority** y de las opciones de seguridad externas.

Si *IGQUserId* está en blanco, no se asocia ningún identificador de usuario con el agente de IGQ y no se realiza la comprobación de autorización correspondiente (aunque es posible que se sigan comprobando otros identificadores de usuario para la autorización).

Para determinar el valor de este atributo, utilice el selector MQCA_IGQ_USER_ID con la llamada MQINQ. La longitud de este atributo la proporciona MQ_USER_ID_LENGTH.

 Este atributo sólo está soportado en z/OS.

InhibitEvent (MQLONG)

Esto controla si se generan sucesos de inhibición (inhibir obtención e inhibir colocación). El valor puede ser uno de los siguientes:

MQEVR_DISABLED

Informes de sucesos inhabilitados.

MQEVR_ENABLED

Informes de sucesos habilitados.

Para obtener más información sobre los sucesos, consulte [Supervisión de sucesos](#).

Para determinar el valor de este atributo, utilice el selector MQIA_INHIBID_EVENT con la llamada MQINQ.

En z/OS, no puede utilizar la llamada MQINQ para determinar el valor de este atributo.

IntraGroupQueuing (MQLONG)

Este atributo solo se aplica si el gestor de colas local es miembro de un grupo de uso compartido de colas. Indica si la cola intragrupo está habilitada para el grupo de uso compartido de colas.

El valor puede ser uno de los siguientes:

MQIGQ_DESACTIVADO

Todos los mensajes destinados a otros gestores de colas del grupo de intercambio de colas se transmiten utilizando canales convencionales.

MQIGQ_HABILITADO

Los mensajes destinados a otros gestores de colas del grupo de intercambio de colas se transmiten utilizando la cola de transmisión compartida si se cumple la siguiente condición:

- La longitud de los datos del mensaje más la cabecera de transmisión no supera los 63 KB (64 512 bytes).

Se recomienda asignar un espacio algo mayor que el tamaño de MQXQH para el encabezado de transmisión; la constante MQ_MSG_HEADER_LENGTH se proporciona para este fin.

Si no se cumple esta condición, el mensaje se transmite utilizando los canales convencionales.

Nota: Cuando se habilita la cola intragrupo, el orden de los mensajes transmitidos mediante la cola de transmisión compartida no se conserva en relación con los transmitidos mediante canales convencionales.

Para determinar el valor de este atributo, utilice el selector MQIA_INTRA_GROUP_QUEUING con la llamada MQINQ.

IPAddressVersion (MQLONG)

Especifica qué versión de dirección IP, ya sea IPv4 o IPv6, se utiliza.

Este atributo sólo es relevante para los sistemas que ejecutan IPv4 y IPv6 y sólo afecta a los canales definidos como que tienen un *TransportType* de MQXPY_TCP cuando se cumple una de las condiciones siguientes:

- El *ConnectionName* del canal es un nombre de host que se resuelve en una dirección IPv4 y IPv6 y no se especifica su parámetro **LocalAddress**.
- Los *ConnectionName* y *LocalAddress* del canal son ambos nombres de host que se resuelven en las direcciones IPv4 y IPv6.

El valor puede ser cualquiera de los valores siguientes:

MQIPADDR_IPV4

IPv4.

MQIPADDR_IPV6

IPv6.

Para determinar el valor de este atributo, utilice el selector MQIA_IP_ADDRESS_VERSION con la llamada MQINQ.

ListenerTimer (MQLONG)

Este es el intervalo de tiempo (en segundos) entre IBM MQ los intentos de reiniciar el listener si se ha producido un error de tipo « APPC » o « TCP/IP ». El valor debe estar entre 5 y 9999, con un valor predeterminado de 60.

Este atributo solo es compatible z/OS con.

Para determinar el valor de este atributo, utilice el selector MQIA_LISTENER_TIMER con la llamada MQINQ.

LocalEvent (MQLONG)

Esto controla si se generan sucesos de error locales. El valor puede ser uno de los siguientes:

MQEVR_DISABLED

Informes de sucesos inhabilitados.

MQEVR_ENABLED

Informes de sucesos habilitados.

Para obtener más información sobre los sucesos, consulte [Supervisión de sucesos](#).

Para determinar el valor de este atributo, utilice el selector MQIA_LOCAL_EVENT con la llamada MQINQ.

En z/OS, no puede utilizar la llamada MQINQ para determinar el valor de este atributo.

LoggerEvent (MQLONG)

Esto controla si se generan sucesos de registro de recuperación. El valor puede ser uno de los siguientes:

MQEVR_DISABLED

Informes de sucesos inhabilitados.

MQEVR_ENABLED

Informes de sucesos habilitados.

Para obtener más información sobre los sucesos, consulte [Supervisión de sucesos](#).

Para determinar el valor de este atributo, utilice el selector MQIA_LOGGER_EVENT con la llamada MQINQ.

 Este atributo sólo está soportado en [Multiplataformas](#).

 LUGroupName (MQCHAR8)

Este es el nombre genérico de la unidad lógica (LU) para el receptor de la unidad lógica (6.2) que gestiona las transmisiones entrantes para el grupo de uso compartido de colas. Si deja este nombre en blanco, no podrá utilizar este listener.

Este atributo solo es compatible z/OS con.

Para determinar el valor de este atributo, utilice el selector MQCA_LU_GROUP_NAME con la llamada MQINQ. La longitud de este atributo viene dada por MQ_LU_NAME_LENGTH.

 LUName (MQCHAR8)

Este es el nombre de la LU que se utilizará para las transmisiones salientes de LU 6.2. Configure esto en la misma LU que utiliza el oyente para las transmisiones entrantes. Si dejas este campo en blanco, se utilizará el LUN predeterminado de APPC/MVS; dado que este valor es variable, asegúrate siempre de configurar LUName si utilizas LU6.2.

Este atributo solo es compatible z/OS con.

Para determinar el valor de este atributo, utilice el selector MQCA_LU_NAME con la llamada MQINQ. La longitud de este atributo viene dada por MQ_LU_NAME_LENGTH.

LU62ARMSuffix (MQCHAR2)

Es el sufijo de SYS1.PARMLIB APPCPMxx, que nombra el LUADD para este iniciador de canal. El z/OS comando SET APPC =xx se ejecuta cuando ARM reinicia el iniciador del canal. Si dejas este campo en blanco, no se ejecutará el comando SET APPC =xx.

Este atributo sólo está soportado en z/OS .

Para determinar el valor de este atributo, utilice el selector MQCA_LU62_ARM_SUFFIX con la llamada MQINQ. La longitud de este atributo la proporciona MQ_ARM_SUFFIX_LENGTH.

LU62Channels (MQLONG)

Este es el número máximo de canales que pueden estar activos, o clientes que pueden estar conectados, que utilizan el protocolo de transmisión LU 6.2.

El valor debe estar comprendido entre 0 y 9999, con un valor predeterminado de 200. Si se establece en cero, no se utiliza el protocolo de transmisión LU- 6.2.

Este atributo solo es compatible z/OS con.

Para determinar el valor de este atributo, utilice el selector MQIA_LU62_CHANNELS con la llamada MQINQ.

MaxActiveChannels (MQLONG)

Este atributo es el número máximo de canales que pueden estar *activos* en cualquier momento.

El valor predeterminado es el especificado en el atributo MaxChannels.

Para z/OS, el valor debe estar comprendido entre 1 y 9 999.

Para todas las demás plataformas, el valor predeterminado es 999 999 999, lo que significa que el número de canales activos es ilimitado, o se puede establecer un número real para imponer un límite.

 No debe cambiar el **MaxActiveChannels** valor en IBM MQ Appliance. Si desea limitar el número máximo de canales de cliente, utilice los atributos MAXINST y MAXINSTC por canal en las definiciones de canal SVRCONN para definir los límites de cada canal SVRCONN. Consulte [la configuración del gestor de colas en la página IBMMQ Appliance de la IBM MQ Appliance documentación](#).

El **MaxActiveChannels** parámetro es un atributo del gestor de colas solo z/OS en. En las demás plataformas, **MaxActiveChannels** es un atributo del `qm.ini` archivo. Consulte [Estrofas del archivo de configuración para colas distribuidas](#) para obtener información sobre cómo configurar el **MaxActiveChannels** atributo en otras plataformas.

Para determinar el valor de este atributo, utilice el selector MQIA_ACTIVE_CHANNELS con la **MQINQ** llamada.

Conceptos relacionados

[Estados de un canal](#)

MaxChannels (MQLONG)

Este atributo es el número máximo de canales que pueden estar *activos* (incluidos los canales de conexión al servidor con clientes conectados).

Para z/OS, el valor debe estar comprendido entre 1 y 9 999, con un valor predeterminado de 200.

 Para IBM MQ Appliance, el valor predeterminado es 999 999 999, y no debe cambiarse. Si desea limitar el número máximo de canales de cliente, utilice los atributos MAXINST y MAXINSTC por canal en las definiciones de canal SVRCONN para definir los límites de cada canal SVRCONN. Consulte [la configuración del gestor de colas en la página IBMMQ Appliance de la IBM MQ Appliance documentación](#).

Un sistema que está ocupado atendiendo conexiones de la red puede necesitar un número mayor que el valor predeterminado. Determine el valor adecuado para su entorno, idealmente observando el comportamiento de su sistema durante las pruebas.

En todas las demás plataformas, el valor predeterminado es 100. Puede establecer **MaxChannels** en un valor distinto para limitar el número máximo de canales actuales si es necesario.

El **MaxChannels** parámetro es un atributo del gestor de colas solo z/OS en. En las demás plataformas, **MaxChannels** es un atributo del `qm.ini` archivo. Consulte [Estrofas del archivo de configuración para colas distribuidas](#) para obtener información sobre cómo configurar el **MaxChannels** atributo en otras plataformas.

Para determinar el valor de este atributo, utilice el selector MQIA_MAX_CHANNELS con la **MQINQ** llamada.

Conceptos relacionados

Estados de un canal

MaxHandles (MQLONG)

Es el número máximo de descriptores de contexto abiertos que cualquier tarea puede utilizar simultáneamente. Cada llamada MQOPEN satisfactoria para una sola cola (o para un objeto que no es una cola) utiliza un descriptor de contexto. Ese descriptor de contexto pasa a estar disponible para su reutilización cuando se cierra el objeto. Sin embargo, cuando se abre una lista de distribución, a cada cola de la lista de distribución se le asigna un descriptor de contexto independiente, y de modo que la llamada MQOPEN utiliza tantos descriptores de contexto como colas hay en la lista de distribución. Esto debe tenerse en cuenta al decidir un valor adecuado para *MaxHandles*.

La llamada MQPUT1 realiza una llamada MQOPEN como parte de su proceso; como resultado, MQPUT1 utiliza tantos manejadores como MQOPEN lo haría, pero los manejadores sólo se utilizan durante la propia llamada MQPUT1.

En z/OS, *tarea* significa una tarea CICS, una tarea MVS o una región dependiente IMS.

El valor está en el rango de 1 a 999.999.999. El valor predeterminado lo determina el entorno:

- En z/OS, el valor predeterminado es 100.
- En todos los demás entornos, el valor predeterminado es 256.

Para determinar el valor de este atributo, utilice el selector MQIA_MAX_MANEJAR con la llamada MQINQ.

MaxMsgLength (MQLONG)

Es la longitud del mensaje *físico* más largo que el gestor de colas puede manejar. Sin embargo, debido a que el atributo de gestor de colas **MaxMsgLength** se puede establecer independientemente del atributo de cola **MaxMsgLength**, el mensaje físico más largo que se puede colocar en una cola es el menor de estos dos valores.

Si el gestor de colas da soporte a la segmentación, una aplicación puede colocar un mensaje lógico que sea más largo que el menor de los dos atributos **MaxMsgLength**, pero sólo si la aplicación especifica el distintivo MQMF_SEGMENTATION_ALLOWED en MQMD. Si se especifica ese distintivo, el límite superior para la longitud de un mensaje lógico es de 999.999.999 bytes, pero normalmente las restricciones de recursos impuestas por el sistema operativo, o por el entorno en el que se ejecuta la aplicación, dan como resultado un límite inferior.

El límite inferior para el atributo **MaxMsgLength** es de 32 KB (32.768 bytes). El límite superior es de 100 MB (104 857 600 bytes).

Para determinar el valor de este atributo, utilice el selector MQIA_MAX_MSG_LENGTH con la llamada MQINQ.

MaxPriority (MQLONG)

Esta es la prioridad de mensaje máxima soportada por el gestor de colas. Las prioridades van de cero (más bajo) a *MaxPriority* (más alto).

Para determinar el valor de este atributo, utilice el selector MQIA_MAX_PRIORITY con la llamada MQINQ.

MaxPropertiesLength (MQLONG)

Se utiliza para controlar el tamaño de las propiedades que pueden fluir con un mensaje. Esto incluye tanto el nombre de propiedad en bytes como el tamaño del valor de propiedad también en bytes.

Para determinar el valor de este atributo, utilice el selector MQIA_MAX_PROPERTIES_LENGTH con la llamada MQINQ.

MaxQFileSize (MQLONG)

El tamaño máximo, en megabytes, que puede alcanzar un archivo de cola.

Tabla 596. Tipos de colas a los que se aplica este atributo

Local	Modelo	Alias	Remoto	Clúster
X	X			

Es posible que un archivo de cola supere el tamaño máximo si se configura con un valor inferior al tamaño actual del archivo de cola. Si eso ocurre, el archivo de cola ya no acepta nuevos mensajes, pero permite que se consuman los mensajes existentes. Cuando el tamaño del archivo de cola ha descendido por debajo del valor configurado, se permite añadir nuevos mensajes a la cola.

Nota: Esta cifra puede diferir del valor del atributo configurado en la cola, ya que internamente el gestor de colas podría necesitar utilizar un tamaño de bloque mayor para alcanzar el tamaño elegido. Consulte *Modificación de IBM MQ archivos de cola* para obtener más información sobre cómo cambiar el tamaño de los archivos de cola y el tamaño y la granularidad de los bloques.

Cuando es necesario cambiar la granularidad porque este atributo se ha incrementado, se escribe el mensaje de AMQ7493W advertencia «Granularidad cambiada» en los registros AMQERR. Esto le indica que debe planificar el vaciado de la cola para IBM MQ poder adoptar la nueva granularidad.

El valor máximo de este atributo es 267 386 880 MB y el valor predeterminado, y el valor migrado, es 2 088 960 MB, que es el máximo actual para una cola con una granularidad igual a 512.

Para determinar el valor de este atributo, utilice el selector MQIA_MAX_Q_FILE_SIZE con la llamada MQINQ.

MaxUncommittedMsgs (MQLONG)

Es el número máximo de mensajes no confirmados que pueden existir en una unidad de trabajo. El número de mensajes no confirmados es la suma de los siguientes mensajes desde el inicio de la unidad de trabajo actual:

- Mensajes transferidos por la aplicación con la opción MQPMO_SYNCPOINT
- Mensajes recuperados por la aplicación con la opción MQGMO_SYNCPOINT
- Mensajes de activación y mensajes de informe COA generados por el gestor de colas para mensajes transferidos con la opción MQPMO_SYNCPOINT
- Mensajes de informe COD generados por el gestor de colas para mensajes recuperados con la opción MQGMO_SYNCPOINT

Los mensajes siguientes no se cuentan como no confirmados:

- Mensajes colocados o recuperados por la aplicación fuera de una unidad de trabajo
- Mensajes desencadenantes o mensajes de informe COA/COD generados por el gestor de colas como resultado de mensajes colocados o recuperados fuera de una unidad de trabajo
- Mensajes de informe de caducidad generados por el gestor de colas (incluso si la llamada que provoca el mensaje de informe de caducidad ha especificado MQGMO_SYNCPOINT)
- Mensajes de suceso generados por el gestor de colas (incluso si la llamada que provoca el mensaje de suceso ha especificado MQPMO_SYNCPOINT o MQGMO_SYNCPOINT)

Nota:

1. Los mensajes de informe de excepción los genera el agente de canal de mensajes (MCA), o la aplicación, y se tratan de la misma forma que los mensajes ordinarios colocados o recuperados por la aplicación.
2. Cuando se coloca un mensaje o segmento con la opción MQPMO_SYNCPOINT, el número de mensajes no confirmados se incrementa en uno independientemente de cuántos mensajes físicos resulten realmente de la colocación. (Puede producirse más de un mensaje físico si el gestor de colas debe subdividir el mensaje o segmento.)

3. Cuando se coloca una lista de distribución con la opción MQPMO_SYNCPOINT, el número de mensajes no confirmados se incrementa en un *para cada mensaje físico que se genera*. Esto puede ser tan pequeño como uno, o tan grande como el número de destinos en la lista de distribución.

El límite inferior para este atributo es 1; el límite superior es 999 999 999. El valor predeterminado es 10000.

Para determinar el valor de este atributo, utilice el selector MQIA_MAX_UNCOMMITTED_MSGS con la llamada MQINQ.

MQIAccounting (MQLONG)

Esto controla la recopilación de información contable para los datos de MQI.

El valor puede ser uno de los siguientes:

MQMON_ON

Recopilar datos de contabilidad de API.

MQMON_OFF

No recopile datos de contabilidad de API. Éste es el valor predeterminado.

Si establece el atributo de gestor de colas ACCTCONO en ENABLED, este valor puede alterarse temporalmente para conexiones individuales utilizando el campo Opciones de la estructura MQCNO. Los cambios en este valor sólo son efectivos para las conexiones con el gestor de colas que se producen después del cambio en el atributo.

Este atributo sólo está soportado en Multiplataformas.

Para determinar el valor de este atributo, utilice el selector MQIA_ACCOUNTING_MQI con la llamada MQINQ.

MQIStatistics (MQLONG)

Esto controla la recopilación de información de supervisión de estadísticas para el gestor de colas.

El valor puede ser uno de los siguientes:

MQMON_ON

Recopilar estadísticas de « MQI ».

MQMON_OFF

No recopila estadísticas de MQI. Éste es el valor predeterminado.

Este atributo sólo está soportado en Multiplataformas.

Para determinar el valor de este atributo, utilice el selector MQIA_STATISTICS_MQI con la llamada MQINQ.

MsgMarkBrowseInterval (MQLONG)

Intervalo de tiempo en milisegundos después del cual el gestor de colas puede eliminar automáticamente la marca de los mensajes de examen.

Se trata de un intervalo de tiempo (en milisegundos) después del cual el gestor de colas puede eliminar automáticamente la marca de los mensajes de examen.

Este atributo describe el intervalo de tiempo durante el cual se espera que los mensajes que se han marcado como examinados por una llamada a MQGET, utilizando la opción de obtención de mensaje MQGMO_MARK_BROWSE_CO_OP, permanezcan marcados como examinados.

El gestor de colas puede desmarcar automáticamente los mensajes examinados que se han marcado como examinados para el conjunto de descriptores de contexto cooperantes cuando se han marcado durante más de este intervalo aproximado.

Esto no afecta al estado de ningún mensaje marcado como examinar, que se ha obtenido mediante una llamada a MQGET, utilizando la opción de obtención de mensaje MQGMO_MARK_BROWSE_HANDLE.

El valor máximo es 999.999.999 y el valor predeterminado es 5000. Un valor especial de -1 para *MsgMarkBrowseInterval* representa un intervalo de tiempo ilimitado.



Atención: Este valor no debe estar por debajo del valor predeterminado de 5000.

Para determinar el valor de este atributo, utilice el selector MQIA_MSG_MARK_BROWSE_INTERVAL con la llamada MQINQ.

OutboundPortMax (MQLONG)

Este es el número de puerto más alto del rango, definido por OutboundPortMin y OutboundPortMax, de números de puerto que se utilizarán para enlazar canales salientes.

El valor es un número entero comprendido entre 0 y 65535, y debe ser igual o superior al valor OutboundPortMin. El valor por omisión es 0.

Este atributo solo es compatible z/OS con.

Para determinar el valor de este atributo, utilice el selector MQIA_OUTBOUND_PORT_MAX con la llamada MQINQ.

OutboundPortMin (MQLONG)

Este es el número de puerto más bajo del rango, definido por OutboundPortMin y OutboundPortMax, de números de puerto que se utilizarán para enlazar canales salientes.

El valor es un número entero comprendido entre 0 y 65535, y debe ser igual o inferior al valor OutboundPortMax. El valor por omisión es 0.

Este atributo solo es compatible z/OS con.

Para determinar el valor de este atributo, utilice el selector MQIA_OUTBOUND_PORT_MIN con la llamada MQINQ.

PerformanceEvent (MQLONG)

Esto controla si se generan sucesos relacionados con el rendimiento. Es uno de los valores siguientes:

MQEVR_DISABLED

Informes de sucesos inhabilitados.

MQEVR_ENABLED

Informes de sucesos habilitados.

Para obtener más información sobre los sucesos, consulte [Supervisión de sucesos](#).

Para determinar el valor de este atributo, utilice el selector MQIA_PERFORMANCE_EVENT con la llamada MQINQ.

Plataforma (MQLONG)

Esto indica el sistema operativo en el que se ejecuta el gestor de colas:

MQPL_AIX

AIX (mismo valor que MQPL_UNIX).

MQPL_ELECTRODOMÉSTICO

IBM MQ Appliance

MQPL_MVS

z/OS (mismo valor que MQPL_ZOS).

MQPL_OS390

z/OS (mismo valor que MQPL_ZOS).

MQPL_OS400

IBM i.

MQPL_UNIX

UNIX.

MQPL_WINDOWS_NT

Windows .

MQPL_ZOS

z/OS.

Para determinar el valor de este atributo, utilice el selector MQIA_PLATFORM con la llamada MQINQ.

PubSubNPInputMsg (MQLONG)

Indica si se debe descartar o mantener un mensaje de entrada no entregado.

El valor puede ser uno de los siguientes:

MQUNDELIVERED_DISCARD

Los mensajes de entrada no persistentes se pueden desechar si no se pueden procesar.

Éste es el valor predeterminado.

MQUNDELIVERED_KEEP

Los mensajes de entrada no persistentes no se desecharán si no se pueden procesar. En esta situación, la interfaz de publicación/suscripción en cola continuará reintentando el proceso a intervalos adecuados y no continúa procesando mensajes posteriores.

Para determinar el valor de este atributo, utilice el selector MQIA_PUBSUB_NP_MSG con la llamada MQINQ.

PubSubNPResponse (MQLONG)

Controla el comportamiento de los mensajes de respuesta no entregados.

El valor puede ser uno de los siguientes:

MQUNDELIVERED_NORMAL

Las respuestas no persistentes que no se pueden colocar en la cola de respuestas se colocan en la cola de mensajes no entregados, si no se pueden colocar en la DLQ, se descartan.

MQUNDELIVERED_SAFE

Las respuestas no persistentes que no se pueden colocar en la cola de respuestas se transfieren a la cola de mensajes no entregados. Si la respuesta no se puede establecer y no se puede colocar en la DLQ, la interfaz de publicación/suscripción en cola retrotraerá la operación actual y, a continuación, volverá a intentarlo a intervalos adecuados y no continuará procesando los mensajes posteriores.

MQUNDELIVERED_DISCARD

Las respuestas no persistentes no se colocan en la cola de respuestas se descartan.

Este es el valor predeterminado para los nuevos gestores de colas.

MQUNDELIVERED_KEEP

Las respuestas no persistentes no se colocan en la cola de mensajes no entregados ni se descartan. En su lugar, la interfaz de publicación/suscripción en cola restituirá la operación actual y, a continuación, volverá a intentarlo a intervalos adecuados.

Para determinar el valor de este atributo, utilice el selector MQIA_PUBSUB_NP_RESP con la llamada MQINQ.

Valor predeterminado para los gestores de colas migrados.

Si el gestor de colas se ha migrado desde IBM MQ V6.0, el valor inicial de este atributo depende de los valores de *DiscardNonPersistentResponse* y *DLQNonPersistentResponse* antes de la migración, como se muestra en la tabla siguiente.

		DLQNonPersistentResponse		
		Sí	No	No establecida
DiscardNonPersistentResponse	Sí	MQUDELIVERED_NORMAL	MQUDELIVERED_DISCARD	MQUDELIVERED_NORMAL
	No	MQUDELIVERED_SAFE	MQUDELIVERED_KEEP	MQUDELIVERED_SAFE
	No establecida	Si SyncPointPersistent = No, MQUDELIVERED_SAFE si no MQUDELIVERED_NORMAL	Si SyncPointPersistent = No, MQUDELIVERED_KEEP si no MQUDELIVERED_DISCARD	Si SyncPointPersistent = No, MQUDELIVERED_SAFE si no MQUDELIVERED_NORMAL

PubSubMaxMsgRetryCount (MQLONG)

Número de reintentos al procesar un mensaje de mandato fallido bajo punto de sincronismo.

El valor puede ser uno de los siguientes:

0 - 999 999 999

El valor predeterminado es 5.

Para determinar el valor de este atributo, utilice el selector MQIA_PUBSUB_MAXMSG_RETRY_COUNT con la llamada MQINQ.

PubSubSyncPoint (MQLONG)

Indica si sólo los mensajes persistentes o todos los mensajes se procesan bajo punto de sincronismo.

El valor puede ser uno de los siguientes:

MQSYNCPPOINT_IFPER

Esto hace que la interfaz de publicación/suscripción en cola reciba mensajes no persistentes fuera del punto de sincronización. Si el daemon recibe una publicación fuera del punto de sincronismo, el daemon reenvía la publicación a los suscriptores que conoce fuera del punto de sincronismo.

Éste es el valor predeterminado.

MQSYNCPPOINT_YES

Esto hace que la interfaz de publicación/suscripción en cola reciba todos los mensajes bajo punto de sincronismo.

Para determinar el valor de este atributo, utilice el selector MQIA_PUBSUB_SYNC_PT con la llamada MQINQ.

PubSubMode (MQLONG)

Si el motor de publicación/suscripción y la interfaz de publicación/suscripción en cola se están ejecutando, permitiendo por lo tanto que las aplicaciones publiquen/suscriban utilizando la interfaz de programación de aplicaciones y las colas que están siendo supervisadas por la interfaz de publicación/suscripción en cola.

El valor puede ser uno de los siguientes:

MQPSM_COMPAT

El motor de publicación/suscripción está ejecutándose. Por lo tanto, es posible publicar/suscribirse utilizando la interfaz de programación de aplicaciones. La interfaz de publicación/suscripción en cola no se está ejecutando, por lo tanto, no se actúa sobre ningún mensaje que se coloque en las colas supervisadas por la interfaz de publicación/suscripción en cola. Este valor se utiliza para la compatibilidad con WebSphere Message Broker V6 o versiones anteriores que utilizan este gestor de colas, porque debe leer las mismas colas de las que normalmente lee la interfaz de publicación/suscripción en cola.

MQPSM_INHABILITADO

El motor de publicación/suscripción y la interfaz de publicación/suscripción en cola no están ejecutándose. Por lo tanto, no es posible publicar/suscribirse utilizando la interfaz de programación de aplicaciones. No se actúa sobre los mensajes de publicación/suscripción que se colocan en las colas supervisadas por la interfaz de publicación/suscripción en cola.

MQPSM_HABILITADO

El motor de publicación/suscripción y la interfaz de publicación/suscripción en cola están ejecutándose. Por lo tanto, es posible publicar/suscribirse utilizando la interfaz de programación de

aplicaciones y las colas supervisadas por la interfaz de publicación/suscripción en cola. Este es el valor predeterminado inicial del gestor de colas.

Para determinar el valor de este atributo, utilice el selector MQIA_PUBSUB_MODE con la llamada MQINQ.

QMGrDesc (MQCHAR64)

Utilice este campo para un comentario que describa el gestor de colas. El contenido del campo no es significativo para el gestor de colas, pero es posible que el gestor de colas requiera que el campo contenga sólo caracteres que se puedan visualizar. No puede contener ningún carácter nulo; si es necesario, se rellena a la derecha con espacios en blanco. En una instalación DBCS, este campo puede contener caracteres DBCS (sujetos a una longitud máxima de campo de 64 bytes).

Nota: Si este campo contiene caracteres que no están en el juego de caracteres del gestor de colas (tal como lo define el atributo de gestor de colas **CodedCharSetId**), estos caracteres podrían traducirse incorrectamente si este campo se envía a otro gestor de colas.

- En z/OS, el valor predeterminado es el nombre de producto y el número de versión.
- En todos los demás entornos, el valor por omisión son espacios en blanco.

Para determinar el valor de este atributo, utilice el selector MQCA_Q_MGR_DESC con la llamada MQINQ. La longitud de este atributo la proporciona MQ_Q_MGR_DESC_LENGTH.

QMGrIdentifier (MQCHAR48)

Este es un nombre único generado internamente para el gestor de colas.

Para determinar el valor de este atributo, utilice el selector MQCA_Q_MGR_IDENTIFIER con la llamada MQINQ. La longitud de este atributo viene dada por MQ_Q_MGR_IDENTIFIER_LENGTH.

Este atributo es compatible con los siguientes entornos:

-  AIX
-  IBM i
-  Linux
-  Windows
-  z/OS

y IBM MQ los clientes conectados a estos sistemas.

QMGrName (MQCHAR48)

Este es el nombre del gestor de colas local, es decir, el nombre del gestor de colas al que está conectada la aplicación.

Los primeros 12 caracteres del nombre se utilizan para construir un identificador de mensaje único (véase el campo MQMD - MsgId). Por lo tanto, los gestores de colas que pueden intercomunicarse deben tener nombres que difieran en los primeros 12 caracteres, para que los identificadores de mensajes sean únicos en la red del gestor de colas.

En z/OS, el nombre es el mismo que el nombre del subsistema, que está limitado a 4 caracteres no en blanco.

Para determinar el valor de este atributo, utilice el selector MQCA_Q_MGR_NAME con la llamada MQINQ. La longitud de este atributo viene dada por MQ_Q_MGR_NAME_LENGTH.

QSGName (MQCHAR4)

Es el nombre del grupo de compartición de colas al que pertenece el gestor de colas local. Si el gestor de colas local no pertenece a un grupo de compartición de colas, el nombre está en blanco.

Para determinar el valor de este atributo, utilice el selector MQCA_QSG_NAME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_QSG_NAME_LENGTH.

 Este atributo sólo está soportado en z/OS.

QueueAccounting (MQLONG)

Esto controla la recopilación de información de contabilidad para las colas.

El valor puede ser uno de los siguientes:

MQMON_NONE

No recopile datos de contabilidad para colas, independientemente del valor del atributo de contabilidad de cola ACCTQ. Éste es el valor predeterminado.

MQMON_OFF

No recopile datos de contabilidad para colas que especifiquen QMGR en el atributo de cola ACCTQ.

MQMON_ON

Recopilar datos de contabilidad para colas que especifican QMGR en el atributo de cola ACCTQ.

Los cambios en este valor sólo son efectivos para las conexiones con el gestor de colas que se producen después del cambio en el atributo.

Para determinar el valor de este atributo, utilice el selector MQIA_ACCOUNTING_Q con la llamada MQINQ.

QueueMonitoring (MQLONG)

Especifica el valor predeterminado para la supervisión en línea de colas.

Si el atributo de cola **QueueMonitoring** se establece en MQMON_Q_MGR, este atributo especifica el valor que asume el canal. El valor puede ser:

MQMON_OFF

La recopilación de datos de supervisión en línea está desactivada. Este es el valor predeterminado inicial del gestor de colas.

MQMON_NONE

La recopilación de datos de supervisión en línea está desactivada para las colas independientemente del valor de su atributo **QueueMonitoring**.

MQMON_LOW

La recopilación de datos de supervisión en línea está activada, con una proporción baja de recopilación de datos.

MQMON_MEDIO

La recopilación de datos de supervisión en línea está activada, con una proporción moderada de recopilación de datos.

MQMON_HIGH

La recopilación de datos de supervisión en línea está activada, con una proporción alta de recopilación de datos.

Para determinar el valor de este atributo, utilice el selector MQIA_MONITORING_Q con la llamada MQINQ.

QueueStatistics (MQLONG)

Esto controla la recopilación de datos estadísticos para las colas.

Es uno de los valores siguientes:

MQMON_NONE

No recopile estadísticas de cola para colas, independientemente del valor del atributo de cola **QueueStatistics**. Éste es el valor predeterminado.

MQMON_OFF

No recopile datos de estadísticas para colas que especifiquen el gestor de colas en el atributo de cola **QueueStatistics**.

MQMON_ON

Recopilar datos estadísticos para las colas que especifican el gestor de colas en el atributo de cola **QueueStatistics**.

Para determinar el valor de este atributo, utilice el selector MQIA_STATISTICS_Q con la llamada MQINQ.

ReceiveTimeout (MQLONG)

Esto especifica cuánto tiempo espera un canal « TCP/IP » a recibir datos, incluidos los latidos, de su interlocutor antes de volver al estado inactivo. Esto solo se aplica a los canales de mensajes, y no a los canales de « MQI ».

El significado exacto de « ReceiveTimeout » se ve alterado por el valor especificado en « ReceiveTimeoutType ». ReceiveTimeoutType se puede establecer en uno de los siguientes valores:

- MQRCVTIME_EQUAL: este valor es el número de segundos que debe esperar el canal. Especifique un valor comprendido entre 0 y 999999.
- MQRCVTIME_ADD: este valor es el número de segundos que se añaden al HBINT negociado y determina cuánto tiempo espera un canal. Especifique un valor comprendido entre 1 y 999999.
- MQRCVTIME_MULTIPLY: este valor es un multiplicador que se aplica al HBINT negociado. Especifique un valor de 0 o un valor comprendido entre 2 y 99.

El valor por omisión es 0.

Establezca ReceiveTimeoutType en MQRCVTIME_MULTIPLY o MQRCVTIME_EQUAL, y ReceiveTimeout en 0, para evitar que un canal agote el tiempo de espera para recibir datos de su socio.

Este atributo solo es compatible z/OS con.

Para determinar el valor de este atributo, utilice el selector MQIA_RECEIVE_TIMEOUT con la llamada MQINQ.

ReceiveTimeoutMin (MQLONG)

Este es el tiempo mínimo, en segundos, que un canal « TCP/IP » espera para recibir datos, incluidos los pulsos de control, de su socio, antes de volver al estado inactivo.

Esto solo se aplica a los canales de mensajes, no a los canales de « MQI ». El valor debe estar comprendido entre 0 y 999999, siendo el valor predeterminado 0.

Si se utiliza « ReceiveTimeoutType » para especificar que el tiempo de espera del canal « TCP/IP » se calcule en relación con el valor negociado de « HBINT », y el valor resultante es inferior al valor de este parámetro, se utilizará este último en su lugar.

Este atributo solo es compatible z/OS con.

Para determinar el valor de este atributo, utilice el selector MQIA_RECEIVE_TIMEOUT_MIN con la llamada MQINQ.

ReceiveTimeoutType (MQLONG)

Este es el parámetro de configuración que se aplica a ReceiveTimeout para definir cuánto tiempo espera un canal TCP/IP a recibir datos, incluidos los latidos, de su socio, antes de volver al estado inactivo. Esto solo se aplica a los canales de mensajes, no a los canales de « MQI ».

El valor puede ser uno de los siguientes:

MQRCVTIME_MULTIPLICAR

ReceiveTimeout Es un multiplicador que se aplica al valor HBINT negociado para determinar cuánto tiempo espera un canal. Éste es el valor predeterminado.

MQRCVTIME_ADD

ReceiveTimeout es un valor, en segundos, que se añade al valor HBINT negociado para determinar cuánto tiempo espera un canal.

MQRCVTIME_EQUAL

ReceiveTimeout es un valor, en segundos, que el canal espera.

Para detener el tiempo de espera de un canal para recibir datos de su socio, establezca ReceiveTimeoutType en MQRCVTIME_MULTIPLY o MQRCVTIME_EQUAL, y ReceiveTimeout en 0.

Este atributo solo es compatible z/OS con.

Para determinar el valor de este atributo, utilice el selector MQIA_RECEIVE_TIMEOUT_TYPE con la llamada MQINQ.

RemoteEvent (MQLONG)

Esto controla si se generan sucesos de error remotos. Es uno de los valores siguientes:

MQEVR_DISABLED

Informes de sucesos inhabilitados.

MQEVR_ENABLED

Informes de sucesos habilitados.

Para obtener más información sobre los sucesos, consulte [Supervisión de sucesos](#).

Para determinar el valor de este atributo, utilice el selector MQIA_REMOTE_EVENT con la llamada MQINQ.

RepositoryName (MQCHAR48)

Es el nombre de un clúster para el que este gestor de colas proporciona un servicio de gestor de repositorios. Si el gestor de colas proporciona este servicio para más de un clúster, *RepositoryNameList* especifica el nombre de un objeto de lista de nombres que identifica los clústeres y *RepositoryName* está en blanco. Al menos uno de *RepositoryName* y *RepositoryNameList* debe estar en blanco.

Para determinar el valor de este atributo, utilice el selector MQCA_REPOSITORY_NAME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_Q_MGR_NAME_LENGTH.

RepositoryNameList (MQCHAR48)

Es el nombre de un objeto de lista de nombres que contiene los nombres de los clústeres para los que este gestor de colas proporciona un servicio de gestor de repositorios. Si el gestor de colas proporciona este servicio sólo para un clúster, el objeto de lista de nombres sólo contiene un nombre. De forma alternativa, se puede utilizar *RepositoryName* para especificar el nombre del clúster, en cuyo caso *RepositoryNameList* está en blanco. Al menos uno de *RepositoryName* y *RepositoryNameList* debe estar en blanco.

Para determinar el valor de este atributo, utilice el selector MQCA_REPOSITORY_NAMELIST con la llamada MQINQ. La longitud de este atributo la proporciona MQ_NAMELIST_NAME_LENGTH.

ScyCase(MQCHAR8)

Especifica si el gestor de colas da soporte a nombres de perfil de seguridad en mayúsculas y minúsculas, o sólo en mayúsculas.

El valor puede ser uno de los siguientes:

MQSCYC_UPPER

Los nombres de perfil de seguridad deben estar en mayúsculas.

MQSCYC_MIXED

Los nombres de perfil de seguridad pueden estar en mayúsculas o en mayúsculas y minúsculas.

Los cambios en este atributo entran en vigor cuando se ejecuta un mandato Renovar seguridad con *SecurityType* (MQSECTYPE_CLASSES) especificado.

 Este atributo sólo está soportado en z/OS.

Para determinar el valor de este atributo, utilice el selector MQIA_SECURITY_CASE con la llamada MQINQ.

SharedQMgrName (MQLONG)

Especifica si *ObjectQmgrName* debe utilizarse o tratarse como el gestor de colas local en una llamada MQOPEN, para una cola compartida, cuando *ObjectQmgrName* es el de otro gestor de colas del grupo de compartición de colas.

El valor puede ser cualquiera de los valores siguientes:

MQSQQM_USE

Se utiliza *ObjectQmgrName* y se abre la cola de transmisión adecuada.

MQSQQM_IGNORE

Si la cola de destino es compartida, y *ObjectQmgrName* es la de un gestor de colas en el mismo grupo de compartición de colas, la apertura se realiza localmente.

Este atributo sólo es válido en z/OS.

Para determinar el valor de este atributo, utilice el selector MQIA_SHARED_Q_Q_MGR_NAME con la llamada MQINQ.

SPLCAP

Indica si las prestaciones de seguridad de Advanced Message Security están disponibles para un gestor de colas.

MQCAP_COMPATIBLE

Este es el valor predeterminado si el componente AMS está instalado para la instalación bajo la que se ejecuta el gestor de colas.

MQCAP_NO_SOPORTADO

SSLEvent (MQLONG)

Esto determina si se generan eventos de « TLS ».

Es uno de los valores siguientes:

MQEVR_ENABLED

Genera eventos de « TLS » de la siguiente manera:

MQRC_CHANNEL_SSL_ERROR

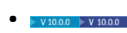
MQEVR_DISABLED

No genere eventos de « TLS »; este es el valor predeterminado.

Para determinar el valor de este atributo, utilice el selector MQIA_SSL_EVENT con la llamada MQINQ.

SSLFIPSRequired (MQLONG)

Notas:

-  En AIX, Linux, y Windows, IBM MQ garantiza FIPS 140-3 el cumplimiento de la normativa a través del GSKit 9 módulo IBM Crypto for C (ICC) criptográfico. La certificación del NIST correspondiente al FIPS 140-3 módulo puede consultarse en <https://csrc.nist.gov/projects/cryptographic-module-validation-program/certificate/4755>.
-  FIPS Actualmente no hay soporte disponible para Linux s390x. Los clientes de esta plataforma que necesiten FIPS asistencia técnica deben seguir utilizando una versión anterior de IBM MQ. La compatibilidad con FIPS 140-3 se habilitará en una futura versión de FixPack una vez que el IBM Crypto for C (ICC) módulo criptográfico haya obtenido su certificación en esta plataforma.

-  El módulo FIPS 140-3 criptográfico integrado IBM Semeru Runtime fue aprobado por el NIST en agosto de 2024. IBM MQ 10.0.0 incorpora compatibilidad con la gestión de IBM MQ classes for JMS conexiones de IBM MQ classes for Java cliente mediante TLS para FIPS 140-3 en Java 8 y IBM Semeru Runtime 11+. La certificación del NIST correspondiente al FIPS 140-3 módulo se puede consultar en <https://csrc.nist.gov/projects/cryptographic-module-validation-program/certificate/4755>.
- En IBM MQ el caso de Containers, las imágenes de IBM MQ Operator contenedor de 3.2.0 y del gestor de colas 9.4.0.0 en adelante se basan en UBI 9. FIPS 140-3 El cumplimiento de IBM MQ la normativa para productos envasados está actualmente pendiente.

 Si FIPS está habilitado, los procesos IBM MQ de control de contenedores utilizan un módulo FIPS 140-3 certificado de OpenSSL. Para obtener más información sobre la certificación del NIST, consulte: <https://access.redhat.com/compliance/fips>. IBM MQ Los gestores de colas que se ejecutan en imágenes de contenedor tienen el mismo nivel FIPS de certificación que la versión de la plataforma de IBM MQ la imagen base.

Esto le permite especificar que solo se utilizarán FIPS algoritmos certificados si la criptografía se ejecuta en IBM MQ, en lugar de en hardware criptográfico. Si se configura hardware criptográfico, los módulos de criptografía utilizados son los proporcionados por el producto de hardware; estos módulos pueden estar o no FIPS certificados para un nivel concreto, dependiendo del producto de hardware utilizado.

El valor es uno de los siguientes valores:

MQSSL_FIPS_NO

Utilice cualquier CipherSpec compatible con la plataforma en uso. Este es el valor predeterminado.

MQSSL_FIPS_YES

 Utilice una biblioteca criptográfica y algoritmos certificados por FIPS para todas las funciones criptográficas, incluyendo TLS, AMS, protección con contraseña y verificación JWT.

 Utilice únicamente algoritmos FIPS criptográficos certificados en el CipherSpecs permitidos en todas las conexiones TLS desde y hacia este gestor de colas.

Este parámetro solo es válido en plataformas z/OS AIX, Linux, and Windows .

Para determinar el valor de este atributo, utilice el selector MQIA_SSL_FIPS_REQUIRED con la llamada MQINQ.

Tareas relacionadas

Especificar que solo FIPS se utilicen certificados CipherSpecs en tiempo de ejecución en el cliente MQI

Referencia relacionada

Federal Information Processing Standards (FIPS) para AIX, Linux, and Windows

SSLKeyResetCount (MQLONG)

Esto especifica cuándo los agentes de canal de mensajes (MCA) de un canal de « TLS » que inician la comunicación restablecen la clave secreta utilizada para el cifrado en el canal.

El valor representa el número total de bytes no cifrados que se envían y se reciben a través del canal antes de renegociar la clave secreta. El número de bytes incluye la información de control que envía el MCA.

El valor es un número comprendido entre 0 y 999 999 999, con un valor por omisión de 0. Si especifica una cuenta de restablecimiento de clave secreta TLS entre 1 byte y 32 KB, los canales TLS utilizarán una cuenta de restablecimiento de clave secreta de 32 KB. El objetivo es evitar el coste de procesamiento que supondría un número excesivo de restablecimientos de claves, lo cual ocurriría si los valores de restablecimiento de la clave secreta de TLS fueran demasiado pequeños.

La clave secreta se renegocia cuando el número total de bytes no cifrados enviados y recibidos por el MCA del canal iniciador excede el valor especificado. Si las pulsaciones de canal están habilitadas, la clave secreta se renegocia antes de que se envíen o reciban los datos después de una pulsación de canal, o cuando el número total de bytes no cifrados excede el valor especificado, lo que ocurra primero.

El recuento de bytes enviados y recibidos para renegociación incluye la información de control enviada y recibida por el canal MCA y se restablece cuando se produce una renegociación.

Utilice un valor de 0 para indicar que las claves secretas nunca se renegocian.

Para determinar el valor de este atributo, utilice el selector MQIA_SSL_RESET_COUNT con la llamada MQINQ.

StartStopEvent (MQLONG)

Esto controla si se generan sucesos de inicio y detención. El valor puede ser uno de los siguientes:

MQEVR_DISABLED

Informes de sucesos inhabilitados.

MQEVR_ENABLED

Informes de sucesos habilitados.

Para obtener más información sobre los sucesos, consulte [Supervisión de sucesos](#).

Para determinar el valor de este atributo, utilice el selector MQIA_START_STOP_EVENT con la llamada MQINQ.

StatisticsInterval (MQLONG)

Especifica la frecuencia (en segundos) con la que se graban los datos de supervisión de estadísticas en la cola de supervisión.

El valor es un entero en el rango de 0 a 604800, con un valor predeterminado de 1800 (30 minutos).

Para determinar el valor de este atributo, utilice el selector MQIA_STATISTICS_INTERVAL con la llamada MQINQ.

SyncPoint (MQLONG)

Esto indica si el gestor de colas local soporta unidades de trabajo y sincronización con las llamadas MQGET, MQPUT y MQPUT1.

MQSP_AVAILABLE

Unidades de trabajo y sincronización disponibles.

MQSP_NOT_AVAILABLE

Unidades de trabajo y sincronización no disponibles.

- En z/OS este valor nunca se devuelve.

Para determinar el valor de este atributo, utilice el selector MQIA_SYNCPOINT con la llamada MQINQ.

Canales TCP (MQLONG)

Este es el número máximo de canales activos, o de clientes conectados, que utilizan el protocolo de transmisión « TCP/IP ».

El valor debe estar comprendido entre 0 y 9999, con un valor predeterminado de 200. Si se especifica 0, no se utiliza « TCP/IP ».

Este atributo solo es compatible z/OS con.

Para determinar el valor de este atributo, utilice el selector MQIA_TCP_CHANNELS con la llamada MQINQ.

TCPKeepAlive (MQLONG)

Esto especifica si se debe utilizar TCP KEEPALIVE para comprobar que el otro extremo de la conexión sigue estando disponible. Si no está disponible, el canal se cierra.

El valor puede ser uno de los siguientes:

MQTCPKEEP_SÍ

Utilice TCP KEEPALIVE tal y como se especifica en el conjunto de datos de configuración del perfil TCP. Si se especifica el atributo de canal « KeepAliveInterval » (KAINT), se utiliza el valor al que está establecido.

MQTCPKEEP_NO

No utilices TCP KEEPALIVE. Éste es el valor predeterminado.

Este atributo solo es compatible z/OS con.

Para determinar el valor de este atributo, utilice el selector MQIA_TCP_KEEP_ALIVE con la llamada MQINQ.

TCPName (MQCHAR8)

Este es el nombre de la pila de « TCP/IP » única o preferida que se utilizará, dependiendo del valor de « TCPStackType ». Este parámetro solo es aplicable en entornos CINET con múltiples pilas. El valor predeterminado es TCPIP.

Este atributo solo es compatible z/OS con.

Para determinar el valor de este atributo, utilice el selector MQCA_TCP_NAME con la llamada MQINQ. La longitud de este atributo viene dada por MQ_TCP_NAME_LENGTH.

TCPStackType (MQLONG)

Esto especifica si el iniciador del canal solo puede utilizar la pila TCP/IP especificada en TCPName, o si, opcionalmente, puede vincularse a cualquier pila TCP/IP seleccionada. Este parámetro solo es aplicable en entornos CINET con múltiples pilas.

El valor puede ser uno de los siguientes:

MQTCPSTACK_SINGLE

El iniciador del canal solo puede utilizar los espacios de direcciones de red de control de tráfico (TCP/IP) indicados en TCPName. Éste es el valor predeterminado.

MQTCPSTACK_MÚLTIPLE

El iniciador del canal puede utilizar cualquier espacio de direcciones TCP/IP del que disponga. Si no se especifica otro para un canal o un oyente, se utiliza por defecto el especificado en TCPName.

Este atributo solo es compatible z/OS con.

Para determinar el valor de este atributo, utilice el selector MQIA_TCP_STACK_TYPE con la llamada MQINQ.

TraceRouteRecording (MQLONG)

Esto controla el registro de la información de ruta de rastreo.

El valor puede ser uno de los siguientes:

MQRECORDING_INHABILITADO

No se permite añadir mensajes de ruta de rastreo.

MQRECORDING_Q

Coloque los mensajes de ruta de rastreo en una cola con nombre fijo.

MQRECORDING_MSG

Transfiera los mensajes de ruta de rastreo a una cola determinada utilizando el propio mensaje. Se trata del valor predeterminado

Para determinar el valor de este atributo, utilice el selector MQIA_TRACE_ROUTE_REGISTRAR con la llamada MQINQ.

TriggerInterval (MQLONG)

Es un intervalo de tiempo (en milisegundos) utilizado para restringir el número de mensajes desencadenantes. Esto sólo es relevante cuando *TriggerType* es MQTT_FIRST. En este caso, los mensajes desencadenantes normalmente se generan sólo cuando llega un mensaje adecuado a la cola y la cola estaba vacía anteriormente. En determinadas circunstancias, sin embargo, se puede generar un mensaje desencadenante adicional con el desencadenante MQTT_FIRST incluso si la cola

no estaba vacía. Estos mensajes desencadenantes adicionales no se generan con más frecuencia que cada *TriggerInterval* milisegundos.

Para obtener más información sobre el desencadenamiento, consulte [Desencadenamiento de canales](#).

El valor no es menor que 0 ni mayor que 999 999 999. El valor predeterminado es 999.999.999.

Para determinar el valor de este atributo, utilice el selector MQIA_TRIGGER_INTERVAL con la llamada MQINQ.

TriggerInterval (MQLONG)

Es un intervalo de tiempo (en milisegundos) utilizado para restringir el número de mensajes desencadenantes. Esto sólo es relevante cuando *TriggerType* es MQTT_FIRST. En este caso, los mensajes desencadenantes normalmente se generan sólo cuando llega un mensaje adecuado a la cola y la cola estaba vacía anteriormente. En determinadas circunstancias, sin embargo, se puede generar un mensaje desencadenante adicional con el desencadenante MQTT_FIRST incluso si la cola no estaba vacía. Estos mensajes desencadenantes adicionales no se generan con más frecuencia que cada *TriggerInterval* milisegundos.

Para obtener más información sobre el desencadenamiento, consulte [Desencadenamiento de canales](#).

El valor no es menor que 0 ni mayor que 999 999 999. El valor predeterminado es 999.999.999.

Para determinar el valor de este atributo, utilice el selector MQIA_TRIGGER_INTERVAL con la llamada MQINQ.

Versión (MQCFST)

Esta es la versión del código IBM MQ como VVRRMMFF, donde:

VV-Versión

RR-Release

MM-Nivel de mantenimiento

FF-Nivel de arreglo

XrCapability(MQLONG)

Esto controla si IBM MQ Telemetry los comandos son compatibles con el gestor de colas.

El valor puede ser uno de los siguientes:

MQCAP_COMPATIBLE

IBM MQ Telemetry Componente instalado y se admiten comandos de telemetría.

MQCAP_NO_COMPATIBLE

IBM MQ Telemetry Componente no instalado.

Este atributo solo es compatible con [Multiplataformas](#).

Para determinar el valor de este atributo, utilice el selector MQIA_XR_CAPABILITY con la MQINQ llamada.

Atributos para colas

Hay cinco tipos de definición de cola. Algunos atributos de cola se aplican a todos los tipos de cola; otros atributos de cola solo se aplican a determinados tipos de cola.

Tipos de colas

El gestor de colas admite los siguientes tipos de definición de colas:

Cola local

Puede almacenar mensajes en una cola local.

z/OS En z/OS puedes convertirla en una cola compartida o privada.

Una cola se conoce en un programa como *local* si es propiedad del gestor de colas al que está conectado el programa. Puede obtener mensajes y transferirlos en las colas locales.

El objeto de definición de cola contiene la información de definición de la cola, así como los mensajes físicos colocados en cola.

Cola del gestor de colas local

La cola existe en el gestor de colas local.

z/OS La cola se conoce como cola privada en z/OS.

z/OS Cola compartida (z/OS solo)

La cola existe en un repositorio compartido al que pueden acceder todos los gestores de colas que pertenecen al grupo de intercambio de colas propietario del repositorio compartido.

Las aplicaciones conectadas a cualquier gestor de colas del grupo de intercambio de colas pueden colocar mensajes en colas de este tipo y eliminarlos de ellas. Estas colas son, en la práctica, iguales que las colas locales. El valor del atributo **QType** queue es MQQT_LOCAL.

Las aplicaciones conectadas al gestor de colas local pueden colocar mensajes en colas de este tipo y eliminarlos de ellas. El valor del atributo **QType** queue es MQQT_LOCAL.

Cola de clúster

Puede almacenar mensajes en una cola de clúster en el gestor de colas donde está definida. Una cola de clúster es una cola que se aloja en un gestor de colas de clúster y que está disponible para otros gestores de colas del clúster. El valor del atributo **QType** queue es MQQT_CLUSTER.

Una definición de cola de clúster se anuncia en otros gestores de colas del clúster. Los otros gestores de colas del clúster pueden transferir mensajes a una cola de clúster sin necesidad de que haya una definición de cola remota correspondiente. Una cola de clúster se puede anunciar en más de un clúster utilizando una lista de nombres de clúster.

Cuando se anuncia una cola, cualquier gestor de colas del clúster puede poner mensajes en ella. Para transferir un mensaje, el gestor de colas debe averiguar, en los repositorios completos, donde está alojada la cola. A continuación, añade información de direccionamiento al mensaje y pone el mensaje a una cola de transmisión de clúster.

Un gestor de colas puede almacenar mensajes para otros gestores de colas en un clúster en varias colas de transmisión. Puede configurar un gestor de colas para almacenar mensajes en varias colas de transmisión de clúster de dos maneras diferentes. Si establece el atributo de gestor de colas **DEFCLXQ** en CHANNEL, se crea automáticamente una cola de transmisión de clúster diferente de SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE para cada canal de clúster emisor. Si establece la opción de cola de transmisión CLCHNAME para que coincida con uno o varios canales de clúster emisor, el gestor de colas puede almacenar mensajes para los canales coincidentes en esa cola de transmisión.



Atención: Si utiliza SYSTEM.CLUSTER.TRANSMIT.QUEUES dedicado con un gestor de colas que se ha actualizado desde una versión del producto anterior a IBM WebSphere MQ 7.5, asegúrese de que SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE tiene la opción SHARE/NOSHARE establecida en **SHARE**.

Una cola de clúster puede ser una cola que se comparte entre miembros de un grupo de compartición de colas en IBM MQ for z/OS.

Cola remota

Una cola remota no es una cola física, sino la definición local de una cola que existe en un gestor de colas remoto. La definición local de la cola remota contiene información que indica al gestor de colas local cómo direccionar los mensajes al gestor de colas remoto.

Las aplicaciones conectadas al gestor de colas local pueden colocar mensajes en colas de este tipo; los mensajes se colocan en la cola de transmisión local utilizada para enrutar mensajes al gestor de

colas remoto. Las aplicaciones no pueden eliminar mensajes de colas remotas. El valor del atributo **QType** queue es MQQT_REMOTE.

También puede utilizar una definición de cola remota para:

- Alias de cola de respuesta

En este caso, el nombre de la definición es el nombre de una cola de respuesta. Para obtener más información, consulte [Alias y clústeres de colas de respuesta](#).

- Alias del gestor de colas

En este caso, el nombre de la definición es un alias para un gestor de colas, y no el nombre de una cola. Para obtener más información, consulte [Alias y clústeres del administrador de colas](#).

Cola alias

No se trata de una cola física, sino de un nombre alternativo para una cola local, una cola compartida, una cola de clúster o una cola remota. El nombre de la cola a la que resuelve el alias forma parte de la definición de la cola del alias.

Las aplicaciones conectadas al gestor de colas local pueden colocar mensajes en colas de este tipo; los mensajes se colocan en la cola a la que resuelve el alias. Las aplicaciones pueden eliminar mensajes de colas de este tipo si el alias se resuelve en una cola local, una cola compartida o una cola de clúster que tenga una instancia local. El valor del atributo **QType** queue es MQQT_ALIAS.

Cola modelo

No se trata de una cola física, sino de un conjunto de atributos de cola a partir de los cuales se puede crear una cola local.

Los mensajes no se pueden almacenar en colas de este tipo.

límites en cola

Puede configurar y supervisar colas que admitan mucho más que el límite predeterminado de dos terabytes utilizado en versiones anteriores de IBM MQ. También tiene la opción de reducir hasta qué tamaño puede aumentar un archivo de la cola.

Para poder configurar colas, puede utilizar el **MAXFSIZE** atributo en colas locales y de modelo, y para supervisar colas, puede utilizar los atributos de estado **CURFSIZE** de cola y **CURMAXFS**.

Para obtener más información, consulte [Modificación de los archivos de colas IBM MQ](#).

Atributos de colas

Algunos atributos de cola se aplican a todos los tipos de cola; otros atributos de cola solo se aplican a determinados tipos de cola. Los tipos de cola a los que se aplica un atributo se muestran en [Tabla 597 en la página 844](#) y en las tablas siguientes.

[Tabla 597 en la página 844](#) resume los atributos específicos de las colas. Los atributos se describen en orden alfabético.

Nota: Los nombres de los atributos que se muestran en esta sección son nombres descriptivos que se utilizan con las llamadas MQINQ MQSET y; los nombres son los mismos que los de los comandos PCF. Cuando se utilizan comandos MQSC para definir, modificar o mostrar atributos, se utilizan nombres cortos alternativos; consulte [los comandos MQSC](#) para obtener más detalles.

En la siguiente tabla, las columnas se aplican de la siguiente manera:

- La columna para colas locales también se aplica a las colas compartidas.
- La columna para las colas de modelo indica qué atributos hereda la cola local creada a partir de la cola de modelo.
- La columna para colas de clúster indica los atributos que se pueden consultar cuando la cola de clúster se abre solo para consulta o para consulta y salida. Si se solicitan otros atributos, la llamada devuelve el código de finalización MQCC_WARNING y el código de motivo MQRC_SELECTOR_NOT_FOR_TYPE (2068).

Si la cola del clúster está abierta para consultas más una o más de las opciones de entrada, exploración o configuración, se aplica en su lugar la columna para colas locales.

Si la cola del clúster se abre solo para consultas, o para consultas y salida, además de especificar el nombre del gestor de colas base, se aplica en su lugar la columna para colas locales.

Tabla 597. Atributos para colas						
Atributo	Descripción	Local	Modelo	Alias	Remoto	Clúster
AlterationDate	Fecha en que se modificó por última vez la definición	X		X	X	
AlterationTime	Fecha de la última modificación de la definición	X		X	X	
BackoutRequeueQName	Nombre de la cola de reencolado por retroceso excesivo	X	X			
BackoutThreshold	Umbral de restituciones	X	X			
BaseQName	Nombre de la cola a la que resuelve el alias			X		
CFStrucName	Nombre de la estructura de la instalación de acoplamiento	X	X			
CLCHNAME	Nombres de canales del emisor de clústeres	✓	✓			
ClusterName	Nombre del clúster al que pertenece la cola	X		X	X	X
ClusterNameList	Nombre del objeto de la lista de nombres que contiene los nombres de los clústeres a los que pertenece la cola	X		X	X	
CLWLQueuePriority	Prioridad de la cola de carga de trabajo del clúster	X		X	X	X
CLWLQueueRank	Clasificación de la cola de carga de trabajo del clúster	X		X	X	X
CLWLUseQ	Usar cola remota	X				
CreationDate	Fecha en la que se creó la cola	X				
CreationTime	Hora en que se creó la cola	X				
CurrentQDepth	Profundidad de cola actual	X				
DefaultPutResponse	Resp predet de transferencia	✓	✓	✓	✓	
DefBind	Enlace predeterminado	X		X	X	X
DefinitionType attribute	Tipo de definición de cola	X	X			
DefInputOpenOption	Opción abierta de entrada predeterminada	X	X			
DefPersistence	Persistencia de mensajes predeterminada	X	X	X	X	X
DefPriority	Prioridad de mensajes predeterminada	✓	✓	✓	✓	✓
DefReadAhead	Lectura anticipada predeterminada	X	X	X		
DistLists	Soporte de lista de distribución	X	X			
HardenGetBackout	Si se debe mantener un recuento preciso de las retiradas	X	X			
IndexType	Tipo de índice	X	X			
InhibitGet	Si se permiten las operaciones para la cola	X	X	X		

Tabla 597. Atributos para colas (continuación)						
Atributo	Descripción	Local	Modelo	Alias	Remoto	Clúster
InhibitPut	Si se permiten las operaciones de inserción en la cola	X	X	X	X	X
InitiationQName	Nombre de la cola de inicio	X	X			
MaxMsgLength	Longitud máxima de mensaje en bytes	X	X			
MaxQDepth	Profundidad máxima de la cola	X	X			
MsgDeliverySequence attribute	Secuencia de entrega de mensajes	X	X			
NonPersistentMessage Class	Objetivo de fiabilidad para mensajes no persistentes	X	X			
OpenInputCount	Número de aperturas para entrada	X				
OpenOutputCount	Número de aperturas para salida	X				
PropertyControl	Control de propiedad	✓	✓	✓		
ProcessName	Nombre de proceso	X	X			
QDepthHighEvent attribute	Si se generan eventos de profundidad de cola alta	X	X			
QDepthHighLimit	Límite máximo para la profundidad de la cola	X	X			
QDepthLowEvent attribute	Si se generan eventos de profundidad de cola baja	X	X			
QDepthLowLimit attribute	Límite inferior para la profundidad de la cola	X	X			
QDepthMaxEvent	Si se generan eventos de cola llena	X	X			
QDesc	Descripción de la cola	X	X	X	X	X
QName	Nombre de cola	X		X	X	X
QServiceInterval	Objetivo para el intervalo de servicio de la cola	X	X			
QServiceIntervalEvent attribute	Si se generan eventos de intervalo de servicio alto o intervalo de servicio correcto	X	X			
QSGDisp attribute	Disposición de grupo de uso compartido de colas	X		X	X	
QueueAccounting	Recopilación de datos contables de la cola	X	X	X	X	X
QueueMonitoring	Datos de supervisión en línea para colas	X	✓			
QueueStatistics	recopilación de datos de estadísticas de cola	X	X	X	X	X
QType	Tipo de cola	X		X	X	X
RemoteQMgrName	Nombre del gestor de colas remoto				X	
RemoteQName	Nombre de la cola remota				X	
RetentionInterval	Intervalo de retención	X	X			
Scope	Si una entrada para la cola también existe en un directorio de celdas	X		X	X	
Shareability	Compartibilidad de colas	X	X			

Tabla 597. Atributos para colas (continuación)						
Atributo	Descripción	Local	Modelo	Alias	Remoto	Clúster
StorageClass	Clase de almacenamiento para cola	X	X			
TriggerControl	Activar control	X	X			
TriggerData	Datos desencadenantes	X	X			
TriggerDepth	Profundidad de desencadenante	X	X			
TriggerMsgPriority	Prioridad de mensaje de umbral para desencadenantes	X	X			
TriggerType	Tipo de desencadenante	X	X			
Usage attribute	Uso de cola	X	X			
XmitQName	Nombre de cola de transmisión				X	

Conceptos relacionados

[Colas de clúster](#)

[Colas locales](#)

[Cómo seleccionar qué tipo de cola de transmisión de clúster se debe utilizar](#)

AlterationDate (MQCHAR12)

Fecha en la que se cambió por última vez la definición.

Tabla 598. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X		X	X	

Es la fecha en la que se modificó por última vez la definición. El formato de la fecha es YYYY-MM-DD, relleno con dos espacios en blanco al final para que la longitud sea de 12 bytes (por ejemplo, 1992-09-23 , donde representa dos caracteres en blanco).

Los valores de determinados atributos (por ejemplo, *CurrentQDepth*) cambian a medida que opera el gestor de colas. Los cambios en estos atributos no afectan a *AlterationDate*.

Para determinar el valor de este atributo, utilice el selector MQCA_ALTERATION_DATE con la llamada MQINQ. La longitud de este atributo la proporciona MQ_DATE_LENGTH.

AlterationTime (MQCHAR8)

Hora a la que se modificó por última vez la definición.

Tabla 599. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X		X	X	

Es la hora en que se modificó por última vez la definición. El formato de la hora es HH.MM.SS utilizando el reloj de 24 horas, con un cero inicial si la hora es inferior a 10 (por ejemplo, 09.10.20).

- En z/OS, la hora es Greenwich Mean Time (GMT), sujeta a que el reloj del sistema se establezca correctamente en GMT.
- En otros entornos, la hora es local.

Los valores de determinados atributos (por ejemplo, *CurrentQDepth*) cambian a medida que opera el gestor de colas. Los cambios en estos atributos no afectan a *AlterationTime*.

Para determinar el valor de este atributo, utilice el selector MQCA_ALTERATION_TIME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_TIME_LENGTH.

BackoutRequeueQName (MQCHAR48)

Este es el nombre de cola de reposición en cola de restitución excesivo. Aparte de permitir que se consulte su valor, el gestor de colas no realiza ninguna acción basada en el valor de este atributo.

Tabla 600. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X			

Las aplicaciones que se ejecutan dentro de WebSphere Application Server y las que utilizan IBM MQ Application Server Facilities utilizan este atributo para determinar dónde deben ir los mensajes que se han restituido. Para todas las demás aplicaciones, el gestor de colas no realiza ninguna acción basada en el valor del atributo.

IBM MQ classes for JMS utiliza este atributo para determinar dónde transferir un mensaje que ya se ha restituido el número máximo de veces especificado por el atributo *BackoutThreshold*.

Para determinar el valor de este atributo, utilice el selector MQCA_BACKOUT_REQ_Q_NAME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_Q_NAME_LENGTH.

BackoutThreshold (MQLONG)

Este es el umbral de restitución. Aparte de permitir que se consulte su valor, el gestor de colas no realiza ninguna acción basada en el valor de este atributo.

Tabla 601. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X			

Las aplicaciones que se ejecutan dentro de WebSphere Application Server y las que utilizan IBM MQ Application Server Facilities utilizarán este atributo para determinar si se debe restituir un mensaje. Para todas las demás aplicaciones, el gestor de colas no realiza ninguna acción basada en el valor del atributo.

IBM MQ classes for JMS utiliza este atributo para determinar cuántas veces se debe permitir que se restituya un mensaje antes de transferir el mensaje a la cola especificada por el atributo *BackoutRequeueQName*.

Para determinar el valor de este atributo, utilice el selector MQIA_BACKOUT_THRESHOLD con la llamada MQINQ.

BaseQName (MQCHAR48)

Este es el nombre de una cola definida en el gestor de colas local.

Tabla 602. Tipos de colas a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
		X		

(Para obtener más información sobre los nombres de colas, consulte MQOD: [campo « ObjectName »](#) (Nombre de la cola de mensajes).) La cola es de uno de los siguientes tipos:

MQQT_LOCAL

Cola local.

MQQT_REMOTE

Definición local de una cola remota.

MQQT_CLUSTER

Cola de clúster.

Para determinar el valor de este atributo, utilice el selector MQCA_BASE_Q_NAME con la llamada MQINQ. La longitud de este atributo viene dada por MQ_Q_NAME_LENGTH.

BaseType (MQCFIN)

El tipo de objeto en el que se resuelve el alias.

Tabla 603. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remoto	Clúster
		X		

Es uno de los valores siguientes:

MQQT_Q

El tipo de objeto base es una cola

MQQT_TEMA

El tipo de objeto base es un tema

CFStrucName (MQCHAR12)

Es el nombre de la estructura del recurso de acoplamiento donde se almacenan los mensajes de la cola. El primer carácter del nombre está en el rango de A a Z, y los caracteres restantes están en el rango de A a Z, de 0 a 9 o en blanco.

Tabla 604. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remoto	Clúster
X	X			

Para obtener el nombre completo de la estructura en el recurso de acoplamiento, sufiijo el valor del atributo del gestor de colas **QSGName** con el valor del atributo de cola **CFStrucName**.

Este atributo sólo se aplica a las colas compartidas; se ignora si *QSGDisp* no tiene el valor MQQSGD_SHARED.

Para determinar el valor de este atributo, utilice el selector MQCA_CF_STRUC_NAME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_CF_STRUC_NAME_LENGTH.

 Este atributo sólo está soportado en z/OS.

ClusterChannelName (MQCHAR20)

ClusterChannelName es el nombre genérico de los canales de envío del clúster que utilizan esta cola como cola de transmisión. El atributo especifica los canales de clúster emisor han enviado mensajes a un canal de clúster receptor desde esta cola de transmisión de clúster.

Tabla 605. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remoto	Clúster
X	X			

La configuración del gestor de colas predeterminado es para todos los canales de clúster emisor para enviar mensajes desde una sola cola de transmisión, SYSTEM.CLUSTER.TRANSMIT.QUEUE. La configuración predeterminada se puede cambiar modificando el atributo del gestor de colas,

DefClusterXmitQueueType. El valor predeterminado del atributo es SCTQ. Puede cambiar el valor a CHANNEL. Si establece el atributo **DefClusterXmitQueueType** en CHANNEL, cada canal de clúster emisor utiliza de forma predeterminada una cola de transmisión de clúster específica, SYSTEM.CLUSTER.TRANSMIT.*ChannelName*.

También puede establecer el atributo de cola de transmisión *ClusterChannelName* en un canal de clúster emisor manualmente. Los mensajes destinados al gestor de colas conectado por el canal de clúster emisor se almacenan en la cola de transmisión que identifica el canal de clúster emisor. No se almacenan en la cola de transmisión de clúster predeterminada. Si establece el atributo *ClusterChannelName* en blancos, el canal conmuta a la cola de transmisión de clúster predeterminada cuando se reinicia el canal. La cola predeterminada es SYSTEM.CLUSTER.TRANSMIT.*ChannelName* o SYSTEM.CLUSTER.TRANSMIT.QUEUE, en función del valor del atributo **DefClusterXmitQueueType** del gestor de colas.

Al especificar asteriscos, "*", en **ClusterChannelName**, puede asociar una cola de transmisión con un conjunto de canales de clúster emisor. Los asteriscos pueden estar al principio, al final o en cualquier posición intermedia de la serie de nombre de canal. **ClusterChannelName** está limitado a una longitud de 20 caracteres: MQ_CHANNEL_NAME_LENGTH.

ClusterName (MQCHAR48)

Es el nombre del clúster al que pertenece la cola.

Tabla 606. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X		X	X	X

Si la cola pertenece a más de un clúster, *ClusterNameList* especifica el nombre de un objeto de lista de nombres que identifica los clústeres y *ClusterName* está en blanco. Al menos uno de *ClusterName* y *ClusterNameList* debe estar en blanco.

Para determinar el valor de este atributo, utilice el selector MQCA_CLUSTER_NAME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_CLUSTER_NAME_LENGTH.

ClusterNameList (MQCHAR48)

Es el nombre de un objeto de lista de nombres que contiene los nombres de los clústeres a los que pertenece esta cola.

Tabla 607. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X		X	X	

Si la cola sólo pertenece a un clúster, el objeto de lista de nombres sólo contiene un nombre. De forma alternativa, se puede utilizar *ClusterName* para especificar el nombre del clúster, en cuyo caso *ClusterNameList* está en blanco. Al menos uno de *ClusterName* y *ClusterNameList* debe estar en blanco.

Para determinar el valor de este atributo, utilice el selector MQCA_CLUSTER_NAMELIST con la llamada MQINQ. La longitud de este atributo la proporciona MQ_NAMELIST_NAME_LENGTH.

CLWLQueuePriority (MQLONG)

Es la prioridad de cola de carga de trabajo de clúster, un valor comprendido entre 0 y 9 que representa la prioridad de la cola.

Tabla 608. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X		X	X	X

Para obtener más información, consulte [Colas de clúster](#).

Para determinar el valor de este atributo, utilice el selector MQIA_CLWL_Q_PRIORITY con la llamada MQINQ.

CLWLQueueRank (MQLONG)

Este es el rango de cola de carga de trabajo de clúster, un valor en el rango de 0 a 9 que representa el rango de la cola.

Tabla 609. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X		X	X	X

Para obtener más información, consulte [Colas de clúster](#).

Para determinar el valor de este atributo, utilice el selector MQIA_CLWL_Q_RANK con la llamada MQINQ.

CLWLUseQ (MQLONG)

Esto define el comportamiento de un MQPUT cuando la cola de destino tiene una instancia local y al menos una instancia de clúster remoto. Si la transferencia se origina en un canal de clúster, este atributo no es aplicable.

Tabla 610. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X				

El valor puede ser uno de los siguientes:

MQCLWL_USEQ_ANY

Utilice colas remotas y locales.

MQCLWL_USEQ_LOCAL

No utilice colas remotas.

MQCLWL_USEQ_AS_Q_MGR

Heredar definición de MQIA_CLWL_USEQ del gestor de colas.

Para obtener más información, consulte [Colas de clúster](#).

Para determinar el valor de este atributo, utilice el selector MQIA_CLWL_USEQ con la llamada MQINQ. La longitud de este atributo la proporciona MQ_CLWL_USEQ_LENGTH.

CreationDate (MQCHAR12)

Es la fecha en la que se creó la cola.

Tabla 611. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X				

El formato de la fecha es YYYY-MM-DD, rellenado con dos espacios en blanco al final para que la longitud sea de 12 bytes (por ejemplo, 2013-09-23--, donde -- representa 2 caracteres en blanco).

- En IBM i, la fecha de creación de una cola puede diferir de la de la entidad del sistema operativo subyacente (archivo o espacio de usuario) que representa la cola.

Para determinar el valor de este atributo, utilice el selector MQCA_CREATION_DATE con la llamada MQINQ. La longitud de este atributo la proporciona MQ_CREATION_DATE_LENGTH.

CreationTime (MQCHAR8)

Es la hora en que se ha creado la cola.

Tabla 612. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X				

El formato de la hora es HH.MM.SS utilizando el reloj de 24 horas, con un cero inicial si la hora es inferior a 10 (por ejemplo, 09.10.20).

- En z/OS, la hora es Greenwich Mean Time (GMT), sujeta a que el reloj del sistema se establezca correctamente en GMT.
- En otros entornos, la hora es local.
- En IBM i, la hora de creación de una cola puede diferir de la de la entidad del sistema operativo subyacente (archivo o espacio de usuario) que representa la cola.

Para determinar el valor de este atributo, utilice el selector MQCA_CREATION_TIME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_CREATION_TIME_LENGTH.

CurrentQDepth (MQLONG)

Es el número de mensajes que hay en la cola actualmente.

Tabla 613. Tipos de colas a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X				

Se incrementa durante una llamada MQPUT y durante la reversión de una llamada MQGET. Se decrementa durante una llamada MQGET sin exploración y durante la reversión de una llamada MQPUT. El efecto de esto es que el recuento incluye mensajes que se han colocado en la cola dentro de una unidad de trabajo, pero que aún no se han confirmado, aunque no sean elegibles para ser recuperados por la llamada MQGET. Del mismo modo, excluye los mensajes que se han recuperado dentro de una unidad de trabajo mediante la llamada MQGET, pero que aún no se han confirmado.

El recuento también incluye los mensajes que han superado su tiempo de caducidad pero que aún no se han descartado, aunque estos mensajes no se pueden recuperar. Consulte [MQMD - Campo de caducidad](#) para obtener más información.

El procesamiento por unidades de trabajo y la segmentación de mensajes pueden provocar *CurrentQDepth* que supere *MaxQDepth*. Sin embargo, esto no afecta a la capacidad de recuperación de los mensajes; todos los mensajes de la cola se pueden recuperar utilizando la llamada MQGET de la forma habitual.

El valor de este atributo fluctúa a medida que opera el gestor de colas.

Para determinar el valor de este atributo, utilice el selector MQIA_CURRENT_Q_DEPTH con la llamada MQINQ.

DefaultPutResponse (MQLONG)

Especifica el tipo de respuesta que debe utilizarse para las operaciones de colocación en la cola cuando una aplicación específica MQPMO_RESPONSE_AS_Q_DEF.

Tabla 614. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X	X	X	

Es uno de los valores siguientes:

MQPRT_SYNC_RESPONSE

La operación de colocación se emite de forma síncrona, devolviendo una respuesta.

MQPRT_ASYNC_RESPONSE

La operación de transferencia se emite de forma asíncrona, devolviendo un subconjunto de campos MQMD.

DefBind (MQLONG)

Este es el enlace predeterminado que se utiliza cuando se especifica MQOO_BIND_AS_Q_DEF en la llamada MQOPEN y la cola es una cola de clúster.

Tabla 615. Tipos de colas a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X		X	X	X

El valor puede ser uno de los siguientes:

MQBND_BIND_ON_OPEN

Enlace fijado por llamada MQOPEN.

MQBND_BIND_NO_FIJO

Encuadernación no fija.

MQBND_BIND_ON_GROUP

Permite a una aplicación solicitar que un grupo de mensajes se asigne en su totalidad a la misma instancia de destino.

Para determinar el valor de este atributo, utilice el selector MQIA_DEF_BIND con la llamada MQINQ.

DefinitionType (MQLONG)

Indica cómo se ha definido la cola.

Tabla 616. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X			

El valor puede ser uno de los siguientes:

MQQDT_PREDEFINED

La cola es una cola permanente creada por el administrador del sistema; sólo el administrador del sistema puede suprimirla.

Las colas predefinidas se crean utilizando el mandato MQSC de DEFINE y solo se pueden suprimir utilizando el mandato MQSC de DELETE . Las colas predefinidas no se pueden crear a partir de colas modelo.

Los comandos pueden ser emitidos por un operador o por un usuario autorizado que envíe un mensaje de comando a la cola de entrada de comandos (para más información, consulte [el atributo CommandInputQName](#)).

MQQDT_PERMANENT_DYNAMIC

La cola es una cola permanente creada por una aplicación que emite una llamada MQOPEN con el nombre de una cola modelo especificada en el descriptor de objeto MQOD. La definición de cola modelo tenía el valor MQQDT_PERMANENT_DYNAMIC para el atributo **DefinitionType** .

Este tipo de cola se puede suprimir utilizando la llamada MQCLOSE. Consulte [“MQCLOSE - Cerrar objeto” en la página 651](#) para obtener más detalles.

El valor del atributo **QSGDisp** para una cola dinámica permanente es MQQSGD_Q_MGR.

MQQDT_TEMPORARY_DYNAMIC

La cola es una cola temporal creada por una aplicación que emite una llamada MQOPEN con el nombre de una cola modelo especificada en el descriptor de objeto MQOD. La definición de cola modelo tenía el valor MQQDT_TEMPORARY_DYNAMIC para el atributo **DefinitionType**.

Este tipo de cola se suprime automáticamente mediante la llamada MQCLOSE cuando la aplicación que la ha creado la cierra.

El valor del atributo **QSGDisp** para una cola dinámica temporal es MQQSGD_Q_MGR.

MQQDT_COMPARTIDO_DINÁMICO

La cola es una cola permanente compartida creada por una aplicación que emite una llamada MQOPEN con el nombre de una cola modelo especificada en el descriptor de objeto MQOD. La definición de cola modelo tenía el valor MQQDT_SHARED_DYNAMIC para el atributo **DefinitionType**.

Este tipo de cola se puede suprimir utilizando la llamada MQCLOSE. Consulte [“MQCLOSE - Cerrar objeto”](#) en la página 651 para obtener más detalles.

El valor del atributo **QSGDisp** para una cola dinámica compartida es MQQSGD_SHARED.

Este atributo en una definición de cola modelo no indica cómo se ha definido la cola modelo, porque las colas modelo siempre están predefinidas. En su lugar, el valor de este atributo en la cola modelo se utiliza para determinar el *DefinitionType* de cada una de las colas dinámicas creadas a partir de la definición de cola modelo utilizando la llamada MQOPEN.

Para determinar el valor de este atributo, utilice el selector MQIA_DEFINITION_TYPE con la llamada MQINQ.

DefInputOpenOption (MQLONG)

Esta es la forma predeterminada en la que se abre la cola para la entrada.

Tabla 617. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X			

Se aplica si se especifica la opción MQOO_INPUT_AS_Q_DEF en la llamada MQOPEN cuando se abre la cola. El valor puede ser uno de los siguientes:

MQOO_INPUT_EXCLUSIVE

Abra la cola para obtener mensajes con acceso exclusivo.

La cola se abre para su uso con las llamadas MQGET posteriores. La llamada falla con el código de razón MQRC_OBJECT_IN_USE si la cola está abierta actualmente por esta u otra aplicación para cualquier tipo de entrada (MQOO_INPUT_SHARED o MQOO_INPUT_EXCLUSIVE).

MQOO_INPUT_SHARED

Abra la cola para obtener mensajes con acceso compartido.

La cola se abre para su uso con las llamadas MQGET posteriores. La llamada puede realizarse correctamente si la cola está abierta actualmente por esta u otra aplicación con MQOO_INPUT_SHARED, pero falla con el código de razón MQRC_OBJECT_IN_USE si la cola está abierta actualmente con MQOO_INPUT_EXCLUSIVE.

Para determinar el valor de este atributo, utilice el selector MQIA_DEF_INPUT_OPEN_OPTION con la llamada MQINQ.

DefPersistence (MQLONG)

Es la persistencia predeterminada de los mensajes en la cola. Se aplica si se especifica MQPER_PERSISTENCE_AS_Q_DEF en el descriptor de mensaje cuando se coloca el mensaje.

Tabla 618. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X	X	X	X

Si hay más de una definición en la vía de acceso de resolución de nombre de cola, la persistencia predeterminada se toma del valor de este atributo en la *primera* definición de la vía de acceso en el momento de la llamada MQPUT o MQPUT1 . Puede ser lo siguiente:

- una cola alias
- Una cola local
- Una definición local de una cola remota
- Un alias de gestor de colas
- Una cola de transmisión (por ejemplo, la cola *DefXmitQName*)

El valor puede ser uno de los siguientes:

MQPER_PERSISTENT

El mensaje sobrevive a las anomalías del sistema y se reinicia el gestor de colas. Los mensajes persistentes no se pueden colocar en:

- Colas dinámicas temporales
- Colas compartidas que se correlacionan con un objeto CFSTRUCT en CFLEVEL (2) o inferior, o donde el objeto CFSTRUCT se define como RECOVER (NO).

Los mensajes persistentes se pueden colocar en colas dinámicas permanentes y colas predefinidas.

MQPER_NOT_PERSISTENT

Normalmente, el mensaje no sobrevive a las anomalías del sistema o a los reinicios del gestor de colas. Esto se aplica incluso si se encuentra una copia intacta del mensaje en el almacenamiento auxiliar durante un reinicio del gestor de colas.

En el caso de colas compartidas, los mensajes no persistentes *sí* sobreviven a los reinicios de los gestores de colas en el grupo de compartición de colas, pero no sobreviven a los errores del recurso de acoplamiento utilizado para almacenar mensajes en las colas compartidas.

Los mensajes persistentes y no persistentes pueden existir en la misma cola.

Para determinar el valor de este atributo, utilice el selector MQIA_DEF_PERSISTENCE con la llamada MQINQ.

DefPriority (MQLONG)

Esta es la prioridad predeterminada para los mensajes de la cola. Esto se aplica si se especifica MQPRI_PRIORITY_AS_Q_DEF en el descriptor de mensaje cuando el mensaje se coloca en la cola.

Tabla 619. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X	X	X	X

Si hay más de una definición en la vía de acceso de resolución de nombre de cola, la prioridad predeterminada para el mensaje se toma del valor de este atributo en la *primera* definición de la vía de acceso en el momento de la operación de colocación. Puede ser lo siguiente:

- una cola alias
- Una cola local
- Una definición local de una cola remota
- Un alias de gestor de colas

- Una cola de transmisión (por ejemplo, la cola *DefXmitQName*)

La forma en que se coloca un mensaje en una cola depende del valor del atributo **MsgDeliverySequence** de la cola:

- Si el atributo **MsgDeliverySequence** es MQMDS_PRIORITY, la posición lógica en la que se coloca un mensaje en la cola depende del valor del campo *Priority* en el descriptor de mensaje.
- Si el atributo **MsgDeliverySequence** es MQMDS_FIFO, los mensajes se colocan en la cola como si tuvieran una prioridad igual a la *DefPriority* de la cola resuelta, independientemente del valor del campo *Priority* en el descriptor de mensaje. Sin embargo, el campo *Priority* conserva el valor especificado por la aplicación que ha colocado el mensaje. Consulte el [atributo MsgDeliverySequence](#) para obtener más información.

Las prioridades están en el rango de cero (menor) a *MaxPriority* (mayor); consulte el atributo [MaxPriority](#).

Para determinar el valor de este atributo, utilice el selector MQIA_DEF_PRIORITY con la llamada MQINQ.

DefReadAhead (MQLONG)

Especifica el comportamiento de lectura anticipada predeterminado para los mensajes no persistentes entregados al cliente.

Tabla 620. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X	X		

DefReadAhead puede establecerse en uno de los siguientes valores::

MQREADA_NO

Los mensajes no persistentes no se envían al cliente antes de que las aplicaciones los soliciten. Puede perderse un mensaje no persistente como máximo, si el cliente finaliza de forma anómala.

MQREADA_SÍ

Los mensajes no persistentes se envían al cliente antes de que una aplicación los solicite. Los mensajes no persistentes se pueden perder si el cliente finaliza de forma anómala o si el cliente no consume todos los mensajes que se envían.

MQREADA_DESACTIVADO

Lectura anticipada de mensajes no persistentes en no habilitados para esta cola. Los mensajes no se envían al cliente independientemente de si la aplicación cliente solicita la lectura anticipada.

Para determinar el valor de este atributo, utilice el selector MQIA_DEF_READ_AHEAD con la llamada MQINQ.

DefPResp (MQLONG)

El atributo de tipo de respuesta put por defecto (DEFPRESP) define el valor utilizado por las aplicaciones cuando el PutResponseType dentro de MQPMO se ha establecido en MQPMO_RESPONSE_AS_Q_DEF. Este atributo es válido para todos los tipos de cola.

Tabla 621. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X	X	X	X

El valor puede ser uno de los siguientes:

SYNC

La operación de colocación se emite de forma síncrona devolviendo una respuesta.

ASYNCR

La operación de transferencia se emite de forma asíncrona, devolviendo un subconjunto de campos MQMD.

Para determinar el valor de este atributo, utilice el selector MQIA_DEF_PUT_RESPONSE_TYPE con la llamada MQINQ.

DistLists (MQLONG)

Indica si los mensajes de lista de distribución se pueden colocar en la cola.

Tabla 622. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X			

Un agente de canal de mensajes (MCA) establece el atributo para informar al gestor de colas local si el gestor de colas del otro extremo del canal soporta listas de distribución. Este último gestor de colas (denominado gestor de colas de *asociación*) es el que recibe a continuación el mensaje, después de que un MCA emisor lo haya eliminado de la cola de transmisión local.

El MCA emisor establece el atributo siempre que establece una conexión con el MCA receptor en el gestor de colas asociado. De este modo, el MCA emisor puede hacer que el gestor de colas local coloque en la cola de transmisión sólo los mensajes que el gestor de colas asociado puede procesar correctamente.

Este atributo se utiliza principalmente con colas de transmisión, pero el proceso descrito se realiza independientemente del uso definido para la cola (consulte [Atributo de uso](#)).

El valor puede ser uno de los siguientes:

MQDL_COMPATIBLE

Los mensajes de lista de distribución pueden almacenarse en la cola y transmitirse al gestor de colas asociado en ese formato. Esto reduce la cantidad de proceso necesaria para enviar el mensaje a varios destinos.

MQDL_NO_COMPATIBLE

Los mensajes de lista de distribución no se pueden almacenar en la cola porque el gestor de colas asociado no da soporte a las listas de distribución. Si una aplicación coloca un mensaje de lista de distribución y ese mensaje se va a colocar en esta cola, el gestor de colas divide el mensaje de lista de distribución y coloca los mensajes individuales en la cola. Esto aumenta la cantidad de proceso necesario para enviar el mensaje a varios destinos, pero garantiza que el gestor de colas asociado procese correctamente los mensajes.

Para determinar el valor de este atributo, utilice el selector MQIA_DIST_LISTS con la llamada MQINQ. Para cambiar el valor de este atributo, utilice la llamada MQSET.

Este atributo no está soportado en z/OS.

HardenGetBackout (MQLONG)

Para cada mensaje, se mantiene un recuento del número de veces que una llamada MQGET recupera el mensaje dentro de una unidad de trabajo, y dicha unidad de trabajo se restituye posteriormente.

Tabla 623. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X			

Este recuento está disponible en el campo *BackoutCount* del descriptor de mensaje después de que se haya completado la llamada MQGET.

El recuento de restituciones de mensajes sobrevive a los reinicios del gestor de colas. Sin embargo, para asegurarse de que el recuento es preciso, la información debe *guardarse* (grabarse en disco u otro dispositivo de almacenamiento permanente) cada vez que una llamada MQGET recupera un mensaje

dentro de una unidad de trabajo para esta cola. Si esto no se hace, el gestor de colas falla y la llamada MQGET se restituye, el recuento puede o no incrementarse.

Sin embargo, el endurecimiento de la información para cada llamada MQGET dentro de una unidad de trabajo impone un coste de proceso adicional, por lo tanto, establezca el atributo **HardenGetBackout** en MQQA_BACKOUT_HARAPARTADO sólo si es esencial que el recuento sea preciso.

En Multiplataformas, el recuento de restituciones de mensajes siempre está protegido, independientemente del valor de este atributo.

Son posibles los siguientes valores:

MQQA_BACKOUT_HARDENED

El refuerzo se utiliza para asegurarse de que el recuento de restituciones para los mensajes de esta cola es preciso.

MQQA_BACKOUT_NOT_HARTIZADO

El refuerzo no se utiliza para asegurarse de que el recuento de restituciones para los mensajes de esta cola es preciso. Por lo tanto, el recuento puede ser menor de lo que debería ser.

Para determinar el valor de este atributo, utilice el selector MQIA_HARDEN_GET_BACKOUT con la llamada MQINQ.

IndexType (MQLONG)

Esto especifica el tipo de índice que el gestor de colas mantiene para los mensajes de la cola.

Tabla 624. Tipos de colas a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X			

El tipo de índice necesario depende de cómo la aplicación recupere los mensajes y de si la cola es compartida o no compartida (véase el atributo QSGDisp). Los siguientes valores son posibles para *IndexType*:

MQIT_NINGUNO

El gestor de colas no mantiene ningún índice para esta cola. Utilice este valor para las colas que suelen procesarse de forma secuencial, es decir, sin utilizar ningún criterio de selección en la llamada MQGET.

MQIT_MSG_ID

El gestor de colas mantiene un índice que utiliza los identificadores de los mensajes de la cola. Utilice estas colas de valores cuando la aplicación suele recuperar mensajes utilizando el identificador de mensaje como criterio de selección en la llamada MQGET.

MQIT_CORREL_ID

El gestor de colas mantiene un índice que utiliza los identificadores de correlación de los mensajes de la cola. Utilice este valor para las colas en las que la aplicación suele recuperar mensajes utilizando el identificador de correlación como criterio de selección en la llamada MQGET.

MQIT_MSG_TOKEN

Importante: Este tipo de índice solo debe utilizarse para colas utilizadas con el IBM MQ producto z/OS Workflow for.

El gestor de colas mantiene un índice que utiliza los tokens de los mensajes de la cola para su uso con las funciones del gestor de carga de trabajo (WLM) de z/OS.

Debe especificar esta opción para las colas gestionadas por WLM; no la especifique para ningún otro tipo de cola. Además, no utilice este valor para una cola en la que una aplicación no utilice las funciones z/OS del gestor de carga de trabajo, sino que recupere mensajes utilizando el token de mensaje como criterio de selección en la llamada MQGET.

MQIT_GROUP_ID

El gestor de colas mantiene un índice que utiliza los identificadores de grupo de los mensajes de la cola. Este valor debe utilizarse para colas en las que la aplicación recupera mensajes utilizando la opción MQGMO_LOGICAL_ORDER en la llamada MQGET.

Una cola con este tipo de índice no puede ser una cola de transmisión. Se debe definir una cola compartida con este tipo de índice para asignarla a un objeto CFSTRUCT en CFLEVEL(3) o superior.

Nota:

1. El orden físico de los mensajes en una cola con tipo de índice MQIT_GROUP_ID no está definido, ya que la cola está optimizada para la recuperación eficiente de mensajes utilizando la opción MQGMO_LOGICAL_ORDER en la llamada MQGET. Esto significa que el orden físico de los mensajes no suele ser el orden en que llegaron a la cola.
2. Si una cola MQIT_GROUP_ID tiene un *MsgDeliverySequence* de MQMDS_PRIORITY, el gestor de colas utiliza las prioridades de mensaje 0 y 1 para optimizar la recuperación de mensajes en orden lógico. Como resultado, el primer mensaje de un grupo no debe tener una prioridad de cero o uno; si la tiene, el mensaje se procesa como si tuviera una prioridad de dos. El *Priority* campo de la estructura MQMD no se modifica.

Para obtener más información sobre los grupos de mensajes, consulte la descripción de las opciones de grupo y segmento en [el campo MQGMO - Opciones](#).

El tipo de índice que debe utilizarse en los distintos casos se muestra en [Tabla 625 en la página 858](#) y [Tabla 626 en la página 859](#).

Tabla 625. Valores sugeridos o requeridos del tipo de índice de cola cuando no se especifica MQGMO_LOGICAL_ORDER		
Criterios de selección en la convocatoria MQGET	Tipo de índice para cola no compartida	Tipo de índice para cola compartida
Ninguna	Cualquiera	Cualquiera
Selección mediante un identificador:		
Identificador de mensaje	MQIT_MSG_ID sugerido	Se requiere MQIT_NONE o MQIT_MSG_ID; se sugiere MQIT_MSG_ID
Identificador de correlación	MQIT_CORREL_ID sugerido	Se requiere MQIT_CORREL_ID
Identificador de grupo	MQIT_GROUP_ID sugerido	Se requiere MQIT_GROUP_ID
Selección mediante dos identificadores:		
Identificador de mensaje más identificador de correlación	MQIT_MSG_ID o MQIT_CORREL_ID sugerido	Se requiere MQIT_NONE o MQIT_MSG_ID o MQIT_CORREL_ID (Por motivos de eficiencia, se sugiere elegir el tipo de índice que coincida con el campo MQMD, que tendrá las claves más distintivas)
Identificador de mensaje más identificador de grupo	MQIT_MSG_ID o MQIT_GROUP_ID sugerido	No soportado
Identificador de correlación más identificador de grupo	Se sugiere MQIT_CORREL_ID o MQIT_GROUP_ID	No soportado
Selección mediante tres identificadores:		

Tabla 625. Valores sugeridos o requeridos del tipo de índice de cola cuando no se especifica MQGMO_LOGICAL_ORDER (continuación)

Criterios de selección en la convocatoria MQGET	Tipo de índice para cola no compartida	Tipo de índice para cola compartida
Identificador de mensaje más identificador de correlación más identificador de grupo	MQIT_MSG_ID o MQIT_CORREL_ID o MQIT_GROUP_ID sugerido	No soportado
Selección utilizando criterios relacionados con el grupo:		
Identificador de grupo más número de secuencia del mensaje	Se requiere MQIT_GROUP_ID	Se requiere MQIT_GROUP_ID
Número de secuencia del mensaje (debe ser 1)	Se requiere MQIT_GROUP_ID	Se requiere MQIT_GROUP_ID
Selección mediante token de mensaje:		
Token de mensaje para uso de la aplicación	No utilice MQIT_MSG_TOKEN	
Token de mensaje para uso de WLM	Se requiere MQIT_MSG_TOKEN	No soportado

Tabla 626. Valores sugeridos o requeridos del tipo de índice de cola cuando se especifica MQGMO_LOGICAL_ORDER

Criterios de selección en la convocatoria MQGET	Tipo de índice para cola no compartida	Tipo de índice para cola compartida
Ninguna	Se requiere MQIT_GROUP_ID	Se requiere MQIT_GROUP_ID
Selección mediante un identificador:		
Identificador de mensaje	Se requiere MQIT_GROUP_ID	No soportado
Identificador de correlación	Se requiere MQIT_GROUP_ID	No soportado
Identificador de grupo	Se requiere MQIT_GROUP_ID	Se requiere MQIT_GROUP_ID
Selección mediante dos identificadores:		
Identificador de mensaje más identificador de correlación	Se requiere MQIT_GROUP_ID	No soportado
Identificador de mensaje más identificador de grupo	Se requiere MQIT_GROUP_ID	No soportado
Identificador de correlación más identificador de grupo	Se requiere MQIT_GROUP_ID	No soportado
Selección mediante tres identificadores:		
Identificador de mensaje más identificador de correlación más identificador de grupo	Se requiere MQIT_GROUP_ID	No soportado

Para determinar el valor de este atributo, utilice el selector MQIA_INDEX_TYPE con la llamada MQINQ.



Este atributo solo es compatible con z/OS.

InhibitGet (MQLONG)

Esto controla si se permiten operaciones get para esta cola.

Tabla 627. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X	X		

Si la cola es una cola alias, las operaciones get deben estar permitidas tanto para el alias como para la cola base en el momento de la operación get, para que la llamada MQGET sea satisfactoria. El valor puede ser uno de los siguientes:

MQQA_GET_INHIBITED

Las operaciones de obtención están inhibidas.

Las llamadas MQGET fallan con el código de razón MQRC_GET_inhibiTED. Esto incluye las llamadas MQGET que especifican MQGMO_BROWSE_FIRST o MQGMO_BROWSE_NEXT.

Nota: Si una llamada MQGET que opera dentro de una unidad de trabajo se completa correctamente, el cambio del valor del atributo **InhibitGet** posteriormente a MQQA_GET_inhibiTED no impide que se confirme la unidad de trabajo.

MQQA_GET_ALLOWED

Las operaciones de obtención están permitidas.

Para determinar el valor de este atributo, utilice el selector MQIA_INHIBID_GET con la llamada MQINQ. Para cambiar el valor de este atributo, utilice la llamada MQSET.

InhibitPut (MQLONG)

Esto controla si se permiten las operaciones de colocación para esta cola.

Tabla 628. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X	X	X	X

Si hay más de una definición en la vía de acceso de resolución de nombre de cola, se deben permitir operaciones de colocación para *cada* definición en la vía de acceso (incluidas las definiciones de alias de gestor de colas) en el momento de la operación de colocación, para que la llamada MQPUT o MQPUT1 sea satisfactoria. El valor puede ser uno de los siguientes:

MQQA_PUT_INHIBITED

Las operaciones de colocación están inhibidas.

Las llamadas MQPUT y MQPUT1 fallan con el código de razón MQRC_PUT_inhibiTED.

Nota: Si una llamada MQPUT que opera dentro de una unidad de trabajo se completa correctamente, el cambio del valor del atributo **InhibitPut** posteriormente a MQQA_PUT_inhibiTED no impide que se confirme la unidad de trabajo.

MQQA_PUT_ALLOWED

Las operaciones de colocación están permitidas.

Para determinar el valor de este atributo, utilice el selector MQIA_INHIBID_PUT con la llamada MQINQ. Para cambiar el valor de este atributo, utilice la llamada MQSET.

InitiationQName (MQCHAR48)

Es el nombre de una cola definida en el gestor de colas local; la cola debe ser de tipo MQQT_LOCAL.

Tabla 629. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remoto	Clúster
X				

El gestor de colas envía un mensaje desencadenante a la cola de inicio cuando es necesario iniciar la aplicación como resultado de un mensaje que llega a la cola a la que pertenece este atributo. La cola de inicio debe ser supervisada por una aplicación de supervisor desencadenante que inicie la aplicación adecuada después de recibir el mensaje desencadenante.

Para determinar el valor de este atributo, utilice el selector MQCA_INITIATION_Q_NAME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_Q_NAME_LENGTH.

MaxMsgLength (MQLONG)

Este es un límite superior para la longitud del mensaje *físico* más largo que se puede colocar en la cola.

Tabla 630. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remoto	Clúster
X	X			

Sin embargo, debido a que el atributo de cola **MaxMsgLength** se puede establecer independientemente del atributo de gestor de colas **MaxMsgLength**, el límite superior real para la longitud del mensaje físico más largo que se puede colocar en la cola es el menor de estos dos valores.

Si el gestor de colas da soporte a la segmentación, es posible que una aplicación coloque un mensaje *lógico* que sea más largo que el menor de los dos atributos **MaxMsgLength**, pero sólo si la aplicación especifica el distintivo MQMF_SEGMENTATION_ALLOWED en MQMD. Si se especifica ese distintivo, el límite superior para la longitud de un mensaje lógico es de 999.999.999 bytes, pero normalmente las restricciones de recursos impuestas por el sistema operativo, o por el entorno en el que se ejecuta la aplicación, dan como resultado un límite inferior.

Un intento de colocar en la cola un mensaje demasiado largo falla con uno de los siguientes códigos de razón:

- MQRC_MSG_TOO_BIG_FOR_Q si el mensaje es demasiado grande para la cola
- MQRC_MSG_TOO_BIG_FOR_Q_MGR si el mensaje es demasiado grande para el gestor de colas, pero no demasiado grande para la cola

El límite inferior para el atributo **MaxMsgLength** es cero; el límite superior es de 100 MB (104 857 600 bytes).

Para obtener más información, consulte [MQPUT-Parámetro BufferLength](#).

Para determinar el valor de este atributo, utilice el selector MQIA_MAX_MSG_LENGTH con la llamada MQINQ.

MaxQDepth (MQLONG)

Este es el límite máximo definido para el número de mensajes físicos que pueden existir en la cola en un momento dado.

Tabla 631. Tipos de colas a los que se aplica este atributo

Local	Modelo	Alias	Remoto	Clúster
X	X			

El intento de colocar un mensaje en una cola que ya contiene **MaxQDepth** mensajes falla con el código de motivo MQRC_Q_FULL.

El procesamiento por unidades de trabajo y la segmentación de mensajes pueden provocar que el número real de mensajes físicos en la cola supere **MaxQDepth**. Sin embargo, esto no afecta a la

capacidad de recuperación del mensaje, ya que todos los mensajes de la cola se pueden recuperar mediante la llamada MQGET.

El valor de este atributo es cero o mayor. El límite superior viene determinado por el entorno:

- En las siguientes plataformas, el valor no puede superar 999 999 999:

-  AIX
-  Linux
-  Windows
-  z/OS

-  En IBM i, el valor no puede superar 640 000.

Nota: El espacio de almacenamiento disponible para la cola podría agotarse incluso si hay menos de **MaxQDepth** mensajes en la cola.

Para determinar el valor de este atributo, utilice el selector MQIA_MAX_Q_DEPTH con la llamada MQINQ.

MsgDeliverySequence (MQLONG)

Tabla 632. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X			

Esto determina el orden en el que la llamada MQGET devuelve mensajes a la aplicación:

MQMDS_FIFO

Los mensajes se devuelven en orden FIFO (primero en entrar, primero en salir).

Una llamada MQGET devuelve el *primer* mensaje que cumple los criterios de selección especificados en la llamada, independientemente de la prioridad del mensaje.

MQMDS_PRIORITY

Los mensajes se devuelven en orden de prioridad.

Una llamada MQGET devuelve el mensaje de *prioridad más alta* que cumple los criterios de selección especificados en la llamada. Dentro de cada nivel de prioridad, los mensajes se devuelven en orden FIFO (primero en entrar, primero en salir).

- En z/OS, si la cola tiene un *IndexType* de MQIT_GROUP_ID, el atributo **MsgDeliverySequence** especifica el orden en el que se devuelven los grupos de mensajes a la aplicación. La secuencia particular en la que se devuelven los grupos viene determinada por la posición o prioridad del primer mensaje de cada grupo. El orden físico de los mensajes en la cola no está definido, ya que la cola está optimizada para una recuperación eficaz de mensajes utilizando la opción MQGMO_LOGICAL_ORDER en la llamada MQGET.
- En z/OS, si *IndexType* es MQIT_GROUP_ID y *MsgDeliverySequence* es MQMDS_PRIORITY, el gestor de colas utiliza las prioridades de mensaje cero y una para optimizar la recuperación de mensajes en orden lógico. Como resultado, el primer mensaje de un grupo no debe tener una prioridad de cero o uno; si lo tiene, el mensaje se procesa como si tuviera una prioridad de dos. El campo *Priority* de la estructura MQMD no se modifica.

Si los atributos relevantes se cambian mientras hay mensajes en la cola, la secuencia de entrega es la siguiente:

- El orden en el que la llamada MQGET devuelve los mensajes viene determinado por los valores de los atributos **MsgDeliverySequence** y **DefPriority** en vigor para la cola en el momento en que el mensaje llega a la cola:

- Si *MsgDeliverySequence* es MQMDS_FIFO cuando llega el mensaje, el mensaje se coloca en la cola como si su prioridad fuera *DefPriority*. Esto no afecta al valor del campo *Priority* en el descriptor de mensaje del mensaje; dicho campo conserva el valor que tenía cuando se colocó el mensaje por primera vez.
- Si *MsgDeliverySequence* es MQMDS_PRIORITY cuando llega el mensaje, el mensaje se coloca en la cola en el lugar adecuado a la prioridad proporcionada por el campo *Priority* en el descriptor de mensaje.

Si el valor del atributo **MsgDeliverySequence** cambia mientras hay mensajes en la cola, el orden de los mensajes en la cola no cambia.

Si el valor del atributo **DefPriority** se cambia mientras hay mensajes en la cola, los mensajes no se entregan necesariamente en orden FIFO, aunque el atributo **MsgDeliverySequence** se establezca en MQMDS_FIFO; los que se colocaron en la cola con la prioridad más alta se entregan primero.

Para determinar el valor de este atributo, utilice el selector MQIA_MSG_DELIVERY_SEQUENCE con la llamada MQINQ.

NonPersistentMessageClass (MQLONG)

El objetivo de fiabilidad para los mensajes no persistentes.

Tabla 633. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X			

Especifica las circunstancias en las que se descartan los mensajes no persistentes colocados en esta cola:

MQNPM_CLASS_NORMAL

Los mensajes no persistentes están limitados al tiempo de vida de la sesión del gestor de colas; los mensajes se descartan en el caso de un reinicio del gestor de colas. Sólo es válido para colas no compartidas y es el valor predeterminado.

MQNPM_CLASS_HIGH

El gestor de colas intenta retener los mensajes no persistentes durante el tiempo de vida de la cola. Es posible que los mensajes no persistentes se pierdan si se produce una anomalía. Este valor se aplica a las colas compartidas.

Para determinar el valor de este atributo, utilice el selector MQIA_NPM_CLASS con la llamada MQINQ.

OpenInputCount (MQLONG)

Es el número de descriptores de contexto que son válidos actualmente para eliminar mensajes de la cola mediante la llamada MQGET.

Tabla 634. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X				

Es el número total de manejadores de este tipo conocidos por el gestor de colas *local*. Si la cola es una cola compartida, el recuento no incluye las aperturas para la entrada que se han realizado para la cola en otros gestores de colas del grupo de compartición de colas al que pertenece el gestor de colas local.

El recuento incluye los descriptores de contexto en los que se ha abierto una cola alias que se resuelve en esta cola para entrada. El recuento no incluye los descriptores de contexto en los que se ha abierto la cola para acciones que no han incluido la entrada (por ejemplo, una cola abierta sólo para examinar).

El valor de este atributo fluctúa a medida que opera el gestor de colas.

Para determinar el valor de este atributo, utilice el selector MQIA_OPEN_INPUT_COUNT con la llamada MQINQ.

OpenOutputCount (MQLONG)

Es el número de descriptores de contexto que son válidos actualmente para añadir mensajes a la cola mediante la llamada MQPUT.

Tabla 635. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X				

Es el número total de manejadores de este tipo conocidos por el gestor de colas *local* ; no incluye las aperturas para salida que se han realizado para esta cola en los gestores de colas remotos. Si la cola es una cola compartida, el recuento no incluye las aperturas para la salida que se han realizado para la cola en otros gestores de colas del grupo de compartición de colas al que pertenece el gestor de colas local.

El recuento incluye los descriptores de contexto en los que se ha abierto una cola alias que se resuelve en esta cola para salida. El recuento no incluye los descriptores de contexto en los que se ha abierto la cola para acciones que no han incluido la salida (por ejemplo, una cola abierta sólo para consulta).

El valor de este atributo fluctúa a medida que opera el gestor de colas.

Para determinar el valor de este atributo, utilice el selector MQIA_OPEN_OUTPUT_COUNT con la llamada MQINQ.

ProcessName (MQCHAR48)

Es el nombre de un objeto de proceso definido en el gestor de colas local. El objeto de proceso identifica un programa que puede dar servicio a la cola.

Tabla 636. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X			

Para determinar el valor de este atributo, utilice el selector MQCA_PROCESS_NAME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_PROCESS_NAME_LENGTH.

PropertyControl (MQLONG)

Especifica cómo se manejan las propiedades de mensaje para los mensajes que se recuperan de las colas utilizando la llamada MQGET con la opción MQGMO_PROPERTIES_AS_Q_DEF.

Tabla 637. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X	X		

El valor puede ser uno de los siguientes:

MQPROP_ALL

Todas las propiedades del mensaje se incluyen con el mensaje cuando se entrega a la aplicación. Las propiedades, excepto las que se encuentran en el descriptor de mensaje (o extensión), se colocan en una o más cabeceras MQRFH2 en los datos del mensaje. Si se proporciona un descriptor de mensaje, el comportamiento es devolver las propiedades en el descriptor de mensaje.

COMPATIBILIDAD de MQPROP_COMPATIBILITY

Si el mensaje contiene una propiedad con el prefijo mcd., jms., usr. o mqext., todas las propiedades de mensaje se entregan a la aplicación en una cabecera MQRFH2 . De lo contrario, todas las propiedades del mensaje, excepto las que se encuentran en el descriptor de mensaje (o extensión), se descartan y dejan de estar accesibles para la aplicación. Este es el valor predeterminado; permite que las

aplicaciones que esperan que las propiedades relacionadas con JMS estén en una cabecera MQRFH2 en los datos del mensaje sigan funcionando sin modificar. Si se proporciona un descriptor de mensaje, el comportamiento es devolver las propiedades en el descriptor de mensaje.

MQPROP_FORCE_MQRFH2

Las propiedades siempre se devuelven en los datos de mensaje en una cabecera MQRFH2 independientemente de si la aplicación especifica un manejador de mensajes. Se ignora un manejador de mensajes válido proporcionado en el campo MsgHandle de la estructura MQGMO en la llamada MQGET. Las propiedades del mensaje no son accesibles a través del manejador de mensajes.

MQPROP_NONE

Todas las propiedades del mensaje, excepto las del descriptor de mensaje (o extensión), se eliminan del mensaje antes de que el mensaje se entregue a la aplicación. Si se proporciona un descriptor de mensaje, el comportamiento es devolver las propiedades en el descriptor de mensaje.

Este parámetro es aplicable a las colas Local, Alias y Modelo. Para determinar su valor, utilice el selector MQIA_PROPERTY_CONTROL con la llamada MQINQ.

QDepthHighEvent (MQLONG)

Esto controla si se generan sucesos de profundidad de cola alta.

Tabla 638. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X			

Un suceso de profundidad de cola alta indica que una aplicación ha colocado un mensaje en una cola y esto ha hecho que el número de mensajes de la cola sea mayor o igual que el umbral de profundidad de cola alta (consulte el atributo **QDepthHighLimit**).

Nota: El valor de este atributo puede cambiar dinámicamente.

El valor puede ser uno de los siguientes:

MQEVR_DISABLED

Informes de sucesos inhabilitados.

MQEVR_ENABLED

Informes de sucesos habilitados.

Para obtener más información sobre los sucesos, consulte [Supervisión de sucesos](#).

Para determinar el valor de este atributo, utilice el selector MQIA_Q_DEPTH_HIGH_EVENT con la llamada MQINQ.

Este atributo está soportado en z/OS, pero la llamada MQINQ no se puede utilizar para determinar su valor.

QDepthHighLimit (MQLONG)

Es el umbral con el que se compara la profundidad de cola para generar un suceso Profundidad de cola alta.

Tabla 639. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X			

Este suceso indica que una aplicación ha colocado un mensaje en una cola y que esto ha hecho que el número de mensajes en la cola sea mayor o igual que el umbral alto de profundidad de cola. Véase el [atributo QDepthHighEvent](#).

El valor se expresa como un porcentaje de la profundidad de cola máxima (atributo **MaxQDepth**), y es mayor o igual que 0 y menor o igual que 100. El valor predeterminado es 80.

Para determinar el valor de este atributo, utilice el selector MQIA_Q_DEPTH_HIGH_LIMIT con la llamada MQINQ.

Este atributo está soportado en z/OS, pero la llamada MQINQ no se puede utilizar para determinar su valor.

QDepthLowEvent (MQLONG)

Esto controla si se generan sucesos de profundidad de cola baja.

Tabla 640. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X			

Un evento de profundidad de cola baja indica que una aplicación ha recuperado un mensaje de una cola, y que esto ha causado que el número de mensajes en la cola sea menor o igual al umbral de profundidad de cola baja (ver atributo [QDepthLowLimit](#)).

Nota: El valor de este atributo puede cambiar dinámicamente.

El valor puede ser uno de los siguientes:

MQEVR_DISABLED

Informes de sucesos inhabilitados.

MQEVR_ENABLED

Informes de sucesos habilitados.

Para obtener más información sobre los sucesos, consulte [Supervisión de sucesos](#).

Para determinar el valor de este atributo, utilice el selector MQIA_Q_DEPTH_LOW_EVENT con la llamada MQINQ.

Este atributo está soportado en z/OS, pero la llamada MQINQ no se puede utilizar para determinar su valor.

QDepthLowLimit (MQLONG)

Es el umbral con el que se compara la profundidad de cola para generar un suceso Profundidad de cola baja.

Tabla 641. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X			

Este suceso indica que una aplicación ha recuperado un mensaje de una cola y que esto ha hecho que el número de mensajes de la cola sea menor o igual que el umbral bajo de profundidad de cola. Véase el atributo [QDepthLowEvent](#).

El valor se expresa como un porcentaje de la profundidad de cola máxima (atributo **MaxQDepth**), y es mayor o igual que 0 y menor o igual que 100. El valor predeterminado es 20.

Para determinar el valor de este atributo, utilice el selector MQIA_Q_DEPTH_LOW_LIMIT con la llamada MQINQ.

Este atributo está soportado en z/OS, pero la llamada MQINQ no se puede utilizar para determinar su valor.

QDepthMaxEvent (MQLONG)

Esto controla si se generan sucesos de cola llena. Un suceso Cola llena indica que una colocación en una cola se ha rechazado porque la cola está llena, es decir, la profundidad de cola ya ha alcanzado su valor máximo.

Tabla 642. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X			

Nota: El valor de este atributo puede cambiar dinámicamente.

El valor puede ser uno de los siguientes:

MQEVR_DISABLED

Informes de sucesos inhabilitados.

MQEVR_ENABLED

Informes de sucesos habilitados.

Para obtener más información sobre los sucesos, consulte [Supervisión de sucesos](#).

Para determinar el valor de este atributo, utilice el selector MQIA_Q_DEPTH_MAX_EVENT con la llamada MQINQ.

Este atributo está soportado en z/OS, pero la llamada MQINQ no se puede utilizar para determinar su valor.

QDesc (MQCHAR64)

Utilice este campo para comentarios descriptivos.

Tabla 643. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X	X	X	X

El contenido del campo no es significativo para el gestor de colas, pero es posible que el gestor de colas requiera que el campo contenga sólo caracteres que se puedan visualizar. No puede contener ningún carácter nulo; si es necesario, se rellena a la derecha con espacios en blanco. En una instalación DBCS, el campo puede contener caracteres DBCS (sujetos a una longitud máxima de campo de 64 bytes).

Nota: Si este campo contiene caracteres que no están en el juego de caracteres del gestor de colas (tal como lo define el atributo de gestor de colas **CodedCharSetId**), estos caracteres podrían traducirse incorrectamente si este campo se envía a otro gestor de colas.

Para determinar el valor de este atributo, utilice el selector MQCA_Q_DESC con la llamada MQINQ. La longitud de este atributo la proporciona MQ_Q_DESC_LENGTH.

QName (MQCHAR48)

Es el nombre de una cola definida en el gestor de colas local.

Tabla 644. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X		X	X	X

Todas las colas definidas en un gestor de colas comparten el mismo espacio de nombres de cola. Por lo tanto, una cola MQQT_LOCAL y una cola MQQT_ALIAS no pueden tener el mismo nombre.

Para determinar el valor de este atributo, utilice el selector MQCA_Q_NAME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_Q_NAME_LENGTH.

QServiceInterval (MQLONG)

Este es el intervalo de servicio utilizado para la comparación para generar sucesos de intervalo de servicio alto y de intervalo de servicio correcto.

Tabla 645. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X			

Véase el atributo [QServiceIntervalEvent](#).

El valor está en unidades de milisegundos, y es mayor o igual que cero, y menor o igual que 999 999 999 999.

Para determinar el valor de este atributo, utilice el selector MQIA_Q_SERVICE_INTERVAL con la llamada MQINQ.

Este atributo está soportado en z/OS, pero la llamada MQINQ no se puede utilizar para determinar su valor.

QServiceIntervalEvent (MQLONG)

Esto controla si se generan sucesos de intervalo de servicio alto o de intervalo de servicio correcto.

Tabla 646. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X			

- Se genera un suceso de intervalo de servicio alto cuando una comprobación indica que no se ha recuperado ningún mensaje de la cola durante al menos el tiempo indicado por el atributo **QServiceInterval**.
- Se genera un suceso de intervalo de servicio correcto cuando una comprobación indica que los mensajes se han recuperado de la cola dentro del tiempo indicado por el atributo **QServiceInterval**.

Nota: El valor de este atributo puede cambiar dinámicamente.

El valor puede ser uno de los siguientes:

MQQSIE_HIGH

Sucesos de intervalo de servicio de cola alto habilitados.

- Los sucesos de intervalo de servicio de cola alto están **habilitados** y
- Los sucesos de intervalo de servicio de cola correcto están **inhabilitados**.

MQQSIE_OK

Sucesos de intervalo de servicio de cola correcto habilitados.

- Los sucesos de intervalo de servicio de cola alto están **inhabilitados** y
- Los sucesos de intervalo de servicio de cola correcto están **habilitados**.

MQQSIE_NONE

No hay sucesos de intervalo de servicio de cola habilitados.

- Los sucesos de intervalo de servicio de cola alto están **inhabilitados** y
- Los sucesos de intervalo de servicio de cola correcto también están **inhabilitados**.

Para las colas compartidas, el valor de este atributo se ignora; se asume el valor MQQSIE_NONE.

Para obtener más información sobre los sucesos, consulte [Supervisión de sucesos](#).

Para determinar el valor de este atributo, utilice el selector MQIA_Q_SERVICE_INTERVAL_EVENT con la llamada MQINQ.

En z/OS, no puede utilizar la llamada MQINQ para determinar el valor de este atributo.

QSGDisp (MQLONG)

Especifica la disposición de la cola.

Tabla 647. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X		X	X	

El valor puede ser uno de los siguientes:

MQQSGD_Q_MGR

El objeto tiene disposición de gestor de colas. Esto significa que la definición de objeto sólo es conocida por el gestor de colas local; la definición no es conocida por otros gestores de colas del grupo de compartición de colas.

Cada gestor de colas del grupo de compartición de colas puede tener un objeto con el mismo nombre y tipo que el objeto actual, pero estos son objetos separados y no hay ninguna correlación entre ellos. Sus atributos no están restringidos a ser iguales entre sí.

MQQSGD_COPY

El objeto es una copia local de una definición de objeto maestro que existe en el repositorio compartido. Cada gestor de colas del grupo de compartición de colas puede tener su propia copia del objeto. Inicialmente, todas las copias tienen los mismos atributos, pero utilizando mandatos MQSC, puede modificar cada copia para que sus atributos difieran de los de las otras copias. Los atributos de las copias se resincronizan cuando se modifica la definición maestra en el repositorio compartido.

MQQSGD_COMPARTIDO

El objeto tiene una disposición compartida. Esto significa que existe en el repositorio compartido una única instancia del objeto que es conocida por todos los gestores de colas del grupo de compartición de colas. Cuando un gestor de colas del grupo accede al objeto, accede a la única instancia compartida del objeto.

Para determinar el valor de este atributo, utilice el selector MQIA_QSG_DISP con la llamada MQINQ.

 Este atributo sólo está soportado en z/OS.

QueueAccounting (MQLONG)

Tabla 648. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X	X	X	

Esto controla la recopilación de datos de contabilidad para la cola. Para que los datos de contabilidad se recopilen para esta cola, los datos de contabilidad para esta conexión también deben estar habilitados, utilizando el atributo ACCTQ de QMGR o el campo Opciones de la estructura MQCNO en la llamada MQCONN.

Este atributo tiene uno de los siguientes valores:

MQMON_Q_MGR

Los datos de contabilidad para esta cola se recopilan basándose en el valor del atributo ACCTQ de QMGR. Ésta es el ajuste predeterminado.

MQMON_OFF

No recopile datos de contabilidad para esta cola.

MQMON_ON

Recopilar datos de contabilidad para esta cola.

Para determinar el valor de este atributo, utilice el selector MQIA_ACCOUNTING_Q con la llamada MQINQ.

QueueMonitoring (MQLONG)

Controla la recopilación de los datos de supervisión para las colas.

Tabla 649. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X			

El valor puede ser uno de los siguientes:

MQMON_Q_MGR

Recopile datos de supervisión según el valor del atributo de gestor de colas **QueueMonitoring**. Éste es el valor predeterminado.

MQMON_OFF

La recopilación de datos de supervisión en línea está desactivada para esta cola.

MQMON_LOW

Si el valor del atributo de gestor de colas **QueueMonitoring** no es MQMON_NONE, se activa la recopilación de datos de supervisión en línea, con una tasa baja de recopilación de datos para esta cola.

MQMON_MEDIO

Si el valor del atributo de gestor de colas **QueueMonitoring** no es MQMON_NONE, la recopilación de datos de supervisión en línea está activada, con una tasa moderada de recopilación de datos para esta cola.

MQMON_HIGH

Si el valor del atributo de gestor de colas **QueueMonitoring** no es MQMON_NONE, se activa la recopilación de datos de supervisión en línea, con una alta tasa de recopilación de datos para esta cola.

Para determinar el valor de este atributo, utilice el selector MQIA_MONITORING_Q con la llamada MQINQ.

QueueStatistics (MQCHAR12)

Tabla 650. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X	X	X	

Esto controla la recopilación de datos estadísticos para la cola.

Este atributo tiene uno de los siguientes valores:

MQMON_Q_MGR

Los datos de contabilidad para esta cola se recopilan basándose en el valor del atributo STATQ de QMGR. Ésta es el ajuste predeterminado.

MQMON_OFF

Desactive la recopilación de datos de estadísticas para esta cola.

MQMON_ON

Habilite la recopilación de datos de estadísticas para esta cola.

Tipo de cola (MQLONG)

Tabla 651. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X		X	X	X

Este es el tipo de cola; tiene uno de los valores siguientes:

MQQT_ALIAS

Definición de cola alias.

MQQT_CLUSTER

Cola de clúster.

MQQT_LOCAL

Cola local.

MQQT_REMOTE

Definición local de una cola remota.

Para determinar el valor de este atributo, utilice el selector MQIA_Q_TYPE con la llamada MQINQ.

RemoteQMgrName (MQCHAR48)

Tabla 652. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
			X	

Es el nombre del gestor de colas remoto en el que se define la cola **RemoteQName**. Si la cola **RemoteQName** tiene un valor **QSGDisp** de MQQSGD_COPY o MQQSGD_SHARED, **RemoteQMgrName** puede ser el nombre del grupo de compartición de colas propietario de **RemoteQName**.

Si una aplicación abre la definición local de una cola remota, **RemoteQMgrName** no debe estar en blanco y no debe ser el nombre del gestor de colas local. Si **XmitQName** está en blanco, se utiliza la cola local con el mismo nombre que **RemoteQMgrName** como cola de transmisión. Si no hay ninguna cola con el nombre **RemoteQMgrName**, se utiliza la cola identificada por el atributo de gestor de colas **DefXmitQName**.

Si esta definición se utiliza para un alias de gestor de colas, **RemoteQMgrName** es el nombre del gestor de colas que se está alias. Puede ser el nombre del gestor de colas local. De lo contrario, si **XmitQName** está en blanco cuando se produce la apertura, debe haber una cola local con un nombre que sea el mismo que **RemoteQMgrName**; esta cola se utiliza como cola de transmisión.

Si esta definición se utiliza para un alias de respuesta, este nombre es el nombre del gestor de colas que debe ser **ReplyToQMgr**.

Nota: No se realiza ninguna validación en el valor especificado para este atributo cuando se crea o modifica la definición de cola.

Para determinar el valor de este atributo, utilice el selector MQCA_REMOTE_Q_MGR_NAME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_Q_MGR_NAME_LENGTH.

RemoteQName (MQCHAR48)

Tabla 653. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
			X	

Es el nombre de la cola tal como se conoce en el gestor de colas remoto **RemoteQMgrName**.

Si una aplicación abre la definición local de una cola remota, cuando se produce la apertura, **RemoteQName** no debe estar en blanco.

Si esta definición se utiliza para una definición de alias de gestor de colas, cuando se produzca la apertura, *RemoteQName* debe estar en blanco.

Si la definición se utiliza para un alias de respuesta, este nombre es el nombre de la cola que va a ser *ReplyToQ*.

Nota: No se realiza ninguna validación en el valor especificado para este atributo cuando se crea o modifica la definición de cola.

Para determinar el valor de este atributo, utilice el selector MQCA_REMOTE_Q_NAME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_Q_NAME_LENGTH.

RetentionInterval (MQLONG)

Es el periodo de tiempo durante el cual se retiene la cola. Una vez transcurrido este tiempo, la cola es apta para su supresión.

Tabla 654. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X			

El tiempo se mide en horas, contando desde la fecha y hora en que se creó la cola. La fecha y hora de creación de la cola se registran en los atributos **CreationDate** y **CreationTime**.

Esta información se proporciona para permitir que una aplicación de mantenimiento o el operador identifique y suprima colas que ya no son necesarias.

Nota: El gestor de colas nunca realiza ninguna acción para suprimir colas basándose en este atributo, o para impedir la supresión de colas con un intervalo de retención que no ha caducado; es responsabilidad del usuario realizar cualquier acción necesaria.

Utilice un intervalo de retención realista para evitar la acumulación de colas dinámicas permanentes (consulte el atributo [DefinitionType](#)). Sin embargo, este atributo también se puede utilizar con colas predefinidas.

Para determinar el valor de este atributo, utilice el selector MQIA_RETENTION_INTERVAL con la llamada MQINQ.

Ámbito (MQLONG)

Esto controla si también existe una entrada para esta cola en un directorio de célula.

Tabla 655. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X		X	X	

Un servicio de nombres instalable proporciona un directorio de célula. El valor puede ser uno de los siguientes:

MQSCO_Q_MGR

La definición de cola tiene ámbito de gestor de colas: la definición de la cola no se extiende más allá del gestor de colas que la posee. Para abrir la cola para salida de algún otro gestor de colas, debe especificarse el nombre del gestor de colas propietario o el otro gestor de colas debe tener una definición local de la cola.

MQSCO_CELL

La definición de cola tiene ámbito de célula: la definición de cola también se coloca en un directorio de célula disponible para todos los gestores de colas de la célula. La cola se puede abrir para salida de cualquiera de los gestores de colas de la célula especificando el nombre de la cola; no es necesario especificar el nombre del gestor de colas propietario de la cola. Sin embargo, la definición de cola no está disponible para ningún gestor de colas de la célula que también tenga una definición local de una cola con ese nombre, ya que la definición local tiene prioridad.

Un servicio de nombres instalable proporciona un directorio de célula.

El modelo y las colas dinámicas no pueden tener ámbito de célula.

Este valor sólo es válido si se ha configurado un servicio de nombres que da soporte a un directorio de célula.

Para determinar el valor de este atributo, utilice el selector MQIA_SCOPE con la llamada MQINQ.

El soporte para este atributo está sujeto a las restricciones siguientes:

- En IBM i, el atributo está soportado, pero sólo MQSCO_Q_MGR es válido.
- En z/OS, el atributo no está soportado.

Compatibilidad (MQLONG)

Esto indica si la cola se puede abrir para entrada varias veces simultáneamente.

Tabla 656. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X			

El valor puede ser uno de los siguientes:

MQQA_SHAREABLE

La cola es compartible.

Se permiten varias aperturas con la opción MQOO_INPUT_SHARED.

MQQA_NOT_SHAREABLE

La cola no se puede compartir.

Una llamada MQOPEN con la opción MQOO_INPUT_SHARED se trata como MQOO_INPUT_EXCLUSIVE.

Para determinar el valor de este atributo, utilice el selector MQIA_SHAREABILITY con la llamada MQINQ.

StorageClass (MQCHAR8)

Es un nombre definido por el usuario que define el almacenamiento físico utilizado para contener la cola. En la práctica, un mensaje se graba en disco sólo si necesita paginarse fuera de su almacenamiento intermedio de memoria.

Tabla 657. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X			

Para determinar el valor de este atributo, utilice el selector MQCA_STORAGE_CLASS con la llamada MQINQ. La longitud de este atributo la proporciona MQ_STORAGE_CLASS_LENGTH.

 Este atributo sólo está soportado en z/OS.

TriggerControl (MQLONG)

Esto controla si los mensajes desencadenantes se graban en una cola de inicio para iniciar una aplicación para dar servicio a la cola.

Tabla 658. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X			

Es uno de los siguientes:

MQTC_OFF

No se deben escribir mensajes desencadenantes para esta cola. El valor de *TriggerType* es irrelevante en este caso.

MQTC_ON

Los mensajes desencadenantes se escribirán para esta cola cuando se produzcan los sucesos desencadenantes adecuados.

Para determinar el valor de este atributo, utilice el selector MQIA_TRIGGER_CONTROL con la llamada MQINQ. Para cambiar el valor de este atributo, utilice la llamada MQSET.

TriggerData (MQCHAR64)

Son datos de formato libre que el gestor de colas inserta en el mensaje desencadenante cuando un mensaje que llega a esta cola hace que se grave un mensaje desencadenante en la cola de inicio.

Tabla 659. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X			

El contenido de estos datos no es significativo para el gestor de colas. Es significativo para la aplicación de supervisor desencadenante que procesa la cola de inicio o para la aplicación que inicia el supervisor desencadenante.

La serie de caracteres no debe contener ningún valor nulo. Se rellena a la derecha con espacios en blanco si es necesario.

Para determinar el valor de este atributo, utilice el selector MQCA_TRIGGER_DATA con la llamada MQINQ. Para cambiar el valor de este atributo, utilice la llamada MQSET. La longitud de este atributo la proporciona MQ_TRIGGER_DATA_LENGTH.

TriggerDepth (MQLONG)

Tabla 660. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X			

Es el número de mensajes de prioridad *TriggerMsgPriority* o superior que deben estar en la cola antes de que se escriba un mensaje desencadenante. Esto se aplica cuando *TriggerType* se establece en MQTT_DEPTH. El valor de *TriggerDepth* es uno o mayor. De lo contrario, este atributo no se utiliza.

Para determinar el valor de este atributo, utilice el selector MQIA_TRIGGER_DEPTH con la llamada MQINQ. Para cambiar el valor de este atributo, utilice la llamada MQSET.

TriggerMsgPriority (MQLONG)

Esta es la prioridad de mensaje por debajo de la cual los mensajes no contribuyen a la generación de mensajes desencadenantes (es decir, el gestor de colas ignora estos mensajes al decidir si se genera un mensaje desencadenante).

Tabla 661. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X			

TriggerMsgPriority puede estar en el rango de cero (más bajo) a *MaxPriority* (más alto; consulte el atributo [MaxPriority](#)); un valor de cero hace que todos los mensajes contribuyan a la generación de mensajes desencadenantes.

Para determinar el valor de este atributo, utilice el selector MQIA_TRIGGER_MSG_PRIORITY con la llamada MQINQ. Para cambiar el valor de este atributo, utilice la llamada MQSET.

TriggerType (MQLONG)

Esto controla las condiciones en las que se graban los mensajes desencadenantes como resultado de los mensajes que llegan a esta cola.

Tabla 662. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X			

Tiene uno de los siguientes valores:

MQTT_NONE

No se graban mensajes desencadenantes como resultado de los mensajes de esta cola. Esto tiene el mismo efecto que establecer *TriggerControl* en MQTC_OFF.

MQTT_FIRST

Se graba un mensaje desencadenante siempre que el número de mensajes de prioridad *TriggerMsgPriority* o superior en la cola cambia de 0 a 1.

MQTT EVERY

Se graba un mensaje desencadenante siempre que llega a la cola un mensaje de prioridad *TriggerMsgPriority* o superior.

MQTT_DEPTH

Un mensaje desencadenante se graba siempre que el número de mensajes de prioridad *TriggerMsgPriority* o superior en la cola es igual o superior a *TriggerDepth*. Después de que se haya grabado el mensaje desencadenante, *TriggerControl* se establece en MQTC_OFF para evitar que se desencadene más hasta que se vuelva a activar explícitamente.

Para determinar el valor de este atributo, utilice el selector MQIA_TRIGGER_TYPE con la llamada MQINQ. Para cambiar el valor de este atributo, utilice la llamada MQSET.

Uso (MQLONG)

Indica para qué se utiliza la cola.

Tabla 663. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X			

El valor puede ser uno de los siguientes:

MQUS_NORMAL

Se trata de una cola que las aplicaciones utilizan al transferir y obtener mensajes; la cola no es una cola de transmisión.

MQUS_TRANSMISSION

Es una cola utilizada para contener mensajes destinados a gestores de colas remotos. Cuando una aplicación envía un mensaje a una cola remota, el gestor de colas local almacena el mensaje temporalmente en la cola de transmisión adecuada en un formato especial. A continuación, un agente de canal de mensajes lee el mensaje de la cola de transmisión y lo transporta al gestor de colas remoto. Para obtener más información sobre cómo configurar la administración remota, consulte [Configuración de gestores de colas para la administración remota](#).

Sólo las aplicaciones con privilegios pueden abrir una cola de transmisión para que MQOO_OUTPUT coloque mensajes directamente en ella. Normalmente, sólo las aplicaciones de programa de utilidad lo hacen. Asegúrese de que el formato de datos del mensaje sea correcto (consulte “MQXQH - Cabecera de cola de transmisión” en la página 618) o se pueden producir errores durante el proceso

de transmisión. El contexto no se pasa ni se establece a menos que se especifique una de las opciones de contexto MQPMO_*_CONTEXT.

Para determinar el valor de este atributo, utilice el selector MQIA_USAGE con la llamada MQINQ.

XmitQName (MQCHAR48)

Es el nombre de la cola de transmisión. Si este atributo no está en blanco cuando se produce una apertura, ya sea para una cola remota o para una definición de alias de gestor de colas, especifica el nombre de la cola de transmisión local que se utilizará para reenviar el mensaje.

Tabla 664. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
			X	

Si **XmitQName** está en blanco, la cola local con un nombre que es el mismo que **RemoteQMgrName** se utiliza como cola de transmisión. Si no hay ninguna cola con el nombre **RemoteQMgrName**, se utiliza la cola identificada por el atributo de gestor de colas **DefXmitQName**.

Este atributo se ignora si la definición se está utilizando como alias de gestor de colas y **RemoteQMgrName** es el nombre del gestor de colas local. Este atributo también se ignora si la definición se utiliza como definición de alias de cola de respuestas.

Para determinar el valor de este atributo, utilice el selector MQCA_XMIT_Q_NAME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_Q_NAME_LENGTH.

Atributos de las listas de nombres

En la tabla siguiente se resumen los atributos específicos de las listas de nombres. Los atributos se describen en orden alfabético.

Las listas de nombres están soportadas en todos los sistemas IBM MQ, más IBM MQ MQI clients conectados a estos sistemas.

Nota: Los nombres de los atributos que se muestran en esta sección son nombres descriptivos utilizados con las llamadas MQINQ y MQSET; los nombres son los mismos que para los mandatos PCF. Cuando se utilizan mandatos MQSC para definir, modificar o visualizar atributos, se utilizan nombres abreviados alternativos; consulte [Mandatos MQSC](#) para obtener más información.

Tabla 665. Atributos de las listas de nombres	
Atributo	Descripción
AlterationDate	Fecha en que se modificó por última vez la definición
AlterationTime	Hora a la que se modificó por última vez la definición
NameCount	Número de nombres en la lista de nombres
NamelistDesc	Descripción de lista de nombres
NamelistName	Nombre de lista de nombres
Nombres	Una lista de nombres de <i>NameCount</i>
NamelistType	Tipo de lista de nombres
QSGDisp	Disposición de grupo de uso compartido de colas

AlterationDate (MQCHAR12)

Es la fecha en la que se modificó por última vez la definición. El formato de la fecha es YYYY-MM-DD, relleno con dos espacios en blanco finales para que la longitud sea de 12 bytes.

Para determinar el valor de este atributo, utilice el selector MQCA_ALTERATION_DATE con la llamada MQINQ. La longitud de este atributo la proporciona MQ_DATE_LENGTH.

AlterationTime (MQCHAR8)

Es la hora en que se modificó por última vez la definición. El formato de la hora es HH.MM.SS.

Para determinar el valor de este atributo, utilice el selector MQCA_ALTERATION_TIME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_TIME_LENGTH.

NameCount (MQLONG)

Es el número de nombres de la lista de nombres. Es mayor o igual que cero. Se define el valor siguiente:

MQNC_MAX_NAMELIST_NAME_COUNT

Número máximo de nombres en una lista de nombres.

Para determinar el valor de este atributo, utilice el selector MQIA_NAME_COUNT con la llamada MQINQ.

NamelistDesc (MQCHAR64)

Utilice este campo para comentarios descriptivos; su valor se establece mediante el proceso de definición. El contenido del campo no es significativo para el gestor de colas, pero es posible que el gestor de colas requiera que el campo contenga sólo caracteres que se puedan visualizar. No puede contener ningún carácter nulo; si es necesario, se rellena a la derecha con espacios en blanco. En una instalación DBCS, este campo puede contener caracteres DBCS (sujetos a una longitud máxima de campo de 64 bytes).

Nota: Si este campo contiene caracteres que no están en el juego de caracteres del gestor de colas (tal como lo define el atributo de gestor de colas **CodedCharSetId**), estos caracteres podrían traducirse incorrectamente si este campo se envía a otro gestor de colas.

Para determinar el valor de este atributo, utilice el selector MQCA_NAMELIST_DESC con la llamada MQINQ.

La longitud de este atributo la proporciona MQ_NAMELIST_DESC_LENGTH.

NamelistName (MQCHAR48)

Es el nombre de una lista de nombres definida en el gestor de colas local. Para obtener más información sobre los nombres de lista de nombres, consulte la sección [Otros nombres de objeto](#).

Cada lista de nombres tiene un nombre que es diferente de los nombres de otras listas de nombres que pertenecen al gestor de colas, pero puede duplicar los nombres de otros objetos del gestor de colas de tipos diferentes (por ejemplo, colas).

Para determinar el valor de este atributo, utilice el selector MQCA_NAMELIST_NAME con la llamada MQINQ.

La longitud de este atributo la proporciona MQ_NAMELIST_NAME_LENGTH.

NamelistType (MQLONG)

Especifica la naturaleza de los nombres de la lista de nombres e indica cómo se utiliza la lista de nombres. Es uno de los valores siguientes:

MQNT_NINGUNO

Lista de nombres sin tipo asignado.

MQNT_Q

Lista de nombres que contiene los nombres de las colas.

MQNT_CLUSTER

Lista de nombres que contiene los nombres de los clústeres.

MQNT_AUTH_INFO

Lista de nombres que contiene los nombres de los objetos de información de autenticación.

Para determinar el valor de este atributo, utilice el selector MQIA_NAMELIST_TYPE con la llamada MQINQ.

 Este atributo sólo está soportado en z/OS.

Nombres (MQCHAR48xNameCount)

Esta es una lista de nombres de *NameCount*, donde cada nombre es el nombre de un objeto definido en el gestor de colas local. Para obtener más información sobre los nombres de objeto, consulte [Reglas para la denominación de objetos de IBM MQ](#).

Para determinar el valor de este atributo, utilice el selector MQCA_NAMES con la llamada MQINQ.

La longitud de cada nombre de la lista la proporciona MQ_OBJECT_NAME_LENGTH.

QSGDisp (MQLONG)

Especifica la disposición de la lista de nombres. El valor puede ser uno de los siguientes:

MQQSGD_Q_MGR

El objeto tiene disposición de gestor de colas: la definición de objeto sólo es conocida por el gestor de colas local; la definición no es conocida por otros gestores de colas del grupo de compartición de colas.

Cada gestor de colas del grupo de compartición de colas puede tener un objeto con el mismo nombre y tipo que el objeto actual, pero estos son objetos separados y no hay ninguna correlación entre ellos. Sus atributos no están restringidos a ser iguales entre sí.

MQQSGD_COPY

El objeto es una copia local de una definición de objeto maestro que existe en el repositorio compartido. Cada gestor de colas del grupo de compartición de colas puede tener su propia copia del objeto. Inicialmente, todas las copias tienen los mismos atributos, pero puede modificar cada copia, utilizando mandatos MQSC, para que sus atributos difieran de los de las otras copias. Los atributos de las copias se resincronizan cuando se modifica la definición maestra en el repositorio compartido.

Para determinar el valor de este atributo, utilice el selector MQIA_QSG_DISP con la llamada MQINQ.

 Este atributo sólo está soportado en z/OS.

Atributos de las definiciones de proceso

En la tabla siguiente se resumen los atributos que son específicos de las definiciones de proceso. Los atributos se describen en orden alfabético.

Nota: Los nombres de los atributos de esta sección son nombres descriptivos utilizados con las llamadas MQINQ y MQSET; los nombres son los mismos que para los mandatos PCF. Cuando se utilizan mandatos MQSC para definir, modificar o visualizar atributos, se utilizan nombres abreviados alternativos; consulte [Mandatos MQSC](#) para obtener más información.

Tabla 666. Atributos de las definiciones de proceso	
Atributo	Descripción
AlterationDate	Fecha en que se modificó por última vez la definición
AlterationTime	Hora a la que se modificó por última vez la definición
ApplId	Identificador de aplicación
ApplType	Tipo de aplicación
EnvData	Datos de entorno
ProcessDesc	Descripción del proceso
ProcessName	Nombre de proceso
QSGDisp	Disposición de grupo de uso compartido de colas
UserData	Datos de usuario

AlterationDate (MQCHAR12)

Es la fecha en la que se modificó por última vez la definición. El formato de la fecha es YYYY-MM-DD, rellenado con dos espacios en blanco finales para que la longitud sea de 12 bytes.

Para determinar el valor de este atributo, utilice el selector MQCA_ALTERATION_DATE con la llamada MQINQ. La longitud de este atributo la proporciona MQ_DATE_LENGTH.

AlterationTime (MQCHAR8)

Es la hora en que se modificó por última vez la definición. El formato de la hora es HH.MM.SS.

Para determinar el valor de este atributo, utilice el selector MQCA_ALTERATION_TIME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_TIME_LENGTH.

ApplId (MQCHAR256)

Es una serie de caracteres que identifica la aplicación que se va a iniciar. Esta información la utiliza una aplicación de supervisor desencadenante que procesa mensajes en la cola de inicio; la información se envía a la cola de inicio como parte del mensaje desencadenante.

El significado de *ApplId* lo determina la aplicación de supervisor desencadenante. El supervisor desencadenante proporcionado por IBM MQ requiere que *ApplId* sea el nombre de un programa ejecutable. Las notas siguientes se aplican a los entornos indicados:

- En z/OS, *ApplId* debe ser:
 - Un identificador de transacción CICS , para las aplicaciones iniciadas utilizando la transacción CKTI del supervisor desencadenante de CICS
 - Un identificador de transacción IMS , para las aplicaciones iniciadas utilizando el supervisor desencadenante de IMS CSQQTRMN
- En Windows, el nombre de programa puede tener como prefijo una unidad y una vía de acceso de directorio.
- En AIX and Linux, el nombre de programa puede tener como prefijo una vía de acceso de directorio.

La serie de caracteres no puede contener nulos. Se rellena a la derecha con espacios en blanco si es necesario.

Para determinar el valor de este atributo, utilice el selector MQCA_APPL_ID con la llamada MQINQ. La longitud de este atributo la proporciona MQ_PROCESS_APPL_ID_LENGTH.

ApplType (MQLONG)

Los tipos de aplicación identifican la naturaleza del programa que se iniciará en respuesta a la recepción de un mensaje desencadenante. Esta información es utilizada por una aplicación de monitorización de activadores que procesa mensajes en la cola de inicio; la información se envía a la cola de inicio como parte del mensaje de activación.

ApplType puede tener cualquier valor, pero se recomiendan los siguientes valores para los tipos estándar. Restringir los tipos de aplicaciones definidas por el usuario a valores comprendidos entre MQAT_USER_FIRST y MQAT_USER_LAST.

MQAT_CICS

CICS transacción.

MQAT_DEF

Tipo de aplicación predeterminado. Véase [DEFINIR PROCESO \(crear una nueva definición de proceso\)](#).

MQAT_OS400

IBM i .

MQAT_UNIX

UNIX .

MQAT_DESCONOCIDO

Aplicación de tipo desconocido.

MQAT_USUARIO

Aplicación de usuario.

MQAT_VENTANAS

Aplicación Windows de 64 bits.

MQAT_WINDOWS_NT

Aplicación Windows de 32 bits.

MQAT_ZOS

z/OS.

MQAT_USUARIO_PRIMERO

Valor más bajo para el tipo de aplicación definido por el usuario.

MQAT_USUARIO_ÚLTIMO

Valor más alto para el tipo de aplicación definido por el usuario.

Para determinar el valor de este atributo, utilice el selector MQIA_APPL_TYPE con la llamada MQINQ.

EnvData (MQCHAR128)

Es una serie de caracteres que contiene información relacionada con el entorno perteneciente a la aplicación que se va a iniciar. Esta información la utiliza una aplicación de supervisor desencadenante que procesa mensajes en la cola de inicio; la información se envía a la cola de inicio como parte del mensaje desencadenante.

El significado de *EnvData* lo determina la aplicación de supervisor desencadenante. El supervisor desencadenante proporcionado por IBM MQ añade *EnvData* a la lista de parámetros pasada a la aplicación iniciada. La lista de parámetros consta de la estructura MQTMC2, seguida de un espacio en blanco, seguida de *EnvData* con los espacios en blanco finales eliminados. Las notas siguientes se aplican a los entornos indicados:

- En z/OS:
 - *EnvData* no es utilizado por las aplicaciones de supervisor desencadenante proporcionadas por IBM MQ.
 - Si ApplType es MQAT_WLM, puede proporcionar valores predeterminados en *EnvData* para los campos ServiceName y ServiceStep en la cabecera de información de trabajo (MQWIH).
- En AIX and Linux, *EnvData* se puede establecer en el carácter & para ejecutar la aplicación iniciada en segundo plano.

La serie de caracteres no puede contener nulos. Se rellena a la derecha con espacios en blanco si es necesario.

Para determinar el valor de este atributo, utilice el selector MQCA_ENV_DATA con la llamada MQINQ. La longitud de este atributo la proporciona MQ_PROCESS_ENV_DATA_LENGTH.

ProcessDesc (MQCHAR64)

Utilice este campo para comentarios descriptivos. El contenido del campo no es significativo para el gestor de colas, pero es posible que el gestor de colas requiera que el campo contenga sólo caracteres que se puedan visualizar. No puede contener ningún carácter nulo; si es necesario, se rellena a la derecha con espacios en blanco. En una instalación DBCS, el campo puede contener caracteres DBCS (sujetos a una longitud máxima de campo de 64 bytes).

Nota: Si este campo contiene caracteres que no están en el juego de caracteres del gestor de colas (tal como lo define el atributo de gestor de colas **CodedCharSetId**), estos caracteres podrían traducirse incorrectamente si este campo se envía a otro gestor de colas.

Para determinar el valor de este atributo, utilice el selector MQCA_PROCESS_DESC con la llamada MQINQ.

La longitud de este atributo la proporciona MQ_PROCESS_DESC_LENGTH.

ProcessName (MQCHAR48)

Es el nombre de una definición de proceso definida en el gestor de colas local.

Cada definición de proceso tiene un nombre que es diferente de los nombres de otras definiciones de proceso que pertenecen al gestor de colas. Pero el nombre de la definición de proceso puede ser el mismo que los nombres de otros objetos de gestor de colas de tipos diferentes (por ejemplo, colas).

Para determinar el valor de este atributo, utilice el selector MQCA_PROCESS_NAME con la llamada MQINQ.

La longitud de este atributo la proporciona MQ_PROCESS_NAME_LENGTH.

QSGDisp (MQLONG)

Especifica la disposición de la definición de proceso. El valor puede ser uno de los siguientes:

MQQSGD_Q_MGR

El objeto tiene disposición de gestor de colas: la definición de objeto sólo es conocida por el gestor de colas local; la definición no es conocida por otros gestores de colas del grupo de compartición de colas.

Cada gestor de colas del grupo de compartición de colas puede tener un objeto con el mismo nombre y tipo que el objeto actual, pero estos son objetos separados y no hay ninguna correlación entre ellos. Sus atributos no están restringidos a ser iguales entre sí.

MQQSGD_COPY

El objeto es una copia local de una definición de objeto maestro que existe en el repositorio compartido. Cada gestor de colas del grupo de compartición de colas puede tener su propia copia del objeto. Inicialmente, todas las copias tienen los mismos atributos, pero puede modificar cada copia, utilizando mandatos MQSC, para que sus atributos difieran de los de las otras copias. Los atributos de las copias se resincronizan cuando se modifica la definición maestra en el repositorio compartido.

Para determinar el valor de este atributo, utilice el selector MQIA_QSG_DISP con la llamada MQINQ.

 Este atributo sólo está soportado en z/OS.

UserData (MQCHAR128)

UserData es una serie de caracteres que contiene información de usuario perteneciente a la aplicación que se va a iniciar. Esta información es para que la utilice una aplicación de supervisor desencadenante que procesa mensajes en la cola de inicio, o la aplicación iniciada por el supervisor desencadenante. La información se envía a la cola de inicio como parte del mensaje desencadenante.

El significado de *UserData* lo determina la aplicación de supervisor desencadenante. El supervisor desencadenante proporcionado por IBM MQ pasa *UserData* a la aplicación iniciada como parte de la lista de parámetros. La lista de parámetros consta de la estructura MQTMC2 (que contiene *UserData*), seguida de un espacio en blanco, seguida de *EnvData* con los espacios en blanco finales eliminados.

La serie de caracteres no puede contener nulos. Se rellena a la derecha con espacios en blanco si es necesario. Para Microsoft Windows, la serie de caracteres no debe contener comillas dobles si la definición de proceso se va a pasar a **runmqtrm**.

Para determinar el valor de este atributo, utilice el selector MQCA_USER_DATA con la llamada MQINQ. La longitud de este atributo la proporciona MQ_PROCESS_USER_DATA_LENGTH.

Códigos de retorno

Por cada IBM MQ llamada a la interfaz de colas de mensajes (MQI) y IBM MQ a la interfaz de administración (MQAI), el gestor de colas o una rutina de salida devuelven un código **de finalización** y un código **de motivo** para indicar si la llamada se ha realizado con éxito o ha fallado.

Las aplicaciones no deben depender de los errores que se comprueban en un orden específico, excepto cuando se indique específicamente. Si puede surgir más de un código de terminación o código de razón de una llamada, el error concreto notificado depende de la implementación.

Las aplicaciones que comprueban la finalización correcta después de una llamada de API de IBM MQ siempre deben comprobar el código de finalización. No asuma el valor del código de terminación, basándose en el valor del código de razón.

Códigos de finalización

El parámetro de código de terminación (*CompCode*) permite al interlocutor ver rápidamente si la llamada se ha completado correctamente, se ha completado parcialmente o ha fallado. A continuación se muestra una lista de códigos de terminación, con más detalles de los que se proporcionan en las descripciones de las llamadas:

MQCC_OK

La llamada se ha completado del todo; se han establecido todos los parámetros de salida. En este caso, el parámetro **Reason** tiene siempre el valor MQRC_NONE.

MQCC_WARNING

La llamada se ha completado parcialmente. Es posible que algunos parámetros de salida se hayan establecido además de los parámetros de salida *CompCode* y *Reason*. El parámetro **Reason** proporciona información adicional sobre la finalización parcial.

MQCC_FAILED

El proceso de la llamada no se ha completado. El estado del gestor de colas no se modifica, excepto cuando se indique específicamente. Se han establecido los parámetros de salida *CompCode* y *Reason*; otros parámetros no se modifican, excepto cuando se indique lo contrario.

La razón puede ser un error en el programa de aplicación, o puede ser el resultado de alguna situación externa al programa, por ejemplo, es posible que se haya revocado la autorización del usuario. El parámetro **Reason** proporciona información adicional sobre el error.

códigos de razón

El parámetro de código de razón (*Reason*) califica el parámetro de código de terminación (*CompCode*).

Si no hay que notificar ninguna razón especial, se devuelve MQRC_NONE. Una llamada que ha finalizado correctamente devuelve MQCC_OK y MQRC_NONE.

Si el código de terminación es MQCC_WARNING o MQCC_FAILED, el gestor de colas siempre informa de una razón calificadora; se proporcionan detalles en la descripción de cada llamada.

Cuando las rutinas de salida de usuario establecen códigos de terminación y razones, deben cumplir estas reglas. Además, cualquier valor de razón especial definido por las salidas de usuario debe ser menor que cero, para asegurarse de que no entra en conflicto con los valores definidos por el gestor de colas. Las salidas pueden establecer razones ya definidas por el gestor de colas, cuando proceda.

También aparecen códigos de razón en:

- El campo *Reason* de la estructura MQDLH.
- El campo *Feedback* de la estructura MQMD.

Para obtener descripciones completas de los códigos de razón, consulte [Mensajes y códigos de razón](#).

Normas para validar las opciones de « MQI »

Esta sección lista las situaciones que producen un código de razón MQRC_OPTIONS_ERROR a partir de una llamada MQOPEN, MQPUT, MQPUT1, MQGET, MQCLOSE o MQSUB.

Llamada MQOPEN

Para las opciones de la llamada MQOPEN:

- Se debe especificar al menos *uno* de los siguientes:
 - MQOO_BROWSE
 - MQOO_INPUT_EXCLUSIVE ¹

- MQOO_INPUT_SHARED ¹
 - MQOO_INPUT_AS_Q_DEF ¹
 - MQOO_INQUIRE
 - MQOO_OUTPUT
 - MQOO_SET
 - MQOO_BIND_ON_OPEN ²
 - MQOO_BIND_NOT_FIXED ²
 - MQOO_BIND_ON_GROUP ²
 - MQOO_BIND_AS_Q_DEF ²
- Solo se permite *uno* de los siguientes:
 - MQOO_READ_AHEAD
 - MQOO_NO_READ_AHEAD
 - MQOO_READ_AHEAD_AS_Q_DEF
1. Solo se permite *uno* de los siguientes:
 - MQOO_INPUT_EXCLUSIVE
 - MQOO_INPUT_SHARED
 - MQOO_INPUT_AS_Q_DEF
 2. Solo se permite *uno* de los siguientes:
 - MQOO_BIND_ON_OPEN
 - MQOO_BIND_NO_CORREGIDO
 - MQOO_BIND_ON_GROUP
 - MQOO_BIND_AS_Q_DEF

Nota: Las opciones listadas anteriormente son mutuamente excluyentes. Sin embargo, como el valor de MQOO_BIND_AS_Q_DEF es cero, si se especifica con cualquiera de las otras dos opciones de vinculación no se genera el código de razón MQRC_OPTIONS_ERROR. MQOO_BIND_AS_Q_DEF se proporciona para ayudar a la documentación del programa.

- Si se especifica MQOO_SAVE_ALL_CONTEXT, también debe especificarse una de las opciones MQOO_INPUT_ *.
- Si se especifica una de las opciones MQOO_SET_ * _CONTEXT o MQOO_PASS_ * _CONTEXT, también debe especificarse MQOO_OUTPUT.
- Si se especifica MQOO_CO_OP, también se debe especificar MQOO_BROWSE
- Si se especifica MQOO_NO_MULTICAST, también debe especificarse MQOO_OUTPUT.

llamada MQPUT

Para las opciones de colocación de mensajes:

- La combinación de MQPMO_SYNCPOINT y MQPMO_NO_SYNCPOINT no está permitida.
- Solo se permite *uno* de los siguientes:
 - MQPMO_DEFAULT_CONTEXT
 - MQPMO_NO_CONTEXT
 - MQPMO_PASS_ALL_CONTEXT
 - MQPMO_PASS_IDENTITY_CONTEXT
 - MQPMO_SET_ALL_CONTEXT
 - MQPMO_SET_IDENTITY_CONTEXT

- Solo se permite *uno* de los siguientes:
 - MQPMO_ASYNC_RESPONSE
 - MQPMO_SYNC_RESPONSE
 - MQPMO_RESPONSE_AS_TOPIC_DEF
 - MQPMO_RESPUESTA_COMO_Q_DEF
- MQPMO_ALTERNATE_USER_AUTHORITY no está permitido (sólo es válido en la llamada MQPUT1).

llamada MQPUT1

Para las opciones de transferencia de mensajes, las reglas son las mismas que para la llamada MQPUT, excepto para lo siguiente:

- Se permite MQPMO_ALTERNATE_USER_AUTHORITY.
- MQPMO_LOGICAL_ORDER no está permitido.

llamada MQGET

Para las opciones get-message:

- Solo se permite *uno* de los siguientes:
 - MQGMO_NO_SYNCPOINT
 - MQGMO_SYNCPOINT
 - MQGMO_SYNCPOINT_IF_PERSISTENT
- Solo se permite *uno* de los siguientes:
 - MQGMO_BROWSE_FIRST
 - MQGMO_BROWSE_MSG_UNDER_CURSOR
 - MQGMO_BROWSE_NEXT
 - MQGMO_MSG_UNDER_CURSOR
- MQGMO_SYNCPOINT no está permitido con ninguno de los siguientes elementos:
 - MQGMO_BROWSE_FIRST
 - MQGMO_BROWSE_MSG_UNDER_CURSOR
 - MQGMO_BROWSE_NEXT
 - MQGMO_LOCK
 - MQGMO_UNLOCK
- MQGMO_SYNCPOINT_IF_PERSISTENT no está permitido con ninguno de los siguientes:
 - MQGMO_BROWSE_FIRST
 - MQGMO_BROWSE_MSG_UNDER_CURSOR
 - MQGMO_BROWSE_NEXT
 - MQGMO_COMPLETE_MSG
 - MQGMO_UNLOCK
- MQGMO_MARK_SKIP_BACKOUT requiere que se especifique MQGMO_SYNCPOINT.
- La combinación de MQGMO_WAIT y MQGMO_SET_SIGNAL no está permitida.
- Si se especifica MQGMO_LOCK, también se debe especificar uno de los siguientes:
 - MQGMO_BROWSE_FIRST
 - MQGMO_BROWSE_MSG_UNDER_CURSOR
 - MQGMO_BROWSE_NEXT
- Si se especifica MQGMO_UNLOCK, sólo se permiten los valores siguientes:

- MQGMO_NO_SYNCPOINT
- MQGMO_NO_WAIT

Llamada MQCLOSE

Para las opciones de la llamada MQCLOSE:

- La combinación de MQCO_DELETE y MQCO_DELETE_PURGE no está permitida.
- Sólo se permite uno de los siguientes:
 - MQCO_KEEP_SUB
 - MQCO_REMOVE_SUB

Llamada MQSUB

Para las opciones de la llamada MQSUB:

- Se debe especificar al menos uno de los siguientes:
 - MQSO_ALTER
 - MQSO_RESUME
 - MQSO_CREATE
- Sólo se permite uno de los siguientes:
 - MQSO_DURABLE
 - MQSO_NO_DURADERO

Nota: Las opciones listadas anteriormente son mutuamente excluyentes. Sin embargo, como el valor de MQSO_NON_DURABLE es cero, si se especifica con MQSO_DURABLE no se genera el código de razón MQRC_OPTIONS_ERROR. Se proporciona MQSO_NON_DURABLE para ayudar a la documentación del programa.

- La combinación de MQSO_GROUP_SUB y MQSO_MANAGED no está permitida.
- MQSO_GROUP_SUB requiere que se especifique MQSO_SET_CORREL_ID.
- Sólo se permite uno de los siguientes:
 - MQSO_ANY_USERID
 - MQSO_FIXED_USERID
- MQSO_NEW_PUBLICATIONS_ONLY está permitido en combinación con:
 - MQSO_CREATE
 - MQSO_ALTER, si se ha establecido MQSO_NEW_PUBLICATIONS_ONLY en la suscripción original
- La combinación de MQSO_PUBLICATIONS_ON_REQUEST y SubLevel mayor que 1 no está permitida.
- Sólo se permite uno de los siguientes:
 - MQSO_CARÁCTER_COMODÍN
 - MQSO_WILDCARD_TOPIC
- MQSO_NO_MULTICAST requiere que se especifique MQSO_MANAGED.

Mensajes del mandato de publicación/suscripción en cola

Una aplicación puede utilizar mensajes de mandato MQRFH2 para controlar una aplicación de publicación/suscripción en cola.

Una aplicación que utiliza MQRFH2 para la publicación/suscripción puede enviar los siguientes mensajes de mandato al SYSTEM.BROKER.CONTROL.QUEUE:

- [“Mensaje de supresión de publicación” en la página 886](#)
- [“Mensaje de anulación de registro de suscriptor” en la página 887](#)
- [“Publicar mensaje” en la página 891](#)
- [“Mensaje Registrar suscriptor” en la página 893](#)
- [“Mensaje de solicitud de actualización” en la página 898](#)

Si está escribiendo aplicaciones de publicación/suscripción en cola, debe comprender estos mensajes, el mensaje de respuesta del gestor de colas y el descriptor de mensaje (MQMD); consulte la información siguiente:

- [“Mensaje de respuesta del gestor de colas” en la página 900](#)
- [“Valores de MQMD para publicaciones reenviadas por un gestor de colas” en la página 906](#)
- [“Valores de MQMD en mensajes de respuesta del gestor de colas” en la página 907](#)
- [“Códigos de razón de publicación/suscripción” en la página 902](#)

Los mandatos están contenidos en una carpeta psc en el campo **NameValueData** de la cabecera MQRFH2 . El mensaje que puede enviar un intermediario en respuesta a un mensaje de mandato está contenido en una carpeta psc1 .

Las descripciones de cada mandato listan las propiedades que puede contener una carpeta. A menos que se especifique lo contrario, las propiedades son opcionales y sólo pueden aparecer una vez.

Los nombres de las propiedades se muestran como <Command>.

Los valores deben estar en formato de serie, por ejemplo: Publish.

Una constante de tipo serie que representa el valor de una propiedad se muestra entre paréntesis, por ejemplo: (MQPSC_PUBLISH).

Las constantes de tipo serie se definen en el archivo de cabecera cmqpsc . h que se proporciona con el gestor de colas.

Mensaje de supresión de publicación

El mensaje del mandato **Delete Publication** se envía a un gestor de colas desde un publicador, o desde otro gestor de colas, para indicar al gestor de colas que suprima las publicaciones retenidas para los temas especificados.

Este mensaje se envía a una cola supervisada por la interfaz de publicación/suscripción en cola del gestor de colas.

La cola de entrada ha de ser la cola a la que se envió la publicación original.

Si tiene autorización para algunos, pero no para todos, los temas especificados en el mensaje de mandato **Delete Publication** , sólo se suprimirán esos temas. Un mensaje **Broker Response** indica qué temas no se suprimen.

De forma similar, si un mandato **Publish** contiene más de un tema, un mandato **Delete Publication** que coincide con algunos de estos temas, pero no todos, suprime sólo las publicaciones para los temas que se especifican en el mandato **Delete Publication** .

Consulte [“Valores de MQMD para publicaciones reenviadas por un gestor de colas” en la página 906](#) para obtener detalles de los parámetros del descriptor de mensaje (MQMD) que son necesarios al enviar un mensaje de mandato al gestor de colas.

Propiedades

Mandato (MQPSC_COMMAND)

El valor es DeletePub (MQPSC_DELETE_PUBCIONTION).

Esta propiedad ha de especificarse.

Tema > (MQPSC_TOPIC)

El valor es una serie de caracteres que contiene un tema para el cual han de suprimirse las publicaciones retenidas. Se pueden incluir caracteres comodín en la serie de caracteres para suprimir publicaciones sobre más de un tema.

Esta propiedad ha de especificarse; puede repetirse para tantos temas como se desee.

DelOpt (MQPSC_DELETE_OPTION)

La propiedad de opciones de supresión puede tener uno de los siguientes valores:

Local (MQPSC_LOCAL)

Todas las publicaciones retenidas para los temas especificados se suprimen en el gestor de colas local (es decir, el gestor de colas al que se envía este mensaje), tanto si se han publicado con la opción Local como si no.

Las publicaciones de otros gestores de colas no se ven afectadas.

None (MQPSC_NONE)

Todas las opciones toman sus valores predeterminados. Esto tiene el mismo efecto que especificar la propiedad DelOpt. Si se especifican otras opciones simultáneamente, se ignora None.

El valor predeterminado si se omite esta propiedad es que todas las publicaciones retenidas para los temas especificados se suprimen en todos los gestores de colas de la red, independientemente de si se han publicado con la opción Local .

Ejemplo

He aquí un ejemplo de NameValueData para un mensaje de comando '**Delete Publication**'. Lo utiliza la aplicación de ejemplo para suprimir, en el gestor de colas local, la publicación retenida que contiene la última puntuación en la coincidencia entre Team1 y Team2.

```
<psc>
  <Command>DeletePub</Command>
  <Topic>Sport/Soccer/State/LatestScore/Team1 Team2</Topic>
  <DelOpt>Local</DelOpt>
</psc>
```

Mensaje de anulación de registro de suscriptor

El mensaje del mandato **Deregister Subscriber** lo envía a un gestor de colas un suscriptor, u otra aplicación en nombre de un suscriptor, para indicar que ya no desea recibir mensajes que coincidan con los parámetros especificados.

Este mensaje se envía a SYSTEM.BROKER.CONTROL.QUEUE, la cola de control del gestor de colas. El usuario debe tener la autorización necesaria para colocar un mensaje en esta cola.

Consulte [Valores de MQMD para publicaciones reenviadas por un gestor de colas](#) para obtener detalles de los parámetros del descriptor de mensaje (MQMD) que son necesarios al enviar un mensaje de mandato al gestor de colas.

Se puede anular el registro de una suscripción individual especificando el tema correspondiente, el punto de suscripción y los valores de filtro de la suscripción original. Si alguno de los valores no se ha especificado (es decir, han tomado los valores predeterminados) en la suscripción original, se deben omitir cuando se anule el registro de la suscripción.

Todas las suscripciones para un suscriptor, o un grupo de suscriptores, se pueden anular del registro utilizando la opción DeregAll . Por ejemplo, si se especifica DeregAll , junto con un punto de suscripción (pero sin tema ni filtro), se anularán el registro de todas las suscripciones para el suscriptor en el punto de suscripción especificado, independientemente del tema y del filtro. Se permite cualquier combinación de tema, filtro y punto de suscripción; si se especifican los tres, sólo puede coincidir una suscripción y se ignora la opción DeregAll .

El mensaje debe enviarlo el suscriptor que ha registrado la suscripción; esto se confirma comprobando el ID de usuario del suscriptor.

Un administrador del sistema también puede anular el registro de las suscripciones utilizando los mandatos MQSC o PCF. Sin embargo, las suscripciones registradas con una cola dinámica temporal están asociadas con la cola, no sólo con el nombre de cola. Si la cola se suprime, ya sea de forma explícita, o mediante la desconexión de la aplicación del gestor de colas, ya no es posible utilizar el mandato **Deregister Subscriber** para anular el registro de las suscripciones para dicha cola. Las suscripciones se pueden anular del registro utilizando el entorno de trabajo del desarrollador y el gestor de colas las elimina automáticamente la próxima vez que coincida con una publicación de la suscripción o la próxima vez que se reinicie el gestor de colas. En circunstancias normales, las aplicaciones deben anular el registro de sus suscripciones antes de suprimir la cola o de desconectarse del gestor de colas.

Si un suscriptor envía un mensaje para anular el registro de una suscripción y recibe un mensaje de respuesta para decir que se ha procesado correctamente, es posible que algunas publicaciones sigan llegando a la cola de suscriptores si el gestor de colas las estaba procesando al mismo tiempo que se estaba anulando el registro de la suscripción. Si los mensajes no se eliminan de la cola, es posible que haya una acumulación de mensajes no procesados en la cola del suscriptor. Si la aplicación ejecuta un bucle que incluye una llamada MQGET con el CorrelId adecuado después de permanecer inactivo durante un tiempo, estos mensajes se borran de la cola.

De forma similar, si el suscriptor utiliza una cola dinámica permanente y anula el registro y cierra la cola con la opción `MQCO_DELETE_PURGE` en una llamada MQCLOSE, es posible que la cola no esté vacía. Si alguna publicación del gestor de colas todavía no se ha confirmado cuando se suprime la cola, la llamada MQCLOSE emite un código de retorno `MQRC_Q_NOT_EMPTY`. La aplicación puede evitar este problema durmiendo y volviendo a emitir la llamada MQCLOSE de vez en cuando.

Propiedades

Mandato (MQPSC_COMMAND)

El valor es `DeregSub` (`MQPSC_DEREGISTER_SUBSCRIBER`).

Esta propiedad ha de especificarse.

Tema (MQPSC_TOPIC)

El valor es una serie que contiene el tema que se va a anular el registro.

Esta propiedad, opcionalmente, se puede repetir si se van a anular el registro de varios temas. Puede omitirse si se especifica `DeregAll` en `<RegOpt>`.

Los temas especificados pueden ser un subconjunto de los que están registrados si el suscriptor desea retener suscripciones para otros temas. Se permiten caracteres comodín, pero una serie de tema que contiene caracteres comodín debe coincidir exactamente con la serie correspondiente que se ha especificado en el mensaje de mandato **Deregister Subscriber**.

SubPoint (MQPSC_SUBSCRIPTION_POINT)

El valor es una serie que especifica el punto de suscripción del que se va a desconectar la suscripción.

Esta propiedad no debe repetirse. Puede omitirse si se especifica un `<Topic>`, o si se especifica `DeregAll` en `<RegOpt>`. Si se omite esta propiedad, ocurre lo siguiente:

- Si **no** se especifica `DeregAll`, las suscripciones que coincidan con la propiedad `<Topic>` (y la propiedad `<Filter>`, si está presente) se dan de baja del punto de suscripción predeterminado.
- Si especifica `DeregAll`, todas las suscripciones (que coincidan con las propiedades `<Topic>` y `<Filter>` si están presentes) se dan de baja de todos los puntos de suscripción.

Tenga en cuenta que no puede especificar el punto de suscripción predeterminado de forma explícita. Por lo tanto, no hay forma de anular el registro de todas las suscripciones de este punto de suscripción solamente; debe especificar los temas.

SubIdentity (MQPSC_SUBSCRIPTION_IDENTITY)

Se trata de una serie de longitud variable con una longitud máxima de 64 caracteres. Se utiliza para representar una aplicación con un interés en una suscripción. El gestor de colas mantiene un conjunto

de identidades de suscriptor para cada suscripción. Cada suscripción puede permitir que su conjunto de identidades contenga una sola identidad o un número ilimitado de identidades.

Si `SubIdentity` está en el conjunto de identidades para la suscripción, se elimina del conjunto. Si el conjunto de identidades queda vacío como resultado de esto, la suscripción se elimina del gestor de colas, a menos que se especifique `LeaveOnly` como valor de la propiedad `RegOpt`. Si el conjunto de identidades sigue conteniendo otras identidades, la suscripción no se elimina del gestor de colas y el flujo de publicación no se interrumpe.

Si se especifica `SubIdentity`, pero la `SubIdentity` no está en el conjunto de identidades para la suscripción, el mandato **Deregister Subscriber** falla con el código de retorno `MQRCCF_SUB_IDENTITY_ERROR`.

Filtro (MQPSC_FILTER)

El valor es una serie que especifica el filtro que se va a anular el registro. Debe coincidir exactamente, incluyendo mayúsculas y minúsculas y cualquier espacio, con un filtro de suscripción que se haya registrado previamente.

Esta propiedad puede, opcionalmente, repetirse si se va a anular el registro de más de un filtro. Puede omitirse si se especifica un `<Topic>`, o si se especifica `DeregAll` en `<RegOpt>`.

Los filtros especificados pueden ser un subconjunto de los registrados si el suscriptor desea retener suscripciones para otros filtros.

RegOpt (MQPSC_REGISTRATION_OPTION)

La propiedad de opciones de registro puede tomar los valores siguientes:

DeregAll

(MQPSC_DEREGISTER_ALL)

Se anulará el registro de todas las suscripciones coincidentes registradas para este suscriptor.

Si especifica `DeregAll`:

- `<Tema>`, `<Subpunto>` y `<Filtro>` pueden omitirse.
- `< Topic>` y `< Filtro >` se pueden repetir, si es necesario.
- `<SubPoint>` no debe repetirse.

Si **no** especifica `DeregAll`:

- Se debe especificar `< Topic>` y se puede repetir si es necesario.
- `<SubPoint>` y `<Filtro>` pueden omitirse.
- `<SubPoint>` no debe repetirse.
- `< Filtro >` se puede repetir, si es necesario.

Si los temas y los filtros se repiten, se eliminarán todas las suscripciones que coincidan con todas las combinaciones de los dos. Por ejemplo, un mandato **Deregister Subscriber** que especifique tres temas y tres filtros intentará eliminar nueve suscripciones.

CorrelAsId

(MQPSC_CORREL_ID_AS_IDENTITY)

El `CorrelId` del descriptor de mensaje (MQMD), que no debe ser cero, se utiliza para identificar al suscriptor. Debe coincidir con el `CorrelId` utilizado en la suscripción original.

FullResp

(MQPSC_FULL_RESPONSE)

Cuando se especifica `FullResp`, todos los atributos de la suscripción se devuelven en el mensaje de respuesta, si el mandato no falla.

Cuando se especifica `FullResp`, no se permite `DeregAll` en el mandato **Deregister Subscriber**. Tampoco es posible especificar varios temas. El mandato falla con el código de retorno `MQRCCF_REG_OPTIONS_ERROR`, en ambos casos.

LeaveOnly

(MQPSC_LEAVE_ONLY)

Cuando se especifica con una SubIdentity que está en el conjunto de identidades para la suscripción, la SubIdentity se elimina del conjunto de identidades para la suscripción. La suscripción no se elimina del gestor de colas, aunque el conjunto de identidades resultante esté vacío. Si el valor SubIdentity no está en el conjunto de identidades, el mandato falla con el código de retorno *MQRCCF_SUB_IDENTITY_ERROR*.

Si se especifica LeaveOnly sin SubIdentity, el comando falla con el código de retorno *MQRCCF_REG_OPTIONS_ERROR*.

Si no se especifica LeaveOnly ni una SubIdentity, la suscripción se elimina independientemente del contenido del conjunto de identidades para la suscripción.

Ninguno

(MQPSC_NONE)

Todas las opciones toman sus valores predeterminados. Esto tiene el mismo efecto que omitir la propiedad de opciones de registro. Si se especifican otras opciones simultáneamente, se ignora None.

VariableUserId

(MQPSC_VARIABLE_USER_ID)

Cuando se especifica, la identidad del suscriptor (cola, gestor de colas y correlid) no está restringida a un único ID de usuario. Esto difiere del comportamiento existente del gestor de colas que asocia el ID de usuario del mensaje de registro original con la identidad del suscriptor y, a partir de entonces, impide que cualquier otro usuario utilice dicha identidad. Si un nuevo suscriptor intenta utilizar la misma identidad, se devuelve el código de retorno *MQRCCF_DUPLICATE_SUBSCRIPTION*.

Cualquier usuario puede modificar o anular el registro de la suscripción cuando tenga la autorización adecuada, evitando la comprobación existente de que el ID de usuario debe coincidir con el del suscriptor original.

Para añadir esta opción a una suscripción existente, el mandato debe proceder del mismo ID de usuario que la propia suscripción original.

Si la suscripción que se va a dar de baja tiene definida VariableUserId, ésta debe definirse en el momento de dar de baja la suscripción para indicar qué suscripción se está dando de baja. De lo contrario, el ID de usuario del mandato **Deregister Subscriber** se utiliza para identificar la suscripción. Esto se altera temporalmente, junto con los otros identificadores de suscriptor, si se proporciona un nombre de suscripción.

El valor predeterminado, si se omite esta propiedad, es que no se establecen opciones de registro.

QMgrName (MQPSC_Q_MGR_NAME)

El valor es el nombre del gestor de colas para la cola de suscriptor. Debe coincidir con el QMgrName utilizado en la suscripción original.

Si se omite esta propiedad, el valor predeterminado es el nombre del gestor de la cola de respuestas (ReplyToQMgr) que hay en el descriptor del mensaje (MQMD). Si el nombre resultante está en blanco, el valor predeterminado es el nombre del gestor de colas.

QName (MQPSC_Q_NAME)

El valor es el nombre de la cola de suscriptores. Debe coincidir con el QName utilizado en la suscripción original.

Si se omite esta propiedad, el valor por defecto es el nombre de ReplyToQ en el descriptor del mensaje (MQMD), que no debe estar en blanco.

SubName (NOMBRE_SUSCRIPCIÓN_MQPSC)

Si especifica SubName en un comando '**Deregister Subscriber**', el valor de SubName tiene prioridad sobre todos los demás campos de identificador excepto el userid, a menos que VariableUserId se establezca en la propia suscripción. Si no se establece VariableUserId,

el comando '**Deregister Subscriber**' sólo tiene éxito si el userid del mensaje de comando coincide con el de la suscripción; de lo contrario, el comando falla con el código de retorno *MQRCCF_DUPLICATE_IDENTITY*.

Si existe una suscripción que coincide con la identidad tradicional de este mandato pero no tiene ningún SubName, el mandato **Deregister Subscriber** falla con el código de retorno *MQRCCF_SUB_NAME_ERROR*. Si se intenta anular el registro de una suscripción que tiene un SubName utilizando un mensaje de mandato que coincide con la identidad tradicional pero sin ningún SubName especificado, el mandato se ejecuta correctamente.

SubUserData(MQPSC_SUBSCRIPTION_USER_DATA)

Es una serie de texto de longitud variable. El valor lo almacena el gestor de colas con la suscripción, pero no influye en la entrega de la publicación al suscriptor. El valor se puede modificar volviendo a registrar en la misma suscripción con un nuevo valor. Este atributo es para el uso de la aplicación.

se devuelve 'SubUserData en la información metatópica (MQCACF_REG_SUB_USER_DATA) para una suscripción, si 'SubUserData' está presente.

Ejemplo

He aquí un ejemplo de NameValueData para un mensaje de comando '**Deregister Subscriber**'. En este ejemplo, la aplicación de ejemplo anula el registro de su suscripción a los temas que contienen la última puntuación para todas las coincidencias. La identidad del abonado, incluido el `CorrelId`, se toma de los valores predeterminados en el MQMD.

```
<psc>
  <Command>DeregSub</Command>
  <RegOpt>CorrelAsId</RegOpt>
  <Topic>Sport/Soccer/State/LatestScore/#</Topic>
</psc>
```

Publicar mensaje

El mensaje del mandato **Publish** se coloca en una cola, o desde un gestor de colas a un suscriptor, para publicar información sobre un tema o temas especificados.

Es necesaria la autorización para colocar un mensaje en una cola y la autorización para publicar información sobre un tema o temas especificados.

Si el usuario tiene autorización para publicar información sobre algunos temas, pero no todos, sólo se utilizan estos temas para publicar; una respuesta de aviso indica qué temas no se utilizan para publicar.

Si un suscriptor tiene suscripciones coincidentes, el gestor de colas reenvía el mensaje **Publish** a las colas de suscriptor definidas en los mensajes de mandato **Register Subscriber** correspondientes.

Consulte [Mensaje de respuesta del gestor de colas](#) para obtener detalles de los parámetros del descriptor de mensaje (MQMD) necesarios al enviar un mensaje de mandato al gestor de colas y utilizados cuando un gestor de colas reenvía una publicación a un suscriptor.

El gestor de colas reenvía el mensaje **Publish** a otros gestores de colas de la red que tienen suscripciones coincidentes, a menos que sea una publicación local.

Los datos de la publicación, si los hay, se incluyen en el cuerpo del mensaje. Los datos se pueden describir en una carpeta <mcd>, en el campo NameValueData de la cabecera MQRFH2.

Propiedades

Mandato (MQPSC_COMMAND)

El valor es Publish (*MQPSC_PUBLISH*).

Esta propiedad ha de especificarse.

Tema (MQPSC_TOPIC)

El valor es una serie de caracteres que contiene un tema que clasifica esta publicación. No se permiten caracteres comodín.

Debe añadir el tema a la lista de nombres `SYSTEM.QPUBSUB.QUEUE.NAMELIST`, consulte [Adición de una corriente](#) para obtener instrucciones sobre cómo completar esta tarea.

Esta propiedad ha de especificarse y puede repetirse, opcionalmente, para tantos temas como se desee.

SubPoint (MQPSC_SUBSCRIPTION_POINT)

Es el punto de suscripción en el que se publica la publicación.

En WebSphere Event Broker 6.0, el valor de la propiedad <SubPoint> es el valor del atributo Subscription Point del nodo Publication que está gestionando la publicación.

En IBM WebSphere MQ 7.0.1, el valor de la propiedad <SubPoint> debe coincidir con el nombre de un punto de suscripción. Consulte [Adición de un punto de suscripción](#).

PubOpt (MQPSC_PUBCRIPTION_OPTION)

La propiedad de opciones de publicación puede tener uno de los siguientes valores:

RetainPub

(MQPSC_RETAIN_PUB)

El gestor de colas debe retener una copia de la publicación. Si esta opción no está establecida, la publicación se suprime tan pronto como el gestor de colas ha enviado la publicación a todos sus suscriptores actuales.

IsRetainedPub

(MQPSC_IS_RETAINED_PUB)

(Sólo puede ser establecido por un gestor de colas.) Esta publicación ha sido retenida por el gestor de colas. El gestor de colas establece esta opción para notificar a un suscriptor que esta publicación se publicó anteriormente y se ha conservado, siempre que la suscripción se haya registrado con la opción " InformIfRetained. Se establece únicamente en respuesta a un mensaje de mandato de registro de suscriptor o de petición de actualización. Las publicaciones retenidas que se envían directamente a suscriptores no tiene establecida esta opción.

Local

(MQPSC_LOCAL)

Esta opción indica al gestor de colas que esta publicación no debe enviarse a otros gestores de colas. Todos los suscriptores registrados en este gestor de colas reciben esta publicación si tienen suscripciones coincidentes.

OtherSubsOnly

(MQPSC_OTHER_SUBS_ONLY)

Esta opción permite el proceso más sencillo de aplicaciones de tipo conferencia, en las que un publicador es también suscriptor del mismo tema. Indica al gestor de colas que no envíe la publicación a la cola de suscriptores del publicador aunque tenga una suscripción coincidente. La cola de suscriptores del editor consta de su QMgrName, QName y CorrelId opcional, tal y como se describe en la siguiente lista.

CorrelAsId

(MQPSC_CORREL_ID_AS_IDENTITY)

El CorrelId del MQMD (que no ha de ser cero) forma parte de la cola de suscriptores del publicador en aplicaciones donde el publicador es también un suscriptor.

Ninguno

(MQPSC_NONE)

Todas las opciones toman sus valores predeterminados. Esto surte el mismo efecto que omitir la propiedad de opciones de publicación. Si se especifican otras opciones simultáneamente, se ignora None.

Para tener varias opciones de publicación, solo es necesario introducir elementos <PubOpt> adicionales.

El valor predeterminado, si se omite esta propiedad, es que no se establece ninguna opción de publicación.

PubTime (MQPSC_PUBLISH_TIMESTAMP)

El valor es una indicación de la hora de publicación, opcional, establecida por el publicador. Tiene 16 caracteres de largo con el formato:

```
YYYYMMDDHHMSSSTH
```

y utiliza la Hora Universal. El gestor de colas no comprueba esta información antes de enviarla a los suscriptores.

SeqNum (MQPSC_NÚMERO_SECUENCIA)

El valor es un número de secuencia opcional establecido por el publicador.

Debe incrementarse en 1 con cada publicación. Sin embargo, esto no lo comprueba el gestor de colas, que simplemente transmite esta información a los suscriptores.

Si las publicaciones sobre el mismo tema se publican en distintos gestores de colas interconectados, es responsabilidad de los publicadores asegurarse de que los números de secuencia, si se utilizan, sean significativos.

QMgrName (MQPSC_Q_MGR_NAME)

El valor es una cadena que contiene el nombre del gestor de colas para la cola de suscriptores del editor, en aplicaciones en las que el editor es también un suscriptor (véase `OtherSubsOnly`).

Si se omite esta propiedad, el valor predeterminado es el nombre del gestor de la cola de respuestas (`ReplyToQMGr`) que hay en el descriptor del mensaje (MQMD). Si el nombre resultante está en blanco, el valor predeterminado es el nombre del gestor de colas.

QName (MQPSC_Q_NAME)

El valor es una cadena que contiene el nombre de la cola de suscriptores del editor, en aplicaciones en las que el editor es también un suscriptor (véase `OtherSubsOnly`).

Si se omite esta propiedad, el valor predeterminado es el nombre de la cola de respuestas (`ReplyToQ`) en el descriptor del mensaje (MQMD), que no ha de estar en blanco si se ha establecido `OtherSubsOnly`.

Ejemplo

Estos son algunos ejemplos de *NameValueData* para un mensaje de comando '**Publish**'.

El primer ejemplo es para una publicación enviada por el simulador de coincidencias en la aplicación de ejemplo para indicar que se ha iniciado una coincidencia.

```
<psc>
  <Command>Publish</Command>
  <Topic>Sport/Soccer/Event/MatchStarted</Topic>
</psc>
```

El segundo ejemplo es para una publicación retenida. Se publica la última puntuación entre Team1 y Team2.

```
<psc>
  <Command>Publish</Command>
  <PubOpt>RetainPub</PubOpt>
  <Topic>Sport/Soccer/State/LatestScore/Team1 Team2</Topic>
</psc>
```

Mensaje Registrar suscriptor

El mensaje del mandato **Register Subscriber** lo envía a un gestor de colas un suscriptor, o otra aplicación en nombre de un suscriptor, para indicar que desea suscribirse a uno o más temas en un punto de suscripción. También se puede especificar un filtro de contenido de mensaje.

En las expresiones de filtro de publicación/suscripción, el anidamiento de paréntesis hace que el rendimiento disminuya exponencialmente. Evite anidar paréntesis a una profundidad mayor que aproximadamente 6.

El mensaje se envía a SYSTEM.BROKER.CONTROL.QUEUE, que es la cola de control del gestor de colas. Se necesita autorización para transferir un mensaje a esta cola, además de la autorización de acceso (establecida por el administrador del sistema del gestor de colas) para el tema o temas de la suscripción.

Si el usuario tiene autorización sobre algunos temas, pero no todos, sólo se registran los que tienen autorización; una respuesta de aviso indica los que no están registrados.

Consulte [“Valores de MQMD en los mensajes de mandato para el gestor de colas”](#) en la página 905 para obtener detalles de los parámetros del descriptor de mensaje (MQMD) que son necesarios al enviar un mensaje de mandato al gestor de colas.

Si la respuesta a la cola es una cola dinámica temporal, el gestor de colas anula automáticamente el registro de la suscripción cuando se cierra la cola.

Propiedades

Mandato (MQPSC_COMMAND)

El valor es RegSub (MQPSC_REGISTER_SUBSCRIBER). Esta propiedad ha de especificarse.

Tema (MQPSC_TOPIC)

El tema para el que el suscriptor desea recibir publicaciones. Los caracteres comodín se pueden especificar como parte del tema.

Si utiliza el mandato MQSC **display sub** para examinar la suscripción creada de esta forma, el valor de la etiqueta < Topic> se muestra como la propiedad TOPICSTR de la suscripción.

Esta propiedad es necesaria y, opcionalmente, se puede repetir para tantos temas como sea necesario.

SubPoint (MQPSC_SUBSCRIPTION_POINT)

El valor es el punto de suscripción al que se adjunta la suscripción.

Si se omite esta propiedad, se utiliza el punto de suscripción predeterminado.

En WebSphere Event Broker 6.0, el valor de la propiedad <SubPoint> debe coincidir con el valor del atributo Subscription Point de los nodos de Publicación a los que están suscritos.

En IBM WebSphere MQ 7.0.1, el valor de la propiedad <SubPoint> debe coincidir con el nombre de un punto de suscripción. Consulte [Adición de un punto de suscripción](#).

Filtro (MQPSC_FILTER)

El valor es una expresión SQL que se utiliza como filtro en el contenido de los mensajes de publicación. Si una publicación en el tema especificado coincide con el filtro, se envía al suscriptor. Esta propiedad corresponde a la serie de selección que se utiliza en las llamadas MQSUB y MQOPEN. Para obtener más información, consulte [Selección del contenido de un mensaje](#)

Si se omite esta propiedad, no se realiza ningún filtrado de contenido.

RegOpt (MQPSC_REGISTRATION_OPTION)

Esta propiedad Opciones de registro puede tomar los valores siguientes:

AddName

(MQPSC_ADD_NAME)

Cuando se especifica para una suscripción existente que coincide con la identidad tradicional de este mandato Registrar suscripción, pero sin ningún valor SubName actual, el SubName especificado en este mandato se añade a la suscripción.

Si se especifica AddName, el campo SubName es obligatorio; de lo contrario, se devuelve MQRCCF_REG_OPTIONS_ERROR.

CorrelAsId

(MQPSC_CORREL_ID_AS_IDENTITY)

El CorrelId del descriptor de mensaje (MQMD) se utiliza al enviar publicaciones coincidentes a la cola de suscriptores. El CorrelId no debe ser cero,

FullResp

(MQPSC_FULL_RESPONSE)

Cuando se especifica, todos los atributos de la suscripción se devuelven en el mensaje de respuesta, si el mandato no falla.

FullResp sólo es válido cuando el mensaje de mandato hace referencia a una única suscripción. Por lo tanto, sólo se permite un tema en el mandato; de lo contrario, el mandato falla con el código de retorno *MQRCCF_REG_OPTIONS_ERROR*.

InformIfRet

(MQPSC_INFORM_IF_RETAINED)

El gestor de colas informa al suscriptor si una publicación se retiene cuando envía un mensaje de publicación en respuesta a un mensaje de mandato **Register Subscriber** o **Request Update**. El gestor de colas lo hace incluyendo la opción de publicación *IsRetainedPub* en el mensaje.

JoinExcl

(MQPSC_JOIN_EXCLUSIVE)

Esta opción indica que la *SubIdentity* especificada se debe añadir como miembro exclusivo del conjunto de identidades para la suscripción y que no se pueden añadir otras identidades al conjunto.

Si la identidad ya se ha unido a 'shared' y es la única entrada del conjunto, el conjunto se cambia a un bloqueo exclusivo mantenido por esta identidad. De lo contrario, si la suscripción tiene actualmente otras identidades en el conjunto de identidades (con acceso compartido), el mandato falla con el código de retorno *MQRCCF_SUBSCRIPTION_IN_USE*.

JoinShared

(MQPSC_JOIN_SHARED)

Esta opción indica que la *SubIdentity* especificada debe añadirse al conjunto de identidades para la suscripción.

Si la suscripción está bloqueada actualmente de forma exclusiva (utilizando la opción *JoinExcl*), el mandato falla con el código de retorno *MQRCCF_SUBSCRIPTION_LOCKED*, a menos que la identidad que tiene la suscripción bloqueada sea la misma que la de este mensaje de mandato. En este caso, el bloqueo se modifica automáticamente a un bloqueo compartido.

Local

(MQPSC_LOCAL)

La suscripción es local y no se distribuye a otros gestores de colas de la red. Las publicaciones realizadas en otros gestores de colas no se entregan a este suscriptor, a menos que también tenga una suscripción global correspondiente.

NewPubsOnly

(MQPSC_NEW_PUBS_ONLY)

Las publicaciones retenidas que existen en el momento en que se registra la suscripción no se envían al suscriptor; sólo se envían nuevas publicaciones.

Si un suscriptor se vuelve a registrar y cambia esta opción para que ya no se establezca, es posible que se vuelva a enviar una publicación que ya se le ha enviado.

NoAlter

(MQPSC_NO_ALTER)

Los atributos de una suscripción coincidente existente no se cambian.

Cuando se está creando una suscripción, esta opción se ignora. Todas las demás opciones especificadas se aplican a la nueva suscripción.

Si una SubIdentity también tiene una de las opciones de unión (JoinExcl o JoinShared), la identidad se añade al conjunto de identidades independientemente de si se especifica NoAlter .

Ninguno

(MQPSC_NONE)

Todas las opciones de registro toman sus valores predeterminados.

Si el suscriptor ya está registrado, sus opciones se restablecen a sus valores predeterminados (tenga en cuenta que esto no tiene el mismo efecto que omitir la propiedad de opciones de registro) y la caducidad de la suscripción se actualiza desde el MQMD del mensaje **Register Subscriber** .

Si se especifican otras opciones de registro al mismo tiempo, se ignora Ninguna .

NonPers

(MQPSC_NON_PERSISTENT)

Las publicaciones que coinciden con esta suscripción se entregan al suscriptor como mensajes no persistentes.

Pers

(MQPSC_PERSISTENT)

Las publicaciones que coinciden con esta suscripción se entregan al suscriptor como mensajes persistentes.

PersAsPub

(MQPSC_PERSISTENT_AS_PUBLISH)

Las publicaciones que coinciden con esta suscripción se entregan al suscriptor con la persistencia especificada por el publicador. Éste es el comportamiento predeterminado.

PersAsQueue

(MQPSC_PERSISTENT_AS_Q)

Las publicaciones que coinciden con esta suscripción se entregan al suscriptor con la persistencia especificada en la cola de suscriptor.

PubOnReqOnly

(MQPSC_PUB_ON_REQUEST_ONLY)

El gestor de colas no envía publicaciones al suscriptor, excepto en respuesta a un mensaje de mandato **Request Update** .

VariableUserId

(MQPSC_VARIABLE_USER_ID)

Cuando se especifica, la identidad del suscriptor (cola, gestor de colas y correlid) no está restringida a un único ID de usuario. Esto difiere del comportamiento existente del gestor de colas que asocia el ID de usuario del mensaje de registro original con la identidad del suscriptor y, a partir de entonces, impide que cualquier otro usuario utilice dicha identidad. Si un nuevo suscriptor intenta utilizar la misma identidad, se devuelve *MQRCCF_DUPLICATE_SUBSCRIPTION* .

Esto permite a cualquier usuario modificar o anular el registro de la suscripción si el usuario tiene la autorización adecuada. Por lo tanto, no es necesario comprobar que el ID de usuario coincide con el del suscriptor original.

Para añadir esta opción a una suscripción existente, el mandato debe proceder del mismo ID de usuario que la propia suscripción original.

Si la suscripción del comando ' **Request Update** ' tiene establecido VariableUserId, éste debe establecerse en el momento de actualizar la petición para indicar a qué suscripción se hace referencia. De lo contrario, el ID de usuario del mandato **Request Update** se utiliza para identificar la suscripción. Esto se altera temporalmente, junto con los otros identificadores de suscriptor, si se proporciona un nombre de suscripción.

Si un mensaje de mandato **Register Subscriber** sin este conjunto de opciones hace referencia a una suscripción existente que tiene este conjunto de opciones, la opción se elimina de esta suscripción y el ID de usuario de la suscripción se arregla ahora. Si ya existe un suscriptor que tiene la misma identidad (cola, gestor de colas e identificador de correlación) pero con un ID de usuario diferente asociado, el mandato falla con el código de retorno *MQRCCF_DUPLICATE_IDENTITY* porque sólo puede haber un ID de usuario asociado a una identidad de suscriptor.

Si se omite la propiedad de opciones de registro y el suscriptor ya está registrado, sus opciones de registro no se modifican y la caducidad de la suscripción se actualiza desde el MQMD del mensaje **Register Subscriber**.

Si el suscriptor todavía no está registrado, se crea una nueva suscripción con todas las opciones de registro que toman sus valores predeterminados.

Los valores por defecto son PersAsPub y ninguna otra opción establecida.

QMgrName (MQPSC_Q_MGR_NAME)

El valor es el nombre del gestor de colas para la cola de suscriptor, a la que el gestor de colas envía las publicaciones coincidentes.

Si se omite esta propiedad, el valor predeterminado es el nombre del gestor de la cola de respuestas (ReplyToQMgr) que hay en el descriptor del mensaje (MQMD). Si el nombre resultante está en blanco, el valor predeterminado es QMgrNamedel gestor de colas.

QName (MQPSC_Q_NAME)

El valor es el nombre de la cola de suscriptores, a la que el gestor de colas envía las publicaciones coincidentes.

Si se omite esta propiedad, el valor por defecto es el nombre ReplyToQ en el descriptor del mensaje (MQMD), que no debe estar en blanco en este caso.

Si la cola es una cola dinámica temporal, se debe especificar la entrega no persistente de publicaciones (NonPers) en la propiedad <RegOpt>.

Si la cola es una cola dinámica temporal, el gestor de colas anula automáticamente el registro de la suscripción cuando se cierra la cola.

SubName (NOMBRE_SUSCRIPCIÓN_MQPSC)

Se trata de un nombre asignado a una suscripción determinada. Puede utilizarlo en lugar del gestor de colas, cola y correlId opcional para hacer referencia a una suscripción.

Si ya existe una suscripción con este 'SubName', cualquier otro atributo de la suscripción (Topic, 'QMgrName, QName, 'CorrelId, 'UserId, 'RegOpts, 'UserSubData, 'y Expiry) se anula con los atributos, si se especifican, que se pasan en el nuevo mensaje de comando 'Registro de abonados'. Sin embargo, si se utiliza **SubName** sin especificar el campo QName y se especifica un ReplyToQ en la cabecera MQMD, la cola de suscriptores cambia para ser el ReplyToQ.

Si ya existe una suscripción que coincide con la identidad tradicional de este mandato, pero no tiene ningún **SubName**, el mandato Registration falla con el código de retorno *MQRCCF_DUPLICATE_SUBSCRIPTION*, a menos que se especifique la opción **AddName**.

Si intenta modificar una suscripción con nombre existente utilizando otro comando 'Registro de abonados' que especifica el mismo 'SubName', y los valores de Tema, 'QMgrName, QName, y 'CorrelId en el nuevo comando coinciden con una suscripción existente diferente, con o sin un 'SubName' definido, el comando falla con el código de retorno 'MQRCCF_DUPPLICAR_SUSCRIPCIÓN'. Esto impide que dos nombres de suscripción hagan referencia a la misma suscripción.

SubIdentity (MQPSC_SUBSCRIPTION_IDENTITY)

Esta serie se utiliza para representar una aplicación con un interés en una suscripción. Es una serie de caracteres de longitud variable con una longitud máxima de 64 caracteres y es opcional. El gestor de colas mantiene un conjunto de identidades de suscriptor para cada suscripción. Cada suscripción puede permitir que su conjunto de identidades contenga sólo una identidad o un número ilimitado de identidades (consulte las opciones **JoinShared** y **JoinExcl**).

Un mandato suscribe que especifica la opción **JoinShared** o **JoinExcl** añade la **SubIdentity** al conjunto de identidades de la suscripción, si todavía no está presente y si el conjunto de identidades existente permite dicha acción; es decir, ningún otro suscriptor se ha unido de forma exclusiva o el conjunto de identidades está vacío.

Cualquier modificación de los atributos de la suscripción como resultado de un mandato Registrar suscriptor en el que se especifica una **SubIdentity**, sólo se ejecuta correctamente si sería el único miembro del conjunto de identidades para esta suscripción. De lo contrario, el mandato falla con el código de retorno *MQRCCF_SUBSCRIPTION_IN_USE*. Esto impide que los atributos de una suscripción cambien sin que otros suscriptores interesados sean conscientes.

Si especifica una serie de caracteres de más de 64 caracteres, el mandato falla con el código de retorno *MQRCCF_SUB_IDENTITY_ERROR*.

SubUserData(MQPSC_SUBSCRIPTION_USER_DATA)

Es una serie de texto de longitud variable. El valor lo almacena el gestor de colas con la suscripción, pero no influye en la entrega de publicaciones al suscriptor. El valor se puede modificar volviendo a registrar en la misma suscripción con un nuevo valor. Este atributo está ahí para el uso de la aplicación.

El "**SubUserData**" se devuelve en la información metatópica (*MQCACF_REG_SUB_USER_DATA*) de una suscripción si está presente.

Si especifica más de uno de los valores de opción de registro *NonPers*, *PersAsPub*, *PersAsQueue*, and *Pers*, sólo se utiliza el último. No puede combinar estas opciones en una suscripción individual.

Ejemplo

He aquí un ejemplo de *NameValueData* para un mensaje de comando '**Register Subscriber**'. En la aplicación de ejemplo, el servicio de resultados utiliza este mensaje para registrar una suscripción a los temas que contienen las últimas puntuaciones en todas las coincidencias, con la opción 'Persistente como publicación' establecida. La identidad del abonado, incluido el *CorrelId*, se toma de los valores predeterminados en el MQMD.

```
<psc>
  <Command>RegSub</Command>
  <RegOpt>PersAsPub</RegOpt>
  <RegOpt>CorrelAsId</RegOpt>
  <Topic>Sport/Soccer/State/LatestScore/#</Topic>
</psc>
```

Mensaje de solicitud de actualización

El mensaje del mandato **Request Update** se envía desde un suscriptor a un gestor de colas, para solicitar las publicaciones retenidas actuales para el tema especificado y el punto de suscripción que coinciden con el filtro especificado (opcional).

Este mensaje se envía a *SYSTEM.BROKER.CONTROL.QUEUE*, la cola de control del gestor de colas. Se necesita autorización para colocar un mensaje en esta cola, además de autorización de acceso para el tema en la solicitud de actualización; esto lo establece el administrador del sistema del gestor de colas.

Este comando se utiliza normalmente si el abonado especificó la opción *PubOnReqOnly* al registrarse. Si el gestor de colas tiene publicaciones retenidas coincidentes, se envían al suscriptor. Si el gestor de colas no tiene publicaciones retenidas coincidentes, la solicitud falla con el código de retorno *MQRCCF_NO_RETAINED_MSG*. El solicitante debe haber registrado previamente una suscripción con los mismos valores de Tema, SubPoint, y Filtro.

Propiedades

Mandato (MQPSC_COMMAND)

El valor es *ReqUpdate* (*MQPSC_REQUEST_UPDATE*). Esta propiedad ha de especificarse.

Tema (MQPSC_TOPIC)

El valor es el tema que solicita el suscriptor; se permiten caracteres comodín.

Esta propiedad debe especificarse, pero sólo se permite una aparición en este mensaje.

SubPoint (MQPSC_SUBSCRIPTION_POINT)

El valor es el punto de suscripción al que se adjunta la suscripción.

Si se omite esta propiedad, se utiliza el punto de suscripción predeterminado.

Filtro (MQPSC_FILTER)

El valor es una expresión ESQL que se utiliza como filtro en el contenido de los mensajes de publicación. Si una publicación en el tema especificado coincide con el filtro, se envía al suscriptor.

La propiedad < Filtro > debe tener el mismo valor que el especificado en la suscripción original para la que está solicitando una actualización.

Si se omite esta propiedad, no se realiza ningún filtrado de contenido.

RegOpt (MQPSC_REGISTRATION_OPTION)

La propiedad de opciones de registro puede tomar el valor siguiente:

CorrelAsId

(MQPSC_CORREL_ID_AS_IDENTITY)

El CorrelId del descriptor de mensaje (MQMD), que no debe ser cero, se utiliza al enviar publicaciones coincidentes a la cola de suscriptores.

Ninguno

(MQPSC_NONE)

Todas las opciones toman sus valores predeterminados. Esto tiene el mismo efecto que omitir la propiedad <RegOpt>. Si se especifican otras opciones simultáneamente, se ignora None.

VariableUserId

(MQPSC_VARIABLE_USER_ID)

Cuando se especifica, la identidad del suscriptor (cola, gestor de colas y correlid) no está restringida a un único ID de usuario. Esto difiere del comportamiento existente del gestor de colas que asocia el ID de usuario del mensaje de registro original con la identidad del suscriptor y, a partir de entonces, impide que cualquier otro usuario utilice dicha identidad. Si un nuevo suscriptor intenta utilizar la misma identidad, el mandato falla con el código de retorno *MQRCCF_DUPLICATE_SUBSCRIPTION*.

Esto permite a cualquier usuario modificar o anular el registro de la suscripción cuando tenga la autorización adecuada. Por lo tanto, no es necesario comprobar que el ID de usuario coincide con el del suscriptor original.

Para añadir esta opción a una suscripción existente, el mandato debe proceder del mismo ID de usuario que la suscripción original.

Si la suscripción del comando ' **Request Update** ' tiene establecido VariableUserId, éste debe establecerse en el momento de actualizar la petición para indicar a qué suscripción se hace referencia. De lo contrario, el ID de usuario del mandato **Request Update** se utiliza para identificar la suscripción. Esto se altera temporalmente, junto con los otros identificadores de suscriptor, si se proporciona un nombre de suscripción.

El valor predeterminado, si se omite esta propiedad, es que no se establecen opciones de registro.

QMgrName (MQPSC_Q_MGR_NAME)

El valor es el nombre del gestor de colas para la cola de suscriptor, a la que el gestor de colas envía la publicación retenida coincidente.

Si se omite esta propiedad, el valor predeterminado es el nombre del gestor de la cola de respuestas (ReplyToQMgr) que hay en el descriptor del mensaje (MQMD). Si el nombre resultante está en blanco, el valor predeterminado es QMgrNamedel gestor de colas.

QName (MQPSC_Q_NAME)

El valor es el nombre de la cola de suscriptores, a la que el gestor de colas envía la publicación retenida coincidente.

Si se omite esta propiedad, el valor por defecto es el nombre de ReplyToQ en el descriptor del mensaje (MQMD), que no debe estar en blanco en este caso.

SubName (NOMBRE_SUSCRIPCIÓN_MQPSC)

Se trata de un nombre asignado a una suscripción determinada. Si se especifica en un comando '**Request Update**', el valor de SubName tiene prioridad sobre todos los demás campos identificadores excepto el userid, a menos que VariableUserId se establezca en la propia suscripción. Si no se establece VariableUserId, el comando *Solicitar actualización* sólo tiene éxito si el userid del mensaje de comando coincide con el de la suscripción. Si el ID de usuario del mensaje de mandato no coincide con el de la suscripción, el mandato falla con el código de retorno *MQRCCF_DUPLICATE_IDENTITY*.

Si se establece VariableUserId, y el userid difiere del de la suscripción, el comando tiene éxito si el userid del nuevo mensaje de comando tiene autoridad para examinar la cola de flujo y poner a la cola de suscriptores de la suscripción. De lo contrario, el mandato falla con el código de retorno *MQRCCF_NOT_AUTHORIZED*.

Si existe una suscripción que coincide con la identidad tradicional de este comando, pero no tiene SubName, el comando '**Request Update**' falla con el código de retorno *MQRCCF_SUB_NAME_ERROR*.

Si se intenta solicitar una actualización para una suscripción que tiene un SubName utilizando un mensaje de mandato que coincide con la identidad tradicional, pero sin ningún SubName especificado, el mandato se ejecuta correctamente.

Ejemplo

He aquí un ejemplo de NameValueData para un mensaje de comando '**Request Update**'. En la aplicación de ejemplo, el servicio de resultados utiliza este mensaje para solicitar publicaciones retenidas que contienen las puntuaciones más recientes para todos los equipos. La identidad del abonado, incluido el CorrelId, se toma de los valores predeterminados en el MQMD.

```
<psc>
  <Command>ReqUpdate</Command>
  <RegOpt>CorrelAsId</RegOpt>
  <Topic>Sport/Soccer/State/LatestScore/#</Topic>
</psc>
```

Mensaje de respuesta del gestor de colas

Un mensaje '**Queue Manager Response**' se envía desde un gestor de colas al ReplyToQ de un editor o un suscriptor, para indicar el éxito o el fracaso de un mensaje de comando recibido por el gestor de colas si el descriptor del mensaje de comando especificaba que se requería una respuesta.

El mensaje de respuesta está contenido en el campo NameValueData de la cabecera MQRFH2, en una carpeta <pscr>.

En el caso de un aviso o error, el mensaje de respuesta contiene la carpeta <psc> del mensaje de mandato, así como la carpeta <pscr>. Los datos del mensaje, si los hay, no están contenidos en el mensaje de respuesta del gestor de colas. Si se trata de un error, ninguno de los mensajes que haya causado el error se procesará; si se trata de un aviso, es posible que alguno de los mensajes se haya procesado correctamente.

Si se produce una anomalía al enviar una respuesta:

- Para los mensajes de publicación, el gestor de colas intenta enviar la respuesta a la cola de mensajes no entregados IBM MQ si la MQPUT falla. Esto permite enviar la publicación a suscriptores incluso si la respuesta no puede devolver al publicador.
- Para otros mensajes o si la respuesta a la publicación no puede enviarse a la cola de mensajes no entregados, se anota un error y, normalmente, el mensaje se sustituye. El comportamiento depende de cómo se haya configurado el nodo MQInput.

Propiedades

Finalización (**MQPSCR_COMPLETE**)

Es el código de terminación, que puede tener uno de estos tres valores.

Aceptar

El mandato ha terminado correctamente

warning

El mandato ha terminado pero con un error

error

El comando ha fallado

Respuesta (**MQPSCR_RESPONSE**)

La respuesta a un mensaje de mandato, si dicho mandato ha devuelto un código de terminación **aviso** o **error**. Contiene una propiedad `<Reason>` y puede contener otras propiedades que indiquen la causa del aviso o error.

Si hay uno o más errores, habrá sólo una carpeta de respuesta que indicará únicamente la causa del primer error. En el caso de uno o más avisos, habrá una carpeta de respuesta para cada aviso.

Razón (**MQPSCR_REASON**)

Es el código de razón que califica al código de terminación, si el código de terminación es un **aviso** o **error**. Se establece en uno de los códigos de error listados en el ejemplo siguiente. La propiedad `<Reason>` está contenida en una carpeta `<Response>`. El código de razón puede ir seguido de cualquier propiedad válida de la carpeta `<psc>` (por ejemplo, un nombre de tema), que indique la causa del error o aviso. Si obtiene un código de razón de ???, comprobar la exactitud de los datos, por ejemplo, corchetes coincidentes (`< >`).

Ejemplos

He aquí algunos ejemplos de `NameValueData` en un mensaje ' **Queue Manager Response** '. Una respuesta satisfactoria podría ser la siguiente:

```
<pscr>
  <Completion>ok</Completion>
</pscr>
```

Este es un ejemplo de respuesta de anomalía; la anomalía es un error de filtro. La primera cadena `NameValueData` contiene la respuesta; la segunda, el comando original.

```
<pscr>
  <Completion>error</Completion>
  <Response>
    <Reason>3150</Reason>
  </Reponse>
</pscr>

<psc>
  ...
  command message (to which
  the queue manager is responding)
  ...
</psc>
```

Este es un ejemplo de respuesta de aviso (debida a temas no autorizados). La primera cadena " `NameValueData` " contiene la respuesta; la segunda cadena " `NameValueData` " contiene el comando original.

```
<pscr>
  <Completion>warning</Completion>
  <Response>
    <Reason>3081</Reason>
    <Topic>topic1</Topic>
  </Reponse>
  <Response>
    <Reason>3081</Reason>
    <Topic>topic2</Topic>
  </Reponse>
```

```

</pscr>

<psc>
...
command message (to which
the queue manager is responding)
...
</psc>

```

Códigos de razón de publicación/suscripción

Estos códigos de razón se pueden devolver en el campo Razón de una carpeta <pscr> de respuesta de publicación/suscripción. También se listan las constantes que se pueden utilizar para representar estos códigos en los lenguajes de programación C o C++.

Las constantes MQRC_ requieren el archivo de cabecera IBM MQ cmqc.h. Las constantes MQRCCF_ requieren el archivo de cabecera IBM MQ cmqcf.h (aparte de MQRCCF_FILTER_ERROR y MQRCCF_INJU_USER, que requieren el archivo de cabecera cmqpsc.h).

Código de razón y texto	Explicación	Emitida por
2336 MQRC_RFH_COMMAND_ERROR	Los valores válidos para el campo <Command> de una carpeta <psc> son: RegSub, DeregSub, Publicar, DeletePub, y ReqUpdate. Cualquier otro valor hace que se emita este código de error.	Cualquier mandato
2337 MQRC_RFH_ERROR_PARÁMETRO	Las carpetas <psc> y <mcd> tienen un conjunto de parámetros válidos que se pueden especificar dentro de ellas. Compruebe las descripciones de estas carpetas y asegúrese de que no ha especificado parámetros incorrectos.	Cualquier mandato
2338 MQRC_RFH_PARÁMETRO_DUPLICADO	Algunos parámetros (por ejemplo, Tema) dentro de una carpeta <psc> se pueden repetir, pero otros (por ejemplo, Mandato) no se pueden repetir. Compruebe que no ha duplicado un parámetro no repetible.	Cualquier mandato
2339 MQRC_RFH_PARM_MISSING	Algunos parámetros de las carpetas <psc> o <mcd> son opcionales y se pueden omitir; algunos son obligatorios y no se deben omitir. Compruebe que ha incluido todos los parámetros obligatorios en las carpetas <psc> y <mcd>.	Cualquier mandato
2551 MQRC_SELECTION_NOT_AVAILABLE	No había ningún proveedor de selección de mensajes ampliado disponible para determinar qué suscriptores con un filtro especificado deben recibir la publicación.	Publicar, registrar suscriptor y solicitar actualización
	No había ningún proveedor de selección de mensajes ampliado disponible para manejar el filtro del suscriptor especificado.	Registrar suscriptor y solicitar actualización

Código de razón y texto	Explicación	Emitida por
2554 ERROR DE MQRC_CONTENT_	Un proveedor de selección de mensajes ampliados ha encontrado un error en la publicación actual o retenida.	Publicar y solicitar actualización
3008 MQRCCF_COMMAND_FAILED	Se ha producido un error interno que ha impedido que el mandato se ejecutara correctamente. El error puede producirse si se vuelve a emitir el mandato. El registro de sucesos del sistema para el gestor de colas contiene información que se debe utilizar al notificar el problema a IBM.	Cualquier mandato
3072 MQRCCF_TOPIC_ERROR	Uno o varios de los valores que ha proporcionado para el parámetro Topic son incorrectos. Compruebe que los valores de Topic se ajustan a las restricciones especificadas.	Cualquier mandato
3073 MQRCCF_NO_REGISTRADO	La combinación de SubPoint, Tema, y Filtro que especificó en su comando DeregSub o ReqUpdate no era una combinación con la que se había registrado previamente o, para el comando DeregSub si se especificó la opción DeregAll, una de las propiedades SubPoint, Tema, o Filtro no se utilizó para dar de baja ninguna suscripción.	Mandatos de anulación de registro de suscriptor y solicitud de actualización
3074 MQRCCF_Q_MGR_NAME_ERROR	El gestor de colas especificado no era válido, o el gestor de colas no estaba disponible o no existía.	Mandatos Anular registro de suscriptor, Publicar, Registrar suscriptor y Solicitar actualización
3076 MQRCCF_Q_NAME_ERROR	El nombre de cola especificado no era válido o la cola no existía en el gestor de colas especificado.	Mandatos Anular registro de suscriptor, Publicar, Registrar suscriptor y Solicitar actualización
3077 MQRCCF_NO_MENSAJE_RETENIDO	No había mensajes retenidos para el tema que ha especificado. Esto puede ser o no un error, en función del diseño del programa de aplicación.	Mandato de solicitud de actualización

Código de razón y texto	Explicación	Emitida por
3079 MQRCCF_INCORRECT_Q	RegSub, DeregSub, y ReqUpdate se envían siempre a la cola SYSTEM.BROKER.CONTROL.QUEUE del gestor de colas al que van destinados. Los mandatos de publicación y supresión se envían a la cola de entrada para el flujo de mensajes de publicación/suscripción concreto para el que están previstos; esto se determina cuando se diseña el flujo de mensajes. Este código de error se devuelve si se envía un mandato a la cola incorrecta.	Cualquier mandato
3080 MQRCCF_ID_CORREL_ERROR	Ha especificado CorrelAsId como uno de sus parámetros RegOpt. Sin embargo, el campo CorrelId del MQMD no contiene un identificador de correlación válido (es decir, se establece en MQCI_NONE).	Mandatos de anulación de registro de suscriptor y registro de suscriptor
3081 MQRCCF_NO_AUTORIZADO	No tiene autorización para realizar la acción solicitada. Los valores de autorización para el gestor de colas los maneja el administrador del sistema utilizando el editor Jerarquía de temas.	Publicar y registrar mandatos de suscriptor
3083 MQRCCF_REG_OPTIONS_ERROR	Ha especificado un parámetro RegOpt no reconocido en la carpeta <psc> que contiene el mandato RegSub o DeregSub .	Mandatos de anulación de registro de suscriptor y registro de suscriptor
3084 MQRCCF_PUB_OPTIONS_ERROR	Ha especificado un parámetro PubOpt no reconocido en la carpeta <psc> que contiene el mandato Publish.	Mandato de publicación
3087 MQRCCF_DEL_OPTIONS_ERROR	Ha especificado un parámetro DelOpt no reconocido en la carpeta <psc> que contiene el mandato DeletePub .	Mandato Suprimir publicación
3150 MQRCCF_FILTER_ERROR	El valor especificado para el parámetro Filtro no es válido. Compruebe la sección que describe la sintaxis válida para las expresiones de filtro y asegúrese de que la expresión se ajusta.	Mandatos Anular registro de suscriptor, Registrar suscriptor y Solicitar actualización
3151 MQRCCF_USUARIO_INCORRECTO	Ya existe una suscripción que coincide con la especificada; sin embargo, la ha registrado un usuario diferente. Una suscripción sólo puede ser cambiada o anulada del registro por el usuario que la registró originalmente.	Mandatos Anular registro de suscriptor, Registrar suscriptor y Solicitar actualización
3152 MQRCCF_DUPLICAR_SUSCRIPCIÓN	Ya existe una suscripción coincidente con un nombre de suscripción diferente.	

Código de razón y texto	Explicación	Emitida por
3153 MQRCCF_ERROR_DE_NOMBRE_SUB	El formato del nombre de suscripción no es válido o ya existe una suscripción coincidente sin nombre de suscripción.	
3154 MQRCCF_ERROR_DE_IDENTIDAD_SECUNDARIA	El parámetro de identidad de suscripción contiene un error. El valor proporcionado supera la longitud máxima permitida, o la identidad de suscripción no es actualmente miembro del conjunto de identidades de la suscripción y no se ha especificado una opción de registro de unión.	
3155 MQRCCF_SUSCRIPCIÓN_EN_USO	Un miembro del conjunto de identidades ha intentado modificar o anular el registro de una suscripción cuando no era el único miembro de este conjunto.	
3156 MQRCCF_SUSCRIPCIÓN_BLOQUEADA	La suscripción está actualmente bloqueada de modo exclusivo por otra identidad.	
3157 MQRCCF_YA_SOMOS_MIEMBROS	Se ha especificado una opción de registro de unión, pero la identidad del suscriptor ya era miembro del conjunto de identidades de la suscripción.	

Valores de MQMD en los mensajes de mandato para el gestor de colas

Las aplicaciones que envían mensajes de mandatos al gestor de colas utilizan los siguientes valores de campos en el descriptor de mensaje (MQMD). Los campos que se dejan como valor predeterminado, o que se pueden establecer en cualquier valor válido de la forma habitual, no se listan aquí.

Informe

Consulte `MsgType` y `CorrelId`.

MsgType

`MsgType` debe establecerse en `MQMT_REQUEST` o `MQMT_DATAGRAM`. Se devolverá `MQRC_MSG_TYPE_ERROR` si `MsgType` no está establecido en uno de estos valores.

`MsgType` debe establecerse en `MQMT_REQUEST` para un mensaje de mandato si siempre se necesita una respuesta. Los distintivos `MQRO_PAN` y `MQRO_NAN` del campo `Informe` no son significativos en este caso.

Si `MsgType` se establece en `MQMT_DATAGRAM`, las respuestas dependen del valor de los distintivos `MQRO_PAN` y `MQRO_NAN` en el campo `Informe`:

- Sólo `MQRO_PAN` significa que el gestor de colas envía una respuesta sólo si el mandato se ejecuta correctamente.
- `MQRO_NAN` solo significa que el gestor de colas envía una respuesta sólo si el mandato falla.
- Si un mandato se completa con un aviso, se envía una respuesta si se establece `MQRO_PAN` o `MQRO_NAN`.
- `MQRO_PAN` + `MQRO_NAN` significa que el gestor de colas envía una respuesta tanto si el mandato se ejecuta correctamente como si falla. Esto tiene el mismo efecto desde la perspectiva del gestor de colas que establecer `MsgType` en `MQMT_REQUEST`.

- Si no se establece MQRO_PAN ni MQRO_NAN, nunca se envía ninguna respuesta.

Formato

Establézcalo en MQFMT_RF_HEADER_2

MsgId

Este campo se establece normalmente en MQMI_NONE, de modo que el gestor de colas genera un valor exclusivo.

CorrelId

Este campo se puede establecer en cualquier valor. Si la identidad del remitente incluye un CorrelId, especifique este valor, junto con MQRO_PASS_CORREL_ID en el campo Informe, para garantizar que se establece en todos los mensajes de respuesta enviados por el gestor de colas al remitente.

ReplyToQ

Este campo define la cola a la que se enviarán las respuestas, si las hay. Puede ser la cola del remitente; esto tiene la ventaja de que el parámetro QName se puede omitir del mensaje. Sin embargo, si las respuestas se van a enviar a una cola diferente, se necesita el parámetro QName .

ReplyToQMgr

Este campo define el gestor de colas para las respuestas. Si deja este campo en blanco (el valor predeterminado), el gestor de colas local pone su propio nombre en este campo.

Valores de MQMD para publicaciones reenviadas por un gestor de colas

Un gestor de colas utiliza estos valores de campos en el descriptor de mensaje (MQMD) cuando envía una publicación a un suscriptor. Todos los demás campos del MQMD se establecen en sus valores predeterminados.

Informe

Informe se establece en MQRO_NONE.

MsgType

MsgType se establece en MQMT_DATAGRAM.

Caducidad

Caducidad se establece en el valor del mensaje Publicar recibido del publicador. En el caso de un mensaje retenido, el tiempo pendiente se reduce en el tiempo aproximado que el mensaje ha estado en el gestor de colas.

Formato

Formato se establece en MQFMT_RF_HEADER_2

MsgId

MsgId se establece en un valor exclusivo.

CorrelId

Si CorrelId forma parte de la identidad del suscriptor, este es el valor especificado por el suscriptor al registrarse. De lo contrario, es un valor distinto de cero elegido por el gestor de colas.

Priority

Prioridad toma el valor establecido por el publicador, o como resuelto si el publicador ha especificado MQPRI_PRIORITY_AS_Q_DEF.

Persistencia

Persistencia toma el valor establecido por el publicador, o como resuelto si el publicador ha especificado MQPER_PERSISTENCE_AS_Q_DEF, a menos que se especifique lo contrario en el mensaje Registrar suscriptor para el suscriptor al que se envía esta publicación.

ReplyToQ

ReplyToQ está en blanco.

ReplyToQMgr

ReplyToQMgr se establece en el nombre del gestor de colas.

UserIdentifier

UserIdentifier es el identificador de usuario del suscriptor, tal como se establece cuando se registra el suscriptor.

AccountingToken

AccountingToken es la señal de contabilidad del suscriptor, tal como se ha establecido cuando el suscriptor se registró por primera vez.

ApplIdentityData

ApplIdentityData son los datos de identidad de la aplicación del abonado, tal y como se establecieron cuando el abonado se registró por primera vez.

PutApplType

PutApplType tiene el valor MQAT_BROKER.

PutApplName

PutApplName se establece en los primeros 28 caracteres del nombre del gestor de colas.

PutDate

PutDate es la fecha en la que el mensaje se puso en la cola del abonado.

PutTime

PutTime es la hora a la que se puso el mensaje en la cola del abonado.

ApplOriginData

ApplOriginData está en blanco.

Valores de MQMD en mensajes de respuesta del gestor de colas

Un gestor de colas utiliza estos valores de campos en el descriptor de mensaje (MQMD) al enviar una respuesta a un mensaje de publicación. Todos los demás campos del MQMD se establecen en sus valores predeterminados.

Informe

Informe se establece en todos los ceros.

MsgType

MsgType se establece en MQMT_REPLY.

Formato

Formato se establece en MQFMT_RF_HEADER_2

MsgId

El valor de MsgId depende de las opciones de Informe del mensaje de mandato original. De forma predeterminada, se establece en MQMI_NONE, para que el gestor de colas genere un valor exclusivo.

CorrelId

El valor de CorrelId depende de las opciones de Informe del mensaje de mandato original. De forma predeterminada, esto significa que el CorrelId se establece en el mismo valor que el MsgId del mensaje de mandato. Esto se puede utilizar para correlacionar mandatos con sus respuestas.

Priority

Prioridad se establece en el mismo valor que en el mensaje de mandato original.

Persistencia

Persistence se establece en el valor establecido en el mensaje de mandato original.

Caducidad

Caducidad se establece en el mismo valor que en el mensaje de mandato original recibido por el gestor de colas.

PutApplType

PutApplType se establece en MQAT_BROKER.

PutApplName

PutApplName se establece en los primeros 28 caracteres del nombre del gestor de colas.

Otros campos de contexto se establecen como si se hubieran generado con MQPMO_PASS_IDENTITY_CONTEXT.

Codificaciones de máquina

Esta sección describe la estructura del campo *Encoding* en el descriptor de mensaje.

Consulte “MQMD - Descriptor de mensaje” en la [página 389](#) para obtener un resumen de los campos de la estructura.

El campo *Encoding* es un entero de 32 bits que se divide en cuatro subcampos separados; estos subcampos identifican:

- La codificación utilizada para enteros binarios
- La codificación utilizada para enteros de decimal empaquetado
- La codificación utilizada para números de coma flotante
- Bits reservados

Cada subcampo se identifica mediante una máscara de bits que tiene 1-bits en las posiciones correspondientes al subcampo y 0-bits en otro lugar. Los bits están numerados de tal manera que el bit 0 es el bit más significativo, y el bit 31 el bit menos significativo. Se definen las máscaras siguientes:

MÁSCARA_ENTERO_MQENC_MÁSCARA

Máscara para codificación de enteros binarios.

Este subcampo ocupa las posiciones de bits 28 a 31 dentro del campo *Encoding*.

MÁSCARA_DECIMAL_MQENC_MÁSCARA

Máscara para codificación de enteros decimales empaquetados.

Este subcampo ocupa las posiciones de bits 24 a 27 dentro del campo *Encoding*.

Máscara MQENC_FLOAT_MASK

Máscara para codificación de coma flotante.

Este subcampo ocupa las posiciones de bits 20 a 23 dentro del campo *Encoding*.

MÁSCARA_RESERVA_MQENC_RESERVADO

Máscara para bits reservados.

Este subcampo ocupa las posiciones de bits 0 a 19 dentro del campo *Encoding*.

Codificación de enteros binarios

Los valores siguientes son válidos para la codificación de entero binario:

MQENC_INTEGER_UNDEFINED

Los enteros binarios se representan utilizando una codificación que no está definida.

MQENC_INTEGER_NORMAL

Los enteros binarios se representan de la forma convencional:

- El byte menos significativo del número tiene la dirección más alta de cualquiera de los bytes del número; el byte más significativo tiene la dirección más baja
- El bit menos significativo de cada byte es adyacente al byte con la siguiente dirección superior; el bit más significativo de cada byte es adyacente al byte con la siguiente dirección inferior

MQENC_INTEGER_REVERSED

Los enteros binarios se representan de la misma forma que MQENC_INTEGER_NORMAL, pero con los bytes ordenados en orden inverso. Los bits dentro de cada byte se organizan de la misma forma que MQENC_INTEGER_NORMAL.

Codificación de enteros decimales empaquetados

Los valores siguientes son válidos para la codificación de enteros decimales empaquetados:

MQENC_DECIMAL_UNDEFINED

Los enteros de decimal empaquetado se representan utilizando una codificación que no está definida.

MQENC_DECIMAL_NORMAL

Los enteros decimales empaquetados se representan de la forma convencional:

- Cada dígito decimal en la forma imprimible del número se representa en decimal empaquetado por un dígito hexadecimal único en el rango de X' 0 'a X' 9'. Cada dígito hexadecimal ocupa cuatro bits, por lo que cada byte del número decimal empaquetado representa dos dígitos decimales en la forma imprimible del número.
- El byte menos significativo del número decimal empaquetado es el byte que contiene el dígito decimal menos significativo. Dentro de ese byte, los cuatro bits más significativos contienen el dígito decimal menos significativo, y los cuatro bits menos significativos contienen el signo. El signo es X'C '(positivo), X'D' (negativo) o X'F' (sin signo).
- El byte menos significativo del número tiene la dirección más alta de cualquiera de los bytes del número; el byte más significativo tiene la dirección más baja.
- El bit menos significativo de cada byte es adyacente al byte con la siguiente dirección superior; el bit más significativo de cada byte es adyacente al byte con la siguiente dirección inferior.

MQENC_DECIMAL_REVERSED

Los enteros decimales empaquetados se representan de la misma forma que MQENC_DECIMAL_NORMAL, pero con los bytes ordenados en orden inverso. Los bits dentro de cada byte se organizan de la misma forma que MQENC_DECIMAL_NORMAL.

Codificación de coma flotante

Los valores siguientes son válidos para la codificación de coma flotante:

MQENC_FLOAT_UNDEFINED

Los números de coma flotante se representan utilizando una codificación que no está definida.

MQENC_FLOAT_IEEE_NORMAL

Los números de coma flotante se representan utilizando el IEEE estándar³ formato de coma flotante, con los bytes ordenados de la siguiente manera:

- El byte menos significativo de la mantisa tiene la dirección más alta de cualquiera de los bytes del número; el byte que contiene el exponente tiene la dirección más baja
- El bit menos significativo de cada byte es adyacente al byte con la siguiente dirección superior; el bit más significativo de cada byte es adyacente al byte con la siguiente dirección inferior

Los detalles de la codificación flotante IEEE se pueden encontrar en el estándar IEEE 754.

MQENC_FLOAT_IEEE_REVERSED

Los números de coma flotante se representan de la misma forma que MQENC_FLOAT_IEEE_NORMAL, pero con los bytes ordenados en orden inverso. Los bits dentro de cada byte se organizan de la misma forma que MQENC_FLOAT_IEEE_NORMAL.

MQENC_FLOAT_S390

Los números de coma flotante se representan utilizando el formato de coma flotante estándar System/390 ; también lo utiliza System/370.

Construcción de codificaciones

Para construir un valor para el campo *Encoding* en MQMD, las constantes relevantes que describen las codificaciones necesarias se pueden añadir juntas (no añadir la misma constante más de una vez) o combinarse utilizando la operación OR a nivel de bit (si el lenguaje de programación soporta operaciones de bits).

Sea cual sea el método utilizado, combine sólo una de las codificaciones MQENC_INTEGER_* con una de las codificaciones MQENC_DECIMAL_* y una de las codificaciones MQENC_FLOAT_*.

³ El Instituto de Ingenieros Eléctricos y Electrónicos

Análisis de codificaciones

El campo *Encoding* contiene subcampos; debido a esto, las aplicaciones que necesitan examinar la codificación entera, decimal empaquetado o flotante deben utilizar una de las técnicas descritas.

Utilización de operaciones de bits

Si el lenguaje de programación da soporte a operaciones de bits, realice los pasos siguientes:

1. Seleccione uno de los valores siguientes, según el tipo de codificación necesario:
 - MQENC_INTEGER_MASK para la codificación de enteros binarios
 - MQENC_DECIMAL_MASK para la codificación de enteros empaquetados
 - MQENC_FLOAT_MASK para la codificación de coma flotanteLlame al valor A.
2. Combine el campo *Encoding* con A utilizando la operación AND a nivel de bit; llame al resultado B.
3. B es la codificación necesaria y se puede probar la igualdad con cada uno de los valores válidos para ese tipo de codificación.

Utilización aritmética

Si el lenguaje de programación *no* da soporte a operaciones de bits, realice los pasos siguientes utilizando aritmética de enteros:

1. Seleccione uno de los valores siguientes, según el tipo de codificación necesario:
 - 1 para la codificación de enteros binarios
 - 16 para la codificación de enteros decimales empaquetados
 - 256 para la codificación de coma flotanteLlame al valor A.
2. Divida el valor del campo *Encoding* por A ; llamar al resultado B.
3. Divida B por 16; llame al resultado C.
4. Multiplique C por 16 y reste de B ; llamar al resultado D.
5. Multiplique D por A ; llamar al resultado E.
6. E es la codificación necesaria y se puede probar la igualdad con cada uno de los valores válidos para ese tipo de codificación.

Resumen de codificaciones de arquitectura de máquina

Las codificaciones para arquitecturas de máquina se muestran en [Tabla 667 en la página 910](#).

Tabla 667. Resumen de codificaciones para arquitecturas de máquina			
Arquitectura de máquina	Codificación de enteros binarios	Codificación de entero decimal empaquetado	Codificación de coma flotante
IBM i	normal	normal	IEEE estándar
Intel x86	reversed	reversed	IEEE invertido
PowerPC	normal	normal	IEEE estándar
System/390	normal	normal	System/390

Opciones de informe y distintivos de mensaje

En esta sección se describen los campos *Report* y *MsgFlags* que forman parte del MQMD del descriptor de mensaje especificado en las llamadas MQGET, MQPUT y MQPUT1 .

Los temas de esta sección describen:

- La estructura del campo de informe y cómo lo procesa el gestor de colas
- Cómo analiza una aplicación el campo de informe
- La estructura del campo de distintivos de mensaje

Para obtener más información sobre el descriptor de mensaje MQMD, consulte [“MQMD - Descriptor de mensaje”](#) en la página 389.

Estructura del campo de informe

Esta información describe la estructura del campo de informe.

El campo *Report* es un entero de 32 bits que se divide en tres subcampos separados. Estos subcampos identifican:

- Opciones de informe que se rechazan si el gestor de colas local no las reconoce
- Opciones de informe que siempre se aceptan, incluso si el gestor de colas local no las reconoce
- Opciones de informe que sólo se aceptan si se cumplen otras condiciones

Cada subcampo se identifica mediante una máscara de bits que tiene 1-bits en las posiciones correspondientes al subcampo y 0-bits en otro lugar. Los bits de un subcampo no son necesariamente adyacentes. Los bits están numerados de tal manera que el bit 0 es el bit más significativo, y el bit 31 el bit menos significativo. Las máscaras siguientes están definidas para identificar los subcampos:

MQRO_RECHAZO_MÁSCARA_INSUFICIENTE

Esta máscara identifica las posiciones de bits dentro del campo *Report* donde las opciones de informe no soportadas por el gestor de colas local hacen que la llamada MQPUT o MQPUT1 falle con el código de terminación MQCC_FAILED y el código de razón MQRC_REPORT_OPTIONS_ERROR.

Este subcampo ocupa las posiciones de bits 3 y 11 a 13.

MQRO_ACCEPT_UNSUP_MASK

Esta máscara identifica las posiciones de bits dentro del campo *Report* donde las opciones de informe que no están soportadas por el gestor de colas local se aceptan en las llamadas MQPUT o MQPUT1 . En este caso, se devuelve el código de terminación MQCC_WARNING con el código de razón MQRC_UNKNOWN_REPORT_OPTION.

Este subcampo ocupa las posiciones de bits 0 a 2, 4 a 10 y 24 a 31.

En este subcampo se incluyen las siguientes opciones de informe:

- MQRO_ACTIVITY
- MQRO_COPY_MSG_ID_TO_CORREL_ID
- MQRO_DEAD_LETTER_Q
- MQRO_DISCARD_MSG
- MQRO_EXCEPTION
- MQRO_EXCEPTION_WITH_DATA
- MQRO_EXCEPTION_WITH_FULL_DATA
- MQRO_EXPIRATION
- MQRO_EXPIRATION_WITH_DATA
- MQRO_EXPIRATION_WITH_FULL_DATA
- MQRO_NAN
- MQRO_NEW_MSG_ID

- MQRO_NONE
- MQRO_PAN
- MQRO_PASS_CORREL_ID
- MQRO_PASS_MSG_ID

MQRO_ACCEPT_UNSUP_IF_XMIT_MASK

Esta máscara identifica las posiciones de bits dentro del campo *Report* donde las opciones de informe que no están soportadas por el gestor de colas local se aceptan en las llamadas MQPUT o MQPUT1 *siempre que* se cumplan las dos condiciones siguientes:

- El mensaje está destinado a un gestor de colas remoto.
- La aplicación no coloca el mensaje directamente en una cola de transmisión local (es decir, la cola identificada por los campos *ObjectQMgrName* y *ObjectName* en el descriptor de objeto especificado en la llamada MQOPEN o MQPUT1 no es una cola de transmisión local).

Se devuelve el código de terminación MQCC_WARNING con el código de razón MQRC_UNKNOWN_REPORT_OPTION si se cumplen estas condiciones y MQCC_FAILED con el código de razón MQRC_REPORT_OPTIONS_ERROR si no es así.

Este subcampo ocupa las posiciones de bits 14 a 23.

En este subcampo se incluyen las siguientes opciones de informe:

- MQRO_COA
- MQRO_COA_WITH_DATA
- MQRO_COA_WITH_FULL_DATA
- MQRO_COD
- MQRO_COD_WITH_DATA
- MQRO_COD_WITH_FULL_DATA

Si se especifica alguna opción en el campo *Report* que el gestor de colas no reconoce, el gestor de colas comprueba cada subcampo a su vez utilizando la operación AND a nivel de bit para combinar el campo *Report* con la máscara para ese subcampo. Si el resultado de esa operación no es cero, se devuelven el código de terminación y los códigos de razón descritos anteriormente.

Si se devuelve MQCC_WARNING, no se define qué código de razón se devuelve si existen otras condiciones de aviso.

La posibilidad de especificar y tener opciones de informe aceptadas que no son reconocidas por el gestor de colas local es útil cuando se envía un mensaje con una opción de informe que es reconocida y procesada por un gestor de colas *remoto*.

Análisis del campo de informe

El campo *Report* contiene subcampos; debido a esto, las aplicaciones que necesitan comprobar si el remitente del mensaje ha solicitado un informe determinado deben utilizar una de las técnicas descritas.

Utilización de operaciones de bits

Si el lenguaje de programación da soporte a operaciones de bits, realice los pasos siguientes:

1. Seleccione uno de los valores siguientes, de acuerdo con el tipo de informe que se va a comprobar:
 - Informe MQRO_COA_WITH_FULL_DATA para COA
 - Informe MQRO_COD_WITH_FULL_DATA para COD
 - MQRO_EXCEPTION_WITH_FULL_DATA para informe de excepciones
 - MQRO_EXPIRATION_WITH_FULL_DATA para informe de caducidad

Llame al valor A.

En z/OS, utilice los valores MQRO_*_WITH_DATA en lugar de los valores MQRO_*_WITH_FULL_DATA.

2. Combine el campo *Report* con A utilizando la operación AND a nivel de bit; llame al resultado B.
3. Pruebe B para la igualdad con cada valor que sea posible para ese tipo de informe.

Por ejemplo, si A es MQRO_EXCEPTION_WITH_FULL_DATA, pruebe la igualdad de B con cada uno de los siguientes para determinar qué ha especificado el remitente del mensaje:

- MQRO_NONE
- MQRO_EXCEPTION
- MQRO_EXCEPTION_WITH_DATA
- MQRO_EXCEPTION_WITH_FULL_DATA

Las pruebas se pueden realizar en el orden que sea más conveniente para la lógica de la aplicación.

Utilice un método similar para probar las opciones MQRO_PASS_MSG_ID o MQRO_PASS_CORREL_ID; seleccione como el valor A cualquiera de estas dos constantes que sea apropiado y, a continuación, continúe como se ha descrito anteriormente.

Utilización aritmética

Si el lenguaje de programación *no* da soporte a operaciones de bits, realice los pasos siguientes utilizando aritmética de enteros:

1. Seleccione uno de los valores siguientes, de acuerdo con el tipo de informe que se va a comprobar:

- Informe MQRO_COA para COA
- Informe MQRO_COD para COD
- MQRO_EXCEPTION para el informe de excepciones
- MQRO_EXPIRATION para informe de caducidad

Llame al valor A.

2. Divida el campo *Report* por A ; llamar al resultado B.
3. Divida B por 8 ; llamar al resultado C.
4. Multiplique C por 8 y resta de B ; llamar al resultado D.
5. Multiplique D por A ; llamar al resultado E.
6. Pruebe E para la igualdad con cada valor que sea posible para ese tipo de informe.

Por ejemplo, si A es MQRO_EXCEPTION, pruebe la igualdad de E con cada uno de los siguientes para determinar qué ha especificado el remitente del mensaje:

- MQRO_NONE
- MQRO_EXCEPTION
- MQRO_EXCEPTION_WITH_DATA
- MQRO_EXCEPTION_WITH_FULL_DATA

Las pruebas se pueden realizar en el orden que sea más conveniente para la lógica de la aplicación.

El pseudocódigo siguiente ilustra esta técnica para los mensajes de informe de excepción:

```
A = MQRO_EXCEPTION
B = Report/A
C = B/8
D = B - C*8
E = D*A
```

Utilice un método similar para probar las opciones MQRO_PASS_MSG_ID o MQRO_PASS_CORREL_ID; seleccione como el valor A cualquiera de estas dos constantes es apropiado y, a continuación, continúe como se ha descrito anteriormente, pero sustituyendo el valor 8 en los pasos anteriores por el valor 2.

Estructura del campo de distintivos de mensaje

Esta información describe la estructura del campo de distintivos de mensaje.

El campo *MsgFlags* es un entero de 32 bits que se divide en tres subcampos separados. Estos subcampos identifican:

- Distintivos de mensaje que se rechazan si el gestor de colas local no los reconoce
- Distintivos de mensajes que siempre se aceptan, incluso si el gestor de colas local no los reconoce
- Distintivos de mensaje que se aceptan sólo si se cumplen otras condiciones

Nota: Todos los subcampos de *MsgFlags* están reservados para que los utilice el gestor de colas.

Cada subcampo se identifica mediante una máscara de bits que tiene 1-bits en las posiciones correspondientes al subcampo y 0-bits en otro lugar. Los bits están numerados de tal manera que el bit 0 es el bit más significativo, y el bit 31 el bit menos significativo. Las máscaras siguientes están definidas para identificar los subcampos:

MQMF_RECHAZAR_MÁSCARA_INSUBVENCIONADA

Esta máscara identifica las posiciones de bits dentro del campo *MsgFlags* donde los distintivos de mensaje que no están soportados por el gestor de colas local hacen que la llamada MQPUT o MQPUT1 falle con el código de terminación MQCC_FAILED y el código de razón MQRC_MSG_FLAGS_ERROR.

Este subcampo ocupa las posiciones de bits 20 a 31.

En este subcampo se incluyen los distintivos de mensaje siguientes:

- MQMF_ÚLTIMO_MENSAJE_EN_GRUPO
- MQMF_ÚLTIMO_SEGMENTO
- MQMF_MSG_IN_GROUP
- SEGMENTO MQMF_SEGMENT
- MQMF_SEGMENTATION_ALLOWED
- MQMF_SEGMENTATION_INHIBIDO

MQMF_ACCEPT_UNSUP_MASK

Esta máscara identifica las posiciones de bits dentro del campo *MsgFlags* donde los distintivos de mensaje que no están soportados por el gestor de colas local se aceptan en las llamadas MQPUT o MQPUT1. El código de terminación es MQCC_OK.

Este subcampo ocupa las posiciones de bits 0 a 11.

MQMF_ACCEPT_UNSUP_IF_XMIT_MASK

Esta máscara identifica las posiciones de bits dentro del campo *MsgFlags* donde los distintivos de mensaje que no están soportados por el gestor de colas local se aceptan en las llamadas MQPUT o MQPUT1 *siempre que* se cumplan las dos condiciones siguientes:

- El mensaje está destinado a un gestor de colas remoto.
- La aplicación no coloca el mensaje directamente en una cola de transmisión local (es decir, la cola identificada por los campos *ObjectQMGrName* y *ObjectName* en el descriptor de objeto especificado en la llamada MQOPEN o MQPUT1 no es una cola de transmisión local).

Se devuelve el código de terminación MQCC_OK si se cumplen estas condiciones y MQCC_FAILED con el código de razón MQRC_MSG_FLAGS_ERROR si no es así.

Este subcampo ocupa las posiciones de bits 12 a 19.

Si hay distintivos especificados en el campo *MsgFlags* que el gestor de colas no reconoce, el gestor de colas comprueba cada subcampo a su vez utilizando la operación AND a nivel de bit para combinar el campo *MsgFlags* con la máscara para ese subcampo. Si el resultado de esa operación no es cero, se devuelven el código de terminación y los códigos de razón descritos anteriormente.

salida de conversión de datos

Esta colección de temas describe la interfaz para la salida de conversión de datos y el proceso realizado por el gestor de colas cuando es necesaria la conversión de datos.

Para obtener más información sobre la conversión de datos, consulte *Conversión de datos en IBM MQ* en <https://www.ibm.com/support/pages/node/317869>.

La salida de conversión de datos se invoca como parte del proceso de la llamada MQGET para convertir los datos del mensaje de aplicación a la representación que necesita la aplicación receptora. La conversión de los datos del mensaje de aplicación es opcional; requiere que se especifique la opción MQGMO_CONVERT en la llamada MQGET.

Se describen los siguientes temas:

- El proceso realizado por el gestor de colas en respuesta a la opción MQGMO_CONVERT; consulte [“Proceso de conversión”](#) en la página 915.
- Convenios de proceso utilizados por el gestor de colas al procesar un formato incorporado; estos convenios también se recomiendan para las salidas escritas por el usuario. Consulte [“Convenios de proceso”](#) en la página 916.
- Consideraciones especiales para convertir mensajes de informe; consulte [“Conversión de mensajes de informe”](#) en la página 921.
- Los parámetros pasados a la salida de conversión de datos; consulte [“MQ_DATA_CONV_EXIT - Salida de conversión de datos”](#) en la página 934.
- Una llamada que se puede utilizar desde la salida para convertir datos de caracteres entre distintas representaciones; consulte [“MQXCNCV - Convertir caracteres”](#) en la página 927.
- El parámetro de estructura de datos que es específico de la salida; consulte [“MQDXP-Parámetro de salida de conversión de datos”](#) en la página 921.

Proceso de conversión

Esta información describe el proceso realizado por el gestor de colas en respuesta a la opción MQGMO_CONVERT.

El gestor de colas realiza las acciones siguientes si se especifica la opción MQGMO_CONVERT en la llamada MQGET y hay un mensaje que se debe devolver a la aplicación:

1. Si se cumple una o más de las condiciones siguientes, no es necesaria ninguna conversión:
 - Los datos del mensaje ya están en el juego de caracteres y la codificación que necesita la aplicación que emite la llamada MQGET. La aplicación debe establecer los campos *CodedCharSetId* y *Encoding* en el parámetro **MsgDesc** de la llamada MQGET en los valores necesarios, antes de emitir la llamada.
 - La longitud de los datos del mensaje es cero.
 - La longitud del parámetro **Buffer** de la llamada MQGET es cero.

En estos casos, el mensaje se devuelve sin conversión a la aplicación que emite la llamada MQGET; los valores *CodedCharSetId* y *Encoding* del parámetro **MsgDesc** se establecen en los valores de la información de control del mensaje y la llamada se completa con una de las siguientes combinaciones de código de terminación y código de razón:

Tabla 668. Combinaciones de código de terminación y código de razón

Código de terminación	Código de razón
MQCC_OK	MQRC_NONE
MQCC_WARNING	MQRC_TRUNCATED_MSG_ACCEPTED
MQCC_WARNING	MQRC_TRUNCATED_MSG_FAILED

Los pasos siguientes sólo se realizan si el juego de caracteres o la codificación de los datos del mensaje difiere del valor correspondiente en el parámetro **MsgDesc** y hay datos que se deben convertir:

2. Si el campo *Format* de la información de control del mensaje tiene el valor MQFMT_NONE, el mensaje se devuelve sin convertir, con el código de terminación MQCC_WARNING y el código de razón MQRC_FORMAT_ERROR.

En todos los demás casos, el proceso de conversión continúa.

3. El mensaje se elimina de la cola y se coloca en un almacenamiento intermedio temporal que tiene el mismo tamaño que el parámetro **Buffer** . Para las operaciones de examen, el mensaje se copia en el almacenamiento intermedio temporal, en lugar de eliminarse de la cola.
4. Si el mensaje debe truncarse para que quepa en el almacenamiento intermedio, se realiza lo siguiente:
 - Si no se ha especificado la opción MQGMO_ACCEPT_TRUNCATED_MSG, el mensaje se devuelve sin convertir, con el código de terminación MQCC_WARNING y el código de razón MQRC_TRUNCATED_MSG_FAILED.
 - Si se ha especificado la opción MQGMO_ACCEPT_TRUNCATED_MSG , el código de terminación se establece en MQCC_WARNING, el código de razón se establece en MQRC_TRUNCATED_MSG_ACCEPTED y el proceso de conversión continúa.
5. Si el mensaje se puede acomodar en el almacenamiento intermedio sin truncamiento, o se ha especificado la opción MQGMO_ACCEPT_TRUNCATED_MSG, se hace lo siguiente:
 - Si el formato es un formato incorporado, el almacenamiento intermedio se pasa al servicio de conversión de datos del gestor de colas.
 - Si el formato no es un formato incorporado, el almacenamiento intermedio se pasa a una salida escrita por el usuario con el mismo nombre que el formato. Si no se puede encontrar la salida, el mensaje se devuelve sin convertir, con el código de terminación MQCC_WARNING y el código de razón MQRC_FORMAT_ERROR.

Si no se produce ningún error, la salida del servicio de conversión de datos o de la salida escrita por el usuario es el mensaje convertido, más el código de terminación y el código de razón que se devolverá a la aplicación que emite la llamada MQGET.
6. Si la conversión es satisfactoria, el gestor de colas devuelve el mensaje convertido a la aplicación. En este caso, el código de terminación y el código de razón devueltos por la llamada MQGET son una de las combinaciones siguientes:

Tabla 669. Combinaciones de código de terminación y código de razón

Código de terminación	Código de razón
MQCC_OK	MQRC_NONE
MQCC_WARNING	MQRC_TRUNCATED_MSG_ACCEPTED

Sin embargo, si la conversión la realiza una salida escrita por el usuario, se pueden devolver otros códigos de razón, incluso cuando la conversión es satisfactoria.

Si la conversión falla, el gestor de colas devuelve el mensaje sin convertir a la aplicación, con los campos *CodedCharSetId* y *Encoding* del parámetro **MsgDesc** establecidos en los valores de la información de control del mensaje y con el código de terminación MQCC_WARNING.

Convenios de proceso

Al convertir un formato incorporado, el gestor de colas sigue los convenios de proceso descritos.

Las salidas escritas por el usuario también deben seguir estas convenciones, aunque el gestor de colas no las aplica. Los formatos incorporados convertidos por el gestor de colas son:

- MQFMT_ADMIN
- MQFMT_CICS (solo z/OS)

- MQFMT_COMMAND_1
- MQFMT_COMMAND_2
- MQFMT_DEAD_LETTER_HEADER
- MQFMT_DIST_HEADER
- MQFMT_EVENT versión 1
- MQFMT_EVENT versión 2
- MQFMT_IMS
- MQFMT_IMS_VAR_STRING
- MQFMT_MD_EXTENSION
- MQFMT_PCF
- MQFMT_REF_MSG_HEADER
- MQFMT_RF_HEADER
- MQFMT_RF_HEADER_2
- MQFMT_STRING
- MQFMT_TRIGGER
- MQFMT_WORK_INFO_HEADER (solo z/OS)
- MQFMT_XMIT_Q_HEADER

1. Si el mensaje se expande durante la conversión y supera el tamaño del parámetro **Buffer** , se realiza lo siguiente:

- Si no se ha especificado la opción MQGMO_ACCEPT_TRUNCATED_MSG, el mensaje se devuelve sin convertir, con el código de terminación MQCC_WARNING y el código de razón MQRC_CONVERTED_MSG_TOO_BIG.
- Si se especifica la opción MQGMO_ACCEPT_TRUNCATED_MSG *era* , el mensaje se trunca, el código de terminación se establece en MQCC_WARNING, el código de razón se establece en MQRC_TRUNCATED_MSG_ACCEPTED y el proceso de conversión continúa.

2. Si se produce un truncamiento (antes o durante la conversión), el número de bytes válidos devueltos en el parámetro **Buffer** puede ser menor que la longitud del almacenamiento intermedio.

Esto puede ocurrir, por ejemplo, si un entero de 4 bytes o un carácter DBCS se encuentra entre los extremos del almacenamiento intermedio. El elemento de información incompleto no se convierte y los bytes del mensaje devuelto no contienen información válida. Esto también puede ocurrir si un mensaje que se ha truncado antes de la conversión se reduce durante la conversión.

Si el número de bytes válidos devueltos es menor que la longitud del almacenamiento intermedio, los bytes no utilizados al final del almacenamiento intermedio se establecen en nulos.

3. Si una matriz o serie se encuentra entre el final del almacenamiento intermedio, se convierten tantos datos como sea posible; sólo no se convierte el elemento de matriz concreto o el carácter DBCS que está incompleto; se convierten los elementos o caracteres de matriz anteriores.

4. Si se produce un truncamiento (antes o durante la conversión), la longitud devuelta para el parámetro **DataLength** es la longitud del mensaje sin convertir antes del truncamiento.

5. Cuando las series se convierten entre juegos de caracteres de un solo byte (SBCS), juegos de caracteres de doble byte (DBCS) o juegos de caracteres de varios bytes (MBCS), las series pueden expandirse o contraerse.

- En los formatos PCF MQFMT_ADMIN, MQFMT_EVENT y MQFMT_PCF, las series de las estructuras MQCFST y MQCFSL se expanden o contraen según sea necesario para acomodar la serie después de la conversión.

Para la estructura de lista de series MQCFSL, las series de la lista pueden expandirse o contraerse en cantidades diferentes. Si esto sucede, el gestor de colas rellena las series más cortas con espacios en blanco para que tengan la misma longitud que la serie más larga después de la conversión.

- En el formato MQFMT_REF_MSG_HEADER, las series a las que se dirigen los campos *SrcEnvOffset*, *SrcNameOffset*, *DestEnvOffset* y *DestNameOffset* se expanden o contraen según sea necesario para acomodar las series después de la conversión.
 - En el formato MQFMT_RF_HEADER, el campo *NameValueString* se expande o contrae según sea necesario para acomodar los pares nombre-valor después de la conversión.
 - En estructuras con tamaños de campo fijos, el gestor de colas permite que las series se expandan o se contraigan dentro de sus campos fijos, siempre que no se pierda información significativa. En este sentido, los espacios en blanco finales y los caracteres que siguen al primer carácter nulo del campo se tratan como insignificantes.
 - Si la serie se expande, pero sólo es necesario descartar caracteres insignificantes para acomodar la serie convertida en el campo, la conversión se realiza correctamente y la llamada se completa con MQCC_OK y el código de razón MQRC_NONE (suponiendo que no haya otros errores).
 - Si la serie se expande, pero la serie convertida requiere que se descarten caracteres significativos para que quepan en el campo, el mensaje se devuelve sin convertir y la llamada se completa con MQCC_WARNING y el código de razón MQRC_CONVERTED_STRING_TOO_BIG.

Nota: El código de razón MQRC_CONVERTED_STRING_TOO_BIG da como resultado en este caso si se ha especificado o no la opción MQGMO_ACCEPT_TRUNCATED_MSG.

 - Si la serie se contrae, el gestor de colas rellena la serie con espacios en blanco hasta la longitud del campo.
6. Para los mensajes que constan de una o más estructuras de cabecera MQ seguidas de datos de usuario, es posible que se conviertan una o más de las estructuras de cabecera, mientras que el resto del mensaje no lo es. Sin embargo, (con dos excepciones) los campos *CodedCharSetId* y *Encoding* de cada estructura de cabecera siempre indican correctamente el juego de caracteres y la codificación de los datos que siguen a la estructura de cabecera.
- Las dos excepciones son las estructuras MQCIH y MQIIH, donde los valores de los campos *CodedCharSetId* y *Encoding* de esas estructuras no son significativos. Para esas estructuras, los datos que siguen a la estructura están en el mismo juego de caracteres y codificación que la propia estructura MQCIH o MQIIH.
7. Si los campos *CodedCharSetId* o *Encoding* de la información de control del mensaje que se está recuperando, o en el parámetro **MsgDesc**, especifican valores que no están definidos o no están soportados, el gestor de colas puede ignorar el error si no es necesario utilizar el valor no definido o no soportado al convertir el mensaje.
- Por ejemplo, si el campo *Encoding* del mensaje especifica una codificación flotante no soportada, pero el mensaje contiene sólo datos enteros, o contiene datos de coma flotante que no requieren conversión (porque las codificaciones flotante de origen y destino son idénticas), es posible que el error no se diagnostique.
- Si se diagnostica el error, el mensaje se devuelve sin convertir, con el código de terminación MQCC_WARNING y uno de los códigos de razón MQRC_SOURCE_*_ERROR o MQRC_TARGET_*_ERROR (según corresponda); los campos *CodedCharSetId* y *Encoding* del parámetro **MsgDesc** se establecen en los valores de la información de control del mensaje.
- Si el error no se diagnostica y la conversión se completa correctamente, los valores devueltos en los campos *CodedCharSetId* y *Encoding* en el parámetro **MsgDesc** son los especificados por la aplicación que emite la llamada MQGET.
8. En todos los casos, si el mensaje se devuelve a la aplicación sin convertir, el código de finalización se establece en MQCC_WARNING, y los campos *CodedCharSetId* y *Encoding* del parámetro **MsgDesc** se establecen en los valores adecuados para los datos sin convertir. Esto también se realiza para MQFMT_NONE.
- El parámetro **Reason** se establece en un código que indica por qué no se ha podido llevar a cabo la conversión, a menos que el mensaje también se haya tenido que truncar; los códigos de razón relacionados con el truncamiento tienen prioridad sobre los códigos de razón relacionados con la conversión. (Para determinar si se ha convertido un mensaje truncado, compruebe los valores devueltos en los campos *CodedCharSetId* y *Encoding* en el parámetro **MsgDesc**.)

Cuando se diagnostica un error, se devuelve un código de razón específico o el código de razón general MQRC_NOT_CONVERT. El código de razón devuelto depende de las prestaciones de diagnóstico del servicio de conversión de datos subyacente.

9. Si se devuelve el código de terminación MQCC_WARNING y hay más de un código de razón relevante, el orden de prioridad es el siguiente:

a. Las razones siguientes tienen prioridad sobre todas las demás; sólo puede surgir una de las razones de este grupo:

- MQRC_SIGNAL_REQUEST_ACCEPTED
- MQRC_TRUNCATED_MSG_ACCEPTED

b. El orden de prioridad dentro de los códigos de razón restantes no está definido.

10. Al finalizar la llamada MQGET:

- El siguiente código de razón indica que el mensaje se ha convertido correctamente:
 - MQRC_NONE
- Los siguientes códigos de razón indican que el mensaje *podría* haberse convertido correctamente (compruebe los campos *CodedCharSetId* y *Encoding* en el parámetro **MsgDesc** para averiguarlo):
 - MQRC_MSG_MARKED_BROWSE_CO_OP
 - MQRC_TRUNCATED_MSG_ACCEPTED
- Los demás códigos de razón indican que el mensaje no se ha convertido.

El siguiente proceso es específico de los formatos incorporados; no se aplica a los formatos definidos por el usuario:

11. Con la excepción de los formatos siguientes:

- MQFMT_ADMIN
- MQFMT_COMMAND_1
- MQFMT_COMMAND_2
- MQFMT_EVENT
- MQFMT_IMS_VAR_STRING
- MQFMT_PCF
- MQFMT_STRING

ninguno de los formatos incorporados se puede convertir de o a conjuntos de caracteres que no tengan caracteres SBCS para los caracteres que son válidos en los nombres de cola. Si se intenta realizar una conversión de este tipo, el mensaje se devuelve sin convertir, con el código de terminación MQCC_WARNING y el código de razón MQRC_SOURCE_CCSID_ERROR o MQRC_TARGET_CCSID_ERROR, según corresponda.

El juego de caracteres Unicode UTF-16 es un ejemplo de un juego de caracteres que no tiene caracteres SBCS para los caracteres que son válidos en los nombres de cola.

12. Si los datos de mensaje para un formato incorporado se truncan, los campos dentro del mensaje que contienen longitudes de series, o recuentos de elementos o estructuras, no se ajustan para reflejar la longitud de los datos devueltos realmente a la aplicación; los valores devueltos para dichos campos dentro de los datos del mensaje son los valores aplicables al mensaje *antes del truncamiento*.

Al procesar mensajes como, por ejemplo, un mensaje MQFMT_ADMIN truncado, asegúrese de que la aplicación no intenta acceder a los datos más allá del final de los datos devueltos.

13. Si el nombre de formato es MQFMT_DEAD_LETTER_HEADER, los datos de mensaje empiezan con una estructura MQDLH, posiblemente seguida de cero o más bytes de datos de mensaje de aplicación. El formato, el juego de caracteres y la codificación de los datos del mensaje de aplicación se definen mediante los campos *Format*, *CodedCharSetId* y *Encoding* en la estructura MQDLH al principio del mensaje. Debido a que la estructura MQDLH y los datos de mensajes de aplicación

pueden tener diferentes conjuntos de caracteres y codificaciones, uno, otro o ambos de la estructura MQDLH y los datos de mensajes de aplicación pueden requerir conversión.

El gestor de colas convierte primero la estructura MQDLH, según sea necesario. Si la conversión es satisfactoria, o la estructura MQDLH no requiere conversión, el gestor de colas comprueba los campos CodedCharSetId y Encoding de la estructura MQDLH para ver si es necesaria la conversión de los datos del mensaje de aplicación. Si la conversión es necesaria, el gestor de colas invoca la salida escrita por el usuario con el nombre proporcionado por el campo Format en la estructura MQDLH, o realiza la propia conversión (si Format es el nombre de un formato incorporado).

Si la llamada MQGET devuelve un código de terminación de MQCC_WARNING, y el código de razón es uno de los que indica que la conversión no ha sido satisfactoria, se aplica uno de los siguientes:

- No se ha podido convertir la estructura MQDLH. En este caso, los datos del mensaje de aplicación tampoco se habrán convertido.
- La estructura MQDLH se ha convertido, pero los datos del mensaje de aplicación no.

La aplicación puede examinar los valores devueltos en los campos CodedCharSetId y Encoding en el parámetro **MsgDesc**, y los de la estructura MQDLH, para determinar cuál de los anteriores se aplica.

14. Si el nombre de formato es MQFMT_XMIT_Q_HEADER, los datos del mensaje empiezan con una estructura MQXQH, posiblemente seguida de cero o más bytes de datos adicionales. Estos datos adicionales suelen ser los datos de mensaje de aplicación (que pueden ser de longitud cero), pero también puede haber una o más estructuras de cabecera de MQ adicionales presentes, al principio de los datos adicionales.

La estructura MQXQH debe estar en el juego de caracteres y la codificación del gestor de colas. El formato, el juego de caracteres y la codificación de los datos que siguen a la estructura MQXQH se proporcionan mediante los campos Format, CodedCharSetId y Encoding en la estructura MQMD contenida en MQXQH. Para cada estructura de cabecera MQ posterior presente, los campos Format, CodedCharSetId y Encoding de la estructura describen los datos que siguen a esa estructura; esos datos son otra estructura de cabecera MQ o los datos de mensaje de aplicación.

Si se especifica la opción MQGMO_CONVERT para un mensaje MQFMT_XMIT_Q_HEADER, los datos del mensaje de aplicación y algunas de las estructuras de cabecera de MQ se convierten, *pero los datos de la estructura MQXQH no son*. Al volver de la llamada MQGET, por lo tanto:

- Los valores de los campos Format, CodedCharSetId y Encoding del parámetro **MsgDesc** describen los datos de la estructura MQXQH y no los datos del mensaje de aplicación; por lo tanto, los valores no son los mismos que los especificados por la aplicación que ha emitido la llamada MQGET.

El efecto de esto es que una aplicación que obtiene repetidamente mensajes de una cola de transmisión con la opción MQGMO_CONVERT especificada debe restablecer los campos CodedCharSetId y Encoding en el parámetro **MsgDesc** a los valores necesarios para los datos de mensaje de aplicación, antes de cada llamada MQGET.

- Los valores de los campos Format, CodedCharSetId y Encoding de la última estructura de cabecera de MQ presente describen los datos del mensaje de aplicación. Si no hay otras estructuras de cabecera MQ presentes, los datos del mensaje de aplicación se describen mediante estos campos en la estructura MQMD dentro de la estructura MQXQH. Si la conversión es satisfactoria, los valores serán los mismos que los especificados en el parámetro **MsgDesc** por la aplicación que ha emitido la llamada MQGET.

Si el mensaje es un mensaje de lista de distribución, la estructura MQXQH va seguida de una estructura MQDH (más sus matrices de registros MQOR y MQPMR), que a su vez puede ir seguida de cero o más estructuras de cabecera MQ adicionales y de cero o más bytes de datos de mensaje de aplicación. Al igual que la estructura MQXQH, la estructura MQDH debe estar en el juego de caracteres y la codificación del gestor de colas, y no se convierte en la llamada MQGET, incluso si se especifica la opción MQGMO_CONVERT.

El proceso de las estructuras MQXQH y MQDH descritas anteriormente está pensado principalmente para que las utilicen los agentes de canal de mensajes cuando obtienen mensajes de las colas de transmisión.

Conversión de mensajes de informe

En general, un mensaje de informe puede contener cantidades variables de datos de mensaje de aplicación, según las opciones de informe especificadas por el remitente del mensaje original. Sin embargo, un informe de actividad puede contener datos pero sin que la opción de informe mencione * _WITH_DATA en la constante.

En concreto, un mensaje de informe puede contener:

1. No hay datos de mensaje de aplicación
2. Algunos de los datos de mensaje de aplicación del mensaje original

Esto ocurre cuando el remitente del mensaje original especifica MQRO_ * _WITH_DATA y el mensaje tiene más de 100 bytes.

3. Todos los datos de mensaje de aplicación del mensaje original

Esto ocurre cuando el remitente del mensaje original especifica MQRO_ * _WITH_FULL_DATA, o especifica MQRO_ * _WITH_DATA y el mensaje tiene 100 bytes o menos.

Cuando el gestor de colas o el agente de canal de mensajes genera un mensaje de informe, copia el nombre de formato del mensaje original en el campo *Format* de la información de control del mensaje de informe. Por lo tanto, el nombre de formato del mensaje de informe puede implicar una longitud de datos diferente de la longitud realmente presente en el mensaje de informe (casos 1 y 2 anteriores).

Si se especifica la opción MQGMO_CONVERT cuando se recupera el mensaje de informe:

- Para el caso 1 anterior, no se invoca la salida de conversión de datos (porque el mensaje de informe no tiene datos).
- Para el caso 3 anterior, el nombre de formato implica correctamente la longitud de los datos del mensaje.
- Pero para el caso 2 anterior, se invoca la salida de conversión de datos para convertir un mensaje que es *más corto* que la longitud implícita por el nombre de formato.

Además, el código de razón pasado a la salida suele ser MQRC_NONE (es decir, el código de razón no indica que el mensaje se haya truncado). Esto sucede porque los datos del mensaje han sido truncados por el *emisor* del mensaje de informe y no por el gestor de colas del receptor en respuesta a la llamada MQGET.

Debido a estas posibilidades, la salida de conversión de datos no debe utilizar el nombre de formato para deducir la longitud de los datos que se le pasan; en su lugar, la salida debe comprobar la longitud de los datos proporcionados y estar preparada para convertir menos datos que la longitud implícita por el nombre de formato. Si los datos se pueden convertir correctamente, la salida debe devolver el código de terminación MQCC_OK y el código de razón MQRC_NONE. La longitud de los datos de mensaje que se van a convertir se pasa a la salida como parámetro **InBufferLength**.

Interfaz de programación sensible al producto

MQDXP-Parámetro de salida de conversión de datos

La estructura MQDXP es un parámetro que el gestor de colas pasa a la salida de conversión de datos cuando se invoca la salida para convertir los datos del mensaje como parte del proceso de la llamada MQGET. Consulte la descripción de la llamada MQ_DATA_CONV_EXIT para obtener detalles de la salida de conversión de datos.

Los datos de tipo carácter en MQDXP están en el juego de caracteres del gestor de colas local; esto lo proporciona el atributo de gestor de colas **CodedCharSetId**. Los datos numéricos en MQDXP están en la codificación de máquina nativa; esto lo proporciona MQENC_NATIVE.

La salida sólo puede cambiar los campos *DataLength*, *CompCode*, *Reason* y *ExitResponse* en MQDXP; los cambios en otros campos se ignoran. Sin embargo, el campo *DataLength* no se puede cambiar si el mensaje que se está convirtiendo es un segmento que sólo contiene parte de un mensaje lógico.

Cuando el control vuelve al gestor de colas desde la salida, el gestor de colas comprueba los valores devueltos en MQDXP. Si los valores devueltos no son válidos, el gestor de colas continúa el proceso como si la salida hubiera devuelto MQXDR_CONVERSION_FAILED en *ExitResponse* ; sin embargo, el gestor de colas ignora los valores de los campos *CompCode* y *Reason* devueltos por la salida en este caso, y utiliza en su lugar los valores que esos campos tenían en *entrada* para la salida. Los siguientes valores en MQDXP hacen que se produzca este proceso:

- El campo *ExitResponse* no es MQXDR_OK y no es MQXDR_CONVERSION_FAILED
- El campo *CompCode* no es MQCC_OK y no es MQCC_WARNING
- *DataLength* campo menor que cero, o *DataLength* campo cambiado cuando el mensaje que se está convirtiendo es un segmento que sólo contiene parte de un mensaje lógico.

La tabla siguiente resume los campos de la estructura.

Tabla 670. Campos en MQDXP		
Campo	Descripción	Tema
<i>StrucId</i>	Identificador de la estructura	StrucId
<i>Version</i>	Número de versión de la estructura	Versión
<i>AppOptions</i>	Opciones de aplicaciones	AppOptions
<i>Encoding</i>	Codificación numérica necesaria para la aplicación	Encoding
<i>CodedCharSetId</i>	Juego de caracteres necesario para la aplicación	CodedCharSetId
<i>DataLength</i>	Longitud en bytes de datos de mensaje	DataLength
<i>CompCode</i>	Código de terminación	CompCode
<i>Reason</i>	Código de razón que califica <i>CompCode</i>	Razón
<i>ExitResponse</i>	Respuesta de la salida	ExitResponse
<i>Hconn</i>	Descriptor de contexto de conexión	Hconn
<i>pEntryPoints</i>	Dirección de la estructura MQIEP	pEntryPoints

Campos

La estructura MQDXP contiene los campos siguientes; los campos se describen en orden alfabético.

AppOptions

Tipo: MQLONG

Se trata de una copia del campo *Options* de la estructura MQGMO especificada por la aplicación que emite la llamada MQGET. Es posible que la salida tenga que examinarlos para determinar si se ha especificado la opción MQGMO_ACCEPT_TRUNCATED_MSG.

Es un campo de entrada para la salida.

CodedCharSetId

Tipo: MQLONG

Este es el identificador de juego de caracteres codificado del juego de caracteres que necesita la aplicación que emite la llamada MQGET; consulte el campo *CodedCharSetId* en la estructura MQMD para obtener más detalles. Si la aplicación especifica el valor especial MQCCSI_Q_MGR en la llamada MQGET, el gestor de colas lo cambia por el identificador de juego de caracteres real del juego de caracteres utilizado por el gestor de colas, antes de invocar la salida.

Si la conversión es satisfactoria, la salida debe copiarla en el campo *CodedCharSetId* del descriptor de mensaje.

Es un campo de entrada para la salida.

CompCode

Tipo: MQLONG

Cuando se invoca la salida, contiene el código de terminación que se devuelve a la aplicación que ha emitido la llamada MQGET, si la salida no hace nada. Siempre es MQCC_WARNING, porque el mensaje se ha truncado o el mensaje requiere conversión y esto todavía no se ha realizado.

En la salida de la salida, este campo contiene el código de terminación que se devolverá a la aplicación en el parámetro **CompCode** de la llamada MQGET; sólo son válidos MQCC_OK y MQCC_WARNING. Consulte la descripción del campo *Reason* para obtener sugerencias sobre cómo la salida puede establecer este campo en la salida.

Es un campo de entrada/salida para la salida.

DataLength

Tipo: MQLONG

Cuando se invoca la salida, este campo contiene la longitud original de los datos del mensaje de aplicación. Si el mensaje se ha truncado para que quepa en el almacenamiento intermedio proporcionado por la aplicación, el tamaño del mensaje proporcionado a la salida es *menor* que el valor de *DataLength*. El tamaño del mensaje proporcionado a la salida siempre lo proporciona el parámetro **InBufferLength** de la salida, independientemente de cualquier truncamiento que se haya producido.

El truncamiento se indica mediante el campo *Reason* que tiene el valor MQRC_TRUNCATED_MSG_ACCEPTED en la entrada de la salida.

La mayoría de las conversiones no necesitan cambiar esta longitud, pero una salida puede hacerlo si es necesario; el valor establecido por la salida se devuelve a la aplicación en el parámetro **DataLength** de la llamada MQGET. Sin embargo, esta longitud no se puede cambiar si el mensaje que se está convirtiendo es un segmento que sólo contiene parte de un mensaje lógico. Esto se debe a que cambiar la longitud haría que los desplazamientos de segmentos posteriores en el mensaje lógico fueran incorrectos.

Tenga en cuenta que, si la salida desea cambiar la longitud de los datos, tenga en cuenta que el gestor de colas ya ha decidido si los datos del mensaje se ajustan al almacenamiento intermedio de la aplicación, basándose en la longitud de los datos *sin convertir*. Esta decisión determina si el mensaje se elimina de la cola (o se mueve el cursor para examinar, para una solicitud de examinar) y no se ve afectado por ningún cambio en la longitud de los datos causado por la conversión. Por este motivo, se recomienda que las salidas de conversión no provoquen un cambio en la longitud de los datos del mensaje de aplicación.

Si la conversión de caracteres implica un cambio de longitud, una serie se puede convertir en otra serie con la misma longitud en bytes, truncando los espacios en blanco finales o rellenando con espacios en blanco según sea necesario.

La salida no se invoca si el mensaje no contiene datos de mensaje de aplicación; por lo tanto, *DataLength* siempre es mayor que cero.

Es un campo de entrada/salida para la salida.

Encoding

Tipo: MQLONG

Codificación numérica necesaria para la aplicación.

Esta es la codificación numérica que necesita la aplicación que emite la llamada MQGET; consulte el campo *Encoding* en la estructura MQMD para obtener más detalles.

Si la conversión es satisfactoria, la salida la copia en el campo *Encoding* del descriptor de mensaje.

Es un campo de entrada para la salida.

ExitOptions

Tipo: MQLONG

Se trata de un campo reservado; su valor es 0.

ExitResponse

Tipo: MQLONG

Respuesta de la salida. Esto lo establece la salida para indicar el éxito o no de la conversión. Debe ser uno de los siguientes:

MQXDR_Correcto

La conversión ha sido satisfactoria.

Si la salida especifica este valor, el gestor de colas devuelve lo siguiente a la aplicación que ha emitido la llamada MQGET:

- El valor del campo *CompCode* en la salida de la salida
- El valor del campo *Reason* en la salida de la salida
- El valor del campo *DataLength* en la salida de la salida
- El contenido del almacenamiento intermedio de salida de la salida *OutBuffer*. El número de bytes devueltos es el menor del parámetro **OutBufferLength** de la salida y el valor del campo *DataLength* en la salida de la salida.

Si los campos *Encoding* y *CodedCharSetId* del parámetro de descriptor de mensaje de la salida *ambos* no se modifican, el gestor de colas devuelve:

- El valor de los campos *Encoding* y *CodedCharSetId* en la estructura MQDXP en *entrada* a la salida.

Si se ha cambiado uno o ambos campos *Encoding* y *CodedCharSetId* en el parámetro de descriptor de mensaje de la salida, el gestor de colas devuelve:

- El valor de los campos *Encoding* y *CodedCharSetId* en el parámetro de descriptor de mensaje de la salida en la salida de la salida

MQXDR_CONVERSION_FAILED

La conversión no ha sido satisfactoria.

Si la salida especifica este valor, el gestor de colas devuelve lo siguiente a la aplicación que ha emitido la llamada MQGET:

- El valor del campo *CompCode* en la salida de la salida
- El valor del campo *Reason* en la salida de la salida
- El valor del campo *DataLength* en *entrada* a la salida
- El contenido del almacenamiento intermedio de entrada de la salida *InBuffer*. El número de bytes devueltos lo proporciona el parámetro **InBufferLength**

Si la salida ha modificado *InBuffer*, los resultados no están definidos.

ExitResponse es un campo de salida de la salida.

Hconn

Tipo: MQHCONN

Se trata de un descriptor de conexión que se puede utilizar en la llamada MQXCNCV. Este descriptor de contexto no es necesariamente el mismo que el descriptor de contexto especificado por la aplicación que ha emitido la llamada MQGET.

pEntryPoints

Tipo: PMQIEP

La dirección de una estructura MQIEP a través de la cual se pueden realizar llamadas de tipo « MQI » y « DCI ».

Reason

Tipo: MQLONG

Código de razón que califica *CompCode*.

Cuando se invoca la salida, contiene el código de razón que se devuelve a la aplicación que ha emitido la llamada MQGET, si la salida elige no hacer nada. Entre los valores posibles se encuentran MQRC_TRUNCATED_MSG_ACCEPTED, que indica que el mensaje se ha truncado en orden de ajuste en el almacenamiento intermedio proporcionado por la aplicación, y MQRC_NOT_CONVERT, que indica que el mensaje requiere conversión pero que todavía no se ha realizado.

En la salida de la salida, este campo contiene la razón que debe devolverse a la aplicación en el parámetro **Reason** de la llamada MQGET; se recomienda lo siguiente:

- Si *Reason* tenía el valor MQRC_TRUNCATED_MSG_ACCEPTED en la entrada a la salida, los campos *Reason* y *CompCode* no deben alterarse, independientemente de si la conversión es satisfactoria o falla.

(Si el campo *CompCode* no es MQCC_OK, la aplicación que recupera el mensaje puede identificar un error de conversión comparando los valores *Encoding* y *CodedCharSetId* devueltos en el descriptor de mensaje con los valores solicitados; por el contrario, la aplicación no puede distinguir un mensaje truncado de un mensaje que encajaba en el almacenamiento intermedio. Por este motivo, MQRC_TRUNCATED_MSG_ACCEPTED debe devolverse con preferencia a cualquiera de las razones que indican un error de conversión.)

- Si *Reason* tenía cualquier otro valor en la entrada para la salida:
 - Si la conversión se realiza correctamente, *CompCode* debe establecerse en MQCC_OK y *Reason* debe establecerse en MQRC_NONE.
 - Si la conversión falla, o el mensaje se expande y se tiene que truncar para que quepa en el almacenamiento intermedio, *CompCode* debe establecerse en MQCC_WARNING (o dejarse sin modificar), y *Reason* establecerse en uno de los valores listados, para indicar la naturaleza de la anomalía.

Tenga en cuenta que si el mensaje después de la conversión es demasiado grande para el almacenamiento intermedio, sólo se debe truncar si la aplicación que ha emitido la llamada MQGET ha especificado la opción MQGMO_ACCEPT_TRUNCATED_MSG:

- Si ha especificado esta opción, se devuelve la razón MQRC_TRUNCATED_MSG_ACCEPTED.
- Si no ha especificado esa opción, el mensaje se devuelve sin convertir, con el código de razón MQRC_CONVERTED_MSG_TOO_BIG.

Los códigos de razón listados se recomiendan para que los utilice la salida para indicar la razón por la que ha fallado la conversión, pero la salida puede devolver otros valores del conjunto de códigos MQRC_* si se considera apropiado. Además, el rango de valores de MQRC_APPL_FIRST a MQRC_APPL_LAST se asignan para que los utilice la salida para indicar las condiciones en las que la salida desea comunicarse con la aplicación que emite la llamada MQGET.

Nota: Si el mensaje no se puede convertir correctamente, la salida debe devolver MQXDR_CONVERSION_FAILED en el campo *ExitResponse*, para que el gestor de colas devuelva el mensaje sin convertir. Esto es cierto independientemente del código de razón devuelto en el campo *Reason*.

MQRC_APPL_PRIMERO

(900, X'384 ') Valor más bajo para el código de razón definido por la aplicación.

MQRC_APPL_LAST

(999, X'3E7') Valor más alto para el código de motivo definido por la aplicación.

MQRC_CONVERTED_MSG_TOO_BIG

(2120, X'848') Los datos convertidos son demasiado grandes para el almacenamiento intermedio.

MQRC_NOT_CONVERTED

(2119, X'847') Los datos del mensaje no se han convertido.

MQRC_SOURCE_CCSID_ERROR

(2111, X'83F') El identificador de juego de caracteres codificado origen no es válido.

MQRC_SOURCE_DECIMAL_ENC_ERROR

(2113, X'841') Codificación de decimal empaquetado en el mensaje no reconocida.

MQRC_SOURCE_FLOAT_ENC_ERROR

(2114, X'842') Codificación de coma flotante en el mensaje no reconocida.

MQRC_SOURCE_INTEGER_ENC_ERROR

(2112, X'840') Codificación de entero de origen no reconocida.

MQRC_TARGET_CCSID_ERROR

(2115, X'843') El identificador de juego de caracteres codificado de destino no es válido.

MQRC_TARGET_DECIMAL_ENC_ERROR

(2117, X'845') Codificación de decimal empaquetado especificada por receptor no reconocida.

MQRC_TARGET_FLOAT_ENC_ERROR

(2118, X'846') Codificación de coma flotante especificada por receptor no reconocida.

MQRC_TARGET_INTEGER_ENC_ERROR

(2116, X'844') Codificación de entero de destino no reconocida.

MQRC_TRUNCATED_MSG_ACCEPTED

(2079, X'81F') Se ha devuelto un mensaje truncado (el proceso se ha completado).

Es un campo de entrada/salida para la salida.

StrucId

Tipo: MQCHAR4

Identificador de estructura.El valor debe ser:

MQDXP_STRUC_ID

Identificador de la estructura de parámetros de salida de conversión de datos.

Para el lenguaje de programación C, también se define la constante MQDXP_STRUC_ID_ARRAY; tiene el mismo valor que MQDXP_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Es un campo de entrada para la salida.

Version

Tipo: MQLONG

Número de versión de la estructura.El valor debe ser:

MQDXP_VERSION_1

Número de versión para la estructura de parámetros de salida de conversión de datos.

La constante siguiente especifica el número de versión de la versión actual:

MQDXP_VERSION_ACTUAL

Versión actual de la estructura de parámetros de salida de conversión de datos.

Nota: Cuando se introduce una nueva versión de esta estructura, el diseño de la parte existente no cambia. Por lo tanto, la salida debe comprobar que el campo *Version* es igual o mayor que la versión más baja que contiene los campos que la salida necesita utilizar.

Es un campo de entrada para la salida.

Declaración C

```
typedef struct tagMQDXP MQDXP;
struct tagMQDXP {
    MQCHAR4  StrucId;           /* Structure identifier */
    MQLONG   Version;          /* Structure version number */
    MQLONG   ExitOptions;      /* Reserved */
    MQLONG   AppOptions;       /* Application options */
    MQLONG   Encoding;         /* Numeric encoding required by
                               application */
    MQLONG   CodedCharSetId;    /* Character set required by application */
    MQLONG   DataLength;       /* Length in bytes of message data */
    MQLONG   CompCode;         /* Completion code */
    MQLONG   Reason;           /* Reason code qualifying CompCode */
    MQLONG   ExitResponse;     /* Response from exit */
    MQHCONN  Hconn;            /* Connection handle */
    PMQIEP   pEntryPoints;     /* Address of the MQIEP structure */
};
```

COBOL declaración (IBM i only)

```
**      MQDXP structure
10 MQDXP.
**      Structure identifier
15 MQDXP-STRUCID          PIC X(4).
**      Structure version number
15 MQDXP-VERSION         PIC S9(9) BINARY.
**      Reserved
15 MQDXP-EXITOPTIONS     PIC S9(9) BINARY.
**      Application options
15 MQDXP-APPOPTIONS     PIC S9(9) BINARY.
**      Numeric encoding required by application
15 MQDXP-ENCODING        PIC S9(9) BINARY.
**      Character set required by application
15 MQDXP-CODEDCHARSETID PIC S9(9) BINARY.
**      Length in bytes of message data
15 MQDXP-DATALENGTH      PIC S9(9) BINARY.
**      Completion code
15 MQDXP-COMPCODE        PIC S9(9) BINARY.
**      Reason code qualifying COMPCODE
15 MQDXP-REASON          PIC S9(9) BINARY.
**      Response from exit
15 MQDXP-EXITRESPONSE    PIC S9(9) BINARY.
**      Connection handle
15 MQDXP-HCONN           PIC S9(9) BINARY.
```

Declaración de ensamblador System/390

MQDXP	DSECT	
MQDXP_STRUCID	DS	CL4 Structure identifier
MQDXP_VERSION	DS	F Structure version number
MQDXP_EXITOPTIONS	DS	F Reserved
MQDXP_APPOPTIONS	DS	F Application options
MQDXP_ENCODING	DS	F Numeric encoding required by application
MQDXP_CODEDCHARSETID	DS	F Character set required by application
MQDXP_DATALENGTH	DS	F Length in bytes of message data
MQDXP_COMPCODE	DS	F Completion code
MQDXP_REASON	DS	F Reason code qualifying COMPCODE
MQDXP_EXITRESPONSE	DS	F Response from exit
MQDXP_HCONN	DS	F Connection handle
*		
MQDXP_LENGTH	EQU	*-MQDXP
	ORG	MQDXP
MQDXP_AREA	DS	CL(MQDXP_LENGTH)

MQXCNCV - Convertir caracteres

La llamada MQXCNCV convierte caracteres de un juego de caracteres a otro utilizando el lenguaje de programación C.

Esta llamada forma parte de la Interfaz de IBM MQ conversión de datos (DCI), que es una de las IBM MQ interfaces del marco.

Nota: La llamada se puede utilizar tanto desde entornos de salida de aplicaciones como de conversión de datos.

Sintaxis

MQXCNCV (*Hconn*, *Opciones*, *CCSID de origen*, *Longitud de origen*, *Búfer de origen*, *CCSID de destino*, *Longitud de destino*, *Búfer de destino*, *Longitud de datos*, *Código de compilación*, *Motivo*)

Parámetros

Hconn

Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas.

En una salida de conversión de datos, Hconn suele ser el identificador que se pasa a la salida de conversión de datos en el Hconn campo de la estructura MQDXP; este identificador no es necesariamente el mismo que el especificado por la aplicación que emitió la llamada MQGET.

 En IBM i, se puede especificar el siguiente valor especial para Hconn:

MQHC_DEF_HCONN

Manejador de conexión predeterminado.

Si ejecuta una aplicación CICS TS 3.2 o superior, asegúrese de que el programa de salida de conversión de caracteres, que invoca la llamada MQXCNCV, esté definido como OPENAPI. Esta definición evita el error 2018 MQRC_HCONN_ERROR provocado por una conexión incorrecta y permite que se complete la MQGET.

Opciones

Tipo: MQLONG - entrada

Opciones que controlan la acción de MQXCNCV.

Se pueden especificar cero o más de las opciones descritas. Para especificar más de una opción, suma los valores (sin añadir la misma constante más de una vez) o combínalos mediante la operación OR bit a bit (si el lenguaje de programación admite operaciones bit a bit).

Opción de conversión predeterminada : La siguiente opción controla el uso de la conversión de caracteres predeterminada:

MQDCC_CONVERSIÓN_POR_DEFAULT

Conversión predeterminada.

Esta opción especifica que se puede utilizar la conversión de caracteres predeterminada si uno o ambos de los conjuntos de caracteres especificados en la llamada no son compatibles. Esto permite al gestor de colas utilizar un juego de caracteres predeterminado especificado por la instalación que se aproxima al juego de caracteres especificado al convertir la cadena.

Nota: El resultado de utilizar un conjunto de caracteres aproximado para convertir la cadena es que algunos caracteres pueden convertirse incorrectamente. Esto se puede evitar utilizando en la cadena solo caracteres que sean comunes tanto al juego de caracteres especificado como al juego de caracteres predeterminado.

Los conjuntos de caracteres predeterminados se definen mediante una opción de configuración cuando se instala o se reinicia el gestor de colas.

Si no se especifica MQDCC_DEFAULT_CONVERSION, el gestor de colas utiliza solo los conjuntos de caracteres especificados para convertir la cadena, y la llamada falla si uno o ambos conjuntos de caracteres no son compatibles.

Esta opción es compatible con los siguientes entornos:

-  AIX

-  IBM i
-  Linux
-  Windows

Opción de relleno : La siguiente opción permite al gestor de colas rellenar la cadena convertida con espacios en blanco o descartar caracteres finales insignificantes, con el fin de que la cadena convertida se ajuste al búfer de destino:

MQDCC_LLENAR_BÚFER_DESTINO

Llenar el búfer de destino.

Esta opción solicita que la conversión se realice de tal manera que el búfer de destino se llene por completo:

- Si la cadena se contrae al convertirse, se añaden espacios en blanco al final para llenar el búfer de destino.
- Si la cadena se expande al convertirse, los caracteres finales que no son significativos se descartan para que la cadena convertida quepa en el búfer de destino. Si esto se puede realizar correctamente, la llamada se completa con MQCC_OK y el código de motivo MQRC_NONE.

Si hay muy pocos caracteres finales insignificantes, se coloca en el búfer de destino toda la cadena que quepa y la llamada se completa con MQCC_WARNING y el código de motivo MQRC_CONVERTED_MSG_TOO_BIG.

Los caracteres insignificantes son:

- Espacios en blanco al final
- Caracteres que siguen al primer carácter nulo de la cadena (pero excluyendo el primer carácter nulo en sí mismo)
- Si la cadena, TargetCCSID, y TargetLength son tales que el búfer de destino no se puede establecer completamente con caracteres válidos, la llamada falla con MQCC_FAILED y el código de motivo MQRC_TARGET_LENGTH_ERROR. Esto puede ocurrir cuando TargetCCSID es un conjunto de caracteres DBCS puro (como UTF-16), pero TargetLength especifica una longitud que es un número impar de bytes.
- TargetLength puede ser menor o mayor que SourceLength. Al regresar de MQXCNCV, DataLength tiene el mismo valor que TargetLength.

Si no se especifica esta opción:

- La cadena puede contraerse o expandirse dentro del búfer de destino según sea necesario. Los caracteres finales insignificantes no se añaden ni se descartan.

Si la cadena convertida cabe en el búfer de destino, la llamada se completa con MQCC_OK y el código de motivo MQRC_NONE.

Si la cadena convertida es demasiado grande para el búfer de destino, se coloca en él la parte de la cadena que cabe y la llamada se completa con MQCC_WARNING y el código de motivo MQRC_CONVERTED_MSG_TOO_BIG. Tenga en cuenta que en este caso se pueden devolver menos de TargetLength bytes.

- TargetLength puede ser menor o mayor que SourceLength. Al regresar de MQXCNCV, DataLength es menor o igual que TargetLength.

Esta opción es compatible con los siguientes entornos:

-  AIX
-  IBM i
-  Linux
-  Windows

Opciones de codificación : Las opciones descritas se pueden utilizar para especificar las codificaciones enteras de las cadenas de origen y destino. La codificación pertinente solo se utiliza cuando el identificador del juego de caracteres correspondiente indica que la representación del juego de caracteres en la memoria principal depende de la codificación utilizada para los enteros binarios. Esto solo afecta a determinados conjuntos de caracteres multibyte (por ejemplo, conjuntos de caracteres UTF-16).

La codificación se ignora si el juego de caracteres es un juego de caracteres de un solo byte (SBCS) o un juego de caracteres multibyte con representación en la memoria principal que no depende de la codificación entera.

Solo se debe especificar uno de los valores MQDCC_SOURCE_*, combinado con uno de los valores MQDCC_TARGET_*:

MQDCC_FUENTE_ENC_NATIVA

La codificación de origen es la predeterminada para el entorno y el lenguaje de programación.

MQDCC_FUENTE_ENC_NORMAL

La codificación de la fuente es normal.

MQDCC_FUENTE_ENCODING_INVERTIDA

La codificación de la fuente se invierte.

MQDCC_FUENTE_ENC_INDETERMINADA

La codificación de origen no está definida.

MQDCC_TARGET_ENC_NATIVE

La codificación de destino es la predeterminada para el entorno y el lenguaje de programación.

MQDCC_OBJETIVO_ENC_NORMAL

La codificación del objetivo es normal.

MQDCC_OBJETIVO_ENC_INVERTIDO

La codificación del objetivo se invierte.

MQDCC_TARGET_ENC_INDETERMINADO

La codificación de destino no está definida.

Los valores de codificación definidos anteriormente se pueden añadir directamente al campo Options . Sin embargo, si la codificación de origen o destino se obtiene del Encoding campo en el MQMD u otra estructura, se debe realizar el siguiente procesamiento:

1. La codificación entera debe extraerse del Encoding campo eliminando las codificaciones flotantes y decimales empaquetadas; consulte [“Análisis de codificaciones” en la página 910](#) para obtener más información sobre cómo hacerlo.
2. La codificación entera resultante del paso 1 debe multiplicarse por el factor adecuado antes de añadirse al Options campo. Estos factores son:
 - MQDCC_SOURCE_ENC_FACTOR para la codificación de origen
 - MQDCC_TARGET_ENC_FACTOR para la codificación de destino

El siguiente código de ejemplo ilustra cómo se podría programar esto en el lenguaje de programación C:

```
Options = (MsgDesc.Encoding & MQENC_INTEGER_MASK)
          * MQDCC_SOURCE_ENC_FACTOR
        + (DataConvExitParms.Encoding & MQENC_INTEGER_MASK)
          * MQDCC_TARGET_ENC_FACTOR;
```

Si no se especifica, las opciones de codificación se establecen por defecto en indefinido (MQDCC_*_ENC_UNDEFINED). En la mayoría de los casos, esto no afecta a la finalización correcta de la llamada MQXCNVC. Sin embargo, si el juego de caracteres correspondiente es un juego de caracteres multibyte con una representación que depende de la codificación (por ejemplo, un juego de caracteres UTF-16), la llamada falla con el código de motivo MQRC_SOURCE_INTEGER_ENC_ERROR o MQRC_TARGET_INTEGER_ENC_ERROR, según corresponda.

Las opciones de codificación son compatibles con los siguientes entornos:

-  AIX
-  IBM i
-  Linux
-  Windows
-  z/OS

Opción predeterminada : Si no se especifica ninguna de las opciones descritas anteriormente, se puede utilizar la siguiente opción:

MQDCC_NINGUNO

No se ha especificado ninguna opción.

MQDCC_NONE se define para facilitar la documentación del programa. No está previsto que esta opción se utilice con ninguna otra, pero como su valor es cero, dicho uso no puede detectarse.

SourceCCSID

Tipo: MQLONG - entrada

Este es el identificador del conjunto de caracteres codificados de la cadena de entrada en `SourceBuffer`.

SourceLength

Tipo: MQLONG - entrada

Es la longitud en bytes de la cadena de entrada en `SourceBuffer` ; debe ser cero o mayor.

SourceBuffer

Tipo: MQCHAR x `SourceLength` - entrada

Este es el búfer que contiene la cadena que se va a convertir de un juego de caracteres a otro.

TargetCCSID

Tipo: MQLONG - entrada

Este es el identificador codificado del juego de caracteres al que `SourceBuffer` se va a convertir.

TargetLength

Tipo: MQLONG - entrada

Es la longitud en bytes del búfer de salida `TargetBuffer` ; debe ser cero o mayor. Puede ser menor o mayor que `SourceLength`.

TargetBuffer

Tipo: MQCHAR x `TargetLength` - salida

Esta es la cadena después de haber sido convertida al conjunto de caracteres definido por `TargetCCSID`. La cadena convertida puede ser más corta o más larga que la cadena sin convertir. El

DataLength parámetro indica el número de bytes válidos devueltos.

DataLength

Tipo: MQLONG - salida

Esta es la longitud de la cadena devuelta en el búfer de salida `TargetBuffer`. La cadena convertida puede ser más corta o más larga que la cadena sin convertir.

CompCode

Tipo: MQLONG - salida

Es uno de los siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_WARNING

Aviso (finalización parcial).

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

Código de motivo calificativo CompCode.

Si CompCode es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si CompCode es MQCC_WARNING:

MQRC_CONVERTED_MSG_TOO_BIG

(2120, X'848') Los datos convertidos son demasiado grandes para el almacenamiento intermedio.

Si CompCode es MQCC_FAILED:

MQRC_DATA_LENGTH_ERROR

(2010, X'7DA') El parámetro longitud de datos no es válido.

MQRC_DBCS_ERROR

(2150, X'866') La serie DBCS no es válida.

MQRC_HCONN_ERROR

(2018, X'7E2') El manejador de conexión no es válido.

MQRC_OPTIONS_ERROR

(2046, X'7FE') Las opciones no son válidas o no son coherentes.

MQRC_RESOURCE_PROBLEM

(2102, X'836') No hay disponibles suficientes recursos del sistema.

MQRC_SOURCE_BUFFER_ERROR

(2145, X'861') El parámetro de almacenamiento intermedio de origen no es válido.

MQRC_SOURCE_CCSDID_ERROR

(2111, X'83F') El identificador de juego de caracteres codificado origen no es válido.

MQRC_SOURCE_INTEGER_ENC_ERROR

(2112, X'840') Codificación de entero de origen no reconocida.

MQRC_SOURCE_LENGTH_ERROR

(2143, X'85F') Parámetro de longitud de origen no válido.

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_TARGET_BUFFER_ERROR

(2146, X'862') Parámetro de almacenamiento intermedio de destino no válido.

MQRC_TARGET_CCSDID_ERROR

(2115, X'843') El identificador de juego de caracteres codificado de destino no es válido.

MQRC_TARGET_INTEGER_ENC_ERROR

(2116, X'844') Codificación de entero de destino no reconocida.

ERROR_DE_LONGITUD_DE_DESTINO_MQRC

(2144, X'860') Parámetro de longitud objetivo no válido.

MQRC_UNEXPECTED_ERROR

(2195, X'893') Se ha producido un error inesperado.

Para obtener información detallada sobre estos códigos, consulte [Mensajes y códigos de motivo](#).

Invocación en C

```
MQXCNCV (Hconn, Options, SourceCCSID, SourceLength, SourceBuffer,  
         TargetCCSID, TargetLength, TargetBuffer, &DataLength,  
         &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQHCONN  Hconn;           /* Connection handle */  
MQLONG   Options;         /* Options that control the action of  
                           MQXCNCV */  
MQLONG   SourceCCSID;     /* Coded character set identifier of string  
                           before conversion */  
MQLONG   SourceLength;    /* Length of string before conversion */  
MQCHAR   SourceBuffer[n]; /* String to be converted */  
MQLONG   TargetCCSID;     /* Coded character set identifier of string  
                           after conversion */  
MQLONG   TargetLength;    /* Length of output buffer */  
MQCHAR   TargetBuffer[n]; /* String after conversion */  
MQLONG   DataLength;      /* Length of output string */  
MQLONG   CompCode;        /* Completion code */  
MQLONG   Reason;          /* Reason code qualifying CompCode */
```

COBOL declaración (IBM i solo)



```
CALL 'MQXCNCV' USING HCONN, OPTIONS, SOURCECCSID, SOURCELENGTH,  
                     SOURCEBUFFER, TARGETCCSID, TARGETLENGTH,  
                     TARGETBUFFER, DATALENGTH, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
** Connection handle  
01 HCONN          PIC S9(9) BINARY.  
** Options that control the action of MQXCNCV  
01 OPTIONS        PIC S9(9) BINARY.  
** Coded character set identifier of string before conversion  
01 SOURCECCSID    PIC S9(9) BINARY.  
** Length of string before conversion  
01 SOURCELENGTH   PIC S9(9) BINARY.  
** String to be converted  
01 SOURCEBUFFER    PIC X(n).  
** Coded character set identifier of string after conversion  
01 TARGETCCSID    PIC S9(9) BINARY.  
** Length of output buffer  
01 TARGETLENGTH   PIC S9(9) BINARY.  
** String after conversion  
01 TARGETBUFFER    PIC X(n).  
** Length of output string  
01 DATALENGTH    PIC S9(9) BINARY.  
** Completion code  
01 COMPCODE        PIC S9(9) BINARY.  
** Reason code qualifying COMPCODE  
01 REASON          PIC S9(9) BINARY.
```

S/390 declaración del ensamblador

```
CALL MQXCNCV,(HCONN,OPTIONS,SOURCECCSID,SOURCELENGTH,      X  
              SOURCEBUFFER,TARGETCCSID,TARGETLENGTH,TARGETBUFFER, X  
              DATALENGTH,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

```
HCONN      DS F      Connection handle  
OPTIONS    DS F      Options that control the action of MQXCNCV  
SOURCECCSID DS F      Coded character set identifier of string before  
* conversion  
SOURCELENGTH DS F      Length of string before conversion  
SOURCEBUFFER DS CL(n) String to be converted  
TARGETCCSID DS F      Coded character set identifier of string after  
* conversion
```

TARGETLENGTH	DS	F	Length of output buffer
TARGETBUFFER	DS	CL(n)	String after conversion
DATALength	DS	F	Length of output string
COMPCODE	DS	F	Completion code
REASON	DS	F	Reason code qualifying COMPCODE

MQ_DATA_CONV_EXIT - Salida de conversión de datos

La llamada MQ_DATA_CONV_EXIT describe los parámetros que se pasan a la salida de conversión de datos.

El gestor de colas no proporciona ningún punto de entrada denominado MQ_DATA_CONV_EXIT (véase la nota de uso [11](#)).

Esta definición forma parte de la Interfaz de IBM MQ conversión de datos (DCI), que es una de las IBM MQ interfaces del marco.

Sintaxis

MQ_DATA_CONV_EXIT (*DataConvExitParms*, *MsgDesc*, *InBufferLength*, *InBuffer*, *OutBufferLength*, *OutBuffer*)

Parámetros

DataConvExitParms

Tipo: MQDXP - entrada/salida

Esta estructura contiene información relacionada con la invocación de la salida. La salida establece información en esta estructura para indicar el resultado de la conversión. Consulte “MQDXP-Parámetro de salida de conversión de datos” en la [página 921](#) para obtener más información sobre los campos de esta estructura.

MsgDesc

Tipo: MQMD - entrada/salida

En la entrada a la salida, este es el descriptor de mensaje asociado con los datos del mensaje pasados a la salida en el **InBuffer** parámetro.

Nota: El **MsgDesc** parámetro pasado a la salida es siempre la versión más reciente de MQMD compatible con el gestor de colas que invoca la salida. Si se pretende que la salida sea portátil entre diferentes entornos, la salida comprobará el **Version** campo en **MsgDesc** para verificar que los campos a los que la salida necesita acceder están presentes en la estructura.

En los siguientes entornos, se pasa a la salida un MQMD e version-2 :

-  AIX
-  IBM i
-  Linux
-  Windows

En todos los demás entornos que admiten la salida de conversión de datos, se pasa a la salida un MQMD e version-1.

En la salida, la salida cambiará los campos **Encoding CodedCharSetId** y a los valores solicitados por la aplicación, si la conversión se ha realizado correctamente; estos cambios se reflejan en la aplicación. Cualquier otro cambio que la salida realice en la estructura se ignora; no se refleja en la aplicación.

Si la salida devuelve MQXDR_OK en el **ExitResponse** campo de la estructura MQDXP, pero no cambia los campos **Encoding CodedCharSetId** o del descriptor de mensajes, el gestor de colas devuelve para esos campos los valores que tenían los campos correspondientes de la estructura MQDXP al entrar en la salida.

InBufferLength

Tipo: MQLONG - entrada

Longitud en bytes de InBuffer.

Esta es la longitud del búfer de entrada InBuffer y especifica el número de bytes que procesará la salida. InBufferLength es el menor entre la longitud de los datos del mensaje antes de la conversión y la longitud del búfer proporcionado por la aplicación en la llamada MQGET.

El valor siempre es mayor que cero.

InBuffer

Tipo: MQBYTEInBufferLength - entrada

Búfer que contiene el mensaje sin convertir.

Contiene los datos del mensaje antes de la conversión. Si la salida no puede convertir los datos, el gestor de colas devuelve el contenido de este búfer a la aplicación una vez que la salida ha finalizado.

Nota: La salida no debe alterarse InBuffer ; si se altera este parámetro, los resultados son indefinidos.

En el lenguaje de programación C, este parámetro se define como un puntero a void.

OutBufferLength

Tipo: MQLONG - entrada

Longitud en bytes de OutBuffer.

Esta es la longitud del búfer de salida OutBuffer y es igual a la longitud del búfer proporcionado por la aplicación en la llamada MQGET.

El valor siempre es mayor que cero.

OutBuffer

Tipo: MQBYTEOutBufferLength - salida

Búfer que contiene el mensaje convertido.

En la salida de la salida, si la conversión se ha realizado correctamente (tal y como indica el valor MQXDR_OK en el ExitResponse campo del **DataConvExitParms** parámetro), OutBuffer contiene los datos del mensaje que se entregarán a la aplicación, en la representación solicitada. Si la conversión no se ha realizado correctamente, se ignorarán los cambios que la salida haya realizado en este búfer.

En el lenguaje de programación C, este parámetro se define como un puntero a void.

Notas de uso

1. Una salida de conversión de datos es una salida escrita por el usuario que recibe el control durante el procesamiento de una llamada MQGET. La función que realiza la salida de conversión de datos la define el proveedor de la salida; sin embargo, la salida debe ajustarse a las reglas descritas aquí y en la estructura de parámetros asociada MQDXP.

Los lenguajes de programación que se pueden utilizar para una salida de conversión de datos vienen determinados por el entorno.

2. La salida solo se invoca si todas las siguientes afirmaciones son ciertas:

- La opción MQGMO_CONVERT se especifica en la llamada MQGET
- El Format campo del descriptor del mensaje no es MQFMT_NONE
- El mensaje no tiene la representación requerida; es decir, uno o ambos de los valores CodedCharSetId y del mensaje Encoding son diferentes del valor especificado por la aplicación en el descriptor de mensajes proporcionado en la llamada MQGET
- El gestor de colas aún no ha realizado la conversión correctamente
- La longitud del búfer de la aplicación es mayor que cero

- La longitud de los datos del mensaje es mayor que cero
 - El código de motivo hasta ahora durante la operación MQGET es MQRC_NONE o MQRC_TRUNCATED_MSG_ACCEPTED
3. Cuando se escriba una salida, considere codificarla de manera que permita convertir los mensajes que hayan sido truncados. Los mensajes truncados pueden surgir de las siguientes maneras:
- La aplicación receptora proporciona un búfer más pequeño que el mensaje, pero especifica la opción MQGMO_ACCEPT_TRUNCATED_MSG en la llamada MQGET.
- En este caso, el Reason campo del **DataConvExitParms** parámetro en la entrada a la salida tiene el valor MQRC_TRUNCATED_MSG_ACCEPTED.
- El remitente del mensaje lo truncó antes de enviarlo. Esto puede ocurrir, por ejemplo, con los mensajes de informe (véase [“Conversión de mensajes de informe”](#) en la [página 921](#) para obtener más detalles).
- En este caso, el Reason campo del **DataConvExitParms** parámetro en la entrada a la salida tiene el valor MQRC_NONE (si la aplicación receptora proporcionó un búfer lo suficientemente grande para el mensaje).
- Por lo tanto, el valor del Reason campo en la entrada a la salida no siempre puede utilizarse para decidir si el mensaje ha sido truncado.
- La característica distintiva de un mensaje truncado es que la longitud proporcionada a la salida en el **InBufferLength** parámetro es menor que la longitud implícita en el nombre del formato contenido en el Format campo del descriptor del mensaje. Por lo tanto, la salida debe comprobar el valor de InBufferLength antes de intentar convertir cualquiera de los datos; la salida no debe dar por sentado que se ha proporcionado la cantidad completa de datos que implica el nombre del formato.
- Si la salida no se ha escrito para convertir mensajes truncados y InBufferLength es menor que el valor esperado, la salida devolverá MQXDR_CONVERSION_FAILED en el ExitResponse campo del **DataConvExitParms** parámetro, con los campos CompCode Reason y establecidos en MQCC_WARNING y MQRC_FORMAT_ERROR.
- Si la salida se ha escrito para convertir mensajes truncados, la salida convertirá tantos datos como sea posible (véase la siguiente nota de uso), teniendo cuidado de no intentar examinar o convertir datos más allá del final de InBuffer. Si la conversión se completa correctamente, la salida dejará el Reason campo del **DataConvExitParms** parámetro sin cambios. Devuelve MQRC_TRUNCATED_MSG_ACCEPTED si el mensaje fue truncado por el gestor de colas del receptor, y MQRC_NONE si el mensaje fue truncado por el remitente del mensaje.
- También es posible que un mensaje se expanda durante la conversión, hasta el punto de que sea más grande que OutBuffer. En este caso, la salida debe decidir si truncar el mensaje; el AppOptions campo del **DataConvExitParms** parámetro indica si la aplicación receptora especificó la opción MQGMO_ACCEPT_TRUNCATED_MSG.
4. Por lo general, todos los datos del mensaje proporcionados a la salida en InBuffer se convierten, o ninguno de ellos. Sin embargo, se produce una excepción a esto si el mensaje se trunca, ya sea antes de la conversión o durante la misma; en este caso, puede haber un elemento incompleto al final del búfer (por ejemplo: 1 byte de un carácter de doble byte o 3 bytes de un entero de 4 bytes). En esta situación, considere omitir el elemento incompleto y establezca los bytes no utilizados en OutBuffer como nulos. Sin embargo, los elementos completos o los caracteres dentro de una matriz o cadena deben convertirse.
5. Cuando se necesita una salida por primera vez, el gestor de colas intenta cargar un objeto que tenga el mismo nombre que el formato (aparte de las extensiones). El objeto cargado debe contener la salida que procesa los mensajes con ese nombre de formato. Considere la posibilidad de que el nombre de la salida y el nombre del objeto que contiene la salida sean idénticos, aunque no todos los entornos lo requieren.
6. Se carga una nueva copia de la salida cuando una aplicación intenta recuperar el primer mensaje que utiliza esa Format desde que la aplicación se conectó al gestor de colas. Para CICS o IMS aplicaciones, esto significa cuando el CICS o IMS subsistema conectado al gestor de colas. También se puede cargar una nueva copia en otros momentos, si el gestor de colas ha descartado una copia

cargada anteriormente. Por este motivo, una salida no debe intentar utilizar almacenamiento estático para comunicar información de una invocación de la salida a la siguiente, ya que la salida puede descargarse entre ambas invocaciones.

7. Si existe una salida proporcionada por el usuario con el mismo nombre que uno de los formatos integrados compatibles con el gestor de colas, la salida proporcionada por el usuario no sustituye a la rutina de conversión integrada. Las únicas circunstancias en las que se invoca dicha salida son:
 - Si la rutina de conversión integrada no puede gestionar las conversiones hacia o desde o `EncodingCodedCharSetId` implicadas, o
 - Si la rutina de conversión integrada no ha podido convertir los datos (por ejemplo, porque hay un campo o carácter que no se puede convertir).
8. El alcance de la salida depende del entorno. **Format** Los nombres deben elegirse de manera que se minimice el riesgo de conflictos con otros formatos. Considere comenzar con caracteres que identifiquen la aplicación que define el nombre del formato.
9. La salida de conversión de datos se ejecuta en un entorno similar al del programa que emitió la llamada `MQGET`; el entorno incluye el espacio de direcciones y el perfil de usuario (cuando corresponda). El programa podría ser un agente de canal de mensajes que envía mensajes a un gestor de colas de destino que no admite la conversión de mensajes. La salida no puede comprometer la integridad del gestor de colas, ya que no se ejecuta en el entorno del gestor de colas.
10. La única llamada de « MQI » que puede utilizar la salida es `MQXCNV`; si se intenta utilizar otras llamadas de « MQI », la operación fallará con el código de motivo `MQRC_CALL_IN_PROGRESS` o se producirán otros errores impredecibles.
11. El gestor de colas no proporciona ningún punto de entrada denominado `MQ_DATA_CONV_EXIT`. Sin embargo, se proporciona un `typedef` para el nombre `MQ_DATA_CONV_EXIT` en el lenguaje de programación C, y este se puede utilizar para declarar la salida escrita por el usuario, con el fin de garantizar que los parámetros sean correctos. El nombre de la salida debe ser el mismo que el nombre del formato (el nombre que figura en el **Format** campo de `MQMD`), aunque esto no es obligatorio en todos los entornos.

El siguiente ejemplo ilustra cómo se puede declarar la salida `MYFORMAT` que procesa el formato en el lenguaje de programación C:

```
#include "cmqc.h"
#include "cmqxc.h"

MQ_DATA_CONV_EXIT MYFORMAT;

void MQENTRY MYFORMAT(
    PMQDXP    pDataConvExitParms, /* Data-conversion exit parameter
                                   block */
    PMQMD     pMsgDesc,           /* Message descriptor */
    MQLONG    InBufferLength,     /* Length in bytes of InBuffer */
    PMQVOID   pInBuffer,         /* Buffer containing the unconverted
                                   message */
    MQLONG    OutBufferLength,    /* Length in bytes of OutBuffer */
    PMQVOID   pOutBuffer)        /* Buffer containing the converted
                                   message */
{
    /* C language statements to convert message */
}
```

12.  Si también está en vigor una salida de cruce de API `z/OS`, esta se invoca después de la salida de conversión de datos.

Invocación en C

```
exitname (&DataConvExitParms, &MsgDesc, InBufferLength,
          InBuffer, OutBufferLength, OutBuffer);
```

Los parámetros pasados a la salida se declaran de la siguiente manera:

```

MQDXP   DataConvExitParms; /* Data-conversion exit parameter block */
MQMD     MsgDesc;          /* Message descriptor */
MQLONG   InBufferLength;   /* Length in bytes of InBuffer */
MQBYTE   InBuffer[n];      /* Buffer containing the unconverted
                             message */
MQLONG   OutBufferLength;  /* Length in bytes of OutBuffer */
MQBYTE   OutBuffer[n];     /* Buffer containing the converted
                             message */

```

COBOL declaración (IBM i solo)

IBM i

```

CALL 'exitname' USING DATACONVEXITPARMS, MSGDESC, INBUFFERLENGTH,
                      INBUFFER, OUTBUFFERLENGTH, OUTBUFFER.

```

Los parámetros pasados a la salida se declaran de la siguiente manera:

```

** Data-conversion exit parameter block
01 DATACONVEXITPARMS.
   COPY CMQDXPV.
** Message descriptor
01 MSGDESC.
   COPY CMQMDV.
** Length in bytes of INBUFFER
01 INBUFFERLENGTH PIC S9(9) BINARY.
** Buffer containing the unconverted message
01 INBUFFER PIC X(n).
** Length in bytes of OUTBUFFER
01 OUTBUFFERLENGTH PIC S9(9) BINARY.
** Buffer containing the converted message
01 OUTBUFFER PIC X(n).

```

System/390 declaración del ensamblador

```

CALL EXITNAME, (DATACONVEXITPARMS, MSGDESC, INBUFFERLENGTH,      X
                INBUFFER, OUTBUFFERLENGTH, OUTBUFFER)

```

Los parámetros pasados a la salida se declaran de la siguiente manera:

DATACONVEXITPARMS	CMQDXPA	,	Data-conversion exit parameter block
MSGDESC	CMQMDA	,	Message descriptor
INBUFFERLENGTH	DS	F	Length in bytes of INBUFFER
INBUFFER	DS	CL(n)	Buffer containing the unconverted message
*			
OUTBUFFERLENGTH	DS	F	Length in bytes of OUTBUFFER
OUTBUFFER	DS	CL(n)	Buffer containing the converted message
*			

Propiedades especificadas como elementos MQRFH2

Las propiedades de descriptor que no son de mensaje se pueden especificar como elementos en las carpetas de cabecera MQRFH2. Visión general de los elementos MQRFH2 que se especifican como propiedades.

Esto mantiene la compatibilidad con las versiones anteriores de los clientes IBM MQ JMS y XMS. En esta sección se describe cómo especificar propiedades en las cabeceras MQRFH2.

Para utilizar elementos MQRFH2 como propiedades, especifique los elementos tal como se describe en [Utilización de IBM MQ clases for Java](#). Esta información complementa la información descrita en [“MQRFH2 – Reglas y formato de la cabecera 2”](#) en la página 509.

Correlación de tipos de datos de propiedad con tipos de datos MQRFH2

Este tema proporciona información sobre los tipos de propiedad de mensaje correlacionados con sus tipos de datos MQRFH2 correspondientes.

Tabla 671. Tipos de datos MQRFH2 soportados	
Tipo de propiedad de mensaje	Tipo de datos MQRFH2
MQBYTE []	bin.hex
MQBOOL	boolean
MQINT8	i1
MQINT16	i2
MQINT32	i4
MQINT64	i8
MQFLOAT32	r4
MQFLOAT64	r8
MQCHAR []	serie

Se supone que cualquier elemento sin un tipo de datos es de tipo "serie".

Un tipo de datos MQRFH2 de `int`, que significa un entero de tamaño no especificado, se trata como si fuera un `i8`.

Un valor nulo se indica mediante el atributo de elemento `xsi:nil='true'`. No utilice el atributo `xsi:nil='false'` para valores no nulos.

Por ejemplo, la propiedad siguiente tiene un valor nulo:

```
<NullProperty xsi:nil='true'></NullProperty>
```

Una propiedad de serie de bytes o caracteres puede tener un valor vacío. Esto se representa mediante un elemento MQRFH2 con un valor de elemento de longitud cero.

Por ejemplo, la propiedad siguiente tiene un valor vacío:

```
<EmptyProperty></EmptyProperty>
```

Carpetas MQRFH2 soportadas

Visión general del uso de campos de descriptor de mensaje como propiedades.

Las carpetas `<jms>`, `<mcd>`, `<mqext>` y `<usr>` se describen en [La cabecera MQRFH2 y JMS](#). La carpeta `<usr>` se utiliza para transportar las propiedades definidas por la aplicación JMS que están asociadas con un mensaje. Los grupos no están permitidos en la carpeta `<usr>`.

[La cabecera MQRFH2 y JMS](#) dan soporte a las siguientes carpetas adicionales:

- `<mq>`

Esta carpeta se utiliza y se reserva para las propiedades definidas por MQ que utiliza IBM MQ.

- `<mq_usr>`

Esta carpeta se puede utilizar para transportar cualquier propiedad definida por la aplicación que no esté expuesta como propiedades definidas por el usuario JMS, ya que es posible que las propiedades no cumplan los requisitos de una propiedad JMS. Esta carpeta puede contener grupos que la carpeta `<usr>` no puede.

- Cualquier carpeta marcada con el atributo `content='properties'`.

Una carpeta de este tipo es equivalente a la carpeta `<mq_usr>` en el contenido.

- `<mmps>`

Esta carpeta se utiliza para las propiedades de publicación/suscripción de IBM MQ.

IBM MQ También es compatible con las siguientes carpetas que ya están en uso por WAS /SIB:

- `<sib>`

Esta carpeta se utiliza y se reserva para las propiedades de los mensajes del sistema SIB de WAS que no se exponen como JMS propiedades, o que están asignadas a propiedades JMS_IBM_*, pero que se exponen a las aplicaciones SIB de WAS; entre ellas se incluyen las propiedades de las rutas de enrutamiento de ida y vuelta.

Al menos algunos no se pueden exponer como propiedades JMS , porque son matrices de bytes. Si la aplicación añade propiedades a esta carpeta, el valor se ignora o se elimina.

- `<sib_usr>`

Esta carpeta se utiliza y se reserva para las propiedades de los mensajes de usuario de WAS /SIB que no pueden mostrarse como JMS propiedades de usuario porque no son de tipos compatibles; se muestran a las aplicaciones de WAS /SIB.

Estas son propiedades de usuario, que puede obtener o establecer a través de la interfaz SIMessage, pero el contenido de la matriz de bytes se correlaciona con el valor de propiedad necesario.

Si la aplicación IBM MQ escribe un elemento bin.hex arbitrario en la carpeta, la aplicación probablemente recibirá un `IOException`, ya que no tiene el formato que se espera restaurar. Si añade algo que no sea un elemento bin.hex , recibirá un `ClassCastException`.

No intente poner propiedades a disposición de WAS /SIB utilizando esta carpeta; utilice en su lugar la `<usr>` carpeta destinada a tal fin.

- `<sib_context>`

Esta carpeta se utiliza para las propiedades de los mensajes del sistema SIB (WAS) que no están disponibles para las aplicaciones de usuario SIB (WAS) ni como JMS propiedades. Estas incluyen propiedades de seguridad y transaccionales que se utilizan para servicios web y similares.

La aplicación no debe añadir propiedades a esta carpeta.

- `<mqema>`

Esta carpeta fue utilizada por WAS /SIB en lugar de la `<mqext>` carpeta.

Los nombres de carpeta MQRFH2 distinguen entre mayúsculas y minúsculas.

Las carpetas siguientes están reservadas, en cualquier combinación de caracteres en minúsculas o mayúsculas:

- Cualquier carpeta con el prefijo mq o wmq ; reservado para su uso por parte de IBM MQ.
- Cualquier carpeta cuyo nombre comience por sib ; está reservada para su uso por parte de WAS /SIB.
- Carpetas `<Root>` y `<Body>` ; reservadas pero no utilizadas.

Las carpetas siguientes no se reconocen como que contienen propiedades de mensaje:

- `<psc>`

Lo utiliza IBM Integration Bus para transmitir mensajes de mandatos de publicación/suscripción al intermediario.

- `<pscr>`

Utilizado por IBM Integration Bus para contener información del intermediario, en respuesta a mensajes de mandato de publicación/suscripción.

- Cualquier carpeta no definida por IBM, que no esté marcada con el atributo `content='properties'` .

No especifique `content='properties'` en las carpetas `<psc>` o `<pscr>` . Si lo hace, estas carpetas se tratan como propiedades y es probable que IBM Integration Bus deje de funcionar como se esperaba.

Si la aplicación está creando mensajes con propiedades, en las cabeceras MQRFH2 que se van a reconocer como una cabecera MQRFH2 que contiene propiedades, la cabecera debe estar en la lista de cabeceras que se pueden encadenar en la cabecera del mensaje.

La MQRFH2 puede ir precedida de cualquier número de cabeceras estándar MQH, o una MQCIH, una MQDLH, una MQIIH, una MQTM, una MQTMC2o una MQXQH. Una serie o un MQCFH finaliza el análisis porque no se pueden encadenar.

Es posible que un mensaje contenga varias cabeceras MQRFH2 que transporten todas las propiedades del mensaje. Las carpetas con el mismo nombre pueden coexistir en diferentes encabezados, salvo que se restrinja de otro modo, por ejemplo, mediante WAS /SIB. Las carpetas se tratan como una carpeta lógica, si todas están en cabeceras significativas.

Mientras que las carpetas de las cabeceras significativas no se pueden fusionar con esas carpetas en cabeceras no significativas, las carpetas con el mismo nombre dentro de las cabeceras significativas se pueden fusionar, eliminando las propiedades en conflicto. Las aplicaciones no deben depender del diseño de las propiedades dentro de su mensaje.

Los grupos MQRFH2 se analizan para ver las propiedades de las carpetas definidas por el usuario, es decir, no las carpetas <wmq>, <jms>, <mcd>, <usr>, <mqext>, <sib>, <sib_usr>, <sib_context>y <mqema> .

Los grupos de las carpetas de propiedades definidas por IBM, excepto las carpetas <wmq> y <mq> , se analizan para las propiedades.

Una carpeta MQRFH2 no puede contener contenido mixto; una carpeta o grupo puede contener grupos o propiedades, o un valor, pero no ambos.

Un segmento de un mensaje, ya sea el primero o un segmento posterior, no puede contener propiedades definidas por IBM MQ que no sean las propiedades del descriptor de mensaje. Por lo tanto, colocar un mensaje que contenga dichas propiedades con el conjunto MQMF_SEGMENT o MQMF_SEGMENTATION_ALLOWED hace que la colocación falle con MQRC_SEGMENTATION_NOT_ALLOWED.

Sin embargo, los grupos de mensajes pueden contener propiedades definidas por IBM MQ .

Generación de cabeceras MQRFH2

Si IBM MQ convierte las propiedades de mensaje a su representación MQRFH2 , debe añadir MQRFH2 al mensaje. Añade MQRFH2 como una cabecera separada o lo fusiona con una cabecera existente.

La generación de nuevas cabeceras MQRFH2 mediante IBM MQ puede interrumpir las cabeceras existentes en un mensaje. Las aplicaciones que analizan un almacenamiento intermedio de mensajes para cabeceras deben tener en cuenta que el número y la posición de las cabeceras en un almacenamiento intermedio pueden cambiar en algunas circunstancias. IBM MQ intenta minimizar el impacto de añadir propiedades a un mensaje fusionando propiedades de mensaje en una cabecera MQRFH2 existente, donde puede. También intenta minimizar el impacto insertando un MQRFH2 generado en una posición fija relativa a otras cabeceras en el almacenamiento intermedio de mensajes.

Una cabecera MQRFH2 generada se coloca a continuación de la MQMDy de cualquier número de cabeceras MQXQH, MQRFH, MQDLH , independientemente del orden en el que se encuentren. La cabecera MQRFH2 generada se coloca inmediatamente antes de la primera cabecera que no es una cabecera MQMD, MQXQH, MQDLHo MQRFH .

 En sistemas z/OS , la cabecera MQRFH2 generada se crea en CCSIDde la aplicación. Esto se define de la siguiente manera:

- Para las aplicaciones LE por lotes que utilizan la interfaz DLL, CCSID es el CODESET asociado con el entorno local actual en el momento en que se emite **MQCONN** (el valor predeterminado es 1047).
- En el caso de las aplicaciones LE por lotes vinculadas a uno de los stubs de MQ, el CCSID es el CODESET asociado a la configuración regional vigente en el momento de la primera llamada a MQI realizada tras **MQCONN** (el valor por defecto es 1047).
- En el caso de las aplicaciones por lotes que no son de LE y se ejecutan en un z/OS UNIX System Services hilo (z/OS UNIX), el valor de CCSID es el valor de THLICCSID en el momento de la primera llamada a `MQI` realizada después de **MQCONN** (el valor por defecto es 1047).
- Para otras aplicaciones por lotes, CCSID es el CCSID del gestor de colas.

Para las aplicaciones LE, el entorno local se puede cambiar utilizando el servicio invocable `setlocale()` / `CEESETL LE`. Para las aplicaciones no LE que se ejecutan en hebras z/OS UNIX, el valor de `THLICCSID` se puede cambiar utilizando la z/OS UNIX macro de correlación **BPXYTHLI**.

Reglas para fusionar MQRFH2 generado

Las reglas siguientes se aplican a la fusión de un MQRFH2 generado con un MQRFH2 existente. La cabecera MQRFH2 generada se fusiona con una cabecera MQRFH2 existente, si:

1. El MQRFH2 existente está en la misma posición que IBM MQ colocaría un MQRFH2 generado o anterior en la cadena de cabecera.
2. El CCSID de las propiedades generadas es el mismo que el `NameValueCCSID` del existente MQRFH2.

De lo contrario, la cabecera generada se coloca por separado en el almacenamiento intermedio, en la posición descrita anteriormente.

Reglas para fusionar carpetas en una MQRFH2 existente

Si las propiedades de mensaje se fusionan en un MQRFH2 existente, el MQRFH2 existente se explora en busca de carpetas que coincidan con las propiedades de mensaje y las fusiona. Si no existe una carpeta coincidente, se añade una carpeta nueva al final de las carpetas existentes. Si existe una carpeta coincidente, se busca en la carpeta. Las propiedades coincidentes se sobrescriben. Las nuevas se añaden al final de la carpeta.

Restricciones de la carpeta MQRFH2

Visión general de las restricciones de carpeta en cabeceras MQRFH2

Las restricciones MQRFH2 se aplican a las carpetas siguientes:

- Los nombres de elemento de la carpeta `<usr>` no deben empezar con el prefijo JMS ; dichos nombres de propiedad están reservados para que los utilice JMS y no son válidos para las propiedades definidas por el usuario.

Un nombre de elemento de este tipo no hace que falle el análisis de MQRFH2, pero no es accesible para las API de propiedad de mensaje de IBM MQ.

- Los nombres de elemento de la carpeta `<usr>` no pueden ser, en cualquier combinación de minúsculas o mayúsculas, NULL, TRUE, FALSE, NOT, AND, OR, BETWEEN, LIKE, IN, IS y ESCAPE. Estos nombres coinciden con las palabras clave SQL y dificultan el análisis de los selectores, porque `<usr>` es la carpeta predeterminada que se utiliza cuando no se especifica ninguna carpeta para una propiedad determinada en un selector.

Un nombre de elemento de este tipo no hace que falle el análisis de MQRFH2, pero no es accesible para las API de propiedad de mensaje de IBM MQ.

- El modelo de contenido de la carpeta `<usr>` es el siguiente:
 - Se puede utilizar cualquier nombre XML válido como nombre de elemento, siempre que no contenga dos puntos.
 - Sólo se permiten elementos simples, no carpetas anidadas.
 - Todos los elementos toman el tipo predeterminado de serie, a menos que lo modifique un atributo `dt="xxx"`.
 - Todos los elementos son opcionales, pero no deben aparecer más de una vez en una carpeta.
- Los nombres de elemento de cualquier carpeta que se considere que contiene propiedades de mensaje no deben contener un punto (.) (Carácter Unicode U+002E), porque se utiliza en los nombres de propiedad para indicar la jerarquía.

Un nombre de elemento de este tipo no hace que falle el análisis de MQRFH2, pero no es accesible para las API de propiedad de mensaje de IBM MQ.

En general, IBM MQ puede analizar las cabeceras MQRFH2 que contienen datos de estilo XML válidos sin errores, aunque determinados elementos de la MQRFH2 no son accesibles a través de las API de propiedad de mensaje IBM MQ .

Conflictos de nombre de elemento MQRFH2

Visión general de los conflictos dentro de los nombres de elemento MQRFH2 .

Sólo se puede adjuntar un valor a una propiedad de mensaje. Si un intento de acceder a una propiedad da lugar a un conflicto de valores, se elige uno con preferencia sobre otro.

La sintaxis IBM MQ para acceder a los elementos MQRFH2 permite la identificación exclusiva de un elemento, si una carpeta no contiene elementos con el mismo nombre. Si una carpeta contiene más de un elemento con el mismo nombre, el valor de la propiedad utilizada es el más cercano a la cabecera del mensaje.

Esto se aplica si dos o más carpetas con el mismo nombre están contenidas en diferentes cabeceras MQRFH2 significativas dentro del mismo mensaje.

Puede producirse un conflicto cuando la llamada MQGET se procesa después de que se haya establecido una propiedad de descriptor que no es de mensaje dos veces: a través de una llamada MQSETMP y directamente en la cabecera MQRFH2 en bruto.

Si esto sucede, la propiedad asociada al mensaje por una llamada de API tiene preferencia sobre una en los datos del mensaje, es decir, la de la cabecera MQRFH2 sin formato. Si se produce un conflicto, se considera que se produce lógicamente antes de los datos del mensaje.

Correlación de nombres de propiedad con nombres de carpeta y elemento MQRFH2

Visión general de las diferencias entre los nombres de propiedad y los nombres de elemento en la cabecera MQRFH2 .

Cuando se utiliza cualquiera de las API definidas que finalmente generan cabeceras MQRFH2 , para especificar propiedades de mensaje (por ejemplo, MQ JMS), el nombre de propiedad no es necesariamente el nombre de elemento en la carpeta MQRFH2 .

Por lo tanto, se produce una correlación entre el nombre de propiedad y el elemento MQRFH2 , y de forma inversa, teniendo en cuenta tanto el nombre de carpeta que contiene el elemento como el nombre de elemento. Algunos ejemplos de IBM MQ classes for JMS ya están documentados en [Usando IBM MQ classes for Java](#).

Tabla 672. Nombres de propiedad correlacionados con la carpeta MQRFH2 y nombres de elemento		
Nombre de propiedad	Nombre de carpeta de MQRFH2	Nombre de elemento MQRFH2
JMSDestination	jms	Dst
JMSType	mcd	Type, Set, Fmt
xxx (definido por el usuario, donde xxx no empieza por JMS)	usr	xxx

Por lo tanto, cuando una aplicación JMS accede a la propiedad JMSDestination , esta se correlaciona con el elemento Dst de la carpeta <jms> .

Al especificar propiedades como elementos MQRFH2 , IBM MQ define sus elementos de la forma siguiente:

Tabla 673. Nombres de propiedad correlacionados con nombres de carpeta, grupo y elemento MQRFH2			
Nombre de propiedad	Nombre de carpeta de MQRFH2	Nombre de grupo MQRFH2	Nombre de elemento MQRFH2
<Property>	<usr>	n/d	<Property>
<folder>. <Property>	<folder>	n/d	<Property>

Tabla 673. Nombres de propiedad correlacionados con nombres de carpeta, grupo y elemento MQRFH2 (continuación)

Nombre de propiedad	Nombre de carpeta de MQRFH2	Nombre de grupo MQRFH2	Nombre de elemento MQRFH2
<folder>. <group>. <Property>	<folder>	<group>	<Property>

Por ejemplo, cuando una aplicación IBM MQ intenta acceder a la propiedad Property1 , se correlaciona con el elemento Property1 en la carpeta <usr> . La propiedad wmq .Property2 se correlaciona con la propiedad Property2 en la carpeta <wmq> .

Si el nombre de propiedad contiene más de uno. , el nombre de elemento MQRFH2 utilizado es el que sigue al final. y los grupos MQRFH2 se utilizan para formar una jerarquía; se permiten grupos MQRFH2 anidados.

Una aplicación IBM MQ accede a la cabecera JMS y a las propiedades específicas del proveedor contenidas en una MQRFH2 en las carpetas <mcd>, <jms>y <mqext> utilizando los nombres abreviados definidos en [Utilización de IBM MQ classes for Java](#) .

Se accede a las propiedades definidas por el usuario de JMS desde la carpeta <usr> . Una aplicación IBM MQ puede utilizar la carpeta <usr> para sus propiedades de aplicación si es aceptable que la propiedad aparezca en las aplicaciones JMS como una de sus propiedades definidas por el usuario.

Si no es aceptable, elija otra carpeta; la carpeta <wmq_usr> se proporciona como ubicación estándar para dichas propiedades noJMS .

Las aplicaciones pueden especificar y utilizar cualquier carpeta MQRFH2 con un uso bien definido, no documentado en “[Propiedades especificadas como elementos MQRFH2](#)” en la página 938 si observa lo siguiente:

1. Es posible que la carpeta ya esté en uso, o que la utilice en el futuro, otra aplicación que proporcione acceso no definido a las propiedades contenidas en ella; consulte [Nombres de propiedad](#) para ver el convenio de denominación sugerido para los nombres de propiedad.
2. Las propiedades no son accesibles para las versiones anteriores del cliente IBM MQ classes for JMS o XMS que sólo puede acceder a la carpeta <usr> para las propiedades definidas por el usuario
3. La carpeta debe estar marcada con el atributo content con el valor establecido en properties, por ejemplo, content='properties'.

“MQSETMP: establecer la propiedad del mensaje” en la página 782 añade automáticamente este atributo según sea necesario. Este atributo no se debe añadir a ninguna de las carpetas definidas por IBM, por ejemplo, <jms> y <usr>. De este modo, hace que el cliente IBM MQ classes for JMS rechace el mensaje antes de IBM WebSphere MQ 7.0. con un MessageFormatException.

Puesto que la carpeta <usr> es la ubicación predeterminada para las propiedades de la sintaxis <Property> , una aplicación IBM MQ y una aplicación JMS para acceder al mismo valor de propiedad definido por el usuario utilizando el mismo nombre.

Nombres de carpeta reservados

Hay varios nombres de carpeta reservados. No puede utilizar nombres como los prefijos de carpeta; por ejemplo, Root .Property1 no accede a una propiedad válida porque Root está reservado. La lista siguiente contiene nombres de carpeta reservados:

- Raíz
- Cuerpo
- Propiedades
- Entorno
- LocalEnvironment
- DestinationList
- ExceptionList

- InputBody
- InputRoot
- InputProperties
- InputLocalEnvironment
- InputDestinationList
- InputExceptionList
- OutputRoot
- OutputLocalEnvironment
- OutputDestinationList
- OutputExceptionList

Correlación de campos de descriptor de propiedad en cabeceras MQRFH2

Cuando una propiedad se convierte en un elemento MQRFH2 , se utilizan los siguientes atributos de elemento para especificar los campos significativos del descriptor de propiedad: describe cómo se convierten los campos MQPD en atributos del elemento MQRFH2 .

Soporte

El campo Descriptor de propiedad de soporte se divide en tres atributos de elemento

- El atributo de elemento **sr** especifica valores en la máscara de bits MQPD_REJECT_UNSUP_MASK.
- El atributo de elemento **sa** especifica valores en la máscara de bits MQPD_ACCEPT_UNSUP_MASK.
- El atributo de elemento **sx** especifica valores en la máscara de bits MQPD_ACCEPT_UNSUP_IF_XMIT_MASK.

Estos atributos de elemento sólo son válidos en la carpeta < mq> y se ignoran si se establecen en elementos de las otras carpetas que contienen propiedades.

Tabla 674. Campos MQPD correlacionados con atributos de elemento MQRFH2		
Valor de soporte	Atributo de elemento MQRFH2	Valor de atributo MQRFH2
MQPD_SUPPORT_OPTIONAL	sa	opcional Éste es el valor predeterminado.
MQPD_SE_REQUIERE_ASISTENCIA	sr	necesario
MQPD_SE_REQUIERE_SOPORTE_SI_ES_LOCAL	sx	local

Contexto

Utilice el atributo de elemento **context** para indicar el contexto de mensaje al que pertenece una propiedad. Utilice un solo valor. Este atributo de elemento es válido en una propiedad de cualquier carpeta que contenga propiedades.

Tabla 675. Valores de contexto correlacionados con valores de atributo MQRFH2	
Valor de contexto	Valor de atributo MQRFH2
MQPD_NO_CONTEXT	ninguno Éste es el valor predeterminado.
MQPD_USER_CONTEXT	usuario

CopyOptions

Utilice el atributo de elemento **copy** para indicar los mensajes en los que se debe copiar una propiedad. Se acepta más de un valor; separe varios valores con una coma. Por ejemplo, **copy='reply'** y **copy='publish,report'** son válidos. Este atributo de elemento es válido en una propiedad de cualquier carpeta que contenga propiedades.

Nota: En la definición de atributo, las comillas simples o las comillas dobles son un uso válido, por ejemplo **copy='reply'** o **copy="report"**

Tabla 676. Valores de CopyOption correlacionados con valores de atributo MQRFH2	
Valor de CopyOption	Valor de atributo MQRFH2
MQPD_COPY_FORWARD	hacia adelante
MQPD_COPIA_RESPUESTA	responder
MQPD_COPIA_INFORME	informe
MQPD_COPY_PUBLICAR	publicar
MQPD_COPY_ALL	todo No especifique esto con ningún otro valor. Cuando se utiliza con otro valor, tiene prioridad sobre cualquier valor excepto none .
MQPD_COPY_DEFAULT	valor predeterminado Éste es el valor predeterminado. Es equivalente a especificar los tres valores MQCOPY_FORWARD, MQCOPY_REPORT y MQCOPY_PUBLISH. No especifique esto con ningún otro valor.
MQPD_COPY_NONE	ninguno No especifique esto con ningún otro valor. Cuando se utiliza con otro valor, esto tiene prioridad.

Restricciones a la carpeta < mq> MQRFH2

Cuando se coloca un mensaje en una cola, se busca una carpeta < mq> para que el mensaje se pueda procesar de acuerdo con sus propiedades definidas por MQ. Para permitir el análisis eficaz de las propiedades definidas por MQ, se aplican las restricciones siguientes a la carpeta:

- MQ sólo actúa sobre las propiedades de la primera carpeta < mq> significativa del mensaje; las propiedades de cualquier otra carpeta < mq> del mensaje se ignoran.
- Si la carpeta está en UTF-8, sólo se permiten caracteres UTF-8 de un solo byte en la carpeta. Un carácter de varios bytes en la carpeta puede hacer que falle el análisis y que se rechace el mensaje.
- No incluya grupos MQRFH2 en la carpeta < mq>. La presencia del carácter Unicode U+003C en un valor de propiedad hará que se rechace el mensaje.
- No utilice series de escape en la carpeta. Una serie de escape se trata como el valor real del elemento.
- Sólo el carácter Unicode U+0020 se trata como un espacio en blanco dentro de la carpeta. Todos los demás caracteres se tratan como significativos y pueden hacer que falle el análisis de la carpeta y que se rechace el mensaje.

Si el análisis de la carpeta < mq> falla, o si la carpeta no observa estas restricciones, el mensaje se rechaza con CompCode **MQCC_FAILED** y Reason **MQRC_RFH_RESTRICTED_FORMAT_ERR**.

Cabeceras MQRFH2 que no son válidas

En el momento en que se procesa una llamada MQPUT, MQPUT1 o MQGET, se puede producir un análisis parcial de las cabeceras MQRFH2 del mensaje para comprobar qué carpetas se incluyen y para determinar si las carpetas contienen propiedades. Visión general de cabeceras MQRFH2 que no son válidas.

Si el análisis parcial del mensaje no se puede completar correctamente porque la estructura no es válida, por ejemplo, el campo StructLength es demasiado pequeño, entonces:

- La llamada MQPUT o MQPUT1 falla con el código de razón MQRC_RFH_ERROR, si se puede determinar que la aplicación incluye alguna opción IBM WebSphere MQ 7, para que las aplicaciones existentes no fallen.
- La llamada MQGET se devuelve correctamente y el MQRFH2 que contiene el error se devuelve en el almacenamiento intermedio que ha proporcionado.

Si el análisis parcial falla porque no se puede detectar si una carpeta determinada contiene propiedades o no, por ejemplo, la carpeta empieza <<jms, por lo que el análisis falla antes de que se determine el nombre de la carpeta, entonces:

- La llamada MQPUT o MQPUT1 falla con el código de razón MQRC_RFH_FORMAT_ERROR, si se puede determinar que la aplicación incluye alguna opción IBM WebSphere MQ 7, para que las aplicaciones existentes no fallen.
- La llamada MQGET se devuelve correctamente y el MQRFH2 que contiene el error se devuelve en el almacenamiento intermedio que ha proporcionado.
- Mientras está internamente dentro del gestor de colas, el mensaje no se rechaza debido a la carpeta con formato incorrecto, pero la carpeta siempre se trata como si no hubiera ninguna propiedad dentro de ella.

Un mensaje puede fluir a través de la red de gestores de colas con una carpeta que contenga un error de sintaxis de este tipo, pero que nunca se analice y se detecte, mientras que una o más carpetas del mensaje son:

- Válido
- Se ha analizado satisfactoriamente
- Se utiliza en el proceso del mensaje

Por lo tanto, la detección no está garantizada.

Si una de las aplicaciones utiliza [“MQSETMP: establecer la propiedad del mensaje”](#) en la página 782, o MQINQMP para acceder a una propiedad, y al hacerlo hace que una carpeta MQRFH2 se analice completamente, detectando un error de forma que el análisis no se puede completar, esto se indica mediante un código de retorno adecuado a la llamada de API. No se pone a disposición de la aplicación ninguna propiedad de la carpeta.

Si se intenta analizar completamente una carpeta MQRFH2 y el analizador encuentra atributos de elemento no reconocidos, o un tipo de datos no reconocido, el análisis continúa y se completa correctamente sin que se emitan avisos; esto no constituye un error de análisis.

Conversión de páginas de códigos

En esta sección se describen los nombres de los conjuntos de códigos y los CCSID, los idiomas nacionales y z/OS conversión, IBM i conversión, la compatibilidad con la conversión Unicode.

Cada sección de idioma nacional incluye la siguiente información:

- Los CCSID nativos compatibles
- Las conversiones de página de códigos que no son compatibles

En la información se utilizan los siguientes términos:

AIX

Indica IBM MQ for AIX.

Linux

Indica IBM MQ para Linux para Intel y IBM MQ para Linux para zSeries.

OS/400

Indica IBM MQ for IBM i.

Windows

Indica IBM MQ for Windows.

z/OS

Indica IBM MQ for z/OS.

El valor predeterminado para la conversión de datos es que la conversión se realice en el sistema de destino (receptor).

Si el producto de origen admite la conversión, se puede configurar un canal e intercambiar datos estableciendo el atributo del canal CONVERT en YES en el origen.

Nota:

1. La conversión de IBM MQ MQI client la información se realiza en el servidor, por lo que este debe admitir la conversión del CCSID del cliente al CCSID del servidor.
2. La conversión podría incluir soporte añadido por CSD/PTF a la última versión de IBM MQ. Compruebe el contenido del último nivel de servicio para ver si necesita instalar un CSD/PTF para habilitar esta conversión.
3. El gestor IBM MQ de colas CCSID debe ser Mixto o SBCS.
4. Algunos CCSID, por ejemplo 850 en AIX, que no son compatibles con el sistema operativo, pueden seguir siendo utilizados por la aplicación y también pueden configurarse como el CCSID del gestor IBM MQ de colas. Esto solo se permite con fines de compatibilidad con versiones anteriores y la conversión fallará si no se instalan las tablas de conversión pertinentes.

Consulte [Tabla 677 en la página 948](#) para obtener una referencia cruzada entre algunos de los números CCSID y algunos nombres de conjuntos de códigos de la industria.

Referencia relacionada

“Idiomas nacionales” en la [página 949](#)

Esta información contiene idiomas soportados por IBM MQ.

Nombres de conjuntos de códigos y CCSID

Nombres de conjuntos de códigos y los CCSID correspondientes a cada nombre de conjunto de códigos.

 **z/OS** IBM MQ for z/OS proporciona más conversiones que las que figuran en las tablas específicas de cada idioma. Para ver una lista completa de conversiones, consulta [Tabla 710 en la página 975](#) en “z/OS soporte para conversión” en la [página 974](#).

Tabla 677. Nombres de conjuntos de códigos y CCSID	
Nombres de conjuntos de códigos	CCSID
ISO 8859-1	819
ISO 8859-2	912
ISO 8859-3	913
ISO 8859-5	915
ISO 8859-6	1089

Tabla 677. Nombres de conjuntos de códigos y CCSID (continuación)

Nombres de conjuntos de códigos	CCSID
ISO 8859-7	813
ISO 8859-8	916
ISO 8859-9	920
ISO 8859-13	921
ISO 8859-15 (euro)	923
big5	950
eucJP	954 5050 33722
eucKR	970
eucTW	964
eucCN	1383
pck	943
GBK	1386
koi8-r	878
UTF8	1208

Idiomas nacionales

Esta información contiene idiomas soportados por IBM MQ.

Los idiomas soportados por IBM MQ son:

- Inglés de EE.UU.-consulte el tema [“Inglés de EE.UU.”](#) en la página 950
- Alemán-consulte el tema [“Alemán”](#) en la página 950
- Danés y noruego-véase el tema [“Danés y noruego”](#) en la página 951
- Finés y sueco-véase el tema [“Finlandés y sueco”](#) en la página 952
- Italiano-consulte el tema [“Italiano”](#) en la página 953
- Español-consulte el tema [“Español”](#) en la página 953
- Inglés/Gaélico del Reino Unido-consulte el tema [“Inglés británico / Gaélico”](#) en la página 954
- Francés-consulte el tema [“Francés”](#) en la página 955
- Multilingüe-consulte el tema [“Multilingüe”](#) en la página 955
- Portugués-consulte el tema [“Portugués”](#) en la página 956
- Islandés-consulte el tema [“Islandés”](#) en la página 957
- Lenguas de Europa Oriental-véase el tema [“Idiomas de Europa del Este”](#) en la página 957
- Cirílico-consulte el tema [“Cirílico”](#) en la página 959
- Estonio-consulte el tema [“Estonio”](#) en la página 959
- Letón y lituano-véase el tema [“Letón y lituano”](#) en la página 961
- Ucraniano-consulte el tema [“Ucraniano”](#) en la página 962
- Griego-consulte el tema [“Griego”](#) en la página 962
- Turco-consulte el tema [“Turco”](#) en la página 963
- Hebreo-consulte el tema [“Hebreo”](#) en la página 964
- Farsi-consulte el tema [“Persa”](#) en la página 966

- Urdu-consulte el tema [“Urdú”](#) en la página 966
- Tailandés-consulte el tema [“Tailandés”](#) en la página 966
- Lao-consulte el tema [“lao”](#) en la página 967
- Vietnamita-consulte el tema [“Vietnamita”](#) en la página 967
- Japonés latín SBCS-consulte el tema [“Japonés Latín SBCS”](#) en la página 968
- Japonés Katakana SBCS-consulte el tema [“Katakana japonés SBCS”](#) en la página 969
- Japonés Kanji/latín mixto-consulte el tema [“Kanji japonés/Latín mixto”](#) en la página 970
- Japonés Kanji/Katakana mixto-consulte el tema [“Kanji japonés/ Katakana mixto”](#) en la página 971
- Coreano-consulte el tema [“Coreano”](#) en la página 972
- Chino simplificado-consulte el tema [“Chino simplificado”](#) en la página 973
- Chino tradicional-consulte el tema [“Chino tradicional”](#) en la página 974

Inglés de EE.UU.

Detalles sobre los CCSID y la conversión de CCSID para el inglés estadounidense.

Tabla 678. CCSID nativos para inglés estadounidense en plataformas compatibles	
Plataforma	CCSID nativos
 IBM i  z/OS	37, 924, 1140
 AIX	819, 923, 5348
 Windows	437, 850, 1252, 5348, 858
 Linux	819, 923
Cliente de Apple	1275

Todas las plataformas que no son de cliente admiten la conversión entre sus CCSID nativos y los CCSID nativos de las demás plataformas, con las siguientes excepciones.

IBM i



Página de códigos:

37

No se convierte a las páginas de códigos 923 y 858

924

No se convierte a las páginas de códigos 437, 858, 1051, 1140, 1252, 1275, 5348

1140

No se convierte a las páginas de códigos 924, 1051, 1275

Alemán

Detalles sobre los CCSID y la conversión CCSID para el alemán.

Tabla 679. CCSID nativos para alemán en plataformas compatibles

Plataforma	CCSID nativos
 IBM i  z/OS	273, 924, 1141
 AIX	819, 923, 5348
 Windows	437, 850, 858, 1252, 5348
 Linux	819, 923
Cliente de Apple	1275

Todas las plataformas que no son de cliente admiten la conversión entre sus CCSID nativos y los CCSID nativos de las demás plataformas, con las siguientes excepciones.

IBM i



Página de códigos:

273

No se convierte a las páginas de códigos 858, 923, 924, 1275

924

No se convierte a las páginas de códigos 273, 437, 858, 1051, 1141, 1252, 1275, 5348

1141

No se convierte a las páginas de códigos 924, 1051, 1275

Danés y noruego

Detalles sobre los CCSID y la conversión CCSID para danés y noruego.

Tabla 680. CCSID nativos para danés y noruego en plataformas compatibles

Plataforma	CCSID nativos
 IBM i  z/OS	277, 924, 1142
 AIX	819, 923, 5348
 Windows	850, 858, 865, 1252, 5348
 Linux	819, 923
Cliente de Apple	1275

Todas las plataformas que no son de cliente admiten la conversión entre sus CCSID nativos y los CCSID nativos de las demás plataformas, con las siguientes excepciones.

IBM i



Página de códigos:

277

No se convierte a las páginas de códigos 858, 923, 924, 1275

924

No se convierte a las páginas de códigos 277, 858, 865, 1051, 1142, 1252, 1275, 5348

1142

No se convierte a las páginas de códigos 924, 865, 1051, 1275

AIX



Página de códigos:

819

No se convierte a la página de códigos 865

Windows



Página de códigos:

865

No se convierte a las páginas de códigos 1051 y 1275

Finlandés y sueco

Detalles sobre los CCSID y la conversión CCSID para finés y sueco.

Tabla 681. CCSID nativos para finés y sueco en plataformas compatibles	
Plataforma	CCSID nativos
 IBM i	278, 924, 1143
 z/OS	
 AIX	819, 923, 5348
 Windows	437, 850, 858, 865, 1252, 5348
 Linux	819, 923
Cliente de Apple	1275

Todas las plataformas que no son de cliente admiten la conversión entre sus CCSID nativos y los CCSID nativos de las demás plataformas, con las siguientes excepciones.

IBM i



Página de códigos:

278

No se convierte a las páginas de códigos 858, 923, 924, 1275

924

No se convierte a las páginas de códigos 278, 437, 858, 865, 1051, 1143, 1252, 1275, 5348

1143

No se convierte a las páginas de códigos 865, 924, 1051, 1275

AIX



Página de códigos:

819

No se convierte a la página de códigos 865

850

No se convierte a la página de códigos 865

Windows



Página de códigos:

865

No se convierte a las páginas de códigos 1051 y 1275

Italiano

Detalles sobre los CCSID y la conversión CCSID para el italiano.

Tabla 682. CCSID nativos para italiano en plataformas compatibles	
Plataforma	CCSID nativos
IBM i z/OS	280, 924, 1144
AIX	819, 923, 5348
Windows	437, 850, 858, 1252, 5348
Linux	819, 923
Cliente de Apple	1275

Todas las plataformas que no son de cliente admiten la conversión entre sus CCSID nativos y los CCSID nativos de las demás plataformas, con las siguientes excepciones.

IBM i



Página de códigos:

280

No se convierte a las páginas de códigos 858, 923, 924, 1275

924

No se convierte a las páginas de códigos 280, 437, 858, 1051, 1144, 1252, 1275, 5348

1144

No se convierte a las páginas de códigos 924, 1051, 1275

Español

Detalles sobre los CCSID y la conversión CCSID para el español.

Tabla 683. CCSID nativos para español en plataformas compatibles

Plataforma	CCSID nativos
 IBM i  z/OS	284, 924, 1145
 AIX	819, 923, 5348
 Windows	437, 850, 858, 1252, 5348
 Linux	819, 923
Cliente de Apple	1275

Todas las plataformas que no son de cliente admiten la conversión entre sus CCSID nativos y los CCSID nativos de las demás plataformas, con las siguientes excepciones.

IBM i



Página de códigos:

284

No se convierte a las páginas de códigos 858, 923, 924, 1275

924

No se convierte a las páginas de códigos 284, 437, 858, 1051, 1145, 1252, 1275, 5348

1145

No se convierte a las páginas de códigos 924, 1051, 1275

Inglés británico / Gaélico

Detalles sobre los CCSID y la conversión CCSID para inglés británico/gaélico.

Tabla 684. CCSID nativos para inglés británico/gaélico en plataformas compatibles

Plataforma	CCSID nativos
 IBM i  z/OS	285, 924, 1146
 AIX	819, 923, 5348
 Windows	437, 850, 858, 1252, 5348
 Linux	819, 923
Cliente de Apple	1275

Todas las plataformas que no son de cliente admiten la conversión entre sus CCSID nativos y los CCSID nativos de las demás plataformas, con las siguientes excepciones.

IBM i



Página de códigos:

285

No se convierte a las páginas de códigos 858, 923, 924, 1275

924

No se convierte a las páginas de códigos 285, 437, 858, 1051, 1146, 1252, 1275, 5348

1146

No se convierte a las páginas de códigos 924, 1051, 1275

Francés

Detalles sobre los CCSID y la conversión CCSID para el francés.

Tabla 685. CCSID nativos para francés en plataformas compatibles	
Plataforma	CCSID nativos
 IBM i	297, 924, 1147
 z/OS	
 AIX	819, 923, 5348
 Windows	437, 850, 858, 1252, 5348
 Linux	819, 923
Cliente de Apple	1275

Todas las plataformas que no son de cliente admiten la conversión entre sus CCSID nativos y los CCSID nativos de las demás plataformas, con las siguientes excepciones.

IBM i



Página de códigos:

297

No se convierte a las páginas de códigos 858, 923, 924, 1275, 5348

924

No se convierte a las páginas de códigos 297, 437, 858, 1051, 1147, 1252, 1275, 5348

1147

No se convierte a las páginas de códigos 924, 1051, 1275

Multilingüe

Detalles sobre los CCSID y la conversión CCSID para multilingüe.

Tabla 686. CCSID nativos para conversión multilingüe en plataformas compatibles	
Plataforma	CCSID nativos
 IBM i	500, 924, 1148
 z/OS	
 AIX	819, 923, 5348
 Windows	437, 850, 858, 1252, 5348

Tabla 686. CCSID nativos para conversión multilingüe en plataformas compatibles (continuación)

Plataforma	CCSID nativos
 Linux	819, 923
Cliente de Apple	1275

Todas las plataformas que no son de cliente admiten la conversión entre sus CCSID nativos y los CCSID nativos de las demás plataformas, con las siguientes excepciones.

IBM i



Página de códigos:

500

No se convierte a las páginas de códigos 858 y 923

924

No se convierte a las páginas de códigos 437, 858, 1051, 1148, 1252, 1275, 5348

1148

No se convierte a las páginas de códigos 924, 1051, 1275

Portugués

Detalles sobre los CCSID y la conversión CCSID para el portugués.

Tabla 687. CCSID nativos para portugués en plataformas compatibles

Plataforma	CCSID nativos
 IBM i	37, 500, 924, 1140
 z/OS	500, 924, 1140
 AIX	819, 923, 5348
 Windows	850, 858, 860, 1252, 5348
 Linux	819, 923
Cliente de Apple	1275

Todas las plataformas que no son de cliente admiten la conversión entre sus CCSID nativos y los CCSID nativos de las demás plataformas, con las siguientes excepciones.

IBM i



Página de códigos:

37

No se convierte a las páginas de códigos 858, 923, 1275

500

No se convierte a las páginas de códigos 858, 923, 1275

924

No se convierte a las páginas de códigos 858, 860, 1051, 1140, 1252, 1275, 5348

1140

No se convierte a las páginas de códigos 860, 924, 1051, 1275

Windows



Página de códigos:

860

No se convierte a las páginas de códigos 1051 y 1275

Islandés

Detalles sobre los CCSID y la conversión CCSID para el islandés.

Tabla 688. CCSID nativos para islandés en plataformas compatibles	
Plataforma	CCSID nativos
 IBM i	871, 924, 1149
 z/OS	
 AIX	819, 923, 5348
 Windows	850, 858, 861, 1252, 5348
 Linux	819, 923
Cliente de Apple	1275

Todas las plataformas que no son de cliente admiten la conversión entre sus CCSID nativos y los CCSID nativos de las demás plataformas, con las siguientes excepciones.

IBM i



Página de códigos:

871

No se convierte a las páginas de códigos 858, 923, 924, 1275, 5348

924

No se convierte a las páginas de códigos 858, 861, 871, 1051, 1149, 1252, 1275, 5348

1149

No se convierte a las páginas de códigos 924, 1051, 1275

Windows



Página de códigos:

861

No se convierte a las páginas de códigos 1051 y 1275

Idiomas de Europa del Este

Detalles sobre los CCSID y la conversión CCSID para los idiomas de Europa del Este. Los idiomas típicos que utilizan estos CCSID son el albanés, el croata, el checo, el húngaro, el polaco, el rumano, el serbio, el eslovaco y el esloveno.

Tabla 689. CCSID nativos para idiomas de Europa del Este en plataformas compatibles

Plataforma	CCSID nativos
 IBM i  z/OS	870, 1153
 Windows	852, 1250, 5346, 9044
 AIX  Linux	912
Cliente Apple de Europa del Este	1282
Cliente rumano de Apple	1285
Cliente Apple croata	1284

Todas las plataformas que no son de cliente admiten la conversión entre sus CCSID nativos y los CCSID nativos de las demás plataformas, con las siguientes excepciones.

z/OS



Página de códigos:

870

No se convierte a las páginas de códigos 1284 y 1285

1153

No se convierte a las páginas de códigos 1250, 1284, 1285

IBM i



Página de códigos:

870

No se convierte a las páginas de códigos 1284, 1285, 5346, 9044

1153

No se convierte a las páginas de códigos 1282, 1284, 1285, 5346, 9044

Linux



Página de códigos:

912

No se convierte a las páginas de códigos 1284 y 1285

Windows



Página de códigos:

852

No se convierte a las páginas de códigos 1284 y 1285

1250

No se convierte a las páginas de códigos 1284 y 1285

9044

No se convierte a las páginas de códigos 912, 1282, 1284, 1285

Cirílico

Detalles sobre los CCSID y la conversión CCSID para el alfabeto cirílico. Los idiomas típicos que utilizan estos CCSID incluyen el bielorruso, el búlgaro, el macedonio y el serbio.

Tabla 690. CCSID nativos para cirílico en plataformas compatibles	
Plataforma	CCSID nativos
 z/OS	1025
 IBM i	880, 1025
 Windows	855, 866, 1131, 1251, 5347
 AIX	915
 Linux	
Cliente de Apple	1283

Todas las plataformas que no son de cliente admiten la conversión entre sus CCSID nativos y los CCSID nativos de las demás plataformas, con las siguientes excepciones.

IBM i



Página de códigos:

880

No se convierte a las páginas de códigos 855, 866, 878, 1131, 5347

1025

No se convierte a las páginas de códigos 878 y 5347

Windows



Página de códigos:

855

No se convierte a la página de códigos 1131

866

No se convierte a la página de códigos 1131

1131

No se convierte a las páginas de códigos 855, 866, 880, 1283

Estonio

Detalles sobre los CCSID y la conversión CCSID para el estonio.

Tabla 691. CCSID nativos para estonio en plataformas compatibles

Plataforma	CCSID nativos
 IBM i  z/OS	1122, 1157
 Windows	902, 922, 1257, 5353, 9449
 AIX  Linux	902, 922

Todas las plataformas admiten la conversión entre sus CCSID nativos y los CCSID nativos de otras plataformas, con las siguientes excepciones.

z/OS



Página de códigos:

1122

No se convierte a las páginas de códigos 902, 1157, 9449

1157

No se convierte a las páginas de códigos 922, 1122, 1257, 9449

IBM i



Página de códigos:

1122

No se convierte a las páginas de códigos 902, 5353, 9449

1157

No se convierte a las páginas de códigos 922, 5353, 9449

Linux



Página de códigos:

902

No se convierte a las páginas de códigos 922, 1122, 9449

922

No se convierte a las páginas de códigos 902, 1157, 9449

Windows



Página de códigos:

5353

No se convierte a la página de códigos 9449

9449

No se convierte a las páginas de códigos 902, 922, 1122, 1157, 1257, 5353

902

No se convierte a las páginas de códigos 922, 1122, 9449

Letón y lituano

Detalles sobre los CCSID y la conversión CCSID para letón y lituano.

Tabla 692. CCSID nativos para letón y lituano en plataformas compatibles	
Plataforma	CCSID nativos
 IBM i  z/OS	1112, 1156
 Windows	901, 921, 1257, 5353, 9449
 AIX  Linux	901, 921

Todas las plataformas admiten la conversión entre sus CCSID nativos y los CCSID nativos de otras plataformas, con las siguientes excepciones.

z/OS



Página de códigos:

1112

No se convierte a las páginas de códigos 901, 1156, 9449

1156

No se convierte a las páginas de códigos 901, 1156, 9449

IBM i



Página de códigos:

1112

No se convierte a la página de códigos 5353

1153

No se convierte a las páginas de códigos 921, 5353, 9449

Linux



Página de códigos:

902

No se convierte a las páginas de códigos 921, 1112, 1257, 9449

921

No se convierte a las páginas de códigos 901, 1156, 9449

Windows



Página de códigos:

901

No se convierte a las páginas de códigos 921, 1112, 1257, 9449

5355

No se convierte a la página de códigos 9449

9449

No se convierte a las páginas de códigos 901, 921, 1112, 1156, 1257

Ucraniano

Detalles sobre los CCSID y la conversión CCSID para el ucraniano.

Tabla 693. CCSID nativos para ucraniano en plataformas compatibles	
Plataforma	CCSID nativos
 IBM i	1123
 z/OS	
 Windows	1124, 1125, 1251, 5347
 AIX	1124
 Linux	

Todas las plataformas admiten la conversión entre sus CCSID nativos y los CCSID nativos de otras plataformas, con las siguientes excepciones.

IBM i



Página de códigos:

1123

No se convierte a la página de códigos 5347

Windows



Página de códigos:

1125

No se convierte a la página de códigos 1123

Griego

Detalles sobre los CCSID y la conversión CCSID para el griego.

Tabla 694. CCSID nativos para griego en plataformas compatibles	
Plataforma	CCSID nativos
 IBM i	875
 z/OS	
 Windows	869, 1253, 5349

Tabla 694. CCSID nativos para griego en plataformas compatibles (continuación)

Plataforma	CCSID nativos
 AIX  Linux NCR	813
Cliente de Apple	1280
Cliente DOS	737

Todas las plataformas que no son de cliente admiten la conversión entre sus CCSID nativos y los CCSID nativos de las demás plataformas, con las siguientes excepciones.

IBM i



Página de códigos:

875

No se convierte a la página de códigos 5349

Windows



Página de códigos:

1253

No se convierte a la página de códigos 737

5349

No se convierte a la página de códigos 737

Turco

Detalles sobre los CCSID y la conversión CCSID para el turco.

Tabla 695. CCSID nativos para turco en plataformas compatibles

Plataforma	CCSID nativos
 IBM i  z/OS	1026
 Windows	857, 1254, 5350
 AIX  Linux	920
Cliente de Apple	1281

Todas las plataformas que no son de cliente admiten la conversión entre sus CCSID nativos y los CCSID nativos de las demás plataformas, con las siguientes excepciones.

IBM i



Página de códigos:

1026

No se convierte a la página de códigos 5350

Hebreo

Detalles sobre los CCSID y la conversión CCSID para el hebreo.

Tabla 696. CCSID nativos para hebreo en plataformas compatibles	
Plataforma	CCSID nativos
 z/OS	424, 803, 4899, 12712
 IBM i	424
 AIX	916, 9048
 Windows	1255, 5351
 Linux	916

Todas las plataformas admiten la conversión entre sus CCSID nativos y los CCSID nativos de otras plataformas, con las siguientes excepciones.

z/OS



Página de códigos:

424

No se convierte a las páginas de códigos 867, 4899, 9048, 12712

803

No se convierte a las páginas de códigos 867, 4899, 5351, 9048, 12712

4899

No se convierte a las páginas de códigos 424, 803, 856, 862, 916, 1255

12712

No se convierte a las páginas de códigos 424, 803, 856, 916, 1255

IBM i



Página de códigos:

424

No se convierte a las páginas de códigos 803, 867, 4899, 5351, 9048, 12712

La página de códigos 424 también convierte a y desde CCSID 4952, que es una variante de 856.

AIX



Página de códigos:

916

No se convierte a las páginas de códigos 867, 4899, 9048, 12712

9048

No se convierte a las páginas de códigos 424, 803, 856, 862, 916, 1255

Windows



Página de códigos:

1255

No se convierte a las páginas de códigos 867, 4899, 9048, 12712

5351

No se convierte a la página de códigos 803

Árabe

Detalles sobre los CCSID y la conversión CCSID para el árabe

Tabla 697. CCSID nativos para árabe en plataformas compatibles	
Plataforma	CCSID nativos
 IBM i	420
 z/OS	
 AIX	1046, 1089
 Windows	720, 864, 1256, 5352
 Linux	1089

Todas las plataformas admiten la conversión entre sus CCSID nativos y los CCSID nativos de otras plataformas, con las siguientes excepciones.

IBM i



Página de códigos:

420

No se convierte a la página de códigos 5352

Linux, Tru64



Página de códigos:

1089

No se convierte a la página de códigos 720

Windows



Página de códigos:

720

No se convierte a las páginas de códigos 1089 y 5352

5352

No se convierte a la página de códigos 720

Persa

Detalles sobre los CCSID y la conversión CCSID para el farsi.

Tabla 698. CCSID nativos para farsi en plataformas compatibles	
Plataforma	CCSID nativos
 IBM i  z/OS	1097
 AIX  Linux  Windows	1098 (véase la nota)

Nota: El CCSID nativo para estas plataformas no se ha estandarizado y podría cambiar.

Todas las plataformas admiten la conversión entre sus CCSID nativos y los CCSID nativos de otras plataformas.

Urdú

Detalles sobre los CCSID y la conversión CCSID para el urdu.

Tabla 699. CCSID nativos para urdu en plataformas compatibles	
Plataforma	CCSID nativos
 IBM i  z/OS	918
 Windows	868
 AIX  Linux	1006

Todas las plataformas admiten la conversión entre sus CCSID nativos y los CCSID nativos de otras plataformas, con las siguientes excepciones.

IBM i



Página de códigos:

918

No se convierte a la página de códigos 1006

Tailandés

Detalles sobre los CCSID y la conversión CCSID para tailandés.

Tabla 700. CCSID nativos para tailandés en plataformas compatibles

Plataforma	CCSID nativos
 IBM i  z/OS	838
 AIX  Linux  Windows	874 (véase la nota)

Nota: El CCSID nativo para estas plataformas no se ha estandarizado y podría cambiar.

Todas las plataformas admiten la conversión entre sus CCSID nativos y los CCSID nativos de otras plataformas.

lao

Detalles sobre los CCSID y la conversión CCSID para el laosiano.

Tabla 701. CCSID nativos para laosiano en plataformas compatibles

Plataforma	CCSID nativos
 IBM i  z/OS	1132
 AIX  Linux  Windows	1133

Todas las plataformas admiten la conversión entre sus CCSID nativos y los CCSID nativos de otras plataformas.

Vietnamita

Detalles sobre los CCSID y la conversión CCSID para el vietnamita.

Tabla 702. CCSID nativos para vietnamita en plataformas compatibles

Plataforma	CCSID nativos
 IBM i  z/OS	1130
 Windows	1258, 5354
 AIX  Linux	1129

Todas las plataformas admiten la conversión entre sus CCSID nativos y los CCSID nativos de otras plataformas, con las siguientes excepciones.

IBM i



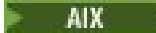
Página de códigos:

1130

No se convierte a las páginas de códigos 1129 y 5354

Japonés Latín SBCS

Detalles sobre los CCSID y la conversión CCSID para japonés latino SBCS.

Tabla 703. CCSID nativos para japonés latino SBCS en plataformas compatibles	
Plataforma	CCSID nativos
 IBM i	1027
 z/OS	
 AIX	932, 5050, 33722 (véase la nota 1)
 Windows	932, 943 (véase la nota 2)
 Linux	943, 5050

Nota:

-  5050 y 33722 son CCSID relacionados con la página de códigos base 954 en AIX. El CCSID informado por el sistema operativo es 33722.
-  Windows NT utiliza la página de códigos 932, pero se representa mejor con el CCSID 943. Sin embargo, no todas las plataformas IBM MQ admiten este CCSID.

En IBM MQ for Windows CCSID 932 se utiliza para representar la página de códigos 932, pero se puede realizar un `./conv/table/ccsid.tbl` cambio en el archivo que modifique el CCSID utilizado a 943.

Todas las plataformas admiten la conversión entre sus CCSID nativos y los CCSID nativos de otras plataformas, con las siguientes excepciones.

z/OS



Página de códigos:

1027

No se convierte a las páginas de códigos 932, 942, 943, 954, 5050, 33722

IBM i



Página de códigos:

1027

No se convierte a la página de códigos 932

AIX



Página de códigos:

932

No se convierte a la página de códigos 1027

5050

No se convierte a la página de códigos 1027

33722

No se convierte a la página de códigos 1027

Linux



Página de códigos:

943

No se convierte a la página de códigos 1027

5050

No se convierte a la página de códigos 1027

Katakana japonés SBCS

Detalles sobre los CCSID y la conversión CCSID para el katakana japonés SBCS.

Tabla 704. CCSID nativos para Katakana SBCS japonés en plataformas compatibles	
Plataforma	CCSID nativos
IBM i z/OS	290
AIX	932, 5050, 33722 (véase la nota 1)
Windows	932, 943 (véase la nota 2)
Linux	943, 5050

Nota:

- 5050 y 33722 son CCSID relacionados con la página de códigos base 954 en AIX. El CCSID informado por el sistema operativo es 33722.
- Windows NT utiliza la página de códigos 932, pero se representa mejor con el CCSID 943. Sin embargo, no todas las plataformas IBM MQ admiten este CCSID.
En IBM MQ for Windows CCSID 932 se utiliza para representar la página de códigos 932, pero se puede realizar un `.. /conv/table/ccsid.tbl` cambio en el archivo que modifique el CCSID utilizado a 943.
- Además de las conversiones anteriores, IBM MQ admite la conversión de CCSID 897 a CCSID 37, 273, 277, 278, 280, 284, 285, 290, 297, 437, 500, 819, 850, 1027 y 1252 en las siguientes plataformas:
 - AIX
 - Linux

Todas las plataformas admiten la conversión entre sus CCSID nativos y los CCSID nativos de otras plataformas, con las siguientes excepciones.

z/OS



Página de códigos:

290

No se convierte a las páginas de códigos 932, 943, 954, 5050, 33722

IBM i



Página de códigos:

290

No se convierte a la página de códigos 932

AIX



Página de códigos:

932

No se convierte a las páginas de códigos 290 y 897

5050

No se convierte a las páginas de códigos 290 y 897

33722

No se convierte a las páginas de códigos 290 y 897

Linux



Página de códigos:

943

No se convierte a las páginas de códigos 290 y 897

5050

No se convierte a las páginas de códigos 290 y 897

Kanji japonés/Latín mixto

Detalles sobre los CCSID y la conversión CCSID para japonés kanji/latino mixto.

Tabla 705. CCSID nativos para kanji japonés/latín mixto en plataformas compatibles	
Plataforma	CCSID nativos
IBM i z/OS	1399, 5035 (“1” en la página 970)
AIX	932, 5050, 33722 (“2” en la página 971)
Windows	932, 943 (“3” en la página 971)
Linux	943, 5050

Nota:

1. 5035 es un CCSID relacionado con la página de códigos 939

2.  5050 y 33722 son CCSID relacionados con la página de códigos base 954 en AIX. El CCSID informado por el sistema operativo es 33722.
3.  Windows NT utiliza la página de códigos 932, pero se representa mejor con el CCSID 943. Sin embargo, no todas las plataformas IBM MQ admiten este CCSID.

En IBM MQ for Windows CCSID 932 se utiliza para representar la página de códigos 932, pero se puede realizar un `./conv/table/ccsid.tbl` cambio en el archivo que modifique el CCSID utilizado a 943.

Todas las plataformas admiten la conversión entre sus CCSID nativos y los CCSID nativos de otras plataformas, con las siguientes excepciones.

z/OS



Página de códigos:

1399

No se convierte a las páginas de códigos 954, 5035, 5050, 33722

5035

No se convierte a las páginas de códigos 954, 1399, 5050, 33722

IBM i



Página de códigos:

1399

No se convierte a la página de códigos 5039

5035

No se convierte a la página de códigos 5039

Kanji japonés/ Katakana mixto

Detalles sobre los CCSID y la conversión CCSID para kanji/katakana mixtos japoneses.

Tabla 706. CCSID nativos para kanji japonés/katakana mixtos en plataformas compatibles	
Plataforma	CCSID nativos
 z/OS	1390, 5026 (véase la nota “1” en la página 971)
 IBM i	5026 (véase la nota “1” en la página 971)
 AIX	932, 5050, 33722 (véase la nota “2” en la página 972)
 Windows	932, 943 (véase la nota “3” en la página 972)
 Linux	943, 5050

Nota:

1.   El modo de un solo byte de los CCSID 1390 y 5026 en EBCDIC contiene caracteres en minúscula en ubicaciones diferentes a la disposición típica o invariable del alfabeto latino básico. Por lo tanto, debe asegurarse de que no se pierdan datos cuando los datos de los mensajes se conviertan a otros CCSID. Además, el uso de estos CCSID como CCSID predeterminado

de un gestor de colas puede causar problemas al comunicarse con otros gestores de colas. Por ejemplo, es posible que los nombres de canal que utilizan caracteres en minúscula no se interpreten correctamente en el sistema remoto. 5026 es un CCSID relacionado con la página de códigos 930. CCSID 5026 es el CCSID que se informa IBM i cuando se selecciona la función Katakana japonés (DBCS).

2.  5050 y 33722 son CCSID relacionados con la página de códigos base 954 en AIX. El CCSID informado por el sistema operativo es 33722.

3.  Windows NT utiliza la página de códigos 932, pero se representa mejor con el CCSID 943. Sin embargo, no todas las plataformas IBM MQ admiten este CCSID.

En Windows, CCSID 932 se utiliza para representar la página de códigos 932, pero se puede realizar un `../conv/table/ccsid.tbl` cambio en el archivo para cambiar el CCSID utilizado a 943.

Todas las plataformas admiten la conversión entre sus CCSID nativos y los CCSID nativos de otras plataformas, con las siguientes excepciones.

z/OS



Página de códigos:

1390

No se convierte a las páginas de códigos 954, 5026, 5050, 33722

No acepta caracteres en minúscula.

5026

No se convierte a las páginas de códigos 954, 1390, 5050, 33722

IBM i



Página de códigos:

5026

No se convierte a las páginas de códigos 1390 y 5039

Coreano

Detalles sobre los CCSID y la conversión CCSID para el coreano.

Tabla 707. CCSID nativos para coreano en plataformas compatibles	
Plataforma	CCSID nativos
 IBM i  z/OS	933, 1364
 AIX  Linux	970
 Windows	949, 1363

Todas las plataformas admiten la conversión entre sus CCSID nativos y los CCSID nativos de otras plataformas, con las siguientes excepciones.

z/OS



Página de códigos:

933

No se convierte a la página de códigos 970

1364

No se convierte a la página de códigos 970

Chino simplificado

Detalles sobre los CCSID y la conversión CCSID para el chino simplificado.

Tabla 708. CCSID nativos para chino simplificado en plataformas compatibles	
Plataforma	CCSID nativos
z/OS	935, 1388
IBM i	935, 1388
AIX	1383, 1386
Windows	1381, 1386 (véase la nota 2)
Linux	1383

Nota:

- Windows utiliza la página de códigos 936, pero esta se representa mejor con el CCSID 1386. Sin embargo, no todas las plataformas IBM MQ admiten este CCSID.

En IBM MQ for Windows CCSID 1381 se utiliza para representar la página de códigos 936, pero se puede realizar un `./conv/table/ccsid.tbl` cambio en el archivo que cambie el CCSID utilizado a 1386.
- IBM MQ Es compatible con el estándar chino GB18030.

En z/OS, Windows y Linux, se proporciona soporte de conversión entre Unicode (UTF-8 y UTF-16) y CCSID 1388 (EBCDIC con extensiones GB18030), Unicode (UTF-8 y UTF-16) y CCSID 5488 (GB18030), y entre CCSID 1388 y CCSID 5488.

Nota: En multiplataformas, el CCSID debe establecerse en 5488 para poder utilizar caracteres GB18030 en los campos de descripción de objetos como colas y temas. Sin embargo, no es posible configurar el CCSID en un gestor de colas creado con el IBM MQ Explorer o el IBM MQ Console. En su lugar, debe crear el gestor de colas utilizando la CLI con un CCSID de 5488 o utilizar la línea de comandos de la CLI para cambiar el CCSID después de crear el gestor de colas.

El sistema operativo proporciona soporte para la conversión entre Unicode (IBM i UTF-8 y UTF-16) y CCSID 1388 (EBCDIC con extensiones GB18030).

Para poder utilizar caracteres de GB18030 en z/OS los campos de descripción de objetos como colas y temas, el gestor de colas debe estar configurado para ejecutarse con un valor `QMCCSID` de 1388.

En z/OS, desde IBM MQ 10.0.0, se proporciona soporte de conversión entre caracteres de nivel 1 y nivel 2 de GB18030-2022 (CCSID 1388) y Unicode (UTF-8, CCSID 1208, y UTF-16, CCSID 1200).

Todas las plataformas admiten la conversión entre sus CCSID nativos y los CCSID nativos de otras plataformas, con las siguientes excepciones.

z/OS



Página de códigos:

935

No se convierte a la página de códigos 1383

1388

No se convierte a la página de códigos 1383

Chino tradicional

Detalles sobre los CCSID y la conversión CCSID para el chino tradicional.

Tabla 709. CCSID nativos para chino tradicional en plataformas compatibles	
Plataforma	CCSID nativos
IBM i z/OS	937
Windows	950
AIX Linux	950, 964

Todas las plataformas admiten la conversión entre sus CCSID nativos y los CCSID nativos de otras plataformas, con las siguientes excepciones.

z/OS



Página de códigos:

937

No se convierte a la página de códigos 964

Linux



Página de códigos:

964

No se convierte a la página de códigos 938

z/OS soporte para conversión

Información sobre las conversiones CCSID compatibles, incluidos los CCSID entre los que se pueden convertir datos y algunas de las diferencias de caracteres que pueden encontrarse

Conversiones compatibles

La siguiente tabla asigna cada CCSID a los demás CCSID a los que se puede convertir o desde los que se puede convertir.

Nota: Algunos CCSID ya no existen en ciertas Java versiones. Para obtener más información, consulte los CSSID que ya no son compatibles.

Tabla 710. IBM MQ for z/OS Compatibilidad con la conversión CCSID	
CCSID	Convierte a y desde CCSIDS
37	256, 273, 275, 277-278, 280, 284-285, 290, 297, 367, 420, 423-424, 437, 500, 720, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-866, 869-871, 874-875, 880, 897, 903-905, 912, 914-916, 920-924, 1009, 1025-1027, 1040-1043, 1047, 1051, 1088, 1097, 1100, 1112, 1114-1115, 1122, 1124, 1126, 1130-1132, 1137, 1140-1149, 1200, 1208, 1250-1255, 1257-1258, 1275, 1280-1281, 1283, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5210-5211, 5346, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 16804, 17248, 17584, 25473, 25479, 25480, 25617, 25619, 25664, 28709
256	37, 273, 277-278, 280, 284-285, 290, 297, 367, 420, 423-424, 437, 500, 737, 775, 819, 833, 836, 838, 850, 852, 857, 860-866, 869-871, 875, 880, 905, 1025-1027, 1112, 1122, 1200, 1208, 1251-1252, 1275, 4386, 4929, 4932, 4934, 4946, 4948, 4953, 4960, 4971, 5123, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9056, 9061, 13121, 13488, 16804, 17248, 17584, 28709
259	437, 808, 850-852, 855-858, 860-865, 867, 869, 872, 874, 899, 901-902, 915, 1098, 1161-1162, 1200, 1208, 1250-1258, 4946, 4948, 4951-4953, 4960, 4970, 5346, 5348, 9044, 9049, 9056, 9061, 9066, 13488, 17248, 17584
273	37, 256, 277-278, 280, 284-285, 290, 297, 367, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855-858, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 923-924, 1009, 1025-1027, 1040-1043, 1047, 1051, 1088, 1100, 1112, 1122, 1140-1149, 1200, 1208, 1250, 1252, 1275, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951-4953, 4960, 4970-4971, 5012, 5123, 5346, 5348, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
274	500, 1047
275	37, 437, 500, 819, 850, 1047, 1200, 1208, 1252, 4946, 5348, 8229, 13488, 17584, 28709
277	37, 256, 273, 278, 280, 284-285, 290, 297, 367, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 923-924, 1009, 1025-1027, 1040-1043, 1047, 1051, 1088, 1100, 1112, 1122, 1140-1149, 1200, 1208, 1252, 1275, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5348, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
278	37, 256, 273, 277, 280, 284-285, 290, 297, 367, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 923-924, 1009, 1025-1027, 1040-1043, 1047, 1051, 1088, 1100, 1112, 1122, 1140-1149, 1200, 1208, 1252, 1275, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5348, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709

Tabla 710. IBM MQ for z/OS Compatibilidad con la conversión CCSID (continuación)	
CCSID	Convierte a y desde CCSIDS
280	37, 256, 273, 277-278, 284-285, 290, 297, 367, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 923-924, 1009, 1025-1027, 1040-1043, 1047, 1051, 1088, 1100, 1112, 1122, 1140-1149, 1200, 1208, 1252, 1275, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5348, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
281	1047
282	500, 1047, 1200, 1208, 13488, 17584
284	37, 256, 273, 277-278, 280, 285, 290, 297, 367, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 923-924, 1009, 1025-1027, 1040-1043, 1047, 1051, 1088, 1100, 1112, 1122, 1140-1149, 1200, 1208, 1252, 1275, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5348, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
285	37, 256, 273, 277-278, 280, 284, 290, 297, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 923-924, 1025-1027, 1040-1043, 1047, 1051, 1088, 1100, 1112, 1122, 1140-1149, 1200, 1208, 1252, 1275, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5348, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
290	37, 256, 273, 277-278, 280, 284-285, 297, 367, 437, 500, 737, 775, 819, 833, 836, 850, 852, 855, 857, 860-865, 870-871, 895-897, 1009, 1025-1027, 1040-1043, 1047, 1088, 1112, 1122, 1139, 1200, 1208, 1252, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 4960, 4992, 5123, 8229, 8482, 9025, 9044, 9049, 9056, 13121, 13488, 17248, 17584, 25473, 25617, 25619, 25664, 28709
293	1200, 1208, 13488, 17584
297	37, 256, 273, 277-278, 280, 284-285, 290, 367, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 923-924, 1009, 1025-1027, 1040-1043, 1047, 1051, 1088, 1100, 1112, 1122, 1140-1149, 1200, 1208, 1252, 1275, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5348, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
300	301, 941, 1200, 1208, 1351, 4396, 8492, 13488, 16684, 17584
301	300, 941, 1200, 1208, 1351, 4396, 8492, 13488, 16684, 17584
367	37, 256, 273, 277-278, 280, 284, 290, 297, 500, 819, 833, 836, 850, 871, 875, 1009, 1026-1027, 1041, 1088, 1115, 1126, 1200, 1208, 4386, 4929, 4932, 4946, 4971, 5123, 5211, 8229, 8482, 9025, 13121, 13488, 17584, 25617, 25664, 28709

Tabla 710. IBM MQ for z/OS Compatibilidad con la conversión CCSID (continuación)	
CCSID	Convierte a y desde CCSIDS
420	37, 256, 424, 437, 500, 720, 737, 775, 819, 850, 852, 857, 860-865, 1008, 1046, 1089, 1098, 1112, 1122, 1127, 1200, 1208, 1252, 1256, 4946, 4948, 4953, 4960, 5104, 5142, 5352, 8229, 8612, 9044, 9049, 9056, 9238, 13488, 16804, 17248, 17584, 28709
423	37, 256, 273, 277-278, 280, 284-285, 297, 437, 500, 737, 775, 813, 819, 838, 850-852, 857, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1009, 1025-1027, 1041-1043, 1112, 1122, 1200, 1208, 1252-1253, 1280, 4909, 4934, 4946, 4948, 4953, 4960, 4970-4971, 5012, 5123, 8229, 9030, 9044, 9049, 9056, 9061, 9066, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 28709
424	37, 256, 420, 437, 500, 737, 775, 803, 819, 836, 850, 852, 856-857, 860-865, 916, 1112, 1122, 1200, 1208, 1252, 1255, 4932, 4946, 4948, 4952-4953, 4960, 5012, 5351, 8229, 8612, 9044, 9049, 9056, 13488, 16804, 17248, 17584, 28709
437	37, 256, 259, 273, 275, 277-278, 280, 284-285, 290, 297, 420, 423-424, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-863, 865-866, 869-871, 874-875, 880, 897, 903, 905, 912, 914-916, 920-924, 1025-1027, 1040-1043, 1047, 1051, 1097, 1098, 1114-1115, 1126, 1140-1149, 1200, 1208, 1252, 1257, 1275, 1280-1281, 1283, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4970-4971, 5012, 5123, 5210-5211, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 13488, 16804, 17584, 25473, 25479, 25617, 25619, 28709
500	37, 256, 273-275, 277-278, 280, 282, 284-285, 290, 297, 367, 420, 423-424, 437, 737, 775, 813, 819, 833, 836, 838, 850-852, 855-858, 860-866, 869-871, 874-875, 880, 891, 895, 897, 903-905, 912, 914-916, 920-924, 1004, 1009-1021, 1023, 1025-1027, 1040-1043, 1046-1047, 1051, 1088-1089, 1097, 1100-1107, 1112, 1114-1115, 1122, 1124-1126, 1129-1133, 1137, 1140-1149, 1200, 1208, 1250-1258, 1275, 1280-1283, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951-4953, 4960, 4970-4971, 5012, 5123, 5142, 5210-5211, 5346, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 9238, 13121, 13488, 16804, 17248, 17584, 25473, 25479, 25480, 25617, 25619, 25664, 28709
720	37, 420, 864, 1200, 1208, 1256, 4960, 8229, 8612, 9056, 13488, 16804, 17248, 17584, 28709
737	37, 256, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 833, 836, 838, 850, 869-871, 875, 880, 905, 1025-1027, 1097, 1200, 1208, 1252-1253, 1280, 4386, 4909, 4929, 4932, 4934, 4946, 4971, 5123, 8229, 8482, 8612, 9025, 9030, 9061, 13121, 13488, 16804, 17584, 28709
775	37, 256, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 833, 836, 838, 850, 870-871, 875, 880, 905, 1025-1027, 1097, 1112, 1122, 1200, 1208, 1252, 1257, 4386, 4929, 4932, 4934, 4946, 4971, 5123, 8229, 8482, 8612, 9025, 9030, 13121, 13488, 16804, 17584, 28709
803	424, 819, 850, 856, 862, 916, 1200, 1208, 1252, 1255, 4946, 4952, 5012, 13488, 17584
806	1200, 1208, 13488, 17584
808	259, 858-859, 872, 923-924, 1140, 1148, 1153-1154, 1200, 1208, 5347, 5348, 13488, 17584

Tabla 710. IBM MQ for z/OS Compatibilidad con la conversión CCSID (continuación)	
CCSID	Convierte a y desde CCSIDS
813	37, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 737, 819, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1200, 1208, 1252-1253, 1280, 4909, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 5349, 8229, 9030, 9044, 9049, 9061, 9066, 13488, 17584, 25473, 25479, 25617, 25619, 28709
819	37, 256, 273, 275, 277-278, 280, 284-285, 290, 297, 367, 420, 423-424, 437, 500, 803, 813, 833, 836, 838, 850, 852, 855, 857-858, 860-861, 863-866, 869-871, 874-875, 880, 897, 903, 905, 912, 914-916, 920-924, 1004, 1025-1027, 1041-1043, 1047, 1051, 1088-1089, 1097, 1098, 1112, 1114, 1122-1123, 1126, 1130, 1132, 1137, 1140-1149, 1200, 1208, 1250-1255, 1257-1258, 1275, 1280-1281, 1283, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5210, 5346, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 16804, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
833	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 437, 500, 737, 775, 819, 836, 850, 852, 855, 857, 860-865, 870-871, 891, 1009, 1025-1027, 1040-1043, 1088, 1112, 1122, 1126, 1200, 1208, 1252, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 4960, 5123, 8229, 8482, 9025, 9044, 9049, 9056, 13121, 13488, 17248, 17584, 25617, 25619, 25664, 28709
834	926, 951, 1200, 1208, 1362, 4930, 9026, 13488, 17584
835	927, 947, 1200, 1208, 4931, 9027, 13488, 17584, 21427
836	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 424, 437, 500, 737, 775, 819, 833, 850, 852, 855, 857, 870-871, 875, 903, 1009, 1025-1027, 1040-1043, 1088, 1112, 1114-1115, 1122, 1200, 1208, 1252, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 4971, 5123, 5210-5211, 8229, 8482, 9025, 9044, 9049, 13121, 13488, 17584, 25479, 25617, 25619, 25664, 28709
837	928, 1200, 1208, 1380, 1385, 4933, 13488, 17584
838	37, 256, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 737, 775, 813, 819, 850, 852, 857, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1112, 1122, 1200, 1208, 1252, 4909, 4934, 4946, 4948, 4953, 4960, 4970-4971, 5012, 5123, 8229, 9030, 9044, 9049, 9056, 9061, 9066, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 28709
848	924, 1148, 1158, 1200, 1208, 5347, 13488, 17584
849	924, 1148, 1154, 1200, 1208, 5347, 13488, 17584
850	37, 256, 259, 273, 275, 277-278, 280, 284-285, 290, 297, 367, 420, 423-424, 437, 500, 737, 775, 803, 813, 819, 833, 836, 838, 852, 855-858, 860-866, 869-871, 874-875, 880, 897, 903, 905, 912, 914-916, 920-924, 1004, 1025-1027, 1040-1043, 1047, 1051, 1088-1089, 1097, 1098, 1100, 1112, 1114, 1122, 1126, 1130, 1132, 1140-1149, 1200, 1208, 1250-1257, 1275, 1280-1281, 1283, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951-4953, 4960, 4970-4971, 5012, 5123, 5210, 5346, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 16804, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
851	259, 423, 500, 875, 1200, 1208, 4971, 13488, 17584

Tabla 710. IBM MQ for z/OS Compatibilidad con la conversión CCSID (continuación)	
CCSID	Convierte a y desde CCSIDS
852	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 819, 833, 836, 838, 850, 855, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 1025-1027, 1040-1043, 1047, 1088, 1097, 1200, 1208, 1250, 1252, 1282, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4970-4971, 5012, 5123, 5346, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 13488, 16804, 17584, 25473, 25479, 25617, 25619, 25664, 28709
855	37, 259, 273, 277-278, 280, 284-285, 290, 297, 437, 500, 819, 833, 836, 850, 852, 857, 866, 870-871, 878, 880, 912, 915, 1025-1027, 1040-1043, 1088, 1200, 1208, 1250-1252, 1283, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 5123, 5346, 5347, 8229, 8482, 9025, 9044, 9049, 13121, 13488, 17584, 25617, 25619, 25664, 28709
856	259, 273, 424, 500, 803, 850, 862, 916, 1200, 1208, 1255, 4946, 4952, 5012, 5351, 13488, 17584
857	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 860-861, 863, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 1025-1027, 1040-1043, 1088, 1097, 1200, 1208, 1252, 1254, 1281, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4970-4971, 5012, 5123, 5350, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 13488, 16804, 17584, 25473, 25479, 25617, 25619, 25664, 28709
858	37, 259, 273, 277-278, 280, 284-285, 297, 437, 500, 808, 819, 850, 860-861, 865, 871-872, 901-902, 923-924, 1047, 1051, 1140-1149, 1153-1157, 1160-1162, 1164, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
859	808, 872, 901-902, 1153-1157, 1160-1162, 1164, 1200, 1208, 13488, 17584
860	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 819, 833, 838, 850, 852, 857-858, 861, 863, 865, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 923-924, 1025-1027, 1041-1043, 1097, 1140, 1145-1146, 1148, 1200, 1208, 1252, 4386, 4909, 4929, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 13488, 16804, 17584, 25473, 25479, 25617, 25619, 28709
861	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 819, 833, 838, 850, 852, 857-858, 860, 863, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 923-924, 1025-1027, 1041-1043, 1097, 1148, 1149, 1200, 1208, 1252, 4386, 4909, 4929, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 13488, 16804, 17584, 25473, 25479, 25617, 25619, 28709
862	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 803, 833, 838, 850, 856, 870-871, 875, 880, 905, 916, 1025-1027, 1097, 1200, 1208, 1252, 1255, 4386, 4929, 4934, 4946, 4952, 4971, 5012, 5123, 5351, 8229, 8482, 8612, 9025, 9030, 12712, 13121, 13488, 16804, 17584, 28709

Tabla 710. IBM MQ for z/OS Compatibilidad con la conversión CCSID (continuación)	
CCSID	Convierte a y desde CCSIDS
863	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 819, 833, 838, 850, 852, 857, 860-861, 865, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 1025-1027, 1041-1043, 1051, 1097, 1140-1149, 1200, 1208, 1252, 1275, 4386, 4909, 4929, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 13488, 16804, 17584, 25473, 25479, 25617, 25619, 28709
864	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 500, 720, 819, 833, 838, 850, 870-871, 875, 880, 905, 918, 1008, 1025-1027, 1046, 1089, 1097, 1127, 1200, 1208, 1252, 1256, 4386, 4929, 4934, 4946, 4960, 4971, 5104, 5123, 5142, 5352, 8229, 8482, 8612, 9025, 9030, 9056, 9238, 13121, 13488, 16804, 17248, 17584, 28709
865	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 819, 833, 838, 850, 858, 860, 863, 870-871, 875, 880, 905, 923-924, 1025-1027, 1097, 1142-1143, 1148, 1200, 1208, 1252, 4386, 4929, 4934, 4946, 4971, 5123, 5348, 8229, 8482, 8612, 9025, 9030, 13121, 13488, 16804, 17584, 28709
866	37, 256, 437, 500, 819, 850, 855, 870, 878, 880, 915, 1025, 1200, 1208, 1251-1252, 1283, 4946, 4951, 5347, 8229, 13488, 17584, 28709
867	259, 1153-1155, 1160, 1200, 1208, 4899, 5351, 9048, 12712, 13488, 17584
868	918, 1006, 1200, 1208, 13488, 17584
869	37, 256, 259, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 737, 813, 819, 838, 850, 852, 857, 860-861, 863, 870-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1200, 1208, 1252-1254, 1280, 4909, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 5349, 8229, 9030, 9044, 9049, 9061, 9066, 13488, 17584, 25473, 25479, 25617, 25619, 28709
870	37, 256, 273, 277-278, 280, 284-285, 290, 297, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-866, 869, 871, 874-875, 880, 897, 903, 912, 915-916, 920, 1009, 1025-1027, 1040-1043, 1047, 1088, 1112, 1122, 1200, 1208, 1250, 1252, 1282, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5346, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
871	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-865, 869, 870, 874-875, 880, 897, 903, 912, 916, 920, 923-924, 1009, 1025-1027, 1040-1043, 1047, 1051, 1088, 1112, 1122, 1140-1149, 1200, 1208, 1252, 1275, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5348, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
872	259, 808, 858-859, 923-924, 1140-1149, 1153-1155, 1200, 1208, 5347, 5348, 13488, 17584

Tabla 710. IBM MQ for z/OS Compatibilidad con la conversión CCSID (continuación)	
CCSID	Convierte a y desde CCSIDS
874	37, 259, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 813, 819, 838, 850, 852, 857, 860-861, 863, 869-871, 875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1200, 1208, 1252, 4909, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 8229, 9030, 9044, 9049, 9061, 9066, 13488, 17584, 25473, 25479, 25617, 25619, 28709
875	37, 256, 273, 277-278, 280, 284-285, 297, 367, 423, 437, 500, 737, 775, 813, 819, 836, 838, 850-852, 857, 860-865, 869-871, 874, 880, 897, 903, 912, 916, 920, 1009, 1025-1027, 1041-1043, 1047, 1088, 1112, 1122, 1200, 1208, 1252-1253, 1280, 4909, 4932, 4934, 4946, 4948, 4953, 4960, 4970-4971, 5012, 5123, 5349, 8229, 9030, 9044, 9049, 9056, 9061, 9066, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
878	855, 866, 880, 915, 1025, 1131, 1200, 1208, 1251, 1283, 4951, 5347, 13488, 17584
880	37, 256, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 737, 775, 813, 819, 838, 850, 852, 855, 857, 860-866, 869-871, 874-875, 878, 897, 903, 912, 915-916, 920, 1009, 1025-1027, 1041-1043, 1112, 1122, 1200, 1208, 1251-1252, 1283, 4909, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5347, 8229, 9030, 9044, 9049, 9056, 9061, 9066, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 28709
891	500, 833, 1088, 1200, 1208, 4929, 9025, 13121, 13488, 17584, 25664
895	290, 500, 1027, 1041, 1200, 1208, 4386, 5123, 8482, 13488, 17584, 25617
896	290, 1027, 1041, 1200, 1208, 4386, 4992, 5123, 8482, 13488, 17584, 25617
897	37, 273, 277-278, 280, 284-285, 290, 297, 423, 437, 500, 813, 819, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 903, 912, 916, 920, 1025-1027, 1041-1043, 1200, 1208, 1252, 4386, 4909, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 8229, 8482, 9030, 9044, 9049, 9061, 9066, 13488, 17584, 25473, 25479, 25617, 25619, 28709
899	259
901	259, 858-859, 902, 923-924, 1140, 1148, 1156-1157, 1200, 1208, 5348, 5353, 13488, 17584
902	259, 858-859, 901, 923-924, 1140, 1148, 1156-1157, 1200, 1208, 5348, 5353, 13488, 17584
903	37, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 813, 819, 836, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 897, 912, 916, 920, 1025-1027, 1041-1043, 1115, 1200, 1208, 1252, 4909, 4932, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 5211, 8229, 9030, 9044, 9049, 9061, 9066, 13488, 17584, 25473, 25479, 25617, 25619, 28709
904	37, 500, 1114, 1200, 1208, 5210, 8229, 13488, 17584, 25480, 28709
905	37, 256, 437, 500, 737, 775, 819, 850, 852, 857, 860-865, 920, 1026, 1112, 1122, 1200, 1208, 1252, 1254, 1281, 4946, 4948, 4953, 4960, 8229, 9044, 9049, 9056, 13488, 17248, 17584, 28709

Tabla 710. IBM MQ for z/OS Compatibilidad con la conversión CCSID (continuación)	
CCSID	Convierte a y desde CCSIDS
912	37, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 813, 819, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 916, 920, 1025-1027, 1041-1043, 1047, 1200, 1208, 1250, 1252, 1282, 4909, 4934, 4946, 4948, 4951, 4953, 4970-4971, 5012, 5123, 5346, 8229, 9030, 9044, 9049, 9061, 9066, 13488, 17584, 25473, 25479, 25617, 25619, 28709
914	37, 437, 500, 819, 850, 1200, 1208, 1252, 1257, 4946, 8229, 13488, 17584, 28709
915	37, 259, 437, 500, 819, 850, 855, 866, 870, 878, 880, 1025, 1131, 1200, 1208, 1251-1252, 1283, 4946, 4951, 5347, 8229, 13488, 17584, 28709
916	37, 273, 277-278, 280, 284-285, 297, 423-424, 437, 500, 803, 813, 819, 838, 850, 852, 856-857, 860-863, 869-871, 874-875, 880, 897, 903, 912, 920, 1025-1027, 1041-1043, 1200, 1208, 1252, 1255, 4909, 4934, 4946, 4948, 4952-4953, 4970-4971, 5012, 5123, 5351, 8229, 9030, 9044, 9049, 9061, 9066, 13488, 17584, 25473, 25479, 25617, 25619, 28709
918	864, 868, 1006, 1200, 1208, 4960, 9056, 13488, 17248, 17584
920	37, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 813, 819, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 1025-1026, 1200, 1208, 1252, 1254, 1281, 4909, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5350, 8229, 9030, 9044, 9049, 9061, 9066, 13488, 17584, 25473, 25479, 28709
921	37, 437, 500, 819, 850, 922, 1112, 1122, 1200, 1208, 1252, 1257, 4946, 5353, 8229, 13488, 17584, 28709
922	37, 437, 500, 819, 850, 921, 1112, 1122, 1200, 1208, 1252, 1257, 4946, 5353, 8229, 13488, 17584, 28709
923	37, 273, 277-278, 280, 284-285, 297, 437, 500, 808, 819, 850, 858, 860-861, 865, 871-872, 901-902, 924, 1047, 1051, 1140-1149, 1153-1158, 1160-1162, 1164, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
924	37, 273, 277-278, 280, 284-285, 297, 437, 500, 808, 819, 848-850, 858, 860-861, 865, 871-872, 901-902, 923, 1047, 1051, 1140-1149, 1153-1157, 1160-1164, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
926	834, 951, 9026
927	835, 947, 1200, 1208, 4931, 9027, 13488, 17584, 21427
928	837, 1200, 1208, 1380, 13488, 17584
930	931-932, 939, 942-943, 1200, 1208, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 13488, 17314, 17584, 25508, 25518, 29614, 33698-33700, 37796
931	930, 932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
932	930-931, 939, 942-943, 1200, 1208, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 13488, 17314, 17584, 25508, 25518, 29614, 33698-33700, 37796

<i>Tabla 710. IBM MQ for z/OS Compatibilidad con la conversión CCSID (continuación)</i>	
CCSID	Convierte a y desde CCSIDS
933	934, 944, 949, 1200, 1208, 1363-1364, 5029, 5045, 5460, 9125, 9555, 13221, 13488, 13651, 17317, 17584, 25510, 25520, 25525, 29616, 29621, 33717, 37813
934	933, 949, 5029, 5045, 5460, 9125, 13221, 17317, 25510, 25525, 29621, 33717, 37813
935	936, 946, 1200, 1208, 1381, 1386, 1388, 5031, 5477, 5482, 5484, 9127, 13223, 13488, 17584, 25512
936	935, 946, 1381, 5031, 5477, 5484, 9127, 13223, 25512
937	938, 948, 950, 1200, 1208, 1370, 5033, 5046, 9142, 13488, 17584, 25514, 25524, 29620
938	937, 950, 1370, 5033, 5046, 9142, 25514
939	930-932, 942-943, 1200, 1208, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 13488, 17314, 17584, 25508, 25518, 29614, 33698-33700, 37796
941	300-301, 1200, 1208, 1351, 4396, 8492, 13488, 16684, 17584
942	930-932, 939, 943, 1200, 1208, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 13488, 17314, 17584, 25508, 25518, 29614, 33698-33700, 37796
943	930-932, 939, 942, 1200, 1208, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 13488, 17314, 17584, 25508, 25518, 29614, 33698-33700, 37796
944	933, 949, 1200, 1208, 5029, 5045, 5460, 9125, 13221, 13488, 17317, 17584, 25520, 25525, 29616, 29621, 33717, 37813
946	935-936, 1200, 1208, 5031, 5484, 9127, 13223, 13488, 17584, 25512
947	835, 927, 1200, 1208, 4931, 9027, 13488, 17584, 21427
948	937, 950, 1200, 1208, 1370, 5033, 5046, 9142, 13488, 17584, 25524, 29620
949	933-934, 944, 1200, 1208, 1363-1364, 5029, 5045, 5460, 9125, 9555, 13221, 13488, 13651, 17317, 17584, 25510, 25520, 25525, 29616, 29621, 33717, 37813
950	937-938, 948, 1200, 1208, 1370, 5033, 5046, 9142, 13488, 17584, 25514, 25524, 29620
951	834, 926, 1200, 1208, 1362, 4930, 9026, 13488, 17584
1004	500, 819, 850, 1200, 1208, 4946, 13488, 17584
1006	868, 918, 1200, 1208, 13 488, 17 584
1008	420, 864, 1200, 1208, 4960, 5104, 8612, 9056, 13488, 16804, 17248, 17584
1009	37, 273, 277-278, 280, 284, 290, 297, 367, 423, 500, 833, 836, 870-871, 875, 880, 1025-1026, 1200, 1208, 4386, 4929, 4932, 4971, 8229, 8482, 9025, 13121, 13488, 17584, 28709
1010	500, 1200, 1208, 13488, 17584

<i>Tabla 710. IBM MQ for z/OS Compatibilidad con la conversión CCSID (continuación)</i>	
CCSID	Convierte a y desde CCSIDS
1011	500, 1200, 1208, 13488, 17584
1.012	500, 1200, 1208, 13488, 17584
1013	500, 1140, 1200, 1208, 13488, 17584
1014	500, 1200, 1208, 13488, 17584
1015	500, 1200, 1208, 13488, 17584
1016	500, 1200, 1208, 13488, 17584
1017	500, 1200, 1208, 13488, 17584
1018	500, 1200, 1208, 13488, 17584
1019	500, 1200, 1208, 13488, 17584
1020	500
1021	500
1023	500
1025	37, 256, 273, 277-278, 280, 284-285, 290, 297, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-866, 869-871, 874-875, 878, 880, 897, 903, 912, 915-916, 920, 1009, 1026-1027, 1040-1043, 1051, 1088, 1112, 1122, 1131, 1200, 1208, 1251-1252, 1283, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5347, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
1026	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-865, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 1009, 1025, 1027, 1040-1043, 1047, 1088, 1112, 1122, 1200, 1208, 1252, 1254, 1281, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5350, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
1027	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-865, 869-871, 874-875, 880, 895-897, 903, 912, 916, 1025-1026, 1040-1043, 1047, 1088, 1112, 1122, 1139, 1200, 1208, 1252, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 4992, 5012, 5123, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
1040	37, 273, 277-278, 280, 284-285, 290, 297, 437, 500, 833, 836, 850, 852, 855, 857, 870-871, 1025-1027, 1041-1043, 1088, 1200, 1208, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 5123, 8229, 8482, 9025, 9044, 9049, 13121, 13488, 17584, 25617, 25619, 25664, 28709
1041	37, 273, 277-278, 280, 284-285, 290, 297, 367, 423, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 895-897, 903, 912, 916, 1025-1027, 1040, 1042-1043, 1088, 1200, 1208, 1252, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4970-4971, 4992, 5012, 5123, 8229, 8482, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 13488, 17584, 25473, 25479, 25617, 25619, 25664, 28709

<i>Tabla 710. IBM MQ for z/OS Compatibilidad con la conversión CCSID (continuación)</i>	
CCSID	Convierte a y desde CCSIDS
1042	37, 273, 277-278, 280, 284-285, 290, 297, 423, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 1025-1027, 1040, 1041, 1043, 1088, 1200, 1208, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4970-4971, 5012, 5123, 8229, 8482, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 13488, 17584, 25473, 25479, 25617, 25619, 25664, 28709
1043	37, 273, 277-278, 280, 284-285, 290, 297, 423, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 1025-1027, 1040, 1041, 1042, 1088, 1114, 1200, 1208, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4970-4971, 5012, 5123, 5210, 8229, 8482, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 13488, 17584, 25473, 25479, 25617, 25619, 25664, 28709
1046	420, 500, 864, 1089, 1127, 1200, 1208, 1256, 4960, 5142, 5352, 8612, 9056, 9238, 13 488, 16 804, 17 248, 17 584
1047	37, 273-275, 277-278, 280, 281, 282, 284-285, 290, 297, 437, 500, 819, 850, 852, 858, 870-871, 875, 912, 923-924, 1026-1027, 1140-1149, 1200, 1208, 1252, 1254, 4946, 4948, 5123, 8229, 8482, 13488, 17584, 28709
1051	37, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 863, 871, 923-924, 1025, 1097, 1140-1149, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
1088	37, 273, 277-278, 280, 284-285, 290, 297, 367, 500, 819, 833, 836, 850, 852, 855, 857, 870-871, 875, 891, 1025-1027, 1040-1043, 1126, 1200, 1208, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 4971, 5123, 8229, 8482, 9025, 9044, 9049, 13121, 13488, 17584, 25617, 25619, 25664, 28709
1089	420, 500, 819, 850, 864, 1046, 1127, 1200, 1208, 1256, 4946, 4960, 5142, 5352, 8612, 9056, 9238, 13488, 16804, 17248, 17584
1097	37, 437, 500, 737, 775, 819, 850, 852, 857, 860-865, 1051, 1098, 1112, 1122, 1200, 1208, 1252, 4946, 4948, 4953, 4960, 8229, 9044, 9049, 9056, 13488, 17248, 17584, 28709
1098	259, 420, 437, 819, 850, 1097, 1200, 1208, 1252, 4946, 8612, 13488, 16804, 17584
1100	37, 273, 277-278, 280, 284-285, 297, 500, 850, 4946, 8229, 28709
1101	500
1102	500
1103	500
1104	500
1105	500
1106	500
1107	500

<i>Tabla 710. IBM MQ for z/OS Compatibilidad con la conversión CCSID (continuación)</i>	
CCSID	Convierte a y desde CCSIDS
1112	37, 256, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 500, 775, 819, 833, 836, 838, 850, 870-871, 875, 880, 905, 921-922, 1025-1027, 1097, 1122, 1200, 1208, 1252, 1257, 4386, 4929, 4932, 4934, 4946, 4971, 5123, 5353, 8229, 8482, 8612, 9025, 9030, 13121, 13488, 16804, 17584, 28709
1114	37, 437, 500, 819, 836, 850, 904, 1043, 1115, 1200, 1208, 4932, 4946, 5210-5211, 8229, 13488, 17584, 25480, 25619, 28709
1115	37, 367, 437, 500, 836, 903, 1114, 1200, 1208, 4932, 5210-5211, 8229, 13488, 17584, 25479, 28709
1122	37, 256, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 500, 775, 819, 833, 836, 838, 850, 870-871, 875, 880, 905, 921-922, 1025-1027, 1097, 1112, 1200, 1208, 1252, 1257, 4386, 4929, 4932, 4934, 4946, 4971, 5123, 5353, 8229, 8482, 8612, 9025, 9030, 13121, 13488, 16804, 17584, 28709
1123	819, 1124-1125, 1148, 1200, 1208, 1251-1252, 1283, 5347, 13488, 17584
1124	37, 500, 1123, 1125, 1200, 1208, 1251, 1283, 5347, 8229, 13488, 17584, 28709
1125	500, 1123, 1124, 1200, 1208, 1251, 1283, 5347, 13488, 17584
1126	37, 367, 437, 500, 819, 833, 850, 1088, 1200, 1208, 1252, 4929, 4946, 8229, 9025, 13121, 13488, 17584, 25664, 28709
1127	420, 864, 1046, 1089, 1256, 4960, 5142, 8612, 9056, 9238, 16804, 17248
1129	500, 1130, 1200, 1208, 1258, 5354, 13488, 17584
1130	37, 500, 819, 850, 1129, 1200, 1208, 1252, 1258, 4946, 5354, 8229, 13488, 17584, 28709
1131	37, 500, 878, 915, 1025, 1200, 1208, 1251, 1283, 5347, 8229, 13488, 17584, 28709
1132	37, 500, 819, 850, 1133, 1200, 1208, 1252, 4946, 8229, 13488, 17584, 28709
1133	500, 1132, 1200, 1208, 13488, 17584
1137	37, 500, 819, 1200, 1208, 8229, 13488, 17584, 28709
1139	290, 1027, 4386, 5123, 8482
1140	37, 273, 277-278, 280, 284-285, 297, 437, 500, 808, 819, 850, 858, 860, 863, 871-872, 901-902, 923-924, 1013, 1047, 1051, 1141-1149, 1153-1157, 1160-1162, 1164, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
1141	37, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 863, 871-872, 923-924, 1047, 1051, 1140, 1142-1149, 1153-1157, 1160-1162, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709

Tabla 710. IBM MQ for z/OS Compatibilidad con la conversión CCSID (continuación)	
CCSID	Convierte a y desde CCSIDS
1142	37, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 863, 865, 871-872, 923-924, 1047, 1051, 1140-1141, 1143-1149, 1153-1157, 1160-1162, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
1143	37, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 863, 865, 871-872, 923-924, 1047, 1051, 1140-1142, 1144-1149, 1153-1157, 1160-1162, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
1144	37, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 863, 871-872, 923-924, 1047, 1051, 1140-1143, 1145-1149, 1153-1157, 1160-1162, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
1145	37, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 860, 863, 871-872, 923-924, 1047, 1051, 1140-1144, 1146-1149, 1153-1157, 1160-1162, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
1146	37, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 860, 863, 871-872, 923-924, 1047, 1051, 1140-1145, 1147-1149, 1153-1157, 1160-1162, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
1147	37, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 863, 871-872, 923-924, 1047, 1051, 1140-1146, 1148-1149, 1153-1157, 1160-1162, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
1148	37, 273, 277-278, 280, 284-285, 297, 437, 500, 808, 819, 848-850, 858, 860-861, 863, 865, 871-872, 901-902, 923-924, 1047, 1051, 1123, 1140-1147, 1149, 1153-1164, 1200, 1208, 1252, 1275, 4899, 4946, 5348, 5349, 8229, 12712, 13488, 17584, 28709
1149	37, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 861, 863, 871-872, 923-924, 1047, 1051, 1140-1148, 1153-1157, 1160-1162, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
1153	808, 858-859, 867, 872, 923-924, 1140-1149, 1154-1157, 1160-1162, 1200, 1208, 5348, 9044, 13488, 17584
1154	808, 849, 858-859, 867, 872, 923-924, 1140-1149, 1153, 1155-1157, 1160-1162, 1200, 1208, 5347, 5348, 13488, 17584
1155	858-859, 867, 872, 923-924, 1140-1149, 1153-1154, 1156-1157, 1160-1162, 1200, 1208, 5348, 5350, 9049, 13488, 17584
1156	858-859, 901-902, 923-924, 1140-1149, 1153-1155, 1157, 1160, 1200, 1208, 5348, 5353, 12712, 13488, 17584
1157	858-859, 901-902, 923-924, 1140-1149, 1153-1156, 1160, 1200, 1208, 5348, 5353, 12712, 13488, 17584
1158	848, 923, 1148, 1200, 1208, 5347, 5348, 13488, 17584
1159	1148, 1200, 1208, 13488, 17584
1160	858-859, 867, 923-924, 1140-1149, 1153-1157, 1161-1162, 1200, 1208, 5348, 13488, 17584

Tabla 710. IBM MQ for z/OS Compatibilidad con la conversión CCSID (continuación)	
CCSID	Convierte a y desde CCSIDS
1161	259, 858-859, 923-924, 1140-1149, 1153-1155, 1160, 5348, 17584
1162	259, 858-859, 923-924, 1140-1149, 1153-1155, 1160, 5348, 17584
1163	924, 1148, 1164, 5354, 17584
1164	858-859, 923-924, 1140, 1148, 1163, 1200, 1208, 5348, 5354, 13488, 17584
1166	1200, 1208, 13488, 17584
1200	37, 256, 259, 273, 275, 277-278, 280, 282, 284-285, 290, 293, 297, 300-301, 367, 420, 423-424, 437, 500, 720, 737, 775, 803, 806, 808, 813, 819, 833-838, 848-852, 855-872, 874-875, 878, 880, 891, 895-897, 901-905, 912, 914-916, 918, 920-924, 927-928, 930, 932-933, 935, 937, 939, 941-944, 946-951, 1004, 1006, 1008-1019, 1025-1027, 1040-1043, 1046-1047, 1051, 1088-1089, 1097-1098, 1112, 1114-1115, 1122-1126, 1129-1133, 1137, 1140-1149, 1153-1160, 1164, 1166, 1208, 1250-1258, 1275-1277, 1280-1285, 1351, 1362-1364, 1370-1371, 1374-1379, 1380-1381, 1385-1386, 1388, 1390, 1399, 4899, 4909, 4930, 4933, 4948, 4951-4952, 4960, 4971, 5012, 5039, 5104, 5123, 5142, 5210, 5346-5354, 8482, 8612, 9027, 9030, 9044, 9048-9049, 9056, 9061, 9066, 9238, 12712, 13121, 13218, 13488, 16684, 16804, 17248, 17584, 21427, 28709
1208	37, 256, 259, 273, 275, 277-278, 280, 282, 284-285, 290, 293, 297, 300-301, 367, 420, 423-424, 437, 500, 720, 737, 775, 803, 806, 808, 813, 819, 833-838, 848-852, 855-872, 874-875, 878, 880, 891, 895-897, 901-905, 912, 914-916, 918, 920-924, 927-928, 930, 932-933, 935, 937, 939, 941-944, 946-951, 1004, 1006, 1008-1019, 1025-1027, 1040-1043, 1046-1047, 1051, 1088-1089, 1097-1098, 1112, 1114-1115, 1122-1126, 1129-1133, 1137, 1140-1149, 1153-1160, 1164, 1166, 1200, 1250-1258, 1275-1277, 1280-1285, 1351, 1362-1364, 1370-1371, 1374-1379, 1380-1381, 1385-1386, 1388, 1390, 1399, 4899, 4909, 4930, 4933, 4948, 4951-4952, 4960, 4971, 5012, 5026, 5035, 5039, 5104, 5123, 5142, 5210, 5346-5354, 8482, 8612, 9027, 9030, 9044, 9048-9049, 9056, 9061, 9066, 9238, 12712, 13121, 13218, 13488, 16684, 16804, 17248, 17584, 21427, 28709
1250	37, 259, 273, 500, 819, 850, 852, 855, 870, 912, 1200, 1208, 1252, 1282, 4946, 4948, 4951, 5346, 8229, 9044, 13488, 17584, 28709
1251	37, 256, 259, 500, 819, 850, 855, 866, 878, 880, 915, 1025, 1123-1125, 1131, 1200, 1208, 1252, 1283, 4946, 4951, 5347, 8229, 13488, 17584, 28709
1252	37, 256, 259, 273, 275, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 737, 775, 803, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-866, 869-871, 874-875, 880, 897, 903, 905, 912, 914-916, 920-924, 1025-1027, 1041, 1047, 1051, 1097-1098, 1112, 1122-1123, 1126, 1130, 1132, 1140-1149, 1200, 1208, 1250-1251, 1254-1255, 1257, 1275, 1280-1281, 1283, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5346, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 16804, 17248, 17584, 25473, 25479, 25617, 28709
1253	37, 259, 423, 500, 737, 813, 819, 850, 869, 875, 1200, 1208, 1280, 4909, 4946, 4971, 5349, 8229, 9061, 13 488, 17 584, 28 709

Tabla 710. IBM MQ for z/OS Compatibilidad con la conversión CCSID (continuación)	
CCSID	Convierte a y desde CCSIDS
1254	37, 259, 500, 819, 850, 857, 869, 905, 920, 1026, 1047, 1200, 1208, 1252, 1281, 4946, 4953, 5350, 8229, 9049, 9061, 13488, 17584, 28709
1255	37, 259, 424, 500, 803, 819, 850, 856, 862, 916, 1200, 1208, 1252, 1281, 4946, 4952, 5012, 5351, 8229, 13 488, 17 584, 28 709
1256	259, 420, 500, 720, 850, 864, 1046, 1089, 1127, 1200, 1208, 4946, 4960, 5142, 5352, 8612, 9056, 9238, 13 488, 16 804, 17 248, 17 584
1257	37, 259, 437, 500, 775, 819, 850, 914, 921-922, 1112, 1122, 1200, 1208, 1252, 4946, 5353, 8229, 13488, 17584, 28709
1258	37, 259, 500, 819, 1129-1130, 1200, 1208, 5354, 8229, 13488, 17584, 28709
1275	37, 256, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 863, 871, 923-924, 1051, 1140-1149, 1200, 1208, 1252, 4946, 5348, 8229, 13488, 17584, 28709
1276	1200, 1208, 13488, 17584
1277	1200, 1208, 13488, 17584
1280	37, 423, 437, 500, 737, 813, 819, 850, 869, 875, 1200, 1208, 1252-1253, 4909, 4946, 4971, 5349, 8229, 9061, 13488, 17584, 28709
1281	37, 437, 500, 819, 850, 857, 905, 920, 1026, 1200, 1208, 1252, 1254-1255, 4946, 4953, 5350, 8229, 9049, 13488, 17584, 28709
1282	500, 852, 870, 912, 1200, 1208, 1250, 4948, 5346, 9044, 13488, 17584
1283	37, 437, 500, 819, 850, 855, 866, 878, 880, 915, 1025, 1123-1125, 1131, 1200, 1208, 1251-1252, 4946, 4951, 5347, 8229, 13488, 17584, 28709
1284	1200, 1208, 13488, 17584
1285	1200, 1208, 13488, 17584
1351	300-301, 941, 1200, 1208, 4396, 8492, 13488, 16684, 17584
1362	834, 951, 1200, 1208, 4930, 9026, 13488, 17584
1363	933, 949, 1200, 1208, 1364, 5029, 5045, 5460, 9125, 9555, 13221, 13488, 13651, 17317, 17584, 25525, 29621, 33717, 37813
1364	933, 949, 1200, 1208, 1363, 5029, 5045, 5460, 9125, 9555, 13221, 13488, 13651, 17317, 17584, 25525, 29621, 33717, 37813
1370	937-938, 948, 950, 1200, 1208, 1371, 5033, 5046, 9142, 13488, 17584, 25514, 25524, 29620
1371	1200, 1208, 1370, 13488, 17584
1374	1200, 1208
1375	1200, 1208
1376	1200, 1208
1377	1200, 1208
1378	1200, 1208
1379	1200, 1208

<i>Tabla 710. IBM MQ for z/OS Compatibilidad con la conversión CCSID (continuación)</i>	
CCSID	Convierte a y desde CCSIDS
1380	837, 928, 1200, 1208, 1385, 4933, 13488, 17584
1381	935-936, 1200, 1208, 1386, 1388, 5031, 5477, 5482, 5484, 9127, 13223, 13488, 17584, 25512
1385	837, 1200, 1208, 1380, 4933, 13488, 17584
1386	935, 1200, 1208, 1381, 1388, 5031, 5477, 5482, 5484, 9127, 13223, 13488, 17584
1388	935, 1200, 1208, 1381, 1386, 5031, 5477, 5482, 5484, 5488, 9127, 13223, 13488, 17584
1390	930-932, 939, 942-943, 1200, 1208, 1399, 5026, 5028, 5035, 5038-5039, 5055, 9122, 9124, 9131, 9135, 13218-13219, 13231, 13488, 17314, 17584, 25508, 25518, 29614, 33698-33700, 37796
1399	930-932, 939, 942-943, 1200, 1208, 1390, 5026, 5028, 5035, 5038-5039, 5050, 9122, 9124, 9131, 9135, 13218-13219, 13231, 13488, 17314, 17584, 25508, 25518, 29614, 33698-33700, 37796
4386	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 437, 500, 737, 775, 819, 833, 836, 850, 852, 855, 857, 860-865, 870-871, 895-897, 1009, 1025-1027, 1040-1043, 1088, 1112, 1122, 1139, 1252, 4929, 4932, 4946, 4948, 4951, 4953, 4960, 4992, 5123, 8229, 8482, 9025, 9044, 9049, 9056, 13121, 17248, 25473, 25617, 25619, 25664, 28709
4396	300-301, 941, 1351, 8492, 16684
4899	867, 1148, 1200, 1208, 5351, 9048, 12712, 13488, 17584
4909	37, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 737, 813, 819, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1200, 1208, 1252-1253, 1280, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 5349, 8229, 9030, 9044, 9049, 9061, 9066, 13488, 17584, 25473, 25479, 25617, 25619, 28709
4929	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 437, 500, 737, 775, 819, 833, 836, 850, 852, 855, 857, 860-865, 870-871, 891, 1009, 1025-1027, 1040-1043, 1088, 1112, 1122, 1126, 1252, 4386, 4932, 4946, 4948, 4951, 4953, 4960, 5123, 8229, 8482, 9025, 9044, 9049, 9056, 13121, 17248, 25617, 25619, 25664, 28709
4930	834, 951, 1200, 1208, 1362, 9026, 13488, 17584
4931	835, 927, 947, 9027, 21427
4932	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 424, 437, 500, 737, 775, 819, 833, 836, 850, 852, 855, 857, 870-871, 875, 903, 1009, 1025-1027, 1040-1043, 1088, 1112, 1114-1115, 1122, 1252, 4386, 4929, 4946, 4948, 4951, 4953, 4971, 5123, 5210-5211, 8229, 8482, 9025, 9044, 9049, 13121, 25479, 25617, 25619, 25664, 28709
4933	837, 1200, 1208, 1380, 1385, 13488, 17584
4934	37, 256, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 737, 775, 813, 819, 838, 850, 852, 857, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1112, 1122, 1252, 4909, 4946, 4948, 4953, 4960, 4970-4971, 5012, 5123, 8229, 9030, 9044, 9049, 9056, 9061, 9066, 17248, 25473, 25479, 25617, 25619, 28709

Tabla 710. IBM MQ for z/OS Compatibilidad con la conversión CCSID (continuación)

CCSID	Convierte a y desde CCSIDS
4946	37, 256, 259, 273, 275, 277-278, 280, 284-285, 290, 297, 367, 420, 423-424, 437, 500, 737, 775, 803, 813, 819, 833, 836, 838, 850, 852, 855-858, 860-866, 869-871, 874-875, 880, 897, 903, 905, 912, 914-916, 920-924, 1004, 1025-1027, 1040-1043, 1047, 1051, 1088-1089, 1097-1098, 1100, 1112, 1114, 1122, 1126, 1130, 1132, 1140-1149, 1250-1257, 1275, 1280-1281, 1283, 4386, 4909, 4929, 4932, 4934, 4948, 4951-4953, 4960, 4970-4971, 5012, 5123, 5210, 5346, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 16804, 17248, 25473, 25479, 25617, 25619, 25664, 28709
4948	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 1025-1027, 1040-1043, 1047, 1088, 1097, 1200, 1208, 1250, 1252, 1282, 4386, 4909, 4929, 4932, 4934, 4946, 4951, 4953, 4970-4971, 5012, 5123, 5346, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 13488, 16804, 17584, 25473, 25479, 25617, 25619, 25664, 28709
4951	37, 259, 273, 277-278, 280, 284-285, 290, 297, 437, 500, 819, 833, 836, 850, 852, 855, 857, 866, 870-871, 878, 880, 912, 915, 1025-1027, 1040-1043, 1088, 1200, 1208, 1250-1252, 1283, 4386, 4929, 4932, 4946, 4948, 4953, 5123, 5346, 5347, 8229, 8482, 9025, 9044, 9049, 13121, 13488, 17584, 25617, 25619, 25664, 28709
4952	259, 273, 424, 500, 803, 850, 856, 862, 916, 1200, 1208, 1255, 4946, 5012, 5351, 13488, 17584
4953	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 1025-1027, 1040-1043, 1088, 1097, 1252, 1254, 1281, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4970-4971, 5012, 5123, 5350, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 16804, 25473, 25479, 25617, 25619, 25664, 28709
4960	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 500, 720, 819, 833, 838, 850, 864, 870-871, 875, 880, 905, 918, 1008, 1025-1027, 1046, 1089, 1097, 1127, 1200, 1208, 1252, 1256, 4386, 4929, 4934, 4946, 4971, 5104, 5123, 5142, 5352, 8229, 8482, 8612, 9025, 9030, 9056, 9238, 13121, 13488, 16804, 17248, 17584, 28709
4970	37, 259, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 813, 819, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1252, 4909, 4934, 4946, 4948, 4953, 4971, 5012, 5123, 8229, 9030, 9044, 9049, 9061, 9066, 25473, 25479, 25617, 25619, 28709
4971	37, 256, 273, 277-278, 280, 284-285, 297, 367, 423, 437, 500, 737, 775, 813, 819, 836, 838, 850-852, 857, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1009, 1025-1027, 1041-1043, 1047, 1088, 1112, 1122, 1200, 1208, 1252-1253, 1280, 4909, 4932, 4934, 4946, 4948, 4953, 4960, 4970, 5012, 5123, 5349, 8229, 9030, 9044, 9049, 9056, 9061, 9066, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
4992	290, 896, 1027, 1041, 4386, 5123, 8482, 25617

<i>Tabla 710. IBM MQ for z/OS Compatibilidad con la conversión CCSID (continuación)</i>	
CCSID	Convierte a y desde CCSIDS
5012	37, 273, 277-278, 280, 284-285, 297, 423-424, 437, 500, 803, 813, 819, 838, 850, 852, 856-857, 860-863, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1200, 1208, 1252, 1255, 4909, 4934, 4946, 4948, 4952-4953, 4970-4971, 5123, 5351, 8229, 9030, 9044, 9049, 9061, 9066, 13488, 17584, 25473, 25479, 25617, 25619, 28709
5026	930-932, 939, 942-943, 1390, 1399, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 1208, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
5028	930-932, 939, 942-943, 1390, 1399, 5026, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
5029	933-934, 944, 949, 1363-1364, 5045, 5460, 9125, 9555, 13221, 13651, 17317, 25510, 25520, 25525, 29616, 29621, 33717, 37813
5031	935-936, 946, 1381, 1386, 1388, 5477, 5482, 5484, 9127, 13223, 25512
5033	937-938, 948, 950, 1370, 5046, 9142, 25514, 25524, 29620
5035	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5038-5039, 9122, 9124, 9131, 9135, 1208, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
5038	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
5039	930-932, 939, 942-943, 1200, 1208, 1390, 1399, 5026, 5028, 5035, 5038, 9122, 9124, 9131, 9135, 13218-13219, 13231, 13488, 17314, 17584, 25508, 25518, 29614, 33698-33700, 37796
5045	933-934, 944, 949, 1363-1364, 5029, 5460, 9125, 9555, 13221, 13651, 17317, 25510, 25520, 25525, 29616, 29621, 33717, 37813
5046	937-938, 948, 950, 1370, 5033, 9142, 25514, 25524, 29620
5104	420, 864, 1008, 1200, 1208, 4960, 8612, 9056, 13488, 16804, 17248, 17584
5123	290, 367, 423, 437, 819, 1027, 1041, 1047, 1140-1149, 1156, 1157, 1160, 1200, 1208, 1252, 4948, 5348, 8482, 13488
5142	420, 500, 864, 1046, 1089, 1127, 1200, 1208, 1256, 4960, 5352, 8612, 9056, 9238, 13 488, 16 804, 17 248, 17 584
5210	37, 437, 500, 819, 836, 850, 904, 1043, 1114-1115, 1200, 1208, 4932, 4946, 5211, 8229, 13488, 17584, 25480, 25619, 28709
5211	37, 367, 437, 500, 836, 903, 1114-1115, 4932, 5210, 8229, 25479, 28709
5346	37, 259, 273, 500, 819, 850, 852, 855, 870, 912, 1200, 1208, 1250, 1252, 1282, 4946, 4948, 4951, 8229, 9044, 13488, 17584, 28709
5347	808, 848-849, 855, 866, 872, 878, 880, 915, 1025, 1123-1125, 1131, 1154, 1158, 1200, 1208, 1251, 1283, 4951, 13488, 17584

Tabla 710. IBM MQ for z/OS Compatibilidad con la conversión CCSID (continuación)	
CCSID	Convierte a y desde CCSIDS
5348	37, 259, 273, 275, 277-278, 280, 284-285, 297, 437, 500, 808, 819, 850, 858, 860-861, 863, 865, 871-872, 901-902, 923-924, 1051, 1140-1149, 1153-1158, 1160-1162, 1164, 1200, 1208, 1252, 1275, 4946, 8229, 13488, 17584, 28709
5349	813, 869, 875, 1148, 1200, 1208, 1253, 1280, 4909, 4971, 9061, 13488, 17584
5350	857, 920, 1026, 1155, 1200, 1208, 1254, 1281, 4953, 9049, 13488, 17584
5351	424, 856, 862, 867, 916, 1200, 1208, 1255, 4899, 4952, 5012, 9048, 12712, 13488, 17584
5352	420, 864, 1046, 1089, 1200, 1208, 1256, 4960, 5142, 8612, 9056, 9238, 13488, 16804, 17248, 17584
5353	901-902, 921-922, 1112, 1122, 1156-1157, 1200, 1208, 1257, 13488, 17584
5354	1129-1130, 1163, 1164, 1200, 1208, 1258, 13488, 17584
5460	933-934, 944, 949, 1363-1364, 5029, 5045, 9125, 9555, 13221, 13651, 17317, 25510, 25520, 25525, 29616, 29621, 33717, 37813
5477	935-936, 1381, 1386, 1388, 5031, 5482, 5484, 9127, 13223, 25512
5482	935, 1381, 1386, 1388, 5031, 5477, 5484, 9127, 13223
5484	935-936, 946, 1381, 1386, 1388, 5031, 5477, 5482, 9127, 13223, 25512
5488	1388
8229	37, 256, 273, 275, 277-278, 280, 284-285, 290, 297, 367, 420, 423-424, 437, 500, 720, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-866, 869-871, 874-875, 880, 897, 903-905, 912, 914-916, 920-924, 1009, 1025-1027, 1040-1043, 1047, 1051, 1088, 1097, 1100, 1112, 1114-1115, 1122, 1124, 1126, 1130-1132, 1137, 1140-1149, 1250-1255, 1257-1258, 1275, 1280-1281, 1283, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5210-5211, 5346, 5348, 8482, 8612, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 16804, 17248, 25473, 25479, 25480, 25617, 25619, 25664, 28709
8482	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 437, 500, 737, 775, 819, 833, 836, 850, 852, 855, 857, 860-865, 870-871, 895-897, 1009, 1025-1027, 1040-1043, 1047, 1088, 1112, 1122, 1139, 1200, 1208, 1252, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 4960, 4992, 5123, 8229, 9025, 9044, 9049, 9056, 13121, 13488, 17248, 17584, 25473, 25617, 25619, 25664, 28709
8492	300-301, 941, 1351, 4396, 16684
8612	37, 256, 420, 424, 437, 500, 720, 737, 775, 819, 850, 852, 857, 860-865, 1008, 1046, 1089, 1098, 1112, 1122, 1127, 1200, 1208, 1252, 1256, 4946, 4948, 4953, 4960, 5104, 5142, 5352, 8229, 9044, 9049, 9056, 9238, 13488, 16804, 17248, 17584, 28709

<i>Tabla 710. IBM MQ for z/OS Compatibilidad con la conversión CCSID (continuación)</i>	
CCSID	Convierte a y desde CCSIDS
9025	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 437, 500, 737, 775, 819, 833, 836, 850, 852, 855, 857, 860-865, 870-871, 891, 1009, 1025-1027, 1040-1043, 1088, 1112, 1122, 1126, 1252, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 4960, 5123, 8229, 8482, 9044, 9049, 9056, 13121, 17248, 25617, 25619, 25664, 28709
9026	834, 926, 951, 1362, 4930
9027	835, 927, 947, 1200, 1208, 4931, 13488, 17584, 21427
9030	37, 256, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 737, 775, 813, 819, 838, 850, 852, 857, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1112, 1122, 1200, 1208, 1252, 4909, 4934, 4946, 4948, 4953, 4960, 4970-4971, 5012, 5123, 8229, 9044, 9049, 9056, 9061, 9066, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 28709
9044	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 1025-1027, 1040-1043, 1047, 1088, 1097, 1153, 1200, 1208, 1250, 1252, 1282, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4970-4971, 5012, 5123, 5346, 8229, 8482, 8612, 9025, 9030, 9049, 9061, 9066, 13121, 13488, 16804, 17584, 25473, 25479, 25617, 25619, 25664, 28709
9048	867, 1200, 1208, 4899, 5351, 12712, 13488, 17584
9049	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 1025-1027, 1040-1043, 1088, 1097, 1155, 1200, 1208, 1252, 1254, 1281, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4970-4971, 5012, 5123, 5350, 8229, 8482, 8612, 9025, 9030, 9044, 9061, 9066, 13121, 13488, 16804, 17584, 25473, 25479, 25617, 25619, 25664, 28709
9056	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 500, 720, 819, 833, 838, 850, 864, 870-871, 875, 880, 905, 918, 1008, 1025-1027, 1046, 1089, 1097, 1127, 1200, 1208, 1252, 1256, 4386, 4929, 4934, 4946, 4960, 4971, 5104, 5123, 5142, 5352, 8229, 8482, 8612, 9025, 9030, 9238, 13121, 13488, 16804, 17248, 17584, 28709
9061	37, 256, 259, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 737, 813, 819, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1200, 1208, 1252-1254, 1280, 4909, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 5349, 8229, 9030, 9044, 9049, 9066, 13488, 17584, 25473, 25479, 25617, 25619, 28709
9066	37, 259, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 813, 819, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1200, 1208, 1252, 4909, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 8229, 9030, 9044, 9049, 9061, 13488, 17584, 25473, 25479, 25617, 25619, 28709
9122	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796

<i>Tabla 710. IBM MQ for z/OS Compatibilidad con la conversión CCSID (continuación)</i>	
CCSID	Convierte a y desde CCSIDS
9124	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
9125	933-934, 944, 949, 1363-1364, 5029, 5045, 5460, 9555, 13221, 13651, 17317, 25510, 25520, 25525, 29616, 29621, 33717, 37813
9127	935-936, 946, 1381, 1386, 1388, 5031, 5477, 5482, 5484, 13223, 25512
9131	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
9135	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
9142	937-938, 948, 950, 1370, 5033, 5046, 25514, 25524, 29620
9238	420, 500, 864, 1046, 1089, 1127, 1200, 1208, 1256, 4960, 5142, 5352, 8612, 9056, 13 488, 16 804, 17 248, 17 584
9555	933, 949, 1363-1364, 5029, 5045, 5460, 9125, 13221, 13651, 17317, 25525, 29621, 33717, 37813
12712	862, 867, 1148, 1156-1157, 1200, 1208, 4899, 5351, 9048, 13488, 17584
13121	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 437, 500, 737, 775, 819, 833, 836, 850, 852, 855, 857, 860-865, 870-871, 891, 1009, 1025-1027, 1040-1043, 1088, 1112, 1122, 1126, 1200, 1208, 1252, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 4960, 5123, 8229, 8482, 9025, 9044, 9049, 9056, 13488, 17248, 17584, 25617, 25619, 25664, 28709
13218	930-932, 939, 942-943, 1200, 1208, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13219, 13231, 13488, 17314, 17584, 25508, 25518, 29614, 33698-33700, 37796
13219	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
13221	933-934, 944, 949, 1363-1364, 5029, 5045, 5460, 9125, 9555, 13651, 17317, 25510, 25520, 25525, 29616, 29621, 33717, 37813
13223	935-936, 946, 1381, 1386, 1388, 5031, 5477, 5482, 5484, 9127, 25512
13231	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 17314, 25508, 25518, 29614, 33698-33700, 37796

Tabla 710. IBM MQ for z/OS Compatibilidad con la conversión CCSID (continuación)	
CCSID	Convierte a y desde CCSIDS
13488	37, 256, 259, 273, 275, 277-278, 280, 282, 284-285, 290, 293, 297, 300-301, 367, 420, 423-424, 437, 500, 720, 737, 775, 803, 806, 808, 813, 819, 833-838, 848-852, 855-872, 874-875, 878, 880, 891, 895-897, 901-905, 912, 914-916, 918, 920-924, 927-928, 930, 932-933, 935, 937, 939, 941-944, 946-951, 1004, 1006, 1008-1019, 1025-1027, 1040-1043, 1046-1047, 1051, 1088-1089, 1097-1098, 1112, 1114-1115, 1122-1126, 1129-1133, 1137, 1140-1149, 1153-1160, 1164, 1166, 1200, 1208, 1250-1258, 1275-1277, 1280-1285, 1351, 1362-1364, 1370-1371, 1380-1381, 1385-1386, 1388, 1390, 1399, 4899, 4909, 4930, 4933, 4948, 4951-4952, 4960, 4971, 5012, 5039, 5104, 5123, 5142, 5210, 5346-5354, 8482, 8612, 9027, 9030, 9044, 9048-9049, 9056, 9061, 9066, 9238, 12712, 13121, 13218, 16684, 16804, 17248, 17584, 21427, 28709
13651	933, 949, 1363-1364, 5029, 5045, 5460, 9125, 9555, 13221, 17317, 25525, 29621, 33717, 37813
16684	300-301, 941, 1200, 1208, 1351, 4396, 8492, 13488, 17584
16804	37, 256, 420, 424, 437, 500, 720, 737, 775, 819, 850, 852, 857, 860-865, 1008, 1046, 1089, 1098, 1112, 1122, 1127, 1200, 1208, 1252, 1256, 4946, 4948, 4953, 4960, 5104, 5142, 5352, 8229, 8612, 9044, 9049, 9056, 9238, 13488, 17248, 17584, 28709
17248	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 500, 720, 819, 833, 838, 850, 864, 870-871, 875, 880, 905, 918, 1008, 1025-1027, 1046, 1089, 1097, 1127, 1200, 1208, 1252, 1256, 4386, 4929, 4934, 4946, 4960, 4971, 5104, 5123, 5142, 5352, 8229, 8482, 8612, 9025, 9030, 9056, 9238, 13121, 13488, 16804, 17584, 28709
17314	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 25508, 25518, 29614, 33698-33700, 37796
17317	933-934, 944, 949, 1363-1364, 5029, 5045, 5460, 9125, 9555, 13221, 13651, 25510, 25520, 25525, 29616, 29621, 33717, 37813
17584	37, 256, 259, 273, 275, 277-278, 280, 282, 284-285, 290, 293, 297, 300-301, 367, 420, 423-424, 437, 500, 720, 737, 775, 803, 806, 808, 813, 819, 833-838, 848-852, 855-872, 874-875, 878, 880, 891, 895-897, 901-905, 912, 914-916, 918, 920-924, 927-928, 930, 932-933, 935, 937, 939, 941-944, 946-951, 1004, 1006, 1008-1019, 1025-1027, 1040-1043, 1046-1047, 1051, 1088-1089, 1097-1098, 1112, 1114-1115, 1122-1126, 1129-1133, 1137, 1140-1149, 1153-1160, 1164, 1166, 1200, 1208, 1250-1258, 1275-1277, 1280-1285, 1351, 1362-1364, 1370-1371, 1380-1381, 1385-1386, 1388, 1390, 1399, 4899, 4909, 4930, 4933, 4948, 4951-4952, 4960, 4971, 5012, 5039, 5104, 5123, 5142, 5210, 5346-5354, 8482, 8612, 9027, 9030, 9044, 9048-9049, 9056, 9061, 9066, 9238, 12712, 13121, 13218, 13488, 16684, 16804, 17248, 21427, 28709
21427	835, 927, 947, 1200, 1208, 4931, 9027, 13488, 17584
25473	37, 273, 277-278, 280, 284-285, 290, 297, 423, 437, 500, 813, 819, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1252, 4386, 4909, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 8229, 8482, 9030, 9044, 9049, 9061, 9066, 25479, 25617, 25619, 28709

<i>Tabla 710. IBM MQ for z/OS Compatibilidad con la conversión CCSID (continuación)</i>	
CCSID	Convierte a y desde CCSIDS
25479	37, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 813, 819, 836, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1115, 1252, 4909, 4932, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 5211, 8229, 9030, 9044, 9049, 9061, 9066, 25473, 25617, 25619, 28709
25480	37, 500, 904, 1114, 5210, 8229, 28709
25508	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25518, 29614, 33698-33700, 37796
25510	933-934, 949, 5029, 5045, 5460, 9125, 13221, 17317, 25525, 29621, 33717, 37813
25512	935-936, 946, 1381, 5031, 5477, 5484, 9127, 13223
25514	937-938, 950, 1370, 5033, 5046, 9142
25518	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 29614, 33698-33700, 37796
25520	933, 944, 949, 5029, 5045, 5460, 9125, 13221, 17317, 25525, 29616, 29621, 33717, 37813
25524	937, 948, 950, 1370, 5033, 5046, 9142, 29620
25525	933-934, 944, 949, 1363-1364, 5029, 5045, 5460, 9125, 9555, 13221, 13651, 17317, 25510, 25520, 29616, 29621, 33717, 37813
25617	37, 273, 277-278, 280, 284-285, 290, 297, 367, 423, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 895-897, 903, 912, 916, 1025-1027, 1040-1043, 1088, 1252, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4970-4971, 4992, 5012, 5123, 8229, 8482, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 25473, 25479, 25619, 25664, 28709
25619	37, 273, 277-278, 280, 284-285, 290, 297, 423, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 1025-1027, 1040-1043, 1088, 1114, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4970-4971, 5012, 5123, 5210, 8229, 8482, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 25473, 25479, 25617, 25664, 28709
25664	37, 273, 277-278, 280, 284-285, 290, 297, 367, 500, 819, 833, 836, 850, 852, 855, 857, 870-871, 875, 891, 1025-1027, 1040-1043, 1088, 1126, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 4971, 5123, 8229, 8482, 9025, 9044, 9049, 13121, 25617, 25619, 28709

Tabla 710. IBM MQ for z/OS Compatibilidad con la conversión CCSID (continuación)	
CCSID	Convierte a y desde CCSIDS
28709	37, 256, 273, 275, 277-278, 280, 284-285, 290, 297, 367, 420, 423-424, 437, 500, 720, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-866, 869-871, 874-875, 880, 897, 903-905, 912, 914-916, 920-924, 1009, 1025-1027, 1040-1043, 1047, 1051, 1088, 1097, 1100, 1112, 1114-1115, 1122, 1124, 1126, 1130-1132, 1137, 1140-1149, 1200, 1208, 1250-1255, 1257-1258, 1275, 1280-1281, 1283, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5210-5211, 5346, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 16804, 17248, 17584, 25473, 25479, 25480, 25617, 25619, 25664
29614	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 33698-33700, 37796
29616	933, 944, 949, 5029, 5045, 5460, 9125, 13221, 17317, 25520, 25525, 29621, 33717, 37813
29620	937, 948, 950, 1370, 5033, 5046, 9142, 25524
29621	933-934, 944, 949, 1363-1364, 5029, 5045, 5460, 9125, 9555, 13221, 13651, 17317, 25510, 25520, 25525, 29616, 33717, 37813
33698	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33699-33700, 37796
33699	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698, 33700, 37796
33700	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33699, 37796
33717	933-934, 944, 949, 1363-1364, 5029, 5045, 5460, 9125, 9555, 13221, 13651, 17317, 25510, 25520, 25525, 29616, 29621, 37813
37796	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700
37813	933-934, 944, 949, 1363-1364, 5029, 5045, 5460, 9125, 9555, 13221, 13651, 17317, 25510, 25520, 25525, 29616, 29621, 33717

Diferencias entre las páginas de códigos EBCDIC y Latin 1

Muchos países tienen su propia página de códigos EBCDIC, por razones históricas. Cuando se definieron las páginas de códigos EBCDIC, se incluyeron los 192 caracteres de todas las páginas de códigos relacionadas con Latin 1, pero los puntos de código son diferentes. Por consiguiente, si utiliza una página de códigos EBCDIC para introducir y mostrar datos, pero una página de códigos diferente (a menudo la página de códigos 500, ya que es la z/OS predeterminada) para realizar la conversión de datos, es posible que algunos caracteres se muestren incorrectamente.

La siguiente tabla muestra las diferencias en los valores hexadecimales de algunas páginas de códigos comunes europeas y americanas. La página de códigos 1047 es la página de códigos predeterminada para el z/OS compilador. Los valores hexadecimales que difieren del valor más común están subrayados.

Tabla 711. Caracteres y sus valores hexadecimales en varias páginas de códigos EBCDIC

Perso naje (nomb re)	37 (EE. UU., Canad á)	273 (Alem ania, Austri a)	277 (Dina marca , Norue ga)	278 (Finla ndia, Suecia)	280 (Italia)	284 (Españ a)	285 (Reino Unido)	297 (Franc ia)	500 (intern aciona l)	871 (Islan dia)	1047 (Latín 1, Sist emas abierto s)
[(corc hete de apertu ra)	<u>BA</u>	<u>63</u>	<u>9E</u>	<u>B5</u>	<u>90</u>	4A	<u>B1</u>	<u>90</u>	4A	<u>AE</u>	<u>DC</u>
] (corch ete de cierre)	<u>BB</u>	<u>Canal de fibra</u>	<u>9F</u>	<u>9F</u>	<u>51</u>	5A	<u>BB</u>	<u>B5</u>	5A	<u>9E</u>	<u>BD</u>
{ (llave de apertu ra)	C0	<u>44</u>	<u>9C</u>	<u>44</u>	<u>44</u>	C0	C0	<u>51</u>	C0	<u>8E</u>	C0
} (llave de cierre)	D0	<u>DC</u>	<u>47</u>	<u>47</u>	<u>54</u>	D0	D0	<u>54</u>	D0	<u>9C</u>	D0
~ (virguli lla)	A1	<u>59</u>	<u>DC</u>	<u>DC</u>	<u>58</u>	<u>BD</u>	<u>AC</u>	<u>BD</u>	A1	<u>Atenci ón al cliente</u>	A1
@ (arrob a)	7C	<u>B5</u>	<u>80</u>	<u>EC</u>	<u>B5</u>	7C	7C	<u>44</u>	7C	<u>CA</u>	7C
(barra vertica l dividid a)	6A	<u>Atenci ón al cliente</u>	<u>70</u>	<u>Atenci ón al cliente</u>	<u>CD</u>	<u>49</u>	6A	<u>DD</u>	6A	6A	6A
# (signo numér ico)	7B	7B	<u>4A</u>	<u>63</u>	<u>B1</u>	<u>69</u>	7B	<u>B1</u>	7B	7B	7B
` (acent o grave)	79	79	79	<u>51</u>	<u>DD</u>	79	79	<u>A0</u>	79	<u>8C</u>	79
^ (acent o circunf lejo)	<u>B0</u>	5F	5F	5F	5F	<u>BA</u>	<u>BA</u>	5F	5F	<u>EC</u>	5F

Tabla 711. Caracteres y sus valores hexadecimales en varias páginas de códigos EBCDIC (continuación)

Perso naje (nomb re)	37 (EE. UU., Canad á)	273 (Alem ania, Austri a)	277 (Dina marca , Norue ga)	278 (Finla ndia, Suecia)	280 (Italia)	284 (Españ a)	285 (Reino Unido)	297 (Franc ia)	500 (intern aciona l)	871 (Islan dia)	1047 (Latín 1, Sistem as abiert os)
§ (indica dor de secció n)	B5	<u>7C</u>	B5	<u>4A</u>	<u>7C</u>	B5	B5	<u>5A</u>	B5	B5	B5
! (signo de exclam ación)	<u>5A</u>	4F	4F	4F	4F	<u>BB</u>	<u>5A</u>	4F	4F	4F	<u>5A</u>
\$ (símb olo de dólar)	5B	5B	<u>67</u>	<u>67</u>	5B	5B	<u>4A</u>	5B	5B	5B	5B
¬ (negac ión lógica)	<u>5F</u>	BA	BA	BA	BA	<u>5F</u>	<u>5F</u>	BA	BA	BA	<u>B0</u>
¨ (diéres is, diéresi s)	BD	BD	BD	BD	BD	<u>A1</u>	BD	<u>A1</u>	BD	BD	<u>BB</u>
£ (signo de libra)	B1	B1	B1	B1	<u>7B</u>	B1	<u>5B</u>	<u>7B</u>	B1	B1	B1
¢ (signo de centav o)	<u>4A</u>	B0	B0	B0	B0	B0	B0	B0	B0	B0	<u>4A</u>
 (barra vertica l, o lógico)	<u>4F</u>	BB	BB	BB	BB	<u>4F</u>	<u>4F</u>	BB	BB	BB	<u>4F</u>

Tabla 711. Caracteres y sus valores hexadecimales en varias páginas de códigos EBCDIC (continuación)

Perso naje (nomb re)	37 (EE. UU., Canad á)	273 (Alem ania, Austri a)	277 (Dina marca , Norue ga)	278 (Finla ndia, Suecia)	280 (Italia)	284 (Españ a)	285 (Reino Unido)	297 (Franc ia)	500 (intern aciona l)	871 (Islan dia)	1047 (Latín 1, Sistem as abierto s)
₣ (mone da intern acional, sputni k)	9F	9F	<u>5A</u>	<u>5A</u>	9F	9F	9F	9F	9F	9F	9F
μ (mu)	A0	A0	A0	A0	A0	A0	A0	<u>79</u>	A0	A0	A0
ß (beta)	59	<u>A1</u>	59	59	59	59	59	59	59	59	59
\ (barra inverti da)	E0	<u>EC</u>	E0	<u>71</u>	<u>48</u>	E0	E0	<u>48</u>	E0	<u>BE</u>	E0
° (símb olo de grado)	90	90	90	90	<u>4A</u>	90	90	<u>4A</u>	90	90	90
ˉ (barra superi or)	AC	AC	AC	AC	AC	AC	<u>A1</u>	AC	AC	AC	AC
´ (acent o agudo)	BE	BE	BE	BE	BE	BE	BE	BE	BE	<u>E0</u>	BE
ä (a con diéresi s)	43	<u>C0</u>	43	<u>C0</u>	43	43	43	43	43	43	43
à (a con acento grave)	44	44	44	44	<u>C0</u>	44	44	<u>7C</u>	44	44	44
â (a con círculo superi or)	47	47	<u>D0</u>	<u>D0</u>	47	47	47	47	47	47	47

Tabla 711. Caracteres y sus valores hexadecimales en varias páginas de códigos EBCDIC (continuación)

Perso naje (nomb re)	37 (EE. UU., Canad á)	273 (Alem ania, Austri a)	277 (Dina marca , Norue ga)	278 (Finla ndia, Suecia)	280 (Italia)	284 (Españ a)	285 (Reino Unido)	297 (Franc ia)	500 (intern aciona l)	871 (Islan dia)	1047 (Latín 1, Sistem as abierto s)
Ä (A con diéresi s)	63	<u>4A</u>	63	<u>7B</u>	63	63	63	63	63	63	63
Å (A con círculo superi or)	67	<u>67</u>	5B	<u>5B</u>	67	67	67	67	67	67	67
æ (ligadu ra de ae)	9C	<u>9C</u>	C0	<u>9C</u>	9C	9C	9C	9C	9C	D0	9C
Æ (ligadu ra AE)	9E	<u>9E</u>	7B	<u>9E</u>	9E	9E	9E	9E	9E	5A	9E
ç (c con cedilla)	48	<u>48</u>	48	<u>48</u>	E0	48	48	E0	48	48	48
ð (eth minús cula)	8C	<u>8C</u>	8C	<u>8C</u>	8C	8C	8C	8C	8C	79	8C
Ð (eth mayús cula)	CA	<u>CA</u>	CA	<u>CA</u>	CA	CA	CA	CA	CA	7C	CA
é (e con acento agudo)	51	<u>51</u>	51	<u>79</u>	5A	51	51	C0	51	51	51
è (e con acento grave)	54	<u>54</u>	54	<u>54</u>	D0	54	54	D0	54	54	54
É (E con acento agudo)	71	<u>71</u>	71	<u>E0</u>	71	71	71	71	71	71	71
ì (i con acento grave)	58	<u>58</u>	58	<u>58</u>	A1	58	58	58	58	58	58

Tabla 711. Caracteres y sus valores hexadecimales en varias páginas de códigos EBCDIC (continuación)

Perso naje (nomb re)	37 (EE. UU., Canad á)	273 (Alem ania, Austri a)	277 (Dina marca , Norue ga)	278 (Finla ndia, Suecia)	280 (Italia)	284 (Españ a)	285 (Reino Unido)	297 (Franc ia)	500 (intern aciona l)	871 (Islan dia)	1047 (Latín 1, Sistem as abierto s)
ñ (n con tilde)	49	<u>49</u>	49	<u>49</u>	49	6A	49	49	49	49	49
Ñ (N con tilde)	69	<u>69</u>	69	<u>69</u>	69	7B	69	69	69	69	69
ø (o con barra)	70	<u>70</u>	6A	<u>70</u>	70	70	70	70	70	70	70
Ø (O con barra)	80	<u>80</u>	7C	<u>80</u>	80	80	80	80	80	80	80
ö (o con diéresi s)	Atenci ón al cliente	<u>6A</u>	Atenci ón al cliente	<u>6A</u>	Atenci ón al cliente	Atenci ón al cliente	Atenci ón al cliente	Atenci ón al cliente	Atenci ón al cliente	A1	Atenci ón al cliente
ò (o con acento grave)	CD	<u>CD</u>	CD	<u>CD</u>	6A	CD	CD	CD	CD	CD	CD
ü (u con diéresi s)	DC	<u>D0</u>	A1	<u>A1</u>	DC	DC	DC	DC	DC	DC	DC
ù (u con acento grave)	DD	<u>DD</u>	DD	<u>DD</u>	79	DD	DD	6A	DD	DD	DD
Ö (O con diéresi s)	EC	<u>E0</u>	EC	<u>7C</u>	EC	EC	EC	EC	EC	5F	EC
Ü (U con diéresi s)	Canal de fibra	<u>5A</u>	Canal de fibra	<u>Canal de fibra</u>	Canal de fibra	Canal de fibra	Canal de fibra	Canal de fibra	Canal de fibra	Canal de fibra	Canal de fibra
Ý (Y con acento agudo)	DC	<u>DC</u>	DC	<u>DC</u>	DC	DC	DC	DC	DC	DC	BA

Tabla 711. Caracteres y sus valores hexadecimales en varias páginas de códigos EBCDIC (continuación)

Personeja (nombre)	37 (EE. UU., Canadá)	273 (Alemania, Austria)	277 (Dinamarca, Noruega)	278 (Finlandia, Suecia)	280 (Italia)	284 (España)	285 (Reino Unido)	297 (Francia)	500 (internacional)	871 (Islandia)	1047 (Latín 1, Sistemas abiertos)
p (thorn minúscula)	8E	8E	8E	8E	8E	8E	8E	8E	8E	C0	8E
P (thorn mayúscula)	AE	AE	AE	AE	AE	AE	AE	AE	AE	4A	AE

Diferencias entre los caracteres de código de control EBCDIC y ASCII

La conversión de caracteres de control entre EBCDIC y ASCII puede causar problemas. Algunas plataformas intentan convertir estos caracteres, pero ASCII solo tiene 32 puntos de código para caracteres de control, mientras que EBCDIC asigna 64, por lo que 32 de los caracteres de control EBCDIC se convierten a valores de caracteres en ASCII. Las siguientes tablas enumeran los caracteres de control que generalmente se conservan durante la conversión de datos. La tabla no incluye el carácter NL (hexadecimal 15). En la mayoría de los CCSID ASCII no hay caracteres de nueva línea. Puede especificar cómo IBM MQ convierte este carácter en la estrofa `AllQueueManagers` del archivo `mqs.ini` o utilizando la variable de entorno `AMQ_CONVEBCDICNEWLINE`.

Nombre	Código hexadecimal EBCDIC	Código hexadecimal ASCII
NUL (nulo)	00	00
SOH (inicio de encabezado)	01	01
STX (inicio del texto)	02	02
ETX (fin de encabezado)	03	03
HT (tabulador horizontal)	05	09
VT (pestaña vertical)	0B	0B
FF (salto de página)	0C	0C
CR (retorno de carro)	0D	0D
SO (cambio de turno)	0E	0E
SI (cambio en)	0F	0F
DLE (escapado de enlace de datos)	10	10
DC1 (control del dispositivo 1)	11	11
DC2 (control del dispositivo 2)	12	12
DC3 (control del dispositivo 3)	13	13
BS (retroceso)	16	08
CAN (cancelar)	18	18

Nombre	Código hexadecimal EBCDIC	Código hexadecimal ASCII
EM (mensaje final)	19	19
FS (separador de archivos)	1C	1C
IGS (separador de grupos de intercambio)	1D	1D
IRS (separador de registros de intercambio)	1E	1E
IUS/ITB (separador de unidades de intercambio o bloque de transmisión intermedio)	1F	1F
LF (salto de línea).	25	0A
ETB (fin del bloque de transmisión)	26	17
ESC (Esc)	27	1B
ENQ (consulta)	2D	05
ACK (acuse de recibo)	2E	06
BEL (campana)	2F	07
SYN (sincronización inactiva)	32	16
EOT (fin del bloque de transmisión)	37	04
DC4 (control del dispositivo 4)	3C	14
NAK (confirmación negativa)	3D	15
SUB (sustituto)	3F	1A

Soporte de conversión de IBM i

Puede encontrar una lista completa de los CCSID y las conversiones soportadas por IBM i en la publicación correspondiente de IBM i.

Las páginas de códigos soportadas se listan en [Correlaciones de CCSID soportadas](#).

Compatibilidad con la conversión Unicode

Algunas plataformas admiten la conversión de datos de usuario a o desde la codificación Unicode. Las dos formas de codificación Unicode compatibles son « UTF-16 » (CCSID 1200, 13488 y 17584) y « UTF-8 » (CCSID 1208). Debe utilizar los CCSID 1200 o 1208, ya que representan la versión Unicode más reciente compatible.

IBM MQ Admite todos los caracteres Unicode definidos en el estándar Unicode 8.0 en la conversión de datos. Este soporte incluye soporte completo para UTF-16, incluyendo pares sustitutos UTF-16 (un par de caracteres UTF-16 de 2 bytes en el rango X' D800 ' hasta X'DFFF' que representan un punto de código Unicode por encima de U+FFFF). Si un CCSID de destino no contiene una asignación para un punto de código representado por un par sustituto e UTF-16, el par de caracteres se convierte en un único carácter de sustitución.

IBM MQ admite la combinación de secuencias de caracteres. Este soporte significa que, en algunos casos, un carácter precompuesto en el CCSID de origen se convierte en una secuencia de caracteres combinados en el CCSID de destino, o viceversa.

Nota: IBM MQ No admite los CCSID del gestor de colas UTF-16, por lo que los datos del encabezado del mensaje no se pueden codificar en UTF-16.

AIXIBM MQ Compatibilidad con Unicode



Se admite la IBM MQ for AIX conversión a y desde los CCSID Unicode compatibles (preferiblemente 1200 o 1208) para los CCSID no Unicode de la siguiente lista:

037
273, 278, 280, 284, 285, 297
423, 437
500
813, 819, 850, 852, 856, 857, 858, 860, 861, 865, 867, 869, 875, 878, 880
901, 902, 912, 915, 916, 920, 923, 924, 932, 933, 935, 937, 938, 939, 942, 943, 948, 949, 950, 954, 964, 970
1026, 1046, 1089
1129, 1130, 1131, 1132, 1133, 1140, 1141, 1142, 1143, 1144, 1145, 1146, 1147, 1148, 1149, 1153, 1156, 1157
1200, 1208, 1250, 1251, 1253, 1254, 1258, 1280, 1281, 1282, 1283, 1284, 1285
1363, 1364, 1381, 1383, 1386, 1388
4899
5026, 5035, 5050, 5346, 5347, 5348, 5349, 5350, 5351, 5352, 5353, 5354, 5488
9044, 9048, 9449
12712
13488
17584
33722

IBM MQ for Windows y Linux compatibilidad con Unicode



En IBM MQ for Windows y IBM MQ for Linux, se admite la conversión a y desde los CCSID Unicode compatibles (preferiblemente 1200 o 1208) para los CCSID no Unicode de la siguiente lista:

037,
277, 278, 280, 284, 285, 290, 297
300, 301
420, 424, 437
500
813, 819, 833, 835, 836, 837, 838, 850, 852, 855, 856, 857, 858, 860, 861, 862, 863, 864, 865, 866, 867, 868, 869, 870, 871, 874, 875, 878, 880, 891, 897
901, 902, 903, 904, 912, 913^{“5” en la página 1007}, 915, 916, 918, 920, 921, 922, 923, 924, 927, 928, 930, 931^{“1” en la página 1007}, 932^{“2” en la página 1007}, 933, 935, 937, 938^{“3” en la página 1007}, 939, 941, 942, 943, 947, 948, 949, 950, 951, 954^{“4” en la página 1007}, 964, 970
1006, 1025, 1026, 1027, 1040, 1041, 1042, 1043, 1046, 1047, 1051, 1088, 1089, 1097, 1098
1112, 1114, 1115, 1122, 1123, 1124, 1129, 1130, 1132, 1133, 1140, 1141, 1142, 1143, 1144, 1145, 1146, 1147, 1148, 1149, 1153, 1156, 1157
1200, 1208, 1250, 1251, 1252, 1253, 1254, 1255, 1256, 1257, 1258, 1275, 1280, 1281, 1282, 1283
1363, 1364, 1374, 1375, 1376, 1377, 1378, 1379, 1380, 1381, 1383, 1386, 1388
4899
5050, 5346, 5347, 5348, 5349, 5350, 5351, 5352, 5353, 5354, 5488^{“5” en la página 1007}
9044, 9048, 9449

12712
13488
17584
33722⁴ en la página 1007

Notas:

1. 931 utiliza 939 para la conversión.
2. 932 utiliza 942 para la conversión.
3. 938 utiliza 948 para la conversión.
4. 954 y 33722 utilizan 5050 para la conversión.
5. En Windows y Linux solo.

IBM i Compatibilidad con Unicode



Para obtener más información sobre la compatibilidad con UNICODE, consulte la publicación IBM i correspondiente a su sistema operativo.

IBM MQ for z/OS Compatibilidad con Unicode



Se admite la IBM MQ for z/OS conversión a y desde los CCSID Unicode compatibles (preferiblemente 1200 o 1208) para los CCSID no Unicode de la siguiente lista:

37
256, 259, 273, 275, 277, 278, 280, 282, 284, 285, 290, 293, 297
300, 301, 367
420, 423, 424, 437
500
720, 737, 775
803, 806, 808, 813, 819, 833, 834, 835, 836, 837, 838, 848, 849, 850, 851, 852, 855, 856, 857, 858,
859, 860, 861, 862, 863, 864, 865, 866, 867, 868, 869, 870, 871, 872, 874, 875, 878, 880, 891, 895,
896, 897
901, 902, 903, 904, 905, 912, 914, 915, 916, 918, 920, 921, 922, 923, 924, 927, 928, 930, 932, 933,
935, 937, 939, 941, 942, 943, 944, 946, 947, 948, 949, 950, 951
1004, 1006, 1008, 1009, 1010, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1025,
1026, 1027, 1040, 1041, 1042, 1043, 1046, 1047, 1051, 1088, 1089, 1097, 1098
1112, 1114, 1115, 1122, 1123, 1124, 1125, 1126, 1129, 1130, 1131, 1132, 1133, 1137, 1140,
1141, 1142, 1143, 1144, 1145, 1146, 1147, 1148, 1149, 1153, 1154, 1155, 1156, 1157, 1158,
1159, 1160, 1161, 1162, 1164
1200, 1208, 1250, 1251, 1252, 1253, 1254, 1255, 1256, 1257, 1258, 1275, 1276, 1277, 1280,
1281, 1282, 1283, 1284, 1285
1351, 1362, 1363, 1364, 1370, 1371, 1380, 1381, 1385, 1386, 1388, 1390, 1399
4899, 4909, 4930, 4933, 4948, 4951, 4952, 4960, 4971
5012 5039 5104 5123 5142 5210 5346 5347 5348 5349 5350 5351 5352 5353 5354 5488
8482 8612
9027 9030 9044 9048 9049 9056 9061 9066 9238 9449
1166
12712
13121, 13218, 13488, 1374, 1375, 1376, 1377, 1378, 1379
16684, 16804
17 248, 17 584
21427

CCSID que ya no son compatibles

Información sobre los conjuntos de códigos y los CCSID correspondientes a cada conjunto de códigos que no es compatible con IBM Semeru Runtime 21.

Los CCSID no son compatibles con AIX, Linux, and Windows



Los CCSID que figuran en [Tabla 712 en la página 1008](#) no son compatibles con IBM Semeru Runtime 21 en AIX, Linux, and Windows.

Tabla 712. Nombres de conjuntos de códigos y CCSID	
Nombres de conjuntos de códigos	CCSID
x-IBM301	cp301, ibm301, 301, ibm-301
x-IBM720	ibm-720, ibm720, 720, cp720
x-IBM835	ibm835, 835, cp835, ibm-835
x-IBM836	cp836, ibm836, 836, ibm-836
x-IBM837	cp837, ibm837, 837, ibm-837
x-IBM867	ibm867, 867, cp867, ibm-867
x-IBM897	ibm-897, ibm897, 897, cp897
IBM00924	ibm924, ebcdic-latin9--euro, cp924, 924, ibm-924, cp00924, ibm00924, ccsid00924
x-IBM927	ibm-927, ibm927, 927, cp927
x-IBM941	ibm-941, ibm941, 941, cp941
x-IBM947	cp947, ibm947, 947, ibm-947
x-IBM951	ibm-951, ibm951, 951, cp951
x-IBM1027	cp1027, ibm1027, ibm-1027, 1027
x-IBM1041	cp1041, ibm1041, ibm-1041, 1041
x-IBM1043	ibm-1043, 1043, cp1043, ibm1043
x-IBM1088	1088, ibm1088, cp1088, ibm-1088
x-IBM1114	cp1114, ibm-1114, 1114, ibm1114
x-IBM1115	cp1115, ibm1115, ibm-1115, 1115
x-IBM1130	ibm1130, ibm-1130, 1130, cp1130
x-IBM1153	cp01153, ibm1153, ccsid01153, ibm01153, ibm-1153, 1153, cp1153
x-MacRoman	macroman, 1275, ibm1275, cp1275, ibm-1275
x-MacGreek	macgreek, ibm1280, ibm-1280, 1280, cp1280
x-MacTurkish	ibm-1281, 1281, cp1281, ibm1281, macturkish
x-MacCentralEurope	cp1282, ibm-1282, 1282, maccentraleurope, ibm1282
x-MacCyrillic	cp1283, ibm1283, maccyrillic, ibm-1283, 1283

<i>Tabla 712. Nombres de conjuntos de códigos y CCSID (continuación)</i>	
Nombres de conjuntos de códigos	CCSID
x-MacCroatian	ibm1284, ibm-1284,1284,cp1284,maccroatian
x-MacRomania	ibm-1285,1285,cp1285, ibm1285,macromania
x-IBM1351	ibm-1351,1351,cp1351, ibm1351
x-IBM1362	ibm-1362,1362,ibm1362, cp1362
x-IBM1363	1363,ibm1363, cp1363,ibm-1363
x-IBM1370	cp1370, ibm-1370,1370,ibm1370
x-IBM1371	ibm1371,ibm-9563,cp9563,9563,x-ibm9563, cp1371,ibm-1371,1371,ibm9563
x-IBM1380	ibm-1380,1380,cp1380, ibm1380
x-IBM1385	1385,ibm1385, cp1385,ibm-1385
x-IBM1386	cp1386, ibm1386, x-ibm1386,ibm-1386, 1386
x-IBM1388	ibm-1388,1388,cp1388, ibm1388
x-IBM1390	ibm1390, ibm-1390,1390,cp1390
x-IBM1399	ibm-1399,1399,cp1399, ibm1399
x-IBM4933	cp4933, ibm4933,4933,ibm-4933
x-IBM16684	cp16684, ibm16684,ibm-16684,16684
UTF16	UTF_16, Unicode, utf16, UnicodeBig, UCS-2

Los CCSID no son compatibles con z/OS



Los CCSID que figuran en [Tabla 713 en la página 1009](#) no son compatibles con IBM Semeru Runtime 21 en z/OS.

<i>Tabla 713. Nombres de conjuntos de códigos y CSSID</i>	
Nombres de conjuntos de códigos	CCSID
x-IBM301	cp301, ibm301, 301, ibm-301
x-IBM835	ibm835, 835, cp835, ibm-835
x-IBM836	cp836, ibm836, 836,ibm-836
x-IBM837	cp837, ibm837, 837, ibm-837
x-IBM897	ibm-897, ibm897, 897, cp897
x-IBM927	ibm-927, ibm927, 927, cp927
x-IBM941	ibm-941, ibm941, 941, cp941
x-IBM947	cp947, ibm947, 947, ibm-947
x-IBM951	ibm-951, ibm951, 951, cp951
x-IBM1041	cp1041, ibm1041, ibm-1041,1041
x-IBM1043	ibm-1043,1043,cp1043, ibm1043

<i>Tabla 713. Nombres de conjuntos de códigos y CSSID (continuación)</i>	
Nombres de conjuntos de códigos	CCSID
x-IBM1088	1088,ibm1088, cp1088,ibm-1088
x-IBM1114	cp1114, ibm-1114,1114,ibm1114
x-IBM1115	cp1115, ibm1115,ibm-1115,1115
x-IBM1130	ibm1130, ibm-1130,1130,cp1130
x-MacRoman	macroman,1275,ibm1275, cp1275,ibm-1275
x-MacGreek	macgreek,ibm1280,ibm-1280,1280,cp1280
x-MacTurkish	ibm-1281,1281,cp1281, ibm1281,macturkish
x-MacCentralEurope	cp1282, ibm-1282,1282,maccentraleurope,ibm1282
x-MacCyrillic	cp1283, ibm1283,maccyrillic,ibm-1283,1283
x-MacCroatian	ibm1284, ibm-1284,1284,cp1284,maccroatian
x-MacRomania	ibm-1285,1285,cp1285, ibm1285,macromania
x-IBM1351	ibm-1351,1351,cp1351, ibm1351
x-IBM1362	ibm-1362,1362,ibm1362, cp1362
x-IBM1370	cp1370, ibm-1370,1370,ibm1370
x-IBM1380	ibm-1380,1380,cp1380, ibm1380
x-IBM1385	1385,ibm1385, cp1385,ibm-1385
x-IBM1386	cp1386, ibm1386, x-ibm1386,ibm-1386, 1386
x-IBM4933	cp4933, ibm4933,4933,ibm-4933
UTF16	UTF_16, Unicode, utf16, UnicodeBig, UCS-2

Nota: Desde IBM MQ 9.4.5, el Java runtime environment (JRE) en z/OS admite IBM Semeru Runtime 21 excepto los CCSID de [Tabla 713 en la página 1009](#).

Referencia relacionada

[Páginas de códigos disponibles para MFT](#)

Estándares de codificación en plataformas de 64 bits

Utilice esta información para obtener información sobre los estándares de codificación en plataformas de 64 bits y los tipos de datos preferidos.

Tipos de datos preferidos

Estos tipos nunca cambian de tamaño y están disponibles tanto en plataformas IBM MQ de 32 bits como de 64 bits:

Tabla 714. Nombres y longitudes de tipos de datos

Nombre	Longitud
MQLONG	4 bytes
MQULONG	4 bytes
MQINT32	4 bytes

Tabla 714. Nombres y longitudes de tipos de datos (continuación)

Nombre	Longitud
MQUINT32	4 bytes
MQINT64	8 bytes
MQUINT64	8 bytes

Tipos de datos estándar en AIX, Linux, and Windows

Aprenda sobre los tipos de datos estándar en aplicaciones de 32 bits AIX, Linux, and Windows AIX and Linuxy 64 bits.

Aplicaciones AIX and Linux de 32 bits



Tabla 715. Nombres y longitudes de tipos de datos para aplicaciones de AIX and Linux 32 bits

Nombre	Longitud
char	1 byte
short	2 bytes
int	4 bytes
Entero largo	4 bytes
float	4 bytes
double	8 bytes
doble largo	8 bytes
puntero	4 bytes
ptrdiff_t	4 bytes
size_t	4 bytes
tiempo_t	4 bytes
clock_t	4 bytes
wchar_t	4 bytes

 Tenga en cuenta que un AIX wchar_t tiene 2 bytes.

Aplicaciones AIX and Linux de 64 bits



Tabla 716. Nombres y longitudes de los tipos de datos para aplicaciones de AIX and Linux 64 bits

Nombre	Longitud
char	1 byte
short	2 bytes
int	4 bytes
Entero largo	8 bytes
float	4 bytes
double	8 bytes

Tabla 716. Nombres y longitudes de los tipos de datos para aplicaciones de AIX and Linux 64 bits (continuación)

Nombre	Longitud
doble largo	8 bytes
puntero	8 bytes
ptrdiff_t	8 bytes
size_t	8 bytes
tiempo_t	8 bytes
clock_t	4 bytes
wchar_t	4 bytes



Tenga en cuenta que un AIX wchar_t tiene 2 bytes.

Windows Aplicaciones de 64 bits



Tabla 717. Nombres y longitudes de tipos de datos para aplicaciones Windows de 64 bits

Nombre	Longitud
char	1 byte
short	2 bytes
int	4 bytes
Entero largo	4 bytes
float	4 bytes
double	8 bytes
doble largo	8 bytes
puntero	8 bytes
	Tenga en cuenta que todos los punteros son de 8 bytes.
ptrdiff_t	8 bytes
size_t	8 bytes
tiempo_t	8 bytes
clock_t	4 bytes
wchar_t	2 bytes
WORD	2 bytes
DWORD	4 bytes
HANDLE	8 bytes
HFile	4 bytes

Consideraciones sobre la codificación en Windows



MANGO hf;

Utilización


```
hf = CreateFile((LPCTSTR) FileName,
                Access,
                ShareMode,
                xihSecAttsNTRestrict,
                Create,
                AttrAndFlags,
                NULL);
```

No utilizar

```
HFILE hf;
hf = (HFILE) CreateFile((LPCTSTR) FileName,
                        Access,
                        ShareMode,
                        xihSecAttsNTRestrict,
                        Create,
                        AttrAndFlags,
                        NULL);
```

ya que esto produce un error.

size_t len fgets

Utilización

```
size_t len
while (fgets(string1, (int) len, fp) != NULL)
len = strlen(buffer);
```

No utilizar

```
int len;

while (fgets(string1, len, fp) != NULL)
len = strlen(buffer);
```

printf

Utilización

```
printf("My struc pointer: %p", pMyStruc);
```

No utilizar

```
printf("My struc pointer: %x", pMyStruc);
```

Si necesita una salida hexadecimal, debe imprimir los 4 bytes superiores e inferiores por separado.

char *ptr

Utilización

```
char * ptr1;
char * ptr2;
size_t bufLen;

bufLen = ptr2 - ptr1;
```

No utilizar

```
char *ptr1;
char *ptr2;
UINT32 bufLen;

bufLen = ptr2 - ptr1;
```

alignBytes

Utilización

```
alignBytes = (unsigned short) ((size_t) address % 16);
```

No utilizar

```
void *address;
unsigned short alignBytes;

alignBytes = (unsigned short) ((UINT32) address % 16);
```

lon

Utilización

```
len = (UINT32) ((char *) address2 - (char *) address1);
```

No utilizar

```
void *address1;
void *address2;
UINT32 len;

len = (UINT32) ((char *) address2 - (char *) address1);
```

sscanf

Utilización

```
MQLONG SBCSprr;

sscanf(line, "%d", &SBCSprr);
```

No utilizar

```
MQLONG SBCSprr;

sscanf(line, "%ld", &SBCSprr);
```

%ld Intenta introducir un tipo de 8 bytes en un tipo de 4 bytes; utilícelo únicamente %l si está trabajando con un tipo de datos long real. MQLONG, UINT32 y INT32 se definen como cuatro bytes, igual que un int en todas IBM MQ las plataformas:

IBM i Referencia de programación de aplicaciones (ILE/RPG)

Programación de aplicaciones para IBM i.

Utilice esta información para ayudarle a desarrollar aplicaciones para IBM i.

- [“Descripciones de tipos de datos en IBM i” en la página 1016](#)
- [“Llamadas de función en IBM i” en la página 1278](#)
- [“Atributos de objetos en IBM i” en la página 1399](#)
- [“Códigos de retorno para IBM i \(ILE RPG\)” en la página 1447](#)
- [“Normas para validar las opciones de « MQI » en IBM i \(ILE RPG\)” en la página 1448](#)
- [“Codificaciones de máquina en IBM i” en la página 1451](#)
- [“Opciones de informe y distintivos de mensaje en IBM i” en la página 1454](#)

Desuso de la modalidad de compatibilidad para aplicaciones RPG y COBOL en IBM i

IBM i

A partir de IBM MQ for IBM i 9.0, el producto ya no ofrece compatibilidad con aplicaciones RPG (COBOL) que utilizan enlaces dinámicos conocidos como modo de compatibilidad. Esta modalidad de funcionamiento era necesaria para las aplicaciones escritas antes de MQSeries 5.1, y las versiones posteriores del producto proporcionaban un entorno de ejecución compatible para estas aplicaciones, aunque los libros de copias necesarios para compilarlas fueron eliminados en IBM WebSphere MQ 6.0.

El enlace dinámico (modalidad de compatibilidad) se suministraba en los siguientes programas en la biblioteca QMQM, que se ha eliminado en IBM MQ for IBM i 9.0:

- AMQVSTUB
- AMQZSTUB
- QMQM
- MQCLOSE
- MQCONN
- MQDISC
- MQGET
- MQINQ
- MQOPEN
- MQPUT
- MQPUT1
- MQSET

A partir de la IBM MQ for IBM i 9.0, las aplicaciones que utilizan esta modalidad de compatibilidad de operación deben recompilarse para utilizar las llamadas MQ de enlace estático que proporcionan los programas de servicio LIBMQM y LIBMQM_R. Programas de ejemplo como AMQ3PUT4 y AMQ3GET4 muestran cómo utilizar este modelo de programación. Para obtener más información sobre la utilización de estas llamadas MQ, consulte [IBM i Application Programming Reference \(ILE/RPG\)](#).

Notas:

- Hay que recodificar las aplicaciones, que actualmente usan la interfaz CALL 'QMQM', para que usen el programa de servicio LIBMQM en su lugar.

Los objetos de programa y programas de servicio de la lista anterior como, por ejemplo, QMQM, MQCONN, MQPUT, AMQVSTUB y AMQZSTUB, se han eliminado en IBM MQ for IBM i 9.0 y las aplicaciones que se codificaron para usar el modo de compatibilidad dejan de funcionar.

- Si las aplicaciones están vinculadas al programa de servicio LIBMQM en IBM MQ for IBM i 8.0, no debería ser necesario recompilar o volver a vincular esas aplicaciones en IBM MQ for IBM i 9.0 o posteriores.
- No es posible instalar más de una versión de IBM MQ for IBM i en la misma partición.

Si desea averiguar si el programa RPG o COBOL utiliza la modalidad de compatibilidad, utilice el mandato **DSPPGMREF** (Display Program References) para visualizar los programas externos llamados por el programa de aplicación. Si hay referencias a los programas enumerados en esta sección, el programa no se ejecutará en IBM MQ for IBM i 9.0 o posterior. En el siguiente ejemplo de **DSPPGMREF**, la salida muestra objetos de programa que están en desuso MQCONN, MQOPEN, MQCLOSE:

```

Program . . . . . : MYAPPPGM
Library . . . . . : MYLIB
Text 'description'. . . . . : ILE/COBOL SAMPLE PUT TO QUEUE (MQPUT)
Number of objects referenced . . . . . : 5
Object . . . . . : MQCONN
Library . . . . . : *LIBL
Object type . . . . . : *PGM
Object . . . . . : MQOPEN
Library . . . . . : *LIBL
Object type . . . . . : *PGM
Object . . . . . : MQCLOSE
Library . . . . . : *LIBL
Object type . . . . . : *PGM

```

Estos programas deben recompilarse utilizando el método Bound Procedural Call que se describe en [Preparación de programas COBOL en IBM i](#).

Si intenta ejecutar un programa de aplicación en IBM MQ for IBM i 9.0 o posterior que utilice el modo de compatibilidad, el primer error más común es un error de tipo « MCH3401 » al intentar llamar al programa MQCONN o QMQM.

Tareas relacionadas

[Preparación de los programas RPG en IBM i](#)

[Utilizando los programas de ejemplo de RPG para IBM i](#)

IBM i

Descripciones de tipos de datos en IBM i

Esta colección de temas proporciona descripciones de los tipos de datos utilizados en la programación de IBM i.

Convenciones utilizadas en la descripción de tipos de datos

Para cada tipo de datos elemental, esta información proporciona una descripción de su uso, en un formato que es independiente del lenguaje de programación. Esto va seguido de declaraciones típicas en la versión ILE del lenguaje de programación RPG. Las definiciones de tipos de datos elementales se incluyen aquí para proporcionar coherencia. RPG utiliza especificaciones "D" donde los campos de trabajo pueden ser declarados usando los atributos que usted necesita. Sin embargo, puede hacerlo en las especificaciones de cálculo donde se utiliza el campo.

Para utilizar los tipos de datos elementales, cree:

- Un miembro /COPY que contiene todos los tipos de datos, o
- Estructura de datos externa (PF) que contiene todos los tipos de datos. A continuación, debe especificar los campos de trabajo con los atributos 'LIKE', el campo de tipo de datos adecuado.

Las ventajas de la segunda opción son que las definiciones se pueden utilizar como 'FIELD REFERENCE FILE' para otros objetos de IBM i. Si una definición de tipo de datos IBM MQ cambia, es relativamente sencillo volver a crear estos objetos.

Tipos de datos elementales en IBM i

Todos los demás tipos de datos descritos en esta sección equivalen directamente a estos tipos de datos elementales o a agregados de estos tipos de datos elementales (matrices o estructuras).

Tabla 718. Nombres, tipos y descripciones de los tipos de datos elementales

Tipo de datos elemental	Tipo de datos	Descripción
MQBOOL	entero con signo de 10 dígitos	El tipo de datos MQBOOL representa un valor booleano. El valor 0 representa falso. Cualquier otro valor representa verdadero. Un MQBOOL debe alinearse como el tipo de datos MQLONG.

Tabla 718. Nombres, tipos y descripciones de los tipos de datos elementales (continuación)

Tipo de datos elemental	Tipo de datos	Descripción
MQBYTE	Campo alfanumérico de 1 byte	El tipo de datos MQBYTE representa un solo byte de datos. No se aplica ninguna interpretación particular al byte. Se trata como una cadena de bits, y no como un número binario o un carácter. No se requiere ninguna alineación especial. A veces se utiliza una matriz de MQBYTE para representar un área de la memoria principal cuya naturaleza desconoce el gestor de colas. Por ejemplo, el área podría contener datos de mensajes de aplicación o una estructura. La alineación de los límites de esta zona debe ser compatible con la naturaleza de los datos que contiene.
MQBYTE16	Campo alfanumérico de 16 bytes	Cada tipo de datos MQBYTE n representa una cadena de n bytes, donde n puede tomar uno de los siguientes valores: 16, 24, 32 o 64. Cada byte se describe mediante el tipo de datos MQBYTE. No se requiere ninguna alineación especial. Si los datos de la cadena son más cortos que la longitud definida de la cadena, los datos deben rellenarse con valores nulos para completar la cadena. Cuando el gestor de colas devuelve cadenas de bytes a la aplicación (por ejemplo, en la llamada MQGET), el gestor de colas siempre rellena con valores nulos hasta alcanzar la longitud definida de la cadena. Hay constantes disponibles que definen las longitudes de los campos de cadena de bytes.
MQBYTE24	Campo alfanumérico de 24 bytes	
MQBYTE32	Campo alfanumérico de 32 bytes	
MQBYTE64	Campo alfanumérico de 64 bytes	

Tabla 718. Nombres, tipos y descripciones de los tipos de datos elementales (continuación)

Tipo de datos elemental	Tipo de datos	Descripción
MQCHAR	Campo alfanumérico de 1 byte	El tipo de datos MQCHAR representa un solo carácter. El identificador del conjunto de caracteres codificados del carácter es el del gestor de colas (véase el CodedCharSetId atributo en el tema CodedCharSetId). No se requiere ninguna alineación especial. Nota: Los datos de los mensajes de aplicación especificados en las llamadas MQGET, MQPUT y MQPUT1 se describen mediante el tipo de datos MQBYTE, no mediante el tipo de datos MQCHAR.

Tabla 718. Nombres, tipos y descripciones de los tipos de datos elementales (continuación)

Tipo de datos elemental	Tipo de datos	Descripción
MQCHAR4	Campo alfanumérico de 4 bytes	Cada tipo de datos <i>MQ CHAR</i> <i>n</i> representa una cadena de <i>n</i> caracteres, donde <i>n</i> puede tomar uno de los siguientes valores: 4, 8, 12, 16, 20, 28, 32, 48, 64, 128 o 256.
MQCHAR8	Campo alfanumérico de 8 bytes	
MQCHAR12	Campo alfanumérico de 12 bytes	
MQCHAR16	Campo alfanumérico de 16 bytes	
MQCHAR20	Campo alfanumérico de 20 bytes	Cada carácter se describe mediante el tipo de datos MQCHAR. No se requiere ninguna alineación especial.
MQCHAR28	Campo alfanumérico de 28 bytes	
MQCHAR32	Campo alfanumérico de 32 bytes	
MQCHAR48	Campo alfanumérico de 48 bytes	
MQCHAR64	Campo alfanumérico de 64 bytes	Si los datos de la cadena son más cortos que la longitud definida de la cadena, los datos deben rellenarse con espacios en blanco para completar la cadena. En algunos casos, se puede utilizar un carácter nulo para terminar la cadena prematuramente, en lugar de rellenarla con espacios en blanco; el carácter nulo y los caracteres que le siguen se tratan como espacios en blanco, hasta la longitud definida de la cadena. Los lugares donde se puede utilizar un valor nulo se identifican en las descripciones de las llamadas y los tipos de datos.
MQCHAR128	Campo alfanumérico de 128 bytes	
MQCHAR256	Campo alfanumérico de 256 bytes	
MQFLOAT32	Número de coma flotante de 4 bytes	El tipo de datos ` MQFLOAT32 ` es un número de 32 bits en coma flotante representado utilizando el formato estándar IEEE en coma flotante. Un objeto de interfaz de programa de aplicación (MQFLOAT32) debe estar alineado en un límite de 4 bytes.

Tabla 718. Nombres, tipos y descripciones de los tipos de datos elementales (continuación)

Tipo de datos elemental	Tipo de datos	Descripción
MQFLOAT64	Número de coma flotante de 8 bytes	El tipo de datos `MQFLOAT64` es un número de coma flotante de 64 bits representado utilizando el formato de coma flotante estándar IEEE. Un objeto de interfaz de programa de aplicación (MQFLOAT64) debe estar alineado en un límite de 8 bytes.
MQHCONFIG	Manejo de la configuración	El tipo de datos MQHCONFIG representa un identificador de configuración, es decir, el componente que se está configurando para un servicio instalable concreto. Un identificador de configuración debe alinearse con su límite natural. Nota: Las aplicaciones deben comprobar únicamente la igualdad de las variables de este tipo.
MQHCONN	entero con signo de 10 dígitos	El tipo de datos MQHCONN representa un identificador de conexión, es decir, la conexión a un gestor de colas concreto. Un identificador de conexión debe alinearse con su límite natural. Nota: Las aplicaciones deben comprobar únicamente la igualdad de las variables de este tipo.
MQHMSG	Descriptor de contexto de mensaje	El tipo de datos MQHMSG representa un identificador de mensaje que da acceso a un mensaje. Un identificador de mensaje debe estar alineado en un límite de 8 bytes. Nota: Las aplicaciones deben comprobar únicamente la igualdad de las variables de este tipo.

Tabla 718. Nombres, tipos y descripciones de los tipos de datos elementales (continuación)

Tipo de datos elemental	Tipo de datos	Descripción
MQHOBj	entero con signo de 10 dígitos	El tipo de datos MQHOBj representa un identificador de objeto que da acceso a un objeto. Un identificador de objeto debe estar alineado con su límite natural. Nota: Las aplicaciones deben comprobar únicamente la igualdad de las variables de este tipo.
MQINT8	entero con signo de 8 bits	El tipo de datos `MQINT8` es un entero con signo de 8 bits que puede tomar cualquier valor en el rango de -128 a +127, a menos que el contexto lo restrinja de otra manera.
MQINT16	Entero con signo de 16 bits	El tipo de datos `MQINT16` es un entero con signo de 16 bits que puede tomar cualquier valor en el rango de -32 768 a +32 767, a menos que el contexto lo restrinja de otra manera. Un objeto de interfaz de programa de aplicación (MQINT16) debe estar alineado en un límite de 2 bytes.
MQINT32	Entero con signo de 32 bits	El tipo de datos «MQINT32» es un entero con signo de 32 bits. Es equivalente a <u>MQLONG</u> .
MQINT64	Entero con signo de 64 bits	El tipo de datos `MQINT64` es un entero con signo de 64 bits que puede tomar cualquier valor en el rango comprendido entre ` -9 223372036854775808` y ` +9223372036854775807`, a menos que el contexto imponga restricciones. Para COBOL, el rango válido está limitado a -999 999 999 999 999 999 hasta +999 999 999 999 999 999. Un objeto de interfaz de programa de aplicación (MQINT64) debe alinearse en un límite de 8 bytes.

Tabla 718. Nombres, tipos y descripciones de los tipos de datos elementales (continuación)

Tipo de datos elemental	Tipo de datos	Descripción
MQLONG	Entero con signo de 32 bits	El tipo de datos MQLONG es un entero binario con signo de 32 bits que puede tomar cualquier valor en el rango -2 147 483 648 a +2 147 483 647, a menos que el contexto lo restrinja de otro modo, alineado en su límite natural.
MQPID	Identificador proceso	El IBM MQ identificador del proceso. Este es el mismo identificador que se utiliza en IBM MQ trace y FFST dumps, pero puede ser diferente del identificador de proceso del sistema operativo.
MQPTR	Puntero	El tipo de datos MQPTR es la dirección de datos de cualquier tipo. Un puntero debe estar alineado en su límite natural; este es un límite de 16 bytes en IBM i. Algunos lenguajes de programación admiten punteros tipados; el lenguaje de programación « MQI » también los utiliza en algunos casos.
MQTID	Identificador de hebra	El identificador de hebra de IBM MQ. Este es el mismo identificador que se utiliza en IBM MQ trace y FFST dumps, pero puede ser diferente del identificador de subprocesos del sistema operativo.
MQUINT8	Entero sin signo de 8 bits	El tipo de datos ` MQUINT8 ` es un entero sin signo de 8 bits que puede tomar cualquier valor en el rango de 0 a +255, a menos que el contexto lo restrinja de otra manera.
MQUINT16	entero sin signo de 16 bits	El tipo de datos ` MQUINT16 ` es un entero sin signo de 16 bits que puede tomar cualquier valor en el rango de 0 a +65 535, a menos que el contexto lo restrinja. Un objeto de interfaz de programa de aplicación (MQUINT16) debe estar alineado en un límite de 2 bytes.

Tabla 718. Nombres, tipos y descripciones de los tipos de datos elementales (continuación)

Tipo de datos elemental	Tipo de datos	Descripción
MQUINT32	Entero de 32 bits sin signo	El tipo de datos « MQUINT32 » es un entero sin signo de 32 bits. Es equivalente a <u>MQULONG</u> .
MQUINT64	Entero de 64 bits sin signo	El tipo de datos ` MQUINT64 ` es un entero sin signo de 64 bits que puede tomar cualquier valor en el rango de 0 a +18 446 744 073 709 551 615, a menos que el contexto lo restrinja. Para COBOL, el rango válido está limitado a 0 a +999 999 999 999 999 999. Un objeto de interfaz de programa de aplicación (MQUINT64) debe alinearse en un límite de 8 bytes.
MQULONG	Entero de 32 bits sin signo	El tipo de datos MQULONG es un entero binario sin signo de 32 bits que puede tomar cualquier valor en el rango de 0 a +4 294 967 294, a menos que el contexto lo restrinja de otra manera. Un MQULONG debe estar alineado en un límite de 4 bytes.
PMQACH	Puntero	Un puntero a una estructura de datos de tipo MQACH.
PMQAIR	Puntero	Un puntero a una estructura de datos de tipo <u>MQAIR</u> .
PMQAXC	Puntero	Un puntero a una estructura de datos de tipo MQAXC.
PMAXP	Puntero	Un puntero a una estructura de datos de tipo MQAXP.
PMQBMHO	Puntero	Un puntero a una estructura de datos de tipo <u>MQBMHO</u> .
PMQBO	Puntero	Un puntero a una estructura de datos de tipo <u>MQBO</u> .
PMQBOOL	Puntero	Un puntero a datos de tipo MQBOOL.
PMQBYTE	Puntero	Un puntero a un tipo de datos MQBYTE.
PMQBYTEn	Puntero	Un puntero a una estructura de datos de tipo MQBYTE <i>n</i> , donde <i>n</i> puede ser 8, 12, 16, 24, 32, 40, 48 o 128.

Tabla 718. Nombres, tipos y descripciones de los tipos de datos elementales (continuación)

Tipo de datos elemental	Tipo de datos	Descripción
PMQCBC	Puntero	Un puntero a una estructura de datos de tipo MQCBC.
PMQCBD	Puntero	Un puntero a una estructura de datos de tipo MQCBD.
PMQCHAR	Puntero	Un puntero a datos de tipo MQCHAR.
PMQCHARV	Puntero a una estructura de datos de tipo MQCHARV	Un puntero a una estructura de datos de tipo MQCHARV.
PMQCHARn	Puntero a datos de tipo MQCHARn	Un puntero a un tipo de datos MQ CHARn, donde n puede ser 4, 8, 12, 20, 28, 32, 64, 128, 256, 264.
PMQCIH	Puntero	Un puntero a una estructura de datos de tipo MQCIH.
PMQCMHO	Puntero	Un puntero a una estructura de datos de tipo MQCMHO.
PMQCNO	Puntero	Un puntero a una estructura de datos de tipo MQCNO.
PMQCSP	Puntero	Un puntero a una estructura de datos de tipo MQCSP.
PMQCTLO	Puntero	Un puntero a una estructura de datos de tipo MQCTLO.
PMQDH	Puntero	Un puntero a una estructura de datos de tipo MQDH.
PMQDHO	Puntero	Un puntero a una estructura de datos de tipo MQDHO.
PMQDLH	Puntero	Un puntero a una estructura de datos de tipo MQDLH.
PMQDMHO	Puntero	Un puntero a una estructura de datos de tipo MQDMHO.
PMQDMPO	Puntero	Un puntero a una estructura de datos de tipo MQDMPO.
PMQEPH	Puntero	Un puntero a una estructura de datos de tipo MQEPH.
PMQFLOAT32	Puntero	Un puntero a datos de tipo MQFLOAT32.
PMQFLOAT64	Puntero	Un puntero a datos de tipo MQFLOAT64.
PMQFUNC	Puntero	Un puntero a una función.
PMQGMO	Puntero	Un puntero a una estructura de datos de tipo MQGMO.

Tabla 718. Nombres, tipos y descripciones de los tipos de datos elementales (continuación)

Tipo de datos elemental	Tipo de datos	Descripción
PMQHCONFIG	Puntero	Un puntero a un tipo de datos de MQHCONFIG.
PMQHCONN	Puntero	Un puntero a un tipo de datos de MQHCONN.
PMQHMSG	Puntero	Un puntero a un tipo de datos MQHMSG.
PMQHOBJ	Puntero a datos de tipo MQHOBJ	Un puntero a datos de tipo MQSMPO.
PMQIIH	Puntero	Un puntero a una estructura de datos de tipo MQIIH.
PMQIMPO	Puntero	Un puntero a una estructura de datos de tipo MQIMPO.
PMQINT8	Puntero	Un puntero a datos de tipo MQINT8.
PMQINT16	Puntero	Un puntero a datos de tipo MQINT16.
PMQINT32	Puntero	El tipo de datos `PMQINT32` es un puntero a datos de tipo `MQINT32`. Es equivalente a PMQLONG .
PMQINT64	Puntero	El tipo de datos `PMQINT64` es un puntero a datos de tipo `MQINT64`.
PMQLONG	Puntero	Un puntero a datos de tipo MQLONG.
PMQMD	Puntero	Un puntero a una estructura de tipo MQMD.
PMQMDE	Puntero	Un puntero a una estructura de datos de tipo MQMDE.
PMQMD1	Puntero	Un puntero a una estructura de datos de tipo MQMDI.
PMQMD2	Puntero	Un puntero a una estructura de datos de tipo MQMD2.
PMQMHBO	Puntero	Un puntero a una estructura de datos de tipo MQMHBO.
PMQOD	Puntero	Un puntero a una estructura de datos de tipo MQOD.
PMQOR	Puntero	Un puntero a una estructura de datos de tipo MQOR.
PMQPD	Puntero	Un puntero a una estructura de datos de tipo MQPD.
PMQPID	Puntero	Un puntero a un identificador de proceso.

Tabla 718. Nombres, tipos y descripciones de los tipos de datos elementales (continuación)

Tipo de datos elemental	Tipo de datos	Descripción
PMQPMO	Puntero	Un puntero a una estructura de datos de tipo MQPMO.
PMQPTR	Puntero	Un puntero a datos de tipo MQPTR.
PMQRFH	Puntero	Un puntero a una estructura de datos de tipo MQRFH.
PMQRFH2	Puntero	Un puntero a una estructura de datos de tipo MQRFH2.
PMQRMH	Puntero	Un puntero a una estructura de datos de tipo MQRMH.
PMQRR	Puntero	Un puntero a una estructura de datos de tipo MQRR.
PMQSCO	Puntero	Un puntero a una estructura de datos de tipo MQSCO.
PMQSD	Puntero	Un puntero a una estructura de datos de tipo MQSD.
PMQSMPO	Puntero	Un puntero a una estructura de datos de tipo MQSMPO.
PMQSRO	Puntero	Un puntero a una estructura de datos de tipo MQSRO.
PMQSTS	Puntero	Un puntero a una estructura de datos de tipo MQSTS.
PMQTID	Puntero	Un puntero a una estructura de datos de tipo MQTID.
PMQTM	Puntero	Un puntero a una estructura de datos de tipo MQTM.
PMQTM2	Puntero	Un puntero a una estructura de datos de tipo MQTM2.
PMQUINT8	Puntero	Un puntero a datos de tipo MQUINT8.
PMQUINT16	Puntero	Un puntero a datos de tipo MQUINT16.
PMQUINT32	Puntero	El tipo de datos `PMQUINT32` es un puntero a datos de tipo `MQUINT32`. Es equivalente a PMQULONG .
PMQUINT64	Puntero	El tipo de datos `PMQUINT64` es un puntero a datos de tipo `MQUINT64`.
PMQULONG	Puntero	Un puntero a datos de tipo MQULONG.
PMQVOID	Puntero	Un puntero.

Tabla 718. Nombres, tipos y descripciones de los tipos de datos elementales (continuación)		
Tipo de datos elemental	Tipo de datos	Descripción
¿Me lo preguntas?	Puntero	Un puntero a una estructura de datos de tipo MQWIH.
PMQXQH	Puntero	Un puntero a una estructura de datos de tipo MQXQH.

Consideraciones sobre idiomas

Este tema contiene información que te ayudará a utilizar el servicio « MQI » desde el lenguaje de programación RPG.

Algunas de estas consideraciones lingüísticas son:

- “Archivos de copia” en la [página 1027](#)
- “Llamadas” en la [página 1029](#)
- “parámetros de llamada” en la [página 1029](#)
- “Estructuras” en la [página 1030](#)
- “Constantes con nombre” en la [página 1030](#)
- “MQI procedimientos” en la [página 1030](#)
- “Consideraciones sobre hebras” en la [página 1030](#)
- “Control de compromiso” en la [página 1031](#)
- “Codificación de las llamadas enlazadas” en la [página 1031](#)
- “convenios de anotación” en la [página 1032](#)

Archivos de copia

Se proporcionan varios archivos COPY para ayudar con la escritura de programas de aplicación RPG que utilizan la colocación de mensajes en colas. Hay tres conjuntos de archivos COPY:

- Los archivos COPY con nombres que terminan con la letra *G* se utilizan con programas que utilizan enlaces estáticos. Estos archivos se inicializan con las excepciones indicadas en [“Estructuras” en la página 1030](#).
- Los archivos COPY con nombres que terminan con la letra *H* se utilizan con programas que utilizan enlaces estáticos, pero **no** se han inicializado.
- Los archivos COPY con nombres que terminan con la letra *R* se utilizan con programas que utilizan enlaces dinámicos. Estos archivos se inicializan con las excepciones indicadas en [“Estructuras” en la página 1030](#).

Los archivos COPY residen en QRPGLSRC en la biblioteca QMQM.

Para cada conjunto de archivos COPY, hay dos archivos que contienen constantes con nombre y un archivo para cada una de las estructuras. Los archivos COPY se resumen en [Tabla 719 en la página 1027](#).

Tabla 719. Archivos COPY de RPG			
Nombre de archivo (enlace estático, inicializado, CMQ* G)	Nombre de archivo (enlace estático, no inicializado, CMQ* H)	Nombre de archivo (enlace dinámico, inicializado, CMQ* R)	Contenido
CMQBOG	CMQBOH	-	Estructura de opciones de inicio
CMQCDG	CMQCDH	CMQCDR	Estructura de definición de canal

Tabla 719. Archivos COPY de RPG (continuación)

Nombre de archivo (enlace estático, inicializado, CMQ* G)	Nombre de archivo (enlace estático, no inicializado, CMQ* H)	Nombre de archivo (enlace dinámico, inicializado, CMQ* R)	Contenido
CMQCFBFG	CMQCFBFH	-	Parámetro de filtro de bits PCF
CMQCFG	-	-	Constantes para PCF y sucesos
CMQCFBSG	CMQCFBSH	-	Serie de bytes PCF
CMQCFGRG	CMQCFGRH	-	parámetro de grupo PCF
CMQCFIFG	CMQCFIFH	-	Parámetro de filtro de enteros PCF
CMQCFHG	CMQCFHH	-	Cabecera PCF
CMQCFILG	CMQCFILH	-	Estructura de parámetros de lista de enteros PCF
CMQCFING	CMQCFINH	-	Estructura de parámetros enteros PCF
CMQCFSG	CMQCFSH	-	Parámetro de filtro de serie PCF
CMQCFSLG	CMQCFSLH	-	Estructura de parámetros de lista de series PCF
CMQCFSTG	CMQCFSTH	-	Estructura de parámetros de serie PCF
CMQCFXLG	CMQCFXLH	-	Nombre abreviado PCF para CFIL64
CMQCFXNG	CMQCFXNH	-	Nombre abreviado PCF para CFIN64
CMQCIHG	CMQCIHH	-	Estructura de cabecera de información de CICS
CMQCNOG	CMQCNOH	-	Estructura de opciones de conexión
CMQCSPG	CMQCSPH	-	Parámetros de seguridad
CMQCXPG	CMQCXPH	CMQCXPR	Estructura de parámetros de salida de canal
CMQDHG	CMQDHH	CMQDHR	Estructura de cabecera de distribución
CMQDLHG	CMQDLHH	CMQDLHR	Estructura de cabecera de mensaje no entregado
CMQDXPG	CMQDXPH	CMQDXPR	Estructura de parámetros de salida de conversión de datos
CMQEPHG	CMQEPHH	-	Estructura de cabecera PCF incorporada
CMQG	-	CMQR	Constantes con nombre para la función principal `MQI`
CMQGMOG	CMQGMOH	CMQGMOR	Estructura de opciones de obtención de mensajes
CMQIIHG	CMQIIHH	CMQIIHR	Estructura de cabecera de información de IMS
CMQMDEG	CMQMDEH	CMQMDER	Estructura de extensión de descriptor de mensaje

Tabla 719. Archivos COPY de RPG (continuación)

Nombre de archivo (enlace estático, inicializado, CMQ* G)	Nombre de archivo (enlace estático, no inicializado, CMQ* H)	Nombre de archivo (enlace dinámico, inicializado, CMQ* R)	Contenido
CMQMDG	CMQMDH	Comandante de la Flota	Estructura del descriptor de mensaje
CMQMD1G	CMQMD1H	CMQMD1R	Estructura de descriptor de mensaje versión 1
CMQMD2G	CMQMD2H	-	Estructura de descriptor de mensaje versión 2
CMQODG	CMQODH	CMQODR	Estructura de descriptor de objeto
CMQORG	CMQORH	CMQORR	Estructura de registro de objeto
CMQPMOG	CMQPMOH	CMQPMOR	Estructura de opciones de colocación de mensaje
CMQPSG	-	-	Constantes para publicación/suscripción
CMQRFHG	CMQRFHH	-	Reglas y estructura de cabecera de formato
CMQRFH2G	CMQRFH2H	-	Estructura de reglas y formateo de cabecera 2
CMQRMHG	CMQRMHH	CMQRMRRHH	Estructura de cabecera de mensaje de referencia
CMQRRG	CMQRRH	CMQRRR	Estructura de registro de respuesta
CMQTMCG	CMQTMCH	CMQTMCR	Estructura de mensaje desencadenante (formato de caracteres)
CMQTMCG2G	CMQTMCG2H	CMQTMCG2R	Estructura de mensaje desencadenante (formato de caracteres) versión 2
CMQTMG	CMQTMH	CMQTMR	Estructura de mensajes desencadenantes
CMQWIHG	CMQWIHH	-	Estructura de cabecera de información de trabajo
CMQXG	-	CMQXR	Constantes con nombre para la salida de conversión de datos
CMQXQHG	CMQXQHH	CMQXQHR	Estructura de cabecera de cola de transmisión

Llamadas

Las llamadas se describen utilizando sus nombres individuales.

parámetros de llamada

Algunos parámetros pasados a la función « MQI » pueden tener más de una función simultánea. Esto se debe a que el valor entero pasado a menudo se prueba en el valor de bits individuales dentro del campo, y no en su valor total. Esto le permite 'añadir' varias funciones juntas y pasarlas como un único parámetro.

Estructuras

Todas las estructuras IBM MQ se definen con valores iniciales para los campos, con las excepciones siguientes:

- Cualquier estructura con un sufijo de H.
- MQTMC
- MQTMC2

Estos valores iniciales se definen en la tabla correspondiente para cada estructura.

Las declaraciones de estructura no contienen sentencias DS . Esto permite a la aplicación declarar una sola estructura de datos o una estructura de datos de varias apariciones, codificando la sentencia DS y, a continuación, utilizando la sentencia /COPY para copiar en el resto de la declaración:

```
D*..1.....2.....3.....4.....5.....6.....7
D* Declare an MQMD data structure with 5 occurrences
DMYMD          DS          5
D/COPY CMQMDR
```

Constantes con nombre

Hay muchos valores enteros y de caracteres que proporcionan el intercambio de datos entre el programa de aplicación y el gestor de colas. Para facilitar un enfoque más legible y coherente para utilizar estos valores, se definen constantes con nombre para ellos. Puede utilizar estas constantes con nombre y no los valores que representan, ya que esto mejora la legibilidad del código fuente del programa.

Cuando el archivo COPY CMQG se incluye en un programa para definir las constantes, el compilador RPG emitirá muchos mensajes de gravedad cero para las constantes que no utiliza el programa; estos mensajes son benignos y se pueden ignorar de forma segura.

MQI procedimientos

Al utilizar las llamadas enlazadas de ILE, debes enlazar los procedimientos de MQI al crear tu programa. Estos procedimientos se exportan desde los siguientes programas de servicio según corresponda:

QMQM/LIBMQM

Este programa de servicio contiene los enlaces de una sola hebra para la versión 5.1 y posteriores. Consulte la sección siguiente para obtener consideraciones especiales al escribir aplicaciones con hebras.

QMQM/LIBMQM_R

Este programa de servicio contiene los enlaces multihebra para la versión 5.1 y posteriores. Consulte la sección siguiente para obtener consideraciones especiales al escribir aplicaciones con hebras.

QMQM/LIBMQIC

Este programa de servicio es para enlazar aplicaciones cliente sin hebras.

QMQM/LIBMQIC_R

Este programa de servicio es para enlazar aplicaciones cliente con hebras.

Utilice el mandato CRTPGM para crear los programas. Por ejemplo, el mandato siguiente crea un programa de una sola hebra que utiliza las llamadas enlazadas ILE:

```
CRTPGM PGM(MYPROGRAM) BNDSRVPGM(QMQM/LIBMQM)
```

Consideraciones sobre hebras

El compilador RPG utilizado para IBM i forma parte de WebSphere Development Toolset y WebSphere Development Studio para IBM i y se conoce como compilador ILE RPG IV.

En general, los programas RPG no deben utilizar los programas de servicio multihebra. Las excepciones son programas RPG creados utilizando el compilador ILE RPG IV y que contienen la palabra clave THREAD(*SERIALIZE) en la especificación de control. Sin embargo, aunque estos programas sean

seguros para hebras, se debe tener muy en cuenta el diseño general de la aplicación, ya que THREAD(*SERIALIZE) fuerza la serialización de los procedimientos RPG a nivel de módulo, y esto puede tener un efecto adverso en el rendimiento global.

Cuando los programas RPG se utilizan como salidas de conversión de datos, se deben convertir en hebras seguras y se deben volver a compilar utilizando el compilador ILE RPG de la versión 4.4 o superior, con THREAD(*SERIALIZE) especificado en la especificación de control.

Para obtener más información sobre las hebras, consulte *IBM i IBM MQ Development Studio: ILE RPG Referencey IBM i IBM MQ Development Studio: ILE RPG Programmer's Guide*.

Control de compromiso

Las funciones de punto de sincronización MQCMIT y MQBACK de « MQI » están disponibles para los programas ILE RPG que se ejecutan en modo normal; estas llamadas permiten al programa confirmar o deshacer los cambios realizados en los recursos de « MQ ».

Codificación de las llamadas enlazadas

MQI Los procedimientos ILE se enumeran en [Tabla 720 en la página 1031](#).

Tabla 720. Llamadas enlazadas ILE RPG soportadas por cada programa de servicio		
Nombre de la llamada	LIBMQM y LIBMQM_R	LIBMQIC y LIBMQIC_R
MQBACK	Y	Y
MQBEGIN	Y	Y
MQCMIT	Y	Y
MQCLOSE	Y	Y
MQCONN	Y	Y
MQCONNX	Y	Y
MQDISC	Y	Y
MQGET	Y	Y
MQINQ	Y	Y
MQOPEN	Y	Y
MQPUT	Y	Y
MQPUT1	Y	Y
MQSET	Y	Y
MQXCNVC	Y	Y

Para utilizar estos procedimientos, debe:

1. Defina los procedimientos externos en las especificaciones "D". Todos ellos están disponibles dentro del miembro de archivo COPY CMQG que contiene las constantes con nombre.
2. Utilice el código de operación CALLP para llamar al procedimiento junto con sus parámetros.

Por ejemplo, la llamada MQOPEN requiere la inclusión del código siguiente:

```
D*****
D**  MQOPEN Call -- Open Object (From COPY file CMQG)      **
D*****
D*
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
DMQOPEN          PR          EXTPROC('MQOPEN')
D* Connection handle
```

```

D HCONN                10I 0 VALUE
D* Object descriptor
D OBJDSC                224A
D* Options that control the action of MQOPEN
D OPTS                 10I 0 VALUE
D* Object handle
D HOBJ                 10I 0
D* Completion code
D CMPCOD               10I 0
D* Reason code qualifying CMPCOD
D REASON               10I 0
D*

```

Para llamar al procedimiento, después de inicializar los diversos parámetros, necesita el código siguiente:

```

...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8
  C          CALLP      MQOPEN(HCONN : MQOD : OPTS : HOBJ :
  C          CMPCOD : REASON)

```

Aquí, la estructura MQOD se define utilizando el miembro COPY CMQODG que lo divide en sus componentes.

convenios de anotación

Los últimos temas de esta sección muestran cómo:

- Se deben invocar las llamadas
- Los parámetros deben declararse
- Deben declararse varios tipos de datos

En varios casos, los parámetros son matrices o series de caracteres con un tamaño que no es fijo. Para estos, se utiliza una "n" en minúsculas para representar una constante numérica. Cuando la declaración para ese parámetro está codificada, "n" debe sustituirse por el valor numérico necesario.

MQAIR (Registro de información de autenticación) en IBM i

La estructura MQAIR representa el registro de información de autenticación.

Visión general

Propósito : La estructura MQAIR permite que una aplicación que se ejecuta como cliente IBM MQ MQAIR especifique información sobre un autenticador que se utilizará para la conexión del cliente. La estructura es un parámetro de entrada en la llamada MQCONN.

Conjunto de caracteres y codificación : Los datos en MQAIR deben estar en el conjunto de caracteres proporcionado por el atributo del **CodedCharSetId** gestor de colas y la codificación del gestor de colas local proporcionada por ENNAT.

- [“Campos” en la página 1032](#)
- [“Valores iniciales” en la página 1034](#)
- [“Declaración RPG” en la página 1035](#)

Campos

La estructura MQAIR contiene los siguientes campos, que se describen en **orden alfabético** :

AICN (entero con signo de 10 dígitos)

Se trata del nombre de host o la dirección de red de un host en el que se ejecuta el servidor LDAP. A continuación puede aparecer un número de puerto opcional, entre paréntesis.

Si el valor es más corto que la longitud del campo, termine el valor con un carácter nulo o rellénelo con espacios en blanco hasta alcanzar la longitud del campo. Si el valor no es válido, la llamada falla con el código de motivo RC2387.

El número de puerto predeterminado es 389.

Este es un campo de entrada. La longitud de este campo viene dada por LNAICN. El valor inicial de este campo son caracteres en blanco.

AITYP (entero con signo de 10 dígitos)

Este es el tipo de información de autenticación que contiene el registro.

El valor debe ser:

AITLDP

Revocación de certificados mediante un servidor LDAP.

Si el valor no es válido, la llamada falla con el código de motivo RC2386.

Este es un campo de entrada. El valor inicial de este campo es AITLDP.

AIPW (entero con signo de 10 dígitos)

Esta es la contraseña necesaria para acceder al servidor CRL LDAP.

Si el valor es más corto que la longitud del campo, termine el valor con un carácter nulo o rellénelo con espacios en blanco hasta alcanzar la longitud del campo. Si el servidor LDAP no requiere contraseña, u omite el nombre de usuario LDAP, *AIPW* debe ser nulo o estar en blanco. Si omite el nombre de usuario LDAP y no *AIPW* es nulo ni está en blanco, la llamada fallará con el código de motivo RC2390.

Este es un campo de entrada. La longitud de este campo viene dada por LNLDPW. El valor inicial de este campo es caracteres en blanco.

AILUL (entero con signo de 10 dígitos)

Es la longitud en bytes del nombre de usuario LDAP al que se refiere el *AILUP* campo *AILUO* o. El valor debe estar comprendido entre cero y LNDISN. Si el valor no es válido, la llamada falla con el código de motivo RC2389.

Si el servidor LDAP involucrado no requiere un nombre de usuario, establezca este campo en cero.

Este es un campo de entrada. El valor inicial de este campo es 0.

AILUO (entero con signo de 10 dígitos)

Este es el desplazamiento en bytes del nombre de usuario LDAP desde el inicio de la estructura MQAIR.

El desplazamiento puede ser positivo o negativo. El campo se ignora si *LDAPUserNameLength* es cero.

Puede utilizar o *LDAPUserNameOffsetLDAPUserNamePtr* para especificar el nombre de usuario LDAP, pero no ambos; consulte la descripción del *LDAPUserNamePtr* campo para obtener más detalles.

Este es un campo de entrada. El valor inicial de este campo es 0.

AILUP (entero con signo de 10 dígitos)

Este es el nombre de usuario LDAP.

Consiste en el nombre distintivo del usuario que intenta acceder al servidor CRL LDAP. Si el valor es más corto que la longitud especificada por *AILUL*, termine el valor con un carácter nulo o rellénelo con espacios en blanco hasta alcanzar la longitud *AILUL*. El campo se ignora si *AILUL* es cero.

Puede proporcionar el nombre de usuario LDAP de dos maneras:

- Al utilizar el campo puntero *AILUP*

En este caso, la aplicación puede declarar una cadena separada de la estructura MQAIR y establecer la dirección *AILUP* de la cadena.

Considere utilizar *AILUP* para lenguajes de programación que admitan el tipo de datos puntero de forma que sea portátil a diferentes entornos (por ejemplo, el lenguaje de programación C).

- Al utilizar el campo de compensación *AILUO*

En este caso, la aplicación debe declarar una estructura compuesta que contenga la estructura MQSCO seguida de la matriz de registros MQAIR seguida de las cadenas de nombre de usuario LDAP, y establecer *AILUO* en la compensación de la cadena de nombre adecuada desde el inicio de la estructura MQAIR. Asegúrese de que este valor sea correcto y tenga un valor que pueda acomodarse dentro de un MQLONG (el lenguaje de programación más restrictivo es COBOL, para el cual el rango válido es de -999 999 999 a +999 999 999).

Considere utilizar *AILUO* para lenguajes de programación que no admiten el tipo de datos puntero, o que implementan el tipo de datos puntero de una manera que podría no ser portátil a diferentes entornos (por ejemplo, el lenguaje de programación COBOL).

Independientemente de la técnica elegida, utilice solo una de *AILUP* y *AILUO*; la llamada fallará con el código de motivo RC2388.

Este es un campo de entrada. El valor inicial de este campo es el puntero nulo en aquellos lenguajes de programación que admiten punteros y, en los demás casos, una cadena de bytes totalmente nulos.

Nota: En plataformas donde el lenguaje de programación no admite el tipo de datos puntero, este campo se declara como una cadena de bytes de la longitud adecuada.

AISID (entero con signo de 10 dígitos)

El valor debe ser:

AISIDV

Identificador del registro de información de autenticación.

Este campo siempre es un campo de entrada. El valor inicial de este campo es AISIDV.

AIVER (entero con signo de 10 dígitos)

El valor debe ser:

AIVER1

Version-1 registro de información de autenticación.

La siguiente constante especifica el número de versión de la versión actual:

AIVERC

Versión actual del registro de información de autenticación.

Este campo siempre es un campo de entrada. El valor inicial de este campo es AIVER1.

Valores iniciales

Tabla 721. Campos en MQAIR para MQAIR		
Nombre de campo	Nombre de la constante	Valor de la constante
<i>AISID</i>	AISIDV	'AIR-'
<i>AIVER</i>	AIVERC	1
<i>AITYP</i>	AITLDP	1
<i>AICN</i>	Ninguna	Cadena nula o espacios en blanco
<i>AILUP</i>	Ninguna	Puntero nulo o bytes nulos
<i>AILUO</i>	Ninguna	0
<i>AILUL</i>	Ninguna	0

Tabla 721. Campos en MQAIR para MQAIR (continuación)		
Nombre de campo	Nombre de la constante	Valor de la constante
AIPW	Ninguna	Cadena nula o espacios en blanco

Notas:

1. El símbolo ~ representa un único carácter en blanco.

Declaración RPG

```

D*.1.....2.....3.....4.....5.....6.....7..
D* MQAIR Structure
D*
D* Structure identifier
D AISID          1          4      INZ('AIR ')
D* Structure version number
D AIVER          5          8I 0  INZ(1)
D* Type of authentication information
D AITYP          9         12I 0  INZ(1)
D* Connection name of CRL LDAP server
D AICN          13         276    INZ
D* Address of LDAP user name
D AILUP         277        292*   INZ(*NULL)
D* Offset of LDAP user name from start of MQAIR structure
D AILUO         293        296I 0  INZ(0)
D* Length of LDAP user name
D AILUL         297        300I 0  INZ(0)
D* Password to access LDAP server
D AIPW          301        332    INZ

```

MQBMHO (Opciones de búfer para el controlador de mensajes) activado IBM i

Estructura que define las opciones del búfer al controlador de mensajes.

Visión general

Propósito : La estructura MQBMHO permite a las aplicaciones especificar opciones que controlan cómo se producen los identificadores de mensajes a partir de los búferes. La estructura es un parámetro de entrada en la llamada MQBUFMH.

Conjunto de caracteres y codificación : Los datos en MQBMHO deben estar en el conjunto de caracteres de la aplicación y la codificación de la aplicación (ENNAT).

- “Campos” en la página 1035
- “Valores iniciales” en la página 1036
- “Declaración RPG” en la página 1036

Campos

La estructura MQBMHO contiene los siguientes campos, que se describen en **orden alfabético** :

BMSID (entero con signo de 10 dígitos)

Estructura del búfer al identificador de mensaje: campo « StrucId ».

Este es el identificador de la estructura. El valor debe ser:

BMSIDV

Identificador para la estructura del búfer al controlador de mensajes.

Este campo siempre es un campo de entrada. El valor inicial de este campo es BMSIDV.

BMVER (entero con signo de 10 dígitos)

Estructura del búfer al identificador de mensaje: campo Versión.

Este es el número de versión de la estructura. El valor debe ser:

BMVER1

Número de versión para la estructura del búfer al controlador de mensajes.

La siguiente constante especifica el número de versión de la versión actual:

BMVERVC

Versión actual de la estructura del búfer al identificador de mensaje.

Este campo siempre es un campo de entrada. El valor inicial de este campo es BMVER1.

BMOPT (entero con signo de 10 dígitos)

Estructura del búfer al identificador de mensaje: campo Opciones.

El valor puede ser:

BMDLPR

Las propiedades que se añaden al identificador del mensaje se eliminan del búfer. Si la llamada falla, no se eliminan ninguna propiedad.

Opciones predeterminadas: Si no necesita la opción descrita, utilice la siguiente opción:

BMNONE

No se ha especificado ninguna opción.

Este campo siempre es un campo de entrada. El valor inicial de este campo es BMDLPR.

Valores iniciales

Tabla 722. Campos en MQBMHO		
Nombre de campo	Nombre de la constante	Valor de la constante
BMSID	BMSIDV	'BMHO'
BMVER	BMVER1	1
BMOPT	BMNONE	0

Declaración RPG

```
D* MQBMHO Structure
D*
D*
D* Structure identifier
D  BMSID              1      4    INZ('BMHO')
D*
D* Structure version number
D  BMVER              5      8I 0 INZ(1)
D*
D* Options that control the action of MQBUFMH
D  BMOPT              9     12I 0 INZ(1)
```

IBM i MQBO (Opciones de inicio) activado IBM i

La estructura MQBO permite a la aplicación especificar opciones relacionadas con la creación de una unidad de trabajo.

Visión general

Propósito : La estructura es un parámetro de entrada/salida en la llamada MQBEGIN.

Conjunto de caracteres y codificación : Los datos en MQBO deben estar en el conjunto de caracteres especificado por el atributo del **CodedCharSetId** gestor de colas y la codificación del gestor de colas local especificada por ENNAT.

- “Campos” en la página 1037
- “Valores iniciales” en la página 1037
- “Declaración RPG” en la página 1038

Campos

La estructura MQBO contiene los siguientes campos, que se describen en **orden alfabético** :

BOOPT (entero con signo de 10 dígitos)

Opciones que controlan la acción de MQBEGIN.

El valor debe ser:

BONONE

No se ha especificado ninguna opción.

Este campo siempre es un campo de entrada. El valor inicial de este campo es BONONE.

BOSID (cadena de caracteres de 4 bytes)

Identificador de estructura.

El valor debe ser:

BOSIDV

Identificador para la estructura begin-options.

Este campo siempre es un campo de entrada. El valor inicial de este campo es BOSIDV.

BOVER (entero con signo de 10 dígitos)

Número de versión de la estructura.

El valor debe ser:

BOVER1

Número de versión para la estructura begin-options.

La siguiente constante especifica el número de versión de la versión actual:

BOVERC

Versión actual de la estructura begin-options.

Este campo siempre es un campo de entrada. El valor inicial de este campo es BOVER1.

Valores iniciales

Tabla 723. Campos en MQBO		
Nombre de campo	Nombre de la constante	Valor de la constante
BOSID	BOSIDV	'B0¬¬'
BOVER	BOVER1	1
BOOPT	BONONE	0

Notas:

1. El símbolo ¬ representa un único carácter en blanco.

Declaración RPG

```
D*..1.....2.....3.....4.....5.....6.....7..  
D* MQBO Structure  
D*  
D* Structure identifier  
D  BOSID          1      4  INZ('BO ')  
D* Structure version number  
D  BOVER          5      8I 0 INZ(1)  
D* Options that control the action of MQBEGIN  
D  BOOPT          9     12I 0 INZ(0)
```

IBM i MQCBC (Contexto de devolución de llamada) activado IBM i

Estructura que describe la rutina de devolución de llamada.

Visión general

Finalidad

La estructura MQCBC se utiliza para especificar la información de contexto que se pasa a una función de devolución de llamada.

La estructura es un parámetro de entrada/salida en la llamada a una rutina consumidora de mensajes.

Versión

La versión actual de MQCBC es CBCV2.

Conjunto de caracteres y codificación

Los datos en MQCBC están en el conjunto de caracteres dado por el atributo del **CodedCharSetId** gestor de colas y la codificación del gestor de colas local dada por ENNAT. Sin embargo, si la aplicación se ejecuta como un IBM MQ cliente, la estructura se encuentra en el juego de caracteres y la codificación del cliente.

- “Campos” en la [página 1038](#)
- “Valores iniciales” en la [página 1044](#)
- “Declaración RPG” en la [página 1044](#)

Campos

La estructura MQCBC contiene los siguientes campos, que se describen en orden alfabético:

CBCBUFFLEN (entero con signo de 10 dígitos)

El búfer puede ser mayor que el valor « MaxMsgLength » definido para el consumidor y que el valor « ReturnedLength » en el MQGMO.

Estructura del contexto de devolución de llamada: campo « BufferLength ».

Esta es la longitud en bytes del búfer de mensajes que se ha pasado a esta función.

La longitud real del mensaje se indica en [DataLength](#) el campo.

La aplicación puede utilizar todo el búfer para sus propios fines durante la duración de la función de devolución de llamada.

Este es un campo de entrada para la función de consumo de mensajes; no es relevante para una función de controlador de excepciones.

CBCCALLBA (entero con signo de 10 dígitos)

Estructura del contexto de devolución de llamada: campo « CallbackArea ».

Este es un campo que está disponible para que lo utilice la función de devolución de llamada.

El gestor de colas no toma ninguna decisión basada en el contenido de este campo, que se transfiere sin cambios desde el campo [CBDCALLBA](#) de la estructura MQCBD, que es un parámetro de la llamada MQCB utilizado para definir la función de devolución de llamada.

Los cambios en el *CBCCALLBA* se conservan en todas las invocaciones de la función de devolución de llamada para un *CBCHOBJ*. Este campo no se comparte con funciones de devolución de llamada para otros identificadores.

Este es un campo de entrada/salida para la función de devolución de llamada. El valor inicial de este campo es un puntero nulo o bytes nulos.

CBCCALLT (entero con signo de 10 dígitos)

Estructura del contexto de devolución de llamada: campo « CallType ».

Campo que contiene información sobre por qué se ha llamado a esta función. Se definen los siguientes tipos de llamadas.

Tipos de llamadas de entrega de mensajes: Estos tipos de llamadas contienen información sobre un mensaje. Los parámetros **CBCLEN CBCBUFFLEN** y son válidos para estos tipos de llamadas.

CBCTMR

La función de consumo de mensajes se ha invocado con un mensaje que se ha eliminado de forma destructiva del identificador del objeto.

Si el valor de *CBCCC* es *CCWARN*, el valor del *Reason* campo es *RC2079* o uno de los códigos que indican un problema de conversión de datos.

CBCTMN

La función de consumo de mensajes se ha invocado con un mensaje que aún no se ha eliminado de forma destructiva del identificador del objeto. El mensaje se puede eliminar de forma destructiva del identificador del objeto utilizando el método *MsgToken*.

Es posible que el mensaje no se haya eliminado porque:

- Las opciones *MQGMO* solicitaron una operación de exploración, *GMBR**
- El mensaje es más grande que el búfer disponible y las opciones *MQGMO* no lo especifican.
gmattm

Si el valor de *CBCCC* es *CCWARN*, el valor del *Reason* campo es *RC2080* o uno de los códigos que indican un problema de conversión de datos.

Tipos de llamadas de control de devolución de llamada: Estos tipos de llamadas contienen información sobre el control de la devolución de llamada y no contienen detalles sobre un mensaje. Estos tipos de llamadas se solicitan utilizando *CBDOPT* en la estructura *MQCBD*.

Los parámetros **CBCLEN CBCBUFFLEN** y no son válidos para estos tipos de llamadas.

CBCTRC

El propósito de este tipo de llamada es permitir que la función de devolución de llamada realice alguna configuración inicial.

La función de devolución de llamada se invoca inmediatamente después de registrarse la devolución de llamada, es decir, al regresar de una llamada *MQCB* utilizando un valor para el *Operation* campo de *CBREG*.

Este tipo de llamada se utiliza tanto para consumidores de mensajes como para controladores de eventos.

Si se solicita, esta es la primera invocación de la función de devolución de llamada.

El valor del campo *CBCREA* es *RCNONE*.

CBCTSC

El propósito de este tipo de llamada es permitir que la función de devolución de llamada realice alguna configuración cuando se inicia, por ejemplo, restablecer los recursos que se limpiaron cuando se detuvo anteriormente.

La función de devolución de llamada se invoca cuando se inicia la conexión utilizando *CTLSR* o *CTLSW*.

Si una función de devolución de llamada se registra dentro de otra función de devolución de llamada, este tipo de llamada se invoca cuando la devolución de llamada regresa.

Este tipo de llamada se utiliza únicamente para los consumidores de mensajes.

El valor del campo *CBCREA* es RCNONE.

CBCTTC

El objetivo de este tipo de llamada es permitir que la función de devolución de llamada realice algunas tareas de limpieza cuando se detiene durante un tiempo, por ejemplo, limpiar los recursos adicionales que se han adquirido durante el consumo de mensajes.

La función de devolución de llamada se invoca cuando se emite una llamada MQCTL utilizando un valor para el *Operation* campo de CTLSP.

Este tipo de llamada se utiliza únicamente para los consumidores de mensajes.

El valor del campo *CBCREA* se establece para indicar el motivo de la parada.

CBCTDC

El propósito de este tipo de llamada es permitir que la función de devolución de llamada realice la limpieza final al final del proceso de consumo. La función de devolución de llamada se invoca cuando:

- La función de devolución de llamada se da de baja mediante una llamada MQCB con BCUNR.
- La cola está cerrada, lo que provoca una baja implícita. En este caso, la función de devolución de llamada se pasa a HOUNUH como el identificador del objeto.
- La llamada MQDISC finaliza, lo que provoca un cierre implícito y, por lo tanto, una baja del registro. En este caso, la conexión no se desconecta inmediatamente y cualquier transacción en curso aún no se ha confirmado.

Si alguna de estas acciones se realiza dentro de la propia función de devolución de llamada, la acción se invoca una vez que la devolución de llamada regresa.

Este tipo de llamada se utiliza tanto para consumidores de mensajes como para controladores de eventos.

Si se solicita, esta es la última invocación de la función de devolución de llamada.

El valor del campo *CBCREA* se establece para indicar el motivo de la parada.

CBCTEC

Función del controlador de eventos

La función del controlador de eventos se ha invocado sin un mensaje cuando:

- Se emite una llamada MQCTL con un valor para el *Operation* campo de CTLSP, o
- El gestor de colas o la conexión se detiene o se suspende.

Esta llamada se puede utilizar para tomar las medidas adecuadas para todas las funciones de devolución de llamada.

Función de consumidor de mensajes

La función de consumo de mensajes se ha invocado sin un mensaje cuando se ha detectado un error (*CBCCC* = CCFAIL) específico del identificador del objeto; por ejemplo, *CBCREA* código = RC2016 .

El valor del *CBCREA* campo se establece para indicar el motivo de la llamada.

Este es un campo de entrada. CBCTMR y CMCTMN solo son aplicables a las funciones de consumo de mensajes.

CBCCC (entero con signo de 10 dígitos)

Estructura del contexto de devolución de llamada: campo « CompCode ».

Este es el código de finalización. Indica si hubo algún problema al consumir el mensaje; puede ser uno de los siguientes:

CCOK

Realización satisfactoria

CCWARN

Aviso (terminación parcial)

CCFAIL

Llamada fallida

Este es un campo de entrada. El valor inicial de este campo es CCOK.

CBCCONNAREA (entero con signo de 10 dígitos)

Estructura del contexto de devolución de llamada: campo « ConnectionArea ».

Este es un campo que está disponible para que lo utilice la función de devolución de llamada.

El gestor de colas no toma ninguna decisión basada en el contenido de este campo y se transfiere sin cambios desde el ConnectionArea campo de la estructura MQCTLO, que es un parámetro de la llamada MQCTL utilizada para controlar la función de devolución de llamada.

Cualquier cambio realizado en este campo por las funciones de devolución de llamada se conserva en todas las invocaciones de la función de devolución de llamada. Esta área se puede utilizar para pasar información que deben compartir todas las funciones de devolución de llamada. A diferencia de *CallbackArea*, esta área es común a todas las devoluciones de llamada para un identificador de conexión.

Este es un campo de entrada y salida. El valor inicial de este campo es un puntero nulo o bytes nulos.

CBCLEN (entero con signo de 10 dígitos)

Es la longitud en bytes de los datos de la aplicación en el mensaje. Si el valor es cero, significa que el mensaje no contiene datos de aplicación.

El campo CBCLEN contiene la longitud del mensaje, pero no necesariamente la longitud de los datos del mensaje transmitidos al consumidor. Es posible que el mensaje se haya truncado. Utilice el campo GMRL en MQGMO para determinar cuántos datos se han transferido al consumidor.

Si el código de motivo indica que el mensaje ha sido truncado, puede utilizar el campo CBCLEN para determinar el tamaño real del mensaje. Esto le permite determinar el tamaño del búfer necesario para alojar los datos del mensaje y, a continuación, emitir una llamada MQCB para actualizar el CBDMML en el MQCBD con un valor adecuado.

Si se especifica la opción GMCONV, el mensaje convertido podría ser mayor que el valor devuelto por DataLength. En tales casos, es probable que la aplicación deba emitir una llamada MQCB para actualizar el CBDMML en el MQCBD a un valor superior al devuelto por el gestor de colas para DataLength.

Para evitar problemas de truncamiento de mensajes, especifique MaxMsgLength como CBDFM. Esto hace que el gestor de colas asigne un búfer para la longitud completa del mensaje después de la conversión de datos. Sin embargo, tenga en cuenta que, aunque se especifique esta opción, es posible que no haya suficiente espacio de almacenamiento disponible para procesar correctamente la solicitud. Las aplicaciones siempre deben comprobar el código de motivo devuelto. Por ejemplo, si no es posible asignar suficiente almacenamiento para convertir el mensaje, este se devuelve a la aplicación sin convertir.

Este es un campo de entrada para la función de consumo de mensajes; no es relevante para una función de controlador de eventos.

CBCFLG (entero con signo de 10 dígitos)

Banderas que contienen información sobre este consumidor.

Se define la siguiente opción:

CBCFBE

Esta bandera se puede devolver si una llamada MQCLOSE anterior que utilizaba la opción COQSC falló con un código de motivo « RC2458 » (no se ha encontrado el archivo).

Este código indica que se está devolviendo el último mensaje leído por adelantado y que el búfer ahora está vacío. Si la aplicación emite otra llamada MQCLOSE utilizando la opción COQSC, se ejecuta correctamente.

Tenga en cuenta que no se garantiza que una aplicación reciba un mensaje con esta bandera activada, ya que aún podría haber mensajes en el búfer de lectura anticipada que no coincidan con los criterios de selección actuales. En este caso, la función de consumidor se invoca con el código de motivo RC2019 .

Si el búfer de lectura anticipada está vacío, se invoca al consumidor con el indicador CBCFBE y el código de motivo RC2518.

Este es un campo de entrada para la función de consumo de mensajes; no es relevante para una función de controlador de eventos.

CBCHOBJ (entero con signo de 10 dígitos)

Estructura del contexto de devolución de llamada: campo CBCHOBJ.

Para una llamada a un consumidor de mensajes, este es el identificador del objeto relacionado con el consumidor de mensajes.

Para un controlador de eventos, este valor es HONONE

La aplicación puede utilizar este identificador y el token del mensaje en el bloque Obtener opciones de mensaje para obtener el mensaje si este no se ha eliminado de la cola.

Este campo siempre es un campo de entrada. El valor inicial de este campo es HOUNUH

CBCRCD (entero con signo de 10 dígitos)

CBCRCD Indica cuánto tiempo espera el gestor de colas antes de intentar volver a conectarse. El campo puede ser modificado por un controlador de eventos para cambiar el retraso o detener la reconexión por completo.

Utilice el **CBCRCD** campo solo si el valor del **Reason** campo en el contexto de devolución de llamada es RC2545.

Al entrar en el controlador de eventos, el valor de **CBCRCD** es el número de milisegundos que el gestor de colas va a esperar antes de intentar volver a conectarse. [Tabla 724 en la página 1042](#) Enumera los valores que se pueden establecer para modificar el comportamiento del gestor de colas al regresar del controlador de eventos.

Tabla 724. CBCRCD valores	
Valor	Descripción
-1	No vuelva a intentar reconectarse. Se devuelve un error a la aplicación.
0	Intenta volver a conectarte inmediatamente.
>0	Espera estos milisegundos antes de volver a intentar la conexión.

CBCREA (entero con signo de 10 dígitos)

Estructura del contexto de devolución de llamada: campo «Motivo».

Este es el código de motivo que califica el **CBCCC**

Este es un campo de entrada. El valor inicial de este campo es RCNONE.

CBCSTATE (entero con signo de 10 dígitos)

Una indicación del estado actual del consumidor. Este campo es de gran valor para una aplicación cuando se pasa un código de motivo distinto de cero a la función del consumidor.

Puede utilizar este campo para simplificar la programación de aplicaciones, ya que no es necesario codificar el comportamiento para cada código de motivo.

Este es un campo de entrada. El valor inicial de este campo es CSNONE

Tabla 725. Valores de CBCSTATE y acciones resultantes		
Estado	Acción del gestor de colas	Valor de la constante
<i>CSNONE</i> Este código de motivo representa una llamada normal sin información adicional sobre el motivo	Ninguna; este es el funcionamiento normal.	0
<i>CSSUST</i> Estos códigos de motivo representan condiciones temporales.	La rutina de devolución de llamada se invoca para informar de la condición y, a continuación, se suspende. Después de un tiempo, el sistema podría intentar la operación de nuevo, lo que puede provocar que se vuelva a producir la misma situación.	1
<i>CSSUSU</i> Estos códigos de motivo representan situaciones en las que la devolución de llamada debe actuar para resolver la situación.	El consumidor se suspende y se llama a la rutina de devolución de llamada para informar de la condición. La rutina de devolución de llamada debe resolver la condición si es posible y, a continuación, REANUDAR o cerrar la conexión.	2
<i>CSSUS</i> Estos códigos de motivo representan fallos que impiden nuevas devoluciones de llamada de mensajes.	El gestor de colas suspende automáticamente la función de devolución de llamada. Si se reanuda la función de devolución de llamada, es probable que vuelva a recibir el mismo código de motivo.	3
<i>CSSTOP</i> Estos códigos de motivo representan el final del consumo de mensajes.	Entregado al controlador de excepciones y a las devoluciones de llamada que especificaron CBDBC. No se pueden consumir más mensajes.	4

CBCSID (entero con signo de 10 dígitos)

Estructura del contexto de devolución de llamada: campo « StrucId ».

Este es el identificador de estructura; el valor debe ser:

CBCSI

Identificador para la estructura del contexto de devolución de llamada.

Este campo siempre es un campo de entrada. El valor inicial de este campo es CBCSI.

CBCVER (entero con signo de 10 dígitos)

Estructura del contexto de devolución de llamada - Campo Versión.

Este es el número de versión de la estructura; el valor debe ser:

CBCV1

Version-1 Estructura del contexto de devolución de llamada.

La siguiente constante especifica el número de versión de la versión actual:

CBCCV

Versión actual de la estructura del contexto de devolución de llamada.

Este campo siempre es un campo de entrada. El valor inicial de este campo es CBCV1.

Valores iniciales

Tabla 726. Campos en MQCBC		
Nombre de campo	Nombre de la constante	Valor de la constante
CBCSID	CBCSI	'CBC↵'
CBCVER	CBCV1	1
CBCCALLT	Ninguna	0
CBCHOBJ	HOUNUH	-1
CBCCALLBA	Ninguna	Puntero nulo o bytes nulos
CBCCONNAREA	Ninguna	Puntero nulo o bytes nulos
CBCCC	CCOK	0
CBCREA	RCNONE	0
CBCSTATE	CSNONE	0
CBCLEN	Ninguna	0
CBCBUFFLEN	Ninguna	0
CBCFLG	Ninguna	0
CBCRCD	ninguno	0

Nota:

1. El símbolo ↵ representa un único carácter en blanco.

Declaración RPG

```
D* MQCBC Structure
D*
D*
D* Structure identifier
D  CBCSID              1      4      INZ('CBC ')
D*
D* Structure version number
D  CBCVER              5      8I 0 INZ(1)
D*
D* Why Function was called
D  CBCCALLT           9      12I 0 INZ(0)
D*
D* Object Handle
D  CBCHOBJ            13      16I 0 INZ(-1)
D*
D* Callback data passed to the function
D  CBCCALLBA          17      32*   INZ(*NULL)
D*
D* MQCTL Data area passed to the function
D  CBCCONNAREA        33      48*   INZ(*NULL)
D*
D* Completion Code
D  CBCCC              49      52I 0 INZ(0)
D*
D* Reason Code
D  CBCREA             53      56I 0 INZ(0)
D*
```



```

D* Consumer State
D CBCSTATE          57      60I 0 INZ(0)
D*
D* Message Data Length
D CBCLEN            61      64I 0 INZ(0)
D*
D* Buffer Length
D CBCBUFFLEN        65      68I 0 INZ(0)
D*
** Flags containing information about
D* this consumer
D CBCFLG             69      72I 0 INZ(0)
D* Ver:1 **
D* Number of milliseconds before reconnect attempt
D CBCRCD             73      76I 0 INZ(0)
D* Ver:2 **
D*

```

MQCBD (descriptor de devolución de llamada) activado IBM i

Estructura que especifica la función de devolución de llamada.

Visión general

Propósito : La estructura MQCBD se utiliza para especificar una función de devolución de llamada y las opciones que controlan su uso por parte del gestor de colas.

La estructura es un parámetro de entrada en la llamada MQCB.

Versión : La versión actual de MQCBD es CBDV1.

Conjunto de caracteres y codificación : Los datos en MQCBD deben estar en el conjunto de caracteres y la codificación del gestor de colas local; estos se proporcionan mediante el atributo del **CodedCharSetId** gestor de colas y ENNAT. Sin embargo, si la aplicación se ejecuta como un IBM MQ MQI client, la estructura debe estar en el juego de caracteres y en la codificación del cliente.

- [“Campos” en la página 1045](#)
- [“Valores iniciales” en la página 1049](#)
- [“Declaración RPG” en la página 1049](#)

Campos

La estructura MQCBD contiene los siguientes campos, que se describen en **orden alfabético** :

CBDSCALLBA (entero con signo de 10 dígitos)

Este es un campo que está disponible para que lo utilice la función de devolución de llamada.

El gestor de colas no toma ninguna decisión basada en el contenido de este campo y se transfiere sin cambios desde el campo CBCCALLBA de la estructura MQCBD, que es un parámetro de la declaración de la función de devolución de llamada.

El valor solo se utiliza en un *Operation* que tenga un valor CBREG, sin ninguna devolución de llamada definida actualmente, y no sustituye a una definición anterior.

Este es un campo de entrada y salida para la función de devolución de llamada. El valor inicial de este campo es un puntero nulo o bytes nulos.

CBDSCALLBF (entero con signo de 10 dígitos)

La función de devolución de llamada se invoca como una llamada a función.

Utilice este campo para especificar un puntero a la función de devolución de llamada.

Debe especificar *CallbackFunction* o *CallbackName*. Si se especifican ambos, se devuelve el código de motivo RC2486.

Si no se *CallbackFunction* establece ni *CallbackName* ni, la llamada falla con el código de motivo RC2486.

Esta opción no es compatible con los siguientes entornos:

- CICS en z/OS
- Lenguajes de programación y compiladores que no admiten referencias a punteros de función

En tales situaciones, la llamada falla con el código de motivo « RC2486 ».

Este es un campo de entrada. El valor inicial de este campo es un puntero nulo o bytes nulos.

CBDSCALLBN (entero con signo de 10 dígitos)

La función de devolución de llamada se invoca como un programa vinculado dinámicamente.

Debe especificar *CallbackFunction* o *CallbackName*. Si se especifican ambos, se devuelve el código de motivo RC2486.

Si cualquiera *CallbackName* de las dos condiciones *CallbackFunction* no se cumple, la llamada falla con el código de motivo « RC2486 ».

El módulo se carga cuando se registra la primera rutina de devolución de llamada que se va a utilizar y se descarga cuando se da de baja la última rutina de devolución de llamada que lo utiliza.

Salvo que se indique lo contrario en el texto siguiente, el nombre se alinea a la izquierda dentro del campo, sin espacios en blanco incrustados; el nombre en sí se rellena con espacios en blanco hasta alcanzar la longitud del campo. En las descripciones que siguen, los corchetes ([]) indican información opcional:

IBMi

El nombre de la devolución de llamada puede tener uno de los siguientes formatos:

- Biblioteca «/» Programa
- Biblioteca «/» ServiceProgram «("FunctionName")»

Por ejemplo, MyLibrary/MyProgram(MyFunction).

El nombre de la biblioteca puede ser *LIBL. Tanto el nombre de la biblioteca como el del programa tienen un límite máximo de 10 caracteres.

AIX and Linux

El nombre de la llamada de retorno es el nombre de un módulo o biblioteca que se puede cargar dinámicamente, con el sufijo del nombre de una función que reside en esa biblioteca. El nombre de la función debe ir entre paréntesis. El nombre de la biblioteca puede ir precedido opcionalmente de una ruta de directorio:

```
[path]library(function)
```

Si no se especifica la ruta, se utiliza la ruta de búsqueda del sistema.

El nombre está limitado a un máximo de 128 caracteres.

Windows

El nombre de la llamada de retorno es el nombre de una biblioteca de enlace dinámico, seguido del nombre de una función que reside en esa biblioteca. El nombre de la función debe ir entre paréntesis. El nombre de la biblioteca puede ir precedido opcionalmente por una ruta de directorio y una unidad:

```
[d:][path]library(function)
```

Si no se especifican la unidad y la ruta, se utiliza la ruta de búsqueda del sistema.

El nombre está limitado a un máximo de 128 caracteres.

z/OS

El nombre de la llamada de retorno es el nombre de un módulo de carga válido para la especificación en el parámetro EP de la macro LINK o LOAD.

El nombre está limitado a un máximo de 8 caracteres.

z/OS CICS

El nombre de la llamada de retorno es el nombre de un módulo de carga válido para la especificación en el parámetro PROGRAM de la macro de comando EXEC CICS LINK.

El nombre está limitado a un máximo de 8 caracteres.

El programa se puede definir como remoto utilizando la opción REMOTESYTEM de la definición del PROGRAMA instalado o mediante el programa de enrutamiento dinámico.

La región CICS remota debe estar conectada para IBM MQ que el programa pueda utilizar llamadas IBM MQ API. Sin embargo, tenga en cuenta que el campo CBCHOBJ de la estructura MQCBC no es válido en un sistema remoto.

Si se produce un error al intentar cargar *CallbackName*, se devuelve uno de los siguientes códigos de error a la aplicación:

- RC2495
- RC2496
- RC2497

También se escribe un mensaje en el registro de errores que contiene el nombre del módulo para el que se intentó la carga y el código de motivo del fallo del sistema operativo.

Este es un campo de entrada. El valor inicial de este campo es una cadena nula o espacios en blanco.

CBDCALLBT (entero con signo de 10 dígitos)

Este es el tipo de la función de devolución de llamada. El valor debe ser uno de los siguientes:

CBTMC

Define esta devolución de llamada como una función consumidora de mensajes.

La función de devolución de llamada del consumidor de mensajes se invoca cuando hay un mensaje disponible en un identificador de objeto que cumple los criterios de selección especificados y se inicia la conexión.

CBTEH

Define esta devolución de llamada como la rutina de eventos asíncronos; no está diseñada para consumir mensajes para un identificador.

Hobj no es necesario en la llamada MQCB que define el controlador de eventos y se ignora si se especifica.

El controlador de eventos se activa en condiciones que afectan a todo el entorno del consumidor de mensajes. La función de consumidor se invoca sin un mensaje cuando se produce un evento, por ejemplo, la detención o la inactividad de un gestor de colas o una conexión. No se requiere para condiciones específicas de un único consumidor de mensajes, por ejemplo, RC2016.

Los eventos se envían a la aplicación, independientemente de si la conexión está iniciada o detenida, excepto en los siguientes entornos:

- CICS sobre z/OS medio ambiente
- aplicaciones sin hilo

Si la persona que llama no pasa uno de estos valores, la llamada falla con un código de motivo de RC2483

Este campo siempre es un campo de entrada. El valor inicial de este campo es CBTMC.

CBDMML (entero con signo de 10 dígitos)

Es la longitud en bytes del mensaje más largo que se puede leer desde el identificador y pasar a la rutina de devolución de llamada. Si un mensaje tiene una longitud mayor, la rutina de devolución de llamada recibe *MaxMsgLength* bytes del mensaje y el código de motivo:

- RC2080 o
- RC2079 si especificó GMATM.

La longitud real del mensaje se indica en el [“CBCLEN \(entero con signo de 10 dígitos\)”](#) en la [página 1041](#) campo de la estructura MQCBC.

Se define el siguiente valor especial:

CBDFM

El sistema ajusta la longitud del búfer para devolver los mensajes sin truncar.

Si no hay suficiente memoria disponible para asignar un búfer para recibir el mensaje, el sistema llama a la función de devolución de llamada con un código de motivo RC2071.

Si, por ejemplo, solicita la conversión de datos y no hay suficiente memoria disponible para convertir los datos del mensaje, el mensaje sin convertir se pasa a la función de devolución de llamada.

Este es un campo de entrada. El valor inicial del campo *MaxMsgLength* es CBDFM.

CBDOPT (entero con signo de 10 dígitos)

Estructura del descriptor de devolución de llamada - Campo Opciones.

Se puede especificar cualquiera de los siguientes elementos, o todos ellos. Para especificar más de una opción, suma los valores (sin añadir la misma constante más de una vez) o combínalos mediante la operación OR bit a bit (si el lenguaje de programación admite operaciones bit a bit). Las combinaciones que no son válidas se indican; cualquier otra combinación es válida.

CBDFAQ

La llamada MQCB falla si el gestor de colas se encuentra en estado de inactividad.

En z/OS, esta opción también fuerza el fallo de la llamada MQCB si la conexión (para una CICS o IMS aplicación) se encuentra en estado de inactividad.

Especifique GMFIQ en las opciones MQGMO pasadas en la llamada MQCB para que se envíe una notificación a los consumidores de mensajes cuando estén inactivos.

Opciones de control : Las siguientes opciones controlan si se llama a la función de devolución de llamada, sin un mensaje, cuando cambia el estado del consumidor:

CBDRC

La función de devolución de llamada se invoca con el tipo de llamada CBCTRC

.

CBDSC

La función de devolución de llamada se invoca con el tipo de llamada CBCTSC.

CBDTC

La función de devolución de llamada se invoca con el tipo de llamada CBCTTC.

CBDDC

La función de devolución de llamada se invoca con el tipo de llamada CBCTDC.

Consulte [“CBCCALLT \(entero con signo de 10 dígitos\)”](#) en la [página 1039](#) para obtener más detalles sobre estos tipos de llamadas.

Opción predeterminada : Si no necesita ninguna de las opciones descritas, utilice la siguiente opción:

CBDNO

Utilice este valor para indicar que no se han especificado otras opciones; todas las opciones toman sus valores predeterminados.

CBDNO se define para facilitar la documentación del programa; no está previsto que esta opción se utilice con ninguna otra, pero como su valor es cero, dicho uso no puede detectarse.

Este es un campo de entrada. El valor inicial del campo *Options* es CBDNO.

CBDSID (entero con signo de 10 dígitos)

Estructura del descriptor de devolución de llamada: campo « StrucId ».

Este es el identificador de estructura; el valor debe ser:

CBDSI

Identificador para la estructura del descriptor de devolución de llamada.

Este campo siempre es un campo de entrada. El valor inicial de este campo es CBDSI.

CBDVER (entero con signo de 10 dígitos)

Estructura del descriptor de devolución de llamada - Campo de versión.

Este es el número de versión de la estructura; el valor debe ser:

CBDV1

Version-1 Estructura del descriptor de devolución de llamada.

La siguiente constante especifica el número de versión de la versión actual:

CBDCV

Versión actual de la estructura del descriptor de devolución de llamada.

Este campo siempre es un campo de entrada. El valor inicial de este campo es CBDV1.

Valores iniciales

Tabla 727. Campos en MQCBD		
Nombre de campo	Nombre de la constante	Valor de la constante
<i>StrucId</i>	CBDSI	'CBD↵'
<i>Version</i>	CBDV1	1
<i>CallBackType</i>	CBTMC	1
<i>Options</i>	CBDNO	0
<i>CallbackArea</i>	Ninguna	Bytes nulos
<i>CallbackFunction</i>	Ninguna	Bytes nulos
<i>CallbackName</i>	Ninguna	Espacios en blanco
<i>MaxMsgLength</i>	CBD FM	-1

Nota:

1. El símbolo ↵ representa un único carácter en blanco.

Declaración RPG

```
D* MQCBD Structure
D*
D*
D* Structure identifier
D  CBDSID          1      4    INZ('CBD ')
D*
D* Structure version number
D  CBDVER          5      8I 0 INZ(1)
D*
D* Callback function type
D  CBDCALLBT       9     12I 0 INZ(1)
```

```

D*
** Options controlling message
D* consumption
D  CBDOPT          13      16I 0 INZ(0)
D*
D* User data passed to the function
D  CBDCALLBA       17      32*
D*
D* FP: Callback function pointer
D  CBDCALLBF       33      48*
D*
D* Callback name
D  CBDCALLBN       49      176   INZ('\0')
D*
D* Maximum message length
D  CBDMML          177     180I 0 INZ(-1)

```

MQCHARV (Serie de longitud variable) en IBM i

Utilice la estructura MQCHARV para describir una serie de longitud variable.

Visión general

Juego de caracteres y codificación: Los datos de MQCHARV deben estar en la codificación del gestor de colas local proporcionado por ENNAT y el juego de caracteres del campo VCHRC dentro de la estructura. Si la aplicación se ejecuta como un IBM MQ MQI client, la estructura debe estar en la codificación del cliente. Algunos juegos de caracteres tienen una representación que depende de la codificación. Si VCHRC es uno de estos juegos de caracteres, la codificación utilizada es la misma que la de los otros campos de MQCHARV. El juego de caracteres identificado por VSCCSID puede ser un juego de caracteres de doble byte (DBCS).

Uso: la estructura MQCHARV direcciona los datos que pueden no estar contiguos con la estructura que los contiene. Para abordar estos datos, se pueden utilizar los campos declarados con el tipo de datos de puntero.

- [“Campos” en la página 1050](#)
- [“Valores iniciales” en la página 1052](#)
- [“Declaración RPG” en la página 1052](#)
- [“Redefinición de CSAPL” en la página 1052](#)

Campos

La estructura MQCHARV contiene los campos siguientes; los campos se describen en **orden alfabético**:

VCHRC (entero con signo de 10 dígitos)

Es el identificador de juego de caracteres de la serie de longitud variable dirigida por el campo VCHRP o VCHRO.

El valor inicial de este campo es CSAPL. Lo define IBM MQ para indicar que el gestor de colas debe cambiarlo por el identificador de juego de caracteres verdadero del gestor de colas. Esto es del mismo modo que se comporta CSQM. Como resultado, el valor CSAPL nunca se asocia con una serie de longitud variable. El valor inicial de este campo se puede cambiar definiendo un valor diferente para la constante CSAPL para la unidad de compilación mediante los medios adecuados para el lenguaje de programación de la aplicación.

VCHRL (entero con signo de 10 dígitos)

Longitud en bytes de la serie de longitud variable a la que se dirige el campo VCHRP o VCHRO.

El valor inicial de este campo es 0. El valor debe ser mayor o igual que cero o el siguiente valor especial que se reconoce:

VSFLT

Si no se especifica VSFLT, los bytes VCHRL se incluyen como parte de la serie. Si hay caracteres nulos, no delimitan la serie.

Si se especifica VSFLT, la serie está delimitada por el primer nulo encontrado en la serie. El propio nulo no se incluye como parte de esa serie.

Nota: El carácter nulo utilizado para terminar una serie si se especifica VSFLT es un valor nulo del conjunto de códigos especificado por VCHRC.

Por ejemplo, en UTF-16 (CCSID 1200, 13488 y 17584), es la codificación Unicode de 2 bytes donde un valor nulo se representa mediante un número de 16 bits de todos los ceros. En UTF-16 es común encontrar bytes únicos establecidos en cero que forman parte de caracteres (por ejemplo, caracteres ASCII de 7 bits), pero las series sólo terminarán en nulo cuando se encuentren dos bytes 'cero' en un límite de bytes par. Es posible obtener dos 'cero' bytes en un límite impar cuando cada uno forma parte de caracteres válidos. Por ejemplo, x '01' x '00' x '00' x '30' representa dos caracteres Unicode válidos y no termina en nulo la serie.

VCHRO (entero con signo de 10 dígitos)

El desplazamiento en bytes de la serie de longitud variable desde el inicio de MQCHARV, o la estructura que la contiene.

Cuando la estructura MQCHARV está incorporada en otra estructura, este valor es el desplazamiento en bytes de la serie de longitud variable desde el inicio de la estructura que contiene esta estructura MQCHARV. Cuando la estructura MQCHARV no está incorporada dentro de otra estructura, por ejemplo, si se especifica como parámetro en una llamada de función, el desplazamiento es relativo al inicio de la estructura MQCHARV.

El desplazamiento puede ser positivo o negativo. Puede utilizar el campo VCHRP o VCHRO para especificar la serie de longitud variable, pero no ambos.

El valor inicial de este campo es 0.

VCHRP (puntero)

Es un puntero a la serie de longitud variable.

Puede utilizar el campo VCHRP o VCHRO para especificar la serie de longitud variable, pero no ambos.

El valor inicial de este campo es un puntero nulo o bytes nulos.

VCHRS (entero con signo de 10 dígitos)

El tamaño en bytes del almacenamiento intermedio direccionado por el campo VCHRP o VCHRO.

Cuando se utiliza la estructura MQCHARV como campo de salida en una llamada a función, este campo debe inicializarse con la longitud del almacenamiento intermedio proporcionado. Si el valor de VCHRL es mayor que VCHRS, sólo se devolverán los bytes de datos VCHRS al interlocutor en el almacenamiento intermedio.

El valor debe ser mayor o igual que cero o el siguiente valor especial que se reconoce:

VSUSL

Si se especifica VSUSL, la longitud del almacenamiento intermedio se obtiene del campo VCHRL de la estructura MQCHARV. Este valor especial no es adecuado cuando se utiliza la estructura como campo de salida y se proporciona un almacenamiento intermedio. Este es el valor inicial de este campo.

Valores iniciales

Tabla 728. Valores iniciales de MQCHARV para constantes		
Nombre de campo	Nombre de constante	Valor de constante
VCHRP	Ninguna	Puntero nulo o bytes nulos.
VCHRO	Ninguna	0
VCHRS	VSUSL	-1
VCHRL	Ninguna	0
VCHRC	CSAPL	-3

Declaración RPG

```
D*.1.....2.....3.....4.....5.....6.....7..
D* MQCHARV Structure
D*
D* Address of variable length string
D VCHRP          1      16*
D* Offset of variable length string
D VCHRO          17     20I 0
D* Size of buffer
D VCHRS          21     24I 0
D* Length of variable length string
D VCHRL          25     28I 0
D* CCSID of variable length string
D VCHRC          29     32I 0
```

Redefinición de CSAPL

A diferencia de los lenguajes de programación soportados en otras plataformas, RPG no tiene una forma de redefinir una constante definida, por lo que debe establecer cada VCHRC específicamente si desea utilizar un valor que no sea CSAPL.

MQCIH (CICS bridge encabezado) en IBM i

La estructura MQCIH describe la información que puede estar presente al inicio de un mensaje enviado al CICS bridge a través de IBM MQ for z/OS.

Visión general

Nombre del formato : FMCICS.

Versión : La versión actual de MQCIH es CIVER2. Los campos que solo existen en la versión más reciente de la estructura se identifican como tales en las descripciones que siguen.

El archivo COPY proporcionado contiene la versión más reciente de MQCIH, con el valor inicial del *CIVER* campo establecido en CIVER2.

Conjunto de caracteres y codificación : Se aplican condiciones especiales al conjunto de caracteres y la codificación utilizados para la estructura MQCIH y los datos de los mensajes de aplicación:

- Las aplicaciones que se conectan al gestor de colas propietario de CICS bridge la cola deben proporcionar una estructura MQCIH que esté en el juego de caracteres y la codificación del gestor de colas. Esto se debe a que, en este caso, no se realiza la conversión de datos de la estructura MQCIH.
- Las aplicaciones que se conectan a otros gestores de colas pueden proporcionar una estructura MQCIH en cualquiera de los conjuntos de caracteres y codificaciones compatibles; la conversión de MQCIH la realiza el agente del canal de mensajes receptor conectado al gestor de colas propietario de la CICS bridge cola.

Nota: Hay una excepción a esto. Si el gestor de colas propietario de la CICS bridge cola utiliza CICS para el encolado distribuido, el MQCIH debe estar en el juego de caracteres y la codificación del gestor de colas propietario de la CICS bridge cola.

- Los datos del mensaje de aplicación que siguen la estructura MQCIH deben estar en el mismo conjunto de caracteres y codificación que la estructura MQCIH. Los campos *CICSICIENC* y de la estructura MQCIH no se pueden utilizar para especificar el juego de caracteres y la codificación de los datos del mensaje de la aplicación.

El usuario debe proporcionar una salida de conversión de datos para convertir los datos del mensaje de la aplicación si estos no están en uno de los formatos integrados compatibles con el gestor de colas.

Uso : Si los valores requeridos por la aplicación son los mismos que los valores iniciales mostrados en Tabla 730 en la página 1062, y el puente está funcionando con AUTH=LOCAL o AUTH=IDENTIFY, la estructura MQCIH puede omitirse del mensaje. En todos los demás casos, la estructura debe estar presente.

El puente acepta una estructura MQCIH de tipo « version-1 » o « version-2 », pero para las transacciones 3270 se debe utilizar una estructura « version-2 ».

La aplicación debe garantizar que los campos documentados como campos "de solicitud" tengan valores adecuados en el mensaje enviado al puente; estos campos se introducen en el puente.

Los campos documentados como campos "de respuesta" son establecidos por el CICS bridge en el mensaje de respuesta que el puente envía a la aplicación. La información sobre errores se devuelve en los campos *CIRET*, *CIFNC*, *CICCCIAC*, *CIREA* y, pero no todos ellos se establecen en todos los casos. Tabla 729 en la página 1053 muestra qué campos están configurados para diferentes valores de *CIRET*.

Tabla 729. Contenido de los campos de información de error en la estructura MQCIH				
CIRET	CIFNC	CICC	CIREA	CIAC
CRC000	-	-	-	-
CRC003	-	-	FBC*	-
CRC002 CRC008	IBM MQ nombre de llamada	IBM MQ <i>CMPCOD</i>	IBM MQ <i>REASON</i>	-
CRC001 CRC006 CRC007 CRC009	CICS EIBFN	CICS EIBRESP	CICS EIBRESP2	-
CRC004 CRC005	-	-	-	CICS ABCODE

- “Campos” en la página 1053
- “Valores iniciales” en la página 1062
- “Declaración RPG” en la página 1063

Campos

La estructura MQCIH contiene los siguientes campos, descritos en **orden alfabético** :

CIAC (cadena de caracteres de 4 bytes)

Código de finalización anómala.

El valor devuelto en este campo solo es significativo si el *CIRET* campo tiene el valor « CRC005 » o « CRC004 ». Si es así, *CIAC* contiene el valor CICS ABCODE.

Este es un campo de respuesta. La longitud de este campo la proporciona LNABNC. El valor inicial de este campo es 4 caracteres en blanco.

Este es un indicador que especifica si los descriptores ADS deben enviarse en las solicitudes BMS SEND y RECEIVE. Los valores siguientes están definidos:

ADNONE

No envíe ni reciba descriptores ADS.

ADSEND

Enviar descriptor ADS.

ADRECV

Recibir descriptor ADS.

ADMSGF

Utilice el formato de mensaje para el descriptor ADS.

Esto hace que el descriptor ADS se envíe o reciba utilizando la forma larga del descriptor ADS. El formulario largo tiene campos alineados en límites de 4 bytes.

El **CIADS** campo debe configurarse de la siguiente manera:

- Si no se utilizan descriptores ADS, establezca el campo en **ADNONE**.
- Si se utilizan descriptores ADS y con el *mismo* CCSID en cada entorno, establezca el campo en la suma de **ADSEND** y **ADRECV**.
- Si se utilizan descriptores ADS, pero con *diferentes* CCSID en cada entorno, establezca el campo en la suma de **ADSEND**, **ADRECV** y **ADMSGF**.

Este es un campo de solicitud que se utiliza únicamente para transacciones 3270. El valor inicial de este campo es **ADNONE**.

CIADS (entero con signo de 10 dígitos)

Enviar/recibir descriptor ADS.

Este es un indicador que especifica si los descriptores ADS deben enviarse en las solicitudes BMS SEND y RECEIVE. Los valores siguientes están definidos:

ADNONE

No envíe ni reciba descriptores ADS.

ADSEND

Enviar descriptor ADS.

ADRECV

Recibir descriptor ADS.

ADMSGF

Utilice el formato de mensaje para el descriptor ADS.

Esto hace que el descriptor ADS se envíe o reciba utilizando la forma larga del descriptor ADS. El formulario largo tiene campos alineados en límites de 4 bytes.

El **CIADS** campo debe configurarse de la siguiente manera:

- Si no se utilizan descriptores ADS, establezca el campo en **ADNONE**.
- Si se utilizan descriptores ADS y con el *mismo* CCSID en cada entorno, establezca el campo en la suma de **ADSEND** y **ADRECV**.
- Si se utilizan descriptores ADS, pero con *diferentes* CCSID en cada entorno, establezca el campo en la suma de **ADSEND**, **ADRECV** y **ADMSGF**.

Este es un campo de solicitud que se utiliza únicamente para transacciones 3270. El valor inicial de este campo es **ADNONE**.

CIAI (cadena de caracteres de 4 bytes)

Clave AID.

Este es el valor inicial de la clave AID cuando se inicia la transacción. Es un valor de 1 byte, alineado a la izquierda.

Este es un campo de solicitud que se utiliza únicamente para transacciones 3270. La longitud de este campo viene dada por **LNATID**. El valor inicial de este campo es 4 espacios en blanco.

CIAUT (cadena de caracteres de 8 bytes)

Contraseña o pasaporte.

Se trata de una contraseña o un pase. Si la autenticación por identificador de usuario está activa para el CICS bridge, *CIAUT* se utiliza con el identificador de usuario en el contexto de identidad MQMD para autenticar al remitente del mensaje.

Este es un campo de solicitud. La longitud de este campo viene dada por LNAUTH. El valor inicial de este campo es 8 espacios en blanco.

CICC (entero con signo de 10 dígitos)

IBM MQ código de finalización o CICS EIBRESP.

El valor devuelto en este campo depende de *CIRET* ; véase [Tabla 729 en la página 1053](#).

Este es un campo de respuesta. El valor inicial de este campo es CCOK.

CICNC (cadena de caracteres de 4 bytes)

Código de transacción Abend.

Este es el código de error que se utiliza para terminar la transacción (normalmente una transacción conversacional que solicita más datos). De lo contrario, este campo se deja en blanco.

Este es un campo de solicitud que se utiliza únicamente para transacciones 3270. La longitud de este campo viene dada por LNCNCL. El valor inicial de este campo es 4 espacios en blanco.

CICP (entero con signo de 10 dígitos)

Posición del cursor.

Esta es la posición inicial del cursor cuando se inicia la transacción. Más adelante, para las transacciones conversacionales, la posición del cursor se encuentra en el vector RECEIVE.

Este es un campo de solicitud que se utiliza únicamente para transacciones 3270. El valor inicial de este campo es 0. Este campo no aparece si *CIVER* es menor que CIVER2.

CICSI (entero con signo de 10 dígitos)

Reservado.

Este es un campo reservado; su valor no es significativo. El valor inicial de este campo es 0.

CICT (entero con signo de 10 dígitos)

Si la tarea puede ser conversacional.

Este es un indicador que especifica si se debe permitir que la tarea emita solicitudes de más información o si debe interrumpirse. El valor debe ser uno de los siguientes:

CTYES

La tarea es conversacional.

CTNO

La tarea no es conversacional.

Este es un campo de solicitud que se utiliza únicamente para transacciones 3270. El valor inicial de este campo es CTNO.

CIENC (entero con signo de 10 dígitos)

Reservado.

Este es un campo reservado; su valor no es significativo. El valor inicial de este campo es 0.

CIEO (entero con signo de 10 dígitos)

Compensación del error en el mensaje.

Esta es la posición de los datos no válidos detectados por la salida del puente. Este campo proporciona la compensación desde el inicio del mensaje hasta la ubicación de los datos no válidos.

Este es un campo de respuesta que se utiliza únicamente para transacciones 3270. El valor inicial de este campo es 0. Este campo no aparece si *CIVER* es menor que *CIVER2*.

CIFAC (cadena de bits de 8 bytes)

Fichas de las instalaciones del puente.

Se trata de un token de puente de 8 bytes. El objetivo de un token de puente es permitir que varias transacciones en una pseudoconversación utilicen el mismo puente (terminal virtual 3270). En el primer mensaje, o único mensaje, de una pseudoconversación, se debe establecer un valor de *FCNONE*; esto indica CICS a que asigne una nueva función de puente para este mensaje. Se devuelve un token de instalación puente en los mensajes de respuesta cuando se especifica un valor *CIFKT* distinto de cero en el mensaje de entrada. Los mensajes de entrada posteriores pueden utilizar el mismo token de la función puente.

Se define el siguiente valor especial:

FCNONE

No se ha especificado ningún token BVT.

Este es un campo de solicitud y respuesta que se utiliza únicamente para transacciones 3270. La longitud de este campo viene dada por *LNFAAC*. El valor inicial de este campo es *FCNONE*.

CIFKT (entero con signo de 10 dígitos)

Hora de liberación de la instalación puente.

Este es el tiempo en segundos que se mantendrá la instalación del puente después de que haya finalizado la transacción del usuario. Para transacciones no conversacionales, el valor debe ser cero.

Este es un campo de solicitud que se utiliza únicamente para transacciones 3270. El valor inicial de este campo es 0.

CIFL (cadena de caracteres de 4 bytes)

Atributos emulados por terminal.

Este es el nombre de un terminal instalado que se utilizará como modelo para la instalación del puente. Un valor en blanco significa que *CIFL* se toma de la definición del perfil de transacción del puente, o se utiliza un valor predeterminado.

Este es un campo de solicitud que se utiliza únicamente para transacciones 3270. La longitud de este campo viene dada por *LNFAAC*. El valor inicial de este campo es 4 espacios en blanco.

CIFLG (entero con signo de 10 dígitos)

Distintivos.

El valor debe ser:

CIFNON

Sin banderas.

Este es un campo de solicitud. El valor inicial de este campo es *CIFNON*.

CIFMT (cadena de caracteres de 8 bytes)

IBM MQ nombre del formato de los datos que siguen a *MQCIH*.

Especifica el nombre del IBM MQ formato de los datos que siguen a la estructura *MQCIH*.

En la llamada *MQPUT* o *MQPUT1*, la aplicación debe establecer este campo con el valor adecuado para los datos. Las reglas para codificar este campo son las mismas que para el *MDFMT* campo en *MQMD*.

Este nombre de formato también se utiliza para el mensaje de respuesta, si el *CIRFM* campo tiene el valor *FMNONE*.

- Para las solicitudes *DPL*, *CIFMT* debe ser el nombre del formato de *COMMAREA*.
- Para 3270 solicitudes, *CIFMT* debe ser *CSQCBDCI*, y *CIRFM* debe ser *CSQCBDCO*.

Las salidas de conversión de datos para estos formatos deben instalarse en el gestor de colas donde se van a ejecutar.

Si el mensaje de solicitud da lugar a la generación de un mensaje de respuesta de error, el mensaje de respuesta de error tiene un nombre de formato FMSTR.

Este es un campo de solicitud. La longitud de este campo viene dada por LNFMT. El valor inicial de este campo es FMNONE.

CIFNC (cadena de caracteres de 4 bytes)

IBM MQ nombre de llamada o función CICS EIBFN.

El valor devuelto en este campo depende de *CIRET* ; véase [Tabla 729 en la página 1053](#). Los siguientes valores son posibles cuando *CIFNC* contiene un nombre IBM MQ de llamada:

CFCONN

Llamada MQCONN.

CFGET

Llamada MQGET.

CFINQ

Llamada MQINQ.

CFOPEN

Llamada MQOPEN.

CFPUT

Llamada MQPUT.

CFPUT1

MQPUT1 llamada.

CFNONE

No hay llamada.

Este es un campo de respuesta. La longitud de este campo viene dada por LNFUNC. El valor inicial de este campo es CFNONE.

CIGWI (entero con signo de 10 dígitos)

Intervalo de espera para la llamada MQGET emitida por la tarea puente.

Este campo solo es aplicable cuando *CIUOW* tiene el valor CUFRST. Permite a la aplicación emisora especificar el tiempo aproximado en milisegundos que las llamadas MQGET emitidas por el puente deben esperar para el segundo mensaje de solicitud y los mensajes posteriores para la unidad de trabajo iniciada por este mensaje. Esto anula el intervalo de espera predeterminado utilizado por el puente. Se pueden utilizar los siguientes valores especiales:

WIDFLT

Intervalo de espera predeterminado.

Esto hace que CICS bridge espere durante el periodo especificado cuando se inició el puente.

WIULIM

Intervalo de espera ilimitado.

Este es un campo de solicitud. El valor inicial de este campo es WIDFLT.

CIII (entero con signo de 10 dígitos)

Reservado.

Este es un campo reservado. El valor debe ser 0. Este campo no aparece si *CIVER* es menor que CIVER2.

CILEN (entero con signo de 10 dígitos)

Longitud de la estructura MQCIH.

El valor debe ser uno de los siguientes:

CILEN1

Longitud de la estructura del encabezado de CICS información de version-1.

CILEN2

Longitud de la estructura del encabezado de CICS información de version-2.

La siguiente constante especifica la longitud de la versión actual:

CILENC

Longitud de la versión actual de la estructura del CICS encabezado de información.

Este es un campo de solicitud. El valor inicial de este campo es CILEN2.

CILT (entero con signo de 10 dígitos)

Tipo de enlace.

Esto indica el tipo de objeto que el puente debe intentar vincular. El valor debe ser uno de los siguientes:

LTPROG

Programa DPL.

LTTRAN

Transacción 3270.

Este es un campo de solicitud. El valor inicial de este campo es LTPROG.

CINTI (cadena de caracteres de 4 bytes)

Siguiente transacción que adjuntar.

Este es el nombre de la siguiente transacción devuelta por la transacción del usuario (normalmente mediante EXEC CICS RETURN TRANSID). Si no hay ninguna transacción siguiente, este campo se deja en blanco.

Este es un campo de respuesta que se utiliza únicamente para transacciones 3270. La longitud de este campo viene determinada por LNTRID. El valor inicial de este campo es 4 espacios en blanco.

CIODL (entero con signo de 10 dígitos)

Longitud de los datos COMMAREA de salida.

Esta es la longitud de los datos de usuario que se devolverán al cliente en un mensaje de respuesta. Esta longitud incluye el nombre del programa de 8 bytes. La longitud de COMMAREA pasada al programa vinculado es el máximo de este campo y la longitud de los datos de usuario en el mensaje de solicitud, menos 8.

Nota: La longitud de los datos de usuario en un mensaje es la longitud del mensaje *sin incluir* la estructura MQCIH.

Si la longitud de los datos de usuario en el mensaje de solicitud es menor que *CIODL*, se utiliza la DATALENGTH opción del LINK comando, lo que permite enviar el LINK de manera eficiente a otra CICS región.

Se puede utilizar el siguiente valor especial:

OLINPT

La longitud de salida es igual a la longitud de entrada.

Este valor puede ser necesario incluso si no se solicita ninguna respuesta, con el fin de garantizar que el COMMAREA pasado al programa vinculado tenga un tamaño suficiente.

Este es un campo de solicitud que se utiliza únicamente para los programas DPL. El valor inicial de este campo OLINPT.

CIREA (entero con signo de 10 dígitos)

IBM MQ razón o código de retroalimentación, o CICS EIBRESP2.

El valor devuelto en este campo depende de *CIRET* ; véase [Tabla 729 en la página 1053](#).

Este es un campo de respuesta. El valor inicial de este campo es RCNONE.

CIRET (entero con signo de 10 dígitos)

Código de retorno del puente.

Este es el código de retorno de la CICS bridge que describe el resultado del procesamiento realizado por el puente. Los *CIAC* campos *CIFNC*, *CICC* *CIREA*, y pueden contener información adicional (véase [Tabla 729 en la página 1053](#)). El valor puede ser uno de los siguientes:

CRC000

(0, X'000') Sin errores.

CRC001

(1, X'001') La CICS instrucción EXEC detectó un error.

CRC002

(2, X'002') IBM MQ La llamada detectó un error.

CRC003

(3, X'003') CICS bridge detectó un error.

CRC004

(4, X'004') CICS bridge finalizó de forma anómala.

CRC005

(5, X'005') La aplicación finalizó de forma anómala.

CRC006

(6, X'006') Se ha producido un error de seguridad.

CRC007

(7, X'007') Programa no disponible.

CRC008

(8, X'008') Segundo mensaje o mensajes posteriores dentro de la unidad de trabajo actual no recibidos dentro del tiempo especificado.

CRC009

(9, X'009') Transacción no disponible.

Este es un campo de respuesta. El valor inicial de este campo es CRC000.

CIRFM (cadena de caracteres de 8 bytes)

IBM MQ nombre del formato del mensaje de respuesta.

Este es el nombre IBM MQ del formato del mensaje de respuesta que se enviará en respuesta al mensaje actual. Las reglas para codificar esto son las mismas que para el *MDFMT* campo en MQMD.

Este es un campo de solicitud que se utiliza únicamente para los programas DPL. La longitud de este campo viene dada por LNFMT. El valor inicial de este campo es FMNONE.

CIRSI (cadena de caracteres de 4 bytes)

Reservado.

Este es un campo reservado. El valor debe ser 4 espacios en blanco. La longitud de este campo viene dada por LNRSID.

CIRS1 (cadena de caracteres de 8 bytes)

Reservado.

Este es un campo reservado. El valor debe ser 8 espacios en blanco.

CIRS2 (cadena de caracteres de 8 bytes)

Reservado.

Este es un campo reservado. El valor debe ser 8 espacios en blanco.

CIRS3 (cadena de caracteres de 8 bytes)

Reservado.

Este es un campo reservado. El valor debe ser 8 espacios en blanco.

CIRS4 (entero con signo de 10 dígitos)

Reservado.

Este es un campo reservado. El valor debe ser 0. Este campo no aparece si *CIVER* es menor que *CIVER2*.

CIRTI (cadena de caracteres de 4 bytes)

Reservado.

Este es un campo reservado. El valor debe ser 4 espacios en blanco. La longitud de este campo viene determinada por *LNTRID*.

CISC (cadena de caracteres de 4 bytes)

Código de inicio de la transacción.

Este es un indicador que especifica si el puente emula una transacción de terminal o una transacción *START*. El valor debe ser uno de los siguientes:

SCSTRT

Inicio.

SCDATA

Datos iniciales.

SCTERM

Finalizar entrada.

SCNONE

Ninguna.

En la respuesta del puente, este campo se establece en el código de inicio adecuado para el siguiente ID de transacción contenido en el *CINTI* campo. Los siguientes códigos de inicio son posibles en la respuesta:

- SCSTRT
- SCDATA
- SCTERM

Este CICS Transaction Server 1.2 campo es solo un campo de solicitud; su valor en la respuesta no está definido.

Para CICS Transaction Server 1.3 y versiones posteriores, este es tanto un campo de solicitud como de respuesta.

Este campo solo se utiliza para transacciones 3270. La longitud de este campo viene dada por *LNSTCO*. El valor inicial de este campo es *SCNONE*.

CISID (cadena de caracteres de 4 bytes)

Identificador de estructura.

El valor debe ser:

CISIDV

Identificador de la estructura del CICS encabezado de información.

Este es un campo de solicitud. El valor inicial de este campo es *CISIDV*.

CITES (entero con signo de 10 dígitos)

Estado al final de la tarea.

Este campo muestra el estado de la transacción del usuario al finalizar la tarea. Se devuelve uno de los siguientes valores:

TENOSY

No sincronizado.

La transacción del usuario aún no se ha completado y no se ha sincronizado. El *MDMT* campo en MQMD es MTRQST en este caso.

TECMIT

Confirmar unidad de trabajo.

La transacción del usuario aún no se ha completado, pero se ha sincronizado la primera unidad de trabajo. El *MDMT* campo en MQMD es MTDGRM en este caso.

TEBACK

Retirar unidad de trabajo.

La transacción del usuario aún no se ha completado. La unidad de trabajo actual se retirará. El *MDMT* campo en MQMD es MTDGRM en este caso.

TEENDT

Finalizar tarea.

La transacción del usuario ha finalizado (o se ha cancelado). El *MDMT* campo en MQMD es MTRPLY en este caso.

Este es un campo de respuesta que se utiliza únicamente para transacciones 3270. El valor inicial de este campo es TENOSY.

CITI (cadena de caracteres de 4 bytes)

Transacción para adjuntar.

Si *CILT* tiene el valor LTTRAN, *CITI* es el identificador de la transacción del usuario que se va a ejecutar; en este caso, se debe especificar un valor que no esté en blanco.

Si *CILT* tiene el valor LTPROG, *CITI* es el código de transacción bajo el cual se ejecutarán todos los programas dentro de la unidad de trabajo. Si el valor especificado está en blanco, se utiliza el código de transacción CICS predeterminado del puente DPL (CKBP). Si el valor no está en blanco, debe haberse definido como CICS una TRANSACCIÓN local con un programa inicial de CSQCBP00. Este campo solo es aplicable cuando *CIUOW* tiene el valor CUFRST o CUONLY.

Este es un campo de solicitud. La longitud de este campo viene determinada por LNTRID. El valor inicial de este campo es 4 espacios en blanco.

CIUOW (entero con signo de 10 dígitos)

Control de la unidad de trabajo.

Esto controla el procesamiento de la unidad de trabajo realizado por el CICS bridge. Puede solicitar al puente que ejecute una sola transacción o uno o varios programas dentro de una unidad de trabajo. El campo indica si la unidad de CICS bridge trabajo debe iniciarse, realizar la función solicitada dentro de la unidad de trabajo actual o finalizar la unidad de trabajo confirmándola o cancelándola. Se admiten varias combinaciones para optimizar los flujos de transmisión de datos.

El valor debe ser uno de los siguientes:

CUONLY

Iniciar la unidad de trabajo, ejecutar la función y, a continuación, confirmar la unidad de trabajo (DPL y 3270).

CUCONT

Datos adicionales para la unidad de trabajo actual (solo 3270).

CUFRST

Iniciar unidad de trabajo y ejecutar función (solo DPL).

CUMIDL

Realizar funciones dentro de la unidad de trabajo actual (solo DPL).

CULAST

Realizar la función y, a continuación, confirmar la unidad de trabajo (solo DPL).

CUCMIT

Confirmar la unidad de trabajo (solo DPL).

CUBACK

Retirar la unidad de trabajo (solo DPL).

Este es un campo de solicitud. El valor inicial de este campo es CUONLY.

CIVER (entero con signo de 10 dígitos)

Número de versión de la estructura.

El valor debe ser uno de los siguientes:

CIVER1

Version-1 Estructura del CICS encabezado de información.

CIVER2

Version-2 Estructura del CICS encabezado de información.

Los campos que solo existen en la versión más reciente de la estructura se identifican como tales en las descripciones de los campos. La siguiente constante especifica el número de versión de la versión actual:

CIVERC

Versión actual de la estructura del CICS encabezado de información.

Este es un campo de solicitud. El valor inicial de este campo es CIVER2.

Valores iniciales

Tabla 730. Campos en MQCIH		
Nombre de campo	Nombre de la constante	Valor de la constante
<i>CISID</i>	CISIDV	'CIH~'
<i>CIVER</i>	CIVER2	2
<i>CILEN</i>	CILEN2	180
<i>CIENC</i>	Ninguna	0
<i>CICSI</i>	Ninguna	0
<i>CIFMT</i>	FMNONE	Espacios en blanco
<i>CIFLG</i>	CIFNON	0
<i>CIRET</i>	CRC000	0
<i>CICC</i>	CCOK	0
<i>CIREA</i>	RCNONE	0
<i>CIUOW</i>	CUONLY	273
<i>CIGWI</i>	WIDFLT	-2
<i>CILT</i>	LTPROG	1
<i>CIODL</i>	OLINPT	-1
<i>CIFKT</i>	Ninguna	0

Tabla 730. Campos en MQCIH (continuación)

Nombre de campo	Nombre de la constante	Valor de la constante
CIADS	ADNONE	0
CICT	CTNO	0
CITES	TENOSY	0
CIFAC	FCNONE	Nulos
CIFNC	CFNONE	Espacios en blanco
CIAC	Ninguna	Espacios en blanco
CIAUT	Ninguna	Espacios en blanco
CIRS1	Ninguna	Espacios en blanco
CIRFM	FMNONE	Espacios en blanco
CIRSI	Ninguna	Espacios en blanco
CIRTI	Ninguna	Espacios en blanco
CITI	Ninguna	Espacios en blanco
CIFL	Ninguna	Espacios en blanco
CIAI	Ninguna	Espacios en blanco
CISC	SCNONE	Espacios en blanco
CICNC	Ninguna	Espacios en blanco
CINTI	Ninguna	Espacios en blanco
CIRS2	Ninguna	Espacios en blanco
CIRS3	Ninguna	Espacios en blanco
CICP	Ninguna	0
CIEO	Ninguna	0
CIII	Ninguna	0
CIRS4	Ninguna	0

Notas:

1. El símbolo ~ representa un único carácter en blanco.

Declaración RPG

```

D*.1.....2.....3.....4.....5.....6.....7..
D* MQCIH Structure
D*
D* Structure identifier
D  CISID          1      4    INZ('CIH ')
D* Structure version number
D  CIVER          5      8I 0 INZ(2)
D* Length of MQCIH structure
D  CILEN          9     12I 0 INZ(180)
D* Reserved
D  CIENC         13     16I 0 INZ(0)
D* Reserved
D  CICS1         17     20I 0 INZ(0)
D* MQ format name of data that followsMQCIH
D  CIFMT         21     28    INZ('      ')
D* Flags

```

```

D CIFLG                29      32I 0 INZ(0)
D* Return code from bridge
D CIRET                33      36I 0 INZ(0)
D* MQ completion code or CICSEIBRESP
D CICC                 37      40I 0 INZ(0)
D* MQ reason or feedback code, or CICSEIBRESP2
D CIREA                41      44I 0 INZ(0)
D* Unit-of-work control
D CIUOW                45      48I 0 INZ(273)
D* Wait interval for MQGET call issuedby bridge task
D CIGWI                49      52I 0 INZ(-2)
D* Link type
D CILT                 53      56I 0 INZ(1)
D* Output COMMAREA data length
D CIODL                57      60I 0 INZ(-1)
D* Bridge facility release time
D CIFKT                61      64I 0 INZ(0)
D* Send/receive ADS descriptor
D CIADS                65      68I 0 INZ(0)
D* Whether task can beconversational
D CICT                 69      72I 0 INZ(0)
D* Status at end of task
D CITES                73      76I 0 INZ(0)
D* Bridge facility token
D CIFAC                77      84      INZ(X'0000000000000000-
D                          00')
D* MQ call name or CICS EIBFNfunction
D CIFNC                85      88      INZ('      ')
D* Abend code
D CIAC                 89      92      INZ
D* Password or passticket
D CIAUT                93     100      INZ
D* Reserved
D CIRS1                101     108      INZ
D* MQ format name of reply message
D CIRFM                109     116      INZ('      ')
D* Remote CICS system ID to use
D CIRSI                117     120      INZ
D* CICS RTRANSID to use
D CIRTI                121     124      INZ
D* Transaction to attach
D CITI                 125     128      INZ
D* Terminal emulated attributes
D CIFL                 129     132      INZ
D* AID key
D CIAI                 133     136      INZ
D* Transaction start code
D CISC                 137     140      INZ('      ')
D* Abend transaction code
D CICNC                141     144      INZ
D* Next transaction to attach
D CINTI                145     148      INZ
D* Reserved
D CIRS2                149     156      INZ
D* Reserved
D CIRS3                157     164      INZ
D* Cursor position
D CICP                 165     168I 0 INZ(0)
D* Offset of error in message
D CIEO                 169     172I 0 INZ(0)
D* Reserved
D CIII                 173     176I 0 INZ(0)
D* Reserved
D CIRS4                177     180I 0 INZ(0)
D*

```

MQCMHO (Crear opciones de controlador de mensajes) activado

IBM i

La **MQCMHO** estructura permite a las aplicaciones especificar opciones que controlan cómo se crean los identificadores de mensajes.

Visión general

Finalidad

La estructura es un parámetro de entrada en la **MQCRTMH** llamada.

Conjunto de caracteres y codificación

Los datos en **MQCMHO** deben estar en el juego de caracteres de la aplicación y la codificación de la aplicación (ENNAT).

- “Campos” en la página 1065
- “Valores iniciales” en la página 1066
- “Declaración RPG” en la página 1066

Campos

La estructura MQCMHO contiene los siguientes campos, que se describen en orden alfabético:

CMOPT (entero con signo de 10 dígitos)

Se puede especificar una de las siguientes opciones:

CMVAL

Cuando **MQSETMP** se llama para establecer una propiedad en este identificador de mensaje, se valida el nombre de la propiedad para garantizar que:

- no contiene caracteres no válidos.
- no comienza por «JMS» o «usr.JMS », salvo en los siguientes casos:
 - JMSCorrelationID
 - JMSReplyTo
 - JMSType
 - JMSXGroupID
 - JMSXGroupSeq

Estos nombres están reservados para JMS propiedades.

- no es una de las siguientes palabras clave, en cualquier combinación de mayúsculas o minúsculas:
 - «Y»
 - «ENTRE»
 - «ESCAPE»
 - «FALSO»
 - «IN»
 - «IS»
 - "like"
 - "not"
 - «NULL»
 - «O»
 - «VERDADERO»
- no comienza con «Cuerpo» o «Raíz» (excepto « Root.MQMD »).

Si la propiedad está definida como e MQ e («mq.*») y se reconoce el nombre, los campos descriptores de la propiedad se establecen en los valores correctos para la propiedad. Si no se reconoce la propiedad, el *Support* campo del descriptor de la propiedad se establece en **PDSUPO** (para obtener más información, consulte [PDSUP](#)).

CMDEFV

Esto especifica que se produce el nivel predeterminado de validación de los nombres de propiedades.

El nivel predeterminado de validación es equivalente al especificado por **CMVAL**.

En una versión futura, se podría definir una opción administrativa que cambie el nivel de validación que se aplicará cuando **CMDEFV** se defina.

Éste es el valor predeterminado.

CMNOVA

No se realiza ninguna validación del nombre de la propiedad. Véase la descripción de **CMVAL**.

Opción predeterminada: Si no se requiere ninguna de las opciones descritas anteriormente en esta sección, se puede utilizar la siguiente opción:

CMNONE

Todas las opciones adoptan sus valores predeterminados. Utilice este valor para indicar que no se han especificado otras opciones. **CMNONE** documentación del programa de ayuda; no está previsto que esta opción se utilice con ninguna otra, pero como su valor es cero, dicho uso no puede detectarse.

Este campo siempre es un campo de entrada. El valor inicial de este campo es **CMDEFV**.

CMSID (entero con signo de 10 dígitos)

Este es el identificador de estructura; el valor debe ser:

CMSIDV

Identificador para crear la estructura de opciones del identificador de mensajes.

Este campo siempre es un campo de entrada. El valor inicial de este campo es **CMSIDV**.

CMVER (entero con signo de 10 dígitos)

Este es el número de versión de la estructura; el valor debe ser:

CMVER1

Version-1 Crear estructura de opciones de control de mensajes.

La siguiente constante especifica el número de versión de la versión actual:

CMVERC

Versión actual de la estructura de opciones para crear un identificador de mensaje.

Este campo siempre es un campo de entrada. El valor inicial de este campo es **CMVER1**.

Valores iniciales

Tabla 731. Campos en MQCMHO		
Nombre de campo	Nombre de la constante	Valor de la constante
<i>CMSID</i>	CMSIDV	'CMHO'
<i>CMVER</i>	CMVER1	1
<i>CMOPT</i>	CMDEFV	0

Declaración RPG

```
D* MQCMHO Structure
D*
D*
D* Structure identifier
D  CMSID           1       4   INZ('CMHO')
D*
```

```

D* Structure version number
D  CNVER                5          8I 0 INZ(1)
D*
D* Options that control the action of MQCRTMH
D  CMOPT                9          12I 0 INZ(0)

```

MQCNO (Opciones de conexión) activado IBM i

La estructura MQCNO permite a la aplicación especificar opciones relacionadas con la conexión al gestor de colas local.

Visión general

Propósito : La estructura es un parámetro de entrada/salida en la llamada MQCONN.

Versión : La versión actual de MQCNO es CNVER6. Los campos que solo existen en las versiones más recientes de la estructura se identifican como tales en las descripciones que siguen.

El archivo COPY proporcionado contiene la versión más reciente de MQCNO compatible con el entorno, pero con el valor inicial del CNVER campo establecido en CNVER1. Para utilizar campos que no están presentes en la estructura version-1, la aplicación debe establecer el CNVER campo en el número de versión requerido.

Conjunto de caracteres y codificación : Los datos en MQCNO deben estar en el conjunto de caracteres dado por el atributo del **CodedCharSetId** gestor de colas y la codificación del gestor de colas local dada por ENNAT.

- [“Campos” en la página 1067](#)
- [“Valores iniciales” en la página 1073](#)
- [“Declaración RPG” en la página 1073](#)

Campos

La estructura MQCNO contiene los siguientes campos, que se describen en **orden alfabético** :

CCDTUL (entero con signo de 10 dígitos)

CCDTUL es la longitud de la cadena identificada por CCDTUP o CCDTUO que contiene un identificador de canal de conexión de cliente (URL) que identifica la ubicación de la tabla de canales de conexión de cliente que se utilizará para la conexión.

Utilice CCDTUL solo cuando la aplicación que emite la llamada MQCONN se esté ejecutando como un IBM MQ MQI client.

Esta es una alternativa programática a la configuración de las variables de entorno [MQCHLLIB](#) y [MQCHLTAB](#).

Si la aplicación no se está ejecutando como cliente, se ignora CCDTUL.

Este campo se ignora si CNVER es menor que CNVER6.

CCDTUO (entero con signo de 10 dígitos)

CCDTUO es la compensación en bytes, desde el inicio de la estructura MQCNO, hasta una cadena que contiene un identificador de canal de conexión (URL) que identifica la ubicación de la tabla de canales de conexión del cliente que se utilizará para la conexión. El desplazamiento puede ser positivo o negativo.

Utilice CCDTUL solo cuando la aplicación que emite la llamada MQCONN se esté ejecutando como un IBM MQ MQI client.

Importante: Solo puede utilizar uno de los siguientes: CCDTUP o CCDTUO. La llamada falla con el código de motivo RC2600 si ambos campos son distintos de cero.

Esta es una alternativa programática a la configuración de las variables de entorno MQCHLLIB y MQCHLTAB.

Si la aplicación no se está ejecutando como cliente, se ignora CCDTUO.

Este campo se ignora si CNVER es menor que CNVER6.

CCDTUP (puntero)

CCDTUP es un puntero opcional a una cadena que contiene un identificador de canal de conexión (URL) para identificar la ubicación de la tabla de canales de conexión del cliente que se utilizará para la conexión.

Utilice CCDTUP solo cuando la aplicación que emite la llamada MQCONN se esté ejecutando como un IBM MQ MQI client.

Importante: Solo puede utilizar uno de los siguientes: CCDTUP o CCDTUO. La llamada falla con el código de motivo RC2600 si ambos campos son distintos de cero.

Esta es una alternativa programática a la configuración de las variables de entorno MQCHLLIB y MQCHLTAB.

Si la aplicación no se está ejecutando como cliente, se ignora CCDTUP.

Este campo se ignora si CNVER es menor que CNVER6.

CNAN (cadena de caracteres de 28 bytes)

El nombre establecido por la aplicación para identificar la conexión con el gestor de colas. El valor inicial del campo es caracteres nulos.

Este campo se ignora si CNVER es menor que CNVER7.

CNCCO (entero con signo de 10 dígitos)

Es el desplazamiento en bytes de una estructura de definición de canal MQCD desde el inicio de la estructura MQCNO.

CNCCP (puntero)

Este es un puntero a una estructura de definición de canal MQCD.

CNCONID (cadena de caracteres de 24 bytes)

Identificador de conexión único. Este campo permite al gestor de colas identificar de forma fiable un proceso de aplicación asignándole un identificador único cuando se conecta por primera vez al gestor de colas.

Las aplicaciones utilizan el identificador de conexión con fines de correlación al realizar llamadas PUT y GET. El gestor de colas asigna un identificador a todas las conexiones, independientemente de cómo se haya establecido la conexión.

Es posible utilizar el identificador de conexión para forzar el final de una unidad de trabajo de larga duración. Para ello, especifique el identificador de conexión utilizando el comando PCF «Stop Connection» o el comando MQSC STOP CONN. Para obtener más información sobre el uso de estos comandos, consulte los enlaces relacionados.

El valor inicial del campo es 24 bytes nulos.

CNCT (cadena de bits de 128 bytes)

Esta es una etiqueta que el administrador de colas asocia con los recursos que se ven afectados por la aplicación durante esta conexión.

Etiqueta de conexión del gestor de colas.

Cada aplicación o instancia de aplicación debe utilizar un valor diferente para la etiqueta, de modo que el gestor de colas pueda serializar correctamente el acceso a los recursos afectados. Consulte las descripciones de las opciones CN*CT* para obtener más detalles. La etiqueta deja de ser válida cuando la aplicación finaliza o emite la llamada MQDISC.

Utilice el siguiente valor especial si no se requiere ninguna etiqueta:

CTNONE

No se ha especificado ninguna etiqueta de conexión.

El valor es un cero binario para la longitud del campo.

Este es un campo de entrada. La longitud de este campo viene determinada por LNCTAG. El valor inicial de este campo es CTNONE. Este campo se ignora si CNVER es menor que CNVER3.

Utilice el campo « ConnTag » (Administrador de colas) al conectarse a un gestor z/OS de colas.

CNNORES2 (cadena de caracteres de 4 bytes)

Un campo reservado para rellenar la estructura hasta un límite de 64 bits. El valor inicial del campo es un cero binario para la longitud del campo.

Este campo se ignora si CNVER es menor que CNVER7.

CNOPT (entero con signo de 10 dígitos)

Opciones que controlan la acción de MQCONN.

Opciones de enlace

Las opciones de encuadernación controlan el tipo de IBM MQ encuadernación que se utiliza; especifique solo una de estas opciones:

CNSBND

Encuadernación estándar.

La opción de enlace estándar hace que la aplicación y el agente gestor de colas local se ejecuten en unidades de ejecución separadas, normalmente en procesos separados. La disposición mantiene la integridad del gestor de colas; es decir, protege al gestor de colas de programas erróneos.

Utilice CNSBND en situaciones en las que la aplicación pueda no haber sido probada completamente o pueda ser poco fiable o poco segura. CNSBND es el valor predeterminado.

CNSBND se define para ayudar en la documentación del programa. No utilice esta opción con ninguna otra opción que controle el tipo de enlace utilizado; sin embargo, dado que su valor es cero, dicho uso no puede detectarse.

Esta opción es compatible con todos los entornos.

CNFBND

Enlace de ruta rápida.

La opción de enlace de ruta rápida hace que la aplicación y el agente gestor de colas local formen parte de la misma unidad de ejecución. La ruta rápida contrasta con el enlace estándar, en el que la aplicación y el agente gestor de colas local se ejecutan en unidades de ejecución separadas.

CNFBND se ignora si el gestor de colas no admite este tipo de enlace; el procesamiento continúa como si la opción no se hubiera especificado.

CNFBND puede resultar ventajoso en situaciones en las que múltiples procesos consumen más recursos que el total utilizado por la aplicación. Una aplicación que utiliza el enlace de ruta rápida se conoce como *aplicación* de confianza.

Tenga en cuenta los siguientes puntos importantes al decidir si utilizar el enlace de ruta rápida:

- **El uso de la opción CNFBND no impide que una aplicación altere o corrompa mensajes y otras áreas de datos pertenecientes al gestor de colas. Utilice esta opción solo en situaciones en las que haya evaluado completamente estas cuestiones.**

- La aplicación no debe utilizar señales asíncronas ni interrupciones de temporizador (como sigkill) con CNFBND. También existen restricciones en el uso de segmentos de memoria compartida.
- La aplicación no debe tener más de un subproceso conectado al gestor de colas en un momento dado.
- La aplicación debe utilizar la MQDISC llamada para desconectarse del gestor de colas.
- La aplicación debe finalizar antes de cerrar el gestor de colas con el endmqm comando.

Los siguientes puntos se aplican al uso de CNFBND en los entornos indicados:

- En IBM i, el trabajo debe ejecutarse con el perfil de usuario QMQM que pertenece al QMQMADM grupo. Además, el programa no debe terminar de forma anómala, ya que de lo contrario podrían producirse resultados impredecibles.

Para obtener más información sobre las implicaciones del uso de aplicaciones de confianza, consulte [Conexión a un gestor de colas mediante la llamada MQCONN](#) y [Restricciones para aplicaciones de confianza](#).

CNSHBD

Enlaces compartidos.

La opción de enlaces compartidos hace que la aplicación y el agente gestor de colas local se ejecuten en unidades de ejecución separadas, normalmente en procesos separados. La disposición mantiene la integridad del gestor de colas; es decir, protege al gestor de colas de programas erróneos. Sin embargo, algunos recursos se comparten entre la aplicación y el agente gestor de colas local. CNSHBD se ignora si el gestor de colas no admite este tipo de enlace. El proceso continuará como si no se hubiese especificado la opción.

CNIBND

Enlaces aislados.

La opción de enlaces aislados hace que la aplicación y el agente gestor de colas local se ejecuten en unidades de ejecución separadas, normalmente en procesos separados. La disposición mantiene la integridad del gestor de colas; es decir, protege al gestor de colas de programas erróneos. El proceso de solicitud y el agente gestor de colas local están aislados entre sí, ya que no comparten recursos. CNIBND se ignora si el gestor de colas no admite este tipo de enlace. El proceso continuará como si no se hubiese especificado la opción.

Opciones para compartir el control

Las siguientes opciones controlan el uso compartido de identificadores entre diferentes subprocesos (unidades de procesamiento paralelo) dentro del mismo proceso. Solo se puede especificar una de estas opciones.

CNHSN

No se comparten identificadores entre subprocesos.

La opción de no compartir manejadores entre subprocesos indica que los manejadores de conexión y objeto solo pueden ser utilizados por el subproceso que provocó la asignación del manejador; es decir, el subproceso que emitió la llamada MQCONN, MQCONNX MQOPEN , o. Los manejadores no pueden ser utilizados por otros subprocesos pertenecientes al mismo proceso.

CNHSB

Compartir manejadores de serie entre subprocesos, con bloqueo de llamadas.

La opción de uso compartido de identificadores de serie entre subprocesos, con bloqueo de llamadas, indica que los identificadores de conexión y de objeto asignados por un subproceso de un proceso pueden ser utilizados por otros subprocesos que pertenezcan al mismo proceso. Sin embargo, solo un subproceso a la vez puede utilizar un identificador concreto, es decir, solo se permite el uso secuencial de un identificador. Si un subproceso intenta utilizar un identificador que ya está siendo utilizado por otro subproceso, la llamada se bloquea (espera) hasta que el identificador esté disponible.

CNHSNB

Compartir un identificador de serie entre subprocesos, sin bloquear llamadas.

La opción de uso compartido del identificador de serie entre subprocesos, sin bloqueo de llamadas, es igual que la opción «con bloqueo», salvo que, si el identificador está siendo utilizado por otro subproceso, la llamada se completa inmediatamente con CCFAIL y RC2219 en lugar de bloquearse hasta que el identificador esté disponible.

Un subproceso puede tener cero o un identificador no compartido, además de cero o más identificadores compartidos:

- Cada MQCONN llamada que especifica CNHSN devuelve un nuevo MQCONNX identificador no compartido en la primera llamada y el mismo identificador no compartido en las llamadas posteriores (suponiendo que no haya ninguna llamada MQDISC intermedia). El código de motivo es « RC2002 » (Aprobado por el cliente) para la segunda llamada y las siguientes.
- Cada MQCONNX llamada que especifica CNHSB o CNHSNB devuelve un nuevo identificador compartido en cada llamada.

Los identificadores de objeto heredan las mismas propiedades de uso compartido que el identificador de conexión especificado en la MQOPEN llamada que creó el identificador de objeto. Además, las unidades de trabajo heredan las mismas propiedades de uso compartido que el identificador de conexión utilizado para iniciar la unidad de trabajo; si la unidad de trabajo se inicia en un subproceso utilizando un identificador compartido, la unidad de trabajo se puede actualizar en otro subproceso utilizando el mismo identificador.

Si no se especifica una opción para compartir identificadores, el valor predeterminado viene determinado por el entorno:

- En el entorno Microsoft del servidor de transacciones (MTS), el valor predeterminado es el mismo que el de CNHSB.
- En otros entornos, el valor predeterminado es el mismo que en CNHSN.

Opciones de reconexión

Las opciones de reconexión determinan si una conexión se puede volver a conectar. Solo las conexiones de los clientes se pueden volver a conectar.

CNRCDF

La opción de reconexión se establece en su valor predeterminado. Si no se establece ningún valor predeterminado, el valor de esta opción se resuelve como DISABLED. El valor de la opción se pasa al servidor y se puede consultar mediante **PCF** y **MQSC**.

CNRC

La aplicación se puede volver a conectar a cualquier gestor de colas coherente con el valor del parámetro MQCONNXQMNAME . Utilice la opción CNRC solo si no hay afinidad entre la aplicación cliente y el gestor de colas con el que estableció inicialmente una conexión. El valor de la opción se pasa al servidor y se puede consultar mediante **PCF** y **MQSC**.

CNRCDF

La aplicación no se puede volver a conectar. El valor de la opción no se transmite al servidor.

CNRCQM

La aplicación solo puede volver a conectarse al gestor de colas con el que se conectó originalmente. Utilice este valor si un cliente puede volver a conectarse, pero existe una afinidad entre la aplicación cliente y el gestor de colas con el que estableció originalmente una conexión. Elija este valor si desea que el cliente se reconecte automáticamente a la instancia en espera de un gestor de colas con elevada disponibilidad. El valor de la opción se pasa al servidor y se puede consultar mediante **PCF** y **MQSC**.

Utilice las opciones CNRC, CNRCDF y CNRCQM solo para conexiones de clientes. Si las opciones se utilizan para una conexión vinculante, MQCONNX falla con el código de finalización y el código MQCC_FAILED de motivo. MQRC_OPTIONS_ERROR

Opción predeterminada : Si no se requiere ninguna de las opciones descritas, se puede utilizar la siguiente opción:

CNNONE

No se especifica ninguna opción.

CNNONE está definido para ayudar en la documentación del programa. No está previsto que esta opción se utilice con ninguna otra opción CN*, pero dado que su valor es cero, dicho uso no puede detectarse.

CNSCO (entero con signo de 10 dígitos)

Es el desplazamiento en bytes de una estructura MQSCO desde el inicio de la estructura MQCNO.

Este campo se ignora si CNVER es menor que CNVER4.

CNSCP (puntero)

Esta es la dirección de una estructura MQSCO.

Este campo se ignora si CNVER es menor que CNVER4.

CNSECPO (entero con signo de 10 dígitos)

Compensación de los parámetros de seguridad. El desplazamiento de la estructura MQCSP utilizada para especificar un ID de usuario y una contraseña.

El valor puede ser positivo o negativo. El valor inicial de este campo es 0.

Este campo se ignora si CNVER es menor que CNVER5.

CNSECPP (puntero)

Puntero de parámetros de seguridad. Dirección de la estructura MQCSP utilizada para especificar un ID de usuario y una contraseña.

El valor inicial de este campo es un puntero nulo o bytes nulos.

Este campo se ignora si CNVER es menor que CNVER5.

CNSID (cadena de caracteres de 4 bytes)

El identificador de estructura para la estructura MQCNO.

El valor debe ser:

CNSIDV

Identificador para la estructura connect-options.

Este campo siempre es un campo de entrada. El valor inicial de este campo es CNSIDV.

CNVER (entero con signo de 10 dígitos)

El número de versión de la estructura MQCNO.

El valor debe ser:

CNVER6

Version-6 Estructura de opciones de conexión.

Esta versión es compatible con todos los entornos.

CNVER7

Version-7 Estructura de opciones de conexión.

Esta versión es compatible con todos los entornos.

La siguiente constante especifica el número de versión de la versión actual:

CNVERC

Versión actual de la estructura connect-options.

Este campo siempre es un campo de entrada. El valor inicial de este campo es CNVER7.

Valores iniciales

Tabla 732. Campos en MQCNO		
Nombre de campo	Nombre de la constante	Valor de la constante
CNSID	CNSIDV	'CNO↵
CNVER	CNVER5	1
CNOPT	CNNONE	0
CNCCO	Ninguna	0
CNCCP	Ninguna	Puntero nulo o bytes nulos
CNCT	CTNONE	Nulos
CNSCP	Ninguna	Puntero nulo o bytes nulos
CNSCO	Ninguna	0
CNCONID	Ninguna	Nulos
CNSECPO	Ninguna	0
CNSECPP	Ninguna	Puntero nulo o bytes nulos
CCDTUL	Ninguna	0
CCDTUO	Ninguna	0
CCDTUP	Ninguna	Puntero nulo o bytes nulos

Notas:

1. El símbolo ↵ representa un único carácter en blanco.

Declaración RPG

```

D*****
D**
D**          IBM MQ for IBM i
D**
D**  FILE NAME:      CMQCNOG
D**
D**  DESCRIPTION:    MQCNO Structure -- Connect Options
D**
D*****
D** <N_OCO_COPYRIGHT>
D** Licensed Materials - Property of IBM
D**
D** 5724-H72
D** (c) Copyright IBM Corp. 1993, 2026. All Rights Reserved.
D**
D** US Government Users Restricted Rights - Use, duplication or
D** disclosure restricted by GSA ADP Schedule Contract with
D** IBM Corp.
D** <NOC_COPYRIGHT>
D*****
D**
D**  FUNCTION:        This file declares the structure MQCNO,
D**                   which is used by the main MQI.
D**
D**  PROCESSOR:       RPG (ILE)
D**
D*****
D*
D*
D*****
D** <BEGIN_BUILDINFO>
D** Generated on:    08/02/16 13:50
D** Build Level:     L000000
D** Build Type:      Production

```

```

D** Pointer Size: 128 Bit **
D** Source File: **
D** CMQCN0G **
D** <END_BUILDINFO> **
D*****
D*
D*..1.....2.....3.....4.....5.....6.....7..
D*
D*
D* MQCNO Structure
D*
D* Structure identifier
D CNSID 1 4 INZ('CNO ')
D* Structure version number
D CNVER 5 8I 0 INZ(1)
D* Options that control the action of MQCONN
D CNOPT 9 12I 0 INZ(0)
D* Ver:1 **
D* Offset of MQCD structure for client connection
D CNCCO 13 16I 0 INZ(0)
D* Address of MQCD structure for client connection
D CNCCP 17 32* INZ(*NULL)
D* Ver:2 **
D* Queue manager connection tag
D CNCT 33 160 INZ(X'0000000000000000-
D 000000000000000000000000-
D 000000000000000000000000-
D 000000000000000000000000-
D 000000000000000000000000-
D 000000000000000000000000-
D 000000000000000000000000-
D 000000000000000000000000-
D 000000000000000000000000-
D 000000000000000000000000-
D 0000000000000000')
D* Ver:3 **
D* Address of MQSCO structure for client connection
D CNSCP 161 176* INZ(*NULL)
D* Offset of MQSCO structure for client connection
D CNSCO 177 180I 0 INZ(0)
D* Ver:4 **
D* Unique Connection Identifier
D CNCONID 181 204 INZ(X'0000000000000000-
D 000000000000000000000000-
D 000000')
D* Offset of MQCSP structure
D CNSECP0 205 208I 0 INZ(0)
D* Address of MQCSP structure
D CNSECPP 209 224* INZ(*NULL)
D* Ver:5 **
D* Address of CCDT URL string
D CNCCDTUP 225 240* INZ(*NULL)
D* Offset of CCDT URL string
D CNCCDTUO 241 244I 0 INZ(0)
D* Length of CCDT URL
D CNCCDTUL 245 248I 0 INZ(0)
D* Ver:6 **
D*
D*****
D** End of CMQCN0G **
D*****

```

MQCSP (Parámetros de seguridad) activado IBM i

Resumen de la estructura del MQCSP para IBM i.

Visión general

Propósito : La estructura MQCSP permite al servicio de autorización autenticar un ID de usuario y una contraseña. Especifique la estructura de los parámetros de seguridad de la conexión MQCSP en una llamada MQCONN.

Conjunto de caracteres y codificación : Los datos en MQCSP deben estar en el conjunto de caracteres dado por el atributo del **CodedCharSetId** gestor de colas y la codificación del gestor de colas local dada por ENNAT.

- “Campos” en la página 1075
- “Valores iniciales” en la página 1077
- “Declaración RPG” en la página 1077

Campos

La estructura MQCSP contiene los siguientes campos, que se describen en **orden alfabético** :

CSAUTHT (entero con signo de 10 dígitos)

Este es el tipo de autenticación que se debe realizar.

Los valores válidos son:

CSAN

No utilices los campos de ID de usuario y contraseña.

CSAUIAP

Autenticar los campos de ID de usuario y contraseña.

Este es un campo de entrada. El valor inicial de este campo es CSAN.

CSCPPL (entero con signo de 10 dígitos)

Esta es la longitud de la contraseña que se utilizará en la autenticación.

La longitud máxima de la contraseña no depende de la plataforma. Si la longitud de la contraseña es mayor que la permitida, la solicitud de autenticación falla con un error « RC2035 ».

Este es un campo de entrada. El valor inicial de este campo es 0.

CSCPPO (entero con signo de 10 dígitos)

Este es el desplazamiento en bytes de la contraseña que se utilizará en la autenticación.

El desplazamiento puede ser positivo o negativo.

Este es un campo de entrada. El valor inicial de este campo es 0.

CSCPPP (puntero)

Esta es la dirección de la contraseña que se utilizará en la autenticación.

Este es un campo de entrada. El valor inicial de este campo es el puntero nulo.

CSCSPUIL (entero con signo de 10 dígitos)

Esta es la longitud del ID de usuario que se utilizará en la autenticación.

La longitud máxima del ID de usuario no depende de la plataforma. Si la longitud del ID de usuario es mayor que la permitida, la solicitud de autenticación falla con un error « RC2035 ».

Este es un campo de entrada. El valor inicial de este campo es 0.

CSCSPUIO (entero con signo de 10 dígitos)

Este es el desplazamiento en bytes del ID de usuario que se utilizará en la autenticación.

El desplazamiento puede ser positivo o negativo.

Este es un campo de entrada. El valor inicial de este campo es 0.

CSCSPUIP (puntero)

Esta es la dirección del ID de usuario que se utilizará en la autenticación.

Este es un campo de entrada. El valor inicial de este campo es el puntero nulo. Este campo se ignora si CSVER es menor que CSVER5.

CSINITKL (entero con signo de 10 dígitos)

Esta es la longitud de la clave inicial para el sistema de protección con contraseña.

Este es un campo de entrada. El valor inicial de este campo es 0.

CSINITKO (entero con signo de 10 dígitos)

Este es el desplazamiento en bytes de la clave inicial para el sistema de protección con contraseña. El desplazamiento puede ser positivo o negativo.

Puede utilizar o *CSINITKPCSINITKO* para especificar la clave inicial, pero no ambos. Para obtener más información, consulte la descripción del *CSINITKP* campo.

Este es un campo de entrada. El valor inicial de este campo es 0.

CSINITKP (puntero)

Esta es la dirección en bytes de la clave inicial para el sistema de protección con contraseña.

Este es un campo de entrada. El valor inicial de este campo es el puntero nulo.

IBM MQ MQI clients puede proporcionar el valor de algunos campos como valores que han sido cifrados utilizando el sistema de IBM MQ protección con contraseña. Los siguientes campos pueden contener valores cifrados:

- La contraseña del repositorio de claves, en la estructura MQSCO.

El algoritmo de cifrado utiliza una clave inicial para cifrar y descifrar estos valores. Si se proporciona una clave inicial cuando los valores de estos campos se cifran utilizando la **runmqicred** utilidad, el cliente deberá especificar la misma clave inicial cuando se conecte al gestor de colas.

La clave inicial especificada mediante este campo anula cualquier clave inicial especificada mediante la variable de entorno *MQS_MQI_KEYFILE* o la *MQIInitialKeyFile* propiedad en la sección Security del archivo de configuración del cliente.

Puede utilizar o *CSINITKPCSINITKO* para especificar la clave inicial, pero no ambos.

CSRE1 (cadena de caracteres de 4 bytes)

Un campo reservado, necesario para la alineación del puntero en IBM i.

Este es un campo de entrada. El valor inicial de este campo es nulo.

CSRS2 (cadena de caracteres de 8 bytes)

Un campo reservado, necesario para la alineación del puntero en IBM i.

Este es un campo de entrada. El valor inicial de este campo es nulo.

CSSID (cadena de caracteres de 4 bytes)

Identificador de estructura.

El valor debe ser:

CSSIDV

Identificador de la estructura de parámetros de seguridad.

CSVER (entero con signo de 10 dígitos)

Número de versión de la estructura.

El valor debe ser:

CSVER1

Version-1 Estructura de los parámetros de seguridad.

CSVER2

Version-2 Estructura de los parámetros de seguridad.

La siguiente constante especifica el número de versión de la versión actual:

CSVERC

Versión actual de la estructura de los parámetros de seguridad.

Este campo siempre es un campo de entrada. El valor inicial de este campo es CSVER1.

Valores iniciales

Tabla 733. Campos en MQCNO		
Nombre de campo	Nombre de la constante	Valor de la constante
CSSID	CSSIDV	'CSP↵'
CSVER	CSVER1	1
CSAUTH	Ninguna	0
CSRE1	Ninguna	Nulos
CSCSPUIP	Ninguna	Puntero nulo
CSCSPUIO	Ninguna	0
CSCSPUIL	Ninguna	0
CSRS2	Ninguna	Nulos
CSCPPP	Ninguna	Puntero nulo
CSCPP0	Ninguna	0
CSCPPL	Ninguna	0
CSINITKP	Ninguna	Puntero nulo
CSINITKO	Ninguna	0
CSINITKL	Ninguna	0

Nota:

1. El símbolo ↵ representa un único carácter en blanco.

Declaración RPG

```

D*..1.....2.....3.....4.....5.....6.....7..
D*
D* MQCSP Structure
D*
D* Structure identifier
D CSSID 1 4 INZ('CSP ')
D* Structure version number
D CSVER 5 8I 0 INZ(1)
D* Type of authentication
D CSAUTH 9 12I 0 INZ(0)
D* Reserved
D CSRE1 13 16 INZ(X'00000000')
D* Address of user ID
D CSCSPUIP 17 32* INZ(*NULL)
D* Offset of user ID
D CSCSPUIO 33 36I 0 INZ(0)
D* Length of user ID
D CSCSPUIL 37 40I 0 INZ(0)
D* Reserved
D CSRS2 41 48 INZ(X'0000000000000000')
D* Address of password
D CSCPPP 49 64* INZ(*NULL)
D* Offset of password
D CSCPP0 65 68I 0 INZ(0)
D* Length of password
D CSCPPL 69 72I 0 INZ(0)

```

IBM i MQCTLO (Estructura de opciones de devolución de llamada de control) activada IBM i

Estructura que especifica la función de devolución de llamada de control.

Visión general

Finalidad

La estructura MQCTLO se utiliza para especificar opciones relacionadas con una función de devolución de llamada de control.

La estructura es un parámetro de entrada y salida en la llamada [MQCTL](#).

Versión

La versión actual de MQCTLO es CTLV1.

Conjunto de caracteres y codificación

Los datos en MQCTLO deben estar en el juego de caracteres proporcionado por el atributo del **CodedCharSetId** gestor de colas y la codificación del gestor de colas local proporcionada por ENNAT. Sin embargo, si la aplicación se ejecuta como un IBM MQ cliente, la estructura debe estar en el juego de caracteres y la codificación del cliente.

- “Campos” en la [página 1078](#)
- “Valores iniciales” en la [página 1079](#)
- “Declaración RPG” en la [página 1079](#)

Campos

La estructura MQCTLO contiene los siguientes campos, que se describen en orden alfabético:

COCONNAREA (entero con signo de 10 dígitos)

Estructura de opciones de control: campo « ConnectionArea ».

Este es un campo que está disponible para que lo utilice la función de devolución de llamada.

El gestor de colas no toma ninguna decisión basada en el contenido de este campo y se transfiere sin cambios desde el campo [CBCONNAREA](#) de la estructura MQCBC, que es un parámetro de la llamada MQCB.

Este campo se ignora para todas las operaciones excepto CTLSR y CTLSW.

Este es un campo de entrada y salida para la función de devolución de llamada. El valor inicial de este campo es un puntero nulo o bytes nulos.

COOPT (entero con signo de 10 dígitos)

Opciones que controlan la acción de MQCTLO.

CTLFQ

Forzar el fallo de la llamada MQCTLO si el gestor de colas o la conexión se encuentran en estado de inactividad.

Especifique GMFIQ en las opciones MQGMO pasadas en la llamada MQCB para que se envíe una notificación a los consumidores de mensajes cuando estén inactivos.

CTLTHR

Esta opción informa al sistema de que la aplicación requiere que todos los consumidores de mensajes, para la misma conexión, se llamen en el mismo subproceso.

Opción predeterminada : Si no necesita ninguna de las opciones descritas, utilice la siguiente opción:

CTLNO

Utilice este valor para indicar que no se han especificado otras opciones; todas las opciones toman sus valores predeterminados. CTLNO se define para facilitar la documentación del programa; no está previsto que esta opción se utilice con ninguna otra, pero como su valor es cero, dicho uso no puede detectarse.

Este es un campo de entrada. El valor inicial del campo *COOPT* es CTLNO.

CORSV (entero con signo de 10 dígitos)

Este es un campo reservado. El valor inicial de este campo es un carácter en blanco.

COSID (entero con signo de 10 dígitos)

Estructura de opciones de control: campo « StrucId ».

Este es el identificador de estructura; el valor debe ser:

CTLSI

Identificador para la estructura de opciones de control.

Este campo siempre es un campo de entrada. El valor inicial de este campo es CTLSI.

COVER (entero con signo de 10 dígitos)

Estructura de opciones de control - Campo Versión.

Este es el número de versión de la estructura; el valor debe ser:

CTLV1

Version-1 Estructura de opciones de control.

La siguiente constante especifica el número de versión de la versión actual:

CTLCV

Versión actual de la estructura de opciones de control.

Este campo siempre es un campo de entrada. El valor inicial de este campo es CTLV1.

Valores iniciales

Tabla 734. Campos en MQCTLO		
Nombre de campo	Nombre de la constante	Valor de la constante
<i>COSID</i>	CTLSI	'CTLO'
<i>COVER</i>	CTLV1	1
<i>COOPT</i>	CTLNO	Nulos
<i>CORSV</i>	Campo reservado	
<i>COCONNAREA</i>	Ninguna	Puntero nulo o bytes nulos

Declaración RPG

```
D* MQCTLO Structure
D*
D*
D* Structure identifier
D  COSID          1          4    INZ('CTLO')
D*
D* Structure version number
D  COVER          5          8I 0 INZ(1)
D*
D* Options that control the action of MQCTL
D  COOPT          9          12I 0 INZ(0)
D*
D* Reserved
D  CORSV         13          16I 0 INZ(-1)
D*
D* MQCTL Data area passed to the function
D  COCONNAREA    17          32*  INZ(*NULL)
```

MQDH (cabecera de distribución) en IBM i

La estructura MQDH describe los datos adicionales que están presentes en un mensaje cuando ese mensaje es un mensaje de lista de distribución almacenado en una cola de transmisión.

Visión general

Propósito : Un mensaje de lista de distribución es un mensaje que se envía a varias colas de destino. Los datos adicionales consisten en la estructura MQDH seguida de una matriz de registros MQOR y una matriz de registros MQPMR.

Esta estructura está destinada a aplicaciones especializadas que colocan mensajes directamente en colas de transmisión o que eliminan mensajes de colas de transmisión (por ejemplo: agentes de canales de mensajes).

Esta estructura no debe ser utilizada por aplicaciones normales que simplemente desean enviar mensajes a listas de distribución. Esas aplicaciones deben utilizar la estructura MQOD para definir los destinos en la lista de distribución, y la estructura MQPMO para especificar las propiedades de los mensajes o recibir información sobre los mensajes enviados a los destinos individuales.

Conjunto de caracteres y codificación : Los datos en MQDH deben estar en el conjunto de caracteres proporcionado por el atributo del **CodedCharSetId** gestor de colas y la codificación del gestor de colas local proporcionada por ENNAT para el lenguaje de programación C.

El conjunto de caracteres y la codificación del MQDH deben configurarse en los campos *MDCSI MDENC* y en:

- El MQMD (si la estructura MQDH se encuentra al inicio de los datos del mensaje), o
- La estructura de encabezado que precede a la estructura MQDH (todos los demás casos).

Uso : Cuando una aplicación envía un mensaje a una lista de distribución y algunos o todos los destinos son remotos, el gestor de colas antepone las estructuras MQXQH y MQDH a los datos del mensaje de la aplicación y coloca el mensaje en la cola de transmisión correspondiente. Por lo tanto, los datos aparecen en la siguiente secuencia cuando el mensaje se encuentra en una cola de transmisión:

- estructura MQXQH
- Estructura MQDH más matrices de registros MQOR y MQPMR
- Datos del mensaje de la aplicación

Dependiendo de los destinos, el gestor de colas puede generar más de un mensaje de este tipo y colocarlos en diferentes colas de transmisión. En este caso, las estructuras MQDH de esos mensajes identifican diferentes subconjuntos de los destinos definidos por la lista de distribución abierta por la aplicación.

Una aplicación que coloca un mensaje de lista de distribución directamente en una cola de transmisión debe ajustarse a la secuencia descrita anteriormente y debe garantizar que la estructura MQDH sea correcta. Si la estructura MQDH no es válida, el gestor de colas puede optar por rechazar la llamada MQPUT o MQPUT1 con el código de motivo RC2135.

Los mensajes solo se pueden almacenar en una cola en forma de lista de distribución si la cola está definida como capaz de admitir mensajes de lista de distribución (consulte el **DistLists** atributo de cola descrito en [“Atributos para colas” en la página 1399](#)). Si una aplicación coloca un mensaje de lista de distribución directamente en una cola que no admite listas de distribución, el gestor de colas divide el mensaje de lista de distribución en mensajes individuales y los coloca en la cola.

- [“Campos” en la página 1080](#)
- [“Valores iniciales” en la página 1083](#)
- [“Declaración RPG” en la página 1084](#)

Campos

La estructura MQDH contiene los siguientes campos, que se describen en **orden alfabético** :

DHCNT (entero con signo de 10 dígitos)

Número de registros MQOR presentes.

Esto define el número de destinos. Una lista de distribución debe contener siempre al menos un destino, por lo que *DHCNT* siempre debe ser mayor que cero.

El valor inicial de este campo es 0.

DHCSI (entero con signo de 10 dígitos)

Identificador del conjunto de caracteres de los datos que siguen a los registros MQOR y MQPMR.

Especifica el identificador del conjunto de caracteres de los datos que siguen a las matrices de registros MQOR y MQPMR; no se aplica a los datos de caracteres de la propia estructura MQDH.

En la llamada MQPUT o MQPUT1, la aplicación debe establecer este campo con el valor adecuado para los datos. Se puede utilizar el siguiente valor especial:

CSINHT

Heredar el identificador del conjunto de caracteres de esta estructura.

Los datos de caracteres en los datos *que siguen* esta estructura están en el mismo conjunto de caracteres que esta estructura.

El gestor de colas cambia este valor en la estructura enviada en el mensaje al identificador real del juego de caracteres de la estructura. Si no se produce ningún error, la llamada MQGET no devuelve el valor CSINHT.

CSINHT no se puede utilizar si el valor del *MDPAT* campo en MQMD es ATBRKR.

El valor inicial de este campo es CSUNDF.

DHENC (entero con signo de 10 dígitos)

Codificación numérica de datos que sigue los registros MQOR y MQPMR.

Esto especifica la codificación numérica de los datos que siguen a las matrices de registros MQOR y MQPMR; no se aplica a los datos numéricos de la propia estructura MQDH.

En la llamada MQPUT o MQPUT1, la aplicación debe establecer este campo con el valor adecuado para los datos.

El valor inicial de este campo es 0.

DHFLG (entero con signo de 10 dígitos)

Banderas generales.

Se puede especificar la siguiente bandera:

DHFNEW

Generar nuevos identificadores de mensajes.

Esta bandera indica que se generará un nuevo identificador de mensaje para cada destino de la lista de distribución. Esto solo se puede configurar cuando no hay registros de mensajes enviados, o cuando los registros están presentes pero no contienen el *PRMID* campo.

El uso de este indicador aplaza la generación de los identificadores de mensaje hasta el último momento posible, es decir, hasta el momento en que el mensaje de la lista de distribución se divide finalmente en mensajes individuales. Esto minimiza la cantidad de información de control que debe fluir con el mensaje de la lista de distribución.

Cuando una aplicación envía un mensaje a una lista de distribución, el gestor de colas establece DHFNEW en el MQDH que genera cuando ambas afirmaciones siguientes son ciertas:

- La aplicación no proporciona registros de mensajes enviados, o los registros proporcionados no contienen el *PRMID* campo.
- El *MDMID* campo en MQMD es MINONE, o el *PMOPT* campo en MQPMO incluye PMNMID

Si no se necesitan indicadores, se puede especificar lo siguiente:

DHFNON

Sin banderas.

Esta constante indica que no se han especificado indicadores. DHFNON se define para facilitar la documentación del programa. No está previsto que esta constante se utilice con ninguna otra, pero como su valor es cero, dicho uso no puede detectarse.

El valor inicial de este campo es DHFNON.

DHFMT (cadena de caracteres de 8 bytes)

Formato del nombre de los datos que siguen a los registros MQOR y MQPMR.

Especifica el nombre del formato de los datos que siguen a las matrices de registros MQOD y MQPMR (el que ocurra en último lugar).

En la llamada MQPUT o MQPUT1, la aplicación debe establecer este campo con el valor adecuado para los datos. Las reglas para codificar este campo son las mismas que para el *MDFMT* campo en MQMD.

El valor inicial de este campo es FMNONE.

DHLEN (entero con signo de 10 dígitos)

Longitud de la estructura MQDH más los registros MQOR y MQPMR siguientes.

Es el número de bytes desde el inicio de la estructura MQDH hasta el inicio de los datos del mensaje que siguen a las matrices de registros MQOR y MQPMR. Los datos se presentan en la siguiente secuencia:

- Estructura MQDH
- Matriz de registros MQOR
- Matriz de registros MQPMR
- Datos de mensaje

Las matrices de registros MQOR y MQPMR se abordan mediante desplazamientos contenidos en la estructura MQDH. Si estas compensaciones dan lugar a bytes no utilizados entre una o varias de las estructuras MQDH, las matrices de registros y los datos del mensaje, dichos bytes no utilizados deben incluirse en el valor de *DHLEN*, pero el gestor de colas no conserva el contenido de dichos bytes. Es válido que la matriz de registros MQPMR preceda a la matriz de registros MQOR.

El valor inicial de este campo es 0.

DHORO (entero con signo de 10 dígitos)

Desplazamiento del primer registro MQOR desde el inicio de MQDH.

Este campo indica la compensación en bytes del primer registro de la matriz de registros de objetos MQOR que contienen los nombres de las colas de destino. Hay *DHCNT* registros en esta matriz. Estos registros (más cualquier byte omitido entre el primer registro de objeto y el campo anterior) se incluyen en la longitud indicada por el *DHLEN* campo.

Una lista de distribución debe contener siempre al menos un destino, por lo que *DHORO* siempre debe ser mayor que cero.

El valor inicial de este campo es 0.

DHPRF (entero con signo de 10 dígitos)

Indicadores que señalan qué campos MQPMR están presentes.

Se pueden especificar cero o más de los siguientes indicadores:

PFMID

El campo identificador de mensaje está presente.

PFCID

El campo identificador de correlación está presente.

PFGID

El campo identificador de grupo está presente.

PFFB

El campo de comentarios está presente.

PFACC

El campo «Accounting-token» está presente.

Si no hay campos MQPMR, se puede especificar lo siguiente:

PFNONE

No hay campos de registro de mensajes enviados.

PFNONE se define para facilitar la documentación del programa. No está previsto que esta constante se utilice con ninguna otra, pero como su valor es cero, dicho uso no puede detectarse.

El valor inicial de este campo es PFNONE.

DHPRO (entero con signo de 10 dígitos)

Desplazamiento del primer registro MQPMR desde el inicio de MQDH.

Este campo indica la compensación en bytes del primer registro de la matriz de registros de mensajes MQPMR put que contienen las propiedades del mensaje. Si está presente, hay *DHCNT* registros en esta matriz. Estos registros (más cualquier byte omitido entre el primer registro del mensaje put y el campo anterior) se incluyen en la longitud indicada por el *DHLEN* campo.

Los registros de mensajes son opcionales; si no se proporcionan registros, *DHPRO* es cero y *DHPRF* tiene el valor PFNONE.

El valor inicial de este campo es 0.

DHSID (cadena de caracteres de 4 bytes)

Identificador de estructura.

El valor debe ser:

DHSIDV

Identificador para la estructura del encabezado de distribución.

El valor inicial de este campo es DHSIDV.

DHVER (entero con signo de 10 dígitos)

Número de versión de la estructura.

El valor debe ser:

DHVER1

Número de versión para la estructura del encabezado de distribución.

La siguiente constante especifica el número de versión de la versión actual:

DHVERC

Versión actual de la estructura del encabezado de distribución.

El valor inicial de este campo es DHVER1.

Valores iniciales

Tabla 735. Campos en MQDH		
Nombre de campo	Nombre de la constante	Valor de la constante
<i>DHSID</i>	DHSIDV	'DH??'
<i>DHVER</i>	DHVER1	1
<i>DHLEN</i>	Ninguna	0
<i>DHENC</i>	Ninguna	0

Tabla 735. Campos en MQDH (continuación)

Nombre de campo	Nombre de la constante	Valor de la constante
DHCSI	CSUNDF	0
DHFMF	FMNONE	Espacios en blanco
DHFLG	DHFNON	0
DHPRF	PFNONE	0
DHCNT	Ninguna	0
DHORO	Ninguna	0
DHPRO	Ninguna	0

Notas:

1. El símbolo ~ representa un único carácter en blanco.

Declaración RPG

```

D*.1....:....2....:....3....:....4....:....5....:....6....:....7..
D* MQDH Structure
D*
D* Structure identifier
D  DHSID          1          4      INZ('DH ')
D* Structure version number
D  DHVER          5          8I 0 INZ(1)
D* Length of MQDH structure plusfollowing MQOR and MQPMR records
D  DHLEN          9          12I 0 INZ(0)
D* Numeric encoding of data that followsthe MQOR and MQPMR records
D  DHENC         13          16I 0 INZ(0)
D* Character set identifier of data thatfollows the MQOR and MQPMR
D* records
D  DHCSI         17          20I 0 INZ(0)
D* Format name of data that follows theMQOR and MQPMR records
D  DHFMF         21          28      INZ(' ')
D* General flags
D  DHFLG         29          32I 0 INZ(0)
D* Flags indicating which MQPMR fieldsare present
D  DHPRF         33          36I 0 INZ(0)
D* Number of MQOR records present
D  DHCNT         37          40I 0 INZ(0)
D* Offset of first MQOR record from startof MQDH
D  DHORO         41          44I 0 INZ(0)
D* Offset of first MQPMR record fromstart of MQDH
D  DHPRO         45          48I 0 INZ(0)

```

IBM i MQDLH (Encabezado de carta muerta) en IBM i

Visión general

Finalidad

La MQDLH estructura describe la información que precede a los datos del mensaje de aplicación de los mensajes en la cola de mensajes no entregados (mensajes no entregados). Un mensaje puede llegar a la cola de mensajes no entregados porque el gestor de colas o el agente del canal de mensajes lo han redirigido a dicha cola. Una aplicación puede colocar el mensaje directamente en la cola.

Nombre de formato

FMDLH

Conjunto de caracteres y codificación

El MQDLH podría estar al principio de los datos del mensaje de la aplicación. Si es así, los campos de la MQDLH estructura están en el juego de caracteres y la codificación indicados por los campos MDCSI y

MDENC. Si no es así, el juego de caracteres y la codificación se establecen mediante los campos MDCSI y MDENC en la estructura de encabezado que precede al MQDLH.

El juego de caracteres debe ser uno que tenga caracteres de un solo byte para los caracteres que son válidos en los nombres de las colas.

Utilización

Las aplicaciones que colocan mensajes directamente en la cola de mensajes no entregados deben anteponer a los datos del mensaje una MQDLH estructura e inicializar los campos con los valores adecuados. Sin embargo, el gestor de colas no requiere que haya una MQDLH estructura presente, ni que se especifiquen valores válidos para los campos.

Si un mensaje es demasiado largo para colocarlo en la cola de mensajes no entregados, la aplicación debe considerar realizar una de las siguientes acciones:

- Truncar los datos del mensaje para que quepan en la cola de mensajes no entregados.
- Graba el mensaje en el almacenamiento auxiliar y coloca un mensaje de informe de excepción en la cola de mensajes no entregados indicando que el mensaje es demasiado largo.
- Descartar el mensaje y devolver un error a su remitente. Si el mensaje es un mensaje crítico. Deseche el mensaje solo si se sabe que el remitente aún tiene una copia del mensaje. Por ejemplo, un mensaje recibido por un agente de canal de mensajes desde un canal de comunicación.

La elección adecuada depende del diseño de la aplicación.

El gestor de colas realiza un procesamiento especial cuando un mensaje que es un segmento se coloca con una MQDLH estructura al principio. Consulte la descripción de la MQMDE estructura para obtener más detalles.

- [“Colocar mensajes en la cola de mensajes no entregados” en la página 1085](#)
- [“Obtener mensajes de la cola de mensajes no entregados” en la página 1086](#)
- [“Campos” en la página 1086](#)
- [“Valores iniciales” en la página 1090](#)
- [“Declaración RPG” en la página 1090](#)

Colocar mensajes en la cola de mensajes no entregados

Si un mensaje se coloca en la cola de mensajes no entregados, la MQMD estructura utilizada para la llamada MQPUT MQPUT1 o debe ser idéntica a la MQMD asociada al mensaje. El MQMD es normalmente el devuelto por la MQGET llamada, excepto en los siguientes casos:

- Los campos MDCSI y MDENC deben configurarse con el juego de caracteres y la codificación que se utilicen para los campos de la MQDLH estructura.
- El campo MDFMT debe establecerse en FMDLH para indicar que los datos comienzan con una MQDLH estructura.
- Los campos de contexto MDACC, MDAID, MDAOD, MDPAN, MDPAT, MDPD, MDPT y MDUID deben configurarse utilizando una opción de contexto adecuada a las circunstancias:
 - Una aplicación que coloque en la cola de mensajes no entregables un mensaje que no esté relacionado con ningún mensaje anterior debe utilizar la opción PMDEFC. La opción PMDEFC hace que el gestor de colas establezca todos los campos de contexto del descriptor de mensajes en sus valores predeterminados.
 - Una aplicación de servidor que coloca en la cola de mensajes no entregados un mensaje que ha recibido debe utilizar la opción PMPASA para conservar la información contextual original.
 - Una aplicación de servidor que coloca en la cola de mensajes no entregados una respuesta a un mensaje que ha recibido debe utilizar la opción PMPASI. La opción PMPASI conserva la información de identidad, pero establece la información de origen como la de la aplicación del servidor.

- Un agente de canal de mensajes que coloque en la cola de mensajes no entregados un mensaje que haya recibido de su canal de comunicación debe utilizar la opción PMSETA. La opción PMSETA conserva la información contextual original.

En la MQDLH estructura misma, los campos deben configurarse de la siguiente manera:

- Los campos *DLFMT*, *DLCSI*, *DLENC* y *DLENL* deben establecerse en los valores que describen los datos que siguen a la MQDLH estructura. Estos valores suelen ser los valores del descriptor del mensaje original.
- Los campos de contexto *DLPAT*, *DLPAN*, *DLPD* y *DLPT* deben establecerse con valores adecuados para la aplicación que coloca el mensaje en la cola de mensajes no entregados. Estos valores no están relacionados con el mensaje original.
- Los demás campos deben configurarse según corresponda.

La aplicación debe garantizar que todos los campos tengan valores válidos y que los campos de caracteres se rellenen con espacios en blanco hasta alcanzar la longitud definida para el campo. Los datos de caracteres no deben terminarse prematuramente utilizando un carácter nulo. El gestor de colas no convierte el carácter nulo y los caracteres posteriores en espacios en blanco en la MQDLH estructura.

Obtener mensajes de la cola de mensajes no entregados

Las aplicaciones que reciben mensajes de la cola de mensajes no entregados deben verificar que los mensajes comiencen con una MQDLH estructura. La aplicación puede determinar si una MQDLH estructura está presente examinando el campo *MDFMT* en el descriptor del mensaje *MQMD*. Si el campo tiene el valor *FMDLH*, los datos del mensaje comienzan con una MQDLH estructura. Los mensajes de la cola de mensajes no entregados pueden truncarse si originalmente eran demasiado largos para la cola a la que estaban destinados.

Campos

La MQDLH estructura contiene los siguientes campos, que se describen en orden alfabético:

DLCSI (entero con signo de 10 dígitos)

Identificador del conjunto de caracteres de los datos que siguen MQDLH.

DLCSI especifica el identificador del conjunto de caracteres de los datos que siguen a la MQDLH estructura. Los datos suelen proceder del mensaje original. No se aplica a los datos de caracteres en la MQDLH estructura misma.

En la MQPUT llamada MQPUT1 o, la aplicación debe establecer este campo con el valor adecuado para los datos. Se puede utilizar el siguiente valor especial:

CSINHT

Heredar el identificador del conjunto de caracteres de esta estructura.

Los datos de caracteres en los datos que siguen esta estructura están en el mismo conjunto de caracteres que esta estructura.

El gestor de colas cambia este valor en la estructura enviada en el mensaje al identificador real del juego de caracteres de la estructura. Si no se produce ningún error, el valor no CSINHT es devuelto por la MQGET llamada.

CSINHT No se puede utilizar si el valor del campo *MDPAT* en *MQMD* es *ATBRKR*.

El valor inicial de este campo es *CSUNDF*.

DLDM (cadena de caracteres de 48 bytes)

Nombre del gestor de colas de destino original.

Este es el nombre del gestor de colas que era el destino original del mensaje.

La longitud de este campo viene dada por *LNQMN*. El valor inicial de este campo es 48 caracteres en blanco.

DLDQ (cadena de caracteres de 48 bytes)

Nombre de la cola de destino original.

Este es el nombre de la cola de mensajes que era el destino original del mensaje.

La longitud de este campo viene dada por LNQN. El valor inicial de este campo es 48 caracteres en blanco.

DLENC (entero con signo de 10 dígitos)

Codificación numérica de los datos que siguen a MQDLH.

DLENC especifica la codificación numérica de los datos que siguen la MQDLH estructura. Los datos suelen proceder del mensaje original. No se aplica a los datos numéricos de la MQDLH propia estructura.

En la MQPUT llamada MQPUT1 o, la aplicación debe establecer este campo con el valor adecuado para los datos.

El valor inicial de este campo es 0.

DLFMT (cadena de caracteres de 8 bytes)

Formatear el nombre de los datos que siguen MQDLH.

Especifica el nombre del formato de los datos que siguen a la MQDLH estructura (normalmente los datos del mensaje original).

En la MQPUT llamada MQPUT1 o, la aplicación debe establecer este campo con el valor adecuado para los datos. Las reglas para codificar este campo son las mismas que las reglas para el campo MDFMT en MQMD.

La longitud de este campo viene dada por LNFMT. El valor inicial de este campo es FMNONE.

DLPAN (cadena de caracteres de 28 bytes)

Nombre de la aplicación que colocó el mensaje en la cola de mensajes no entregados (mensajes no entregados).

El formato del nombre depende del campo DLPAT. Consulte la descripción del campo MDPAN en [“MQMD \(descriptor de mensaje\) en IBM i”](#) en la página 1131.

Si es el gestor de colas el que redirige el mensaje a la cola de mensajes no entregados, DLPAN contiene los primeros 28 caracteres del nombre del gestor de colas. El nombre se rellena con espacios en blanco si es necesario.

La longitud de este campo viene dada por LNPAN. El valor inicial de este campo es 28 caracteres en blanco.

DLPAT (entero con signo de 10 dígitos)

Tipo de aplicación que coloca el mensaje en la cola de mensajes no entregados (mensajes no entregados).

Este campo tiene el mismo significado que el campo MDPAT en el descriptor del mensaje MQMD (véase [“MQMD \(descriptor de mensaje\) en IBM i”](#) en la página 1131 para más detalles).

Si es el gestor de colas el que redirige el mensaje a la cola de mensajes no entregados, DLPAT tiene el valor ATQM.

El valor inicial de este campo es 0.

DLPD (cadena de caracteres de 8 bytes)

Fecha en la que el mensaje se colocó en la cola de mensajes no entregados (mensajes no entregados).

El formato utilizado para la fecha cuando este campo es generado por el gestor de colas es:

- YYYYMMDD

donde los caracteres representan:

YYYY

año (cuatro dígitos numéricos)

MM

mes del año (del 01 al 12)

DD

día del mes (del 01 al 31)

La hora del meridiano de Greenwich (GMT) se utiliza para los campos DLPD y DLPT, siempre que el reloj del sistema esté ajustado con precisión a la GMT.

La longitud de este campo viene dada por LNPDAT. El valor inicial de este campo es ocho caracteres en blanco.

DLPT (cadena de caracteres de 8 bytes)

Hora en la que el mensaje se colocó en la cola de mensajes no entregados (mensajes no entregados).

El formato utilizado para la hora en que este campo es generado por el gestor de colas es:

- HHMMSSTH

donde los caracteres representan (en orden):

HH

horas (de 00 a 23)

MM

minutos (de 00 a 59)

SS

segundos (de 00 a 59; véase la nota más adelante en este tema)

T

décimas de segundo (0 a 9)

H

centésimas de segundo (0 a 9)

Nota: Si el reloj del sistema está sincronizado con un estándar de tiempo preciso, es posible que se devuelva 60 o 61 para los segundos en DLPT. El segundo adicional se produce cuando se insertan segundos intercalares en el estándar horario mundial.

La hora del meridiano de Greenwich (GMT) se utiliza para los campos DLPD y DLPT, siempre que el reloj del sistema esté ajustado con precisión a la GMT.

La longitud de este campo viene dada por LNPTIM. El valor inicial de este campo es ocho caracteres en blanco.

DLREA (entero con signo de 10 dígitos)

Mensaje de motivo recibido en la cola de mensajes no entregados (mensajes no entregados).

Esto identifica el motivo por el cual el mensaje se colocó en la cola de mensajes no entregados en lugar de en la cola de destino original. Debe ser uno de los valores FB* o RC* (por ejemplo, RC2053). Consulte la descripción del *MDFB* campo en [“MQMD \(descriptor de mensaje\) en IBM i”](#) en la [página 1131](#) para obtener más información sobre los valores FB* comunes que pueden aparecer.

Si el valor está en el rango FBIFST a FBILST, el código de error IMS real se puede determinar restando FBIERR del valor del *DLREA* campo.

Algunos FB* valores solo aparecen en este campo. Se refieren a mensajes de repositorio, mensajes de activación o mensajes de cola de transmisión que se transfieren a la cola de mensajes no entregados. Estos valores son:

FBABEG

No se puede iniciar la aplicación.

Una aplicación que procesa un mensaje de activación no pudo iniciar la aplicación indicada en el campo TMAI del mensaje de activación; consulte [“MQTM - Mensaje de activación” en la página 1263](#).

FBATYP

Error de tipo de aplicación.

Una aplicación que procesa un mensaje de activación no pudo iniciarse porque el campo TMAT del mensaje de activación no es válido; véase [“MQTM - Mensaje de activación” en la página 1263](#).

FBOCD

Canal receptor de clúster eliminado.

El mensaje estaba en una cola de transmisión de clúster destinada a una cola de clúster que se abrió con la opción FBIERR. El canal del receptor del clúster remoto que se iba a utilizar para transmitir el mensaje a la cola de destino se eliminó antes de que se pudiera enviar el mensaje. Debido a que se especificó FBIERR, solo se puede utilizar el canal seleccionado al abrir la cola para transmitir el mensaje. Como este canal ya no está disponible, el mensaje se colocó en la cola de mensajes no entregados.

FBNARM

El mensaje no es un mensaje de repositorio.

FBSBCX

Mensaje detenido por salida de definición automática del canal.

FBSBMX

Mensaje detenido por salida de mensaje del canal.

FBTM

MQTM Estructura no válida o faltante.

El campo MDFMT en MQMD especifica FMTM, pero el mensaje no comienza con una estructura MQTM válida. Por ejemplo, el *TMSID* mnemotécnico llamativo podría no ser válido. Es *TMVER* posible que no se reconozca. La longitud del mensaje desencadenante podría ser insuficiente para contener la MQTM estructura.

FBXQME

Mensaje en la cola de transmisión con formato incorrecto.

Un agente del canal de mensajes ha detectado que un mensaje de la cola de transmisión no tiene el formato correcto. El agente del canal de mensajes coloca el mensaje en la cola de mensajes no entregados utilizando este código de respuesta.

El valor inicial de este campo es RCNONE.

DLSID (cadena de caracteres de 4 bytes)

Identificador de estructura.

El valor debe ser:

DLSIDV

Identificador para la estructura del encabezado de carta muerta.

El valor inicial de este campo es DLSIDV.

DLVER (entero con signo de 10 dígitos)

Número de versión de la estructura.

El valor debe ser:

DLVER1

Número de versión para la estructura del encabezado de correo no entregado.

La siguiente constante especifica el número de versión de la versión actual:

DLVERC

Versión actual de la estructura del encabezado de correo no entregado.

El valor inicial de este campo es DLVER1.

Valores iniciales

Tabla 736. Entrada de campos MQDLH		
Nombre de campo	Nombre de la constante	Valor de la constante
DLSID	DLSIDV	'DLH↵'
DLVER	DLVER1	1
DLREA	RCNONE	0
DLDQ	Ninguna	Espacios en blanco
DLDM	Ninguna	Espacios en blanco
DLENC	Ninguna	0
DLCSI	CSUNDF	0
DLFMT	FMNONE	Espacios en blanco
DLPAT	Ninguna	0
DLPAN	Ninguna	Espacios en blanco
DLPD	Ninguna	Espacios en blanco
DLPT	Ninguna	Espacios en blanco

Notas:

1. El símbolo ↵ representa un único carácter en blanco.

Declaración RPG

```
D*.1.....2.....3.....4.....5.....6.....7..
D* MQDLH Structure
D*
D* Structure identifier
D DLSID 1 4 INZ('DLH ')
D* Structure version number
D DLVER 5 8I 0 INZ(1)
D* Reason message arrived on dead-letter(undelivered-message) queue
D DLREA 9 12I 0 INZ(0)
D* Name of original destination queue
D DLDQ 13 60 INZ
D* Name of original destination queue manager
D DLDM 61 108 INZ
D* Numeric encoding of data that followsMQDLH
D DLENC 109 112I 0 INZ(0)
D* Character set identifier of data thatfollows MQDLH
D DLCSI 113 116I 0 INZ(0)
D* Format name of data that followsMQDLH
D DLFMT 117 124 INZ(' ')
D* Type of application that put messageon dead-letter
D* (undelivered-message)queue
D DLPAT 125 128I 0 INZ(0)
D* Name of application that put messageon dead-letter
D* (undelivered-message)queue
D DLPAN 129 156 INZ
D* Date when message was put ondead-letter (undelivered-message)queue
D DLPD 157 164 INZ
D* Time when message was put on thedead-letter (undelivered-message)queue
D DLPT 165 172 INZ
```

IBM i MQDMHO (Opciones para eliminar el identificador de mensaje) activado IBM i

La **MQDMHO** estructura permite a las aplicaciones especificar opciones que controlan cómo se eliminan los identificadores de mensajes.

Visión general

Propósito : La estructura es un parámetro de entrada en la **MQDLTMH** llamada.

Conjunto de caracteres y codificación : Los datos en **MQDMHO** deben estar en el conjunto de caracteres de la aplicación y la codificación de la aplicación (ENNAT).

- “Campos” en la [página 1091](#)
- “Valores iniciales” en la [página 1091](#)
- “Declaración RPG” en la [página 1092](#)

Campos

La estructura MQDMHO contiene los siguientes campos, que se describen en **orden alfabético** :

DMOPT (entero con signo de 10 dígitos)

El valor debe ser:

DMNONE

No se ha especificado ninguna opción.

Este campo siempre es un campo de entrada. El valor inicial de este campo es **DMNONE**.

DMSID (entero con signo de 10 dígitos)

Este es el identificador de estructura; el valor debe ser:

DMSIDV

Identificador para la estructura de opciones del identificador de mensajes eliminados.

Este campo siempre es un campo de entrada. El valor inicial de este campo es **DMSIDV**.

DMVER (entero con signo de 10 dígitos)

Este es el número de versión de la estructura; el valor debe ser:

DMVER1

Version-1 Eliminar la estructura de opciones del identificador de mensaje.

La siguiente constante especifica el número de versión de la versión actual:

DMVERC

Versión actual de la estructura de opciones de eliminación de mensajes.

Este campo siempre es un campo de entrada. El valor inicial de este campo es **DMVER1**.

Valores iniciales

Tabla 737. Campos en MQDMHO		
Nombre de campo	Nombre de la constante	Valor de la constante
DMSID	DMSIDV	' DMHO '
DMVER	DMVER1	1

Tabla 737. Campos en MQDMHO (continuación)

Nombre de campo	Nombre de la constante	Valor de la constante
DMOPT	DMNONE	0

Declaración RPG

```

D* MQDMHO Structure
D*
D*
D* Structure identifier
D  DMSID          1      4    INZ('DMHO')
D*
D* Structure version number
D  DMVER          5      8I 0 INZ(1)
D*
D* Options that control the action of MQDLTMH
D  DMOPT          9     12I 0 INZ(0)

```

IBM i MQDMPO (Eliminar opciones de propiedades del mensaje) activado

Estructura que define las opciones de la propiedad «eliminar mensaje».

Visión general

Propósito : La estructura MQDMPO permite a las aplicaciones especificar opciones que controlan cómo se eliminan las propiedades de los mensajes. La estructura es un parámetro de entrada en la llamada MQDLTMP.

Conjunto de caracteres y codificación : Los datos en MQDMPO deben estar en el conjunto de caracteres y la codificación de la aplicación (ENNAT).

- “Campos” en la página 1092
- “Valores iniciales” en la página 1093
- “Declaración RPG” en la página 1093

Campos

La estructura MQDMPO contiene los siguientes campos, descritos en orden alfabético:

DPOPT (entero con signo de 10 dígitos)

Eliminar estructura de opciones de propiedad de mensaje - campo DPOPT.

Opciones de ubicación: Las siguientes opciones se refieren a la ubicación relativa de la propiedad en comparación con el cursor de la propiedad.

DPSELF

Elimina la primera propiedad que coincide con el nombre especificado.

DPDELC

Elimina la propiedad a la que apunta el cursor de propiedad; es decir, la última propiedad consultada mediante la opción IPINQF o IPINQN.

El cursor de propiedad se restablece cuando se reutiliza el identificador de mensaje. También se restablece cuando se especifica el identificador de mensaje en el campo HMSG de MQGMO en una llamada MQGET, o en la estructura MQPMO en una llamada MQPUT.

El cursor de propiedad se restablece cuando se reutiliza el identificador de mensaje o cuando el identificador de mensaje se especifica en el campo HMSG de la estructura MQGMO en una estructura MQGET en una llamada MQGET o en una estructura MQPMO en una llamada MQPUT.

La llamada falla con el código de finalización CCFAIL y el motivo RC2471 si se utiliza esta opción cuando aún no se ha establecido el cursor de propiedad. También falla con estos códigos si la propiedad a la que apunta el cursor de propiedad ya ha sido eliminada.

Si ninguna de estas opciones es necesaria, se puede utilizar la siguiente opción:

DPNONE

No se ha especificado ninguna opción.

El valor inicial de este campo de entrada es DPDELF.

DPSID (entero con signo de 10 dígitos)

Eliminar estructura de opciones de propiedad de mensaje - campo DPSID.

Este es el identificador de la estructura. El valor debe ser:

DPSIDV

Identificador para la estructura de opciones de la propiedad de mensaje eliminado.

Este campo es siempre un campo de entrada. El valor inicial de este campo es DPSIDV.

DPVER (entero con signo de 10 dígitos)

Eliminar estructura de opciones de propiedad del mensaje: campo DPVER.

Este es el número de versión de la estructura. El valor debe ser:

DPVER1

Número de versión para la estructura de opciones de la propiedad de eliminación de mensajes.

La siguiente constante especifica el número de versión de la versión actual:

DPVERC

Estructura actual de las opciones de eliminación de propiedades de mensajes.

Este campo es siempre un campo de entrada. El valor inicial de este campo es DPVER1.

Valores iniciales

Tabla 738. Campos en MQDPMO		
Nombre de campo	Nombre de la constante	Valor de la constante
DPSID	DPSIDV	'DMPO'
DPVER	DPVER1	1
DPOPT	Opciones que controlan la acción de MQDLTMP	DPNONE

Declaración RPG

```
D* MQDPMO Structure
D*
D*
D* Structure identifier
D  DPSID              1      4    INZ('DMPO')
D*
D* Structure version number
D  DPVER              5      8I 0  INZ(1)
D*
** Options that control the action of
D* MQDLTMP
D  DPOPT              9     12I 0  INZ(0)
```

MQEPH (encabezado PCF integrado) en IBM i

Visión general

Finalidad

La estructura MQEPH describe los datos adicionales que están presentes en un mensaje cuando dicho mensaje es un mensaje de formato de comando programable (PCF). El *EPFMT* campo define los parámetros PCF que siguen esta estructura y esto le permite seguir los datos del mensaje PCF con otros encabezados.

Nombre de formato

EPFMT

Conjunto de caracteres y codificación

Los datos en MQEPH deben estar en el juego de caracteres y la codificación del gestor de colas local; esto viene dado por el atributo del **CCSID** gestor de colas.

Configure el juego de caracteres y la codificación del MQEPH en los campos *MDCSI MDENC* y en:

- El MQMD (si la estructura MQEPH se encuentra al inicio de los datos del mensaje), o
- La estructura de encabezado que precede a la estructura MQEPH (en todos los demás casos).

Utilización

No se pueden utilizar estructuras MQEPH para enviar comandos al servidor de comandos ni a ningún otro servidor que acepte PCF del gestor de colas.

Del mismo modo, el servidor de comandos o cualquier otro servidor gestor de colas que acepte PCF no generan respuestas ni eventos que contengan estructuras MQEPH.

- “Campos” en la [página 1094](#)
- “Valores iniciales” en la [página 1096](#)
- “Declaración RPG” en la [página 1096](#)

Campos

La estructura MQEPH contiene los siguientes campos, que se describen en **orden alfabético** :

EPCSI (entero con signo de 10 dígitos)

Este es el identificador del conjunto de caracteres de los datos que siguen a la estructura MQEPH y los parámetros PCF asociados; no se aplica a los datos de caracteres de la propia estructura MQEPH.

El valor inicial de este campo es EPCUND.

EPENC (entero con signo de 10 dígitos)

Se trata de la codificación numérica de los datos que siguen la estructura MQEPH y los parámetros PCF asociados; no se aplica a los datos de caracteres de la propia estructura MQEPH.

El valor inicial de este campo es 0.

EPFLG (entero con signo de 10 dígitos)

Están disponibles los valores siguientes:

EPNONE

No se han especificado banderas. *MDCSI* EPNONE se define para ayudar en la documentación del programa. No está previsto que esta constante se utilice con ninguna otra, pero como su valor es cero, dicho uso no puede detectarse.

EPCSEM

El conjunto de caracteres de los parámetros que contienen datos de caracteres se especifica individualmente dentro del *CCSID* campo de cada estructura. El conjunto de caracteres de los campos *EPSIDEPFMT* y se define mediante el *CCSID* en la estructura de encabezado que precede a la estructura MQEPH, o mediante el *MDCSI* campo en MQMD si MQEPH se encuentra al principio del mensaje.

El valor inicial de este campo es EPNONE.

EPFMT (cadena de caracteres de 8 bytes)

Este es el nombre del formato de los datos que siguen la estructura MQEPH y los parámetros PCF asociados.

El valor inicial de este campo es EPFMNO.

EPLEN (entero con signo de 10 dígitos)

Es la cantidad de datos que preceden a la siguiente estructura de encabezado. Incluye:

- La longitud del encabezado MQEPH
- La longitud de todos los parámetros PCF que siguen al encabezado
- Cualquier relleno en blanco que siga a esos parámetros

EPLEN debe ser un múltiplo de 4.

La parte de longitud fija de la estructura se define mediante EPSTLF.

El valor inicial de este campo es 68.

EPPCFH (MQCFH)

Este es el encabezado del formato de comando programable (PCF), que define los parámetros PCF que siguen la estructura MQEPH. Esto le permite seguir los datos del mensaje PCF con otras cabeceras.

El encabezado PCF se define inicialmente con los siguientes valores:

Tabla 739. Campos en EPPCFH		
Nombre de campo	Nombre de la constante	Valor de la constante
EP3TYP	CFTNON	0
EP3LEN	FHLENV	36
EP3VER	FHVER3	3
EP3CMD	CMNONE	0
EP3SEQ	Ninguna	1
EP3CTL	CFCLST	1
EEP3CC	CCOK	0
EP3REA	RCNONE	0
EP3CNT	Ninguna	0

La aplicación debe cambiar EP3TYP de CFTNON a un tipo de estructura válido para el uso que está haciendo del encabezado PCF incrustado.

EPSID (cadena de caracteres de 4 bytes)

El valor debe ser:

EPSTID

Identificador de la estructura de encabezado PCF incrustada.

El valor inicial de este campo es EPSTID.

EPVER (entero con signo de 10 dígitos)

El valor puede ser:

EPVER1

Número de versión para la estructura de encabezado PCF incrustada.

La siguiente constante especifica el número de versión de la versión actual:

EPVER3

Versión actual de la estructura del encabezado PCF integrado.

El valor inicial de este campo es EPVER3.

Valores iniciales

Tabla 740. Campos en MQEPH		
Nombre de campo	Nombre de la constante	Valor de la constante
EPSID	EPSTID	'EP↵↵'
EPVER	EPVER1	1
EPLen	EPSTLF	68
EPENC	Ninguna	0
EPCSI	EPCUND	0
EPFMT	EPFMNO	Espacios en blanco
EPFLG	EPNONE	0
EPPCFH	Nombres y valores tal y como se definen en Tabla 739 en la página 1095	0

Nota:

1. El símbolo ↵ representa un único carácter en blanco.

Declaración RPG

```
D*.1.....2.....3.....4.....5.....6.....7..
D* MQEPH Structure
D*
D* Structure identifier
D EPSID          1          4
D* Structure version number
D EPVER          5          8I 0
D* Total length of MQEPH including MQCFHand parameter structures
D* that follow
D EPLEN          9          12I 0
D* Numeric encoding of data that follows last PCF parameter structure
D EPENC          13         16I 0
D* Character set identifier of data that follows last PCF parameter
D* structure
D EPCSI          17         20I 0
D* Format name of data that follows last PCF parameter structure
D EPFMT          21         28
D* Flags
D EPFLG          29         32I 0
D* Programmable Command Format Header
D EP3TYP          33         36I 0
D EP3LEN          37         40I 0
D EP3VER          41         44I 0
D EP3CMD          45         48I 0
D EP3SEQ          49         52I 0
D EP3CTL          53         56I 0
D EP3CC           57         60I 0
D EP3REA          61         64I 0
D EP3CNT          65         68I 0
```

MQGMO (Opciones de obtención de mensajes) activado IBM i

La estructura MQGMO permite a la aplicación especificar opciones que controlan cómo se eliminan los mensajes de las colas.

Visión general

Finalidad

La estructura es un parámetro de entrada/salida en la llamada MQGET.

Versión

La versión actual de MQGMO es GMVER4. Los campos que solo existen en las versiones más recientes de la estructura se identifican como tales en las descripciones que siguen.

El archivo COPY proporcionado contiene la versión más reciente de MQGMO compatible con el entorno, pero con el valor inicial del *GMVER* campo establecido en GMVER1. Para utilizar campos que no están presentes en la estructura version-1, la aplicación debe establecer el *GMVER* campo en el número de versión requerido.

Conjunto de caracteres y codificación

Los datos en MQGMO deben estar en el conjunto de caracteres proporcionado por el atributo del **CodedCharSetId** gestor de colas y la codificación del gestor de colas local proporcionada por ENNAT. Sin embargo, si la aplicación se ejecuta como un IBM MQ cliente, la estructura debe estar en el juego de caracteres y la codificación del cliente.

- “Campos” en la [página 1097](#)
- “Valores iniciales” en la [página 1118](#)
- “Declaración RPG” en la [página 1118](#)

Campos

La estructura MQGMO contiene los siguientes campos, que se describen en orden alfabético:

GMGST (cadena de caracteres de 1 byte)

Indicador que señala si el mensaje recuperado pertenece a un grupo.

Tiene uno de los siguientes valores:

GSNIG

El mensaje no está en un grupo.

GSMIG

El mensaje está en un grupo, pero no es el último del grupo.

GSLMIG

El mensaje es el último del grupo.

Este valor también es el valor devuelto si el grupo consta de un solo mensaje.

Este campo es un campo de salida. El valor inicial de este campo es GSNIG. Este campo se ignora si *GMVER* es menor que GMVER2.

GMMH (entero con signo de 10 dígitos)

descriptor de contexto de mensaje

Si se especifica la opción GMPRAQ y el atributo de cola PRPCTL no está establecido en PRPRFH, este es el identificador de un mensaje que se rellena con las propiedades del mensaje que se recupera de la cola. El identificador se crea mediante una llamada MQCRTMH. Las propiedades que ya están asociadas al descriptor se borran antes de recuperar un mensaje.

También se puede especificar el siguiente valor:

MQHM_NINGUNO

No se ha proporcionado ningún identificador de mensaje.

No se requiere ningún descriptor de mensaje en la llamada MQGET si se proporciona un identificador de mensaje válido y se utiliza en la salida para contener las propiedades del mensaje; el descriptor de mensaje asociado al identificador de mensaje se utiliza para los campos de entrada.

Si se especifica un descriptor de mensaje en la llamada MQGET, siempre tiene prioridad sobre el descriptor de mensaje asociado a un identificador de mensaje.

Si se especifica GMPRRF, o se especifica GMPRAQ y el atributo de cola PRPCTL es PRPRFH, la llamada falla con el código de motivo RC2026 cuando no se especifica ningún parámetro descriptor de mensaje.

Al regresar de la llamada MQGET, las propiedades y el descriptor de mensajes asociados con este identificador de mensajes se actualizan para reflejar el estado del mensaje recuperado (así como el descriptor de mensajes, si se proporcionó uno en la llamada MQGET). Las propiedades del mensaje se pueden consultar utilizando la llamada MQINQMP.

A excepción de las extensiones del descriptor de mensajes, cuando están presentes, una propiedad que se puede consultar con la llamada MQINQMP no está contenida en los datos del mensaje; si el mensaje de la cola contenía propiedades en los datos del mensaje, estas se eliminan de los datos del mensaje antes de que los datos se devuelvan a la aplicación.

Si no se proporciona ningún identificador de mensaje o la versión es inferior a GMVER4, debe proporcionar un descriptor de mensaje válido en la llamada MQGET. Las propiedades del mensaje (excepto aquellas contenidas en el descriptor del mensaje) se devuelven en los datos del mensaje, según el valor de las opciones de propiedad en la estructura MQGMO y el atributo de cola PRPCTL.

Este campo es siempre un campo de entrada. El valor inicial de este campo es HMNONE. Este campo se ignora si *GMVER* es menor que GMVER4.

GMMO (entero con signo de 10 dígitos)

Opciones que controlan los criterios de selección utilizados para MQGET.

Estas opciones permiten a la aplicación elegir qué campos del **MSGDSC** parámetro se utilizan para seleccionar el mensaje devuelto por la llamada MQGET. La aplicación establece las opciones requeridas en este campo y, a continuación, establece los campos correspondientes en el **MSGDSC** parámetro con los valores requeridos para esos campos. Solo los mensajes que tienen esos valores en el MQMD para el mensaje son candidatos para la recuperación utilizando ese **MSGDSC** parámetro en la llamada MQGET. Los campos para los que no se especifica la opción de coincidencia correspondiente se ignoran al seleccionar el mensaje que se va a devolver. Si no se van a utilizar criterios de selección en la llamada MQGET (es decir, se acepta cualquier mensaje), *GMMO* debe establecerse en MONONE.

Si se especifica GMLOGO, solo determinados mensajes podrán ser devueltos por la siguiente llamada MQGET:

- Si no hay ningún grupo o mensaje lógico actual, solo los mensajes que sean *MDSEQ* iguales a 1 e *MDOFF* iguales a 0 podrán ser devueltos. En esta situación, se puede utilizar una o varias de las siguientes opciones para seleccionar cuál de los mensajes elegibles es el que se devuelve:
 - MOMSGI
 - MOCORI
 - MOGRPI
- Si hay un mensaje actual de grupo o lógico, solo el siguiente mensaje del grupo o el siguiente segmento del mensaje lógico es válido para su devolución, y esto no se puede modificar especificando opciones MO*.

En ambos casos, se pueden especificar opciones de coincidencia que no sean aplicables, pero el valor del campo relevante en el **MSGDSC** parámetro debe coincidir con el valor del campo correspondiente en el mensaje que se va a devolver; la llamada falla con el código de motivo RC2247 si no se cumple esta condición.

GMMO se ignora si se especifica GMMUC o GMBRWC.

Se puede especificar una o varias de las siguientes opciones:

MOMSGI

Recuperar el mensaje con el identificador de mensaje especificado.

Esta opción especifica que el mensaje que se va a recuperar debe tener un identificador de mensaje que coincida con el valor del *MDMID* campo en el **MSGDSC** parámetro de la llamada

MQGET. Esta coincidencia se suma a cualquier otra coincidencia que pueda aplicarse (por ejemplo, el identificador de correlación).

Si no se especifica esta opción, se ignora el *MDMID* campo del **MSGDSC** parámetro y se aceptan todos los identificadores de mensaje.

Nota: El identificador de mensaje MINONE es un valor especial que coincide con cualquier identificador de mensaje en el MQMD para el mensaje. Por lo tanto, especificar MOMSGI con MINONE es lo mismo que no especificar MOMSGI.

MOCORI

Recuperar mensaje con el identificador de correlación especificado.

Esta opción especifica que el mensaje que se va a recuperar debe tener un identificador de correlación que coincida con el valor del *MDCID* campo en el **MSGDSC** parámetro de la llamada MQGET. Esta coincidencia se suma a cualquier otra coincidencia que pueda aplicarse (por ejemplo, el identificador del mensaje).

Si no se especifica esta opción, se ignora el *MDCID* campo del **MSGDSC** parámetro y se aceptan todos los identificadores de correlación.

Nota: El identificador de correlación CINONE es un valor especial que coincide con cualquier identificador de correlación en el MQMD para el mensaje. Por lo tanto, especificar MOCORI con CINONE es lo mismo que no especificar MOCORI.

MOGRPI

Recuperar mensaje con identificador de grupo especificado.

Esta opción especifica que el mensaje que se va a recuperar debe tener un identificador de grupo que coincida con el valor del *MDGID* campo en el **MSGDSC** parámetro de la llamada MQGET. Esta coincidencia se suma a cualquier otra coincidencia que pueda aplicarse (por ejemplo, el identificador de correlación).

Si no se especifica esta opción, se ignora el *MDGID* campo del **MSGDSC** parámetro y se aceptan todos los identificadores de grupo.

Nota: El identificador de grupo GINONE es un valor especial que coincide con cualquier identificador de grupo en el MQMD para el mensaje. Por lo tanto, especificar MOGRPI con GINONE es lo mismo que no especificar MOGRPI.

MOSEQN

Recuperar el mensaje con el número de secuencia especificado.

Esta opción especifica que el mensaje que se va a recuperar debe tener un número de secuencia de mensajes que coincida con el valor del *MDSEQ* campo en el **MSGDSC** parámetro de la llamada MQGET. Esta coincidencia se suma a cualquier otra coincidencia que pueda aplicarse (por ejemplo, el identificador de grupo).

Si no se especifica esta opción, se ignora el *MDSEQ* campo del **MSGDSC** parámetro y se acepta cualquier número de secuencia de mensajes.

MOOFFS

Recuperar mensaje con desplazamiento especificado.

Esta opción especifica que el mensaje que se va a recuperar debe tener un desplazamiento que coincida con el valor del *MDOFF* campo en el **MSGDSC** parámetro de la llamada MQGET. Esta coincidencia se suma a cualquier otra coincidencia que pueda aplicarse (por ejemplo, el número de secuencia del mensaje).

Si no se especifica esta opción, se ignora el *MDOFF* campo del **MSGDSC** parámetro y se acepta cualquier desplazamiento.

Si no se especifica ninguna de las opciones descritas, se puede utilizar la siguiente opción:

MONONE

No hay coincidencias.

Esta opción específica que no se utilizarán coincidencias en la selección del mensaje que se devolverá; por lo tanto, todos los mensajes de la cola son elegibles para su recuperación (pero están sujetos al control de las opciones GMAMSA, GMASGA y GMCMPM).

MONONE se define para ayudar en la documentación del programa. No está previsto que esta opción se utilice con ninguna otra opción MO*, pero como su valor es cero, dicho uso no puede detectarse.

Este campo es un campo de entrada. El valor inicial de este campo es MOMSGI con MOCORI. Este campo se ignora si *GMVER* es menor que *GMVER2*.

Nota: El valor inicial del campo *GMMO* se define para garantizar la compatibilidad con los gestores de colas de versiones anteriores. Sin embargo, al leer una serie de mensajes de una cola sin utilizar criterios de selección, este valor inicial requiere que la aplicación restablezca los campos *MDMID* *MDCID* y a *MINONE* y *CINONE* antes de cada llamada MQGET. La necesidad de restablecer *MDMID* y *MDCID* puede evitarse configurando *GMVER* en *GMVER2* y *GMMO* en *MONONE*.

GMOPT (entero con signo de 10 dígitos)

Opciones que controlan la acción de MQGET.

Se pueden especificar cero o más de las opciones descritas a continuación. Si se requiere más de uno, los valores se pueden sumar (no añada la misma constante más de una vez). Las combinaciones de opciones que no son válidas se indican; todas las demás combinaciones son válidas.

Opciones de espera : Las siguientes opciones se refieren a la espera de mensajes que llegan a la cola:

GMWT

Espera a que llegue el mensaje.

La aplicación debe esperar hasta que llegue un mensaje adecuado. El tiempo máximo que espera la aplicación se especifica en *GMWT*.

Si las solicitudes MQGET se inhiben, o si las solicitudes MQGET se inhiben mientras esperan, la espera se cancela y la llamada se completa con CCFAIL y el código de motivo RC2016, independientemente de si hay mensajes adecuados en la cola.

Esta opción se puede utilizar con las opciones GMBRWF o GMBRWN.

Si varias aplicaciones están esperando en la misma cola compartida, la aplicación o aplicaciones que se activan cuando llega un mensaje adecuado se describen más adelante en esta sección.

Nota: En la siguiente descripción, una llamada MQGET de exploración es aquella que especifica una de las opciones de exploración, pero no GMLK; una llamada MQGET que especifica la opción GMLK se trata como una llamada sin exploración.

- Si hay una o más llamadas MQGET sin exploración en espera, pero no hay llamadas MQGET con exploración en espera, se activa una.
- Si hay una o más llamadas MQGET de exploración en espera, pero no hay llamadas MQGET que no sean de exploración en espera, se activan todas.
- Si hay una o más llamadas MQGET sin exploración y una o más llamadas MQGET con exploración en espera, se activa una llamada MQGET sin exploración y ninguna, algunas o todas las llamadas MQGET con exploración. (El número de llamadas MQGET de exploración activadas no se puede predecir, ya que depende de las consideraciones de programación del sistema operativo y otros factores)

Si hay más de una llamada MQGET sin exploración en espera en la misma cola, solo se activa una; en esta situación, el gestor de colas intenta dar prioridad a las llamadas sin exploración en espera en el siguiente orden:

1. Solicitudes específicas de obtención-espera que solo pueden satisfacerse con determinados mensajes, por ejemplo, aquellos con un *MDMIDMDCID* específico (o ambos).
2. Solicitudes generales de espera que pueden satisfacerse con cualquier mensaje.

Se deben tener en cuenta los siguientes puntos:

- Dentro de la primera categoría, no se da prioridad adicional a las solicitudes get-wait más específicas, por ejemplo, aquellas que especifican tanto *MDMID* como *MDCID*.
- Dentro de cualquiera de las dos categorías, no se puede predecir qué aplicación se seleccionará. En particular, la solicitud que lleva más tiempo en espera no es necesariamente la seleccionada.
- La longitud de la ruta y las consideraciones de programación de prioridades del sistema operativo pueden hacer que una aplicación en espera con una prioridad del sistema operativo inferior a la esperada recupere el mensaje.
- También podría ocurrir que una aplicación que no está en espera recupere el mensaje antes que otra que sí lo está.

GMWT se ignora si se especifica con GMBRWC o GMMUC; no se genera ningún error.

GMNWT

Regrese inmediatamente si no hay ningún mensaje adecuado.

La aplicación no debe esperar si no hay ningún mensaje adecuado disponible. Es lo contrario de la opción GMWT y se define para facilitar la documentación del programa. Es el valor predeterminado si no se especifica ninguno.

GMFIQ

Falla si el gestor de colas está inactivo.

Esta opción fuerza el fallo de la llamada MQGET si el gestor de colas se encuentra en estado de inactividad.

Si esta opción se especifica junto con GMWT y la espera está pendiente en el momento en que el gestor de colas entra en estado de inactividad:

- La espera se cancela y la llamada devuelve el código de finalización CCFAIL con el código de motivo RC2161 .

Si no se especifica GMFIQ y el gestor de colas entra en estado de inactividad, la espera no se cancela.

Opciones de Syncpoint : Las siguientes opciones se refieren a la participación de la llamada MQGET dentro de una unidad de trabajo:

GMSYP

Obtener mensaje con control de punto de sincronización.

La solicitud es operar dentro de los protocolos normales de unidad de trabajo. El mensaje se marca como no disponible para otras aplicaciones, pero solo se elimina de la cola cuando se confirma la unidad de trabajo. El mensaje vuelve a estar disponible si se cancela la unidad de trabajo.

Si no se especifica esta opción o GMNSYP, la solicitud GET no se encuentra dentro de una unidad de trabajo.

Esta opción no es válida con ninguna de las siguientes opciones:

- GMBRWF
- GMBRWC
- GMBRWN
- GMLK
- GMNSYP
- GMPSYP
- GMUNLK

GMPSYP

Obtener mensaje con control de punto de sincronización si el mensaje es persistente.

La solicitud es operar dentro de los protocolos normales de unidad de trabajo, pero solo si el mensaje recuperado es persistente. Un mensaje persistente tiene el valor PEPER en el *MDPER* campo de MQMD.

- Si el mensaje es persistente, el gestor de colas procesa la llamada como si la aplicación hubiera especificado GMSYP.
- Si el mensaje no es persistente, el gestor de colas procesa la llamada como si la aplicación hubiera especificado GMNSYP (consulte la siguiente sección para obtener más detalles).

Esta opción no es válida con ninguna de las siguientes opciones:

- GMBRWF
- GMBRWC
- GMBRWN
- GMCMPM
- GMNSYP
- GMSYP
- GMUNLK

GMNSYP

Recibir mensajes sin control de puntos de sincronización.

La solicitud es operar fuera de los protocolos normales de unidad de trabajo. El mensaje se elimina inmediatamente de la cola (a menos que se trate de una solicitud de exploración). El mensaje no puede volver a estar disponible al retroceder la unidad de trabajo.

Esta opción se asume si se especifica GMBRWF o GMBRWN.

Si no se especifican esta opción y GMSYP, la solicitud GET no se encuentra dentro de una unidad de trabajo.

Esta opción no es válida con ninguna de las siguientes opciones:

- GMSYP
- GMPSYP

Opciones de exploración : Las siguientes opciones se refieren a la exploración de mensajes en la cola:

GMBRWF

Navegar desde el inicio de la cola.

Cuando se abre una cola con la opción OOBROW, se establece un cursor de exploración, situado lógicamente antes del primer mensaje de la cola. Las llamadas MQGET posteriores que especifiquen la opción GMBRWF, GMBRWN o GMBRWC se pueden utilizar para recuperar mensajes de la cola de forma no destructiva. El cursor de exploración marca la posición, dentro de los mensajes de la cola, desde la que la siguiente llamada MQGET con GMBRWN busca un mensaje adecuado.

Una llamada MQGET con GMBRWF hace que se ignore la posición anterior del cursor de exploración. Se recupera el primer mensaje de la cola que cumple las condiciones especificadas en el descriptor de mensajes. El mensaje permanece en la cola y el cursor de exploración se coloca sobre este mensaje.

Después de esta llamada, el cursor de navegación se coloca en el mensaje que se ha devuelto. Si el mensaje se elimina de la cola antes de que se emita la siguiente llamada MQGET con GMBRWN, el cursor de exploración permanece en la posición que ocupaba el mensaje en la cola, aunque esa posición ahora esté vacía.

La opción GMMUC se puede utilizar con una llamada MQGET sin exploración, si es necesario, para eliminar el mensaje de la cola.

El cursor de exploración no se mueve mediante una llamada MQGET que no sea de exploración utilizando el mismo *HOBJ* identificador. Tampoco se mueve mediante una llamada MQGET de exploración que devuelve un código de finalización CCFAIL o un código de motivo RC2080 .

La opción GMLK se puede especificar junto con esta opción para bloquear el mensaje que se está explorando.

GMBRWF se puede especificar con cualquier combinación válida de las opciones GM* y MO* que controlan el procesamiento de mensajes en grupos y segmentos de mensajes lógicos.

Si se especifica GMLOGO, los mensajes se examinan en orden lógico. Si se omite esa opción, los mensajes se examinan en orden físico. Cuando se especifica GMBRWF, es posible cambiar entre el orden lógico y el orden físico, pero las llamadas MQGET posteriores que utilicen GMBRWN deben explorar la cola en el mismo orden que la llamada más reciente que especificó GMBRWF para el identificador de cola.

La información de grupo y segmento que el gestor de colas conserva para las llamadas MQGET que examinan los mensajes de la cola es independiente de la información de grupo y segmento que el gestor de colas conserva para las llamadas MQGET que eliminan mensajes de la cola. Cuando se especifica GMBRWF, el gestor de colas ignora la información del grupo y del segmento para la exploración, y escanea la cola como si no hubiera ningún grupo actual ni ningún mensaje lógico actual. Si la llamada MQGET se realiza correctamente (código de finalización CCOK o CCWARN), la información del grupo y del segmento para la exploración se establece en la del mensaje devuelto; si la llamada falla, la información del grupo y del segmento permanece igual que antes de la llamada.

Esta opción no es válida con ninguna de las siguientes opciones:

- GMBRWC
- GMBRWN
- GMMUC
- GMSYP
- GMPSYP
- GMUNLK

También es un error si la cola no se abrió para su exploración.

GMBRWN

Navegar desde la posición actual en la cola.

El cursor de exploración avanza al siguiente mensaje de la cola que cumple los criterios de selección especificados en la llamada MQGET. El mensaje se devuelve a la aplicación, pero permanece en la cola.

Una vez que se ha abierto una cola para su exploración, la primera llamada de exploración que utiliza el identificador tiene el mismo efecto, independientemente de si especifica la opción GMBRWF o GMBRWN.

Si el mensaje se elimina de la cola antes de que se emita la siguiente llamada MQGET con GMBRWN, el cursor de exploración permanece lógicamente en la posición que ocupaba el mensaje en la cola, aunque esa posición ahora esté vacía.

Los mensajes se almacenan en la cola de dos maneras:

- FIFO dentro de la prioridad (MSPRIO), o
- FIFO independientemente de la prioridad (MSFIFO)

El **MsgDeliverySequence** atributo que indica qué método se aplica (véase [“Atributos para colas”](#) en la página 1399 para más detalles).

Si la cola tiene un *MsgDeliverySequence* de MSPRIO y llega un mensaje a la cola que tiene una prioridad mayor que la que señala actualmente el cursor de exploración, ese mensaje no se

encuentra durante el barrido actual de la cola con GMBRWN. Solo se puede encontrar después de restablecer el cursor de exploración con GMBRWF (o volviendo a abrir la cola).

La opción GMMUC se puede utilizar posteriormente con una llamada MQGET sin exploración, si es necesario, para eliminar el mensaje de la cola.

El cursor de exploración no se mueve mediante llamadas MQGET que no sean de exploración y que utilicen el mismo *HOB*J identificador.

La opción GMLK se puede especificar junto con esta opción para bloquear el mensaje que se está explorando.

GMBRWN se puede especificar con cualquier combinación válida de las opciones GM* y MO* que controlan el procesamiento de mensajes en grupos y segmentos de mensajes lógicos.

Si se especifica GMLOGO, los mensajes se examinan en orden lógico. Si se omite esa opción, los mensajes se examinan en orden físico. Cuando se especifica GMBRWF, es posible cambiar entre el orden lógico y el orden físico, pero las llamadas MQGET posteriores que utilicen GMBRWN deben explorar la cola en el mismo orden que la llamada más reciente que especificó GMBRWF para el identificador de cola. La llamada falla con el código de motivo RC2259 si no se cumple esta condición.

Nota: Se debe tener especial cuidado si se utiliza una llamada MQGET para navegar más allá del final de un grupo de mensajes (o mensaje lógico que no está en un grupo) cuando no se especifica GMLOGO. Por ejemplo, si el último mensaje del grupo precede al primer mensaje del grupo en la cola, al utilizar GMBRWN para navegar más allá del final del grupo, especificando MOSEQN con *MDSEQ* establecido en 1 (para encontrar el primer mensaje del siguiente grupo), se volvería a mostrar el primer mensaje del grupo ya explorado. Esto podría ocurrir de forma inmediata, o después de varias llamadas MQGET (si hay grupos intermedios).

La posibilidad de un bucle infinito se puede evitar abriendo la cola dos veces para navegar:

- Utilice el primer descriptor para examinar solamente el primer mensaje de cada grupo.
- Utilice el segundo descriptor para examinar solamente los mensajes de un grupo específico.
- Utilice las opciones MO* para mover el segundo cursor de exploración a la posición del primer cursor de exploración, antes de explorar los mensajes del grupo.
- No utilice GMBRWN para navegar más allá del final de un grupo.

La información de grupo y segmento que el gestor de colas conserva para las llamadas MQGET que examinan los mensajes de la cola es independiente de la información de grupo y segmento que conserva para las llamadas MQGET que eliminan mensajes de la cola.

Esta opción no es válida con ninguna de las siguientes opciones:

- GMBRWF
- GMBRWC
- GMMUC
- GMSYP
- GMPSYP
- GMUNLK

También es un error si la cola no se abrió para su exploración.

GMBRWC

Buscar mensaje bajo el cursor de búsqueda.

Esta opción hace que el mensaje al que apunta el cursor de exploración se recupere de forma no destructiva, independientemente de las opciones MO* especificadas en el *GMMO* campo de MQGMO.

El mensaje señalado por el cursor de navegación es el último que se recuperó utilizando la opción GMBRWF o GMBRWN. La llamada falla si ninguna de estas llamadas se ha emitido para esta cola

desde que se abrió, o si el mensaje que estaba bajo el cursor de exploración se ha recuperado de forma destructiva desde entonces.

La posición del cursor de exploración no cambia con esta llamada.

La opción GMMUC se puede utilizar con una llamada MQGET sin exploración, si es necesario, para eliminar el mensaje de la cola.

El cursor de exploración no se mueve mediante una llamada MQGET que no sea de exploración utilizando el mismo *HOBJ* identificador. Tampoco se mueve por una llamada MQGET de exploración que devuelve un código de finalización CCFAIL o un código de motivo « RC2080 » (no se ha encontrado).

Si se especifica GMBRWC con GMLK:

- Si ya hay un mensaje bloqueado, debe ser el que se encuentra debajo del cursor, por lo que se devuelve sin desbloquearlo ni volver a bloquearlo; el mensaje permanece bloqueado.
- Si no hay ningún mensaje bloqueado, el mensaje situado debajo del cursor de exploración (si lo hay) se bloquea y se devuelve a la aplicación; si no hay ningún mensaje debajo del cursor de exploración, la llamada falla.

Si se especifica GMBRWC sin GMLK:

- Si ya hay un mensaje bloqueado, debe ser el que se encuentra debajo del cursor. Este mensaje se devuelve a la aplicación y, a continuación, se desbloquea. Dado que el mensaje ahora está desbloqueado, no hay garantía de que se pueda volver a consultar o recuperar de forma destructiva (otra aplicación podría recuperarlo de forma destructiva al obtener mensajes de la cola).
- Si no hay ningún mensaje bloqueado, el mensaje situado debajo del cursor de exploración (si lo hay) se devuelve a la aplicación; si no hay ningún mensaje debajo del cursor de exploración, la llamada falla.

Si se especifica GMCMPM con GMBRWC, el cursor de exploración debe identificar un mensaje con un *MDOFF* campo en MQMD que sea cero. Si no se cumple esta condición, la llamada falla con el código de motivo RC2246 .

La información de grupo y segmento que el gestor de colas conserva para las llamadas MQGET que examinan los mensajes de la cola es independiente de la información de grupo y segmento que conserva para las llamadas MQGET que eliminan mensajes de la cola.

Esta opción no es válida con ninguna de las siguientes opciones:

- GMBRWF
- GMBRWN
- GMMUC
- GMSYP
- GMPSYP
- GMUNLK

También es un error si la cola no se abrió para su exploración.

GMMUC

Obtener mensaje bajo el cursor de exploración.

Esta opción hace que se recupere el mensaje al que apunta el cursor de exploración, independientemente de las opciones MO* especificadas en el *GMMO* campo de MQGMO. El mensaje se elimina de la cola.

El mensaje señalado por el cursor de navegación es el último que se recuperó utilizando la opción GMBRWF o GMBRWN.

Si se especifica GMCMPM con GMMUC, el cursor de exploración debe identificar un mensaje con un *MDOFF* campo en MQMD que sea cero. Si no se cumple esta condición, la llamada falla con el código de motivo RC2246 .

Esta opción no es válida con ninguna de las siguientes opciones:

- GMBRWF
- GMBRWC
- GMBRWN
- GMUNLK

También es un error si la cola no se abrió tanto para la exploración como para la entrada. Si el cursor de exploración no está apuntando actualmente a un mensaje recuperable, la llamada MQGET devuelve un error.

Opciones de bloqueo : Las siguientes opciones se refieren al bloqueo de mensajes en la cola:

GMLK

Bloquear mensaje.

Esta opción bloquea el mensaje que se está explorando, de modo que el mensaje se vuelve invisible para cualquier otro identificador abierto para la cola. La opción solo se puede especificar si también se especifica una de las siguientes opciones:

- GMBRWF
- GMBRWN
- GMBRWC

Solo se puede bloquear un mensaje por cada identificador de cola, pero puede ser un mensaje lógico o un mensaje físico:

- Si se especifica GMCMPM, todos los segmentos de mensaje que componen el mensaje lógico se bloquean en el identificador de cola (si están todos presentes en la cola y disponibles para su recuperación).
- Si no se especifica GMCMPM, solo se bloquea un único mensaje físico en el identificador de cola. Si este mensaje es un segmento de un mensaje lógico, el segmento bloqueado impide que otras aplicaciones que utilizan GMCMPM recuperen o examinen el mensaje lógico.

El mensaje bloqueado es siempre el que se encuentra debajo del cursor de exploración, y el mensaje se puede eliminar de la cola mediante una llamada MQGET posterior que especifique la opción GMMUC. Otras llamadas MQGET que utilizan el identificador de cola también pueden eliminar el mensaje (por ejemplo, una llamada que especifica el identificador del mensaje bloqueado).

Si la llamada devuelve el código de finalización CCFAIL o CCWARN con el código de motivo RC2080, no se bloquea ningún mensaje.

Si la aplicación decide no eliminar el mensaje de la cola, el bloqueo se libera mediante:

- Emisión de otra llamada MQGET para este identificador, especificando GMBRWF o GMBRWN (con o sin GMLK); el mensaje se desbloquea si la llamada se completa con CCOK o CCWARN, pero permanece bloqueado si la llamada se completa con CCFAIL. No obstante, se aplican las excepciones siguientes:
 - El mensaje no se desbloquea si CCWARN se devuelve con un RC2080 (Aviso de desbloqueo).
 - El mensaje se desbloquea si se devuelve CCFAIL con un RC2033 (Error de control de flujo).

Si también se especifica GMLK, el mensaje devuelto se bloquea. Si no se especifica GMLK, no hay ningún mensaje bloqueado después de la llamada.

Si se especifica GMWT y no hay ningún mensaje disponible inmediatamente, el desbloqueo del mensaje original se produce antes del inicio de la espera (siempre que la llamada no presente ningún error).

- Emisión de otra llamada MQGET para este identificador, con GMBRWC (sin GMLK); el mensaje se desbloquea si la llamada se completa con CCOK o CCWARN, pero permanece bloqueado si la llamada se completa con CCFAIL. Sin embargo, se aplica la siguiente excepción:
 - El mensaje no se desbloquea si CCWARN se devuelve con un RC2080 (Aviso de desbloqueo).
- Emisión de otra llamada MQGET para este identificador con GMUNLK.
- Emitir una llamada MQCLOSE para este identificador (ya sea explícitamente o implícitamente al finalizar la aplicación).

No se requiere ninguna opción especial abierta para especificar esta opción, salvo OOBROW, que es necesaria para especificar la opción de exploración correspondiente.

Esta opción no es válida con ninguna de las siguientes opciones:

- GMSYP
- GMPSYP
- GMUNLK

GMUNLK

Desbloquear mensaje.

El mensaje que se va a desbloquear debe haber sido bloqueado previamente mediante una llamada MQGET con la opción GMLK. Si no hay ningún mensaje bloqueado para este identificador, la llamada finaliza con CCWARN y RC2209 .

Los parámetros **MSGDSC**, **BUFLEN**, **BUFFERDATLEN** y no se comprueban ni se modifican si se especifica GMUNLK. No se devuelve ningún mensaje en *BUFFER*.

No se requiere ninguna opción especial abierta para especificar esta opción (aunque se necesita OOBROW para emitir la solicitud de bloqueo en primer lugar).

Esta opción no es válida con ninguna otra opción, excepto las siguientes:

- GMNWT
- GMNSYP

Ambas opciones se asumen independientemente de si se especifican o no.

Opciones de datos del mensaje : Las siguientes opciones se refieren al procesamiento de los datos del mensaje cuando este se lee desde la cola:

GMATM

Permitir el truncamiento de los datos del mensaje.

Si el búfer de mensajes es demasiado pequeño para contener el mensaje completo, esta opción permite que la llamada MQGET llene el búfer con todo el mensaje que pueda contener, emita un código de finalización de advertencia y complete su procesamiento. Esto implica lo siguiente:

- Al navegar por los mensajes, el cursor de navegación avanza hasta el mensaje devuelto.
- Al eliminar mensajes, el mensaje devuelto se elimina de la cola.
- Se devuelve RC2079 el código de motivo si no se produce ningún otro error.

Sin esta opción, el búfer se sigue llenando con todo el mensaje que puede contener, se emite un código de finalización de advertencia, pero el procesamiento no se completa. Esto implica lo siguiente:

- Al navegar por los mensajes, el cursor de navegación no avanza.
- Al eliminar mensajes, el mensaje no se elimina de la cola.
- Se devuelve RC2080 el código de motivo si no se produce ningún otro error.

GMCONV

Convertir datos de mensajes.

Esta opción solicita que los datos de la aplicación en el mensaje se conviertan para ajustarse a los *MDCSI* valores *MDENC* y especificados en el **MSGDSC** parámetro de la llamada MQGET, antes de que los datos se copien en el **BUFFER** parámetro.

El *MDFMT* campo especificado cuando se envió el mensaje es asumido por el proceso de conversión para identificar la naturaleza de los datos del mensaje. La conversión de los datos del mensaje la realiza el gestor de colas para los formatos integrados y una salida escrita por el usuario para otros formatos.

- Si la conversión se realiza correctamente, los campos *MDCSI MDENC* y especificados en el **MSGDSC** parámetro no se modifican al regresar de la llamada MQGET.
- Si la conversión no se puede realizar correctamente (pero la llamada MQGET se completa sin errores), los datos del mensaje se devuelven sin convertir y los campos *MDCSI MDENC* y en *MSGDSC* se establecen en los valores del mensaje sin convertir. El código de finalización en este caso es CCWARN.

En cualquier caso, por lo tanto, estos campos describen el identificador del conjunto de caracteres y la codificación de los datos del mensaje que se devuelven en el **BUFFER** parámetro.

Consulte el *MDFMT* campo descrito en “MQMD (descriptor de mensaje) en IBM i” en la [página 1131](#) para obtener una lista de los nombres de formato para los que el gestor de colas realiza la conversión.

Opciones de grupo y segmento : Las siguientes opciones se refieren al procesamiento de mensajes en grupos y segmentos de mensajes lógicos. Estas definiciones pueden ser útiles para comprender las opciones:

Mensaje físico

Es la unidad más pequeña de información que se puede colocar o eliminar de una cola; a menudo corresponde a la información especificada o recuperada en una sola llamada MQPUT, MQPUT1 o MQGET. Cada mensaje físico tiene su propio descriptor de mensaje (MQMD). Por lo general, los mensajes físicos se distinguen por valores diferentes para el identificador de mensaje (*MDMID* campo en MQMD), aunque esto no lo impone el gestor de colas.

Mensaje lógico

Esta es una sola unidad de información de la solicitud. En ausencia de restricciones del sistema, un mensaje lógico sería igual que un mensaje físico. Sin embargo, cuando los mensajes lógicos son grandes, las limitaciones del sistema pueden hacer que sea aconsejable o necesario dividir un mensaje lógico en dos o más mensajes físicos, denominados segmentos.

Un mensaje lógico que ha sido segmentado consta de dos o más mensajes físicos que tienen el mismo identificador de grupo no nulo (*MDGID* campo en MQMD) y el mismo número de secuencia de mensajes (*MDSEQ* campo en MQMD). Los segmentos se distinguen por valores diferentes para el desplazamiento del segmento (*MDOFF* campo en MQMD), que proporciona el desplazamiento de los datos en el mensaje físico desde el inicio de los datos en el mensaje lógico. Dado que cada segmento es un mensaje físico, los segmentos de un mensaje lógico suelen tener identificadores de mensaje diferentes.

Un mensaje lógico que no ha sido segmentado, pero cuya segmentación ha sido permitida por la aplicación emisora, también tiene un identificador de grupo distinto de nulo, aunque en este caso solo hay un mensaje físico con ese identificador de grupo si el mensaje lógico no pertenece a un grupo de mensajes. Los mensajes lógicos cuya segmentación ha sido inhibida por la aplicación emisora tienen un identificador de grupo nulo (GINONE), a menos que el mensaje lógico pertenezca a un grupo de mensajes.

Grupo de mensajes

Se trata de un conjunto de uno o más mensajes lógicos que tienen el mismo identificador de grupo no nulo. Los mensajes lógicos del grupo se distinguen por valores diferentes del número de secuencia del mensaje, que es un número entero comprendido entre 1 y n, donde n es el número de mensajes lógicos del grupo. Si uno o más de los mensajes lógicos están segmentados, hay más de n mensajes físicos en el grupo.

GMLOGO

Los mensajes en grupos y segmentos de mensajes lógicos se devuelven en orden lógico.

Esta opción controla el orden en que se devuelven los mensajes mediante llamadas MQGET sucesivas para el identificador de cola. La opción debe especificarse en cada una de esas llamadas para que surta efecto.

Si se especifica GMLOGO para llamadas MQGET sucesivas para el identificador de cola, los mensajes en grupos se devuelven en el orden indicado por sus números de secuencia de mensajes, y los segmentos de mensajes lógicos se devuelven en el orden indicado por sus desplazamientos de segmento. Este orden puede diferir del orden en que esos mensajes y segmentos aparecen en la cola.

Nota: Especificar GMLOGO no tiene consecuencias adversas en los mensajes que no pertenecen a grupos y que no son segmentos. En efecto, dichos mensajes se tratan como si cada uno perteneciera a un grupo de mensajes compuesto por un solo mensaje. Por lo tanto, es perfectamente seguro especificar GMLOGO al recuperar mensajes de colas que pueden contener una mezcla de mensajes en grupos, segmentos de mensajes y mensajes no segmentados que no están en grupos.

Para devolver los mensajes en el orden requerido, el gestor de colas conserva la información del grupo y del segmento entre llamadas MQGET sucesivas. Esta información identifica el grupo de mensajes actual y el mensaje lógico actual para el identificador de cola, la posición actual dentro del grupo y el mensaje lógico, y si los mensajes se están recuperando dentro de una unidad de trabajo. Dado que el gestor de colas conserva esta información, la aplicación no necesita establecer la información del grupo y del segmento antes de cada llamada MQGET. En concreto, esto significa que la aplicación no necesita configurar los campos *MDGID*, *MDSEQMDOFF* y en *MQMD*. Sin embargo, la aplicación debe configurar correctamente la opción *GMSYP* o *GMNSYP* en cada llamada.

Cuando se abre la cola, no hay ningún grupo de mensajes actual ni ningún mensaje lógico actual. Un grupo de mensajes se convierte en el grupo de mensajes actual cuando la llamada MQGET devuelve un mensaje que tiene el indicador *MFMIG*. Con GMLOGO especificado en llamadas sucesivas, ese grupo permanece como el grupo actual hasta que se devuelve un mensaje que tiene:

- *MFLMIG* sin *MFSEG* (es decir, el último mensaje lógico del grupo no está segmentado), o
- *MFLMIG* con *MFLSEG* (es decir, el mensaje devuelto es el último segmento del último mensaje lógico del grupo).

Cuando se devuelve un mensaje de este tipo, el grupo de mensajes se termina y, una vez completada con éxito la llamada MQGET, ya no existe ningún grupo actual. De manera similar, un mensaje lógico se convierte en el mensaje lógico actual cuando la llamada MQGET devuelve un mensaje con el indicador *MFSEG*, y ese mensaje lógico finaliza cuando se devuelve el mensaje con el indicador *MFLSEG*.

Si no se especifican criterios de selección, las llamadas sucesivas a MQGET devuelven (en el orden correcto) los mensajes del primer grupo de mensajes de la cola, luego los mensajes del segundo grupo de mensajes, y así sucesivamente, hasta que no haya más mensajes disponibles. Es posible seleccionar los grupos de mensajes concretos que se devuelven especificando una o varias de las siguientes opciones en el *GMMO* campo:

- *MOMSGI*
- *MOCORI*
- *MOGRPI*

Sin embargo, estas opciones solo son efectivas cuando no hay ningún grupo de mensajes actual ni ningún mensaje lógico; consulte el *GMMO* campo descrito en este tema.

Tabla 741 en la página 1110 muestra los valores de los campos *MDMID*, *MDCID*, *MDGIDMDOFF*, *MDSEQ*, y que el gestor de colas busca cuando intenta encontrar un mensaje para devolver en

la llamada MQGET. Esto se aplica tanto a la eliminación de mensajes de la cola como a la exploración de mensajes en la cola. Las columnas de la tabla tienen los siguientes significados:

LOG ORD

Indica si se ha especificado la opción GMLOGO en la llamada.

Cur grp

Indica si existe un grupo de mensajes actual antes de la llamada.

Cur log msg

Indica si existe un mensaje lógico actual antes de la llamada.

Otras columnas

Mostrar los valores que busca el gestor de colas. «Anterior» indica el valor devuelto para el campo en el mensaje anterior para el identificador de cola.

Tabla 741. Opciones de MQGET relacionadas con mensajes en grupos y segmentos de mensajes lógicos							
Opcion es especificadas	Estado de grupo y mensaje lógico antes de la llamada		Valores que busca el gestor de colas				
	Cur grp	Cur log msg	MDMID	MDCID	MDGID	MDSEQ	MDOFF
Sí	No	No	Controlado por GMMO	Controlado por GMMO	Controlado por GMMO	1	0
Sí	No	Sí	Cualquier identificador de mensaje	Cualquier identificador de correlación	Identificador de grupo anterior	1	Desplazamient o anterior + longitud de segmento anterior
Sí	Sí	No	Cualquier identificador de mensaje	Cualquier identificador de correlación	Identificador de grupo anterior	Número de secuencia anterior + 1	0
Sí	Sí	Sí	Cualquier identificador de mensaje	Cualquier identificador de correlación	Identificador de grupo anterior	Número de secuencia anterior	Desplazamient o anterior + longitud de segmento anterior
No	Cualqui era de los dos	Cualqui era de los dos	Controlado por GMMO	Controlado por GMMO	Controlado por GMMO	Controlado por GMMO	Controlado por GMMO

Cuando hay varios grupos de mensajes en la cola y son aptos para su devolución, los grupos se devuelven en el orden determinado por la posición en la cola del primer segmento del primer mensaje lógico de cada grupo (es decir, los mensajes físicos que tienen números de secuencia de mensajes 1 y desplazamientos 0 determinan el orden en que se devuelven los grupos aptos).

La opción GMLOGO afecta a las unidades de trabajo de la siguiente manera:

- Si el primer mensaje lógico o segmento de un grupo se recupera dentro de una unidad de trabajo, todos los demás mensajes lógicos y segmentos del grupo deben recuperarse dentro de una unidad de trabajo, si se utiliza el mismo identificador de cola. Sin embargo, no es necesario recuperarlos dentro de la misma unidad de trabajo. Esto permite dividir un grupo de mensajes compuesto por muchos mensajes físicos en dos o más unidades de trabajo consecutivas para el controlador de cola.

- Si el primer mensaje lógico o segmento de un grupo no se recupera dentro de una unidad de trabajo, ninguno de los demás mensajes lógicos y segmentos del grupo podrá recuperarse dentro de una unidad de trabajo, si se utiliza el mismo identificador de cola.

Si no se cumplen estas condiciones, la llamada MQGET falla con el código de motivo RC2245 .

Cuando se especifica GMLOGO, el MQGMO proporcionado en la llamada MQGET no debe ser inferior a GMVER2, y el MQMD no debe ser inferior a MDVER2. Si no se cumple esta condición, la llamada falla con el código de motivo RC2256 o RC2257 , según corresponda.

Si no se especifica GMLOGO para llamadas MQGET sucesivas para el identificador de cola, los mensajes se devuelven sin tener en cuenta si pertenecen a grupos de mensajes o si son segmentos de mensajes lógicos. Esto significa que los mensajes o segmentos de un grupo concreto o mensaje lógico pueden devolverse desordenados, o pueden mezclarse con mensajes o segmentos de otros grupos o mensajes lógicos, o con mensajes que no pertenecen a ningún grupo ni son segmentos. En esta situación, los mensajes concretos que devuelven las llamadas sucesivas a MQGET están controlados por las opciones MO* especificadas en dichas llamadas (consulte el *GMMO* campo descrito en [“MQGMO \(Opciones de obtención de mensajes\) activado IBM i”](#) en la página 1096 para obtener más información sobre estas opciones).

Esta es la técnica que se puede utilizar para reiniciar un grupo de mensajes o un mensaje lógico a mitad de camino, después de que se haya producido un fallo del sistema. Cuando el sistema se reinicia, la aplicación puede establecer los campos *MDGID*, *MDSEQGMMO* , *MDOFF* y con los valores adecuados y, a continuación, emitir la llamada MQGET con *GMSYP* o *GMNSYP* establecidos según sea necesario, pero sin especificar GMLOGO. Si esta llamada tiene éxito, el gestor de colas conserva la información del grupo y del segmento, y las llamadas MQGET posteriores que utilicen ese identificador de cola pueden especificar GMLOGO con normalidad.

La información de grupo y segmento que el gestor de colas conserva para la llamada MQGET es independiente de la información de grupo y segmento que conserva para la llamada MQPUT. Además, el gestor de colas conserva información separada para:

- Llamadas MQGET que eliminan mensajes de la cola.
- Llamadas MQGET que examinan los mensajes de la cola.

Para cualquier identificador de cola dado, la aplicación es libre de mezclar llamadas MQGET que especifican GMLOGO con llamadas MQGET que no lo hacen, pero se deben tener en cuenta los siguientes puntos:

- Si no se especifica GMLOGO, cada llamada MQGET correcta hace que el gestor de colas establezca la información guardada del grupo y del segmento en los valores correspondientes al mensaje devuelto; esto sustituye la información existente del grupo y del segmento conservada por el gestor de colas para el identificador de cola. Solo se modifica la información adecuada a la acción de la llamada (navegar o eliminar).
- Si no se especifica GMLOGO, la llamada no falla si hay un grupo de mensajes actual o un mensaje lógico; sin embargo, la llamada podría tener éxito con un código de finalización CCWARN. La [Tabla 742 en la página 1112](#) muestra los distintos casos que se pueden presentar. En estos casos, si el código de finalización no es CCOK, el código de motivo es uno de los siguientes:
 - RC2241
 - RC2242
 - RC2245

Nota: El gestor de colas no comprueba la información del grupo y del segmento al examinar una cola, ni al cerrar una cola que se abrió para examinarla pero no para introducir datos; en esos casos, el código de finalización es siempre CCOK (suponiendo que no haya otros errores).

Tabla 742. Resultado cuando la llamada MQGET o MQCLOSE no es coherente con la información del grupo y del segmento

La llamada actual es	La llamada anterior fue MQGET con GMLOGO	La llamada anterior fue MQGET sin GMLOGO
MQGET con GMLOGO	CCFAIL	CCFAIL
MQGET sin GMLOGO	CCWARN	CCOK
MQCLOSE con un grupo o mensaje lógico sin terminar	CCWARN	CCOK

Se recomienda a las aplicaciones que solo desean recuperar mensajes y segmentos en orden lógico que especifiquen GMLOGO, ya que es la opción más sencilla de usar. Esta opción hace que la aplicación no tenga que gestionar la información de grupo y segmento, pues el gestor de colas lo hace en su lugar. Sin embargo, las aplicaciones especializadas pueden necesitar más control del que proporciona la opción GMLOGO, lo que se puede conseguir sin especificar dicha opción. Si se hace esto, la aplicación debe asegurarse de que los campos *MDMID*, *MDCID*, *MDGIDMDOFF*, *MDSEQ*, y en MQMD, y las opciones MO* en GMMO en MQGMO, estén configurados correctamente antes de cada llamada MQGET.

Por ejemplo, una aplicación que desee reenviar los mensajes físicos que recibe, independientemente de si dichos mensajes se encuentran en grupos o segmentos de mensajes lógicos, no debe especificar GMLOGO. Esto se debe a que, en una red compleja con múltiples rutas entre los gestores de colas de envío y recepción, los mensajes físicos pueden llegar desordenados. Al no especificar GMLOGO y el PMLOGO correspondiente en la llamada MQPUT, la aplicación de reenvío puede recuperar y reenviar cada mensaje físico tan pronto como llega, sin tener que esperar a que llegue el siguiente en orden lógico.

GMLOGO se puede especificar con cualquiera de las otras opciones GM* y con varias de las opciones MO* en las circunstancias adecuadas.

GMCMPPM

Solo se pueden recuperar mensajes lógicos completos.

Esta opción especifica que la llamada MQGET solo puede devolver un mensaje lógico completo. Si el mensaje lógico está segmentado, el gestor de colas vuelve a ensamblar los segmentos y devuelve el mensaje lógico completo a la aplicación; el hecho de que el mensaje lógico estuviera segmentado no es evidente para la aplicación que lo recupera.

Nota: Esta es la única opción que hace que el gestor de colas vuelva a ensamblar los segmentos de mensajes. Si no se especifica, los segmentos se devuelven individualmente a la aplicación si están presentes en la cola (y satisfacen los demás criterios de selección especificados en la llamada MQGET). Por lo tanto, las aplicaciones que no deseen recibir segmentos individuales deben especificar siempre GMCMPPM.

Para utilizar esta opción, la aplicación debe proporcionar un búfer lo suficientemente grande como para alojar el mensaje completo, o especificar la opción GMATM.

Si la cola contiene mensajes segmentados en los que faltan algunos segmentos (quizás porque se han retrasado en la red y aún no han llegado), especificar GMCMPPM impide la recuperación de segmentos que pertenecen a mensajes lógicos incompletos. Sin embargo, esos segmentos de mensaje siguen contribuyendo al valor del atributo **CurrentQDepth** queue; esto significa que puede que no haya mensajes lógicos recuperables, aunque *CurrentQDepth* sea mayor que cero.

En el caso de los mensajes persistentes, el gestor de colas solo puede reensamblar los segmentos dentro de una unidad de trabajo:

- Si la llamada MQGET se ejecuta dentro de una unidad de trabajo definida por el usuario, se utiliza esa unidad de trabajo. Si la llamada falla durante el proceso de reensamblaje, el gestor de colas restablece en la cola cualquier segmento que se haya eliminado durante el reensamblaje. Sin embargo, el fallo no impide que la unidad de trabajo se confirme correctamente.

- Si la llamada se realiza fuera de una unidad de trabajo definida por el usuario y no existe ninguna unidad de trabajo definida por el usuario, el gestor de colas crea una unidad de trabajo solo para la duración de la llamada. Si la llamada se realiza correctamente, el gestor de colas confirma automáticamente la unidad de trabajo (la aplicación no necesita hacerlo). Si la llamada falla, el gestor de colas revierte la unidad de trabajo.
- Si la llamada se ejecuta fuera de una unidad de trabajo definida por el usuario, pero existe una unidad de trabajo definida por el usuario, el gestor de colas no puede realizar el reensamblaje. Si el mensaje no requiere reensamblaje, la llamada aún puede tener éxito. Pero si el mensaje requiere reensamblaje, la llamada falla con el código de motivo RC2255 .

Para los mensajes no persistentes, el gestor de colas no requiere que haya una unidad de trabajo disponible para realizar el reensamblaje.

Cada mensaje físico que es un segmento tiene su propio descriptor de mensaje. Para los segmentos que constituyen un único mensaje lógico, la mayoría de los campos del descriptor del mensaje son iguales para todos los segmentos del mensaje lógico; normalmente, solo los campos *MDMID*, *MDOFF* *MDMFL* y difieren entre los segmentos del mensaje lógico. Sin embargo, si un segmento se coloca en una cola de mensajes no entregados en un gestor de colas intermedio, el controlador DLQ recupera el mensaje especificando la opción *GMCONV*, lo que puede provocar que se modifique el juego de caracteres o la codificación del segmento. Si el controlador DLQ envía correctamente el segmento, es posible que este tenga un juego de caracteres o una codificación diferentes a los de los demás segmentos del mensaje lógico cuando finalmente llegue al gestor de colas de destino.

Un mensaje lógico compuesto por segmentos en los que los campos *MDCSI*, *MDENC*, o ambos difieren no puede ser reensamblado por el gestor de colas en un único mensaje lógico. En su lugar, el gestor de colas vuelve a ensamblar y devuelve los primeros segmentos consecutivos al principio del mensaje lógico que tienen los mismos identificadores de conjunto de caracteres y codificaciones, y la llamada *MQGET* se completa con el código de finalización *CCWARN* y el código de motivo RC2243 o RC2244 , según corresponda. Esto ocurre independientemente de si se especifica *GMCONV*. Para recuperar los segmentos restantes, la aplicación debe volver a emitir la llamada *MQGET* sin la opción *GMCMPPM*, recuperando los segmentos uno por uno. *GMLOGO* se puede utilizar para recuperar los segmentos restantes en orden.

También es posible que una aplicación que coloca segmentos establezca otros campos en el descriptor del mensaje con valores que difieren entre segmentos. Sin embargo, no hay ninguna ventaja en hacerlo si la aplicación receptora utiliza *GMCMPPM* para recuperar el mensaje lógico. Cuando el gestor de colas vuelve a ensamblar un mensaje lógico, devuelve en el descriptor del mensaje los valores del descriptor del mensaje del primer segmento; la única excepción es el *MDMFL* campo, que el gestor de colas establece para indicar que el mensaje reensamblado es el único segmento.

Si se especifica *GMCMPPM* para un mensaje de informe, el gestor de colas realiza un procesamiento especial. El gestor de colas comprueba la cola para ver si todos los mensajes de informe de ese tipo de informe relacionados con los diferentes segmentos del mensaje lógico están presentes en la cola. Si es así, se pueden recuperar como un único mensaje especificando *GMCMPPM*. Para que esto sea posible, los mensajes de informe deben ser generados por un gestor de colas o MCA que admita segmentación, o bien la aplicación de origen debe solicitar al menos 100 bytes de datos de mensaje (es decir, deben especificarse las opciones *RO*D* o *RO*F* adecuadas). Si no se dispone de la totalidad de los datos de la solicitud para un segmento, los bytes que faltan se sustituyen por valores nulos en el mensaje de informe devuelto.

Si se especifica *GMCMPPM* con *GMMUC* o *GMBRWC*, el cursor de exploración debe colocarse en un mensaje con un *MDOFF* campo en *MQMD* que tenga un valor de 0. Si no se cumple esta condición, la llamada falla con el código de motivo RC2246 .

GMCMPPM implica *GMASGA*, por lo que no es necesario especificarlo.

GMCMPPM se puede especificar con cualquiera de las otras opciones *GM**, excepto *GMPSYP*, y con cualquiera de las opciones *MO**, excepto *MOOFFS*.

GMAMSA

Todos los mensajes del grupo deben estar disponibles.

Esta opción especifica que los mensajes de un grupo solo estarán disponibles para su recuperación cuando todos los mensajes del grupo estén disponibles. Si la cola contiene grupos de mensajes en los que faltan algunos mensajes (quizás porque se han retrasado en la red y aún no han llegado), especificar GMAMSA impide la recuperación de mensajes que pertenecen a grupos incompletos. Sin embargo, esos mensajes siguen contribuyendo al valor del atributo **CurrentQDepth** queue; esto significa que puede que no haya grupos de mensajes recuperables, aunque **CurrentQDepth** sea mayor que cero. Si no hay otros mensajes recuperables, se devuelve RC2033 el código de motivo una vez transcurrido el intervalo de espera especificado (si lo hay).

El procesamiento de GMAMSA depende de si también se especifica GMLOGO:

- Si se especifican ambas opciones, GMAMSA solo tiene efecto cuando no hay ningún grupo actual ni mensaje lógico. Si hay un grupo actual o un mensaje lógico, se ignora GMAMSA. Esto significa que GMAMSA puede permanecer activado cuando procesa mensajes en orden lógico.
- Si se especifica GMAMSA sin GMLOGO, GMAMSA siempre tiene efecto. Esto significa que la opción debe desactivarse después de que el primer mensaje del grupo se haya eliminado de la cola, para poder eliminar los mensajes restantes del grupo.

La finalización satisfactoria de una llamada MQGET que especifica GMAMSA significa que, en el momento en que se emitió la llamada MQGET, todos los mensajes del grupo estaban en la cola. Sin embargo, ten en cuenta que otras aplicaciones aún pueden eliminar mensajes del grupo (el grupo no está bloqueado para la aplicación que recupera el primer mensaje del grupo).

Si no se especifica esta opción, los mensajes pertenecientes a grupos se pueden recuperar incluso cuando el grupo está incompleto.

GMAMSA implica GMASGA, por lo que no es necesario especificarlo.

GMAMSA se puede especificar con cualquiera de las otras opciones GM* y con cualquiera de las opciones MO*.

GMASGA

Todos los segmentos de un mensaje lógico deben estar disponibles.

Esta opción especifica que los segmentos de un mensaje lógico solo estarán disponibles para su recuperación cuando todos los segmentos del mensaje lógico estén disponibles. Si la cola contiene mensajes segmentados en los que faltan algunos segmentos (quizás porque se han retrasado en la red y aún no han llegado), especificar GMASGA impide la recuperación de segmentos que pertenecen a mensajes lógicos incompletos. Sin embargo, esos segmentos siguen contribuyendo al valor del atributo **CurrentQDepth** de cola; esto significa que podría no haber mensajes lógicos recuperables, aunque **CurrentQDepth** sea mayor que cero. Si no hay otros mensajes recuperables, se devuelve RC2033 el código de motivo una vez transcurrido el intervalo de espera especificado (si lo hay).

El procesamiento de GMASGA depende de si también se especifica GMLOGO:

- Si se especifican ambas opciones, GMASGA solo tiene efecto cuando no hay ningún mensaje lógico actual. Si hay un mensaje lógico actual, se ignora GMASGA. Esto significa que GMASGA puede permanecer activado al procesar mensajes en orden lógico.
- Si se especifica GMASGA sin GMLOGO, GMASGA siempre tiene efecto. Esto significa que la opción debe desactivarse después de que el primer segmento del mensaje lógico se haya eliminado de la cola, para poder eliminar los segmentos restantes del mensaje lógico.

Si no se especifica esta opción, los segmentos del mensaje se pueden recuperar incluso cuando el mensaje lógico está incompleto.

Si bien tanto GMCMPM como GMASGA requieren que todos los segmentos estén disponibles antes de que se pueda recuperar cualquiera de ellos, el primero devuelve el mensaje completo, mientras que el segundo permite recuperar los segmentos uno por uno.

Si se especifica GMASGA para un mensaje de informe, el gestor de colas realiza un procesamiento especial. El gestor de colas comprueba la cola para ver si hay al menos un mensaje de informe para cada uno de los segmentos que componen el mensaje lógico completo. Si es así, se cumple la condición GMASGA. Sin embargo, el gestor de colas no comprueba el tipo de los mensajes de informe presentes, por lo que puede haber una mezcla de tipos de informe en los mensajes de informe relacionados con los segmentos del mensaje lógico. Por lo tanto, el éxito de GMASGA no implica que GMCMPM tenga éxito. Si hay una mezcla de tipos de informes presentes para los segmentos de un mensaje lógico concreto, esos mensajes de informe deben recuperarse uno por uno.

GMASGA se puede especificar con cualquiera de las otras opciones GM* y con cualquiera de las opciones MO*.

Opción predeterminada : Si no se requiere ninguna de las opciones descritas, se puede utilizar la siguiente opción:

GMNONE

No se ha especificado ninguna opción.

Este valor se puede utilizar para indicar que no se han especificado otras opciones; todas las opciones adoptan sus valores predeterminados. GMNONE se define para facilitar la documentación del programa; no está previsto que esta opción se utilice con ninguna otra, pero como su valor es cero, dicho uso no puede detectarse.

El valor inicial del campo *GMOPT* es GMNWT.

GMRE1 (cadena de caracteres de 1 byte)

Reservado.

Este es un campo reservado. El valor inicial de este campo es un carácter en blanco. Este campo se ignora si *GMVER* es menor que GMVER2.

GMRL (entero con signo de 10 dígitos)

Longitud de los datos del mensaje devueltos (bytes).

Este es un campo de salida que el gestor de colas establece con la longitud en bytes de los datos del mensaje devueltos por la llamada MQGET en el **BUFFER** parámetro. Si el gestor de colas no admite esta capacidad, *GMRL* se establece en el valor RLUNDF.

Cuando los mensajes se convierten entre codificaciones o conjuntos de caracteres, los datos del mensaje pueden cambiar de tamaño en ocasiones. Al regresar de la llamada MQGET:

- Si no *GMRL* es RLUNDF, el número de bytes de datos del mensaje devueltos viene dado por *GMRL*.
- Si *GMRL* tiene el valor RLUNDF, el número de bytes de datos del mensaje devueltos suele estar determinado por el menor de *BUFLN* y *DATLEN*, pero puede ser inferior a este si la llamada MQGET se completa con el código de motivo RC2079 . Si esto ocurre, los bytes insignificantes del **BUFFER** parámetro se establecen en nulos.

Se define el siguiente valor especial:

RLUNDF

Longitud de los datos devueltos no definida.

El valor inicial de este campo es RLUNDF. Este campo se ignora si *GMVER* es menor que GMVER3.

GMRQN (cadena de caracteres de 48 bytes)

Nombre resuelto de la cola de destino.

Este es un campo de salida que el gestor de colas establece con el nombre local de la cola desde la que se recuperó el mensaje, tal y como se define en el gestor de colas local. Esto difiere del nombre utilizado para abrir la cola si:

- Se abrió una cola alias (en cuyo caso, se devuelve el nombre de la cola local a la que se resolvió el alias), o

- Se abrió una cola modelo (en cuyo caso, se devuelve el nombre de la cola local dinámica).

La longitud de este campo viene dada por LNQN. El valor inicial de este campo es 48 caracteres en blanco.

GMRS2 (cadena de caracteres de 1 byte)

Reservado.

Este es un campo reservado. El valor inicial de este campo es un carácter en blanco. Este campo se ignora si *GMVER* es menor que *GMVER4*.

GMSEG (cadena de caracteres de 1 byte)

Indicador que señala si se permite una segmentación adicional del mensaje recuperado.

Tiene uno de los siguientes valores:

SEGIHB

No se permite la segmentación.

SEGALW

Segmentación permitida.

Se trata de un campo de salida. El valor inicial de este campo es *SEGIHB*. Este campo se ignora si *GMVER* es menor que *GMVER2*.

GMSG1 (entero con signo de 10 dígitos)

Señal.

Este es un campo reservado; su valor no es significativo. El valor inicial de este campo es 0.

GMSG2 (entero con signo de 10 dígitos)

Identificador de señal.

Este es un campo reservado; su valor no es significativo.

GMSID (cadena de caracteres de 4 bytes)

Identificador de estructura.

El valor debe ser:

GMSIDV

Identificador para la estructura de opciones de get-message.

Este campo es siempre un campo de entrada. El valor inicial de este campo es *GMSIDV*.

GMSST (cadena de caracteres de 1 byte)

Indicador que señala si el mensaje recuperado es un segmento de un mensaje lógico.

Tiene uno de los siguientes valores:

SSNSEG

El mensaje no es un segmento.

SSSEG

El mensaje es un segmento, pero no es el último segmento del mensaje lógico.

SSLSEG

El mensaje es el último segmento del mensaje lógico.

Este es también el valor devuelto si el mensaje lógico consta de un solo segmento.

Este campo es un campo de salida. El valor inicial de este campo es *SSNSEG*. Este campo se ignora si *GMVER* es menor que *GMVER2*.

GMTOK (cadena de bits de 16 bytes)

Token de mensaje.

Este es un campo reservado; su valor no es significativo. Se define el siguiente valor especial:

MTKNON

Sin token de mensaje.

El valor es un cero binario para la longitud del campo.

La longitud de este campo viene dada por LNMTOK. El valor inicial de este campo es MTKNON. Este campo se ignora si *GMVER* es menor que *GMVER3*.

GMVER (entero con signo de 10 dígitos)

Número de versión de la estructura.

El valor debe ser uno de los siguientes:

GMVER1

Version-1 Estructura de opciones de get-message.

GMVER2

Version-2 Estructura de opciones de get-message.

GMVER3

Version-3 Estructura de opciones de get-message.

GMVER4

Version-4 Estructura de opciones de get-message.

Los campos que solo existen en las versiones más recientes de la estructura se identifican como tales en las descripciones de los campos. La siguiente constante especifica el número de versión de la versión actual:

GMVERC

Versión actual de la estructura de opciones de get-message.

Este campo es siempre un campo de entrada. El valor inicial de este campo es *GMVER1*.

GMVER (entero con signo de 10 dígitos)

Número de versión de la estructura.

El valor debe ser uno de los siguientes:

GMVER1

Version-1 Estructura de opciones de get-message.

GMVER2

Version-2 Estructura de opciones de get-message.

GMVER3

Version-3 Estructura de opciones de get-message.

GMVER4

Version-4 Estructura de opciones de get-message.

Los campos que solo existen en las versiones más recientes de la estructura se identifican como tales en las descripciones de los campos. La siguiente constante especifica el número de versión de la versión actual:

GMVERC

Versión actual de la estructura de opciones de get-message.

Este campo es siempre un campo de entrada. El valor inicial de este campo es *GMVER1*.

GMWI (entero con signo de 10 dígitos)

Intervalo de espera.

Este es el tiempo aproximado, expresado en milisegundos, que la llamada MQGET espera a que llegue un mensaje adecuado (es decir, un mensaje que cumpla los criterios de selección especificados en el **MSGDSC** parámetro de la llamada MQGET; consulte el *MDMID* campo descrito en [“MQMD \(descriptor de mensaje\) en IBM i” en la página 1131](#) para obtener más detalles). Si no se ha recibido ningún

mensaje adecuado tras transcurrido este tiempo, la llamada finaliza con CCFAIL y el código de motivo RC2033.

GMWI se utiliza con la opción GMWT. Si no se especifica esta opción, se ignora. Si se especifica, *GMWI* debe ser mayor o igual a cero, o el siguiente valor especial:

WIULIM

Intervalo de espera ilimitado.

El valor inicial de este campo es 0.

Valores iniciales

Tabla 743. Campos en MQGMO		
Nombre de campo	Nombre de la constante	Valor de la constante
GMSID	GMSIDV	'GMO↯'
GMVER	GMVER1	1
GMOPT	GMNWT	0
GMWI	Ninguna	0
GMSG1	Ninguna	0
GMSG2	Ninguna	0
GMRQN	Ninguna	Espacios en blanco
GMMO	MOMSGI + MOCORI	3
GMGST	GSNIG	' '
GMSST	SSNSEG	' '
GMSEG	SEGIHB	' '
GMRE1	Ninguna	' '
GMTOK	MTKNON	Nulos
GMRL	RLUNDF	-1
GMRS2	Ninguna	' '
GMMH	HMNONE	0

Notas:

1. El símbolo ↯ representa un único carácter en blanco.

Declaración RPG

```

D*.1.....2.....3.....4.....5.....6.....7..
D*
D* MQGMO Structure
D*
D* Structure identifier
D  GMSID          1      4    INZ('GMO ')
D* Structure version number
D  GMVER          5      8I 0 INZ(1)
D* Options that control the action ofMQGET
D  GMOPT          9     12I 0 INZ(0)
D* Wait interval
D  GMWI          13     16I 0 INZ(0)
D* Signal
D  GMSG1         17     20I 0 INZ(0)
D* Signal identifier
D  GMSG2         21     24I 0 INZ(0)

```

```

D* Resolved name of destination queue
D GMRQN          25      72      INZ
D* Options controlling selection criteria used for MQGET
D GMMO          73      76I 0 INZ(3)
D* Flag indicating whether messageretrieved is in a group
D GMGST         77      77      INZ(' ')
D* Flag indicating whether messageretrieved is a segment of a
D* logicalmessage
D GMSST         78      78      INZ(' ')
D* Flag indicating whether furthersegmentation is allowed for themessage
D* retrieved
D GMSEG         79      79      INZ(' ')
D* Reserved
D GMRE1         80      80      INZ
D* Message token
D GMTOK         81      96      INZ(X'0000000000000000-
D              0000000000000000')
D* Length of message data returned(bytes)
D GMRL          97     100I 0 INZ(-1)
D* Reserved
D GMRS2        101     104I 0 INZ(0)
D* Message handle
D GMMH        105     112I 0 INZ(0)

```

IBM i MQIIH (IMS encabezado de información) en IBM i

La estructura MQIIH describe la información que debe estar presente al inicio de un mensaje enviado al puente IMS a través de IBM MQ for z/OS.

Visión general

Nombre del formato : FMIMS.

Conjunto de caracteres y codificación : Se aplican condiciones especiales al conjunto de caracteres y la codificación utilizados para la estructura MQIIH y los datos de los mensajes de aplicación:

- Las aplicaciones que se conectan al gestor de colas propietario de la cola puente IMS deben proporcionar una estructura MQIIH que esté en el juego de caracteres y la codificación del gestor de colas. Esto se debe a que, en este caso, no se realiza la conversión de datos de la estructura MQIIH.
- Las aplicaciones que se conectan a otros gestores de colas pueden proporcionar una estructura MQIIH en cualquiera de los conjuntos de caracteres y codificaciones compatibles; la conversión de MQIIH la realiza el agente del canal de mensajes receptor conectado al gestor de colas propietario de la cola puente IMS.

Nota: Hay una excepción a esto. Si el gestor de colas propietario de la cola puente IMS utiliza CICS para el en cola distribuido, el MQIIH debe estar en el juego de caracteres y la codificación del gestor de colas propietario de la cola puente IMS.

- Los datos del mensaje de aplicación que siguen la estructura MQIIH deben tener el mismo juego de caracteres y codificación que la estructura MQIIH. Los campos *IICSIIENC* y de la estructura MQIIH no se pueden utilizar para especificar el juego de caracteres y la codificación de los datos del mensaje de la aplicación.

El usuario debe proporcionar una salida de conversión de datos para convertir los datos del mensaje de la aplicación si estos no están en uno de los formatos integrados compatibles con el gestor de colas.

- [“Autenticación de pases para aplicaciones de puent IMS” en la página 1119](#)
- [“Campos” en la página 1120](#)
- [“Valores iniciales” en la página 1123](#)
- [“Declaración de RPG” en la página 1123](#)

Autenticación de pases para aplicaciones de puent IMS

Ahora los IBM MQ administradores pueden especificar el nombre de la aplicación que se utilizará para autenticar los pases de entrada, para las aplicaciones puente IMS. Para ello, el nombre de la aplicación se

especifica como un nuevo atributo PTKTAPPL para la definición del objeto STGCLASS, como una cadena alfanumérica de 1 a 8 caracteres.

Un valor en blanco significa que la autenticación se realiza como en versiones anteriores de IBM MQ, es decir, no se incluye ningún nombre de aplicación en la solicitud de autenticación y, en su lugar, se utiliza el valor MVSxxxx a.

El valor de 1 a 8 caracteres alfanuméricos debe seguir las reglas para los nombres de las aplicaciones de pases descritas en las RACF publicaciones.

IBM MQ Los administradores y RACF los administradores deben ponerse de acuerdo sobre los nombres de aplicación válidos que se utilizarán. El RACF administrador debe crear un perfil en la clase PTKTDATA que otorgue acceso de LECTURA a los ID de usuario de todas las aplicaciones a las que se les concederá acceso. El IBM MQ administrador debe crear o modificar las definiciones STGCLASS necesarias que especifican el nombre de la aplicación que se utilizará para la autenticación de pases.

Para obtener información relacionada, consulte la *Referencia de comandos de Script (MQSC)*.

Campos

La estructura MQIIH contiene los siguientes campos, que se describen en **orden alfabético** :

IIAUT (cadena de caracteres de 8 bytes)

RACF contraseña o ticket de acceso.

Esto es opcional; si se especifica, se utiliza con el ID de usuario en el contexto de seguridad MQMD para crear un UTOKEN que se envía a IMS para proporcionar un contexto de seguridad. Si no se especifica, se utiliza el ID de usuario sin verificación. Esto depende de la configuración de los RACF interruptores, que pueden requerir la presencia de un autenticador.

Esto se ignora si el primer byte está en blanco o es nulo. Se puede utilizar el siguiente valor especial:

IAUNON

Sin autenticación.

La longitud de este campo viene dada por LNAUTH. El valor inicial de este campo es IAUNON.

IICMT (cadena de caracteres de 1 byte)

Modo de compromiso.

Consulte la *Referencia OTMA* para obtener más información sobre los IMS modos de confirmación. El valor debe ser uno de los siguientes:

ICMCTS

Confirmar y enviar.

Este modo implica una doble cola de salida, pero con tiempos de ocupación de la región más cortos. Las transacciones de ruta rápida y conversacionales no pueden ejecutarse con este modo.

ICMSTC

Enviar y luego confirmar.

El valor inicial de este campo es ICMCTS.

IICSI (entero con signo de 10 dígitos)

Reservado.

Este es un campo reservado; su valor no es significativo. El valor inicial de este campo es 0.

IIENC (entero con signo de 10 dígitos)

Reservado.

Este es un campo reservado; su valor no es significativo. El valor inicial de este campo es 0.

IIFLG (entero con signo de 10 dígitos)

Distintivos.

El valor debe ser:

IINONE

Sin banderas.

El valor inicial de este campo es IINONE.

IIFMT (cadena de caracteres de 8 bytes)

IBM MQ nombre del formato de los datos que siguen a MQIIH.

Esto especifica el nombre IBM MQ del formato de los datos que siguen a la estructura MQIIH.

En la llamada MQPUT o MQPUT1, la aplicación debe establecer este campo con el valor adecuado para los datos. Las reglas para codificar este campo son las mismas que para el *MDFMT* campo en MQMD.

La longitud de este campo viene dada por LNFMT. El valor inicial de este campo es FMNONE.

IILEN (entero con signo de 10 dígitos)

Longitud de la estructura MQIIH.

El valor debe ser:

IILEN1

Longitud de la estructura del IMS encabezado de información.

El valor inicial de este campo es IILEN1.

IILTO (cadena de caracteres de 8 bytes)

Anulación lógica del terminal.

Esto se coloca en el campo IO PCB. Es opcional; si no se especifica, se utiliza el nombre TPIPE. Se ignora si el primer byte está en blanco o es nulo.

La longitud de este campo viene dada por LNLTOV. El valor inicial de este campo es 8 caracteres en blanco.

IIMMN (cadena de caracteres de 8 bytes)

Nombre del mapa de servicios de formato de mensajes.

Esto se coloca en el campo IO PCB. Es opcional. En la entrada representa el MID, en la salida representa el MOD. Se ignora si el primer byte está en blanco o es nulo.

La longitud de este campo viene dada por LNMFMN. El valor inicial de este campo es 8 caracteres en blanco.

IIRFM (cadena de caracteres de 8 bytes)

IBM MQ nombre del formato del mensaje de respuesta.

Este es el nombre IBM MQ del formato del mensaje de respuesta que se enviará en respuesta al mensaje actual. Las reglas para codificar esto son las mismas que para el *MDFMT* campo en MQMD.

La longitud de este campo viene dada por LNFMT. El valor inicial de este campo es FMNONE.

IIRSV (cadena de caracteres de 1 byte)

Reservado.

Este es un campo reservado; debe dejarse en blanco.

IISEC (cadena de caracteres de 1 byte)

Ámbito de seguridad.

Esto indica el procesamiento de IMS seguridad requerido. Los valores siguientes están definidos:

ISSCHK

Compruebe el ámbito de seguridad.

Se construye un ACEE en la región de control, pero no en la región dependiente.

ISSFUL

Ámbito de seguridad completo.

Se crea un ACEE almacenado en caché en la región de control y un ACEE no almacenado en caché en la región dependiente. Si utiliza ISSFUL, debe asegurarse de que el ID de usuario para el que se ha creado el ACEE tenga acceso a los recursos utilizados en la región dependiente.

Si no se especifican ISSCHK e ISSFUL para este campo, se asume ISSCHK.

El valor inicial de este campo es ISSCHK.

IISID (cadena de caracteres de 4 bytes)

Identificador de estructura.

El valor debe ser:

IISIDV

Identificador de la estructura del IMS encabezado de información.

El valor inicial de este campo es IISIDV.

IITID (cadena de bits de 16 bytes)

Identificador de la instancia de transacción.

Este campo es utilizado por los mensajes de salida de IMS por lo que se ignora en la primera entrada. Si *IITST* se establece en ITSIC, esto debe proporcionarse en la siguiente entrada, y en todas las entradas posteriores, para permitir IMS que correlacione los mensajes con la conversación correcta. Se puede utilizar el siguiente valor especial:

ITINON

No hay ID de instancia de transacción.

La longitud de este campo viene dada por LNTIID. El valor inicial de este campo es ITINON.

IITST (cadena de caracteres de 1 byte)

Estado de la transacción.

Esto indica el IMS estado de la conversación. Esto se ignora en la primera entrada porque no existe ninguna conversación. En entradas posteriores, indica si una conversación está activa o no. En la salida, se establece mediante IMS. El valor debe ser uno de los siguientes:

ITSIC

En conversación.

ITSNIC

No en conversación.

ITSARC

Devolver los datos del estado de la transacción en formato estructurado.

Este valor solo se utiliza con el IMS/DISPLAY TRAN comando. Hace que los datos del estado de la transacción se devuelvan en forma IMS estructurada en lugar de en forma de caracteres. Consulte [Escribir IMS programas de transacción a través de IBM MQ](#) para obtener más detalles.

El valor inicial de este campo es ITSNIC.

IIVER (entero con signo de 10 dígitos)

Número de versión de la estructura.

El valor debe ser:

IIVER1

Número de versión para la estructura del IMS encabezado de información.

La siguiente constante especifica el número de versión de la versión actual:

IIVERC

Versión actual de la estructura del IMS encabezado de información.

El valor inicial de este campo es IIVER1.

Valores iniciales

Tabla 744. Campos en MQIIH		
Nombre de campo	Nombre de la constante	Valor de la constante
IISID	IISIDV	'IIH↵'
IIVER	IIVER1	1
IILEN	IILEN1	84
IIENC	Ninguna	0
IICSI	Ninguna	0
IIFMT	FMNONE	Espacios en blanco
IIFLG	IINONE	0
IILTO	Ninguna	Espacios en blanco
IIMMN	Ninguna	Espacios en blanco
IIRFM	FMNONE	Espacios en blanco
IIAUT	IAUNON	Espacios en blanco
IITID	ITINON	Nulos
IITST	ITSNIC	' '
IICMT	ICMCTS	'0'
IISEC	ISSCHK	'C'
IIRSV	Ninguna	' '

Notas:

1. El símbolo ↵ representa un único carácter en blanco.

Declaración de RPG

```
D*.1....:....2....:....3....:....4....:....5....:....6....:....7..  
D*  
D* MQIIH Structure  
D*  
D* Structure identifier  
D IISID 1 4 INZ('IIH ')  
D* Structure version number  
D IIVER 5 8I 0 INZ(1)  
D* Length of MQIIH structure  
D IILEN 9 12I 0 INZ(84)  
D* Reserved  
D IIENC 13 16I 0 INZ(0)  
D* Reserved  
D IICSI 17 20I 0 INZ(0)  
D* MQ format name of data that followsMQIIH  
D IIFMT 21 28 INZ(' ')  
D* Flags  
D IIFLG 29 32I 0 INZ(0)  
D* Logical terminal override  
D IILTO 33 40 INZ  
D* Message format services map name
```

D	IIMMN	41	48	INZ
D*	MQ format name of reply message			
D	IIRFM	49	56	INZ('')
D*	RACF password or passticket			
D	IIAUT	57	64	INZ('')
D*	Transaction instance identifier			
D	IITID	65	80	INZ(X'0000000000000000-0000000000000000')
D				
D*	Transaction state			
D	IITST	81	81	INZ('')
D*	Commit mode			
D	IICMT	82	82	INZ('0')
D*	Security scope			
D	IISEC	83	83	INZ('C')
D*	Reserved			
D	IIRSV	84	84	INZ

IBM i MQIMPO (Opciones de propiedad del mensaje de consulta) activado

La estructura MQIMPO permite a las aplicaciones especificar opciones que controlan cómo se consultan las propiedades de los mensajes.

Visión general

Propósito : La estructura es un parámetro de entrada en la llamada MQINQMP.

Conjunto de caracteres y codificación : Los datos en MQIMPO deben estar en el conjunto de caracteres de la aplicación y la codificación de la aplicación (ENNAT).

- “Campos” en la [página 1124](#)
- “Valores iniciales” en la [página 1130](#)
- “Declaración RPG” en la [página 1130](#)

Campos

La estructura MQIMPO contiene los siguientes campos, que se describen en **orden alfabético** :

IPOPT (entero con signo de 10 dígitos)

Las siguientes opciones controlan la acción de MQINQMP. Puede especificar una o varias de estas opciones. Para especificar más de una opción, suma los valores (sin añadir la misma constante más de una vez) o combínalos mediante la operación OR bit a bit (si el lenguaje de programación admite operaciones bit a bit). Las combinaciones de opciones que no son válidas se indican; todas las demás combinaciones son válidas.

Opciones de datos de valor: Las siguientes opciones se refieren al procesamiento de los datos de valor cuando se recupera la propiedad del mensaje.

IPCVAL

Esta opción solicita que el valor de la propiedad se convierta para ajustarse a los *IPREQCSI* valores *IPREQENC* y especificados antes de que la llamada MQINQMP devuelva el valor de la propiedad en el *Value* área.

- Si la conversión se realiza correctamente, los campos *IPRETCSI* *IPRETENC* y se establecen igual que *IPREQCSI* y *IPREQENC* al regresar de la llamada MQINQMP.
- Si la conversión falla, pero la llamada MQINQMP se completa sin errores, el valor de la propiedad se devuelve sin convertir.

Si la propiedad es una cadena, los campos *IPRETCSI* *IPRETENC* y se establecen en el juego de caracteres y la codificación de la cadena sin convertir.

El código de finalización es CCWARN en este caso, con el código de motivo RC2466. El cursor de propiedad avanza hasta la propiedad devuelta.

Si el valor de la propiedad se amplía durante la conversión y supera el tamaño del **Value** parámetro, el valor se devuelve sin convertir, con el código de finalización CCFAIL; el código de motivo se establece en RC2469.

El **DataLength** parámetro de la llamada MQINQMP devuelve la longitud a la que se habría convertido el valor de la propiedad, con el fin de permitir que la aplicación determine el tamaño del búfer necesario para alojar el valor de la propiedad convertido. El cursor de propiedad no ha cambiado.

Esta opción también solicita que:

- Si el nombre de la propiedad contiene un comodín y
- El *IPRETNAMECHRP* campo se inicializa con una dirección o desplazamiento para el nombre devuelto,

A continuación, el nombre devuelto se convierte para ajustarse a los *IPREQCSI* valores *IPREQENC* y.

- Si la conversión se realiza correctamente, el *VSCCSID* campo de *IPRETNAMECHRP* y la codificación del nombre devuelto se establecen en el valor de entrada de *IPREQCSI* y *IPREQENC*.
- Si la conversión falla, pero la llamada MQINQMP se completa sin errores ni advertencias, el nombre devuelto no se convierte. El código de finalización es CCWARN en este caso, con el código de motivo RC2492.

El cursor de propiedad avanza hasta la propiedad devuelta. RC2466 se devuelve si ni el valor ni el nombre se convierten.

Si el nombre devuelto se expande durante la conversión y supera el tamaño del *VSBuFSIZE* campo del *RequestedName*, la cadena devuelta se deja sin convertir, con el código de finalización CCFAIL y el código de motivo establecido en RC2465.

El *VSLength* campo de la estructura MQCHARV devuelve la longitud a la que se habría convertido el valor de la propiedad, con el fin de permitir que la aplicación determine el tamaño del búfer necesario para alojar el valor de la propiedad convertido. El cursor de propiedad no ha cambiado.

Tipo de IPCT

Esta opción solicita que el valor de la propiedad se convierta de su tipo de datos actual al tipo de datos especificado en el **Type** parámetro de la llamada MQINQMP.

- Si la conversión se realiza correctamente, el **Type** parámetro no cambia al devolver la llamada MQINQMP.
- Si la conversión falla, pero la llamada MQINQMP se completa sin errores, la llamada falla con el motivo « RC2470 ». El cursor de propiedad no ha cambiado.

Si la conversión del tipo de datos provoca que el valor se amplíe durante la conversión y el valor convertido supera el tamaño del **Value** parámetro, el valor se devuelve sin convertir, con el código de finalización CCFAIL y el código de motivo establecido en RC2469.

El **DataLength** parámetro de la llamada MQINQMP devuelve la longitud a la que se habría convertido el valor de la propiedad, con el fin de permitir que la aplicación determine el tamaño del búfer necesario para alojar el valor de la propiedad convertido. El cursor de propiedad no ha cambiado.

Si el valor del **Type** parámetro de la llamada MQINQMP no es válido, la llamada falla con el motivo « RC2473 ».

Si la conversión de tipo de datos solicitada no es compatible, la llamada falla con el motivo « RC2470 ». Se admiten las siguientes conversiones de tipos de datos:

Tabla 745. Conversiones de tipos de datos compatibles

Tipo de datos de propiedad	Tipos de datos de destino soportados
TYPBOL	TYPSTR, TYPI8, TYPI16, TYPI32, TYPI64
TYPBST	TYPSTR
TYPI8	TYPSTR, TYPI16, TYPI32, TYPI64
TYPI16	TYPSTR, TYPI32, TYPI64
TYPI32	TYPSTR, TYPI64
TYPI64	TYPSTR
TYPF32	TYPSTR, TYPF64
TYPF64	TYPSTR
TYPSTR	TYPBOL, TYPI8, TYPI16, TYPI32, TYPI64, TYPF32, TYPF64
TYPNUL	Ninguna

Las normas generales que rigen las conversiones soportadas son las siguientes:

- Los valores numéricos de las propiedades se pueden convertir de un tipo de datos a otro, siempre que no se pierdan datos durante la conversión.

Por ejemplo, el valor de una propiedad con el tipo de datos TYPI32 puede convertirse en un valor con el tipo de datos TYPI64, pero no puede convertirse en un valor con el tipo de datos TYPI16.

- Un valor de propiedad de cualquier tipo de datos se puede convertir a una serie.
- Un valor de propiedad de serie se puede convertir a cualquier otro tipo de datos, siempre que la serie tenga el formato correcto para la conversión. Si una aplicación intenta convertir un valor de propiedad de cadena que no está formateado correctamente, IBM MQ devuelve el código de motivo RC2472.
- Si una aplicación intenta una conversión que no es compatible, IBM MQ devuelve el código de motivo RC2470.

Las reglas específicas para convertir el valor de una propiedad de un tipo de datos a otro son las siguientes:

- Al convertir un valor de propiedad TYPBOL en una cadena, el valor TRUE se convierte en la cadena «TRUE» y el valor false se convierte en la cadena «FALSE».
- Al convertir un valor de propiedad TYPBOL a un tipo de datos numérico, el valor TRUE se convierte en uno y el valor FALSE se convierte en cero.
- Al convertir el valor de una propiedad de cadena en un valor TYPBOL, la cadena «TRUE» o «1» se convierte en TRUE, y la cadena «FALSE» o «0» se convierte en FALSE.

Tenga en cuenta que los términos «TRUE» y «FALSE» no distinguen entre mayúsculas y minúsculas.

No se puede convertir ninguna otra cadena; IBM MQ devuelve el código de motivo « RC2472 ».

- Al convertir el valor de una propiedad de cadena en un valor con el tipo de datos TYPI8, TYPI16, TYPI32 o TYPI64, la cadena debe tener el siguiente formato:

```
[blanks][sign]digits
```

El significado de los componentes de la serie es el siguiente:

blanks

Caracteres en blanco opcionales al principio

sign

Un carácter opcional de signo más (+) o signo menos (-).

digits

Una secuencia continua de caracteres de dígitos (0-9). Al menos, debe estar presente un carácter de dígito.

Después de la secuencia de caracteres de dígitos, la serie puede contener otros caracteres que no son caracteres de dígitos, pero la conversión se detiene tan pronto como se llega al primero de estos caracteres. Se da por sentado que la serie representa un entero decimal.

IBM MQ devuelve el código de motivo RC2472 si la cadena no tiene el formato correcto.

- Al convertir el valor de una propiedad de cadena en un valor con el tipo de datos TYPF32 o TYPF64, la cadena debe tener el siguiente formato:

```
[blanks][sign]digits[.digits][e_char[e_sign]e_digits]
```

El significado de los componentes de la serie es el siguiente:

blanks

Caracteres en blanco opcionales al principio

sign

Un carácter opcional de signo más (+) o signo menos (-).

digits

Una secuencia continua de caracteres de dígitos (0-9). Al menos, debe estar presente un carácter de dígito.

e_char

Un carácter exponente, que puede ser «E» o «e».

e_sign

Un signo más (+) o menos (-) opcional para el exponente.

e_digits

Una secuencia contigua de caracteres de dígitos (0-9) para el exponente. Al menos, debe estar presente un carácter de dígito, si la serie contiene un carácter de exponente.

Detrás de la secuencia de caracteres de dígitos, o los caracteres opcionales que representan un exponente, la serie puede contener otros caracteres que no son caracteres de dígitos, pero la conversión se detiene tan pronto como se llega al primero de estos caracteres. Se da por supuesto que la serie representa un número de coma flotante decimal con un exponente que es una potencia de 10.

IBM MQ devuelve el código de motivo RC2472 si la cadena no tiene el formato correcto.

- Al convertir un valor de propiedad numérico en una cadena, el valor se convierte en la representación de cadena del valor como número decimal, no en la cadena que contiene el carácter ASCII para ese valor. Por ejemplo, el número entero 65 se convierte en la cadena «65», no en la cadena «A».
- Al convertir el valor de una propiedad de cadena de bytes a una cadena, cada byte se convierte en los dos caracteres hexadecimales que representan el byte. Por ejemplo, la matriz de bytes {0xF1, 0x12, 0x00, 0xFF} se convierte en la cadena "F11200FF".

IPQLEN

Consultar el tipo y la longitud del valor de la propiedad. La longitud se devuelve en el **DataLength** parámetro de la llamada MQINQMP. El valor de la propiedad no se devuelve.

Si se especifica un *ReturnedName* búfer, el *VSLength* campo de la estructura MQCHARV se rellena con la longitud del nombre de la propiedad. El nombre de la propiedad no se devuelve.

Opciones de iteración: Las siguientes opciones se refieren a la iteración sobre propiedades, utilizando un nombre con un carácter comodín

IPINQF

Consultar la primera propiedad que coincida con el nombre especificado. Después de esta llamada, se establece un cursor en la propiedad que se devuelve.

Éste es el valor predeterminado.

La opción IPINQC se puede utilizar posteriormente con una llamada MQINQMP, si es necesario, para volver a consultar la misma propiedad.

Tenga en cuenta que solo hay un cursor de propiedad; por lo tanto, si cambia el nombre de la propiedad especificado en la llamada MQINQMP, el cursor se restablece.

Esta opción no es válida con ninguna de las siguientes opciones:

IPINQN

IPINQC

IPINQN

Consulta la siguiente propiedad que coincide con el nombre especificado, continuando la búsqueda desde el cursor de la propiedad. El cursor avanza hasta la propiedad que se devuelve.

Si esta es la primera llamada MQINQMP para el nombre especificado, se devuelve la primera propiedad que coincide con el nombre especificado.

La opción IPINQC se puede utilizar posteriormente con una llamada MQINQMP si es necesario, para volver a consultar la misma propiedad.

Si la propiedad situada bajo el cursor se ha eliminado, MQINQMP devuelve la siguiente propiedad coincidente después de la que se ha eliminado.

Si se añade una propiedad que coincide con el comodín mientras se está realizando una iteración, es posible que la propiedad se devuelva o no durante la finalización de la iteración. La propiedad se devuelve una vez que la iteración se reinicia utilizando IPINQF.

Una propiedad que coincida con el comodín que se eliminó, mientras la iteración estaba en curso, no se devuelve después de su eliminación.

Esta opción no es válida con ninguna de las siguientes opciones:

IPINQF

IPINQC

IPINQC

Recupera el valor de la propiedad apuntada por el cursor de propiedad. La propiedad señalada por el cursor de propiedades es la última consultada, utilizando la opción IPINQF o IPINQN.

El cursor de propiedad se restablece cuando se reutiliza el identificador de mensaje, cuando el identificador de mensaje se especifica en el *MsgHandle* campo de MQGMO en una llamada MQGET, o cuando el identificador de mensaje se especifica en los campos *OriginalMsgHandle* o *NewMsgHandle* o de la estructura MQPMO en una llamada MQPUT.

Si se utiliza esta opción cuando aún no se ha establecido el cursor de propiedad, o si se ha eliminado la propiedad a la que apunta el cursor de propiedad, la llamada falla con el código de finalización CCFAIL y el motivo « RC2471 » (El cursor de propiedad no existe).

Esta opción no es válida con ninguna de las siguientes opciones:

IPINQF

IPINQN

Si no se requiere ninguna de las opciones descritas anteriormente, se puede utilizar la siguiente opción:

IPNONE

Utilice este valor para indicar que no se han especificado otras opciones; todas las opciones toman sus valores predeterminados.

IPNONE ayuda a la documentación del programa; no está previsto que esta opción se utilice con ninguna otra, pero como su valor es cero, dicho uso no puede detectarse.

Este campo siempre es un campo de entrada. El valor inicial de este campo es IPINQF.

IPREQCSI (entero con signo de 10 dígitos)

El conjunto de caracteres al que se convertirá el valor de la propiedad consultada si el valor es una cadena de caracteres. Este es también el conjunto de caracteres al que se debe convertir *ReturnedName* el cuando se especifica IPCVAL o IPCTYP.

El valor inicial de este campo es CSAPL.

IPREQENC (entero con signo de 10 dígitos)

Esta es la codificación a la que se debe convertir el valor de la propiedad consultada cuando se especifica IPCVAL o IPCTYP.

El valor inicial de este campo es ENNAT.

IPRE1 (entero con signo de 10 dígitos)

Este es un campo reservado. El valor inicial de este campo es un carácter en blanco.

IPRETCSI (entero con signo de 10 dígitos)

En la salida, este es el conjunto de caracteres del valor devuelto si el **Type** parámetro de la llamada MQINQMP es TYPSTR.

Si se especifica la opción IPCVAL y la conversión se ha realizado correctamente, el *ReturnedCCSID* campo, al regresar, tiene el mismo valor que el valor pasado.

El valor inicial de este campo es cero.

IPRETENC (entero con signo de 10 dígitos)

En la salida, esta es la codificación del valor devuelto.

Si se especifica la opción IPCVAL y la conversión se ha realizado correctamente, el *ReturnedEncoding* campo, al regresar, tiene el mismo valor que el valor pasado.

El valor inicial de este campo es ENNAT.

IPRETNAMCHRP (entero con signo de 10 dígitos)

El nombre real de la propiedad solicitada.

Al introducir datos, se puede pasar un búfer de cadena utilizando el *VSPtr* campo *VSOffset* o de la estructura MQCHARV. La longitud del búfer de cadena se especifica utilizando el *VSBuFSIZE* campo de la estructura MQCHARV.

Al regresar de la llamada MQINQMP, el búfer de cadena se completa con el nombre de la propiedad que se consultó, siempre que el búfer de cadena fuera lo suficientemente largo como para contener el nombre completo. El *VSLength* campo de la estructura MQCHARV se rellena con la longitud del nombre de la propiedad. El *VSCCSID* campo de la estructura MQCHARV se rellena para indicar el juego de caracteres del nombre devuelto, independientemente de si la conversión del nombre ha fallado o no.

Este es un campo de entrada/salida. El valor inicial de este campo es MQCHARV_DEFAULT.

IPSID (entero con signo de 10 dígitos)

Este es el identificador de la estructura. El valor debe ser:

IPSIDV

Identificador para la estructura de opciones de propiedad del mensaje de consulta.

Este campo siempre es un campo de entrada. El valor inicial de este campo es IPSIDV.

IPTYP (entero con signo de 10 dígitos)

Una representación de cadena del tipo de datos de la propiedad.

Si la propiedad se especificó en un encabezado MQRFH2 y no se reconoce el atributo dt MQRFH2, este campo se puede utilizar para determinar el tipo de datos de la propiedad. *TypeString* Se devuelve en el conjunto de caracteres codificados 1208 (UTF-8) y son los primeros ocho bytes del valor del dt atributo de la propiedad que no se ha podido reconocer

Este campo siempre es de salida. El valor inicial de este campo es la cadena nula en el lenguaje de programación C y 8 caracteres en blanco en otros lenguajes de programación.

IPVER (entero con signo de 10 dígitos)

Este es el número de versión de la estructura. El valor debe ser:

IPVER1

Número de versión para la estructura de opciones de propiedades del mensaje de consulta.

La siguiente constante especifica el número de versión de la versión actual:

IPVERC

Versión actual de la estructura de opciones de propiedades de los mensajes de consulta.

Este campo siempre es un campo de entrada. El valor inicial de este campo es IPVER1.

Valores iniciales

Tabla 746. Campos en MQIPMO		
Nombre de campo	Nombre de la constante	Valor de la constante
IPSID	IPSIDV	'IMPO'
IPVER	IPVER1	1
IPOPT	IPINQF	
IPREQENC	ENNAT	
IPREQCSI	CSAPL	
IPRETENC	ENNAT	
IPRETCSI	0	
IPRE1	0	
IPRETNAMCHRP		
IPTYP		vacíos

Declaración RPG

```
D* MQIMPO Structure
D*
D*
D* Structure identifier
D IPSID      1  4 INZ('IMPO')
```

```

D*
D* Structure version number
D IPVER          5    8I 0 INZ(1)
D*
D* Options that control the action of
D* MQINQMP
D IPOPT          9   12I 0 INZ(0)
D*
D* Requested encoding of Value
D IPREQENC       13   16I 0 INZ(273)
D*
D* Requested character set identifier
D* of Value
D IPREQCSI       17   20I 0 INZ(-3)
D*
D* Returned encoding of Value
D IPRETENC       21   24I 0 INZ(273)
D*
D* Returned character set identifier of
D* Value
D IPRETCSI       25   28I 0 INZ(0)
D*
D* Reserved
D IPRE1          29   32I 0 INZ(0)
D*
D* Returned property name
D* Address of variable length string
D IPRETNAMCHRP   33   48* INZ(*NULL)
D* Offset of variable length string
D IPRETNAMCHRO   49   52I 0 INZ(0)
D* Size of buffer
D IPRETNAMVSBS   53   56I 0 INZ(-1)
D* Length of variable length string
D IPRETNAMCHRL   57   60I 0 INZ(0)
D* CCSID of variable length string
D IPRETNAMCHRC   61   64I 0 INZ(-3)
D*
D* Property data type as a string
D IPTYP         65   72  INZ

```

MQMD (descriptor de mensaje) en IBM i

Visión general

Propósito : La estructura MQMD contiene la información de control que acompaña a los datos de la aplicación cuando un mensaje viaja entre las aplicaciones emisora y receptora. La estructura es un parámetro de entrada/salida en las llamadas MQGET, MQPUT y MQPUT1.

Versión : La versión actual de MQMD es MDVER2. Los campos que solo existen en las versiones más recientes de la estructura se identifican como tales en las descripciones que siguen.

El archivo COPY proporcionado contiene la versión más reciente de MQMD compatible con el entorno, pero con el valor inicial del MDVER campo establecido en MDVER1. Para utilizar campos que no están presentes en la estructura version-1, la aplicación debe establecer el MDVER campo en el número de versión requerido.

Hay disponible una declaración para la estructura `version-1` con el nombre `MQMD1`.

Conjunto de caracteres y codificación : Los datos en MQMD deben estar en el conjunto de caracteres dado por el atributo del **CodedCharSetId** gestor de colas y la codificación del gestor de colas local dada por ENNAT. Sin embargo, si la aplicación se ejecuta como un IBM MQ MQI client, la estructura debe estar en el juego de caracteres y en la codificación del cliente.

Si los gestores de colas de envío y recepción utilizan diferentes conjuntos de caracteres o codificaciones, los datos de MQMD se convierten automáticamente. No es necesario que la aplicación convierta el MQMD.

- [“Uso de diferentes versiones de MQMD” en la página 1132](#)
- [“Contexto de mensaje” en la página 1132](#)
- [“Caducidad de mensaje” en la página 1133](#)

- “Campos” en la [página 1133](#)
- “Valores iniciales” en la [página 1175](#)
- “Declaración RPG” en la [página 1176](#)

Uso de diferentes versiones de MQMD

Un MQMD de « version-2 » (Mensaje de datos de movimiento) es, en general, equivalente al uso de un MQMD de « version-1 » (Mensaje de datos de movimiento) y al prefijo de los datos del mensaje con una estructura MQMDE. Sin embargo, si todos los campos de la estructura MQMDE tienen sus valores predeterminados, se puede omitir MQMDE. version-1 Se utilizan MQMD y MQMDE tal y como se describe más adelante en esta sección.

- En las llamadas MQPUT y MQPUT1, si la aplicación proporciona un MQMD version-1, la aplicación puede, opcionalmente, anteponer a los datos del mensaje un MQMDE, estableciendo el MDFMT campo en MQMD en FMMDE para indicar que hay un MQMDE presente. Si la aplicación no proporciona un MQMDE, el gestor de colas asume los valores predeterminados para los campos del MQMDE.

Nota: Varios de los campos que existen en el MQMD version-2 pero no en el MQMD version-1 son campos de entrada/salida en las llamadas MQPUT y MQPUT1. Sin embargo, el gestor de colas no devuelve ningún valor en los campos equivalentes del MQMDE en la salida de las llamadas MQPUT y MQPUT1; si la aplicación requiere esos valores de salida, debe utilizar un MQMD version-2.

- En la llamada MQGET, si la aplicación proporciona un MQMD version-1, el gestor de colas antepone al mensaje devuelto un MQMDE, pero solo si uno o varios de los campos del MQMDE tienen un valor distinto del predeterminado. El MDFMT campo en MQMD tendrá el valor FMMDE para indicar que hay un MQMDE presente.

Los valores predeterminados que el gestor de colas utilizó para los campos del MQMDE son los mismos que los valores iniciales de dichos campos, que se muestran en [Tabla 748 en la página 1175](#).

Cuando un mensaje se encuentra en una cola de transmisión, algunos de los campos de MQMD se establecen en valores concretos; consulte “MQXQH (encabezado de cola de transmisión) en IBM i” en la [página 1273](#) para obtener más detalles.

Contexto de mensaje

Algunos campos de MQMD contienen el contexto del mensaje. Por norma, se utiliza para:

- *El contexto de identidad* se refiere a la aplicación que envió originalmente el mensaje
- *El contexto de origen* se refiere a la aplicación que envió el mensaje más recientemente
- *El contexto del usuario* se refiere a la aplicación que envió originalmente el mensaje.

Estas dos aplicaciones pueden ser la misma aplicación, pero también pueden ser aplicaciones diferentes (por ejemplo, cuando se reenvía un mensaje de una aplicación a otra).

Aunque el contexto de identidad y origen suele tener los significados descritos anteriormente, el contenido de ambos tipos de campos de contexto en MQMD depende en realidad de las opciones PM* que se especifican al enviar el mensaje. Como resultado, el contexto de identidad no se relaciona necesariamente con la aplicación que envió originalmente el mensaje, y el contexto de origen no se relaciona necesariamente con la aplicación que envió el mensaje más recientemente; depende del diseño del conjunto de aplicaciones.

Hay un tipo de aplicación que nunca altera el contexto del mensaje, concretamente el agente del canal de mensajes (MCA). Los MCA que reciben mensajes de gestores de colas remotos utilizan la opción de contexto PMSETA en la llamada MQPUT o MQPUT1. Esto permite al MCA receptor conservar exactamente el contexto del mensaje que viajó con el mensaje desde el MCA emisor. Sin embargo, el resultado es que el contexto de origen no se relaciona con la aplicación que envió el mensaje más recientemente (el MCA receptor), sino que se relaciona con una aplicación anterior que envió el mensaje (posiblemente la propia aplicación de origen).

Para obtener más información, consulte [Contexto del mensaje](#).

Caducidad de mensaje

Los mensajes que han caducado en una cola cargada (una cola que se ha abierto) se eliminan automáticamente de la cola en un plazo razonable tras su caducidad. Algunas otras características nuevas de esta versión de IBM MQ pueden hacer que las colas cargadas se escaneen con menos frecuencia que en la versión anterior del producto; sin embargo, los mensajes caducados en las colas cargadas siempre se eliminan dentro de un plazo razonable tras su caducidad.

Campos

La estructura MQMD contiene los siguientes campos, que se describen en orden alfabético:

MDACC (cadena de bits de 32 bytes)

Ficha contable.

Esto forma parte del *contexto* de identidad del mensaje. Para obtener más información sobre el contexto de los mensajes, consulte [Contexto de los mensajes](#) y [Control de la información de contexto](#).

MDACC permite que una aplicación haga que el trabajo realizado como resultado del mensaje se cobre adecuadamente. El gestor de colas trata esta información como una cadena de bits y no comprueba su contenido.

Cuando el gestor de colas genera esta información, se configura de la siguiente manera:

- El primer byte del campo se establece en la longitud de la información contable presente en los bytes siguientes; esta longitud está comprendida entre cero y 30, y se almacena en el primer byte como un entero binario.
- El segundo byte y los siguientes (según lo especificado en el campo de longitud) se establecen con la información contable adecuada para el entorno.
 -  **z/OS** En z/OS la información contable se establece:
 - Para z/OS batch, la información contable de la tarjeta JES JOB o de un estado JES ACCT en la tarjeta EXEC (los separadores de coma se cambian a X'FF'). Esta información se trunca, si es necesario, a 31 bytes.
 - Para TSO, el número de cuenta del usuario.
 - Para CICS, el identificador de unidad de trabajo LU (6.2) (UEPUOWDS) (26 bytes).
 - Para IMS, el nombre PSB de 8 caracteres concatenado con el token de recuperación de IMS 16 caracteres.
 -  **IBM i** En IBM i, la información contable se establece en el código contable del trabajo.
 -  **Linux**  **AIX** En AIX and Linux, la información contable se establece en el identificador numérico del usuario, en caracteres ASCII.
 -  **Windows** En Windows, la información contable se establece en un identificador Windows NT de seguridad (SID) en formato comprimido. El SID identifica de forma única el identificador de usuario almacenado en el MDUID campo. Cuando el SID se almacena en el MDACC campo, se omite la autoridad de identificación de 6 bytes (ubicada en el tercer byte y los bytes posteriores del SID). Por ejemplo, si el Windows NT SID tiene una longitud de 28 bytes, se almacenan 22 bytes de información del SID en el MDACC campo.
- El último byte se establece en el tipo de token de contabilidad, uno de los siguientes valores:

ATTCIC

CICS Identificador LUOW.

ATTDOS

Token contable predeterminado de PC DOS.

ATTWNT

Windows identificador de seguridad.

ATT400

IBM i ficha contable.

ATTUNX

AIX and Linux identificador numérico.

ATTUSR

Token contable definido por el usuario.

ATTUNK

Tipo de token contable desconocido.

El tipo de token contable se establece en un valor explícito solo en los siguientes entornos:

-  AIX
-  IBM i
-  Windows

y para IBM MQ MQI clients conectados a estos sistemas.

En otros entornos, el tipo de token contable se establece en el valor ATTUNK. En estos entornos, el MDPAT campo se puede utilizar para deducir el tipo de token contable recibido.

- Todos los demás bytes se establecen en cero binario.

Para las llamadas MQPUT y MQPUT1, este es un campo de entrada/salida si se especifica PMSETI o PMSETA en el **PMO** parámetro. Si no se especifica ni PMSETI ni PMSETA, este campo se ignora en la entrada y es un campo de solo salida. Para obtener más información sobre el contexto de los mensajes, consulte [Contexto de los mensajes](#) y [Control de la información de contexto](#).

Tras la finalización satisfactoria de una llamada MQPUT o MQPUT1, este campo contiene el MDACC que se transmitió con el mensaje si se colocó en una cola. Este será el valor de MDACC que se conservará con el mensaje si se retiene (véase la descripción de PMRET en “MQPMO (Opciones de mensajes de envío) activado IBM i” en la página 1198 para obtener más detalles sobre las publicaciones retenidas), pero no se utilizará como MDACC cuando el mensaje se envíe como publicación a los suscriptores, ya que estos proporcionan un valor para anular MDACC en todas las publicaciones que se les envían. Si el mensaje no tiene contexto, el campo es completamente binario cero.

Este es un campo de salida para la llamada MQGET.

Este campo no está sujeto a ninguna traducción basada en el conjunto de caracteres del gestor de colas: el campo se trata como una cadena de bits, y no como una cadena de caracteres.

El gestor de colas no hace nada con la información de este campo. La aplicación debe interpretar la información si desea utilizarla con fines contables.

Se puede utilizar el siguiente valor especial para el MDACC campo:

ACNONE

No se especifica ningún token contable.

El valor es un cero binario para la longitud del campo.

La longitud de este campo viene determinada por LNACCT. El valor inicial de este campo es ACNONE.

MDAID (cadena de caracteres de 32 bytes)

Datos de la aplicación relacionados con la identidad.

Esto forma parte del *contexto* de identidad del mensaje. Para obtener más información sobre el contexto de los mensajes, consulte [Contexto de los mensajes](#) y [Control de la información de contexto](#).

MDAID Es información definida por el conjunto de aplicaciones y que puede utilizarse para proporcionar información adicional sobre el mensaje o su remitente. El gestor de colas trata esta

información como datos de caracteres, pero no define su formato. Cuando el gestor de colas genera esta información, está completamente en blanco.

Para las llamadas MQPUT y MQPUT1, este es un campo de entrada/salida si se especifica PMSETI o PMSETA en el **PMO** parámetro. Si hay un carácter nulo, el gestor de colas convierte el carácter nulo y los caracteres siguientes en espacios en blanco. Si no se especifica ni PMSETI ni PMSETA, este campo se ignora en la entrada y es un campo de solo salida. Para obtener más información sobre el contexto de los mensajes, consulte [Contexto de los mensajes](#) y [Control de la información de contexto](#).

Tras la finalización satisfactoria de una llamada MQPUT o MQPUT1, este campo contiene el MDAID que se transmitió con el mensaje si se colocó en una cola. Este será el valor de MDAID que se conservará con el mensaje si se retiene (véase la descripción de PMRET para obtener más detalles sobre las publicaciones retenidas), pero no se utilizará como MDAID cuando el mensaje se envíe como publicación a los suscriptores, ya que estos proporcionan un valor para anular MDAID en todas las publicaciones que se les envían. Si el mensaje no tiene contexto, el campo queda completamente en blanco.

Este es un campo de salida para la llamada MQGET. La longitud de este campo viene determinada por LNAIDD. El valor inicial de este campo es 32 caracteres en blanco.

MDAOD (cadena de caracteres de 4 bytes)

Datos de la aplicación relacionados con el origen.

Esto forma parte del *contexto* original del mensaje. Para obtener más información sobre el contexto de los mensajes, consulte [Contexto de los mensajes](#) y [Control de la información de contexto](#).

MDAOD Es información definida por el conjunto de aplicaciones que se puede utilizar para proporcionar información adicional sobre el origen del mensaje. Por ejemplo, podrían establecerlo las aplicaciones que se ejecutan con la autoridad de usuario adecuada para indicar si los datos de identidad son fiables.

El gestor de colas trata esta información como datos de caracteres, pero no define su formato. Cuando el gestor de colas genera esta información, está completamente en blanco.

Para las llamadas MQPUT y MQPUT1, este es un campo de entrada/salida si se especifica PMSETA en el **PMO** parámetro. Cualquier información que siga a un carácter nulo dentro del campo se descarta. El carácter nulo y cualquier carácter posterior son convertidos en espacios en blanco por el gestor de colas. Si no se especifica PMSETA, este campo se ignora en la entrada y es un campo de solo salida.

Tras la finalización satisfactoria de una llamada MQPUT o MQPUT1, este campo contiene el MDAOD que se transmitió con el mensaje si se colocó en una cola. Este será el valor de MDAOD que se conservará con el mensaje si se retiene (véase la descripción de PMRET para obtener más detalles sobre las publicaciones retenidas), pero no se utilizará como MDAOD cuando el mensaje se envíe como publicación a los suscriptores, ya que estos proporcionan un valor para anular MDAOD en todas las publicaciones que se les envían. Si el mensaje no tiene contexto, el campo queda completamente en blanco.

Este es un campo de salida para la llamada MQGET. La longitud de este campo viene dada por LNAORD. El valor inicial de este campo es 4 caracteres en blanco.

MDBOC (entero con signo de 10 dígitos)

Umbral de restitución.

Este es un recuento del número de veces que el mensaje ha sido devuelto previamente por la llamada MQGET como parte de una unidad de trabajo y, posteriormente, descartado. Se proporciona como ayuda para la aplicación en la detección de errores de procesamiento basados en el contenido de los mensajes. El recuento excluye las llamadas MQGET que especificaron cualquiera de las opciones GMBRW*.

La precisión de este recuento se ve afectada por el **HardenGetBackout** atributo queue; véase [“Atributos para colas” en la página 1399](#).

Este es un campo de salida para la llamada MQGET. Se ignora para las llamadas MQPUT y MQPUT1. El valor inicial de este campo es 0.

MDCID (cadena de bits de 24 bytes)

Identificador de correlación.

Se trata de una cadena de bytes que la aplicación puede utilizar para relacionar un mensaje con otro, o para relacionar el mensaje con otras tareas que la aplicación está realizando. El identificador de correlación es una propiedad permanente del mensaje y persiste tras los reinicios del gestor de colas. Dado que el identificador de correlación es una cadena de bytes y no una cadena de caracteres, el identificador de correlación no se convierte entre conjuntos de caracteres cuando el mensaje fluye de un gestor de colas a otro.

Para las llamadas MQPUT y MQPUT1, la aplicación puede especificar cualquier valor. El gestor de colas transmite este valor con el mensaje y lo entrega a la aplicación que emite la solicitud de obtención del mensaje.

Si la aplicación especifica PMNCID, el gestor de colas genera un identificador de correlación único que se envía con el mensaje y también se devuelve a la aplicación remitente al salir de la llamada MQPUT o MQPUT1.

Este identificador de correlación generado se conserva con el mensaje si este se retiene y se utiliza como identificador de correlación cuando el mensaje se envía como una publicación a los suscriptores que especifican CINONE en el SDCID campo del MQSD transmitido en la llamada MQSUB.

Consulte [“MQPMO \(Opciones de mensajes de envío\) activado IBM i”](#) en la página 1198 para obtener más detalles sobre las publicaciones conservadas

Cuando el gestor de colas o un agente de canal de mensajes genera un mensaje de informe, establece el MDCID campo de la forma especificada por el MDREP campo del mensaje original, ya sea ROCMTC o ROPCI. Las aplicaciones que generan mensajes de informe también deben hacerlo.

Para la llamada MQGET, MDCID es uno de los cinco campos que se pueden utilizar para seleccionar un mensaje concreto que se va a recuperar de la cola. Consulte la descripción del campo MDMID para obtener más información sobre cómo especificar valores para este campo.

Especificar CINONE como identificador de correlación tiene el mismo efecto que no especificar MOCORI, es decir, cualquier identificador de correlación coincidirá.

Si se especifica la opción GMMUC en el **GMO** parámetro de la llamada MQGET, este campo se ignora.

Al regresar de una llamada MQGET, el MDCID campo se establece en el identificador de correlación del mensaje devuelto (si lo hay).

Se pueden utilizar los siguientes valores especiales:

CINONE

No se ha especificado ningún identificador de correlación.

El valor es un cero binario para la longitud del campo.

CINEWS

El mensaje es el inicio de una nueva sesión.

Este valor es reconocido por el CICS bridge como indicativo del inicio de una nueva sesión, es decir, el inicio de una nueva secuencia de mensajes.

Para la llamada MQGET, este es un campo de entrada/salida. Para las llamadas MQPUT y MQPUT1, este es un campo de entrada si no se especifica PMNCID, y un campo de salida si se especifica PMNCID. La longitud de este campo viene dada por LNCID. El valor inicial de este campo es CINONE.

MDCSI (entero con signo de 10 dígitos)

Especifica el identificador del juego de caracteres de los datos de caracteres del mensaje.

Nota: Los datos de caracteres en MQMD y las demás IBM MQ estructuras de datos que son parámetros en las llamadas deben estar en el juego de caracteres del gestor de colas. Esto se define mediante el atributo **CodedCharSetId** del gestor de colas; consulte [“Atributos del gestor de colas en IBM i”](#) en la página 1432 para obtener más información sobre este atributo.

Se pueden utilizar los siguientes valores especiales:

CSQM

Identificador del juego de caracteres del gestor de colas.

Los datos de caracteres del mensaje están en el juego de caracteres del gestor de colas.

En las llamadas MQPUT y MQPUT1, el gestor de colas cambia este valor en el MQMD enviado con el mensaje al identificador de juego de caracteres verdadero del gestor de colas. Como resultado, el valor CSQM nunca es devuelto por la llamada MQGET.

CSINHT

Heredar el identificador del conjunto de caracteres de esta estructura.

Los datos de caracteres del mensaje están en el mismo juego de caracteres que esta estructura; este es el juego de caracteres del gestor de colas. (Solo para MQMD, CSINHT tiene el mismo significado que CSQM).

El gestor de colas cambia este valor en el MQMD enviado con el mensaje al identificador real del juego de caracteres de MQMD. Si no se produce ningún error, la llamada MQGET no devuelve el valor CSINHT.

CSINHT no se puede utilizar si el valor del MDPAT campo en MQMD es ATBRKR.

CSEMBD

Identificador de conjunto de caracteres incrustado.

Los datos de caracteres del mensaje se encuentran en un conjunto de caracteres con el identificador que se incluye en los propios datos del mensaje. Puede haber cualquier número de identificadores de conjunto de caracteres incrustados en los datos del mensaje, que se aplican a diferentes partes de los datos. Este valor debe utilizarse para los mensajes PCF que contengan datos en una combinación de conjuntos de caracteres. Los mensajes PCF tienen un nombre de formato FMPCF.

Especifique este valor solo en las llamadas MQPUT y MQPUT1. Si se especifica en la llamada MQGET, impide la conversión del mensaje.

En las llamadas MQPUT y MQPUT1, el gestor de colas cambia los valores CSQM y CSINHT en el MQMD enviado con el mensaje, tal y como se ha descrito anteriormente, pero no cambia el MQMD especificado en la llamada MQPUT o MQPUT1. No se realiza ninguna otra comprobación sobre el valor especificado.

Las aplicaciones que recuperan mensajes deben comparar este campo con el valor que espera la aplicación; si los valores difieren, es posible que la aplicación tenga que convertir los datos de caracteres del mensaje.

Si se especifica la opción GMCONV en la llamada MQGET, este campo es un campo de entrada/salida. El valor especificado por la aplicación es el identificador codificado del conjunto de caracteres al que deben convertirse los datos del mensaje si es necesario. Si la conversión se realiza correctamente o no es necesaria, el valor no cambia (excepto que el valor CSQM o CSINHT se convierte al valor real). Si la conversión no se realiza correctamente, el valor tras la llamada a MQGET representa el identificador del juego de caracteres codificado del mensaje no convertido que se devuelve a la aplicación.

De lo contrario, este es un campo de salida para la llamada MQGET y un campo de entrada para las llamadas MQPUT y MQPUT1. El valor inicial de este campo es CSQM.

MDENC (entero con signo de 10 dígitos)

Codificación numérica de datos de mensaje.

Esto especifica la codificación numérica de los datos numéricos en el mensaje; no se aplica a los datos numéricos en la propia estructura MQMD. La codificación numérica define la representación utilizada para los enteros binarios, los enteros decimales empaquetados y los números de coma flotante.

En la llamada MQPUT o MQPUT1, la aplicación debe establecer este campo con el valor adecuado para los datos. El gestor de colas no comprueba que el campo sea válido. Se define el siguiente valor especial:

ENNAT

Codificación nativa de la máquina.

La codificación es la predeterminada para el lenguaje de programación y la máquina en la que se ejecuta la aplicación.

Nota: El valor de esta constante depende del lenguaje de programación y del entorno. Por este motivo, las aplicaciones deben compilarse utilizando los archivos de encabezado, macro, COPY o INCLUDE adecuados para el entorno en el que se ejecutará la aplicación.

Las aplicaciones que envían mensajes normalmente deben especificar ENNAT. Las aplicaciones que recuperan mensajes deben comparar este campo con el valor ENNAT; si los valores difieren, es posible que la aplicación tenga que convertir los datos numéricos del mensaje. La opción GMCONV se puede utilizar para solicitar al gestor de colas que convierta el mensaje como parte del procesamiento de la llamada MQGET.

Si se especifica la opción GMCONV en la llamada MQGET, este campo es un campo de entrada/salida. El valor especificado por la aplicación es la codificación a la que se deben convertir los datos del mensaje si es necesario. Si la conversión se realiza correctamente o no es necesaria, el valor permanece sin cambios. Si la conversión no se realiza correctamente, el valor tras la llamada a MQGET representa la codificación del mensaje no convertido que se devuelve a la aplicación.

En otros casos, se trata de un campo de salida para la llamada MQGET y un campo de entrada para las llamadas MQPUT y MQPUT1. El valor inicial de este campo es ENNAT.

MDEXP (entero con signo de 10 dígitos)

Duración del mensaje.

Es un periodo de tiempo expresado en décimas de segundo, establecido por la aplicación que envía el mensaje. El mensaje se debe poder seleccionar para descartarlo si no se ha suprimido de la cola de destino antes de que transcurra este período de tiempo.

El valor se reduce para reflejar el tiempo que el mensaje permanece en la cola de destino y también en cualquier cola de transmisión intermedia si el envío se realiza a una cola remota. También puede ser decrementado por los agentes del canal de mensajes para reflejar los tiempos de transmisión, si estos son significativos. Del mismo modo, una aplicación que reenvíe este mensaje a otra cola podría disminuir el valor si fuera necesario, en caso de que haya retenido el mensaje durante un tiempo considerable. Sin embargo, el tiempo de expiración se considera aproximado y no es necesario decrementar el valor para reflejar pequeños intervalos de tiempo.

Cuando una aplicación recupera el mensaje mediante la llamada MQGET, el MDEXP campo representa el tiempo restante del tiempo de expiración original.

Una vez transcurrido el tiempo de caducidad de un mensaje, el gestor de colas puede descartarlo. En las implementaciones actuales, el mensaje se descarta cuando se produce una llamada MQGET de exploración o no exploración que habría devuelto el mensaje si no hubiera caducado ya. Por ejemplo, una llamada MQGET sin exploración con el GMMO campo en MQGMO establecido en MONONE que lee desde una cola ordenada FIFO provocará que se descarten todos los mensajes caducados hasta el primer mensaje no caducado. Con una cola ordenada por prioridad, la misma llamada descartará los mensajes caducados de mayor prioridad y los mensajes de igual prioridad que llegaron a la cola antes que el primer mensaje no caducado.

Un mensaje que ha caducado nunca se devuelve a una aplicación (ya sea mediante una llamada MQGET de exploración o sin exploración), por lo que el valor del MDEXP campo del descriptor del mensaje tras una llamada MQGET correcta es mayor que cero o el valor especial EIULIM.

Si un mensaje se coloca en una cola remota, es posible que caduque (y se descarte) mientras se encuentra en una cola de transmisión intermedia, antes de llegar a la cola de destino.

Se genera un informe cuando se descarta un mensaje caducado, si el mensaje especificaba una de las opciones de informe ROEXP*. Si no se especifica ninguna de estas opciones, no se genera dicho informe; se supone que el mensaje ya no es relevante después de este período de tiempo (quizás porque un mensaje posterior lo ha sustituido).

Cualquier otro programa que descarte mensajes basándose en el tiempo de caducidad también debe enviar un mensaje de informe adecuado si se ha solicitado.

Nota:

1. Si se envía un mensaje con un MDEXP tiempo de espera de cero, la llamada MQPUT o MQPUT1 falla con el código de motivo RC2013; o se genera ningún mensaje de informe en este caso.
2. Dado que un mensaje cuyo tiempo de caducidad ha expirado puede no descartarse realmente hasta más tarde, es posible que haya mensajes en una cola que hayan superado su tiempo de caducidad y que, por lo tanto, no sean aptos para su recuperación. No obstante, estos mensajes cuentan para el número de mensajes en la cola a todos los efectos, incluido el desencadenamiento de profundidad.
3. Se genera un informe de caducidad, si se solicita, cuando el mensaje se descarta realmente, no cuando se convierte en susceptible de ser descartado.
4. El descarte de un mensaje caducado y la generación de un informe de caducidad, si se solicita, nunca forman parte de la unidad de trabajo de la aplicación, incluso si el mensaje se programó para su descarte como resultado de una llamada MQGET que opera dentro de una unidad de trabajo.
5. Si un mensaje a punto de caducar se recupera mediante una llamada MQGET dentro de una unidad de trabajo y, posteriormente, se revoca dicha unidad de trabajo, es posible que el mensaje pueda descartarse antes de que se pueda recuperar de nuevo.
6. Si un mensaje a punto de caducar queda bloqueado por una llamada MQGET con GMLK, es posible que el mensaje pueda descartarse antes de que pueda recuperarse mediante una llamada MQGET con GMMUC; si eso ocurre, se devuelve el código de motivo RC2034 en esta llamada MQGET posterior.
7. Cuando se recupera un mensaje de solicitud con un tiempo de caducidad superior a cero, la aplicación puede realizar una de las siguientes acciones al enviar el mensaje de respuesta:
 - Copie el tiempo de caducidad restante del mensaje de solicitud al mensaje de respuesta.
 - Establezca el tiempo de caducidad en el mensaje de respuesta en un valor explícito mayor que cero.
 - Establezca el tiempo de caducidad en el mensaje de respuesta a EIULIM.

La acción a realizar depende del diseño del conjunto de aplicaciones. Sin embargo, la acción predeterminada para colocar mensajes en una cola de mensajes no entregados (mensajes no entregados) debe ser conservar el tiempo de caducidad restante del mensaje y continuar disminuyéndolo.

8. Los mensajes de activación siempre se generan con EIULIM.
 9. Un mensaje (normalmente en una cola de transmisión) que tiene un MDFMT nombre de FMXQH tiene un segundo descriptor de mensaje dentro del MQXQH. Por lo tanto, tiene dos MDEXP campos asociados. En este caso, deben tenerse en cuenta los siguientes puntos adicionales:
 - Cuando una aplicación coloca un mensaje en una cola remota, el gestor de colas coloca inicialmente el mensaje en una cola de transmisión local y antepone a los datos del mensaje de la aplicación una estructura MQXQH. El gestor de colas establece los valores de los dos MDEXP campos para que sean iguales a los especificados por la aplicación.
- Si una aplicación coloca un mensaje directamente en una cola de transmisión local, los datos del mensaje deben comenzar con una estructura MQXQH y el nombre del formato debe ser FMXQH (pero el gestor de colas no lo exige). En este caso, la aplicación no necesita establecer los valores de estos dos MDEXP campos para que sean iguales. (El gestor de colas no comprueba que el MDEXP campo dentro de MQXQH contenga un valor válido, ni siquiera que los datos del mensaje sean lo suficientemente largos como para incluirlo)

- Cuando se recupera un mensaje con el MDFMT nombre FMXQH de una cola (ya sea una cola normal o de transmisión), el gestor de colas decrementa ambos MDEXP campos con el tiempo de espera en la cola. No se produce ningún error si los datos del mensaje no son lo suficientemente largos como para incluir el MDEXP campo en el MQXQH.
- El gestor de colas utiliza el MDEXP campo del descriptor de mensajes independiente (es decir, no el del descriptor de mensajes integrado en la estructura MQXQH) para comprobar si el mensaje es susceptible de ser descartado.
- Si los valores iniciales de los dos MDEXP campos fueran diferentes, sería posible que el MDEXP tiempo en el descriptor de mensaje independiente cuando se recupera el mensaje fuera mayor que cero (por lo que el mensaje no sería susceptible de descartarse), mientras que el tiempo según el MDEXP campo en el MQXQH hubiera transcurrido. En este caso, el MDEXP campo en el MQXQH se establece en cero.

Se reconoce el siguiente valor especial:

EIULIM

Vida útil ilimitada.

El mensaje tiene una validez ilimitada.

Este es un campo de salida para la llamada MQGET y un campo de entrada para las llamadas MQPUT y MQPUT1. El valor inicial de este campo es EIULIM.

MDFB (entero con signo de 10 dígitos)

Código de retorno o de razón.

Se utiliza con un mensaje de tipo MTRPRT para indicar la naturaleza del informe, y solo tiene sentido con ese tipo de mensaje. El campo puede contener uno de los valores FB* o uno de los valores RC*. Los códigos de retroalimentación se agrupan de la siguiente manera:

FBNONE

No se ha proporcionado ningún comentario.

FBSFST

Valor más bajo para los comentarios generados por el sistema.

FBSLST

Máximo valor para los comentarios generados por el sistema.

El rango de códigos de retroalimentación generados por el sistema, desde FBSFST hasta FBSLST, incluye los códigos de retroalimentación generales que se enumeran más adelante en esta sección (FB*), así como los códigos de motivo (RC*) que pueden aparecer cuando el mensaje no se puede colocar en la cola de destino.

FBAFST

Valor más bajo para los comentarios generados por la aplicación.

FBALST

Máximo valor para los comentarios generados por la aplicación.

Las aplicaciones que generan mensajes de informe no deben utilizar códigos de respuesta en el rango del sistema (excepto FBQUIT), a menos que deseen simular mensajes de informe generados por el gestor de colas o el agente de canal de mensajes.

En las llamadas MQPUT o MQPUT1, el valor especificado debe ser FBNONE o estar dentro del rango del sistema o del rango de la aplicación. Esto se comprueba independientemente del valor de MDMT.

Códigos generales de comentarios :

FBCOA

Confirmación de llegada a la cola de destino (véase ROCOA).

FBCOD

Confirmación de entrega a la aplicación receptora (véase ROCOD).

FBEXP

Mensaje caducado.

El mensaje se descartó porque no se había eliminado de la cola de destino antes de que expirara su tiempo de validez.

FBPAN

Notificación de acción positiva (véase ROPAN).

FBNAN

Notificación de acción negativa (véase RONAN).

FBQUIT

La aplicación debería terminar.

Esto puede ser utilizado por un programa de programación de cargas de trabajo para controlar el número de instancias de un programa de aplicación que se están ejecutando. El envío de un mensaje MTRPRT con este código de respuesta a una instancia del programa de aplicación indica a dicha instancia que debe detener el procesamiento. Sin embargo, el cumplimiento de esta convención es una cuestión que compete a la aplicación; no es impuesta por el gestor de colas.

IMS-códigos de retroalimentación del puente : Cuando el puente IMS recibe un código de detección -OTMA IMS distinto de cero, el puente IMS convierte el código de detección de hexadecimal a decimal, añade el valor FBIERR (300) y coloca el resultado en el MDFB campo del mensaje de respuesta. Esto hace que el código de retroalimentación tenga un valor comprendido entre FBIFST (301) y FBILST (399) cuando se produce un error IMS -OTMA.

El IMS puente puede generar los siguientes códigos de retroalimentación:

FBDLZ

Longitud de datos cero.

La longitud del segmento era cero en los datos de aplicación del mensaje.

FBDLN

Longitud de datos negativa.

La longitud del segmento era negativa en los datos de aplicación del mensaje.

FBDLTB

Longitud de datos demasiado grande.

La longitud de un segmento era demasiado grande en los datos de aplicación del mensaje.

FBBUFO

Desbordamiento de almacenamiento intermedio.

El valor de uno de los campos de longitud provocaría que los datos desbordaran el búfer de mensajes.

FBLOB1

Longitud errónea en uno.

El valor de uno de los campos de longitud era un byte demasiado corto.

FBIIH

Estructura MQIIH no válida o inexistente.

El MDFMT campo en MQMD especifica FMIMS, pero el mensaje no comienza con una estructura MQIIH válida.

FBNAFI

ID de usuario no autorizado para su uso en IMS.

El ID de usuario contenido en el descriptor de mensaje MQMD, o la contraseña contenida en el IIAUT campo de la estructura MQIIH, no superó la validación realizada por el puente IMS. Como resultado, el mensaje no se transmitió a IMS.

FBIERR

Error inesperado devuelto por IMS.

Se ha producido un IMSerror inesperado. Consulte el registro IBM MQ de errores del sistema en el que reside el puente IMS para obtener más información sobre el error.

FBIFST

Valor más bajo para la IMSretroalimentación generada.

IMSLos códigos de retroalimentación generados ocupan el rango FBIFST (300) a FBILST (399). El código de IMS sentido -OTMA en sí mismo es MDFB menos FBIERR.

FBILST

Mayor valor para los comentarios IMS generados.

CICS-códigos de retroalimentación del puente : El siguiente puede generar los siguientes códigos de CICS bridge retroalimentación:

FBCAAB

Aplicación cancelada.

El programa de aplicación especificado en el mensaje se ha interrumpido. Este código de respuesta solo aparece en el DLREA campo de la estructura MQDLH.

FBCANS

No se puede iniciar la aplicación.

El EXEC CICS LINK para el programa de aplicación especificado en el mensaje ha fallado. Este código de respuesta solo aparece en el DLREA campo de la estructura MQDLH.

FBCBRF

CICS bridge Terminó de forma anómala sin completar el procesamiento normal de errores.

FBCCSSE

Identificador de conjunto de caracteres no válido.

FBCIHE

CICS Estructura del encabezado de información ausente o no válida.

FBCCAE

Longitud de CICS commarea no válida.

FBCCE

Identificador de correlación no válido.

FBCDLQ

Cola de mensajes no entregados no disponible.

La CICS bridge tarea no pudo copiar una respuesta a esta solicitud en la cola de mensajes no entregados. La solicitud fue rechazada.

FBCENE

Codificación no válida.

FBCINE

CICS bridge Se ha producido un error inesperado.

Este código de respuesta solo aparece en el DLREA campo de la estructura MQDLH.

FBCNTA

Identificador de usuario no autorizado o contraseña no válida.

Este código de respuesta solo aparece en el DLREA campo de la estructura MQDLH.

FBCUBO

Unidad de trabajo retirada.

La unidad de trabajo se canceló por una de las siguientes razones:

- Se detectó un error al procesar otra solicitud dentro de la misma unidad de trabajo.

- Se produjo un CICS error mientras se estaba ejecutando la unidad de trabajo.

FBCUWE

Campo de control de unidad CIUOW de trabajo no válido.

MQ Códigos de motivo : Para los mensajes de informe de excepción, MDFB contiene un código de motivo MQ. Entre los posibles códigos de motivo se encuentran:

RC2051

(2051, X'803') Se inhibieron las llamadas de puesta en cola.

RC2053

(2053, X'805') La cola ya contiene el número máximo de mensajes.

RC2035

(2035, X'7F3') Acceso no autorizado.

RC2056

(2056, X'808') No hay espacio disponible en el disco para la cola.

RC2048

(2048, X'800') La cola no admite mensajes persistentes.

RC2031

(2031, X'7EF') La longitud del mensaje supera el máximo permitido por el gestor de colas.

RC2030

(2030, X'7EE') La longitud del mensaje supera el máximo permitido para la cola.

Este es un campo de salida para la llamada MQGET y un campo de entrada para las llamadas MQPUT y MQPUT1. El valor inicial de este campo es FBNONE.

MDFMT (cadena de caracteres de 8 bytes)

Nombre de formato de los datos del mensaje.

Este es un nombre que el remitente del mensaje puede utilizar para indicar al destinatario la naturaleza de los datos contenidos en el mensaje. Se puede especificar cualquier carácter del juego de caracteres del gestor de colas para el nombre, pero se recomienda que el nombre se limite a lo siguiente:

- Mayúsculas de la A a la Z
- Dígitos numéricos del 0 al 9

Si se utilizan otros caracteres, es posible que no se pueda traducir el nombre entre los conjuntos de caracteres de los gestores de colas de envío y recepción.

El nombre debe rellenarse con espacios en blanco hasta alcanzar la longitud del campo, o bien debe utilizarse un carácter nulo para terminar el nombre antes del final del campo; el carácter nulo y cualquier carácter posterior se tratan como espacios en blanco. No especifique un nombre con espacios en blanco al principio o en medio. Para la llamada MQGET, el gestor de colas devuelve el nombre relleno con espacios en blanco hasta alcanzar la longitud del campo.

El gestor de colas no comprueba que el nombre cumpla con las recomendaciones descritas anteriormente.

Los nombres que comienzan por « MQ » en mayúsculas, minúsculas y mayúsculas y minúsculas mezcladas tienen significados definidos por el gestor de colas; no debe utilizar nombres que comiencen por estas letras para sus propios formatos. Los formatos integrados del gestor de colas son:

FMNONE

Sin nombre de formato.

La naturaleza de los datos es indefinida. Esto significa que los datos no se pueden convertir cuando el mensaje se recupera de una cola utilizando la opción GMCONV.

Si se especifica GMCONV en la llamada MQGET y el juego de caracteres o la codificación de los datos del mensaje difiere de la especificada en el **MSGDSC** parámetro, el mensaje se devuelve con los siguientes códigos de finalización y motivo (suponiendo que no haya otros errores):

- Código de finalización CCWARN y código de motivo RC2110 si los datos FMNONE se encuentran al principio del mensaje.
- Código de finalización CCOK y código de motivo RCNONE si los datos FMNONE se encuentran al final del mensaje (es decir, precedidos por una o varias estructuras de encabezado MQ). En este caso, las estructuras de encabezado « MQ » se convierten al juego de caracteres y la codificación solicitados.

FMADMN

Mensaje de solicitud/respuesta del servidor de comandos.

El mensaje es una solicitud o respuesta del servidor de comandos en formato de comando programable (PCF). Los mensajes de este formato se pueden convertir si se especifica la opción GMCONV en la llamada MQGET. Para obtener más información sobre el uso de mensajes con formato de comando programable, consulte [Uso de formatos de comando programables](#).

FMCICS

CICS encabezado de información.

Los datos del mensaje comienzan con el CICS encabezado de información MQCIH, seguido de los datos de la aplicación. El nombre del formato de los datos de la aplicación viene dado por el CIFMT campo de la estructura MQCIH.

FMCMD1

Mensaje de respuesta de comando tipo 1.

El mensaje es un mensaje de respuesta del servidor de comandos MQSC que contiene el recuento de objetos, el código de finalización y el código de motivo. Los mensajes de este formato se pueden convertir si se especifica la opción GMCONV en la llamada MQGET.

FMCMD2

Mensaje de respuesta de comando tipo 2.

El mensaje es un mensaje de respuesta del servidor de comandos MQSC que contiene información sobre los objetos solicitados. Los mensajes de este formato se pueden convertir si se especifica la opción GMCONV en la llamada MQGET.

FMDLH

Encabezado de carta sin respuesta.

Los datos del mensaje comienzan con el encabezado de carta muerta MQDLH. Los datos del mensaje original siguen inmediatamente a la estructura MQDLH. El nombre del formato de los datos del mensaje original viene dado por el DLFMT campo de la estructura MQDLH; consulte [“MQDLH \(Encabezado de carta muerta\) en IBM i”](#) en la [página 1084](#) para obtener más detalles sobre esta estructura. Los mensajes de este formato se pueden convertir si se especifica la opción GMCONV en la llamada MQGET.

Los informes COA y COD no se generan para los mensajes que tienen un MDFMT de FMDLH.

FMDH

Encabezado de la lista de distribución.

Los datos del mensaje comienzan con el encabezado de la lista de distribución MQDH, que incluye las matrices de registros MQOR y MQPMR. El encabezado de la lista de distribución puede ir seguido de datos adicionales. El formato de los datos adicionales (si los hay) viene dado por el DHFMT campo de la estructura MQDH; véase [“MQDH \(cabecera de distribución\) en IBM i”](#) en la [página 1079](#) para obtener más detalles sobre esta estructura. Los mensajes con formato FMDH se pueden convertir si se especifica la opción GMCONV en la llamada MQGET.

FMEVNT

Mensaje de evento.

El mensaje es un mensaje de evento MQ que informa de un evento que ha ocurrido. Los mensajes de evento tienen la misma estructura que los comandos programables; para obtener más información sobre esta estructura, consulte [Estructuras para comandos y respuestas](#). Para obtener información sobre eventos, consulte [Supervisión de eventos](#).

Version-1 Los mensajes de evento se pueden convertir si se especifica la opción GMCONV en la llamada MQGET.

FMIMS

IMS encabezado de información.

Los datos del mensaje comienzan con el IMS encabezado de información MQIIH, seguido de los datos de la aplicación. El nombre del formato de los datos de la aplicación viene dado por el *IIFMT* campo de la estructura MQIIH. Los mensajes de este formato se pueden convertir si se especifica la opción GMCONV en la llamada MQGET.

FMIMVS

IMS cadena variable.

El mensaje es una IMS cadena variable, que es una cadena con el formato 11zzccc, donde:

11

Es un campo de longitud de 2 bytes que especifica la longitud total del elemento de cadena IMS variable. Esta longitud es igual a la longitud de 11 (2 bytes), más la longitud de zz (2 bytes), más la longitud de la propia cadena de caracteres. 11 es un entero binario de 2 bytes en la codificación especificada por el MDENC campo.

zz

Es un campo de 2 bytes que contiene indicadores significativos para IMS. zz es una cadena de bytes que consta de dos campos de cadenas de bits de 1 byte y se transmite sin cambios del remitente al destinatario (es decir, no zz está sujeta a ninguna conversión).

ccc

es una cadena de caracteres de longitud variable que contiene 11-4 caracteres. ccc está en el conjunto de caracteres especificado por el MDCSI campo.

Los mensajes de este formato se pueden convertir si se especifica la opción GMCONV en la llamada MQGET.

FMMDE

Extensión del descriptor de mensajes.

Los datos del mensaje comienzan con la extensión del descriptor de mensaje MQMDE y, opcionalmente, van seguidos de otros datos (normalmente, los datos del mensaje de la aplicación). El nombre del formato, el conjunto de caracteres y la codificación de los datos que siguen al MQMDE se indican en los campos MEFMT, MECSI MEENC y del MQMDE. Consulte [“MQMDE \(Extensión del descriptor de mensajes\) en IBM i”](#) en la página 1177 para obtener más información sobre esta estructura. Los mensajes de este formato se pueden convertir si se especifica la opción GMCONV en la llamada MQGET.

FMPCF

Mensaje definido por el usuario en formato de comando programable (PCF).

El mensaje es un mensaje definido por el usuario que se ajusta a la estructura de un mensaje de formato de comando programable (PCF). Los mensajes de este formato se pueden convertir si se especifica la opción GMCONV en la llamada MQGET. Consulte [Uso de formatos de comandos programables](#) para obtener más información sobre el uso de mensajes con formatos de comandos programables.

FMRMH

Encabezado del mensaje de referencia.

Los datos del mensaje comienzan con el encabezado de mensaje de referencia MQRMH y, opcionalmente, van seguidos de otros datos. El nombre del formato, el juego de caracteres y la codificación de los datos se indican en los campos RMFMT, RMCSI RMENC y del MQRMH. Consulte

“MQRMH (Encabezado del mensaje de referencia) en IBM i” en la página 1226 para obtener más información sobre esta estructura. Los mensajes de este formato se pueden convertir si se especifica la opción GMCONV en la llamada MQGET.

FMRFH

Reglas y formato del encabezado.

Los datos del mensaje comienzan con las reglas y el encabezado de formato MQRFH, y opcionalmente van seguidos de otros datos. El nombre del formato, el juego de caracteres y la codificación de los datos (si los hay) se indican en los campos RFFMT, RFCSI RFENC y del MQRFH. Los mensajes de este formato se pueden convertir si se especifica la opción GMCONV en la llamada MQGET.

FMRFH2

Reglas y formato del encabezado, versión 2.

Los datos del mensaje comienzan con las reglas version-2 y el encabezado de formato MQRFH2, y opcionalmente van seguidos de otros datos. El nombre del formato, el juego de caracteres y la codificación de los datos opcionales (si los hay) se indican en los campos RF2FMT, RF2CSI RF2ENC y del campo MQRFH2. Los mensajes de este formato se pueden convertir si se especifica la opción GMCONV en la llamada MQGET.

FMSTR

Mensaje compuesto íntegramente por caracteres.

Los datos del mensaje de la aplicación pueden ser una cadena SBCS (conjunto de caracteres de un solo byte) o una cadena DBCS (conjunto de caracteres de doble byte). Los mensajes de este formato se pueden convertir si se especifica la opción GMCONV en la llamada MQGET.

FMTM

Mensaje desencadenante.

El mensaje es un mensaje desencadenante, descrito por la estructura MQTM; consulte “MQTM - Mensaje de activación” en la página 1263 para obtener más detalles sobre esta estructura. Los mensajes de este formato se pueden convertir si se especifica la opción GMCONV en la llamada MQGET.

¿FMWIH?

Cabecera de información laboral.

Los datos del mensaje comienzan con el encabezado de información de trabajo MQWIH, seguido de los datos de la aplicación. El nombre del formato de los datos de la aplicación viene dado por el WIFMT campo de la estructura MQWIH.

FMXQH

Cabecera de la cola de transmisión.

Los datos del mensaje comienzan con el encabezado de la cola de transmisión MQXQH. Los datos del mensaje original siguen inmediatamente a la estructura MQXQH. El nombre del formato de los datos del mensaje original viene dado por el MDFMT campo de la estructura MQMD, que forma parte de la cabecera de la cola de transmisión MQXQH. Consulte “MQXQH (encabezado de cola de transmisión) en IBM i” en la página 1273 para obtener más información sobre esta estructura.

Los informes COA y COD no se generan para los mensajes que tienen un MDFMT de FMXQH.

Este es un campo de salida para la llamada MQGET y un campo de entrada para las llamadas MQPUT y MQPUT1. La longitud de este campo viene dada por LNFMT. El valor inicial de este campo es FMNONE.

MDGID (cadena de bits de 24 bytes)

Identificador de grupo.

Se trata de una cadena de bytes que se utiliza para identificar el grupo de mensajes concreto o el mensaje lógico al que pertenece el mensaje físico. MDGID también se utiliza si se permite la

segmentación del mensaje. En todos estos casos, MDGID tiene un valor distinto de nulo y se establece uno o varios de los siguientes indicadores en el MDMFL campo :

- MFMIG
- MFLMIG
- MFSEG
- MFLSEG
- MFSEGA

Si no se establece ninguna de estas banderas, MDGID tiene el valor nulo especial GINONE.

Este campo no necesita ser establecido por la aplicación en la llamada MQPUT o MQGET si:

- En la llamada MQPUT, se especifica PMLOGO.
- En la llamada MQGET, no se especifica MOGRPI.

Considere utilizar estas llamadas para mensajes que no sean mensajes de informe. Sin embargo, si la aplicación requiere más control, o si la llamada es de tipo « MQPUT1 », la aplicación debe asegurarse de que MDGID se establezca en un valor adecuado.

Los grupos y segmentos de mensajes solo se pueden procesar correctamente si el identificador del grupo es único. Por este motivo, las aplicaciones no deben generar sus propios identificadores de grupo; en su lugar, deben realizar una de las siguientes acciones:

- Si se especifica PMLOGO, el gestor de colas genera automáticamente un identificador de grupo único para el primer mensaje del grupo o segmento del mensaje lógico, y utiliza ese identificador de grupo para los mensajes restantes del grupo o segmentos del mensaje lógico, por lo que la aplicación no necesita realizar ninguna acción especial. Considere utilizar este procedimiento.
- Si no se especifica PMLOGO, la aplicación debe solicitar al gestor de colas que genere el identificador de grupo, estableciendo MDGID en GINONE en la primera llamada MQPUT o MQPUT1 para un mensaje en el grupo o segmento del mensaje lógico. El identificador de grupo devuelto por el gestor de colas en la salida de esa llamada debe utilizarse entonces para los mensajes restantes del grupo o segmentos del mensaje lógico. Si un grupo de mensajes contiene mensajes segmentados, se debe utilizar el mismo identificador de grupo para todos los segmentos y mensajes del grupo.

Cuando no se especifica PMLOGO, los mensajes de los grupos y segmentos de mensajes lógicos se pueden colocar en cualquier orden (por ejemplo, en orden inverso), pero el identificador de grupo debe ser asignado por la primera llamada MQPUT o MQPUT1 que se emita para cualquiera de esos mensajes.

En las llamadas de entrada a MQPUT y MQPUT1, el gestor de colas utiliza el valor detallado en PMOPT. En la salida de las llamadas MQPUT y MQPUT1, el gestor de colas establece este campo en el valor que se envió con el mensaje si el objeto abierto es una cola única y no una lista de distribución, pero lo deja sin cambios si el objeto abierto es una lista de distribución. En este último caso, si la aplicación necesita conocer los identificadores de grupo generados, deberá proporcionar registros MQPMR que contengan el PRGID campo.

Al introducir la llamada MQGET, el gestor de colas utiliza el valor detallado en la Tabla 1. En la salida de la llamada MQGET, el gestor de colas establece este campo con el valor del mensaje recuperado.

Se define el siguiente valor especial:

GINONE

No se ha especificado ningún identificador de grupo.

El valor es un cero binario para la longitud del campo. Este es el valor que se utiliza para los mensajes que no están en grupos, ni son segmentos de mensajes lógicos, y para los que no se permite la segmentación.

La longitud de este campo viene determinada por LNGID. El valor inicial de este campo es GINONE. Este campo se ignora si MDVER es menor que MDVER2.

MDMFL (entero con signo de 10 dígitos)

Indicadores de mensajes.

Son indicadores que especifican los atributos del mensaje o controlan su procesamiento. Las banderas se dividen en las siguientes categorías:

- Indicador de segmentación
- Indicadores de estado

Estos se describen a continuación.

Indicadores de segmentación : cuando un mensaje es demasiado grande para una cola, el intento de colocarlo en la cola suele fallar. La segmentación es una técnica mediante la cual el gestor de colas o la aplicación divide el mensaje en partes más pequeñas denominadas segmentos y coloca cada segmento en la cola como un mensaje físico independiente. La aplicación que recupera el mensaje puede recuperar los segmentos uno por uno o solicitar al gestor de colas que vuelva a ensamblar los segmentos en un único mensaje que se devuelve mediante la llamada MQGET. Esto último se consigue especificando la opción GMCMPM en la llamada MQGET y proporcionando un búfer lo suficientemente grande como para albergar el mensaje completo. (Consulte [“MQGMO \(Opciones de obtención de mensajes\) activado IBM i”](#) en la [página 1096](#) para obtener más información sobre la opción GMCMPM) La segmentación de un mensaje puede producirse en el gestor de colas de envío, en un gestor de colas intermedio o en el gestor de colas de destino.

Puede especificar una de las siguientes opciones para controlar la segmentación de un mensaje:

MFSEGI

Segmentación inhibida.

Esta opción evita que el mensaje sea dividido en segmentos por el gestor de colas. Si se especifica para un mensaje que ya es un segmento, esta opción evita que el segmento se divida en segmentos más pequeños.

El valor de esta bandera es cero binario. Este es el valor predeterminado.

MFSEGA

Segmentación permitida.

Esta opción permite que el mensaje sea dividido en segmentos por el gestor de colas. Si se especifica para un mensaje que ya es un segmento, esta opción permite dividir el segmento en segmentos más pequeños. MFSEGA se puede configurar sin necesidad de configurar MFSEG o MFLSEG.

Cuando el gestor de colas segmenta un mensaje, activa el indicador MFSEG en la copia del MQMD que se envía con cada segmento, pero no modifica la configuración de estos indicadores en el MQMD proporcionado por la aplicación en la llamada MQPUT o MQPUT1. Para el último segmento del mensaje lógico, el gestor de colas también activa el indicador MFLSEG en el MQMD que se envía con el segmento.

Nota: Se debe tener cuidado cuando se envían mensajes con MFSEGA pero sin PMLOGO. Si el mensaje es:

- No es un segmento, y
- No en grupo, y
- No se reenvía,

La aplicación debe recordar restablecer el MDGID campo a GINONE antes de cada llamada a MQPUT o MQPUT1, con el fin de que el gestor de colas genere un identificador de grupo único para cada mensaje. Si no se hace así, mensajes no relacionados podrían terminar sin querer con el mismo identificador de grupo, lo que podría llevar a un procesamiento incorrecto después. Consulte las descripciones del campo MDGID y la opción PMLOGO para obtener más información sobre cuándo se debe restablecer el MDGID campo.

El gestor de colas divide los mensajes en segmentos según sea necesario para garantizar que los segmentos (más cualquier dato de encabezado que pueda ser necesario) quepan en la cola.

Sin embargo, existe un límite mínimo para el tamaño de un segmento generado por el gestor de colas, y solo el último segmento creado a partir de un mensaje puede ser inferior a este límite. (El límite inferior para el tamaño de un segmento generado por una aplicación es de un byte) Los segmentos generados por el gestor de colas pueden tener longitudes desiguales. El gestor de colas procesa el mensaje de la siguiente manera:

- Los formatos definidos por el usuario se dividen en límites que son múltiplos de 16 bytes. Esto significa que el gestor de colas no generará segmentos menores de 16 bytes (excepto el último segmento).
- Los formatos integrados distintos de FMSTR se dividen en puntos adecuados según la naturaleza de los datos presentes. Sin embargo, el gestor de colas nunca divide un mensaje en medio de una estructura de encabezado e MQ. Esto significa que un segmento que contiene una única estructura de encabezado MQ no puede ser dividido por el gestor de colas y, como resultado, el tamaño mínimo posible del segmento para ese mensaje es superior a 16 bytes.

El segundo segmento o los siguientes generados por el gestor de colas comenzarán con uno de los siguientes:

- Estructura de encabezado « MQ »
- El inicio de los datos del mensaje de la aplicación
- A mitad del mensaje de solicitud de datos
- FMSTR se divide sin tener en cuenta la naturaleza de los datos presentes (SBCS, DBCS o SBCS/DBCS mixtos). Cuando la cadena es DBCS o una mezcla de SBCS/DBCS, esto puede dar lugar a segmentos que no se pueden convertir de un juego de caracteres a otro. El gestor de colas nunca divide los mensajes FMSTR en segmentos de menos de 16 bytes (excepto el último segmento).
- Los MDENC campos MDFMT, MDCSI y del MQMD de cada segmento son establecidos por el gestor de colas para describir correctamente los datos presentes al inicio del segmento; el nombre del formato será el nombre de un formato integrado o el nombre de un formato definido por el usuario.
- El MDREP campo en el MQMD de los segmentos con MDOFF valores superiores a cero se modifica de la siguiente manera:
 - Para cada tipo de informe, si la opción del informe es RO*D, pero el segmento no puede contener ninguno de los primeros 100 bytes de datos de usuario (es decir, los datos que siguen a cualquier estructura de encabezado MQ que pueda estar presente), la opción del informe se cambia a RO*.

El gestor de colas sigue las reglas anteriores, pero por lo demás divide los mensajes de forma impredecible; no haga suposiciones sobre dónde se divide un mensaje

En el caso de los mensajes persistentes, el gestor de colas solo puede realizar la segmentación dentro de una unidad de trabajo:

- Si la llamada MQPUT o MQPUT1 se ejecuta dentro de una unidad de trabajo definida por el usuario, se utiliza dicha unidad de trabajo. Si la llamada falla durante el proceso de segmentación, el gestor de colas elimina cualquier segmento que se haya colocado en la cola como resultado de la llamada fallida. Sin embargo, el fallo no impide que la unidad de trabajo se confirme correctamente.
- Si la llamada se realiza fuera de una unidad de trabajo definida por el usuario y no existe ninguna unidad de trabajo definida por el usuario, el gestor de colas crea una unidad de trabajo solo para la duración de la llamada. Si la llamada se realiza correctamente, el gestor de colas confirma automáticamente la unidad de trabajo (la aplicación no necesita hacerlo). Si la llamada falla, el gestor de colas revierte la unidad de trabajo.
- Si la llamada se ejecuta fuera de una unidad de trabajo definida por el usuario, pero existe una unidad de trabajo definida por el usuario, el gestor de colas no puede realizar la segmentación. Si el mensaje no requiere segmentación, la llamada puede seguir realizándose correctamente. Pero si el mensaje requiere segmentación, la llamada falla con el código de motivo RC2255.

Para los mensajes no persistentes, el gestor de colas no requiere que haya una unidad de trabajo disponible para realizar la segmentación.

Se debe prestar especial atención a la conversión de datos de los mensajes que puedan estar segmentados:

- Si la conversión de datos solo la realiza la aplicación receptora en la llamada MQGET y la aplicación especifica la opción GMCMPM, la salida de conversión de datos recibirá el mensaje completo para que la salida lo convierta, y el hecho de que el mensaje se haya segmentado no será evidente para la salida.
- Si la aplicación receptora recupera un segmento cada vez, se invocará la salida de conversión de datos para convertir un segmento cada vez. Por lo tanto, la salida debe ser capaz de convertir los datos de un segmento independientemente de los datos de cualquiera de los otros segmentos.

Si la naturaleza de los datos del mensaje es tal que la segmentación arbitraria de los datos en límites de 16 bytes puede dar lugar a segmentos que no pueden ser convertidos por la salida, o si el formato es FMSTR y el conjunto de caracteres es DBCS o una mezcla de SBCS/DBCS, la propia aplicación emisora debe crear y colocar los segmentos, especificando MFSEGI para suprimir una segmentación adicional. De esta manera, la aplicación emisora puede garantizar que cada segmento contenga información suficiente para que la salida de conversión de datos pueda convertir el segmento correctamente.

- Si se especifica la conversión del remitente para un agente de canal de mensajes de envío (MCA), el MCA solo convierte los mensajes que no son segmentos de mensajes lógicos; el MCA nunca intenta convertir mensajes que son segmentos.

Esta bandera es una bandera de entrada en las llamadas MQPUT y MQPUT1, y una bandera de salida en la llamada MQGET. En la última llamada, el gestor de colas también repite el valor del indicador en el GMSEG campo de MQGMO.

El valor inicial de este indicador es MFSEGI.

Indicadores de estado : Son indicadores que señalan si el mensaje físico pertenece a un grupo de mensajes, es un segmento de un mensaje lógico, ambas cosas o ninguna. Se puede especificar uno o varios de los siguientes elementos en la llamada MQPUT o MQPUT1, o devolverlos mediante la llamada MQGET:

MFMIIG

El mensaje es miembro de un grupo.

MFLMIIG

El mensaje es el último mensaje lógico de un grupo.

Si se establece esta bandera, el gestor de colas activa MFMIIG en la copia de MQMD que se envía con el mensaje, pero no modifica la configuración de estas banderas en el MQMD proporcionado por la aplicación en la llamada MQPUT o MQPUT1.

Es válido que un grupo esté compuesto por un solo mensaje lógico. Si este es el caso, MFLMIIG está establecido, pero el MDSEQ campo tiene el valor uno.

MFSEG

El mensaje es un segmento de un mensaje lógico.

Cuando se especifica MFSEG sin MFLSEG, la longitud de los datos del mensaje de aplicación en el segmento (excluyendo las longitudes de cualquier estructura de encabezado e MQ e que pueda estar presente) debe ser al menos uno. Si la longitud es cero, la llamada a MQPUT o MQPUT1 falla con el código de motivo RC2253.

MFLSEG

El mensaje es el último segmento de un mensaje lógico.

Si se establece este indicador, el gestor de colas activa MFSEG en la copia de MQMD que se envía con el mensaje, pero no modifica la configuración de estos indicadores en el MQMD proporcionado por la aplicación en la llamada MQPUT o MQPUT1.

Es válido que un mensaje lógico conste de un solo segmento. Si este es el caso, MFLSEG se establece, pero el MD0FF campo tiene el valor cero.

Cuando se especifica MFLSEG, se permite que la longitud de los datos del mensaje de aplicación en el segmento (excluyendo las longitudes de cualquier estructura de encabezado que pueda estar presente) sea cero.

La aplicación debe garantizar que estas banderas se establezcan correctamente al enviar mensajes. Si se especifica PMLOGO, o se especificó en la llamada MQPUT anterior para el identificador de cola, la configuración de los indicadores debe ser coherente con la información de grupo y segmento conservada por el gestor de colas para el identificador de cola. Las siguientes condiciones se aplican a las llamadas MQPUT sucesivas para el identificador de cola cuando se especifica PMLOGO:

- Si no hay ningún grupo actual ni mensaje lógico, todas estas banderas (y combinaciones de ellas) son válidas.
- Una vez especificado MFMIG, debe permanecer activado hasta que se especifique MFLMIG. La llamada falla con el código de motivo RC2241 si no se cumple esta condición.
- Una vez especificado MFSEG, debe permanecer activado hasta que se especifique MFLSEG. La llamada falla con el código de motivo RC2242 si no se cumple esta condición.
- Una vez que se ha especificado MFSEG sin MFMIG, MFMIG debe permanecer desactivado hasta que se haya especificado MFLSEG. La llamada falla con el código de motivo RC2242 si no se cumple esta condición.

La [tabla 1](#) muestra las combinaciones válidas de los indicadores y los valores utilizados para diversos campos.

Estas banderas son banderas de entrada en las llamadas MQPUT y MQPUT1, y banderas de salida en la llamada MQGET. En la última llamada, el gestor de colas también repite los valores de los indicadores en los campos GMGST GMSST y de MQGMO.

Indicadores predeterminados : Se puede especificar lo siguiente para indicar que el mensaje tiene atributos predeterminados:

MFNONE

Sin indicadores de mensaje (atributos de mensaje predeterminados).

Esto inhibe la segmentación e indica que el mensaje no pertenece a ningún grupo y no es un segmento de un mensaje lógico. MFNONE se define para facilitar la documentación del programa. No está previsto que esta bandera se utilice con ninguna otra, pero como su valor es cero, dicho uso no puede detectarse.

El MDMFL campo se divide en subcampos; para más detalles, véase [“Opciones de informe y distintivos de mensaje en IBM i”](#) en la [página 1454](#).

El valor inicial de este campo es MFNONE. Este campo se ignora si MDVER es menor que MDVER2.

MDMID (cadena de bits de 24 bytes)

Identificador del mensaje.

Se trata de una cadena de bytes que se utiliza para distinguir un mensaje de otro. Por lo general, no debe haber dos mensajes con el mismo identificador de mensaje, aunque el gestor de colas no lo prohíbe. El identificador del mensaje es una propiedad permanente del mensaje y persiste tras los reinicios del gestor de colas. Dado que el identificador del mensaje es una cadena de bytes y no una cadena de caracteres, el identificador del mensaje no se convierte entre conjuntos de caracteres cuando el mensaje fluye de un gestor de colas a otro.

Para las llamadas MQPUT y MQPUT1, si la aplicación especifica MINONE o PMNMID, el gestor de colas genera un identificador de mensaje único cuando se envía el mensaje y lo coloca en el descriptor de mensaje enviado con el mensaje. El gestor de colas también devuelve este identificador de mensaje en el descriptor de mensaje perteneciente a la aplicación emisora. La aplicación puede utilizar este valor para registrar información sobre mensajes concretos y responder a consultas de otras partes de la aplicación.

Un MDMID generado por el gestor de colas consta de un identificador de producto de 4 bytes (AMQ- o CSQ- en ASCII o EBCDIC, donde - representa un único carácter en blanco), seguido de una implementación específica del producto de una cadena única. En IBM MQ esto contiene los primeros 12 caracteres del nombre del gestor de colas y un valor derivado del reloj del sistema. Por lo tanto, todos los gestores de colas que puedan intercomunicarse deben tener nombres que difieran en los primeros 12 caracteres, para garantizar que los identificadores de mensajes sean únicos. La capacidad de generar una cadena única también depende de que el reloj del sistema no se haya retrasado. Para eliminar la posibilidad de que un identificador de mensaje generado por el gestor de colas duplique uno generado por la aplicación, esta última debe evitar generar identificadores con caracteres iniciales comprendidos entre A y I en ASCII o EBCDIC (X'41' a X'49' y X' C1 ' a X' C9 '). Sin embargo, no se impide que la aplicación genere identificadores con caracteres iniciales dentro de estos rangos.

Si el mensaje se coloca en un tema, el gestor de colas genera identificadores de mensaje únicos según sea necesario para cada mensaje publicado. Si la aplicación especifica PMNMID, el gestor de colas genera un identificador de mensaje único para devolverlo en la salida. Si la aplicación especifica MINONE, el valor del MDMID campo en el MQMD no cambia al regresar de la llamada.

Consulte la descripción de PMRET en [PMOPT](#) para obtener más detalles sobre las publicaciones retenidas.

Si el mensaje se envía a una lista de distribución, el gestor de colas genera identificadores de mensaje únicos según sea necesario, pero el valor del MDMID campo en MQMD no cambia al regresar de la llamada, incluso si se especificó MINONE o PMNMID. Si la aplicación necesita conocer los identificadores de mensajes generados por el gestor de colas, la aplicación debe proporcionar registros MQPMR que contengan el PRMID campo.

La aplicación emisora también puede especificar un valor concreto para el identificador del mensaje, distinto de MINONE; esto impide que el gestor de colas genere un identificador de mensaje único. Una aplicación que reenvía un mensaje puede utilizar esta función para propagar el identificador del mensaje original.

El gestor de colas no utiliza este campo por sí mismo, salvo para:

- Genera un valor único si se solicita, tal y como se ha descrito anteriormente
- Entregar el valor a la aplicación que emite la solicitud de obtención del mensaje
- Copie el valor en el MDCID campo de cualquier mensaje de informe que genere sobre este mensaje (dependiendo de las MDREP opciones)

Cuando el gestor de colas o un agente de canal de mensajes genera un mensaje de informe, establece el MDMID campo de la forma especificada por el MDREP campo del mensaje original, ya sea RONMI o ROPMI. Las aplicaciones que generan mensajes de informe también deben hacer esto.

Para la llamada MQGET, MDMID es uno de los cinco campos que se pueden utilizar para seleccionar un mensaje concreto que se va a recuperar de la cola. Normalmente, la llamada MQGET devuelve el siguiente mensaje de la cola, pero si se necesita un mensaje concreto, se puede obtener especificando uno o varios de los cinco criterios de selección, en cualquier combinación; estos campos son:

- MDMID
- MDCID
- MDGID
- MDSEQ
- MDOFF

La aplicación establece uno o varios de estos campos con los valores requeridos y, a continuación, establece las opciones de coincidencia MO* correspondientes en el GMMO campo de MQGMO para indicar que esos campos deben utilizarse como criterios de selección. Solo los mensajes que tengan los valores especificados en esos campos son candidatos para su recuperación. El valor predeterminado para el GMMO campo (si no lo modifica la aplicación) es que coincida tanto con el identificador del mensaje como con el identificador de correlación.

Normalmente, el mensaje devuelto es el primer mensaje de la cola que cumple los criterios de selección. Pero si se especifica GMBRWN, el mensaje devuelto es el siguiente mensaje que cumple los criterios de selección; la búsqueda de este mensaje comienza con el mensaje siguiente a la posición actual del cursor.

Nota: La cola se escanea secuencialmente en busca de un mensaje que satisfaga los criterios de selección, por lo que los tiempos de recuperación serán más lentos que si no se especificaran criterios de selección, especialmente si hay que escanear muchos mensajes antes de encontrar uno adecuado.

Consulte la [Tabla 1](#) para obtener más información sobre cómo se utilizan los criterios de selección en diversas situaciones.

Especificar MINONE como identificador de mensaje tiene el mismo efecto que no especificar MOMSGI, es decir, cualquier identificador de mensaje coincidirá.

Este campo se ignora si se especifica la opción GMMUC en el **GMO** parámetro de la llamada MQGET.

Al regresar de una llamada MQGET, el MDMID campo se establece en el identificador del mensaje devuelto (si lo hay).

Se puede utilizar el siguiente valor especial:

MINONE

No se ha especificado ningún identificador de mensaje.

El valor es un cero binario para la longitud del campo.

Este es un campo de entrada/salida para las llamadas MQGET, MQPUT y MQPUT1. La longitud de este campo viene dada por LNMID. El valor inicial de este campo es MINONE.

MDMT (entero con signo de 10 dígitos)

Tipo de mensaje.

Esto indica el tipo de mensaje. Los tipos de mensajes se agrupan de la siguiente manera:

MTSFST

Valor más bajo para los tipos de mensajes definidos por el sistema.

MTSLST

Valor más alto para los tipos de mensajes definidos por el sistema.

Los siguientes valores están definidos actualmente dentro del rango del sistema:

MTDGRM

Mensaje que no requiere respuesta.

El mensaje es uno que no requiere una respuesta.

MTRQST

Mensaje que requiere respuesta.

El mensaje es uno que requiere una respuesta.

El nombre de la cola a la que se debe enviar la respuesta debe especificarse en el MDRQ campo. El MDREP campo indica cómo se deben configurar el MDMID y MDCID el de la respuesta.

MTRPLY

Responder a un mensaje de solicitud anterior.

El mensaje es la respuesta a un mensaje de solicitud anterior (MTRQST). El mensaje debe enviarse a la cola indicada por el MDRQ campo del mensaje de solicitud. El MDREP campo de la solicitud debe utilizarse para controlar cómo se establecen el MDMID y MDCID el de la respuesta.

Nota: El gestor de colas no impone la relación solicitud-respuesta; esta es una responsabilidad de la aplicación.

MTRPRT

Denunciar mensaje.

El mensaje informa sobre algún suceso esperado o inesperado, normalmente relacionado con algún otro mensaje (por ejemplo, se ha recibido un mensaje de solicitud que contenía datos no válidos). El mensaje debe enviarse a la cola indicada por el MDRQ campo del descriptor del mensaje original. El MDFB campo debe configurarse para indicar la naturaleza del informe. El MDREP campo del mensaje original se puede utilizar para controlar cómo se deben configurar los campos MDMID y MDCID del mensaje del informe.

Los mensajes de informe generados por el gestor de colas o el agente del canal de mensajes siempre se envían a la MDRQ cola, con los campos MDFB MDCID y configurados como se ha descrito anteriormente.

En futuras versiones de la biblioteca de funciones de gestión de mensajes (MQI) podrían definirse otros valores dentro del rango del sistema, los cuales serán aceptados sin error por las llamadas `MQPUT` y `MQPUT1`.

También se pueden utilizar valores definidos por la aplicación. Deben estar dentro del siguiente rango:

MTAFST

Valor más bajo para los tipos de mensajes definidos por la aplicación.

MTALST

Valor más alto para los tipos de mensajes definidos por la aplicación.

Para las llamadas MQPUT y MQPUT1, el MDMT valor debe estar dentro del rango definido por el sistema o del rango definido por la aplicación; si no es así, la llamada falla con el código de motivo RC2029.

Este es un campo de salida para la llamada MQGET y un campo de entrada para las llamadas MQPUT y MQPUT1. El valor inicial de este campo es MTDGRM.

MDOFF (entero con signo de 10 dígitos)

Desplazamiento de datos en el mensaje físico desde el inicio del mensaje lógico.

Es el desplazamiento en bytes de los datos del mensaje físico desde el inicio del mensaje lógico del que forman parte los datos. Estos datos se denominan *segmento*. El desplazamiento está comprendido entre 0 y 999 999 999. Un mensaje físico que no es un segmento de un mensaje lógico tiene un desplazamiento de cero.

Este campo no necesita ser establecido por la aplicación en la llamada MQPUT o MQGET si:

- En la llamada MQPUT, se especifica PMLOGO.
- En la llamada MQGET, no se especifica MOOFFS.

Estas son las formas recomendadas de utilizar estas llamadas para mensajes que no son mensajes de informe. Sin embargo, si la aplicación no cumple estas condiciones, o si la llamada es de tip MQPUT1, la aplicación debe asegurarse de que MDOFF se establece en un valor adecuado.

En las llamadas de entrada a MQPUT y MQPUT1, el gestor de colas utiliza el valor detallado en [la Tabla 1](#). En la salida de las llamadas MQPUT y MQPUT1, el gestor de colas establece este campo con el valor que se envió con el mensaje.

Para un mensaje de informe que informa sobre un segmento de un mensaje lógico, el MDOLN campo (siempre que no sea OLUNDF) se utiliza para actualizar el desplazamiento en la información del segmento conservada por el gestor de colas.

Al introducir la llamada MQGET, el gestor de colas utiliza el valor detallado en [la Tabla 1](#). En la salida de la llamada MQGET, el gestor de colas establece este campo con el valor del mensaje recuperado.

El valor inicial de este campo es cero. Este campo se ignora si MDVER es menor que MDVER2.

MDOLN (entero con signo de 10 dígitos)

Longitud del mensaje original.

Este campo solo es relevante para los mensajes de informe que son segmentos. Especifica la longitud del segmento del mensaje al que se refiere el mensaje de informe; no especifica la longitud del mensaje lógico del que forma parte el segmento, ni la longitud de los datos del mensaje de informe.

Nota: Al generar un mensaje de informe para un mensaje que es un segmento, el gestor de colas y el agente del canal de mensajes copian en el MQMD para el mensaje de informe los campos MDGID, MDSEQ, MDOFF y MDMFL del mensaje original. Como resultado, el mensaje del informe también es un segmento. Se recomienda que las aplicaciones que generan mensajes de informe hagan lo mismo y se aseguren de que el MDOLN campo esté configurado correctamente.

Se define el siguiente valor especial:

OLUNDF

Longitud original del mensaje no definida.

MDOLN es un campo de entrada en las llamadas MQPUT y MQPUT1, pero el valor proporcionado por la aplicación solo se acepta en circunstancias particulares:

- Si el mensaje que se envía es un segmento y también es un mensaje de informe, el gestor de colas acepta el valor especificado. El valor debe ser:
 - Mayor que cero si el segmento no es el último segmento
 - No menos que cero si el segmento es el último segmento
 - No menos que la longitud de los datos presentes en el mensaje
- Si no se cumplen estas condiciones, la llamada falla con el código de motivo RC2252.
- Si el mensaje que se envía es un segmento pero no un mensaje de informe, el gestor de colas ignora el campo y utiliza en su lugar la longitud de los datos del mensaje de la aplicación.
- En todos los demás casos, el gestor de colas ignora el campo y utiliza el valor OLUNDF en su lugar.

Este es un campo de salida en la llamada MQGET.

El valor inicial de este campo es OLUNDF. Este campo se ignora si MDVER es menor que MDVER2.

MDPAN (cadena de caracteres de 28 bytes)

Nombre de la aplicación que transfiere el mensaje.

Esto forma parte del *contexto* original del mensaje. Para obtener más información sobre el contexto de los mensajes, consulte [Contexto de los mensajes](#) y [Control de la información de contexto](#).

El formato del MDPAN depende del valor de MDPAT.

Cuando este campo es establecido por el gestor de colas (es decir, para todas las opciones excepto PMSETA), se establece en un valor determinado por el entorno:

-  En z/OS, el gestor de colas utiliza:
 - Para z/OS batch, el nombre del trabajo de 8 caracteres de la tarjeta JES JOB
 - Para TSO, el identificador de usuario TSO de 7 caracteres
 - Para CICS, el applid de 8 caracteres, seguido del tranid de 4 caracteres
 - Para IMS, el identificador del sistema de IMS 8 caracteres, seguido del nombre PSB de 8 caracteres
 - Para XCF, el nombre del grupo XCF de 8 caracteres, seguido del nombre del miembro XCF de 16 caracteres
 - Para un mensaje generado por un gestor de colas, los primeros 28 caracteres del nombre del gestor de colas
 - Para colas distribuidas sin CICS, el nombre de trabajo de 8 caracteres del iniciador del canal seguido del nombre de 8 caracteres del módulo que coloca en la cola de mensajes no entregados seguido de un identificador de tarea de 8 caracteres.

- Para el MQSeriesJava procesamiento de enlaces de idioma con IBM MQ for z/OS el nombre de trabajo de 8 caracteres del espacio de direcciones creado para el z/OS UNIX System Services entorno. Normalmente, será un identificador de usuario TSO con un único carácter numérico añadido.

El nombre o los nombres se rellenan a la derecha con espacios en blanco, al igual que cualquier espacio restante en el campo. Cuando hay más de un nombre, no hay separador entre ellos.

-  En PC DOS y Windows sistemas, el gestor de colas utiliza:
 - Para una CICS solicitud, el nombre CICS de la transacción
 - Para una aplicación CICS no ejecutable, los 28 caracteres más a la derecha del nombre completo del ejecutable
-  En IBM i, el gestor de colas utiliza el nombre completo del trabajo.
-   En AIX and Linux, el gestor de colas utiliza:
 - Para una CICS solicitud, el nombre CICS de la transacción
 - Para una noCICS aplicación, los 14 caracteres más a la derecha del nombre completo del ejecutable, si está disponible para el gestor de colas, y espacios en blanco en caso contrario (por ejemplo, en AIX)
- En VSE/ESA, el gestor de colas utiliza el identificador de aplicación de 8 caracteres, seguido del identificador de transacción de 4 caracteres.

Para las llamadas MQPUT y MQPUT1, este es un campo de entrada/salida si se especifica PMSETA en el **PMO** parámetro. Cualquier información que siga a un carácter nulo dentro del campo se descarta. El carácter nulo y cualquier carácter posterior son convertidos en espacios en blanco por el gestor de colas. Si no se especifica PMSETA, este campo se ignora en la entrada y es un campo de solo salida.

Este es un campo de salida para la llamada MQGET. La longitud de este campo viene dada por LNPAN. El valor inicial de este campo es 28 caracteres en blanco.

MDPAT (entero con signo de 10 dígitos)

Tipo de aplicación que ha transferido el mensaje.

Esto forma parte del **contexto** original del mensaje. Para obtener más información sobre el contexto de los mensajes, consulte [Contexto de los mensajes](#) y [Control de la información de contexto](#).

MDPAT puede tener uno de los siguientes tipos estándar. También se pueden utilizar tipos definidos por el usuario, pero deben restringirse a valores comprendidos entre ATUFST y ATULST.

ATAIX

AIX aplicación (mismo valor que ATUNIX).

ATBRKR

el intermediario de ubicación.

ATCICS

CICS transacción.

ATCICB

CICS bridge.

ATVSE

CICS/VSE transacción.

ATDOS

IBM MQ MQI client aplicación en PC DOS.

ATDQM

Agente gestor de colas distribuidas.

ATGUAR

Aplicación Tandem Guardian (mismo valor que ATNSK).

ATIMS

IMS .

ATIMSB

IMS puente.

ATJAVA

Java.

ATMVS

MVS o aplicación TSO (mismo valor que ATZOS).

NOTA

Lotus Notes Solicitud de agente.

ATNSK

Aplicación del núcleo Tandem NonStop.

AT390

OS/390 aplicación (mismo valor que ATZOS).

AT400

IBM i .

ATQM

Gestor de colas.

ATUNIX

UNIX .

ATVOS

Aplicación Stratus VOS.

ATWIN

Aplicación Windows de 16 bits.

ATWINT

Aplicación Windows de 32 bits.

ATXCF

XCF.

ATZOS

z/OS .

ATDEF

Tipo de aplicación predeterminado.

Este es el tipo de aplicación predeterminado para la plataforma en la que se ejecuta la aplicación.

Nota: El valor de esta constante depende del entorno.

ATUNK

Tipo de aplicación desconocido.

Este valor se puede utilizar para indicar que se desconoce el tipo de aplicación, aunque exista otra información contextual.

ATUFST

Valor más bajo para el tipo de aplicación definido por el usuario.

ATULST

Valor más alto para el tipo de aplicación definido por el usuario.

También puede darse el siguiente valor especial:

ATNCON

No hay información contextual presente en el mensaje.

Este valor lo establece el gestor de colas cuando se envía un mensaje sin contexto (es decir, cuando se especifica la opción de contexto PMNOC).

Cuando se recupera un mensaje, MDPAT se puede comprobar este valor para decidir si el mensaje tiene contexto (se recomienda que nunca MDPAT se establezca en ATNCON, por parte de una aplicación que utilice PMSETA, si alguno de los otros campos de contexto no está en blanco).

ATSIB

Indica un mensaje originado en otro IBM MQ producto de mensajería y que ha llegado a través del puente SIB (Service- Integration Bus, mensajería de servicio).

Cuando el gestor de colas genera esta información como resultado de una aplicación, el campo se establece en un valor determinado por el entorno.

 Tenga en cuenta que en IBM i, el campo está configurado en AT400; el gestor de colas nunca utiliza ATCICS en IBM i.

Para las llamadas MQPUT y MQPUT1, este es un campo de entrada/salida si se especifica PMSETA en el **PMO** parámetro. Si no se especifica PMSETA, este campo se ignora en la entrada y es un campo de solo salida.

Tras la finalización satisfactoria de una llamada MQPUT o MQPUT1, este campo contiene el MDPAT que se transmitió con el mensaje si se colocó en una cola. Este será el valor de MDPAT que se conservará con el mensaje si se retiene (véase la descripción de PMRET para obtener más detalles sobre las publicaciones retenidas), pero no se utilizará como MDPAT cuando el mensaje se envíe como publicación a los suscriptores, ya que estos proporcionan un valor para anular MDPAT en todas las publicaciones que se les envían. Si el mensaje no tiene contexto, el campo se establece en ATNCON.

Este es un campo de salida para la llamada MQGET. El valor inicial de este campo es ATNCON.

MDPD (cadena de caracteres de 8 bytes)

Fecha en que se transfirió el mensaje.

Esto forma parte del *contexto* original del mensaje. Para obtener más información sobre el contexto de los mensajes, consulte [Contexto de los mensajes](#) y [Control de la información de contexto](#).

El formato utilizado para la fecha cuando este campo es generado por el gestor de colas es:

- AAAAMMDD

donde los caracteres representan:

AAAA

año (cuatro dígitos numéricos)

MM

mes del año (del 01 al 12)

DD

día del mes (del 01 al 31)

La hora del meridiano de Greenwich (GMT) se utiliza para los campos MDPD MDPT y, siempre que el reloj del sistema esté ajustado con precisión a la GMT.

Si el mensaje se incluyó como parte de una unidad de trabajo, la fecha es aquella en la que se incluyó el mensaje, y no la fecha en la que se confirmó la unidad de trabajo.

Para las llamadas MQPUT y MQPUT1, este es un campo de entrada/salida si se especifica PMSETA en el **PMO** parámetro. El gestor de colas no comprueba el contenido del campo, salvo que se descarte cualquier información que siga a un carácter nulo dentro del campo. El carácter nulo y cualquier carácter posterior son convertidos en espacios en blanco por el gestor de colas. Si no se especifica PMSETA, este campo se ignora en la entrada y es un campo de solo salida.

Tras la finalización satisfactoria de una llamada MQPUT o MQPUT1, este campo contiene el MDPD que se transmitió con el mensaje si se colocó en una cola. Este será el valor de MDPD que se conservará con el mensaje si se retiene (véase la descripción de PMRET para obtener más detalles sobre las publicaciones retenidas), pero no se utilizará como MDPD cuando el mensaje se envíe como publicación a los suscriptores, ya que estos proporcionan un valor para anular MDPD en todas las

publicaciones que se les envían. Si el mensaje no tiene contexto, el campo queda completamente en blanco.

Este es un campo de salida para la llamada MQGET. La longitud de este campo viene dada por LNPDAT. El valor inicial de este campo es 8 caracteres en blanco.

MDPER (entero con signo de 10 dígitos)

Persistencia de mensajes.

Esto indica si el mensaje sobrevive a fallos del sistema y reinicios del gestor de colas. Para las llamadas MQPUT y MQPUT1, el valor debe ser uno de los siguientes:

PEPER

El mensaje es persistente.

Esto significa que el mensaje sobrevive a los fallos del sistema y a los reinicios del gestor de colas. Una vez que se ha enviado el mensaje y se ha confirmado la unidad de trabajo del remitente (si el mensaje se envía como parte de una unidad de trabajo), el mensaje se conserva en un almacenamiento auxiliar. Permanece allí hasta que el mensaje se elimina de la cola y se confirma la unidad de trabajo del getter (si el mensaje se recupera como parte de una unidad de trabajo).

Cuando se envía un mensaje persistente a una cola remota, se utiliza un mecanismo de almacenamiento y reenvío para retener el mensaje en cada gestor de colas a lo largo de la ruta hasta el destino, hasta que se sabe que el mensaje ha llegado al siguiente gestor de colas.

Los mensajes persistentes no pueden colocarse en:

- Colas dinámicas temporales
- Colas compartidas en las que el nivel de estructura de la instalación de acoplamiento es inferior a tres, o la estructura de la instalación de acoplamiento no es recuperable.

Los mensajes persistentes se pueden colocar en colas dinámicas permanentes, colas predefinidas y colas compartidas donde el nivel de estructura de la instalación de acoplamiento es 3 y la instalación de acoplamiento es recuperable.

PENPER

El mensaje no es persistente.

Esto significa que el mensaje normalmente no sobrevive a fallos del sistema o reinicios del gestor de colas. Esto se aplica incluso si se encuentra una copia intacta del mensaje en el almacenamiento auxiliar durante el reinicio del gestor de colas.

En el caso especial de las colas compartidas, los mensajes no persistentes *sobreviven* al reinicio de los gestores de colas del grupo de intercambio de colas, pero no sobreviven a los fallos del dispositivo de acoplamiento utilizado para almacenar mensajes en las colas compartidas.

PEQDEF

El mensaje tiene persistencia predeterminada.

- Si la cola es una cola de clúster, la persistencia del mensaje se toma del **DefPersistence** atributo definido en el gestor de colas de destino que posee la instancia concreta de la cola en la que se coloca el mensaje. Por lo general, todas las instancias de una cola de clúster tienen el mismo valor para el **DefPersistence** atributo, aunque esto no es obligatorio.

El valor de **DefPersistence** se copia en el **MDPER** campo cuando el mensaje se coloca en la cola de destino. Si **DefPersistence** se cambia posteriormente, los mensajes que ya se han colocado en la cola no se ven afectados.

- Si la cola no es una cola de clúster, la persistencia del mensaje se toma del **DefPersistence** atributo definido en el gestor de colas local, incluso si el gestor de colas de destino es remoto.

Si hay más de una definición en la ruta de resolución del nombre de la cola, la persistencia predeterminada se toma del valor de este atributo en la primera definición de la ruta. Puede ser lo siguiente:

- una cola alias

- Una cola local
- Una definición local de una cola remota
- Un alias del gestor de colas
- Una cola de transmisión (por ejemplo, la DefXmitQName cola)

El valor de **DefPersistence** se copia en el MDPER campo cuando se envía el mensaje. Si **DefPersistence** se modifica posteriormente, los mensajes que ya se hayan enviado no se verán afectados.

En una misma cola pueden coexistir mensajes persistentes y no persistentes.

Al responder a un mensaje, las aplicaciones deben utilizar normalmente para el mensaje de respuesta la persistencia del mensaje de solicitud.

Para una llamada MQGET, el valor devuelto es PEPER o PENPER.

Este es un campo de salida para la llamada MQGET y un campo de entrada para las llamadas MQPUT y MQPUT1. El valor inicial de este campo es PEQDEF.

MDPRI (entero con signo de 10 dígitos)

Prioridad del mensaje.

Para las llamadas MQPUT y MQPUT1, el valor debe ser mayor o igual a cero; cero es la prioridad más baja. También se puede utilizar el siguiente valor especial:

PRQDEF

Prioridad predeterminada para la cola.

- Si la cola es una cola de clúster, la prioridad del mensaje se toma del **DefPriority** atributo definido en el gestor de colas de destino que posee la instancia concreta de la cola en la que se coloca el mensaje. Por lo general, todas las instancias de una cola de clúster tienen el mismo valor para el **DefPriority** atributo, aunque esto no es obligatorio.

El valor de **DefPriority** se copia en el MDPRI campo cuando el mensaje se coloca en la cola de destino. Si **DefPriority** se cambia posteriormente, los mensajes que ya se han colocado en la cola no se ven afectados.

- Si la cola no es una cola de clúster, la prioridad del mensaje se toma del **DefPriority** atributo definido en el gestor de colas local, incluso si el gestor de colas de destino es remoto.

Si hay más de una definición en la ruta de resolución del nombre de la cola, la prioridad predeterminada se toma del valor de este atributo en la primera definición de la ruta. Puede ser lo siguiente:

- una cola alias
- Una cola local
- Una definición local de una cola remota
- Un alias del gestor de colas
- Una cola de transmisión (por ejemplo, la DefXmitQName cola)

El valor de **DefPriority** se copia en el MDPRI campo cuando se envía el mensaje. Si **DefPriority** se modifica posteriormente, los mensajes que ya se hayan enviado no se verán afectados.

El valor devuelto por la llamada MQGET siempre es mayor o igual a cero; nunca se devuelve el valor PRQDEF.

Si se envía un mensaje con una prioridad superior al máximo admitido por el gestor de colas local (este máximo viene dado por el **MaxPriority** atributo del gestor de colas), el mensaje es aceptado por el gestor de colas, pero se coloca en la cola con la prioridad máxima del gestor de colas; la llamada MQPUT o MQPUT1 se completa con CCWARN y el código de motivo RC2049. Sin embargo, el MDPRI campo conserva el valor especificado por la aplicación que envió el mensaje.

Al responder a un mensaje, las aplicaciones deben utilizar normalmente para el mensaje de respuesta la prioridad del mensaje de solicitud. En otras situaciones, especificar PRQDEF permite realizar el ajuste de prioridad sin cambiar la aplicación.

Este es un campo de salida para la llamada MQGET y un campo de entrada para las llamadas MQPUT y MQPUT1. El valor inicial de este campo es PRQDEF.

MDPT (cadena de caracteres de 8 bytes)

Hora en que se transfirió el mensaje.

Esto forma parte del **contexto** original del mensaje. Para obtener más información sobre el contexto de los mensajes, consulte [Contexto de los mensajes](#) y [Control de la información de contexto](#).

El formato utilizado para la hora en que este campo es generado por el gestor de colas es:

- HHMMSSSTH

donde los caracteres representan (en orden):

HH

horas (de 00 a 23)

MM

minutos (de 00 a 59)

SS

segundos (de 00 a 59; véase [la nota](#))

T

décimas de segundo (0 a 9)

H

centésimas de segundo (0 a 9)

Nota: Si el reloj del sistema está sincronizado con un estándar de tiempo muy preciso, en raras ocasiones es posible que se devuelva 60 o 61 para los segundos en MDPT. Esto ocurre cuando se insertan segundos intercalares en el estándar horario global.

La hora del meridiano de Greenwich (GMT) se utiliza para los campos MDPD MDPT y, siempre que el reloj del sistema esté ajustado con precisión a la GMT.

Si el mensaje se incluyó como parte de una unidad de trabajo, la hora es la hora en que se incluyó el mensaje, y no la hora en que se confirmó la unidad de trabajo.

Para las llamadas MQPUT y MQPUT1, este es un campo de entrada/salida si se especifica PMSETA en el **PMO** parámetro. El gestor de colas no comprueba el contenido del campo, salvo que se descarte cualquier información que siga a un carácter nulo dentro del campo. El carácter nulo y cualquier carácter posterior son convertidos en espacios en blanco por el gestor de colas. Si no se especifica PMSETA, este campo se ignora en la entrada y es un campo de solo salida.

Tras la finalización satisfactoria de una llamada MQPUT o MQPUT1, este campo contiene el MDPT valor que se transmitió con el mensaje si se colocó en una cola. Este será el valor de MDPT que se conservará con el mensaje si se retiene (véase la descripción de PMRET para obtener más detalles sobre las publicaciones retenidas), pero no se utilizará como MDPT cuando el mensaje se envíe como publicación a los suscriptores, ya que estos proporcionan un valor para anular MDPT en todas las publicaciones que se les envían. Si el mensaje no tiene contexto, el campo queda completamente en blanco.

Este es un campo de salida para la llamada MQGET. La longitud de este campo viene dada por LNPTIM. El valor inicial de este campo es 8 caracteres en blanco.

MDREP (entero con signo de 10 dígitos)

Opciones para los mensajes de informe.

Un mensaje de informe es un mensaje sobre otro mensaje, utilizado para informar a una aplicación sobre eventos esperados o inesperados relacionados con el mensaje original. El MDREP campo permite a la aplicación que envía el mensaje original especificar qué mensajes de informe se

requieren, si los datos del mensaje de la aplicación deben incluirse en ellos y también (tanto para los informes como para las respuestas) cómo deben configurarse los identificadores de mensaje y correlación en el mensaje de informe o respuesta. Se puede solicitar cualquiera o todos (o ninguno) de los siguientes tipos de mensajes de informe:

- Excepción
- Caducidad
- Confirmar a la llegada (COA)
- Confirmar en el momento de la entrega (COD)
- notificación de acción positiva (PAN)
- notificación de acción negativa (NAN)

Si se requiere más de un tipo de mensaje de informe, o se necesitan otras opciones de informe, los valores se pueden sumar (no añada la misma constante más de una vez).

La aplicación que recibe el mensaje de informe puede determinar el motivo por el que se generó el informe examinando el MDFB campo en el MQMD; consulte el MDFB campo para obtener más detalles.

El uso de opciones de informe al enviar un mensaje a un tema puede provocar que se generen y envíen a la aplicación cero, uno o muchos mensajes de informe. Esto se debe a que el mensaje de publicación puede enviarse a cero, una o muchas aplicaciones suscritas.

Opciones de excepción : Puede especificar una de las siguientes opciones para solicitar un mensaje de informe de excepción.

REACTIVIDAD

Informes de actividad requeridos

Esta opción de informe permite generar un informe de actividad cada vez que las aplicaciones compatibles procesan un mensaje con esta opción de informe configurada.

Los mensajes con esta opción de informe activada deben ser aceptados por cualquier gestor de colas, incluso si no «entienden» la opción. Esto permite configurar la opción de informe en cualquier mensaje de usuario, incluso si son procesados por gestores de colas anteriores. Para lograrlo, la opción de informe se coloca en el subcampo ROAUM.

Si un proceso (ya sea un gestor de colas o un proceso de usuario) realiza una actividad en un mensaje con ROACT establecido, puede optar por generar y enviar un informe de actividad.

La opción de informe de actividad permite rastrear la ruta de cualquier mensaje a través de una red de gestores de colas. La opción de informe se puede especificar en cualquier mensaje de usuario actual y, al instante, se puede comenzar a calcular la ruta del mensaje a través de la red. Si la aplicación que genera el mensaje no puede habilitar la generación de informes de actividad, se puede habilitar utilizando una salida de cruce de API proporcionada por los administradores del gestor de colas.

Se aplican varias condiciones a los informes de actividad:

1. La ruta será menos detallada si hay menos gestores de colas en la red capaces de generar informes de actividad.
2. Es posible que los informes de actividad no sean fácilmente «ordenables» para determinar la ruta seguida.
3. Es posible que los informes de actividad no puedan encontrar una ruta hasta el destino solicitado.

ROEXC

Se requieren informes de excepciones.

Este tipo de informe puede ser generado por un agente de canal de mensajes cuando se envía un mensaje a otro gestor de colas y el mensaje no puede entregarse a la cola de destino especificada. Por ejemplo, la cola de destino o una cola de transmisión intermedia podrían estar llenas, o el mensaje podría ser demasiado grande para la cola.

La generación del mensaje del informe de excepción depende de la persistencia del mensaje original y de la velocidad del canal de mensajes (normal o rápido) por el que viaja el mensaje original:

- Para todos los mensajes persistentes y para los mensajes no persistentes que viajan a través de canales de mensajes normales, el informe de excepciones solo se genera si la acción especificada por la aplicación de envío para la condición de error se puede completar correctamente. La aplicación emisora puede especificar una de las siguientes acciones para controlar la disposición del mensaje original cuando se produce la condición de error:
 - RODLQ (esto hace que el mensaje original se coloque en la cola de mensajes no entregados).
 - RODISC (esto hace que se descarte el mensaje original).

Si la acción especificada por la aplicación emisora no se puede completar correctamente, el mensaje original permanece en la cola de transmisión y no se genera ningún mensaje de informe de excepción.

- En el caso de los mensajes no persistentes que viajan a través de canales de mensajes rápidos, el mensaje original se elimina de la cola de transmisión y se genera el informe de excepción, incluso si la acción especificada para la condición de error no se puede completar correctamente. Por ejemplo, si se especifica RODLQ, pero el mensaje original no se puede colocar en la cola de mensajes no entregados porque (por ejemplo) esa cola está llena, se genera el mensaje de informe de excepción y se descarta el mensaje original.

Consulte [Persistencia de mensajes](#) para obtener más información sobre los canales de mensajes normales y rápidos.

No se genera un informe de excepción si la aplicación que envió el mensaje original puede ser notificada de forma sincrónica del problema mediante el código de motivo devuelto por la llamada MQPUT o MQPUT1.

Las aplicaciones también pueden enviar informes de excepción para indicar que un mensaje que han recibido no puede procesarse (por ejemplo, porque se trata de una transacción de débito que haría que la cuenta superara su límite de crédito).

Los datos del mensaje original no se incluyen en el mensaje del informe.

No especifique más de uno de ROEXC, ROEXCD y ROEXCF.

ROEXCD

Informes de excepciones con datos requeridos.

Es igual que ROEXC, salvo que los primeros 100 bytes de los datos del mensaje de aplicación del mensaje original se incluyen en el mensaje de informe. Si el mensaje original contiene una o más estructuras de encabezado MQ, estas se incluyen en el mensaje de informe, además de los 100 bytes de datos de la aplicación.

No especifique más de uno de ROEXC, ROEXCD y ROEXCF.

ROEXCF

Informes de excepciones con todos los datos necesarios.

Es igual que ROEXC, salvo que todos los datos del mensaje de aplicación del mensaje original se incluyen en el mensaje de informe.

No especifique más de uno de ROEXC, ROEXCD y ROEXCF.

Opciones de caducidad : puede especificar una de las siguientes opciones para solicitar un mensaje de informe de caducidad.

ROEXP

Se requieren informes de vencimiento.

Este tipo de informe es generado por el gestor de colas si el mensaje se descarta antes de su entrega a una aplicación porque ha superado su tiempo de caducidad (véase el MDEXP campo).

Si esta opción no está activada, no se generará ningún mensaje de informe si se descarta un mensaje por este motivo (incluso si se ha especificado una de las opciones ROEXC*).

Los datos del mensaje original no se incluyen en el mensaje del informe.

No especifique más de uno de ROEXP, ROEXPD y ROEXPF.

ROEXPD

Informes de caducidad con los datos requeridos.

Es igual que ROEXP, salvo que los primeros 100 bytes de los datos del mensaje de aplicación del mensaje original se incluyen en el mensaje de informe. Si el mensaje original contiene una o más estructuras de encabezado MQ, estas se incluyen en el mensaje de informe, además de los 100 bytes de datos de la aplicación.

No especifique más de uno de ROEXP, ROEXPD y ROEXPF.

ROEXPF

Se requieren informes de vencimiento con datos completos.

Es igual que ROEXP, salvo que todos los datos del mensaje de aplicación del mensaje original se incluyen en el mensaje de informe.

No especifique más de uno de ROEXP, ROEXPD y ROEXPF.

Opciones de confirmación a la llegada : puede especificar una de las siguientes opciones para solicitar un mensaje de informe de confirmación a la llegada.

ROCOA

Se requieren informes de confirmación a la llegada.

Este tipo de informe lo genera el gestor de colas propietario de la cola de destino cuando el mensaje se coloca en dicha cola. Los datos del mensaje original no se incluyen en el mensaje del informe.

Si el mensaje se incluye como parte de una unidad de trabajo y la cola de destino es una cola local, el mensaje de informe COA generado por el gestor de colas solo estará disponible para su recuperación si se confirma la unidad de trabajo.

No se genera un informe COA si el MDFMT campo del descriptor de mensaje es FMXQH o FMDLH. Esto evita que se genere un informe COA si el mensaje se coloca en una cola de transmisión o si no se puede entregar y se coloca en una cola de mensajes no entregados.

No especifique más de uno de ROCOA, ROCOAD y ROCOAF.

ROCOAD

Confirmar los informes a la llegada con los datos requeridos.

Es igual que ROCOA, salvo que los primeros 100 bytes de los datos del mensaje de aplicación del mensaje original se incluyen en el mensaje de informe. Si el mensaje original contiene una o más estructuras de encabezado MQ, estas se incluyen en el mensaje de informe, además de los 100 bytes de datos de la aplicación.

No especifique más de uno de ROCOA, ROCOAD y ROCOAF.

ROCOAF

Confirmar los informes a la llegada con todos los datos requeridos.

Es igual que ROCOA, salvo que todos los datos del mensaje de aplicación del mensaje original se incluyen en el mensaje de informe.

No especifique más de uno de ROCOA, ROCOAD y ROCOAF.

Opciones de descarte y caducidad : puede especificar la siguiente opción para establecer el tiempo de caducidad y el indicador de descarte para los mensajes de informe.

ROPDAE

Establecer el tiempo de caducidad del mensaje de informe y la bandera de descarte.

Esta opción garantiza que los mensajes de informe y los mensajes de respuesta hereden el tiempo de caducidad y el indicador de descarte (si se debe descartar o no) de sus mensajes originales. Con esta opción activada, los mensajes de informe y respuesta:

1. Heredar la bandera RODISC (si estaba establecida).
2. Heredar el tiempo de caducidad restante del mensaje, si el mensaje no es un informe de caducidad. Si el mensaje es un informe de caducidad, el tiempo de caducidad se establece en 60 segundos.

Con esta opción activada, se aplica lo siguiente:

Nota:

1. Los mensajes de informe y respuesta se generan con un indicador de descarte y un valor de caducidad, y no pueden permanecer en el sistema.
2. Se impide que los mensajes de ruta de seguimiento lleguen a las colas de destino en los gestores de colas que no tienen habilitada la ruta de seguimiento.
3. Se evita que las colas se llenen de informes que no se pueden entregar si se interrumpen los enlaces de comunicación.
4. Las respuestas del servidor de comandos heredan el tiempo de caducidad restante de la solicitud.

Opciones de confirmación en la entrega : Puede especificar una de las siguientes opciones para solicitar un mensaje de informe de confirmación en la entrega.

ROCOD

Se requieren informes de confirmación de entrega.

Este tipo de informe lo genera el gestor de colas cuando una aplicación recupera el mensaje de la cola de destino de tal forma que el mensaje se elimina de la cola. Los datos del mensaje original no se incluyen en el mensaje del informe.

Si el mensaje se recupera como parte de una unidad de trabajo, el mensaje de informe se genera dentro de la misma unidad de trabajo, por lo que el informe no estará disponible hasta que se confirme la unidad de trabajo. Si se cancela la unidad de trabajo, no se envía el informe.

No se genera un informe COD si el MDFMT campo del descriptor del mensaje es FMDLH. Esto evita que se genere un informe COD si el mensaje no se puede entregar y se coloca en una cola de mensajes no entregados.

ROCOD no es válido si la cola de destino es una cola XCF.

No especifique más de uno de ROCOD, ROCODD y ROCODF.

ROCODD

Confirmar los informes de entrega con los datos requeridos.

Es igual que ROCOD, salvo que los primeros 100 bytes de los datos del mensaje de aplicación del mensaje original se incluyen en el mensaje de informe. Si el mensaje original contiene una o más estructuras de encabezado MQ, estas se incluyen en el mensaje de informe, además de los 100 bytes de datos de la aplicación.

Si se especifica GMATM en la llamada MQGET para el mensaje original y el mensaje recuperado está truncado, la cantidad de datos del mensaje de aplicación que se coloca en el mensaje de informe es el mínimo de:

- La longitud del mensaje original
- 100 bytes.

ROCODD no es válido si la cola de destino es una cola XCF.

No especifique más de uno de ROCOD, ROCODD y ROCODF.

ROCODF

Confirmar los informes de entrega con todos los datos necesarios.

Es igual que ROCOD, salvo que todos los datos del mensaje de aplicación del mensaje original se incluyen en el mensaje del informe.

ROCODF no es válido si la cola de destino es una cola XCF.

No especifique más de uno de ROCOD, ROCODD y ROCODF.

Opciones de notificación de acciones : puede especificar una o ambas de las siguientes opciones para solicitar que la aplicación receptora envíe un mensaje de informe de acción positiva o negativa.

ROPAN

Se requieren informes de notificación de acciones positivas.

Este tipo de informe es generado por la aplicación que recupera el mensaje y actúa en consecuencia. Indica que la acción solicitada en el mensaje se ha realizado correctamente. La aplicación que genera el informe determina si se debe incluir algún dato en el informe.

Aparte de transmitir esta solicitud a la aplicación que recupera el mensaje, el gestor de colas no realiza ninguna acción basada en esta opción. Es responsabilidad de la aplicación de recuperación generar el informe si procede.

RONAN

Se requieren informes de notificación de acciones negativas.

Este tipo de informe es generado por la aplicación que recupera el mensaje y actúa en consecuencia. Indica que la acción solicitada en el mensaje no se ha realizado correctamente. La aplicación que genera el informe determina si se debe incluir algún dato en el informe. Por ejemplo, puede ser conveniente incluir algunos datos que indiquen por qué no se pudo realizar la solicitud.

Aparte de transmitir esta solicitud a la aplicación que recupera el mensaje, el gestor de colas no realiza ninguna acción basada en esta opción. Es responsabilidad de la aplicación de recuperación generar el informe si procede.

La determinación de qué condiciones corresponden a una acción positiva y cuáles a una acción negativa es responsabilidad de la aplicación. Sin embargo, se recomienda que, si la solicitud solo se ha realizado parcialmente, se genere un informe NAN en lugar de un informe PAN, si así se solicita. También se recomienda que cada condición posible corresponda a una acción positiva o a una acción negativa, pero no a ambas.

Opciones de identificador de mensaje : Puede especificar una de las siguientes opciones para controlar cómo se debe configurar el MDMID del mensaje del informe (o del mensaje de respuesta).

RONMI

Nuevo identificador de mensaje.

Esta es la acción predeterminada e indica que si se genera un informe o una respuesta como resultado de este mensaje, se generará MDMID un nuevo para el informe o el mensaje de respuesta.

ROPMI

Pasar identificador de mensaje.

Si se genera un informe o una respuesta como resultado de este mensaje, el MDMID de este mensaje debe copiarse en el MDMID del informe o mensaje de respuesta.

El MsgId de un mensaje de publicación será diferente para cada suscriptor que reciba una copia de la publicación y, por lo tanto, el MsgId copiado en el informe o mensaje de respuesta será diferente para cada uno.

Si no se especifica esta opción, se asume RONMI.

Opciones del identificador de correlación : Puede especificar una de las siguientes opciones para controlar cómo se debe configurar el MDCID del mensaje del informe (o del mensaje de respuesta).

ROCMTC

Copiar el identificador del mensaje al identificador de correlación.

Esta es la acción predeterminada e indica que, si se genera un informe o una respuesta como resultado de este mensaje, el MDMID de este mensaje se copiará en el MDCID del informe o mensaje de respuesta.

El MsgId de un mensaje de publicación será diferente para cada suscriptor que reciba una copia de la publicación y, por lo tanto, el MsgId copiado en el CorrelId del informe o mensaje de respuesta será diferente para cada uno.

ROPCI

Pasar identificador de correlación.

Si se genera un informe o una respuesta como resultado de este mensaje, el MDCID de este mensaje debe copiarse en el MDCID del informe o mensaje de respuesta.

El MDCID de un mensaje de publicación será específico para un suscriptor, a menos que utilice la opción SOSCID y establezca el SCDIC campo en el MQSD en CINONE. Por lo tanto, es posible que el MDCID copiado en el MDCID del informe o mensaje de respuesta sea diferente para cada uno.

Si no se especifica esta opción, se asume ROCMTC.

Se recomienda que los servidores que responden a solicitudes o generan mensajes de informe comprueben si las opciones ROPMI o ROPCI se han establecido en el mensaje original. Si fuera así, los servidores deberían tomar las medidas descritas para esas opciones. Si no se establece ninguno de los dos, los servidores deben realizar la acción predeterminada correspondiente.

: Puede especificar una de las siguientes opciones para controlar la disposición del mensaje original cuando no se puede entregar a la cola de destino. Estas opciones solo se aplican a aquellas situaciones que darían lugar a la generación de un mensaje de informe de excepción si la aplicación emisora lo hubiera solicitado. La aplicación puede configurar las opciones de disposición independientemente de los informes de excepción solicitados.

RODLQ

Colocar el mensaje en la cola de mensajes no entregados.

Esta es la acción predeterminada e indica que el mensaje debe colocarse en la cola de mensajes no entregados si no puede entregarse en la cola de destino. Esto ocurre en las siguientes situaciones:

- Cuando la aplicación que envió el mensaje original no puede ser notificada de forma sincrónica del problema mediante el código de motivo devuelto por la llamada MQPUT o MQPUT1. Se genera un mensaje de informe de excepción, si el remitente lo ha solicitado.
- Cuando la aplicación que publicó el mensaje original lo estaba colocando en un tema

Se generará un mensaje de informe de excepción, si así lo ha solicitado el remitente.

RODISC

Descartar mensaje.

Esto indica que el mensaje debe descartarse si no puede entregarse a la cola de destino. Esto ocurre en las siguientes situaciones:

- Cuando la aplicación que envió el mensaje original no puede ser notificada de forma sincrónica del problema mediante el código de motivo devuelto por la llamada MQPUT o MQPUT1. Se genera un mensaje de informe de excepción, si el remitente lo ha solicitado.
- Cuando la aplicación que publicó el mensaje original lo estaba colocando en un tema

Se generará un mensaje de informe de excepción, si así lo ha solicitado el remitente.

Si es necesario devolver el mensaje original al remitente, sin que el mensaje original se coloque en la cola de mensajes no entregados, el remitente debe especificar RODISC con ROEXCF.

Opción predeterminada : Si no se requieren opciones de informe, puede especificar lo siguiente:

RONONE

No se necesita ningún informe.

Este valor se puede utilizar para indicar que no se han especificado otras opciones. RONONE se define para ayudar en la documentación del programa. No está previsto que esta opción se utilice con ninguna otra, pero como su valor es cero, dicho uso no puede detectarse.

Información general:

1. Todos los tipos de informes requeridos deben ser solicitados específicamente por la aplicación que envía el mensaje original. Por ejemplo, si se solicita un informe COA pero no un informe de excepción, se genera un informe COA cuando el mensaje se coloca en la cola de destino, pero no se genera ningún informe de excepción si la cola de destino está llena cuando llega el mensaje. Si no se establece ninguna MDREP opción, el gestor de colas o el agente de canal de mensajes (MCA) no generan mensajes de informe.

Algunas opciones de informe se pueden especificar aunque el gestor de colas local no las reconozca; esto resulta útil cuando la opción debe ser procesada por el gestor de colas de destino. Consulte [“Opciones de informe y distintivos de mensaje en IBM i”](#) en la [página 1454](#) para obtener más detalles.

Si se solicita un mensaje de informe, se debe especificar el nombre de la cola a la que se debe enviar el informe en el MDRQ campo. Cuando se recibe un mensaje de informe, la naturaleza del informe puede determinarse examinando el MDFB campo en el descriptor del mensaje.

2. Si el gestor de colas o MCA que genera un mensaje de informe no puede colocar el mensaje de informe en la cola de respuesta (por ejemplo, porque la cola de respuesta o la cola de transmisión están llenas), el mensaje de informe se coloca en su lugar en la cola de mensajes no entregados. Si eso también falla, o no hay una cola de mensajes no entregados, la acción que se tome dependerá del tipo de mensaje del informe:
 - Si el mensaje del informe es un informe de excepción, el mensaje que provocó la generación del informe de excepción se deja en su cola de transmisión; esto garantiza que el mensaje no se pierda.
 - Para todos los demás tipos de informes, el mensaje del informe se descarta y el procesamiento continúa con normalidad. Esto se hace porque el mensaje original ya se ha entregado correctamente (en el caso de los mensajes de informe COA o COD) o porque ya no tiene interés (en el caso de los mensajes de informe de caducidad).

Una vez que un mensaje de informe se ha colocado correctamente en una cola (ya sea la cola de destino o una cola de transmisión intermedia), el mensaje ya no está sujeto a un procesamiento especial, sino que se trata como cualquier otro mensaje.

3. Cuando se genera el informe, se abre la MDRQ cola y se coloca el mensaje del informe utilizando la autoridad del MDUID en el MQMD del mensaje que provoca el informe, excepto en los siguientes casos:
 - Los informes de excepción generados por un MCA receptor se incluyen con la autoridad que utilizó el MCA cuando intentó enviar el mensaje que provocó el informe. El CDPA atributo channel determina el identificador de usuario utilizado.
 - Los informes COA generados por el gestor de colas se incluyen con la autoridad que se utilizó cuando el mensaje que provocó el informe se colocó en el gestor de colas que generó el informe. Por ejemplo, si el mensaje fue enviado por un MCA receptor utilizando el identificador de usuario del MCA, el gestor de colas envía el informe COA utilizando el identificador de usuario del MCA.

Las aplicaciones que generan informes deben utilizar normalmente la misma autoridad que habrían utilizado para generar una respuesta; normalmente, esta debería ser la autoridad del identificador de usuario del mensaje original.

Si el informe tiene que enviarse a un destino remoto, los remitentes y destinatarios pueden decidir si aceptarlo, del mismo modo que lo hacen con otros mensajes.

4. Si se solicita un mensaje de informe con datos:
 - El mensaje de informe siempre se genera con la cantidad de datos solicitada por el remitente del mensaje original. Si el mensaje del informe es demasiado grande para la cola de respuestas, se

produce el procesamiento descrito anteriormente; el mensaje del informe nunca se trunca para que quepa en la cola de respuestas.

- Si el MDFMT del mensaje original es FMXQH, los datos incluidos en el informe no incluyen el MQXQH. Los datos del informe comienzan con el primer byte de los datos más allá del MQXQH en el mensaje original. Esto ocurre independientemente de si la cola es una cola de transmisión.
5. Si se recibe un mensaje de informe COA, COD o de caducidad en la cola de respuesta, se garantiza que el mensaje original llegó, se entregó o caducó, según corresponda. Sin embargo, si se solicita uno o más de estos mensajes de informe y no se reciben, no se puede asumir lo contrario, ya que puede haber ocurrido una de las siguientes situaciones:
- a. El mensaje del informe se retrasa porque un enlace no funciona.
 - b. El mensaje de informe se retiene porque existe una condición de bloqueo en una cola de transmisión intermedia o en la cola de respuesta (por ejemplo, la cola está llena o inhibida para inserciones).
 - c. El mensaje del informe se encuentra en una cola de mensajes no entregados.
 - d. Cuando el gestor de colas intentaba generar el mensaje de informe, no pudo colocarlo en la cola adecuada, ni tampoco en la cola de mensajes no entregados, por lo que no se pudo generar el mensaje de informe.
 - e. Se produjo un fallo en el gestor de colas entre la acción notificada (llegada, entrega o vencimiento) y la generación del mensaje de informe correspondiente. (Esto no ocurre con los mensajes de informe COD si la aplicación recupera el mensaje original dentro de una unidad de trabajo, ya que el mensaje de informe COD se genera dentro de la misma unidad de trabajo)

Los mensajes de informe de excepciones pueden retenerse de la misma manera por las razones 1, 2 y 3 mencionadas anteriormente. Sin embargo, cuando un MCA no puede generar un mensaje de informe de excepción (el mensaje de informe no se puede colocar ni en la cola de respuesta ni en la cola de mensajes no entregados), el mensaje original permanece en la cola de transmisión del remitente y el canal se cierra. Esto ocurre independientemente de si el mensaje de informe debía generarse en el extremo emisor o receptor del canal.

6. Si el mensaje original se bloquea temporalmente (lo que da lugar a la generación de un mensaje de informe de excepción y a que el mensaje original se coloque en una cola de mensajes no entregados), pero el bloqueo se despeja y una aplicación lee el mensaje original de la cola de mensajes no entregados y lo vuelve a colocar en su destino, puede ocurrir lo siguiente:
- Aunque se haya generado un mensaje de informe de excepción, el mensaje original finalmente llega correctamente a su destino.
 - Se genera más de un mensaje de informe de excepción con respecto a un único mensaje original, ya que el mensaje original puede encontrar otro bloqueo más adelante.

Informar de los mensajes al publicar en un tema :

1. Se pueden generar informes al publicar un mensaje en un tema. Este mensaje se enviará a todos los suscriptores del tema, que pueden ser cero, uno o muchos. Esto debe tenerse en cuenta al elegir las opciones de informe, ya que podrían generarse muchos mensajes de informe como resultado.
2. Al enviar un mensaje a un tema, puede haber muchas colas de destino a las que se les debe enviar una copia del mensaje. Si alguna de estas colas de destino tiene un problema, como estar llena, la finalización correcta de MQPUT dependerá de la configuración de NPMGDLV o PMGDLV (dependiendo de la persistencia del mensaje). Si la configuración es tal que la entrega del mensaje a la cola de destino debe realizarse correctamente (por ejemplo, se trata de un mensaje persistente a un suscriptor duradero y PMGDLV está establecido en ALL o ALLDUR), entonces se considera que se ha realizado correctamente si se cumple uno de los siguientes criterios:
 - Exitosa inserción en la cola de suscriptores
 - Uso de RODLQ y envío correcto a la cola de mensajes no entregados si la cola del suscriptor no puede aceptar el mensaje

- Uso de RODISC si la cola de suscriptores no puede aceptar el mensaje.

Informar mensajes para segmentos de mensajes :

1. Se pueden solicitar mensajes de informe para los mensajes que permiten la segmentación (véase la descripción del indicador MFSEGA). Si el gestor de colas considera necesario segmentar el mensaje, se puede generar un mensaje de informe para cada uno de los segmentos que posteriormente cumpla la condición pertinente. Por lo tanto, las aplicaciones deben estar preparadas para recibir múltiples mensajes de informe para cada tipo de mensaje de informe solicitado. El MDGID campo del mensaje de informe puede utilizarse para correlacionar los múltiples informes con el identificador de grupo del mensaje original, y el MDFB campo para identificar el tipo de cada mensaje de informe.
2. Si se utiliza GMLOGO para recuperar mensajes de informe para segmentos, tenga en cuenta que las llamadas sucesivas a MQGET pueden devolver informes de diferentes tipos. Por ejemplo, si se solicitan informes COA y COD para un mensaje segmentado por el gestor de colas, las llamadas MQGET para los mensajes de informe pueden devolver los mensajes de informe COA y COD intercalados de forma impredecible. Esto se puede evitar utilizando la opción GMCMPM (opcionalmente con GMATM). GMCMPM hace que el gestor de colas vuelva a ensamblar los mensajes de informe que tienen el mismo tipo de informe. Por ejemplo, la primera llamada MQGET podría reensamblar todos los mensajes COA relacionados con el mensaje original, y la segunda llamada MQGET podría reensamblar todos los mensajes COD. El que se vuelve a ensamblar primero depende del tipo de mensaje de informe que se produzca primero en la cola.
3. Las aplicaciones que colocan segmentos pueden especificar diferentes opciones de informe para cada segmento. Sin embargo, hay que tener en cuenta los siguientes puntos:
 - Si los segmentos se recuperan utilizando la opción GMCMPM, el gestor de colas solo acepta las opciones de informe del primer segmento.
 - Si los segmentos se recuperan uno por uno y la mayoría de ellos tienen una de las opciones ROCOD*, pero al menos un segmento no la tiene, no será posible utilizar la opción GMCMPM para recuperar los mensajes de informe con una sola llamada MQGET, ni utilizar la opción GMASGA para detectar cuándo han llegado todos los mensajes de informe.
4. En una red de tipo « MQ », es posible que los gestores de colas tengan capacidades diferentes. Si un mensaje de informe para un segmento es generado por un gestor de colas o MCA que no admite segmentación, el gestor de colas o MCA no incluirá de forma predeterminada la información de segmento necesaria en el mensaje de informe, lo que puede dificultar la identificación del mensaje original que provocó la generación del informe. Esta dificultad puede evitarse solicitando los datos con el mensaje de informe, es decir, especificando las opciones RO*D o RO*F adecuadas. Sin embargo, tenga en cuenta que si se especifica RO*D, es posible que se devuelvan menos de 100 bytes de datos del mensaje de la aplicación a la aplicación que recupera el mensaje de informe, si este ha sido generado por un gestor de colas o MCA que no admite segmentación.

Contenido del descriptor de mensajes para un mensaje de informe : Cuando el gestor de colas o el agente de canal de mensajes (MCA) genera un mensaje de informe, establece los campos del descriptor de mensajes con los siguientes valores y, a continuación, envía el mensaje de la forma habitual.

Tabla 747. Valores utilizados para los campos MQMD cuando un mensaje de informe es generado por el sistema

Campo de MQMD	Valor utilizado
MDSID	MDSIDV
MDVER	MDVER2
MDREP	RONONE
MDMT	MTRPRT
MDEXP	EIULIM
MDFB	Según corresponda a la naturaleza del informe (FBCOA, FBCOD, FBEXP o un valor RC*)

Tabla 747. Valores utilizados para los campos MQMD cuando un mensaje de informe es generado por el sistema (continuación)

Campo de MQMD	Valor utilizado
MDENC	Copiado del descriptor del mensaje original
MDCSI	Copiado del descriptor del mensaje original
MDFMT	Copiado del descriptor del mensaje original
MDPRI	Copiado del descriptor del mensaje original
MDPER	Copiado del descriptor del mensaje original
MDMID	Según lo especificado por las opciones de informe en el descriptor del mensaje original
MDCID	Según lo especificado por las opciones de informe en el descriptor del mensaje original
MDBOC	0
MDRQ	Espacios en blanco
MDRM	Nombre de gestor de colas
MDUID	Según lo establecido por la opción PMPASI
MDACC	Según lo establecido por la opción PMPASI
MDAID	Según lo establecido por la opción PMPASI
MDPAT	ATQM, o según corresponda para el agente del canal de mensajes
MDPAN	Los primeros 28 bytes del nombre del gestor de colas o del agente del canal de mensajes. Para los mensajes de informe generados por el puente IMS, este campo contiene el nombre del grupo XCF y el nombre del miembro XCF del IMS sistema al que se refiere el mensaje.
MDPD	Fecha en la que se envía el mensaje del informe
MDPT	Hora en que se envía el mensaje del informe
MDAOD	Espacios en blanco
MDGID	Copiado del descriptor del mensaje original
MDSEQ	Copiado del descriptor del mensaje original
MDOFF	Copiado del descriptor del mensaje original
MDMFL	Copiado del descriptor del mensaje original
MDOLN	Copiado del descriptor del mensaje original si no es OLUNDF, y establecido en la longitud de los datos del mensaje original en caso contrario

Se recomienda que una aplicación que genere un informe establezca valores similares, excepto en los siguientes casos:

- El MDRM campo se puede dejar en blanco (el gestor de colas lo cambiará por el nombre del gestor de colas local cuando se envíe el mensaje).
- Los campos de contexto deben configurarse utilizando la opción que se habría utilizado para una respuesta, normalmente PMPASI.

Análisis del campo del informe : El MDREP campo contiene subcampos; por este motivo, las aplicaciones que necesiten comprobar si el remitente del mensaje solicitó un informe concreto deben utilizar una de las técnicas descritas en [“Análisis del campo de informe en IBM i”](#) en la página 1455.

Este es un campo de salida para la llamada MQGET y un campo de entrada para las llamadas MQPUT y MQPUT1. El valor inicial de este campo es RONONE.

MDRM (cadena de caracteres de 48 bytes)

Nombre del gestor de colas de respuesta.

Este es el nombre del gestor de colas al que se debe enviar el mensaje de respuesta o el mensaje de informe. MDRQ es el nombre local de una cola definida en este gestor de colas.

Si el MDRM campo está en blanco, el gestor de colas local busca el **MDRQ** nombre en sus definiciones de colas. Si existe una definición local de una cola remota con este nombre, el **MDRM** valor del mensaje transmitido se sustituye por el valor del **RemoteQMgrName** atributo de la definición de la cola remota, y este valor se devolverá en el descriptor del mensaje cuando la aplicación receptora emita una llamada MQGET para el mensaje. Si no existe una definición local de una cola remota, el MDRM que se transmite con el mensaje es el nombre del gestor de colas local.

Si se especifica el nombre, puede contener espacios en blanco al final; el primer carácter nulo y los caracteres que le siguen se tratan como espacios en blanco. Sin embargo, no se comprueba que el nombre cumpla las reglas de nomenclatura para los gestores de colas, ni que este nombre sea conocido por el gestor de colas emisor; esto también es válido para el nombre transmitido, si se **MDRM** sustituye en el mensaje transmitido.

Si no se requiere una cola de respuesta, se recomienda (aunque esto no se comprueba) que el MDRM campo se deje en blanco; el campo no debe dejarse sin inicializar.

Para la llamada MQGET, el gestor de colas siempre devuelve el nombre relleno con espacios en blanco hasta alcanzar la longitud del campo.

Este es un campo de salida para la llamada MQGET y un campo de entrada para las llamadas MQPUT y MQPUT1. La longitud de este campo viene dada por LNQM. El valor inicial de este campo es 48 caracteres en blanco.

MDRQ (cadena de caracteres de 48 bytes)

Nombre de la cola de respuesta.

Este es el nombre de la cola de mensajes a la que la aplicación que emitió la solicitud de obtención del mensaje debe enviar los mensajes MTRPLY y MTRPRT. El nombre es el nombre local de una cola definida en el gestor de colas identificado por MDRM. Esta cola no debería ser una cola modelo, aunque el gestor de colas de envío no lo verifica cuando se envía el mensaje.

Para las llamadas MQPUT y MQPUT1, este campo no debe estar en blanco si el MDMT campo tiene el valor MTRQST o si el MDREP campo solicita algún mensaje de informe. Sin embargo, el valor especificado (o sustituido) se transmite a la aplicación que emite la solicitud de obtención del mensaje, independientemente del tipo de mensaje.

Si el MDRM campo está en blanco, el gestor de colas local busca el nombre MDRQ en sus propias definiciones de colas. Si existe una definición local de una cola remota con este nombre, el MDRQ valor del mensaje transmitido se sustituye por el valor del **RemoteQName** atributo de la definición de la cola remota, y este valor se devolverá en el descriptor del mensaje cuando la aplicación receptora emita una llamada MQGET para el mensaje. Si no existe una definición local de una cola remota, no MDRQ se modifica.

Si se especifica el nombre, puede contener espacios en blanco al final; el primer carácter nulo y los caracteres que le siguen se tratan como espacios en blanco. Sin embargo, no se comprueba que el nombre cumpla las reglas de nomenclatura para colas; esto también es válido para el nombre transmitido, si se MDRQ sustituye en el mensaje transmitido. La única comprobación que se realiza es que se haya especificado un nombre, si las circunstancias lo requieren.

Si no se requiere una cola de respuesta, se recomienda (aunque esto no se comprueba) que el MDRQ campo se deje en blanco; el campo no debe dejarse sin inicializar.

Para la llamada MQGET, el gestor de colas siempre devuelve el nombre relleno con espacios en blanco hasta alcanzar la longitud del campo.

Si un mensaje que requiere un mensaje de informe no se puede entregar y el mensaje de informe tampoco se puede entregar a la cola especificada, tanto el mensaje original como el mensaje de informe se envían a la cola de mensajes no entregados (mensajes no entregados). Véase el **DeadLetterQName** atributo descrito en [“Atributos del gestor de colas en IBM i”](#) en la página 1432.

Este es un campo de salida para la llamada MQGET y un campo de entrada para las llamadas MQPUT y MQPUT1. La longitud de este campo viene dada por LNQN. El valor inicial de este campo es 48 caracteres en blanco.

MDSEQ (entero con signo de 10 dígitos)

Número de secuencia del mensaje lógico en el grupo.

Los números de secuencia comienzan en 1 y aumentan en 1 por cada nuevo mensaje lógico del grupo, hasta un máximo de 999 999 999. Un mensaje físico que no pertenece a un grupo tiene un número de secuencia 1.

Este campo no necesita ser establecido por la aplicación en la llamada MQPUT o MQGET si:

- En la llamada MQPUT, se especifica PMLOGO.
- En la llamada MQGET, no se especifica MOSEQN.

Estas son las formas recomendadas de utilizar estas llamadas para mensajes que no son mensajes de informe. Sin embargo, si la aplicación requiere más control, o si la llamada es de tipo « MQPUT1 », la aplicación debe asegurarse de que MDSEQ se establezca en un valor adecuado.

En las llamadas de entrada a MQPUT y MQPUT1, el gestor de colas utiliza el valor detallado en la [Tabla 1](#). En la salida de las llamadas MQPUT y MQPUT1, el gestor de colas establece este campo con el valor que se envió con el mensaje.

Al introducir la llamada MQGET, el gestor de colas utiliza el valor detallado en la [Tabla 1](#). En la salida de la llamada MQGET, el gestor de colas establece este campo con el valor del mensaje recuperado.

El valor inicial de este campo es uno. Este campo se ignora si MDVER es menor que MDVER2.

MDSID (cadena de caracteres de 4 bytes)

Identificador de estructura.

El valor debe ser:

MDSIDV

Identificador para la estructura del descriptor de mensajes.

Este campo siempre es un campo de entrada. El valor inicial de este campo es MDSIDV.

MDUID (cadena de caracteres de 12 bytes)

Identificador de usuario.

Esto forma parte del *contexto* de identidad del mensaje. Para obtener más información sobre el contexto de los mensajes, consulte [Contexto de los mensajes](#) y [Control de la información de contexto](#).

MDUID Especifica el identificador de usuario de la aplicación que originó el mensaje. El gestor de colas trata esta información como datos de caracteres, pero no define su formato.

Una vez recibido el mensaje, MDUID se puede utilizar en el ODAU campo del **OBJDSC** parámetro de una llamada MQOPEN o MQPUT1 posterior, de modo que la comprobación de autorización se realice para el MDUID usuario en lugar de para la aplicación que realiza la apertura.

Cuando el gestor de colas genera esta información para una llamada MQPUT o MQPUT1, utiliza un identificador de usuario determinado a partir del entorno.

Cuando el identificador de usuario se determina a partir del entorno:

-  En z/OS, el gestor de colas utiliza:
 - Para el lote, el identificador de usuario de la tarjeta JES JOB o la tarea iniciada

- Para TSO, el identificador de usuario que inicia sesión
- Para CICS, el identificador de usuario asociado a la tarea
- Para IMS, el identificador de usuario depende del tipo de aplicación:
 - Durante:

- Regiones BMP sin mensaje
- Regiones IFP sin mensaje
- Regiones BMP y IFP de mensajes que no han emitido una llamada GU correcta

El gestor de colas utiliza el identificador de usuario de la tarjeta JES JOB de la región o el identificador de usuario TSO. Si están en blanco o son nulos, utiliza el nombre del bloque de especificación del programa (PSB).

- Durante:
 - Regiones BMP y IFP de mensajes que han emitido una llamada GU con éxito
 - Regiones MPP

El gestor de colas utiliza uno de los siguientes:

- El identificador de usuario registrado asociado al mensaje
- El nombre del terminal lógico (LTERM)
- El identificador de usuario de la tarjeta JES JOB de la región
- El identificador de usuario TSO
- El nombre PSB

-  En IBM i, el gestor de colas utiliza el nombre del perfil de usuario asociado al trabajo de la aplicación.
-   En AIX and Linux, el gestor de colas utiliza:
 - El nombre de inicio de sesión de la aplicación
 - El identificador de usuario efectivo del proceso si no hay ningún inicio de sesión disponible
 - El identificador de usuario asociado a la transacción, si la aplicación es una CICS transacción
- En VSE/ESA, este es un campo reservado.
-  En Windows, el administrador de colas utiliza los primeros 12 caracteres del nombre de usuario que ha iniciado sesión.

Para las llamadas MQPUT y MQPUT1, este es un campo de entrada/salida si se especifica PMSETI o PMSETA en el **PMO** parámetro. Cualquier información que siga a un carácter nulo dentro del campo se descarta. El carácter nulo y cualquier carácter posterior son convertidos en espacios en blanco por el gestor de colas. Si no se especifica PMSETI o PMSETA, este campo se ignora en la entrada y es un campo de solo salida.

Tras la finalización satisfactoria de una llamada MQPUT o MQPUT1, este campo contiene el MDUID que se transmitió con el mensaje si se colocó en una cola. Este será el valor de MDUID que se conservará con el mensaje si se retiene (véase la descripción de PMRET para obtener más detalles sobre las publicaciones retenidas), pero no se utilizará como MDUID cuando el mensaje se envíe como publicación a los suscriptores, ya que estos proporcionan un valor para anular MDUID en todas las publicaciones que se les envían. Si el mensaje no tiene contexto, el campo queda completamente en blanco.

Este es un campo de salida para la llamada MQGET. La longitud de este campo viene dada por LNUID. El valor inicial de este campo es 12 caracteres en blanco.

MDVER (entero con signo de 10 dígitos)

Número de versión de la estructura.

El valor debe ser uno de los siguientes:

MDVER1

Version-1 Estructura del descriptor de mensajes.

MDVER2

Version-2 Estructura del descriptor de mensajes.

Nota: Cuando se utiliza un MQMD (version-2), el gestor de colas realiza comprobaciones adicionales en cualquier estructura de encabezado MQPUT (MQ) que pueda estar presente al principio de los datos del mensaje de la aplicación; para obtener más detalles, consulte las notas de uso de la llamada MQPUT.

Los campos que solo existen en la versión más reciente de la estructura se identifican como tales en las descripciones de los campos. La siguiente constante especifica el número de versión de la versión actual:

MDVERC

Versión actual de la estructura del descriptor de mensajes.

Este campo siempre es un campo de entrada. El valor inicial de este campo es MDVER1.

Valores iniciales

Tabla 748. Campos en MQMD		
Nombre de campo	Nombre de la constante	Valor de la constante
MDSID	MDSIDV	'MD--'
MDVER	MDVER1	1
MDREP	RONONE	0
MDMT	MTDGRM	8
MDEXP	EIULIM	-1
MDFB	FBNONE	0
MDENC	ENNAT	Depende del entorno
MDCSI	CSQM	0
MDFMT	FMNONE	Espacios en blanco
MDPRI	PRQDEF	-1
MDPER	PEQDEF	2
MDMID	MINONE	Nulos
MDCID	CINONE	Nulos
MDBOC	Ninguna	0
MDRQ	Ninguna	Espacios en blanco
MDRM	Ninguna	Espacios en blanco
MDUID	Ninguna	Espacios en blanco
MDACC	ACNONE	Nulos
MDAID	Ninguna	Espacios en blanco
MDPAT	ATNCON	0
MDPAN	Ninguna	Espacios en blanco

Tabla 748. Campos en MQMD (continuación)

Nombre de campo	Nombre de la constante	Valor de la constante
MDPD	Ninguna	Espacios en blanco
MDPT	Ninguna	Espacios en blanco
MDAOD	Ninguna	Espacios en blanco
MDGID	GINONE	Nulos
MDSEQ	Ninguna	1
MDOFF	Ninguna	0
MDMFL	MFNONE	0
MDOLN	OLUNDF	-1
Notas: 1. El símbolo ~ representa un único carácter en blanco.		

Declaración RPG

```

D*.1.....2.....3.....4.....5.....6.....7..
D*
D* MQMD Structure
D*
D* Structure identifier
D MDSID          1      4    INZ('MD ')
D* Structure version number
D MDVER          5      8I 0 INZ(1)
D* Options for report messages
D MDREP          9     12I 0 INZ(0)
D* Message type
D MDMT          13     16I 0 INZ(8)
D* Message lifetime
D MDEXP         17     20I 0 INZ(-1)
D* Feedback or reason code
D MDFB          21     24I 0 INZ(0)
D* Numeric encoding of message data
D MDENC          25     28I 0 INZ(273)
D* Character set identifier of messagedata
D MDCSI          29     32I 0 INZ(0)
D* Format name of message data
D MDFMT          33     40    INZ(' ')
D* Message priority
D MDPRI          41     44I 0 INZ(-1)
D* Message persistence
D MDPER          45     48I 0 INZ(2)
D* Message identifier
D MDMID          49     72    INZ(X'00000000000000-
D                               000000000000000000-
D                               000000000000')
D* Correlation identifier
D MDCID          73     96    INZ(X'00000000000000-
D                               000000000000000000-
D                               000000000000')
D* Backout counter
D MDBOC          97    100I 0 INZ(0)
D* Name of reply queue
D MDRQ          101    148    INZ
D* Name of reply queue manager
D MDRM          149    196    INZ
D* User identifier
D MDUID          197    208    INZ
D* Accounting token
D MDACC          209    240    INZ(X'00000000000000-
D                               000000000000000000-
D                               000000000000000000-
D                               000000')
D* Application data relating to identity
D MDAID          241    272    INZ

```

```

D* Type of application that put the message
D MDPAT 273 276I 0 INZ(0)
D* Name of application that put the message
D MDPAN 277 304 INZ
D* Date when message was put
D MDPD 305 312 INZ
D* Time when message was put
D MDPT 313 320 INZ
D* Application data relating to origin
D MDAOD 321 324 INZ
D* Group identifier
D MDGID 325 348 INZ(X'00000000000000-
D 00000000000000000000-
D 000000000000')
D* Sequence number of logical message within group
D MDSEQ 349 352I 0 INZ(1)
D* Offset of data in physical message from start of logical message
D MDOFF 353 356I 0 INZ(0)
D* Message flags
D MDMFL 357 360I 0 INZ(0)
D* Length of original message
D MDOLN 361 364I 0 INZ(-1)

```

MQMDE (Extensión del descriptor de mensajes) en IBM i

Visión general

Propósito : La estructura MQMDE describe los datos que a veces aparecen antes de los datos del mensaje de la aplicación. La estructura contiene los campos MQMD que existen en el MQMD version-2, pero no en el MQMD version-1.

Nombre del formato : FMMDE.

Conjunto de caracteres y codificación : Los datos en MQMDE deben estar en el conjunto de caracteres proporcionado por el atributo del **CodedCharSetId** gestor de colas y la codificación del gestor de colas local proporcionada por ENNAT para el lenguaje de programación C.

El juego de caracteres y la codificación del MQMDE deben configurarse en los campos *MDCSI MDENC* y en:

- El MQMD (si la estructura MQMDE se encuentra al inicio de los datos del mensaje), o
- La estructura de encabezado que precede a la estructura MQMDE (todos los demás casos).

Si el MQMDE no está en el juego de caracteres y la codificación del gestor de colas, el MQMDE se acepta pero no se respeta, es decir, el MQMDE se trata como datos de mensaje.

Uso : Las aplicaciones normales deben utilizar un MQMD de tipo « version-2 », en cuyo caso no encontrarán una estructura MQMDE. Sin embargo, las aplicaciones especializadas y aquellas que siguen utilizando un MQMD (version-1) pueden encontrarse con un MQMDE en algunas situaciones. La estructura MQMDE puede darse en las siguientes circunstancias:

- Especificado en las llamadas MQPUT y MQPUT1
- Devuelto por la llamada MQGET
- En mensajes en colas de transmisión
- “MQMDE especificado en llamadas MQPUT y MQPUT1” en la [página 1178](#)
- “MQMDE devuelto por la llamada MQGET” en la [página 1178](#)
- “MQMDE en mensajes en colas de transmisión” en la [página 1179](#)
- “Campos” en la [página 1179](#)
- “Valores iniciales” en la [página 1181](#)
- “Declaración RPG” en la [página 1181](#)

MQMDE especificado en llamadas MQPUT y MQPUT1

En las llamadas MQPUT y MQPUT1, si la aplicación proporciona un MQMD version-1, la aplicación puede, opcionalmente, anteponer a los datos del mensaje un MQMDE, estableciendo el *MDFMT* campo en MQMD en FMMDE para indicar que hay un MQMDE presente. Si la aplicación no proporciona un MQMDE, el gestor de colas asume los valores predeterminados para los campos del MQMDE. Los valores predeterminados que utiliza el gestor de colas son los mismos que los valores iniciales de la estructura; consulte [Tabla 750 en la página 1181](#).

Si la aplicación proporciona un MQMD version-2 y antepone un MQMDE a los datos del mensaje de la aplicación, las estructuras se procesan como se muestra en [Tabla 749 en la página 1178](#).

Tabla 749. Acción del gestor de colas cuando se especifica MQMDE en MQPUT o e MQPUT1			
MQMD Version	Valores de los campos « version-2 » (Estado de la solicitud de pago)	Valores de los campos correspondientes en MQMDE	Medidas adoptadas por el gestor de colas
1	-	Válido	MQMDE tiene el honor de
2	Valor predeterminado	Válido	MQMDE tiene el honor de
2	No predeterminado	Válido	MQMDE se trata como datos de mensaje
1 o 2	Cualquiera	No válido	La llamada falla con un código de motivo adecuado
1 o 2	Cualquiera	MQMDE tiene un juego de caracteres o una codificación incorrectos, o es una versión no compatible	MQMDE se trata como datos de mensaje

Hay un caso especial. Si la aplicación utiliza un MQMD de tipo « version-2 » (es decir, el indicador MFSEG o MFLSEG está activado) para insertar un mensaje que es un segmento, y el nombre del formato en el MQMD es FMDLH, el gestor de colas genera una estructura MQMDE y la inserta *entre* la estructura MQDLH y los datos que le siguen. En el MQMD que el gestor de colas conserva con el mensaje, los campos « version-2 » se establecen en sus valores predeterminados.

Varios de los campos que existen en el MQMD version-2 pero no en el MQMD version-1 son campos de entrada/salida en MQPUT y MQPUT1. Sin embargo, el gestor de colas no devuelve ningún valor en los campos equivalentes del MQMDE en la salida de las llamadas MQPUT y MQPUT1; si la aplicación requiere esos valores de salida, debe utilizar un MQMD version-2.

MQMDE devuelto por la llamada MQGET

En la llamada MQGET, si la aplicación proporciona un MQMD version-1, el gestor de colas antepone al mensaje devuelto un MQMDE, pero solo si uno o varios de los campos del MQMDE tienen un valor distinto del predeterminado. El gestor de colas establece el *MDFMT* campo en MQMD con el valor FMMDE para indicar que hay un MQMDE presente.

Si la aplicación proporciona un MQMDE al inicio del **BUFFER** parámetro, el MQMDE se ignora. Al regresar de la llamada MQGET, se sustituye por MQMDE para el mensaje (si es necesario) o se sobrescribe con los datos del mensaje de la aplicación (si no es necesario MQMDE).

Si la llamada MQGET devuelve un MQMDE, los datos del MQMDE suelen estar en el juego de caracteres y la codificación del gestor de colas. Sin embargo, el MQMDE puede estar en otro juego de caracteres y codificación si:

- El MQMDE se trató como datos sobre la llamada MQPUT o MQPUT1 (consulte [Tabla 749 en la página 1178](#) para conocer las circunstancias que pueden provocar esto).
- El mensaje se recibió desde un gestor de colas remoto conectado mediante una conexión de tipo « TCP », y el agente del canal de mensajes receptor (MCA) no estaba configurado correctamente (consulte [Seguridad de IBM MQ for IBM i objetos](#) para obtener más información).

MQMDE en mensajes en colas de transmisión

Los mensajes en las colas de transmisión tienen como prefijo la estructura MQXQH, que contiene en su interior un MQMD (version-1). También puede haber un MQMDE, situado entre la estructura MQXQH y los datos del mensaje de la aplicación, pero normalmente solo estará presente si uno o varios de los campos del MQMDE tienen un valor distinto al predeterminado.

También pueden aparecer otras estructuras IBM MQ de encabezado entre la estructura MQXQH y los datos del mensaje de la aplicación. Por ejemplo, cuando está presente el encabezado de carta muerta MQDLH y el mensaje no es un segmento, el orden es:

- MQXQH (que contiene un MQMD e version-1)
- MQMDE
- MQDLH
- Datos del mensaje de la aplicación

Campos

La estructura MQMDE contiene los siguientes campos, que se describen en **orden alfabético** :

MECSI (entero con signo de 10 dígitos)

Identificador del conjunto de caracteres de los datos que siguen a MQMDE.

Especifica el identificador del conjunto de caracteres de los datos que siguen a la estructura MQMDE; no se aplica a los datos de caracteres de la propia estructura MQMDE.

En la llamada MQPUT o MQPUT1, la aplicación debe establecer este campo con el valor adecuado para los datos. El gestor de colas no comprueba que este campo sea válido. Se puede utilizar el siguiente valor especial:

CSINHT

Heredar el identificador del conjunto de caracteres de esta estructura.

Los datos de caracteres en los datos *que siguen* esta estructura están en el mismo conjunto de caracteres que esta estructura.

El gestor de colas cambia este valor en la estructura enviada en el mensaje al identificador real del juego de caracteres de la estructura. Si no se produce ningún error, la llamada MQGET no devuelve el valor CSINHT.

CSINHT no se puede utilizar si el valor del *MDPAT* campo en MQMD es ATBRKR.

El valor inicial de este campo es CSUNDF.

MEENC (entero con signo de 10 dígitos)

MEENC (entero con signo de 10 dígitos)

Esto especifica la codificación numérica de los datos que siguen a la estructura MQMDE; no se aplica a los datos numéricos de la propia estructura MQMDE.

En la llamada MQPUT o MQPUT1, la aplicación debe establecer este campo con el valor adecuado para los datos. El gestor de colas no comprueba que el campo sea válido. Consulte el *MDENC*

campo descrito en “MQMD (descriptor de mensaje) en IBM i” en la [página 1131](#) para obtener más información sobre las codificaciones de datos.

El valor inicial de este campo es ENNAT.

MEFLG (entero con signo de 10 dígitos)

Banderas generales.

Se puede especificar la siguiente bandera:

MEFNON

Sin banderas.

El valor inicial de este campo es MEFNON.

MEFMT (cadena de caracteres de 8 bytes)

Formatear el nombre de los datos que siguen a MQMDE.

Especifica el nombre del formato de los datos que siguen a la estructura MQMDE.

En la llamada MQPUT o MQPUT1, la aplicación debe establecer este campo con el valor adecuado para los datos. El gestor de colas no comprueba que este campo sea válido. Consulte el *MDFMT* campo descrito en “MQMD (descriptor de mensaje) en IBM i” en la [página 1131](#) para obtener más información sobre los nombres de formato.

El valor inicial de este campo es FMNONE.

MEGID (cadena de bits de 24 bytes)

Identificador de grupo.

Véase el *MDGID* campo descrito en “MQMD (descriptor de mensaje) en IBM i” en la [página 1131](#). El valor inicial de este campo es GINONE.

MELEN (entero con signo de 10 dígitos)

Longitud de la estructura MQMDE.

Se define el siguiente valor:

MELEN2

Longitud de la estructura de extensión del descriptor de mensajes « version-2 ».

El valor inicial de este campo es MELEN2.

MEMFL (entero con signo de 10 dígitos)

Indicadores de mensajes.

Véase el *MDMFL* campo descrito en “MQMD (descriptor de mensaje) en IBM i” en la [página 1131](#). El valor inicial de este campo es MFNONE.

MEOFF (entero con signo de 10 dígitos)

Desplazamiento de datos en el mensaje físico desde el inicio del mensaje lógico.

Véase el *MDOFF* campo descrito en “MQMD (descriptor de mensaje) en IBM i” en la [página 1131](#). El valor inicial de este campo es 0.

MEOLN (entero con signo de 10 dígitos)

Longitud del mensaje original.

Véase el *MDOLN* campo descrito en “MQMD (descriptor de mensaje) en IBM i” en la [página 1131](#). El valor inicial de este campo es OLUNDF.

MESEQ (entero con signo de 10 dígitos)

Número de secuencia del mensaje lógico en el grupo.

Véase el *MDSEQ* campo descrito en “MQMD (descriptor de mensaje) en IBM i” en la página 1131. El valor inicial de este campo es 1.

MESID (cadena de caracteres de 4 bytes)

Identificador de estructura.

El valor debe ser:

MESIDV

Identificador de la estructura de extensión del descriptor de mensajes.

El valor inicial de este campo es MESIDV.

MEVER (entero con signo de 10 dígitos)

Número de versión de la estructura.

El valor debe ser:

MEVER2

Version-2 Estructura de extensión del descriptor de mensajes.

La siguiente constante especifica el número de versión de la versión actual:

MEVERC

Versión actual de la estructura de extensión del descriptor de mensajes.

El valor inicial de este campo es MEVER2.

Valores iniciales

Tabla 750. Campos en MQMDE		
Nombre de campo	Nombre de la constante	Valor de la constante
<i>MESID</i>	MESIDV	'MDE¬'
<i>MEVER</i>	MEVER2	2
<i>MELEN</i>	MELEN2	72
<i>MEENC</i>	ENNAT	Depende del entorno
<i>MECSI</i>	CSUNDF	0
<i>MEFMT</i>	FMNONE	Espacios en blanco
<i>MEFLG</i>	MEFNON	0
<i>MEGID</i>	GINONE	Nulos
<i>MESEQ</i>	Ninguna	1
<i>MEOFF</i>	Ninguna	0
<i>MEMFL</i>	MFNONE	0
<i>MEOLN</i>	OLUNDF	-1
Notas:		
1. El símbolo ¬ representa un único carácter en blanco.		

Declaración RPG

```

D*..1.....2.....3.....4.....5.....6.....7..
D*
D* MQMDE Structure
D*
```

```

D* Structure identifier
D MESID 1 4 INZ('MDE ')
D* Structure version number
D MEVER 5 8I 0 INZ(2)
D* Length of MQMDE structure
D MELEN 9 12I 0 INZ(72)
D* Numeric encoding of data that followsMQMDE
D MEENC 13 16I 0 INZ(273)
D* Character-set identifier of data thatfollows MQMDE
D MECSI 17 20I 0 INZ(0)
D* Format name of data that followsMQMDE
D MEFMT 21 28 INZ(' ')
D* General flags
D MEFLG 29 32I 0 INZ(0)
D* Group identifier
D MEGID 33 56 INZ(X'0000000000000000-
D 00000000000000000000-
D 000000000000')
D* Sequence number of logical messagewithin group
D MESEQ 57 60I 0 INZ(1)
D* Offset of data in physical messagefrom start of logical message
D MEOFF 61 64I 0 INZ(0)
D* Message flags
D MEMFL 65 68I 0 INZ(0)
D* Length of original message
D MEOLN 69 72I 0 INZ(-1)

```

IBM i MQMHBO (Opciones de control de mensajes al búfer) activado IBM i

Estructura que define las opciones del controlador de mensajes al búfer

Visión general

Propósito : La estructura MQMHBO permite a las aplicaciones especificar opciones que controlan cómo se producen los búferes a partir de los identificadores de mensajes. La estructura es un parámetro de entrada en la llamada MQMHBUFF.

Conjunto de caracteres y codificación : Los datos en MQMHBO deben estar en el conjunto de caracteres de la aplicación y la codificación de la aplicación (ENNAT).

- “Campos” en la [página 1182](#)
- “Valores iniciales” en la [página 1183](#)
- “Declaración RPG” en la [página 1183](#)

Campos

La estructura MQMHBO contiene los siguientes campos, que se describen en **orden alfabético** :

MBOPT (entero con signo de 10 dígitos)

Estructura de opciones del controlador de mensajes al búfer: campo MBOPT.

Estas opciones controlan la acción de MQMHBUFF.

Debe especificar la siguiente opción:

MBPRRF

Al convertir propiedades de un identificador de mensaje a un búfer, conviértalas al formato MQRFH2.

Opcionalmente, también puede especificar la siguiente opción. Para especificar más de una opción, suma los valores (sin añadir la misma constante más de una vez) o combínalos mediante la operación OR bit a bit (si el lenguaje de programación admite operaciones bit a bit).

MBDLPR

Las propiedades que se añaden al búfer se eliminan del identificador del mensaje. Si la llamada falla, no se eliminan ninguna propiedad.

Este campo siempre es un campo de entrada. El valor inicial de este campo es MBPRRF.

MBSID (entero con signo de 10 dígitos)

Identificador de mensaje para la estructura de opciones del búfer: campo MBSID.

Este es el identificador de la estructura. El valor debe ser:

MBSIDV

Identificador para el controlador de mensajes a la estructura de opciones del búfer.

Este campo siempre es un campo de entrada. El valor inicial de este campo es « isMBSIDV ».

MBVER (entero con signo de 10 dígitos)

Este es el número de versión de la estructura. El valor debe ser:

MBVER1

Número de versión para el identificador de mensaje a la estructura de opciones del búfer.

La siguiente constante especifica el número de versión de la versión actual:

MBVERC

Versión actual del identificador de mensaje para la estructura de opciones del búfer.

Este campo siempre es un campo de entrada. El valor inicial de este campo es MBVER1.

Valores iniciales

Tabla 751. Campos en MQMHBO		
Nombre de campo	Nombre de la constante	Valor de la constante
MVSID	MBSIDV	'MHBO'
MBVER	MBVER1	1
MBOPT	MBPRRF	

Notas:

1. El valor «Cadena nula» o «Espacios en blanco» denota un carácter en blanco.

Declaración RPG

```
D* MQMHBO Structure
D*
D*
D* Structure identifier
D MBSID 1 4 INZ('MHBO')
D*
D* Structure version number
D MBVER 5 8I 0 INZ(1)
D*
D* Options that control the action of MQMHBUF
D MBOPT 9 12I 0 INZ(1)
```

IBM i MQOD (Descriptor de objeto) activado IBM i

La estructura MQOD se utiliza para especificar un objeto por su nombre.

Visión general

Propósito : Los siguientes tipos de objetos son válidos:

- Cola o lista de distribución
- Lista de nombres
- Definición de proceso
- Gestor de colas

- Tema

La estructura es un parámetro de entrada/salida en las llamadas MQOPEN y MQPUT1.

Versión : La versión actual de MQOD es ODVER4. Los campos que solo existen en las versiones más recientes de la estructura se identifican como tales en las descripciones que siguen.

El archivo COPY proporcionado contiene la versión más reciente de MQOD compatible con el entorno, pero con el valor inicial del *ODVER* campo establecido en ODVER1. Para utilizar campos que no están presentes en la estructura version-1, la aplicación debe establecer el *ODVER* campo en el número de versión requerido.

Para abrir una lista de distribución, *ODVER* debe ser ODVER2 o superior.

Conjunto de caracteres y codificación : Los datos en MQOD deben estar en el conjunto de caracteres proporcionado por el atributo del **CodedCharSetId** gestor de colas y la codificación del gestor de colas local proporcionada por ENNAT. Sin embargo, si la aplicación se ejecuta como un IBM MQ cliente, la estructura debe estar en el juego de caracteres y la codificación del cliente.

- [“Campos” en la página 1184](#)
- [“Valores iniciales” en la página 1191](#)
- [“Declaración RPG” en la página 1192](#)

Campos

La estructura MQOD contiene los siguientes campos, que se describen en **orden alfabético** :

ODASI (cadena de bits de 40 bytes)

Identificador de seguridad alternativo.

Se trata de un identificador de seguridad que se transmite junto con el *ODAU* al servicio de autorización para permitir que se realicen las comprobaciones de autorización adecuadas. *ODASI* se utiliza solo si:

- OOALTU se especifica en la llamada MQOPEN, o
- PMALTU se especifica en la llamada MQPUT1,

y el *ODAU* campo no está completamente en blanco hasta el primer carácter nulo o el final del campo.

El *ODASI* campo tiene la siguiente estructura:

- El primer byte es un entero binario que contiene la longitud de los datos significativos que le siguen; el valor excluye el propio byte de longitud. Si no hay ningún identificador de seguridad, la longitud es cero.
- El segundo byte indica el tipo de identificador de seguridad presente; son posibles los siguientes valores:

SITWNT

Windows identificador de seguridad.

SITNON

Sin identificador de seguridad.

- El tercer byte y los siguientes hasta la longitud definida por el primer byte contienen el identificador de seguridad propiamente dicho.
- Los bytes restantes en el campo se establecen en cero binario.

Se puede utilizar el siguiente valor especial:

SINONE

No se ha especificado ningún identificador de seguridad.

El valor es un cero binario para la longitud del campo.

Este es un campo de entrada. La longitud de este campo viene dada por LNSCID. El valor inicial de este campo es SINONE. Este campo se ignora si *ODVER* es menor que *ODVER3*.

ODAU (cadena de caracteres de 12 bytes)

Identificador de usuario alternativo.

Si se especifica *OOALTU* para la llamada *MQOPEN*, o *PMALTU* para la llamada *MQPUT1*, este campo contiene un identificador de usuario alternativo que se utilizará para comprobar la autorización para la apertura, en lugar del identificador de usuario con el que se está ejecutando actualmente la aplicación. Sin embargo, algunas comprobaciones siguen realizándose con el identificador de usuario actual (por ejemplo, las comprobaciones de contexto).

Si no se especifican *OOALTU* y *PMALTU* y este campo está completamente en blanco hasta el primer carácter nulo o el final del campo, la apertura solo podrá realizarse con éxito si no se necesita autorización de usuario para abrir este objeto con las opciones especificadas.

Si no se especifica ni *OOALTU* ni *PMALTU*, este campo se ignora.

Este es un campo de entrada. La longitud de este campo viene dada por *LNUID*. El valor inicial de este campo es 12 caracteres en blanco.

ODDN (cadena de caracteres de 48 bytes)

Nombre de cola dinámica.

Este es el nombre de una cola dinámica que se creará mediante la llamada *MQOPEN*. Esto solo es relevante cuando *ODON* especifica el nombre de una cola modelo; en todos los demás casos, *ODDN* se ignora.

Los caracteres válidos en el nombre son los mismos que para *ODON*, excepto que también es válido el asterisco. Un nombre que esté en blanco (o en el que solo aparezcan espacios en blanco antes del primer carácter nulo) no es válido si *ODON* es el nombre de una cola de modelos.

Si el último carácter no en blanco del nombre es un asterisco (*), el gestor de colas sustituye el asterisco por una cadena de caracteres que garantiza que el nombre generado para la cola sea único en el gestor de colas local. Para permitir un número suficiente de caracteres para ello, el asterisco solo es válido en las posiciones 1 a 33. No debe haber ningún carácter que no sea un espacio en blanco o un carácter nulo después del asterisco.

Es válido que el asterisco aparezca en la primera posición del carácter, en cuyo caso el nombre se compone únicamente de los caracteres generados por el gestor de colas.

Este es un campo de entrada. La longitud de este campo viene dada por *LNQN*. El valor inicial de este campo es 'AMQ.*', rellenado con espacios en blanco.

ODIDC (entero con signo de 10 dígitos)

Número de colas que no se pudieron abrir.

Este es el número de colas de la lista de distribución que no se pudieron abrir correctamente. Si está presente, este campo también se establece al abrir una cola única que no está en una lista de distribución.

Nota: Si está presente, este campo solo se establece si el **CMPCOD** parámetro de la llamada *MQOPEN* o *MQPUT1* es *CCOK* o *CCWARN*; no se establece si el **CMPCOD** parámetro es *CCFAIL*.

Se trata de un campo de salida. El valor inicial de este campo es 0. Este campo se ignora si *ODVER* es menor que *ODVER2*.

ODKDC (entero con signo de 10 dígitos)

Número de colas locales abiertas correctamente.

Este es el número de colas de la lista de distribución que se resuelven en colas locales y que se han abierto correctamente. El recuento no incluye las colas que se resuelven en colas remotas (aunque inicialmente se utilice una cola de transmisión local para almacenar el mensaje). Si está presente, este campo también se establece al abrir una cola única que no está en una lista de distribución.

Se trata de un campo de salida. El valor inicial de este campo es 0. Este campo se ignora si *ODVER* es menor que *ODVER2*.

ODMN (cadena de caracteres de 48 bytes)

Nombre del gestor de colas de objetos.

Este es el nombre del gestor de colas en el que se define el *ODON* objeto. Los caracteres válidos en el nombre son los mismos que para *ODON* (véase anteriormente). Un nombre que está completamente en blanco hasta el primer carácter nulo o el final del campo indica el gestor de colas al que está conectada la aplicación (el gestor de colas local).

Los siguientes puntos se aplican a los tipos de objetos indicados:

- Si *ODOT* es OTTOP, OTNLST, OTPRO u OTQM, *ODMN* debe estar en blanco o ser el nombre del gestor de colas local.
- Si *ODON* es el nombre de una cola modelo, el gestor de colas crea una cola dinámica con los atributos de la cola modelo y devuelve en el *ODMN* campo el nombre del gestor de colas en el que se crea la cola; este es el nombre del gestor de colas local. Una cola modelo solo se puede especificar en la llamada MQOPEN; una cola modelo no es válida en la llamada MQPUT1.
- Si *ODON* es el nombre de una cola de clúster y *ODMN* está en blanco, el destino real de los mensajes enviados utilizando el identificador de cola devuelto por la llamada MQOPEN es elegido por el gestor de colas (o la salida de carga de trabajo del clúster, si hay una instalada) de la siguiente manera:
 - Si se especifica OOBND0, el gestor de colas selecciona una instancia de la cola del clúster durante el procesamiento de la llamada MQOPEN, y todos los mensajes enviados mediante este identificador de cola se envían a esa instancia.
 - Si se especifica OOBNDN, el gestor de colas puede elegir una instancia diferente de la cola de destino (que reside en un gestor de colas diferente del clúster) para cada llamada MQPUT sucesiva que utilice este identificador de cola.

Si la aplicación necesita enviar un mensaje a una instancia *específica* de una cola de clúster (es decir, una instancia de cola que reside en un gestor de colas concreto del clúster), la aplicación debe especificar el nombre de ese gestor de colas en el *ODMN* campo. Esto obliga al gestor de colas local a enviar el mensaje al gestor de colas de destino especificado.

- Si el objeto que se abre es una lista de distribución (es decir, *ODREC* es mayor que cero), *ODMN* debe estar en blanco o ser una cadena nula. Si no se cumple esta condición, la llamada falla con el código de motivo RC2153.

Este es un campo de entrada/salida para la llamada MQOPEN cuando *ODON* es el nombre de una cola de modelos, y un campo de solo entrada en todos los demás casos. La longitud de este campo viene dada por LNQM. El valor inicial de este campo es 48 caracteres en blanco.

ODON (cadena de caracteres de 48 bytes)

El nombre del objeto.

Este es el nombre local del objeto tal y como se define en el gestor de colas identificado por *ODMN*. El nombre puede contener los siguientes caracteres:

- Letras mayúsculas (A - Z)
- Letras minúsculas (a - z)
- Dígitos numéricos (0 - 9)
- Punto (.), barra inclinada (/), guión bajo (_), porcentaje (%)

El nombre no debe contener espacios en blanco al principio ni en el interior, pero puede contener espacios en blanco al final. Se puede utilizar un carácter nulo para indicar el final de los datos significativos en el nombre; el carácter nulo y cualquier carácter que le siga se tratan como espacios en blanco. Las siguientes restricciones se aplican en los entornos indicados:

- En los sistemas que utilizan EBCDIC Katakana, no se pueden utilizar caracteres en minúscula.

- En IBM i, los nombres que contengan caracteres en minúscula, barra inclinada o porcentaje deben escribirse entre comillas cuando se especifiquen en comandos. Estas comillas no deben especificarse para nombres que aparecen como campos en estructuras o como parámetros en llamadas.

Los siguientes puntos se aplican a los tipos de objetos indicados:

- Si *ODON* es el nombre de una cola modelo, el gestor de colas crea una cola dinámica con los atributos de la cola modelo y devuelve en el *ODON* campo el nombre de la cola creada. Una cola modelo solo se puede especificar en la llamada *MQOPEN*; una cola modelo no es válida en la llamada *MQPUT1*.
- Si el objeto que se abre es una lista de distribución (es decir, *ODREC* está presente y es mayor que cero), *ODON* debe estar en blanco o ser una cadena nula. Si no se cumple esta condición, la llamada falla con el código de motivo RC2152.
- Si *ODOT* es *OTQM*, se aplican reglas especiales; en este caso, el nombre debe estar completamente en blanco hasta el primer carácter nulo o el final del campo.
- Si *ODON* es el nombre de una cola alias con *TARGETTYPE(TOPIC)*, primero se realiza una comprobación de seguridad en la cola alias nombrada, como es habitual en el uso de colas alias. Si esta comprobación de seguridad se realiza correctamente, esta llamada *MQOPEN* continuará y se comportará como una *MQOPEN* de un *OTTOP*, incluyendo la realización de una comprobación de seguridad contra el objeto de tema administrativo.

Este es un campo de entrada/salida para la llamada *MQOPEN* cuando *ODON* es el nombre de una cola de modelos, y un campo de solo entrada en todos los demás casos. La longitud de este campo viene dada por *LNQN*. El valor inicial de este campo es 48 caracteres en blanco.

El nombre completo del tema se puede construir a partir de dos campos diferentes: *ODON* y *ODOS*. Para obtener más información sobre cómo se utilizan estos dos campos, consulte [Combinación de cadenas de temas](#).

ODORO (entero con signo de 10 dígitos)

Desplazamiento del primer registro de objeto desde el inicio de *MQOD*.

Es el desplazamiento en bytes del primer registro de objeto *MQOR* desde el inicio de la estructura *MQOD*. El desplazamiento puede ser positivo o negativo. *ODORO* Se utiliza solo cuando se abre una lista de distribución. El campo se ignora si *ODREC* es cero.

Cuando se abre una lista de distribución, se debe proporcionar una matriz de uno o más registros de objetos *MQOR* para especificar los nombres de las colas de destino en la lista de distribución. Esto se puede hacer de dos maneras:

- Al utilizar el campo de compensación *ODORO*

En este caso, la aplicación debe declarar su propia estructura que contenga un *MQOD* seguido de la matriz de registros *MQOR* (con tantos elementos de matriz como sean necesarios) y establecer *ODORO* en la compensación del primer elemento de la matriz desde el inicio del *MQOD*. Se debe tener cuidado para garantizar que esta compensación sea correcta.

- Al utilizar el campo puntero *ODORP*

En este caso, la aplicación puede declarar la matriz de estructuras *MQOR* por separado de la estructura *MQOD* y establecer *ODORP* en la dirección de la matriz.

Independientemente de la técnica elegida, se debe utilizar uno de *ODORP* los dos valores *ODORO* y; la llamada falla con el código de motivo RC2155 si ambos son cero o si ambos son distintos de cero.

Este es un campo de entrada. El valor inicial de este campo es 0. Este campo se ignora si *ODVER* es menor que *ODVER2*.

ODORP (puntero)

Dirección del primer registro de objeto.

Esta es la dirección del primer registro de objetos MQOR. *ODORP* Se utiliza solo cuando se abre una lista de distribución. El campo se ignora si *ODREC* es cero.

Este es un campo de entrada. El valor inicial de este campo es el puntero nulo. Se puede utilizar *ODORP* o *ODORO* para especificar los registros de objetos, pero no ambos; consulte la descripción del *ODORO* campo anteriormente para obtener más detalles. Si no *ODORP* se utiliza, debe establecerse en el puntero nulo o en bytes nulos. Este campo se ignora si *ODVER* es menor que *ODVER2*.

ODOS (MQCHARV)

ODOS especifica el nombre largo del objeto que se va a utilizar.

Este campo solo se utiliza para determinados valores de *ODOT*. Consulte la descripción de *ODOT* para obtener más información sobre qué valores indican que se utiliza este campo.

Si *ODOS* se especifica incorrectamente, según la descripción de cómo utilizar la estructura *MQCHARV*, o si supera la longitud máxima, la llamada falla con el código de motivo RC2441.

Este es un campo de entrada. Los valores iniciales de los campos de esta estructura son los mismos que los de la estructura *MQCHARV*.

El nombre completo del tema se puede construir a partir de dos campos diferentes: *ODON* y *ODOS*. Para obtener más información sobre cómo se utilizan estos dos campos, consulte [Combinación de cadenas de temas](#). Este campo se ignora si *ODVER* es menor que *ODVER4*.

ODOT (entero con signo de 10 dígitos)

Tipo de objeto.

Tipo de objeto al que se hace referencia en *ODON*. Los valores posibles son:

OTQ

Cola. El nombre del objeto se encuentra en *ODON*.

OTNLST

Lista de nombres. El nombre del objeto se encuentra en *ODON*.

OTPRO

. El nombre del objeto se encuentra en *ODON*.

OTQM

Gestor de colas. El nombre del objeto se encuentra en *ODON*.

OTTOP

. El nombre completo del tema se puede construir a partir de dos campos diferentes: *ODON* y *ODOS*.

Para obtener más información sobre cómo se utilizan esos dos campos, consulte [Combinación de cadenas de temas](#).

Si no se encuentra el objeto identificado por el *ODON* campo, la llamada fallará con el código de motivo RC2425, incluso si hay una cadena especificada en *ODOS*.

Este campo siempre es un campo de entrada. El valor inicial de este campo es OTQ.

ODREC (entero con signo de 10 dígitos)

Número de registros de objetos presentes.

Este es el número de registros de objetos MQOR que ha proporcionado la aplicación. Si este número es mayor que cero, indica que se está abriendo una lista de distribución, siendo *ODREC* el número de colas de destino en la lista. Es válido que una lista de distribución contenga un solo destino.

El valor de no *ODREC* debe ser inferior a cero, y si es superior a cero, *ODOT* debe ser OTQ; la llamada falla con el código de motivo RC2154 si no se cumplen estas condiciones.

Este es un campo de entrada. El valor inicial de este campo es 0. Este campo se ignora si *ODVER* es menor que *ODVER2*.

ODRMN (cadena de caracteres de 48 bytes)

Nombre del gestor de colas resuelto.

Este es el nombre del gestor de colas de destino después de que el gestor de colas local haya realizado la resolución de nombres. El nombre devuelto es el nombre del gestor de colas propietario de la cola identificada por *ODRQN*. *ODRMN* puede ser el nombre del administrador de colas local.

Si *ODRQN* es una cola compartida que pertenece al grupo de uso compartido de colas al que pertenece el gestor de colas local, *ODRMN* es el nombre del grupo de uso compartido de colas. Si la cola es propiedad de otro grupo de uso compartido de colas, *ODRQN* puede ser el nombre del grupo de uso compartido de colas o el nombre de un gestor de colas que sea miembro del grupo de uso compartido de colas (la naturaleza del valor devuelto viene determinada por las definiciones de colas que existen en el gestor de colas local).

Solo se devuelve un valor distinto de blanco si el objeto es una cola única abierta para exploración, entrada o salida (o cualquier combinación). Si el objeto abierto es cualquiera de los siguientes, *ODRMN* se establece en espacios en blanco:

- No es una cola
- Una cola, pero no abierta para exploración, entrada o salida
- Una cola de clúster con *OOBNDN* especificado (o con *OOBNDQ* en vigor cuando el **DefBind** atributo de cola tiene el valor *BNDNOT*)
- Una lista de distribución

Se trata de un campo de salida. La longitud de este campo viene dada por *LNQN*. El valor inicial de este campo es la cadena nula en C y 48 caracteres en blanco en otros lenguajes de programación. Este campo se ignora si *ODVER* es menor que *ODVER3*.

ODRO (MQCHARV)

ODRO es el nombre largo del objeto después de que el gestor de colas resuelva el nombre proporcionado en *ODON*.

Este campo solo se devuelve para determinados tipos de objetos, temas y alias de cola que hacen referencia a un objeto de tema.

Si se proporciona el nombre largo del objeto en *ODOS* y no se proporciona nada en *ODON*, el valor devuelto en este campo es el mismo que el proporcionado en *ODOS*.

Si se omite este campo (es decir, si *ODRO.VSBufSize* es cero), no se devuelve *ODRO* el, pero se devuelve la longitud en *ODRO.VSLength*. Si la longitud es menor que la completa *ODRO*, se trunca y devuelve tantos caracteres de la derecha como quepan en la longitud proporcionada.

Si *ODRO* se especifica incorrectamente, según la descripción de cómo utilizar la estructura *MQCHARV*, o si supera la longitud máxima, la llamada falla con el código de motivo RC2520. Este campo se ignora si *ODVER* es menor que *ODVER4*.

ODRQN (cadena de caracteres de 48 bytes)

Nombre de la cola resuelta.

Este es el nombre de la cola de destino después de que el administrador de colas local haya realizado la resolución de nombres. El nombre devuelto es el nombre de una cola que existe en el gestor de colas identificado por *ODRMN*.

Solo se devuelve un valor distinto de blanco si el objeto es una cola única abierta para exploración, entrada o salida (o cualquier combinación). Si el objeto abierto es cualquiera de los siguientes, *ODRQN* se establece en espacios en blanco:

- No es una cola
- Una cola, pero no abierta para exploración, entrada o salida
- Una lista de distribución

- Una cola de alias que hace referencia a un objeto de tema (consulte en “[ODRO \(MQCHARV\)](#)” en la [página 1189](#) su lugar)

Se trata de un campo de salida. La longitud de este campo viene dada por LNQN. El valor inicial de este campo es la cadena nula en C y 48 caracteres en blanco en otros lenguajes de programación. Este campo se ignora si *ODVER* es menor que *ODVER3*.

ODRRO (entero con signo de 10 dígitos)

Desplazamiento del primer registro de respuesta desde el inicio de MQOD.

Es el desplazamiento en bytes del primer registro de respuesta MQRR desde el inicio de la estructura MQOD. El desplazamiento puede ser positivo o negativo. *ODRRO* Se utiliza solo cuando se abre una lista de distribución. El campo se ignora si *ODREC* es cero.

Cuando se abre una lista de distribución, se puede proporcionar una matriz de uno o más registros de respuesta MQRR para identificar las colas que no se han podido abrir (*RRCC* campo en MQRR) y el motivo de cada fallo (*RRREA* campo en MQRR). Los datos se devuelven en la matriz de registros de respuesta en el mismo orden en que aparecen los nombres de las colas en la matriz de registros de objetos. El gestor de colas establece los registros de respuesta solo cuando el resultado de la llamada es mixto (es decir, algunas colas se abrieron correctamente, mientras que otras fallaron, o todas fallaron pero por motivos diferentes); el código de motivo RC2136 de la llamada indica este caso. Si el mismo código de motivo se aplica a todas las colas, ese motivo se devuelve en el **REASON** parámetro de la llamada MQOPEN o MQPUT1, y los registros de respuesta no se establecen. Los registros de respuesta son opcionales, pero si se proporcionan, deben ser *ODREC*.

Los registros de respuesta se pueden proporcionar de la misma manera que los registros de objetos, ya sea especificando un desplazamiento en *ODRRO*, o especificando una dirección en *ODRRP*; consulte la descripción anterior *ODORO* de para obtener más detalles sobre cómo hacerlo. Sin embargo, no se puede utilizar *ODRRP* más de uno de *ODRRO* y; la llamada falla con el código de motivo RC2156 si ambos son distintos de cero.

En la llamada MQPUT1, estos registros de respuesta se utilizan para devolver información sobre los errores que se producen al enviar el mensaje a las colas de la lista de distribución, así como los errores que se producen al abrir las colas. El código de finalización y el código de motivo de la operación de inserción de una cola sustituyen a los de la operación de apertura de esa cola solo si el código de finalización de esta última era CCOK o CCWARN.

Este es un campo de entrada. El valor inicial de este campo es 0. Este campo se ignora si *ODVER* es menor que *ODVER2*.

ODRRP (puntero)

Dirección del registro de primera respuesta.

Esta es la dirección del primer registro de respuesta MQRR. *ODRRP* Se utiliza solo cuando se abre una lista de distribución. El campo se ignora si *ODREC* es cero.

Se puede utilizar *ODRRPODRRO* o para especificar los registros de respuesta, pero no ambos; consulte la descripción anterior del *ODRRO* campo para obtener más detalles. Si no *ODRRP* se utiliza, debe establecerse en el puntero nulo o en bytes nulos.

Este es un campo de entrada. El valor inicial de este campo es el puntero nulo. Este campo se ignora si *ODVER* es menor que *ODVER2*.

ODSID (cadena de caracteres de 4 bytes)

Identificador de estructura.

El valor debe ser:

ODSIDV

Identificador para la estructura del descriptor de objeto.

Este campo siempre es un campo de entrada. El valor inicial de este campo es *ODSIDV*.

ODSS (MQCHARV)

ODSS contiene la cadena utilizada para proporcionar los criterios de selección que se utilizan al recuperar mensajes de una cola.

ODSS no se debe proporcionar en los siguientes casos:

- Si no *ODOT* es OTQ
- Si la cola que se está abriendo no se abre utilizando una de las opciones de entrada, OOINP*

Si *ODSS* se proporciona en estos casos, la llamada falla con el código de motivo RC2516.

Si *ODSS* se especifica incorrectamente, según la descripción de cómo utilizar la estructura MQCHARV, o si supera la longitud máxima, la llamada falla con el código de motivo RC2519. Este campo se ignora si *ODVER* es menor que ODVER4.

ODUDC (entero con signo de 10 dígitos)

Número de colas remotas abiertas satisfactoriamente.

Este es el número de colas de la lista de distribución que se resuelven en colas remotas y que se abrieron correctamente. Si está presente, este campo también se establece al abrir una cola única que no está en una lista de distribución.

Se trata de un campo de salida. El valor inicial de este campo es 0. Este campo se ignora si *ODVER* es menor que ODVER2.

ODVER (entero con signo de 10 dígitos)

Número de versión de la estructura.

El valor debe ser uno de los siguientes:

ODVER1

Version-1 estructura descriptora de objetos.

ODVER2

Version-2 estructura descriptora de objetos.

ODVER3

Version-3 estructura descriptora de objetos.

ODVER4

Version-4 estructura descriptora de objetos.

Los campos que solo existen en las versiones más recientes de la estructura se identifican como tales en las descripciones de los campos. La siguiente constante especifica el número de versión de la versión actual:

ODVERC

Versión actual de la estructura del descriptor de objetos.

Este campo siempre es un campo de entrada. El valor inicial de este campo es ODVER1.

Valores iniciales

Tabla 752. Campos en MQOD		
Nombre de campo	Nombre de la constante	Valor de la constante
ODSID	ODSIDV	'OD--'
ODVER	ODVER1	1
ODOT	OTQ	1
ODON	Ninguna	Espacios en blanco
ODMN	Ninguna	Espacios en blanco

Tabla 752. Campos en MQOD (continuación)

Nombre de campo	Nombre de la constante	Valor de la constante
ODDN	Ninguna	'AMQ.*'
ODAU	Ninguna	Espacios en blanco
ODREC	Ninguna	0
ODKDC	Ninguna	0
ODUDC	Ninguna	0
ODIDC	Ninguna	0
ODORO	Ninguna	0
ODRRO	Ninguna	0
ODORP	Ninguna	Puntero nulo o bytes nulos
ODRRP	Ninguna	Puntero nulo o bytes nulos
ODASI	SINONE	Nulos
ODRQN	Ninguna	Espacios en blanco
ODRMN	Ninguna	Espacios en blanco
ODOS	Según se define para MQCHARV	Según se define para MQCHARV
ODRO	Según lo dispuesto en ODOS	Según lo dispuesto en ODOS
ODSS	Ninguna	Espacios en blanco
Notas: 1. El símbolo ~ representa un único carácter en blanco.		

Declaración RPG

```

D*.1....:....2....:....3....:....4....:....5....:....6....:....7..
D*
D* MQOD Structure
D*
D*
D* Structure identifier
D  ODSID          1      4      INZ('OD ')
D*
D* Structure version number
D  ODVER          5      8I 0 INZ(1)
D*
D* Object type
D  ODOT           9     12I 0 INZ(1)
D*
D* Object name
D  ODON          13     60      INZ
D*
D* Object queue manager name
D  ODMN          61     108      INZ
D*
D* Dynamic queue name
D  ODDN         109     156      INZ('AMQ.*')
D*
D* Alternate user identifier
D  ODAU         157     168      INZ
D*
** Number of object records
D* present
D  ODREC         169    172I 0 INZ(0)
D*

```

```

** Number of local queues opened
D* successfully
D  ODKDC          173    176I 0 INZ(0)
D*
** Number of remote queues opened
D* successfully
D  ODUDC          177    180I 0 INZ(0)
D*
** Number of queues that failed to
D* open
D  ODIDC          181    184I 0 INZ(0)
D*
** Offset of first object record
D* from start of MQOD
D  ODORO          185    188I 0 INZ(0)
D*
** Offset of first response record
D* from start of MQOD
D  ODRRO          189    192I 0 INZ(0)
D*
D* Address of first object record
D  ODORP          193    208*   INZ(*NULL)
D*
** Address of first response
D* record
D  ODRRP          209    224*   INZ(*NULL)
D*
D* Alternate security identifier
D  ODASI          225    264   INZ(X'0000000000000000-
D                                000000000000000000000000-
D                                000000000000000000000000-
D                                000000000000')
D*
D* Resolved queue name
D  ODRQN          265    312   INZ
D*
D* Resolved queue manager name
D  ODRMN          313    360   INZ
D*
D* reserved field
D  ODRE1          361    364I 0 INZ(0)
D*
D* reserved field
D  ODRS2          365    368I 0 INZ(0)
D*
D* Object long name
D* Address of variable length string
D  ODOSCHRP       369    384*   INZ(*NULL)
D* Offset of variable length string
D  ODOSCHRO       385    388I 0 INZ(0)
D* Size of buffer
D  ODOSVSBS       389    392I 0 INZ(-1)
D* Length of variable length string
D  ODOSCHRL       393    396I 0 INZ(0)
D* CCSID of variable length string
D  ODOSCHRC       397    400I 0 INZ(-3)
D*
D* Message Selector
D* Address of variable length string
D  ODSSCHRP       401    416*   INZ(*NULL)
D* Offset of variable length string
D  ODSSCHRO       417    420I 0 INZ(0)
D* Size of buffer
D  ODSSVSBS       421    424I 0 INZ(-1)
D* Length of variable length string
D  ODSSCHRL       425    428I 0 INZ(0)
D* CCSID of variable length string
D  ODSSCHRC       429    432I 0 INZ(-3)
D*
D* Resolved long object name
D* Address of variable length string
D  ODRSOCHRP      433    448*   INZ(*NULL)
D* Offset of variable length string
D  ODRSOCHRO      449    452I 0 INZ(0)
D* Size of buffer
D  ODRSOVSBS      453    456I 0 INZ(-1)
D* Length of variable length string
D  ODRSOCHRL      457    460I 0 INZ(0)
D* CCSID of variable length string
D  ODRSOCHRC      461    464I 0 INZ(-3)
D*

```

```
D* Alias queue resolved object type
D  ODRT                      465    468I 0 INZ(0)
```

IBM i MQOR (Registro de objeto) en IBM i

La estructura MQOR se utiliza para especificar el nombre de la cola y el nombre del gestor de colas de una sola cola de destino.

Visión general

Propósito : MQOR es una estructura de entrada para las llamadas MQOPEN y MQPUT1.

Conjunto de caracteres y codificación : Los datos en MQOR deben estar en el conjunto de caracteres dado por el atributo del **CodedCharSetId** gestor de colas y la codificación del gestor de colas local dada por ENNAT. Sin embargo, si la aplicación se ejecuta como un IBM MQ cliente, la estructura debe estar en el juego de caracteres y la codificación del cliente.

Uso : Al proporcionar una matriz de estas estructuras en la llamada MQOPEN, es posible abrir una lista de colas; esta lista se denomina *lista* de distribución. Cada mensaje enviado utilizando el identificador de cola devuelto por esa llamada MQOPEN se coloca en cada una de las colas de la lista, si la cola se ha abierto correctamente.

- “Campos” en la [página 1194](#)
- “Valores iniciales” en la [página 1194](#)
- “Declaración RPG” en la [página 1195](#)

Campos

La estructura MQOR contiene los siguientes campos, que se describen en **orden alfabético** :

ORMN (cadena de caracteres de 48 bytes)

Nombre del gestor de colas de objetos.

Es igual que el *ODMN* campo de la estructura MQOD (véase MQOD para más detalles).

Este campo siempre es un campo de entrada. El valor inicial de este campo es 48 caracteres en blanco.

ORON (cadena de caracteres de 48 bytes)

El nombre del objeto.

Es igual que el *ODON* campo de la estructura MQOD (véase MQOD para más detalles), salvo que:

- Debe ser el nombre de una cola.
- No debe ser el nombre de una cola modelo.

Este campo siempre es un campo de entrada. El valor inicial de este campo es 48 caracteres en blanco.

Valores iniciales

Tabla 753. Campos en MQOR		
Nombre de campo	Nombre de la constante	Valor de la constante
ORON	Ninguna	Espacios en blanco
ORMN	Ninguna	Espacios en blanco

Declaración RPG

```
D*..1.....2.....3.....4.....5.....6.....7..  
D*  
D* MQOR Structure  
D*  
D* Object name  
D  ORON                1      48    INZ  
D* Object queue manager name  
D  ORMN                49     96    INZ
```

MQPD - Descriptor de propiedad

El **MQPD** se utiliza para definir los atributos de una propiedad.

Visión general

Propósito : La estructura es un parámetro de entrada/salida en la llamada MQSETMP y un parámetro de salida en la llamada MQINQMP.

Conjunto de caracteres y codificación : Los datos en MQPD deben estar en el conjunto de caracteres de la aplicación y la codificación de la aplicación (ENNAT).

- [“Campos” en la página 1195](#)
- [“Valores iniciales” en la página 1197](#)
- [“Declaración RPG” en la página 1198](#)

Campos

La estructura MQPD contiene los siguientes campos, que se describen en **orden alfabético** :

PDCT (entero con signo de 10 dígitos)

Esto describe a qué contexto de mensaje pertenece la propiedad.

Cuando un gestor de colas recibe un mensaje que contiene una propiedad IBM MQ definida que el gestor de colas reconoce como incorrecta. El gestor de colas corrige el valor del *PDCT* campo.

Se puede especificar la siguiente opción:

PDUSC

La propiedad está asociada al contexto de usuario.

No se requiere ninguna autorización especial para poder establecer una propiedad asociada al contexto de usuario utilizando la llamada MQSETMP.

Si no se requiere la opción descrita anteriormente, se puede utilizar la siguiente opción:

PDNOC

La propiedad no está asociada a un contexto de mensaje.

Un valor no reconocido se rechaza con un *PDREA* código de RC2482

Este es un campo de entrada/salida para la llamada MQSETMP y un campo de salida de la llamada MQINQMP. El valor inicial de este campo es PDNOC.

PDCPYOPT (entero con signo de 10 dígitos)

Esto describe el tipo de mensajes en los que se debe copiar la propiedad.

Este es un campo de salida exclusiva para propiedades reconocidas IBM MQy definidas; IBM MQ establece el valor adecuado.

Cuando un gestor de colas recibe un mensaje que contiene una propiedad IBM MQ definida que el gestor de colas reconoce como incorrecta. El gestor de colas corrige el valor del *CopyOptions* campo.

Puede especificar una o varias de estas opciones. Para especificar más de una opción, suma los valores (sin añadir la misma constante más de una vez) o combínalos mediante la operación OR bit a bit (si el lenguaje de programación admite operaciones bit a bit).

COPFOR

Esta propiedad se copia en un mensaje que se reenvía.

COPPUB

Esta propiedad se copia en el mensaje recibido por un suscriptor cuando se publica un mensaje.

COPREP

Esta propiedad se copia en un mensaje de respuesta.

COPRP

Esta propiedad se copia en un mensaje de informe.

COPALL

Esta propiedad se copia en todos los tipos de mensajes posteriores.

COPNON

Esta propiedad no se copia en un mensaje.

Opción predeterminada: Se puede especificar la siguiente opción para proporcionar el conjunto predeterminado de opciones de copia:

COPDEF

Esta propiedad se copia en un mensaje reenviado, en un mensaje de informe o en un mensaje recibido por un suscriptor cuando se publica un mensaje.

Esto equivale a especificar la combinación de opciones COPFOR, más COPRP, más COPPUB.

Si no se requiere ninguna de las opciones descritas anteriormente, utilice la siguiente opción:

COPNON

Utilice este valor para indicar que no se han especificado otras opciones de copia; desde el punto de vista de la programación, no existe ninguna relación entre esta propiedad y los mensajes posteriores. Esto siempre se devuelve para las propiedades del descriptor de mensajes.

Este es un campo de entrada/salida para la llamada MQSETMP y un campo de salida de la llamada MQINQMP. El valor inicial de este campo es COPDEF.

PDOPT (entero con signo de 10 dígitos)

El valor debe ser:

PDNONE

No se ha especificado ninguna opción

Este campo siempre es un campo de entrada. El valor inicial de este campo es PDNONE.

PDSID (entero con signo de 10 dígitos)

Este es el identificador de estructura; el valor debe ser:

PSIDV

Identificador de la estructura del descriptor de propiedad.

Este campo siempre es un campo de entrada. El valor inicial de este campo es **PSIDV**.

PDSUP (entero con signo de 10 dígitos)

Este campo describe el nivel de compatibilidad con la propiedad del mensaje que se requiere del gestor de colas para que el mensaje que contiene esta propiedad se coloque en una cola. Esto solo se aplica a las propiedades IBM MQ definidas; la compatibilidad con el resto de propiedades es opcional.

El campo se establece automáticamente en el valor correcto cuando el gestor de colas conoce la propiedad IBM MQ definida. Si no se reconoce la propiedad, se asigna PDSUPO. Cuando un gestor de colas recibe un mensaje que contiene una propiedad IBM MQ definida que el gestor de colas reconoce como incorrecta. El gestor de colas corrige el valor del *PDSUP* campo.

Al establecer una propiedad IBM MQ definida mediante la llamada MQSETMP en un identificador de mensaje en el que se ha establecido la opción CMNOVA, *PDSUP* se convierte en un campo de entrada. Esto permite que una aplicación coloque una propiedad IBM MQ definida, con el valor correcto, cuando la propiedad no es compatible con el gestor de colas conectado, pero el mensaje debe procesarse en otro gestor de colas.

El valor PDSUPO siempre se asigna a propiedades que no son propiedades IBM MQ definidas.

La llamada MQINQMP devuelve uno de los siguientes valores, o se puede especificar uno de ellos, cuando se utiliza la llamada MQSETMP en un identificador de mensaje en el que se ha establecido la opción CMNOVA:

PDSUPO

La propiedad es aceptada por un gestor de colas aunque no sea compatible. La propiedad se puede descartar para que el mensaje fluya a un gestor de colas que no admite propiedades de mensajes. Este valor también se asigna a propiedades que no están IBM MQ definidas.

PDSUPR

Se requiere soporte para la propiedad. El mensaje es rechazado por un gestor de colas que no admite la IBM MQ propiedad definida. La llamada MQPUT (MQPUT1) falla con el código de finalización CCFAIL y el código de motivo RC2490.

PDSUPL

El mensaje es rechazado por un gestor de colas que no admite la propiedad IBM MQ definida si el mensaje está destinado a una cola local. La llamada MQPUT (MQPUT1) falla con el código de finalización CCFAIL y el código de motivo RC2490.

La llamada MQPUT (MQPUT1) se realiza correctamente si el mensaje está destinado a un gestor de colas remoto.

Este es un campo de salida en la llamada MQINQMP y un campo de entrada en la llamada MQSETMP si el identificador de mensaje se creó con la opción CMNOVA establecida. El valor inicial de este campo es PDSUPO.

PDVER (entero con signo de 10 dígitos)

Este es el número de versión de la estructura; el valor debe ser:

PDVER1

Version-1 estructura del descriptor de propiedad.

La siguiente constante especifica el número de versión de la versión actual:

PDVERC

Versión actual de la estructura del descriptor de propiedad.

Este campo siempre es un campo de entrada. El valor inicial de este campo es **PDVER1**.

Valores iniciales

Tabla 754. Campos en MQPD		
Nombre de campo	Nombre de la constante	Valor de la constante
<i>PDSID</i>	PDSIDV	'PD'
<i>PDVER</i>	PDVER1	1
<i>PDOPT</i>	PDNONE	0
<i>PDSUP</i>	PDSUPO	0
<i>PDCT</i>	PDNOC	0
<i>PDOPYOPT</i>	COPDEF	0

Declaración RPG

```
D* MQDMHO Structure
D*
D*
D* Structure identifier
D  DMSID              1      4      INZ('DMHO')
D*
D* Structure version number
D  DMVER              5      8I 0 INZ(1)
D*
D* Options that control the action of MQDLTMH
D  DMOPT              9      12I 0 INZ(0)
```

MQPMO (Opciones de mensajes de envío) activado IBM i

La estructura MQPMO permite a la aplicación especificar opciones que controlan cómo se colocan los mensajes en las colas o se publican en los temas.

Visión general

Finalidad

La estructura es un parámetro de entrada/salida en las llamadas MQPUT y MQPUT1.

Versión

La versión actual de MQPMO es PMVER2. Los campos que solo existen en las versiones más recientes de la estructura se identifican como tales en las descripciones que siguen.

El archivo COPY proporcionado contiene la versión más reciente de MQPMO compatible con el entorno, pero con el valor inicial del *PMVER* campo establecido en PMVER1. Para utilizar campos que no están presentes en la estructura version-1, la aplicación debe establecer el *PMVER* campo en el número de versión requerido.

Conjunto de caracteres y codificación

Los datos en MQPMO deben estar en el juego de caracteres proporcionado por el atributo del **CodedCharSetId** gestor de colas y la codificación del gestor de colas local proporcionada por ENNAT. Sin embargo, si la aplicación se ejecuta como un IBM MQ cliente, la estructura debe estar en el juego de caracteres y la codificación del cliente.

- “Campos” en la [página 1198](#)
- “Valores iniciales” en la [página 1212](#)
- “Declaración RPG” en la [página 1213](#)

Campos

La estructura MQPMO contiene los siguientes campos, descritos en orden alfabético:

PMCT (entero con signo de 10 dígitos)

Identificador de objeto de la cola de entrada.

Si se especifica PMPASI o PMPASA, este campo debe contener el identificador de la cola de entrada desde la que se toma la información de contexto que se asociará con el mensaje que se está insertando.

Si no se especifican PMPASI y PMPASA, este campo se ignora.

Este es un campo de entrada. El valor inicial de este campo es 0.

PMIDC (entero con signo de 10 dígitos)

Número de mensajes que no se pudieron enviar.

Este es el número de mensajes que no se pudieron enviar a las colas de la lista de distribución. El recuento incluye colas que no se pudieron abrir y colas que se abrieron correctamente, pero en las

que falló la operación de inserción. Este campo también se establece cuando se envía un mensaje a una cola única que no está en una lista de distribución.

Nota: Este campo solo se establece si el **CMPCOD** parámetro en la llamada MQPUT o MQPUT1 es CCOK o CCWARN; no se establece si el **CMPCOD** parámetro es CCFAIL.

Se trata de un campo de salida. El valor inicial de este campo es 0. Este campo no se establece si *PMVER* es menor que *PMVER2*.

PMKDC (entero con signo de 10 dígitos)

Número de mensajes enviados correctamente a las colas locales.

Este es el número de mensajes que la llamada actual MQPUT o MQPUT1 ha enviado correctamente a las colas de la lista de distribución que son colas locales. El recuento no incluye los mensajes enviados a colas que se resuelven en colas remotas (aunque inicialmente se utilice una cola de transmisión local para almacenar el mensaje). Este campo también se establece cuando se envía un mensaje a una cola única que no está en una lista de distribución.

Se trata de un campo de salida. El valor inicial de este campo es 0. Este campo no se establece si *PMVER* es menor que *PMVER2*.

PMOPT (entero con signo de 10 dígitos)

Opciones que controlan la acción de MQPUT y MQPUT1.

Se puede especificar cualquiera de los siguientes o ninguno. Si se requiere más de uno, los valores se pueden sumar (no añada la misma constante más de una vez). Las combinaciones que no son válidas se indican; cualquier otra combinación es válida.

Opciones de publicación : Las siguientes opciones controlan la forma en que se publican los mensajes en un tema.

PMSRTO

La información introducida en los campos MDRQ y MDRM del MQMD de esta publicación no se transmite a los suscriptores. Si esta opción se utiliza con una opción de informe que requiere un ReplyToQ,, la llamada falla con RC2027 .

PMRET

La publicación que se envía debe ser conservada por el gestor de colas. Esto permite a un suscriptor solicitar una copia de esta publicación después de su publicación, utilizando la llamada MQSUBRQ. También permite enviar una publicación a las aplicaciones que se suscriben después de la fecha de publicación, a menos que elijan no recibirla utilizando la opción SONEWP. Si se envía una solicitud a una publicación que se ha retenido, esto se indica mediante la propiedad del mensaje « mq.IsRetained » (Solicitante de publicación retenida) de dicha publicación.

Solo se puede conservar una publicación en cada nodo del árbol de temas. Esto significa que si ya existe una publicación retenida para este tema, publicada por cualquier otra aplicación, se sustituye por esta publicación. Por lo tanto, es mejor evitar que haya más de un editor que conserve mensajes sobre el mismo tema.

Cuando un suscriptor solicita publicaciones retenidas, la suscripción utilizada puede contener un comodín en el tema, en cuyo caso varias publicaciones retenidas podrían coincidir (en varios nodos del árbol de temas) y se podrían enviar varias publicaciones a la aplicación solicitante. Consulte la descripción de la [“MQSUBRQ - Solicitud de suscripción”](#) en la [página 797](#) convocatoria para obtener más detalles.

Si se utiliza esta opción y no se puede conservar la publicación, el mensaje no se publica y la llamada falla con RC2479 .

Opciones de Syncpoint : Las siguientes opciones se refieren a la participación de la llamada MQPUT o MQPUT1 dentro de una unidad de trabajo:

PMSYP

Colocar mensaje con control de punto de sincronización.

La solicitud es operar dentro de los protocolos normales de unidad de trabajo. El mensaje no es visible fuera de la unidad de trabajo hasta que esta se haya confirmado. Si se cancela la unidad de trabajo, se elimina el mensaje.

Si no se especifican esta opción y PMNSYP, la solicitud put no se encuentra dentro de una unidad de trabajo.

PMSYP no debe especificarse con PMNSYP.

PMNSYP

Poner mensaje sin control de punto de sincronización.

La solicitud es operar fuera de los protocolos normales de unidad de trabajo. El mensaje está disponible de inmediato y no se puede eliminar revirtiendo una unidad de trabajo.

Si no se especifican esta opción y PMSYP, la solicitud put no se encuentra dentro de una unidad de trabajo.

PMNSYP no debe especificarse con PMSYP.

Opciones de identificador de mensaje e identificador de correlación : Las siguientes opciones solicitan al gestor de colas que genere un nuevo identificador de mensaje o identificador de correlación:

PMNMID

Generar un nuevo identificador de mensaje.

Esta opción hace que el gestor de colas sustituya el contenido del *MDMID* campo en MQMD por un nuevo identificador de mensaje. Este identificador de mensaje se envía con el mensaje y se devuelve a la aplicación en la salida de la llamada MQPUT o MQPUT1.

Esta opción también se puede especificar cuando el mensaje se envía a una lista de distribución; consulte la descripción del *PRMID* campo en la estructura MQPMR para obtener más detalles.

El uso de esta opción libera a la aplicación de la necesidad de restablecer el *MDMID* campo a MINONE antes de cada llamada a MQPUT o MQPUT1.

PMNCID

Generar un nuevo identificador de correlación.

Esta opción hace que el gestor de colas sustituya el contenido del *MDCID* campo en MQMD por un nuevo identificador de correlación. Este identificador de correlación se envía con el mensaje y se devuelve a la aplicación en la salida de la llamada MQPUT o MQPUT1.

Esta opción también se puede especificar cuando el mensaje se envía a una lista de distribución; consulte la descripción del *PRCID* campo en la estructura MQPMR para obtener más detalles.

El PMNCID es útil en situaciones en las que la aplicación requiere un identificador de correlación único.

Opciones de grupo y segmento : La siguiente opción se refiere al procesamiento de mensajes en grupos y segmentos de mensajes lógicos. Estas definiciones pueden ser útiles para comprender la opción:

Mensaje físico

Es la unidad más pequeña de información que se puede colocar o eliminar de una cola; a menudo corresponde a la información especificada o recuperada en una sola llamada MQPUT, MQPUT1 o MQGET. Cada mensaje físico tiene su propio descriptor de mensaje (MQMD). Por lo general, los mensajes físicos se distinguen por valores diferentes para el identificador de mensaje (*MDMID* campo en MQMD), aunque esto no lo impone el gestor de colas.

Mensaje lógico

Esta es una sola unidad de información de la solicitud. En ausencia de restricciones del sistema, un mensaje lógico sería igual que un mensaje físico. Sin embargo, cuando los mensajes lógicos son grandes, las limitaciones del sistema pueden hacer que sea aconsejable o necesario dividir un mensaje lógico en dos o más mensajes físicos, denominados *segmentos*.

Un mensaje lógico que ha sido segmentado consta de dos o más mensajes físicos que tienen el mismo identificador de grupo no nulo (*MDGID* campo en MQMD) y el mismo número de secuencia de mensajes (*MDSEQ* campo en MQMD). Los segmentos se distinguen por valores diferentes para el desplazamiento del segmento (*MDOFF* campo en MQMD), que proporciona el desplazamiento de los datos en el mensaje físico desde el inicio de los datos en el mensaje lógico. Dado que cada segmento es un mensaje físico, los segmentos de un mensaje lógico suelen tener identificadores de mensaje diferentes.

Un mensaje lógico que no ha sido segmentado, pero cuya segmentación ha sido permitida por la aplicación emisora, también tiene un identificador de grupo distinto de nulo, aunque en este caso solo hay un mensaje físico con ese identificador de grupo si el mensaje lógico no pertenece a un grupo de mensajes. Los mensajes lógicos cuya segmentación ha sido inhibida por la aplicación emisora tienen un identificador de grupo nulo (GINONE), a menos que el mensaje lógico pertenezca a un grupo de mensajes.

Grupo de mensajes

Se trata de un conjunto de uno o más mensajes lógicos que tienen el mismo identificador de grupo no nulo. Los mensajes lógicos del grupo se distinguen por valores diferentes del número de secuencia del mensaje, que es un número entero comprendido entre 1 y n, donde n es el número de mensajes lógicos del grupo. Si uno o más de los mensajes lógicos están segmentados, hay más de n mensajes físicos en el grupo.

PMLOGO

Los mensajes en grupos y segmentos de mensajes lógicos se ordenan de forma lógica.

Esta opción indica al gestor de colas cómo la aplicación agrupa los mensajes en segmentos de mensajes lógicos. Sólo se puede especificar en la llamada MQPUT; no es válida en la llamada MQPUT1.

Si se especifica PMLOGO, indica que la aplicación utiliza llamadas MQPUT sucesivas para:

- Poner los segmentos de cada mensaje lógico en el orden de desplazamiento de segmento creciente, empezando desde 0, sin huecos.
- Coloque todos los segmentos en un mensaje lógico antes de colocarlos en el siguiente mensaje lógico.
- Poner los mensajes lógicos de cada grupo de mensajes en el orden de número de secuencia de mensaje creciente, empezando desde 1, sin huecos.
- Coloque todos los mensajes lógicos en un grupo de mensajes antes de colocar los mensajes lógicos en el siguiente grupo de mensajes.

Este orden se denomina «orden lógico».

Dado que la aplicación ha indicado al gestor de colas cómo agrupa los mensajes en segmentos y mensajes lógicos, la aplicación no tiene que mantener y actualizar la información sobre los grupos y segmentos en cada llamada MQPUT, ya que el gestor de colas se encarga de ello. En concreto, esto significa que la aplicación no necesita establecer los campos *MDGID*, *MDSEQ* *MDOFF* y en MQMD, ya que el gestor de colas los establece con los valores adecuados. La aplicación solo necesita establecer el *MDMFL* campo en MQMD para indicar cuándo los mensajes pertenecen a grupos o son segmentos de mensajes lógicos, y para indicar el último mensaje de un grupo o el último segmento de un mensaje lógico.

Una vez iniciado un grupo de mensajes o un mensaje lógico, las llamadas MQPUT posteriores deben especificar los indicadores MF* adecuados en *MDMFL* en MQMD. Si la aplicación intenta colocar un mensaje que no pertenece a un grupo cuando hay un grupo de mensajes sin terminar, o colocar un mensaje que no es un segmento cuando hay un mensaje lógico sin terminar, la llamada falla con el código de motivo RC2241 o RC2242, según corresponda. Sin embargo, el gestor de colas conserva la información sobre el grupo de mensajes actual o el mensaje lógico actual, y la aplicación puede terminarlos enviando un mensaje (posiblemente sin datos de mensaje de aplicación) especificando MFLMIG o MFLSEG según corresponda, antes de volver a emitir la llamada MQPUT para colocar el mensaje que no está en el grupo o que no es un segmento.

Tabla 755 en la página 1202 muestra las combinaciones de opciones y indicadores que son válidas, y los valores de los campos *MDGID*, *MDSEQMDOFF*, y que el gestor de colas utiliza en cada caso. Las combinaciones de opciones y distintivos que no aparecen en la tabla no son válidas. Las columnas de la tabla tienen los siguientes significados:

LOG ORD

Indica si se ha especificado la opción PMLOGO en la llamada.

MIG

Indica si se ha especificado la opción MFMIG o MFLMIG en la llamada.

SEG

Indica si se ha especificado la opción MFSEG o MFLSEG en la llamada.

SEG OK

Indica si la opción MFSEGA se ha especificado en la llamada.

Cur grp

Indica si existe un grupo de mensajes actual antes de la llamada.

Cur log msg

Indica si existe un mensaje lógico actual antes de la llamada.

Otras columnas

Muestra los valores que utiliza el gestor de colas. «Anterior» indica el valor utilizado para el campo en el mensaje anterior para el identificador de cola.

PMRLOC

Especifica que el campo PMRQN de la estructura MQPMO debe completarse con el nombre de la cola local en la que se coloca realmente el mensaje. El campo « ResolvedQMGrName » (Nombre del gestor de colas) se completa de forma similar con el nombre del gestor de colas local que aloja la cola local. Consulte OORLOQ para saber qué significa esto. Si un usuario está autorizado para realizar una inserción en una cola, entonces tiene la autoridad necesaria para especificar este indicador en la llamada MQPUT. No se necesita ninguna autorización especial.

Tabla 755. Opciones de MQPUT relacionadas con los mensajes de grupos y los segmentos de mensajes lógicos								
Opciones especificadas				Estado de grupo y mensaje lógico antes de la llamada		Valores que utiliza el gestor de colas		
LOG ORD	MIG	SEG	SEG OK	Cur grp	Cur log msg	MDGID	MDSEQ	MDOFF
Sí	No	No	No	No	No	GINONE	1	0
Sí	No	No	Sí	No	No	Nuevo ID de grupo	1	0
Sí	No	Sí	Sí o no	No	No	Nuevo ID de grupo	1	0
Sí	No	Sí	Sí o no	No	Sí	ID de grupo anterior	1	Desplazamiento anterior + longitud de segmento anterior
Sí	Sí	Sí o no	Sí o no	No	No	Nuevo ID de grupo	1	0
Sí	Sí	Sí o no	Sí o no	Sí	No	ID de grupo anterior	Número de secuencia anterior + 1	0

Tabla 755. Opciones de MQPUT relacionadas con los mensajes de grupos y los segmentos de mensajes lógicos (continuación)

Opciones especificadas				Estado de grupo y mensaje lógico antes de la llamada		Valores que utiliza el gestor de colas		
Sí	Sí	Sí	Sí o no	Sí	Sí	ID de grupo anterior	Número de secuencia anterior	Desplazamiento anterior + longitud de segmento anterior
No	No	No	No	Sí o no	Sí o no	GINONE	1	0
No	No	No	Sí	Sí o no	Sí o no	Nuevo ID de grupo si es GINONE, de lo contrario, valor en el campo	1	0
No	No	Sí	Sí o no	Sí o no	Sí o no	Nuevo ID de grupo si es GINONE, de lo contrario, valor en el campo	1	Valor en campo
No	Sí	No	Sí o no	Sí o no	Sí o no	Nuevo ID de grupo si es GINONE, de lo contrario, valor en el campo	Valor en campo	0
No	Sí	Sí	Sí o no	Sí o no	Sí o no	Nuevo ID de grupo si es GINONE, de lo contrario, valor en el campo	Valor en campo	Valor en campo

Nota:

- PMLOGO no es válido en la llamada MQPUT1.
- Para el *MDMID* campo, el gestor de colas genera un nuevo identificador de mensaje si se especifica PMNMID o MINONE, y utiliza el valor del campo en caso contrario.
- Para el *MDCID* campo, el gestor de colas genera un nuevo identificador de correlación si se especifica PMNCID y, en caso contrario, utiliza el valor del campo.

Cuando se especifica PMLOGO, el gestor de colas requiere que todos los mensajes de un grupo y los segmentos de un mensaje lógico se coloquen con el mismo valor en el *MDPER* campo de MQMD, es decir, todos deben ser persistentes o todos deben ser no persistentes. Si no se cumple esta condición, la llamada MQPUT falla con el código de motivo RC2185 .

La opción PMLOGO afecta a las unidades de trabajo de la siguiente manera:

- Si el primer mensaje físico de un grupo o mensaje lógico se incluye en una unidad de trabajo, todos los demás mensajes físicos del grupo o mensaje lógico deben incluirse en una unidad de trabajo, si se utiliza el mismo identificador de cola. Sin embargo, no es necesario que se incluyan en la misma unidad de trabajo. Esto permite dividir un grupo de mensajes o un mensaje lógico compuesto por muchos mensajes físicos en dos o más unidades de trabajo consecutivas para el identificador de cola.
- Si el primer mensaje físico de un grupo o mensaje lógico no se coloca dentro de una unidad de trabajo, ninguno de los demás mensajes físicos del grupo o mensaje lógico se puede colocar dentro de una unidad de trabajo, si se utiliza el mismo descriptor de contexto de cola.

Si no se cumplen estas condiciones, la llamada MQPUT falla con el código de motivo RC2245 .

Cuando se especifica PMLOGO, el MQMD suministrado en la llamada MQPUT no debe ser inferior a MDVER2. Si no se cumple esta condición, la llamada falla con el código de motivo RC2257 .

Si no se especifica PMLOGO, los mensajes en grupos y segmentos de mensajes lógicos se pueden colocar en cualquier orden, y no es necesario colocar grupos de mensajes completos ni mensajes lógicos completos. Es responsabilidad de la aplicación garantizar que los campos *MDGID*, *MDSEQ*, *MDOFFMDMFL* y tengan valores adecuados.

Esta es la técnica que se puede utilizar para reiniciar un grupo de mensajes o un mensaje lógico a mitad de camino, después de que se haya producido un fallo del sistema. Cuando el sistema se reinicia, la aplicación puede establecer los campos *MDGID*, *MDSEQ*, *MDOFFMDPER* *MDMFL*, y con los valores adecuados y, a continuación, emitir la llamada MQPUT con PMSYP o PMNSYP establecidos según *sea necesario*, pero sin especificar PMLOGO. Si esta llamada tiene éxito, el gestor de colas conserva la información del grupo y del segmento, y las llamadas MQPUT posteriores que utilicen ese identificador de cola pueden especificar PMLOGO con normalidad.

La información de grupo y segmento que el gestor de colas retiene para la llamada MQPUT está separada de la información de grupo y segmento que retiene para la llamada MQGET.

Para cualquier identificador de cola dado, la aplicación puede mezclar llamadas MQPUT que especifican PMLOGO con llamadas MQPUT que no lo especifican, pero se deben tener en cuenta los siguientes puntos:

- Si no se especifica PMLOGO, cada llamada MQPUT correcta hace que el gestor de colas establezca la información de grupo y segmento para el identificador de cola en los valores especificados por la aplicación; esto sustituye la información de grupo y segmento existente que conserva el gestor de colas para el identificador de cola.
- Si no se especifica PMLOGO, la llamada no falla si hay un grupo de mensajes actual o un mensaje lógico; sin embargo, la llamada podría tener éxito con un código de finalización CCWARN. La [Tabla 756 en la página 1204](#) muestra los distintos casos que se pueden presentar. En estos casos, si el código de finalización no es CCOK, el código de motivo es uno de los siguientes (según corresponda):
 - RC2241
 - RC2242
 - RC2185
 - RC2245

Nota: El gestor de colas no comprueba la información de grupo y de segmento para la llamada MQPUT1.

<i>Tabla 756. Resultado cuando la llamada MQPUT o MQCLOSE no es coherente con la información de grupo y segmento</i>		
La llamada actual es	La llamada anterior fue MQPUT con PMLOGO	La llamada anterior fue MQPUT sin PMLOGO
MQPUT con PMLOGO	CCFAIL	CCFAIL
MQPUT sin PMLOGO	CCWARN	CCOK
MQCLOSE con un grupo o mensaje lógico sin terminar	CCWARN	CCOK

Se recomienda a las aplicaciones que simplemente desean ordenar mensajes y segmentos de forma lógica que especifiquen PMLOGO, ya que es la opción más sencilla de usar. Esta opción hace que la aplicación no tenga que gestionar la información de grupo y segmento, pues el gestor de colas lo hace en su lugar. Sin embargo, las aplicaciones especializadas pueden necesitar más control del que proporciona la opción PMLOGO, lo cual se puede lograr sin especificar dicha

opción. Si se hace esto, la aplicación debe asegurarse de que los campos *MDGID*, *MDSEQ* *MDOFF*, *MDMFL* y en *MQMD* estén configurados correctamente antes de cada llamada a *MQPUT* o *MQPUT1*.

Por ejemplo, una aplicación que desee reenviar mensajes físicos que recibe, independientemente de si dichos mensajes se encuentran en grupos o segmentos de mensajes lógicos, no debe especificar *PMLOGO*. Hay dos razones para ello:

- Si los mensajes se recuperan y se ordenan, al especificar *PMLOGO* se asigna un nuevo identificador de grupo a los mensajes, lo que puede dificultar o imposibilitar que el remitente de los mensajes correlacione cualquier mensaje de respuesta o informe que resulte del grupo de mensajes.
- En una red compleja con múltiples rutas entre los gestores de colas de emisión y recepción, los mensajes físicos pueden llegar fuera de secuencia. Al no especificar *PMLOGO* y el *GMLOGO* correspondiente en la llamada *MQGET*, la aplicación de reenvío puede recuperar y reenviar cada mensaje físico tan pronto como llega, sin necesidad de esperar a que llegue el siguiente en orden lógico.

Las aplicaciones que generan mensajes de informe para mensajes en grupos o segmentos de mensajes lógicos tampoco deben especificar *PMLOGO* al insertar el mensaje de informe.

PMLOGO se puede especificar con cualquiera de las otras opciones *PM**.

Opciones de contexto : Las siguientes opciones controlan el procesamiento del contexto del mensaje:

PMNOC

No se debe asociar ningún contexto al mensaje.

Tanto el contexto de identidad como el de origen están configurados para indicar que no hay contexto. Esto significa que los campos de contexto en *MQMD* se establecen en:

- Espacios en blanco para campos de caracteres
- Valores nulos para campos de bytes
- Ceros para campos numéricos

PMDEFC

Utilizar el contexto predeterminado.

El mensaje debe tener asociada información contextual predeterminada, tanto sobre la identidad como sobre el origen. El gestor de colas establece los campos de contexto en el descriptor de mensajes de la siguiente manera:

Tabla 757. Valores predeterminados de la información contextual para los campos MQMD

Campo de MQMD	Valor utilizado
<i>MDUID</i>	Determinado a partir del entorno si es posible; establecido en blanco en caso contrario.
<i>MDACC</i>	Determinado a partir del entorno si es posible; establecido en <i>ACNONE</i> en caso contrario.
<i>MDAID</i>	Establecido en blancos.
<i>MDPAT</i>	Determinado por el entorno.
<i>MDPAN</i>	Determinado a partir del entorno si es posible; establecido en blanco en caso contrario.
<i>MDPD</i>	Establecer la fecha en la que se envía el mensaje.
<i>MDPT</i>	Establecer la hora en la que se envía el mensaje.
<i>MDAOD</i>	Establecido en blancos.

Para obtener más información sobre el contexto de los mensajes, consulte [Contexto de los mensajes](#) y [Control de la información de contexto](#).

Esta es la acción predeterminada si no se especifican opciones de contexto.

PMPASI

Pasar el contexto de identidad desde un identificador de cola de entrada.

El mensaje debe tener información contextual asociada. El contexto de identidad se toma del identificador de cola especificado en el *PMCT* campo. La información sobre el contexto de origen es generada por el gestor de colas de la misma manera que para PMDEFC (véase la tabla anterior para los valores). Para obtener más información sobre el contexto de los mensajes, consulte [Contexto de los mensajes](#) y [Control de la información de contexto](#).

Para la llamada MQPUT, la cola debe haberse abierto con la opción OOPASI (o una opción que la implique). Para la llamada `MQPUT1`, se lleva a cabo la misma comprobación de autorización que para la llamada `MQOPEN` con la opción `OOPASI`.

PMPASA

Pasa todo el contexto desde un identificador de cola de entrada.

El mensaje debe tener información contextual asociada. Tanto la identidad como el contexto de origen se toman del identificador de cola especificado en el *PMCT* campo. Para obtener más información sobre el contexto de los mensajes, consulte [Contexto de los mensajes](#) y [Control de la información de contexto](#).

Para la llamada MQPUT, la cola debe haberse abierto con la opción OOPASA (o una opción que la implique). Para la llamada `MQPUT1`, se lleva a cabo la misma comprobación de autorización que para la llamada `MQOPEN` con la opción `OOPASA`.

PMSETI

Establecer el contexto de identidad desde la aplicación.

El mensaje debe tener información contextual asociada. La aplicación especifica el contexto de identidad en la estructura MQMD. La información sobre el contexto de origen es generada por el gestor de colas de la misma manera que para PMDEFC (véase la tabla anterior para los valores). Para obtener más información sobre el contexto de los mensajes, consulte [Contexto de los mensajes](#) y [Control de la información de contexto](#).

Para la llamada MQPUT, la cola debe haberse abierto con la opción OOSSETI (o una opción que la implique). Para la llamada `MQPUT1`, se lleva a cabo la misma comprobación de autorización que para la llamada `MQOPEN` con la opción `OOSSETI`.

PMSETA

Establecer todo el contexto desde la aplicación.

El mensaje debe tener información contextual asociada. La aplicación especifica la identidad y el contexto de origen en la estructura MQMD. Para obtener más información sobre el contexto de los mensajes, consulte [Contexto de los mensajes](#) y [Control de la información de contexto](#).

Para la llamada MQPUT, la cola debe haberse abierto con la opción OOSSETA. Para la llamada `MQPUT1`, se lleva a cabo la misma comprobación de autorización que para la llamada `MQOPEN` con la opción `OOSSETA`.

Solo se puede especificar una de las opciones de contexto PM*. Si no se especifica ninguna de estas opciones, se asume PMDEFC.

Poner tipos de respuesta. Las siguientes opciones controlan la respuesta devuelta a una llamada MQPUT o MQPUT1. Solo puede especificar una de estas opciones. Si no se especifican PMARES y PMSRES, se asume PMRASQ o PMRAST.

PMARES

La opción PMARES solicita que se complete una operación MQPUT o un MQPUT1 sin que la aplicación espere a que el gestor de colas complete la llamada. El uso de esta opción puede mejorar el rendimiento de la mensajería, especialmente en aplicaciones que utilizan enlaces de

cliente. Una aplicación puede comprobar periódicamente, utilizando el verbo MQSTAT, si se ha producido algún error durante llamadas asíncronas anteriores.

Con esta opción, solo se garantiza que se completen los siguientes campos en el MQMD:

- MDAID
- MDPAT
- MDPAN
- MDAOD

Además, si se especifican como opciones PMNMID o PMNCID, o ambas, también se completan los valores MDMID y MDCID devueltos. (El PMNMID se puede especificar implícitamente dejando en blanco el campo MDMID).

Solo se completan los campos especificados anteriormente. El resto de información que normalmente se devolvería en la estructura MQMD o MQPMO no está definida.

Al solicitar una respuesta asíncrona para MQPUT o MQPUT1, un CMPCOD y REASON de CCOK y RCNONE no significa necesariamente que el mensaje se haya colocado correctamente en una cola. Al desarrollar una aplicación de MQI que utilice respuestas asíncronas de inserción y requiera la confirmación de que los mensajes se han insertado en una cola, debes comprobar los códigos CMPCOD y REASON de las operaciones de inserción y utilizar también MQSTAT para consultar la información sobre errores asíncronos.

Aunque el éxito o el fracaso de cada llamada individual a MQPUT/MQPUT1 puede que no se devuelva inmediatamente, el primer error que se produjo en una llamada asíncrona se puede determinar más adelante mediante una llamada a MQSTAT.

Si un mensaje persistente en syncpoint no se entrega mediante una respuesta asíncrona put y se intenta confirmar la transacción, la confirmación falla y la transacción se revoca con un código de finalización CCFAIL y un motivo. RC2003 La aplicación puede realizar una llamada a MQSTAT para determinar la causa de un fallo anterior de MQPUT o MQPUT1

Si un mensaje no persistente bajo syncpoint no se entrega utilizando una respuesta asíncrona put y se intenta confirmar la transacción, la confirmación se realiza correctamente, pero el mensaje no persistente se pierde. Para evitar esta situación, la aplicación debe realizar una llamada a MQSTAT para determinar si ha habido algún fallo anterior de MQPUT o MQPUT1 antes de llamar a MQCMIT. Si se produjeran fallos, la aplicación debería llamar a MQBACK.

PMSRES

Especificar este valor para una opción de venta en la estructura MQPMO garantiza que la operación MQPUT o MQPUT1 siempre se emita de forma sincrónica. Si la operación se realiza correctamente, se completan todos los campos del MQMD y el MQPMO. Se proporciona para garantizar una respuesta sincrónica independientemente del valor de respuesta predeterminado definido en el objeto de cola o tema.

PMRASQ

Si se especifica este valor para una llamada MQPUT, el tipo de respuesta de inserción utilizado se toma del valor DEFPRESP especificado en la cola cuando fue abierta por la aplicación. Si una aplicación cliente está conectada a un gestor de colas en un nivel anterior a IBM WebSphere MQ 7.0, se comporta como si se hubiera especificado PMSRES.

Si se especifica esta opción para una llamada a MQPUT1, no se utiliza el valor DEFPRESP de la definición de la cola. Si la llamada MQPUT1 utiliza PMSYP, se comporta como PMARES, y si utiliza PMNSYP, se comporta como PMSRES.

PMRAST

Este es un sinónimo de PMRASQ para usar con objetos temáticos.

Otras opciones : Las siguientes opciones controlan la comprobación de autorizaciones y lo que ocurre cuando el gestor de colas está inactivo:

PMALTU

Validar con el identificador de usuario especificado.

Esto indica que el *ODAU* campo del **OBJDSC** parámetro de la llamada MQPUT1 contiene un identificador de usuario que se utilizará para validar la autoridad para colocar mensajes en la cola. La llamada solo puede tener éxito si está *ODAU* autorizado para abrir la cola con las opciones especificadas, independientemente de si el identificador de usuario con el que se ejecuta la aplicación está autorizado para hacerlo. (Sin embargo, esto no se aplica a las opciones de contexto especificadas, que siempre se comparan con el identificador de usuario bajo el cual se ejecuta la aplicación)

Esta opción solo es válida con la llamada MQPUT1.

PMFIQ

Falla si el gestor de colas está inactivo.

Esta opción fuerza el fallo de la llamada MQPUT o MQPUT1 si el gestor de colas se encuentra en estado de inactividad.

La llamada devuelve el código de finalización CCFAIL con el código de motivo RC2161 .

Opción predeterminada : Si no se requiere ninguna de las opciones descritas anteriormente, se puede utilizar la siguiente opción:

PMNONE

No se ha especificado ninguna opción.

Este valor se puede utilizar para indicar que no se han especificado otras opciones; todas las opciones adoptan sus valores predeterminados. PMNONE se define para facilitar la documentación del programa; no está previsto que esta opción se utilice con ninguna otra, pero como su valor es cero, dicho uso no puede detectarse.

Este es un campo de entrada. El valor inicial del *PMOPT* campo es PMNONE.

PMPRF (entero con signo de 10 dígitos)

Indicadores que señalan qué campos MQPMR están presentes.

Este campo contiene indicadores que deben establecerse para indicar qué campos MQPMR están presentes en los registros de mensajes enviados proporcionados por la aplicación. *PMPRF* Se utiliza solo cuando el mensaje se envía a una lista de distribución. El campo se ignora si *PMREC* es cero, o si tanto *PMPRO* como *PMPRP* son cero.

Para los campos que están presentes, el gestor de colas utiliza para cada destino los valores de los campos del registro de mensajes de envío correspondiente. Para los campos que no están presentes, el gestor de colas utiliza los valores de la estructura MQMD.

Se pueden especificar uno o varios de los siguientes indicadores para indicar qué campos están presentes en los registros del mensaje enviado:

PFMID

El campo identificador de mensaje está presente.

PFCID

El campo identificador de correlación está presente.

PFGID

El campo identificador de grupo está presente.

PFFB

El campo de comentarios está presente.

PFACC

El campo «Accounting-token» está presente.

Si se especifica este indicador, se debe especificar PMSETI o PMSETA en el *PMOPT* campo; si no se cumple esta condición, la llamada falla con el RC2158 código de motivo.

Si no hay campos MQPMR, se puede especificar lo siguiente:

PFNONE

No hay campos de registro de mensajes enviados.

Si se especifica este valor, debe ser cero *PMREC* , o bien *PMPRO* y *PMPRP* deben ser cero.

PFNONE se define para facilitar la documentación del programa. No está previsto que esta constante se utilice con ninguna otra, pero como su valor es cero, dicho uso no puede detectarse.

Si *PMPRF* contiene indicadores que no son válidos, o se proporcionan registros de mensajes pero *PMPRF* tiene el valor *PFNONE*, la llamada falla con el código de motivo RC2158 .

Este es un campo de entrada. El valor inicial de este campo es *PFNONE*. Este campo se ignora si *PMVER* es menor que *PMVER2*.

PMPRO (entero con signo de 10 dígitos)

Desplazamiento del primer registro de mensaje enviado desde el inicio de *MQPMO*.

Es el desplazamiento en bytes del primer registro de mensaje *MQPMR* desde el inicio de la estructura *MQPMO*. El desplazamiento puede ser positivo o negativo. *PMPRO* Se utiliza solo cuando el mensaje se envía a una lista de distribución. El campo se ignora si *PMREC* es cero.

Cuando el mensaje se envía a una lista de distribución, se puede proporcionar una matriz de uno o más registros de mensajes *MQPMR* para especificar determinadas propiedades del mensaje para cada destino individualmente; estas propiedades son:

- identificador de mensaje
- identificador de correlación
- Identificador de grupo
- valor de retroalimentación
- Señal de contabilidad

No es necesario especificar todas estas propiedades, pero sea cual sea el subconjunto elegido, los campos deben especificarse en el orden correcto. Consulte la descripción de la estructura *MQPMR* para obtener más detalles.

Normalmente, debería haber tantos registros de mensajes enviados como registros de objetos especificados por *MQOD* cuando se abre la lista de distribución; cada registro de mensajes enviados proporciona las propiedades del mensaje para la cola identificada por el registro de objetos correspondiente. Las colas de la lista de distribución que no se pueden abrir deben seguir teniendo registros de mensajes asignados en las posiciones adecuadas de la matriz, aunque en este caso se ignoran las propiedades del mensaje.

Es posible que el número de registros de mensajes enviados difiera del número de registros de objetos. Si hay menos registros de mensajes enviados que registros de objetos, las propiedades de los mensajes para los destinos que no tienen registros de mensajes enviados se toman de los campos correspondientes del descriptor de mensajes *MQMD*. Si hay más registros de mensajes que registros de objetos, los excedentes no se utilizan (aunque debe seguir siendo posible acceder a ellos). Los registros de mensajes son opcionales, pero si se proporcionan, deben ser *PMREC* .

Los registros de mensajes enviados se pueden proporcionar de forma similar a los registros de objetos en *MQOD*, ya sea especificando un desplazamiento en *PMPRO*, o especificando una dirección en *PMPRP* ; para obtener más información sobre cómo hacerlo, consulte el *ODORO* campo descrito en [“MQOD \(Descriptor de objeto\) activado IBM i” en la página 1183](#).

No se puede utilizar *PMPRP* más de uno de *PMPRO* y; la llamada falla con el código de motivo RC2159 si ambos son distintos de cero.

Este es un campo de entrada. El valor inicial de este campo es 0. Este campo se ignora si *PMVER* es menor que *PMVER2*.

PMPRP (puntero)

Dirección del primer registro de mensaje enviado.

Esta es la dirección del primer registro de mensajes *MQPMR*. *PMPRP* Se utiliza solo cuando el mensaje se envía a una lista de distribución. El campo se ignora si *PMREC* es cero.

Se puede utilizar *PMPRPPMPRO* o para especificar los registros de mensajes enviados, pero no ambos; consulte la descripción del campo *PMRRO* para obtener más detalles. Si no *PMPRP* se utiliza, debe establecerse en el puntero nulo o en bytes nulos.

Este es un campo de entrada. El valor inicial de este campo es el puntero nulo. Este campo se ignora si *PMVER* es menor que *PMVER2*.

PMREC (entero con signo de 10 dígitos)

Número de registros de mensajes enviados o registros de respuesta presentes.

Este es el número de registros de mensajes MQPMR o registros de respuesta MQRR que ha proporcionado la aplicación. Este número solo puede ser mayor que cero si el mensaje se envía a una lista de distribución. Los registros de mensajes y los registros de respuesta son opcionales: la aplicación no tiene por qué proporcionar ningún registro, o puede optar por proporcionar registros de un solo tipo. Sin embargo, si la aplicación proporciona registros de ambos tipos, deberá proporcionar *PMREC* registros de cada tipo.

El valor de no tiene *PMREC* por qué ser el mismo que el número de destinos de la lista de distribución. Si se proporcionan demasiados registros, los excedentes no se utilizan; si se proporcionan muy pocos registros, se utilizan valores predeterminados para las propiedades del mensaje para aquellos destinos que no tienen registros de mensajes introducidos (consulte *PMPRO* más adelante en este tema).

Si *PMREC* es menor que cero, o es mayor que cero pero el mensaje no se está colocando en una lista de distribución, la llamada falla con el código de motivo RC2154 .

Este es un campo de entrada. El valor inicial de este campo es 0. Este campo se ignora si *PMVER* es menor que *PMVER2*.

PMRMN (cadena de caracteres de 48 bytes)

Nombre resuelto del gestor de colas de destino.

Este es el nombre del gestor de colas de destino después de que el gestor de colas local haya realizado la resolución de nombres. El nombre devuelto es el nombre del gestor de colas propietario de la cola identificada por *PMRQN*, y puede ser el nombre del gestor de colas local.

Si *PMRQN* es una cola compartida que pertenece al grupo de uso compartido de colas al que pertenece el gestor de colas local, *PMRMN* es el nombre del grupo de uso compartido de colas. Si la cola pertenece a otro grupo de uso compartido de colas, *PMRQN* puede ser el nombre del grupo de uso compartido de colas o el nombre de un gestor de colas que sea miembro del grupo de uso compartido de colas (la naturaleza del valor devuelto viene determinada por las definiciones de colas que existen en el gestor de colas local).

Solo se devuelve un valor distinto de blanco si el objeto es una cola única; si el objeto es una lista de distribución o un tema, el valor devuelto es indefinido.

Se trata de un campo de salida. La longitud de este campo viene dada por *LNQMN*. El valor inicial de este campo es 48 caracteres en blanco.

PMRQN (cadena de caracteres de 48 bytes)

Nombre resuelto de la cola de destino.

Este es el nombre de la cola de destino después de que el administrador de colas local haya realizado la resolución de nombres. El nombre devuelto es el nombre de una cola que existe en el gestor de colas identificado por *PMRMN*.

Solo se devuelve un valor distinto de blanco si el objeto es una cola única; si el objeto es una lista de distribución o un tema, el valor devuelto es indefinido.

Se trata de un campo de salida. La longitud de este campo viene dada por *LNQN*. El valor inicial de este campo es 48 caracteres en blanco.

PMRRO (entero con signo de 10 dígitos)

Compensación del primer registro de respuesta desde el inicio de MQPMO.

Es el desplazamiento en bytes del primer registro de respuesta MQRR desde el inicio de la estructura MQPMO. El desplazamiento puede ser positivo o negativo. *PMRRO* Se utiliza solo cuando el mensaje se envía a una lista de distribución. El campo se ignora si *PMREC* es cero.

Cuando el mensaje se envía a una lista de distribución, se puede proporcionar una matriz de uno o más registros de respuesta MQRR para identificar las colas a las que no se envió correctamente el mensaje (*RRCC* campo en MQRR) y el motivo de cada fallo (*RRREA* campo en MQRR). Es posible que el mensaje no se haya enviado porque la cola no se ha podido abrir o porque la operación de inserción ha fallado. El gestor de colas establece los registros de respuesta solo cuando el resultado de la llamada es mixto (es decir, algunos mensajes se enviaron correctamente, mientras que otros fallaron, o todos fallaron pero por motivos diferentes); el código de motivo RC2136 de la llamada indica este caso. Si el mismo código de motivo se aplica a todas las colas, ese motivo se devuelve en el **REASON** parámetro de la llamada MQPUT o MQPUT1, y los registros de respuesta no se establecen.

Normalmente, debería haber tantos registros de respuesta como registros de objeto especificados por MQOD cuando se abre la lista de distribución; cuando sea necesario, cada registro de respuesta se establece en el código de finalización y el código de motivo para la inserción en la cola identificada por el registro de objeto correspondiente. Las colas de la lista de distribución que no se pueden abrir deben seguir teniendo registros de respuesta asignados en las posiciones adecuadas de la matriz, aunque se establezcan con el código de finalización y el código de motivo resultantes de la operación de apertura, en lugar de la operación de inserción.

Es posible que el número de registros de respuesta difiera del número de registros de objetos. Si hay menos registros de respuesta que registros de objetos, es posible que la aplicación no pueda identificar todos los destinos para los que ha fallado la operación de inserción, ni los motivos de los fallos. Si hay más registros de respuesta que registros de objetos, los excedentes no se utilizan (aunque debe seguir siendo posible acceder a ellos). Los registros de respuesta son opcionales, pero si se proporcionan, deben ser *PMREC*.

Los registros de respuesta se pueden proporcionar de forma similar a los registros de objetos en MQOD, ya sea especificando un desplazamiento en *PMRRO*, o especificando una dirección en *PMRRP*; para obtener más información sobre cómo hacerlo, consulte el *ODORO* campo descrito en “MQOD (Descriptor de objeto) activado IBM i” en la página 1183. Sin embargo, no se puede utilizar *PMRRP* más de uno de *PMRRO* y; la llamada falla con el código de motivo RC2156 si ambos son distintos de cero.

Para la llamada « MQPUT1 », este campo debe ser cero. Esto se debe a que la información de respuesta (si se solicita) se devuelve en los registros de respuesta especificados por el descriptor de objeto MQOD.

Este es un campo de entrada. El valor inicial de este campo es 0. Este campo se ignora si *PMVER* es menor que *PMVER2*.

PMRRP (puntero)

Dirección del registro de primera respuesta.

Esta es la dirección del primer registro de respuesta MQRR. *PMRRP* Se utiliza solo cuando el mensaje se envía a una lista de distribución. El campo se ignora si *PMREC* es cero.

Se puede utilizar *PMRRPPMRRO* o para especificar los registros de respuesta, pero no ambos; consulte la descripción del campo *PMRRO* para obtener más detalles. Si no *PMRRP* se utiliza, debe establecerse en el puntero nulo o en bytes nulos.

Para la llamada al método `MQPUT1`, este campo debe ser el puntero nulo o bytes nulos. Esto se debe a que la información de respuesta (si se solicita) se devuelve en los registros de respuesta especificados por el descriptor de objeto MQOD.

Este es un campo de entrada. El valor inicial de este campo es el puntero nulo. Este campo se ignora si *PMVER* es menor que *PMVER2*.

PMSID (cadena de caracteres de 4 bytes)

Identificador de estructura.

El valor debe ser:

PMSIDV

Identificador para la estructura de opciones de mensajes enviados.

Este campo siempre es un campo de entrada. El valor inicial de este campo es PMSIDV.

Me parto de risa (MQLONG)

El nivel de suscripción al que se dirige esta publicación.

Solo las suscripciones con el valor más alto *PMSL* inferior o igual a este valor reciben esta publicación. Este valor debe estar comprendido entre cero y 9; cero es el nivel más bajo.

El valor inicial de este campo es 9.

PMT0 (entero con signo de 10 dígitos)

Reservado.

Este es un campo reservado; su valor no es significativo. El valor inicial de este campo es -1.

PMUDC (entero con signo de 10 dígitos)

Número de mensajes enviados correctamente a colas remotas.

Este es el número de mensajes que la llamada MQPUT o MQPUT1 actual ha enviado correctamente a las colas de la lista de distribución que se resuelven en colas remotas. Los mensajes que el administrador de colas retiene temporalmente en forma de lista de distribución cuentan como el número de destinos individuales que contienen esas listas de distribución. Este campo también se establece cuando se envía un mensaje a una cola única que no está en una lista de distribución.

Se trata de un campo de salida. El valor inicial de este campo es 0. Este campo no se establece si *PMVER* es menor que *PMVER2*.

PMVER (entero con signo de 10 dígitos)

Número de versión de la estructura.

El valor debe ser uno de los siguientes:

PMVER1

Version-1 Estructura de opciones de put-message.

PMVER2

Version-2 Estructura de opciones de put-message.

Los campos que solo existen en la versión más reciente de la estructura se identifican como tales en las descripciones de los campos. La siguiente constante especifica el número de versión de la versión actual:

PMVERC

Versión actual de la estructura de opciones de put-message.

Este campo siempre es un campo de entrada. El valor inicial de este campo es PMVER1.

Valores iniciales

Tabla 758. Campos en MQPMO		
Nombre de campo	Nombre de la constante	Valor de la constante
<i>PMSID</i>	PMSIDV	'PMO→'
<i>PMVER</i>	PMVER1	1
<i>PMOPT</i>	PMNONE	0
<i>PMT0</i>	Ninguna	-1

Tabla 758. Campos en MQPMO (continuación)

Nombre de campo	Nombre de la constante	Valor de la constante
PMCT	Ninguna	0
PMKDC	Ninguna	0
PMUDC	Ninguna	0
PMIDC	Ninguna	0
PMRQN	Ninguna	Espacios en blanco
PMRMN	Ninguna	Espacios en blanco
PMREC	Ninguna	0
PMPRF	PFNONE	0
PMPRO	Ninguna	0
PMRRO	Ninguna	0
PMPRP	Ninguna	Puntero nulo o bytes nulos
PMRRP	Ninguna	Puntero nulo o bytes nulos
Nota: 1. El símbolo ~ representa un único carácter en blanco.		

Declaración RPG

```

D*.1.....2.....3.....4.....5.....6.....7..
D* MQPMO Structure
D*
D* Structure identifier
D  PMSID          1      4    INZ('PMO ')
D* Structure version number
D  PMVER          5      8I 0 INZ(1)
D* Options that control the action of MQPUT and MQPUT1
D  PMOPT          9     12I 0 INZ(0)
D* Reserved
D  PMTO          13     16I 0 INZ(-1)
D* Object handle of input queue
D  PMCT          17     20I 0 INZ(0)
D* Number of messages sent successfully to local queues
D  PMKDC         21     24I 0 INZ(0)
D* Number of messages sent successfully to remote queues
D  PMUDC         25     28I 0 INZ(0)
D* Number of messages that could not be sent
D  PMID         29     32I 0 INZ(0)
D* Resolved name of destination queue
D  PMRQN         33      80    INZ
D* Resolved name of destination queue manager
D  PMRMN         81     128    INZ
D* Number of put message records or response records present
D  PMREC        129     132I 0 INZ(0)
D* Flags indicating which MQPMR fields are present
D  PMPRF        133     136I 0 INZ(0)
D* Offset of first put message record from start of MQPMO
D  PMPRO        137     140I 0 INZ(0)
D* Offset of first response record from start of MQPMO
D  PMRRO        141     144I 0 INZ(0)
D* Address of first put message record
D  PMPRP        145     160*   INZ(*NULL)
D* Address of first response record
D  PMRRP        161     176*   INZ(*NULL)
D* Original message handle
D  PMOMH        177     184I 0
D* New message handle
D  PMNMH        185     190I 0
D* The action being performed
    
```

D	PMACT	191	194I 0
D*	Reserved		
D	PMRE1	195	198I 0

IBM i MQPMR (Registro de colocación de mensajes) en IBM i

La estructura MQPMR se utiliza para especificar varias propiedades de mensaje para un único destino cuando se transfiere un mensaje a una lista de distribución.

Visión general

Finalidad: MQPMR es una estructura de entrada/salida para las llamadas MQPUT y MQPUT1 .

Conjunto de caracteres y codificación: Los datos de MQPMR deben estar en el juego de caracteres proporcionado por el atributo de gestor de colas **CodedCharSetId** y la codificación del gestor de colas local proporcionada por ENNAT. Sin embargo, si la aplicación se ejecuta como un cliente IBM MQ , la estructura debe estar en el juego de caracteres y la codificación del cliente.

Uso: al proporcionar una matriz de estas estructuras en la llamada MQPUT o MQPUT1 , es posible especificar valores diferentes para cada cola de destino en una lista de distribución. Algunos de los campos son sólo de entrada, otros son de entrada/salida.

Nota: Esta estructura es inusual en que no tiene un diseño fijo. Los campos de esta estructura son opcionales, y la presencia o ausencia de cada campo se indica mediante los distintivos del campo *PMPRF* en MQPMO. Los campos que están presentes **deben aparecer en el siguiente orden** :

- *PRMID*
- *PRCID*
- *PRGID*
- *PRFB*
- *PRACC*

Los campos ausentes no ocupan ningún espacio en el registro.

Puesto que MQPMR no tiene un diseño fijo, no se proporciona ninguna definición del mismo en el archivo COPY. El programador de aplicaciones debe crear una declaración que contenga los campos necesarios para la aplicación y establecer los distintivos en *PMPRF* para indicar los campos que están presentes.

- [“Campos” en la página 1214](#)
- [“Valores iniciales” en la página 1216](#)
- [“Declaración RPG” en la página 1216](#)

Campos

La estructura MQPMR contiene los campos siguientes; los campos se describen en **orden alfabético**:

PRACC (serie de bits de 32 bytes)

Señal de contabilidad.

Es la señal de contabilidad que se debe utilizar para el mensaje enviado a la cola con un nombre especificado por el elemento correspondiente en la matriz de estructuras MQOR proporcionada en la llamada MQOPEN o MQPUT1 . Se procesa de la misma forma que el campo *MDACC* en MQMD para una colocación en una sola cola. Consulte la descripción de *MDACC* en [“MQMD \(descriptor de mensaje\) en IBM i” en la página 1131](#) para obtener información sobre el contenido de este campo.

Si este campo no está presente, se utiliza el valor de MQMD.

Este es un campo de entrada.

PRCID (serie de bits de 24 bytes)

Identificador de correlación.

Es el identificador de correlación que se debe utilizar para el mensaje enviado a la cola con el nombre especificado por el elemento correspondiente en la matriz de estructuras MQOR proporcionada en la llamada MQOPEN o MQPUT1 . Se procesa de la misma forma que el campo *MDCID* en MQMD para una colocación en una sola cola.

Si este campo no está presente en el registro MQPMR, o hay menos registros MQPMR que destinos, el valor de MQMD se utiliza para los destinos que no tienen un registro MQPMR que contenga un campo *PRCID* .

Si se especifica PMNCID, se genera un *único* identificador de correlación nuevo y se utiliza para todos los destinos de la lista de distribución, independientemente de si tienen registros MQPMR. Esto es diferente de la forma en que se procesa PMNMID (consulte el campo *PRMID*).

Es un campo de entrada/salida.

PRFB (entero con signo de 10 dígitos)

Código de retorno o de razón.

Este es el código de retorno que se debe utilizar para el mensaje enviado a la cola con el nombre especificado por el elemento correspondiente en la matriz de estructuras MQOR proporcionada en la llamada MQOPEN o MQPUT1 . Se procesa de la misma forma que el campo *MDFB* en MQMD para una colocación en una sola cola.

Si este campo no está presente, se utiliza el valor de MQMD.

Este es un campo de entrada.

PRGID (serie de bits de 24 bytes)

Identificador de grupo.

Es el identificador de grupo que se debe utilizar para el mensaje enviado a la cola con el nombre especificado por el elemento correspondiente en la matriz de estructuras MQOR proporcionada en la llamada MQOPEN o MQPUT1 . Se procesa de la misma forma que el campo *MDGID* en MQMD para una colocación en una sola cola.

Si este campo no está presente en el registro MQPMR, o hay menos registros MQPMR que destinos, el valor de MQMD se utiliza para los destinos que no tienen un registro MQPMR que contenga un campo *PRGID* . El valor se procesa tal como se documenta en [Tabla 755 en la página 1202](#), pero con las diferencias siguientes:

- En los casos en los que se utilizaría un nuevo identificador de grupo, el gestor de colas genera un identificador de grupo diferente para cada destino (es decir, ningún destino tiene el mismo identificador de grupo).
- En los casos en los que se utilizaría el valor del campo, la llamada falla con el código de razón RC2258.

Es un campo de entrada/salida.

PRMID (serie de bits de 24 bytes)

Identificador del mensaje.

Es el identificador de mensaje que se debe utilizar para el mensaje enviado a la cola con el nombre especificado por el elemento correspondiente en la matriz de estructuras MQOR proporcionada en la llamada MQOPEN o MQPUT1 . Se procesa de la misma forma que el campo *MDMID* en MQMD para una colocación en una sola cola.

Si este campo no está presente en el registro MQPMR, o hay menos registros MQPMR que destinos, el valor de MQMD se utiliza para los destinos que no tienen un registro MQPMR que contenga un campo *PRMID* . Si ese valor es MINONE, se genera un nuevo identificador de mensaje para *cada* de esos destinos (es decir, dos de esos destinos no tienen el mismo identificador de mensaje).

Si se especifica PMNMID, se generan nuevos identificadores de mensaje para todos los destinos de la lista de distribución, independientemente de si tienen registros MQPMR. Esto es diferente de la forma en que se procesa PMNCID (consulte el campo *PRCID*).

Es un campo de entrada/salida.

Valores iniciales

No hay valores iniciales definidos para esta estructura, ya que no se proporciona ninguna declaración de estructura. La siguiente declaración de ejemplo muestra cómo el programador de aplicaciones debe declarar la estructura si se necesitan todos los campos.

Declaración RPG

```
D*..1.....2.....3.....4.....5.....6.....7..  
D* MQPMR Structure  
D*  
D* Message identifier  
D PRMID 1 24  
D* Correlation identifier  
D PRCID 25 48  
D* Group identifier  
D PRGID 49 72  
D* Feedback or reason code  
D PRFB 73 76I 0  
D* Accounting token  
D PRACC 77 108
```

IBM i MQRFH (Reglas y encabezado de formato) en IBM i

La estructura MQRFH define el diseño de las reglas y el encabezado de formato.

Visión general

Propósito : Este encabezado se puede utilizar para enviar datos de cadena en forma de pares nombre-valor.

Nombre del formato : FMRFH.

Conjunto de caracteres y codificación : Los campos de la estructura MQRFH (incluido *RFNVS*) están en el conjunto de caracteres y la codificación especificados por los campos *MDCSI MDENC* y de la estructura de encabezado que precede a MQRFH, o por los campos de la estructura MQMD si MQRFH se encuentra al principio de los datos del mensaje de la aplicación.

El juego de caracteres debe ser uno que tenga caracteres de un solo byte para los caracteres que son válidos en los nombres de las colas.

- [“Campos” en la página 1216](#)
- [“Valores iniciales” en la página 1218](#)
- [“Declaración RPG” en la página 1219](#)

Campos

La estructura MQRFH contiene los siguientes campos, descritos en **orden alfabético** :

RFCSI (entero con signo de 10 dígitos)

Identificador del conjunto de caracteres de los datos que siguen *RFNVS*.

Esto especifica el identificador del conjunto de caracteres de los datos que siguen *RFNVS* ; no se aplica a los datos de caracteres de la propia estructura MQRFH.

En la llamada MQPUT o MQPUT1, la aplicación debe establecer este campo con el valor adecuado para los datos. Se puede utilizar el siguiente valor especial:

CSINHT

Heredar el identificador del conjunto de caracteres de esta estructura.

Los datos de caracteres en los datos *que siguen* esta estructura están en el mismo conjunto de caracteres que esta estructura.

El gestor de colas cambia este valor en la estructura enviada en el mensaje al identificador real del juego de caracteres de la estructura. Si no se produce ningún error, la llamada MQGET no devuelve el valor CSINHT.

CSINHT no se puede utilizar si el valor del *MDPAT* campo en MQMD es ATBRKR.

El valor inicial de este campo es CSUNDF.

Codificación numérica de los datos que siguen a *RFNVS*.

Esto especifica la codificación numérica de los datos que siguen *RFNVS* ; no se aplica a los datos numéricos de la propia estructura MQRFH.

En la llamada MQPUT o MQPUT1, la aplicación debe establecer este campo con el valor adecuado para los datos.

El valor inicial de este campo es ENNAT.

RFFLG (entero con signo de 10 dígitos)

Distintivos.

Se puede especificar lo siguiente:

RFNONE

Sin banderas.

El valor inicial de este campo es RFNONE.

RFFMT (cadena de caracteres de 8 bytes)

Formatear el nombre de los datos que siguen *RFNVS*.

Especifica el nombre del formato de los datos que siguen *RFNVS*.

En la llamada MQPUT o MQPUT1, la aplicación debe establecer este campo con el valor adecuado para los datos. Las reglas para codificar este campo son las mismas que para el *MDFMT* campo en MQMD.

El valor inicial de este campo es FMNONE.

RFLLEN (entero con signo de 10 dígitos)

Longitud total del MQRFH, incluyendo *RFNVS*.

Es la longitud en bytes de la estructura MQRFH, incluido el *RFNVS* campo al final de la estructura. La longitud no incluye ningún dato de usuario que siga al *RFNVS* campo.

Para evitar problemas con la conversión de datos de usuario en algunos entornos, considere utilizar *RFLLEN* como múltiplo de cuatro.

La siguiente constante indica la longitud de la parte *fija* de la estructura, es decir, la longitud excluyendo el *RFNVS* campo:

RFLENV

Longitud de la parte fija de la estructura MQRFH.

El valor inicial de este campo es RFLENV.

RFNVS (cadena de caracteres de n bytes)

Cadena que contiene pares nombre-valor.

Se trata de una cadena de caracteres de longitud variable que contiene pares nombre-valor con el siguiente formato:

```
name1 value1 name2 value2 name3 value3 ...
```

Cada nombre o valor debe estar separado del nombre o valor adyacente por uno o más caracteres en blanco; estos espacios en blanco no son significativos. Un nombre o valor puede contener espacios significativos anteponiendo y posponiendo el nombre o valor con el carácter de comillas; todos los caracteres entre la comilla de apertura y la comilla de cierre correspondiente se tratan como significativos. En el siguiente ejemplo, el nombre es FAMOUS_WORDS y el valor es Hello World:

```
FAMOUS_WORDS "Hello World"
```

Un nombre o valor puede contener cualquier carácter excepto el carácter nulo (que actúa como delimitador para *RFNVS*). Sin embargo, para facilitar la interoperabilidad, una aplicación podría preferir restringir los nombres a los siguientes caracteres:

- Primer carácter: letra mayúscula o minúscula (de la A a la Z, o de la a a la z), o guión bajo.
- Caracteres siguientes: letras mayúsculas o minúsculas, dígitos decimales (del 0 al 9), guión bajo, guión o punto.

Si un nombre o valor contiene una o más comillas, el nombre o valor debe estar entre comillas, y cada comilla dentro de la cadena debe duplicarse:

```
Famous_Words "The program displayed ""Hello World"""
```

Los nombres y los valores distinguen entre mayúsculas y minúsculas, es decir, las letras minúsculas no se consideran iguales a las mayúsculas. Por ejemplo, FAMOUS_WORDS y Famous_Words son dos nombres diferentes.

La longitud en bytes de *RFNVS* es igual a *RFLen* menos *RFLenV*. Para evitar problemas con la conversión de los datos de usuario en algunos entornos, se recomienda que esta longitud sea un múltiplo de cuatro. *RFNVS* debe rellenarse con espacios en blanco hasta alcanzar esta longitud, o terminarse antes colocando un carácter nulo después del último carácter significativo de la cadena. El carácter nulo y los bytes que le siguen, hasta la longitud especificada de *RFNVS*, se ignoran.

Nota: Debido a que la longitud de este campo no es fija, el campo se omite de las declaraciones de la estructura que se proporcionan para los lenguajes de programación compatibles.

RFID (cadena de caracteres de 4 bytes)

Identificador de estructura.

El valor debe ser:

RFIDV

Identificador para reglas y estructura de encabezado de formato.

El valor inicial de este campo es RFIDV.

RFVER (entero con signo de 10 dígitos)

Número de versión de la estructura.

El valor debe ser:

RFVER1

Version-1 normas y estructura de formato de los encabezados.

El valor inicial de este campo es RFVER1.

Valores iniciales

Tabla 759. Campos en MQRFH		
Nombre de campo	Nombre de la constante	Valor de la constante
<i>RFID</i>	RFIDV	'RFH↵'
<i>RFVER</i>	RFVER1	1
<i>RFLen</i>	RFLenV	32

Tabla 759. Campos en MQRFH (continuación)

Nombre de campo	Nombre de la constante	Valor de la constante
RFENC	ENNAT	Depende del entorno
RFCSI	CSUNDF	0
RFFMT	FMNONE	Espacios en blanco
RFFLG	RFNONE	0
Notas: 1. El símbolo ~ representa un único carácter en blanco.		

Declaración RPG

```

D*.1....2....3....4....5....6....7..
D* MQRFH Structure
D*
D* Structure identifier
D  RFSID          1      4    INZ('RFH ')
D* Structure version number
D  RFVER          5      8I 0 INZ(1)
D* Total length of MQRFH includingNameValueString
D  RFLEN          9     12I 0 INZ(32)
D* Numeric encoding of data that followsNameValueString
D  RFENC         13     16I 0 INZ(273)
D* Character set identifier of data thatfollows NameValueString
D  RFCSI         17     20I 0 INZ(0)
D* Format name of data that followsNameValueString
D  RFFMT         21     28    INZ(' ')
D* Flags
D  RFFLG         29     32I 0 INZ(0)
    
```

IBM i MQRFH2 (Reglas y formato del encabezado 2) en IBM i

La estructura MQRFH2 define el formato de las reglas version-2 y el encabezado de formato.

Visión general

Propósito : Este encabezado se puede utilizar para enviar datos que han sido codificados utilizando una sintaxis similar a XML. Un mensaje puede contener dos o más estructuras MQRFH2 en serie, con datos de usuario opcionalmente después de la última estructura MQRFH2 de la serie.

Nombre del formato : FMRFH2.

Conjunto de caracteres y codificación : Se aplican reglas especiales al conjunto de caracteres y la codificación utilizados para la estructura MQRFH2 :

- Los campos distintos de RF2NVD están en el conjunto de caracteres y la codificación que se indican en los campos MDCSI MDENC y de la estructura de encabezado que precede a MQRFH2, o en los campos de la estructura MQMD si el MQRFH2 se encuentra al principio de los datos del mensaje de la aplicación.

El juego de caracteres debe ser uno que tenga caracteres de un solo byte para los caracteres que son válidos en los nombres de las colas.

Cuando se especifica GMCONV en la llamada MQGET, el gestor de colas convierte estos campos al juego de caracteres y la codificación solicitados.

- RF2NVD está en el conjunto de caracteres dado por el RF2NVC campo. Solo ciertos conjuntos de caracteres Unicode son válidos para RF2NVC (consulte la descripción de RF2NVC para obtener más detalles).

Algunos juegos de caracteres tienen una representación que depende de la codificación. Si *RF2NVC* es uno de estos conjuntos de caracteres, *RF2NVD* debe tener la misma codificación que los demás campos del MQRFH2.

Cuando se especifica GMCONV en la llamada MQGET, el gestor de colas convierte *RF2NVD* a la codificación solicitada, pero no cambia su juego de caracteres.

- “Campos” en la [página 1220](#)
- “Valores iniciales” en la [página 1225](#)
- “Declaración RPG” en la [página 1225](#)

Campos

La estructura MQRFH2 contiene los siguientes campos, que se describen en orden alfabético:

RF2CSI (entero con signo de 10 dígitos)

Identificador del conjunto de caracteres de los datos que siguen al último *RF2NVD* campo.

Especifica el identificador del conjunto de caracteres de los datos que siguen al último *RF2NVD* campo. No se aplica a los datos de caracteres de la estructura MQRFH2 en sí misma

En la llamada MQPUT o MQPUT1, la aplicación debe establecer este campo con el valor adecuado para los datos. Se puede utilizar el siguiente valor especial:

CSINHT

Heredar el identificador del conjunto de caracteres de esta estructura.

Los datos de caracteres en los datos *que siguen* esta estructura están en el mismo conjunto de caracteres que esta estructura.

El gestor de colas cambia este valor en la estructura enviada en el mensaje al identificador real del juego de caracteres de la estructura. Si no se produce ningún error, la llamada MQGET no devuelve el valor CSINHT.

CSINHT no se puede utilizar si el valor del *MDPAT* campo en MQMD es ATBRKR.

El valor inicial de este campo es CSINHT.

RF2ENC (entero con signo de 10 dígitos)

Codificación numérica de los datos que siguen al último *RF2NVD* campo.

Especifica la codificación numérica de los datos que siguen al último *RF2NVD* campo; no se aplica a los datos numéricos de la estructura MQRFH2.

En la llamada MQPUT o MQPUT1, la aplicación debe establecer este campo con el valor adecuado para los datos.

El valor inicial de este campo es ENNAT.

RF2FLG (entero con signo de 10 dígitos)

Distintivos.

Se debe especificar el siguiente valor:

RFNONE

Sin banderas.

El valor inicial de este campo es RFNONE.

RF2FMT (cadena de caracteres de 8 bytes)

Formato del nombre de los datos que siguen al último *RF2NVD* campo.

Especifica el nombre del formato de los datos que siguen al último *RF2NVD* campo.

En la llamada MQPUT o MQPUT1, la aplicación debe establecer este campo con el valor adecuado para los datos. Las reglas para codificar este campo son las mismas que para el *MDFMT* campo en MQMD.

El valor inicial de este campo es FMNONE.

RF2LEN (entero con signo de 10 dígitos)

Longitud total de un MQRFH2, incluyendo todos los campos *RF2NVLRF2NVD* y.

Esta es la longitud en bytes de la estructura MQRFH2, incluidos los campos *RF2NVL RF2NVD* y al final de la estructura. Es válido que haya varios pares de campos *RF2NVL RF2NVD* y al final de la estructura, en la secuencia:

```
length1, data1, length2, data2, ...
```

RF2LEN no incluye ningún dato de usuario que pueda seguir al último *RF2NVD* campo al final de la estructura.

Para evitar problemas con la conversión de datos de usuario en algunos entornos, considere utilizar *RF2LEN* como múltiplo de cuatro.

La siguiente constante indica la longitud de la parte *fija* de la estructura, es decir, la longitud excluyendo los campos *RF2NVL RF2NVD* y :

RFLEN2

Longitud de la parte fija de una estructura MQRFH2.

El valor inicial de este campo es RFLEN2.

RF2NVC (entero con signo de 10 dígitos)

Identificador del conjunto de caracteres de *RF2NVD*.

Especifica el identificador del conjunto de caracteres codificados de los datos del *RF2NVD* campo. Esto difiere del conjunto de caracteres de las demás cadenas de la estructura MQRFH2 y puede diferir del conjunto de caracteres de los datos (si los hay) que siguen al último *RF2NVD* campo al final de la estructura.

RF2NVC debe tener uno de los siguientes valores CCSID:

1200

UTF-16, versión soportada más reciente de Unicode

13488

UTF-16, subconjunto de Unicode versión 2.0

17584

UTF-16, subconjunto de Unicode versión 3.0 (incluye símbolo del Euro)

1208

UTF-8, versión soportada más reciente de Unicode

Para los conjuntos de caracteres UTF-16, la codificación (orden de bytes) del *RF2NVD* debe ser la misma que la codificación de los demás campos de la estructura MQRFH2. Los caracteres sustitutos (X' D800 ' hasta X'DFFF') no son compatibles.

Nota: Si *RF2NVC* no tiene uno de los valores enumerados anteriormente y la estructura MQRFH2 requiere conversión en la llamada MQGET, la llamada se completa con el código de motivo RC2111 y el mensaje se devuelve sin convertir.

El valor inicial de este campo es 1208.

RF2NVD (cadena de caracteres de n bytes)

Datos de nombre/valor.

Se trata de una cadena de caracteres de longitud variable que contiene datos codificados utilizando una sintaxis similar a XML. La longitud en bytes de esta cadena la proporciona el *RF2NVL* campo que precede al *RF2NVD* campo; esta longitud debe ser un múltiplo de cuatro.

Los campos *RF2NVL* *RF2NVD* y son opcionales, pero si están presentes deben aparecer como un par y ser adyacentes. El par de campos se puede repetir tantas veces como sea necesario, por ejemplo:

```
length1 data1 length2 data2 length3 data3
```

Dado que estos campos son opcionales, se omiten en las declaraciones de la estructura que se proporcionan para los distintos lenguajes de programación compatibles.

RF2NVD Es inusual porque no se convierte al juego de caracteres especificado en la llamada MQGET cuando se recupera el mensaje con la opción GMCONV activa; *RF2NVD* permanece en su juego de caracteres original. Sin embargo, *RF2NVD* se convierte a la codificación especificada en la llamada MQGET.

Sintaxis de los datos nombre/valor : La cadena consta de una única «carpeta» que contiene cero o más propiedades. La carpeta está delimitada por etiquetas XML de inicio y fin con el mismo nombre que la carpeta:

```
<folder> property1 property2 ... </folder>
```

Los caracteres que siguen a la etiqueta de fin de carpeta, hasta la longitud definida por *RF2NVL*, deben estar en blanco. Dentro de la carpeta, cada propiedad se compone de un nombre y un valor, y opcionalmente un tipo de datos:

```
<name dt="datatype">value</name>
```

En estos ejemplos:

- Los caracteres delimitadores (<, =, ", / y >) deben especificarse exactamente como se muestran.
- name es el nombre especificado por el usuario para la propiedad; consulte el siguiente ejemplo para obtener más información sobre los nombres.
- datatype es un tipo de datos opcional especificado por el usuario de la propiedad; consulte el siguiente ejemplo para ver los tipos de datos válidos.
- value es el valor especificado por el usuario de la propiedad; consulte los párrafos siguientes para obtener más información sobre los valores.
- Los espacios en blanco son significativos entre el > carácter que precede a un valor y el < carácter que sigue al valor, y debe haber al menos un espacio en blanco antes de dt=. En otros casos, los espacios en blanco se pueden codificar libremente entre etiquetas, o antes o después de las etiquetas (por ejemplo, para mejorar la legibilidad); estos espacios en blanco no son significativos.

Si las propiedades están relacionadas entre sí, se pueden agrupar encerrándolas entre etiquetas XML de inicio y fin con el mismo nombre que el grupo:

```
<folder> <group> property1 property2 ... </group> </folder>
```

Los grupos se pueden anidar dentro de otros grupos, sin límite, y un grupo puede aparecer más de una vez dentro de una carpeta. También es válido que una carpeta contenga algunas propiedades en grupos y otras propiedades fuera de grupos.

Nombres de propiedades, grupos y carpetas : los nombres de propiedades, grupos y carpetas deben ser nombres de etiquetas XML válidos, con la excepción del carácter dos puntos, que no está permitido en un nombre de propiedad, grupo o carpeta. En concreto:

- Los nombres deben comenzar con una letra o un guión bajo. Las letras válidas se definen en la especificación XML « W3C » (Clasificación de caracteres Unicode) y consisten esencialmente en las categorías Unicode Ll, Lu, Lo, Lt y Nl.
- Los caracteres restantes de un nombre pueden ser letras, dígitos decimales, guiones bajos, guiones o puntos. Estos corresponden a las categorías Unicode Ll, Lu, Lo, Lt, Nl, Mc, Mn, Lm y Nd.
- Los caracteres compatibles con Unicode (X' F900 ' y superiores) no están permitidos en ninguna parte del nombre.

- Los nombres no deben comenzar con la cadena XML en ninguna combinación de mayúsculas o minúsculas.

Además:

- Los nombres distinguen entre mayúsculas y minúsculas. Por ejemplo, ABC, abc y Abc son tres nombres diferentes.
- Cada carpeta tiene un espacio de nombres independiente. Como resultado, un grupo o propiedad en una carpeta no entra en conflicto con un grupo o propiedad del mismo nombre en otra carpeta.
- Los grupos y las propiedades ocupan el mismo espacio de nombres dentro de una carpeta. Como resultado, una propiedad no puede tener el mismo nombre que un grupo dentro de la carpeta que contiene esa propiedad.

Por lo general, los programas que analizan el *RF2NVD* campo deben ignorar las propiedades o grupos cuyos nombres no reconoce el programa, siempre que dichas propiedades o grupos estén correctamente formados.

Tipos de datos de las propiedades : cada propiedad puede tener un tipo de datos opcional. Si se especifica, el tipo de datos debe ser uno de los siguientes valores, en mayúsculas, minúsculas o mixtas:

Tabla 760. Tipos de datos y su uso	
Tipo de datos	Se utiliza para
string	Cualquier secuencia de caracteres. Algunos caracteres deben especificarse utilizando secuencias de escape.
boolean	El carácter 0 o 1 (1 denota TRUE).
bin.hex	Dígitos hexadecimales que representan octetos.
i1	Número entero en el rango de -128 a +127, expresado utilizando solo dígitos decimales y un signo opcional.
i2	Número entero en el rango de -32 768 a +32 767, expresado utilizando solo dígitos decimales y un signo opcional.
i4	Número entero en el rango comprendido entre -2 147 483 648 y +2 147 483 647, expresado utilizando únicamente dígitos decimales y un signo opcional.
i8	Número entero en el rango comprendido entre -9 223 372 036 854 775 808 y +9 223 372 036 854 775 807, expresado utilizando únicamente dígitos decimales y un signo opcional.
int	Número entero en el rango comprendido entre -9 223 372 036 854 775 808 y +9 223 372 036 854 775 807, expresado utilizando únicamente dígitos decimales y un signo opcional. Esto se puede utilizar en lugar de i1, i2, i4, o i8 si el remitente no desea indicar una precisión concreta.

Tabla 760. Tipos de datos y su uso (continuación)	
Tipo de datos	Se utiliza para
r4	Número de coma flotante con magnitud en el rango 1.175E-37 a 3.402 823 47E+38, expresado mediante dígitos decimales, signo opcional, dígitos fraccionarios opcionales y exponente opcional.
r8	Número de coma flotante con magnitud en el rango 2.225E-307 a 1.797 693 134 862 3E+308 expresado mediante dígitos decimales, signo opcional, dígitos fraccionarios opcionales y exponente opcional.

Valores de las propiedades : El valor de una propiedad puede estar compuesto por cualquier carácter, excepto los caracteres especiales que tienen una secuencia de escape asociada obligatoria. Cada aparición del valor de un carácter marcado como «obligatorio» en la siguiente tabla debe sustituirse por la secuencia de escape correspondiente. La tabla también muestra caracteres que tienen una secuencia de escape opcional asociada. Cada aparición del valor de un carácter marcado como «opcional» puede sustituirse por la secuencia de escape correspondiente, pero no es obligatorio.

Tabla 761. Caracteres escapados y su uso		
Carácter	Secuencia de escape	Utilización
&	&	Obligatorio
<	<	Obligatorio
>	>	Opcional
"	"	Opcional
'	'	Opcional

Nota: El & carácter al inicio de una secuencia de escape no debe ser sustituido por & ; .

En el siguiente ejemplo, los espacios en blanco del valor son significativos; sin embargo, no se necesitan secuencias de escape:

```
<Famous_Words>The program displayed "Hello World"</Famous_Words>
```

RF2NVL (entero con signo de 10 dígitos)

Longitud de *RF2NVD*.

Especifica la longitud en bytes de los datos del *RF2NVD* campo. Para evitar problemas con la conversión de los datos (si los hubiera) que siguen al *RF2NVD* campo, *RF2NVL* debería ser un múltiplo de cuatro.

Nota: Los campos *RF2NVL* *RF2NVD* y son opcionales, pero si están presentes deben aparecer como un par y ser adyacentes. El par de campos se puede repetir tantas veces como sea necesario, por ejemplo:

```
length1 data1 length2 data2 length3 data3
```

Dado que estos campos son opcionales, se omiten en las declaraciones de la estructura que se proporcionan para los distintos lenguajes de programación compatibles.

RF2SID (cadena de caracteres de 4 bytes)

Identificador de estructura.

El valor debe ser:

RFSIDV

Identificador para reglas y estructura de encabezado de formato.

El valor inicial de este campo es RFSIDV.

RF2VER (entero con signo de 10 dígitos)

Número de versión de la estructura.

El valor debe ser:

RFVER2

Version-2 normas y estructura de formato de los encabezados.

El valor inicial de este campo es RFVER2.

Valores iniciales

Tabla 762. Campos en MQRFH2		
Nombre de campo	Nombre de la constante	Valor de la constante
RF2SID	RFSIDV	'RFH↵'
RF2VER	RFVER2	2
RF2LEN	RFLen2	36
RF2ENC	ENNAT	Depende del entorno
RF2CSI	CSINHT	-2
RF2FMT	FMNONE	Espacios en blanco
RF2FLG	RFNONE	0
RF2NVC	Ninguna	1208

Notas:

1. El símbolo ↵ representa un único carácter en blanco.

Declaración RPG

```
D*..1.....2.....3.....4.....5.....6.....7..
D*
D* MQRFH2 Structure
D*
D* Structure identifier
D RF2SID          1          4    INZ('RFH ')
D* Structure version number
D RF2VER          5          8I 0 INZ(2)
D* Total length of MQRFH2 including allNameValueLength and
D* NameValueDatafields
D RF2LEN          9         12I 0 INZ(36)
D* Numeric encoding of data that followslast NameValueData field
D RF2ENC          13        16I 0 INZ(273)
D* Character set identifier of data thatfollows last NameValueData field
D RF2CSI          17        20I 0 INZ(-2)
D* Format name of data that follows lastNameValueData field
D RF2FMT          21        28    INZ(' ')
D* Flags
D RF2FLG          29        32I 0 INZ(0)
D* Character set identifier ofNameValueData
D RF2NVC          33        36I 0 INZ(1208)
```

MQRMH (Encabezado del mensaje de referencia) en IBM i

La estructura MQRMH define el formato del encabezado de un mensaje de referencia.

Visión general

Propósito : Este encabezado se utiliza con salidas de canal de mensajes escritas por el usuario para enviar grandes cantidades de datos (denominados "datos masivos") de un gestor de colas a otro. La diferencia con respecto a la mensajería normal es que los datos masivos no se almacenan en una cola, sino que solo se almacena una *referencia* a los datos masivos en la cola. Esto reduce la posibilidad de que unos pocos mensajes IBM MQ grandes agoten los recursos.

Nombre del formato : FMRMH.

Conjunto de caracteres y codificación : Los datos de caracteres en MQRMH y las cadenas a las que se refieren los campos de desplazamiento deben estar en el conjunto de caracteres del gestor de colas local; esto viene dado por el atributo del **CodedCharSetId** gestor de colas. Los datos numéricos en MQRMH deben estar en la codificación nativa de la máquina; esto viene dado por el valor de ENNAT para el lenguaje de programación C.

El juego de caracteres y la codificación del MQRMH deben establecerse en los campos *MDCSI MDENC* y en:

- El MQMD (si la estructura MQRMH se encuentra al inicio de los datos del mensaje), o
- La estructura de encabezado que precede a la estructura MQRMH (todos los demás casos).

Uso : Una aplicación envía un mensaje compuesto por un MQRMH, pero omitiendo los datos masivos. Cuando un agente de canal de mensajes (MCA) lee el mensaje de la cola de transmisión, se invoca una salida de mensaje proporcionada por el usuario para procesar el encabezado del mensaje de referencia. La salida puede añadir al mensaje de referencia los datos masivos identificados por la estructura MQRMH, antes de que el MCA envíe el mensaje a través del canal al siguiente gestor de colas.

En el extremo receptor, debe existir una salida de mensajes que espere los mensajes de referencia. Cuando se recibe un mensaje de referencia, la salida debe crear el objeto a partir de los datos masivos que siguen al MQRMH en el mensaje y, a continuación, transmitir el mensaje de referencia sin los datos masivos. El mensaje de referencia puede recuperarse posteriormente mediante una aplicación que lea el mensaje de referencia (sin los datos masivos) de una cola.

Normalmente, la estructura MQRMH es todo lo que hay en el mensaje. Sin embargo, si el mensaje se encuentra en una cola de transmisión, uno o más encabezados adicionales precederán a la estructura MQRMH.

También se puede enviar un mensaje de referencia a una lista de distribución. En este caso, la estructura MQDH y sus registros relacionados preceden a la estructura MQRMH cuando el mensaje se encuentra en una cola de transmisión.

Nota: No se debe enviar un mensaje de referencia como mensaje segmentado, ya que la salida del mensaje no puede procesarlo correctamente.

- [“Conversión de datos” en la página 1226](#)
- [“Campos” en la página 1227](#)
- [“Valores iniciales” en la página 1231](#)
- [“Declaración RPG” en la página 1232](#)

Conversión de datos

A efectos de conversión de datos, la conversión de la estructura MQRMH incluye la conversión de los datos del entorno de origen, el nombre del objeto de origen, los datos del entorno de destino y el nombre del objeto de destino. Cualquier otro byte dentro de *RMLEN* los bytes del inicio de la estructura se descarta o tiene valores indefinidos después de la conversión de datos. Los datos masivos se convertirán siempre que se cumplan todas las siguientes condiciones:

- Los datos masivos están presentes en el mensaje cuando se realiza la conversión de datos.

- El *RMFMT* campo en *MQRMH* tiene un valor distinto de *FMNONE*.
- Existe una salida de conversión de datos escrita por el usuario con el nombre de formato especificado.

Sin embargo, tenga en cuenta que, por lo general, los datos masivos no están presentes en el mensaje cuando este se encuentra en una cola y, por lo tanto, la opción *GMCONV* no los convertirá.

Campos

La estructura *MQRMH* contiene los siguientes campos, que se describen en **orden alfabético** :

RMCSI (entero con signo de 10 dígitos)

Identificador del conjunto de caracteres de datos masivos.

Especifica el identificador del conjunto de caracteres de los datos masivos; no se aplica a los datos de caracteres de la propia estructura *MQRMH*.

En la llamada *MQPUT* o *MQPUT1*, la aplicación debe establecer este campo con el valor adecuado para los datos. Se puede utilizar el siguiente valor especial:

CSINHT

Heredar el identificador del conjunto de caracteres de esta estructura.

Los datos de caracteres en los datos *que siguen* esta estructura están en el mismo conjunto de caracteres que esta estructura.

El gestor de colas cambia este valor en la estructura enviada en el mensaje al identificador real del juego de caracteres de la estructura. Si no se produce ningún error, la llamada *MQGET* no devuelve el valor *CSINHT*.

CSINHT no se puede utilizar si el valor del *MDPAT* campo en *MQMD* es *ATBRKR*.

El valor inicial de este campo es *CSUNDF*.

RMDEL (entero con signo de 10 dígitos)

Longitud de los datos del entorno de destino.

Si este campo es cero, no hay datos del entorno de destino y *RMDEO* se ignora.

RMDEO (entero con signo de 10 dígitos)

Compensación de los datos del entorno de destino.

Este campo especifica el desplazamiento de los datos del entorno de destino desde el inicio de la estructura *MQRMH*. El creador del mensaje de referencia puede especificar los datos del entorno de destino, si los conoce. Por ejemplo, los datos del entorno de destino podrían ser la ruta del directorio del objeto donde se almacenarán los datos masivos. Sin embargo, si el creador no conoce los datos del entorno de destino, es responsabilidad de la salida de mensajes proporcionada por el usuario determinar cualquier información del entorno que sea necesaria.

La longitud de los datos del entorno de destino viene dada por *RMDEL* ; si esta longitud es cero, no hay datos del entorno de destino y *RMDEO* se ignora. Si están presentes, los datos del entorno de destino deben residir completamente dentro de *RMLen* bytes desde el inicio de la estructura.

Las aplicaciones no deben asumir que los datos del entorno de destino son contiguos con ninguno de los datos a los que se refieren los *RMDNO* campos *RMSEO*, *RMSNO*, y.

El valor inicial de este campo es 0.

RMDL (entero con signo de 10 dígitos)

Longitud de los datos masivos.

El *RMDL* campo especifica la longitud de los datos masivos a los que hace referencia la estructura *MQRMH*.

Si los datos masivos están presentes en el mensaje, los datos comienzan en un desplazamiento de *RMLen* bytes desde el inicio de la estructura MQRMH. La longitud de todo el mensaje menos *RMLen* da la longitud de los datos masivos presentes.

Si hay datos en el mensaje, *RMDL* especifica la cantidad de esos datos que es relevante. Lo normal es que *RMDL* tenga el mismo valor que la longitud de los datos presentes en el mensaje.

Si la estructura MQRMH representa los datos restantes del objeto (a partir del desplazamiento lógico especificado), se puede utilizar el valor cero para *RMDL*, si los datos masivos no están presentes en el mensaje.

Si no hay datos, el final de MQRMH coincide con el final del mensaje.

El valor inicial de este campo es 0.

RMDNL (entero con signo de 10 dígitos)

Longitud del nombre del objeto de destino.

Si este campo es cero, no hay nombre de objeto de destino y *RMDNO* se ignora.

RMDNO (entero con signo de 10 dígitos)

Desplazamiento del nombre del objeto de destino.

Este campo especifica el desplazamiento del nombre del objeto de destino desde el inicio de la estructura MQRMH. El nombre del objeto de destino puede ser especificado por el creador del mensaje de referencia, si ese dato es conocido por el creador. Sin embargo, si el creador no conoce el nombre del objeto de destino, es responsabilidad de la salida de mensaje proporcionada por el usuario identificar el objeto que se va a crear o modificar.

La longitud del nombre del objeto de destino viene dada por *RMDNL* ; si esta longitud es cero, no hay nombre de objeto de destino y *RMDNO* se ignora. Si está presente, el nombre del objeto de destino debe residir completamente dentro de *RMLen* bytes desde el inicio de la estructura.

Las aplicaciones no deben asumir que el nombre del objeto de destino es contiguo a ninguno de los datos a los que se refieren los *RMSEORMDEO* campos, *RMSNO* y.

El valor inicial de este campo es 0.

RMDO (entero con signo de 10 dígitos)

Bajo desplazamiento de datos masivos.

Este campo especifica el desplazamiento bajo de los datos masivos desde el inicio del objeto del que forman parte dichos datos masivos. El desplazamiento de los datos masivos desde el inicio del objeto se denomina *desplazamiento* lógico. No se trata del desplazamiento físico de los datos masivos desde el inicio de la estructura MQRMH; ese desplazamiento viene dado por *RMLen*.

Para permitir el envío de objetos grandes mediante mensajes de referencia, el desplazamiento lógico se divide en dos campos, y el desplazamiento lógico real viene dado por la suma de estos dos campos:

- *RMDO* representa el resto obtenido al dividir el desplazamiento lógico entre 1 000 000 000. Por lo tanto, es un valor comprendido entre 0 y 999 999 999.
- *RMDO2* representa el resultado obtenido cuando el desplazamiento lógico se divide por 1 000 000 000. Por lo tanto, es el número de múltiplos completos de 1 000 000 000 que existen en el desplazamiento lógico. El número de múltiplos está comprendido entre 0 y 999 999 999.

El valor inicial de este campo es 0.

RMDO2 (entero con signo de 10 dígitos)

Alto desplazamiento de datos masivos.

Este campo especifica el desplazamiento alto de los datos masivos desde el inicio del objeto del que forman parte los datos masivos. Es un valor comprendido entre 0 y 999 999 999. Consulte *RMDO* para obtener detalles.

El valor inicial de este campo es 0.

RMENC (entero con signo de 10 dígitos)

Codificación numérica de datos masivos.

Esto especifica la codificación numérica de los datos masivos; no se aplica a los datos numéricos de la propia estructura MQRMH.

En la llamada MQPUT o MQPUT1, la aplicación debe establecer este campo con el valor adecuado para los datos.

El valor inicial de este campo es ENNAT.

RMFLG (entero con signo de 10 dígitos)

Indicadores de mensajes de referencia.

Se definen los siguientes indicadores:

RMLAST

El mensaje de referencia contiene o representa la última parte del objeto.

Esta bandera indica que el mensaje de referencia representa o contiene la última parte del objeto al que se hace referencia.

RMNLST

El mensaje de referencia no contiene ni representa la última parte del objeto.

RMNLST se define para ayudar a la documentación del programa. No está previsto que esta opción se utilice con ninguna otra, pero como su valor es cero, dicho uso no puede detectarse.

El valor inicial de este campo es RMNLST.

RMFMT (cadena de caracteres de 8 bytes)

Nombre del formato de los datos masivos.

Esto especifica el nombre del formato de los datos masivos.

En la llamada MQPUT o MQPUT1, la aplicación debe establecer este campo con el valor adecuado para los datos. Las reglas para codificar este campo son las mismas que para el *MDFMT* campo en MQMD.

El valor inicial de este campo es FMNONE.

RMLEN (entero con signo de 10 dígitos)

Longitud total de MQRMH, incluidas las cadenas al final de los campos fijos, pero sin los datos masivos.

El valor inicial de este campo es cero.

RMOII (cadena de bits de 24 bytes)

Identificador de instancia de objeto.

Este campo se puede utilizar para identificar una instancia específica de un objeto. Si no es necesario, se debe establecer el siguiente valor:

CEBOLLA

No se ha especificado ningún identificador de instancia de objeto.

El valor es un cero binario para la longitud del campo.

La longitud de este campo viene dada por LNOIID. El valor inicial de este campo es OIINON.

RMOT (cadena de caracteres de 8 bytes)

Tipo de objeto.

Este es un nombre que puede utilizar la salida de mensaje para reconocer los tipos de mensajes de referencia que admite. Considere la posibilidad de hacer que el nombre se ajuste a las mismas reglas que el *RMFMT* campo.

El valor inicial de este campo es 8 espacios en blanco.

RMSEL (entero con signo de 10 dígitos)

Longitud de los datos del entorno de origen.

Si este campo es cero, no hay datos del entorno de origen y *RMSEO* se ignora.

El valor inicial de este campo es 0.

RMSEO (entero con signo de 10 dígitos)

Compensación de los datos del entorno de origen.

Este campo especifica el desplazamiento de los datos del entorno de origen desde el inicio de la estructura MQRMH. Los datos del entorno de origen pueden ser especificados por el creador del mensaje de referencia, si dichos datos son conocidos por el creador. Por ejemplo, los datos del entorno de origen podrían ser la ruta del directorio del objeto que contiene los datos masivos. Sin embargo, si el creador no conoce los datos del entorno de origen, es responsabilidad de la salida de mensaje proporcionada por el usuario determinar cualquier información del entorno que sea necesaria.

La longitud de los datos del entorno de origen viene dada por *RMSEL* ; si esta longitud es cero, no hay datos del entorno de origen y *RMSEO* se ignora. Si están presentes, los datos del entorno de origen deben residir completamente dentro de *RMLen* bytes desde el inicio de la estructura.

Las aplicaciones no deben dar por sentado que los datos del entorno comienzan inmediatamente después del último campo fijo de la estructura ni que son contiguos a ninguno de los datos a los que hacen referencia los *RMDNO* campos *RMSNO*, *RMDEO* y.

El valor inicial de este campo es 0.

RMSID (cadena de caracteres de 4 bytes)

Identificador de estructura.

El valor debe ser:

RMSIDV

Identificador para la estructura del encabezado del mensaje de referencia.

El valor inicial de este campo es RMSIDV.

RMSNL (entero con signo de 10 dígitos)

Longitud del nombre del objeto fuente.

Si este campo es cero, no hay nombre de objeto de origen y *RMSNO* se ignora.

El valor inicial de este campo es 0.

RMSNO (entero con signo de 10 dígitos)

Desplazamiento del nombre del objeto de origen.

Este campo especifica el desplazamiento del nombre del objeto de origen desde el inicio de la estructura MQRMH. El nombre del objeto de origen puede ser especificado por el creador del mensaje de referencia, si ese dato es conocido por el creador. Sin embargo, si el creador no conoce el nombre del objeto de origen, es responsabilidad del mensaje de salida proporcionado por el usuario identificar el objeto al que se va a acceder.

La longitud del nombre del objeto fuente viene dada por *RMSNL* ; si esta longitud es cero, no hay nombre de objeto fuente y *RMSNO* se ignora. Si está presente, el nombre del objeto fuente debe residir completamente dentro de *RMLen* bytes desde el inicio de la estructura.

Las aplicaciones no deben asumir que el nombre del objeto fuente es contiguo a ninguno de los datos a los que hacen referencia los *RMSEORMDNO* campos, *RMDEO* y.

El valor inicial de este campo es 0.

RMVER (entero con signo de 10 dígitos)

Número de versión de la estructura.

El valor debe ser:

RMVER1

Version-1 Estructura del encabezado del mensaje de referencia.

La siguiente constante especifica el número de versión de la versión actual:

RMVERC

Versión actual de la estructura del encabezado del mensaje de referencia.

El valor inicial de este campo es RMVER1.

Valores iniciales

Tabla 763. Campos en MQRMH		
Nombre de campo	Nombre de la constante	Valor de la constante
<i>RMSID</i>	RMSIDV	'RMH↵'
<i>RMVER</i>	RMVER1	1
<i>RMLen</i>	Ninguna	0
<i>RMENC</i>	ENNAT	Depende del entorno
<i>RMCSI</i>	CSUNDF	0
<i>RMFMT</i>	FMNONE	Espacios en blanco
<i>RMFLG</i>	RMNLST	0
<i>RMOT</i>	Ninguna	Espacios en blanco
<i>RM0II</i>	CEBOLLA	Nulos
<i>RMSEL</i>	Ninguna	0
<i>RMSEO</i>	Ninguna	0
<i>RMSNL</i>	Ninguna	0
<i>RMSNO</i>	Ninguna	0
<i>RMDEL</i>	Ninguna	0
<i>RMDEO</i>	Ninguna	0
<i>RMDNL</i>	Ninguna	0
<i>RMDNO</i>	Ninguna	0
<i>RMDL</i>	Ninguna	0
<i>RMD0</i>	Ninguna	0
<i>RMD02</i>	Ninguna	0

Tabla 763. Campos en MQRMH (continuación)

Nombre de campo	Nombre de la constante	Valor de la constante
Notas: 1. El símbolo ~ representa un único carácter en blanco.		

```

D*..1.....2.....3.....4.....5.....6.....7..
D*
D* MQRMH Structure
D*
D* Structure identifier
D  RMSID          1      4    INZ('RMH ')
D* Structure version number
D  RMVER          5      8I 0 INZ(1)
D* Total length of MQRMH, including strings at end of fixed fields, but not
D* the bulk data
D  RMLen          9     12I 0 INZ(0)
D* Numeric encoding of bulk data
D  RMENC         13     16I 0 INZ(273)
D* Character set identifier of bulk data
D  RMCSI         17     20I 0 INZ(0)
D* Format name of bulk data
D  RMFMT         21     28    INZ('      ')
D* Reference message flags
D  RMFLG         29     32I 0 INZ(0)
D* Object type
D  RMOT          33     40    INZ
D* Object instance identifier
D  RMOII         41     64    INZ(X'00000000000000-
D                          00000000000000000000-
D                          000000000000')
D* Length of source environment data
D  RMSEL         65     68I 0 INZ(0)
D* Offset of source environment data
D  RMSEO         69     72I 0 INZ(0)
D* Length of source object name
D  RMSNL         73     76I 0 INZ(0)
D* Offset of source object name
D  RMSNO         77     80I 0 INZ(0)
D* Length of destination environment data
D  RMDEL         81     84I 0 INZ(0)
D* Offset of destination environment data
D  RMDEO         85     88I 0 INZ(0)
D* Length of destination object name
D  RMDNL         89     92I 0 INZ(0)
D* Offset of destination object name
D  RMDNO         93     96I 0 INZ(0)
D* Length of bulk data
D  RMDL          97    100I 0 INZ(0)
D* Low offset of bulk data
D  RMDO         101    104I 0 INZ(0)
D* High offset of bulk data
D  RMDO2        105    108I 0 INZ(0)

```

Declaración RPG

MQRR (Registro de respuesta) en IBM i

La estructura MQRR se utiliza para recibir el código de finalización y el código de motivo resultantes de la operación de apertura o inserción para una sola cola de destino, cuando el destino es una lista de distribución.

Visión general

Propósito : MQRR es una estructura de salida para las llamadas MQOPEN, MQPUT y MQPUT1.

Conjunto de caracteres y codificación : Los datos en MQRR deben estar en el conjunto de caracteres dado por el atributo del **CodedCharSetId** gestor de colas y la codificación del gestor de colas local dada

por ENNAT. Sin embargo, si la aplicación se ejecuta como un IBM MQ cliente, la estructura debe estar en el juego de caracteres y la codificación del cliente.

Uso : Al proporcionar una matriz de estas estructuras en las llamadas MQOPEN y MQPUT, o en la llamada MQPUT1, es posible determinar los códigos de finalización y los códigos de motivo para todas las colas de una lista de distribución cuando el resultado de la llamada es mixto, es decir, cuando la llamada tiene éxito para algunas colas de la lista, pero falla para otras. El código de motivo « RC2136 » de la llamada indica que los registros de respuesta (si los proporciona la aplicación) han sido establecidos por el gestor de colas.

- “Campos” en la página 1233
- “Valores iniciales” en la página 1233
- “Declaración RPG” en la página 1233

Campos

La estructura MQRR contiene los siguientes campos, que se describen en **orden alfabético** :

RRCC (entero con signo de 10 dígitos)

Código de finalización para la cola.

Este es el código de finalización resultante de la operación de apertura o inserción para la cola con el nombre especificado por el elemento correspondiente en la matriz de estructuras MQOR proporcionada en la llamada MQOPEN o MQPUT1.

Este campo siempre es de salida. El valor inicial de este campo es CCOK.

RRREA (entero con signo de 10 dígitos)

Código de motivo para la cola.

Este es el código de motivo resultante de la operación de apertura o inserción para la cola con el nombre especificado por el elemento correspondiente en la matriz de estructuras MQOR proporcionada en la llamada MQOPEN o MQPUT1.

Este campo siempre es de salida. El valor inicial de este campo es RCNONE.

Valores iniciales

Tabla 764. Campos en MQRR		
Nombre de campo	Nombre de la constante	Valor de la constante
RRCC	CCOK	0
RRREA	RCNONE	0

Declaración RPG

```
D*..1.....2.....3.....4.....5.....6.....7..
D*
D* MQRR Structure
D*
D* Completion code for queue
D  RRCC              1      4I 0 INZ(0)
D* Reason code for queue
D  RRRR              5      8I 0 INZ(0)
```

IBM i MQSCO (opciones de configuración de TLS) en IBM i

La estructura MQSCO (que incluye los campos « TLS » de la estructura MQCD) permite a una aplicación que se ejecuta como un IBM MQ MQI client especificar opciones de configuración que controlan el uso de TLS para la conexión del cliente cuando el protocolo del canal es TCP/IP.

Visión general

Propósito : La estructura es un parámetro de entrada en la llamada MQCONN.

Si el protocolo del canal del cliente no es « TCP/IP », se ignora la estructura MQSCO.

Conjunto de caracteres y codificación : Los datos en MQSCO deben estar en el conjunto de caracteres proporcionado por el atributo del **CodedCharSetId** gestor de colas y la codificación del gestor de colas local proporcionada por ENNAT.

- “Campos” en la [página 1234](#)
- “Valores iniciales” en la [página 1238](#)
- “Declaración RPG” en la [página 1239](#)

Campos

La estructura MQSCO contiene los siguientes campos, que se describen en **orden alfabético** :

SCAIC (entero con signo de 10 dígitos)

Este es el número de registros de información de autenticación (MQAIR) a los que se refieren los campos *SCAIPSCAIO* o. Para obtener más información, consulte “MQAIR (Registro de información de autenticación) en IBM i” en la [página 1032](#). El valor debe ser mayor o igual que cero. Si el valor no es válido, la llamada falla con el código de motivo RC2383.

Este es un campo de entrada. El valor inicial de este campo es 0.

SCAIO (entero con signo de 10 dígitos)

Este es el desplazamiento en bytes del primer registro de información de autenticación desde el inicio de la estructura MQSCO. El desplazamiento puede ser positivo o negativo. El campo se ignora si *SCAIC* es cero.

Puede utilizar o *SCAIPSCAIO* para especificar los registros MQAIR, pero no ambos; consulte la descripción del *SCAIP* campo para obtener más detalles.

Este es un campo de entrada. El valor inicial de este campo es 0.

SCAIP (entero con signo de 10 dígitos)

Esta es la dirección del primer registro de información de autenticación. El campo se ignora si *SCAIC* es cero.

Puede proporcionar la matriz de registros MQAIR de dos maneras:

- Al utilizar el campo puntero *SCAIP*

En este caso, la aplicación puede declarar una matriz de registros MQAIR separada de la estructura MQSCO y establecer *SCAIP* la dirección de la matriz.

Considere utilizar *SCAIP* para lenguajes de programación que admitan el tipo de datos puntero de forma que sea portátil a diferentes entornos (por ejemplo, el lenguaje de programación C).

- Al utilizar el campo de compensación *SCAIO*

En este caso, la aplicación debe declarar una estructura compuesta que contenga un MQSCO seguido de la matriz de registros MQAIR, y establecer *SCAIO* en la compensación del primer registro de la matriz desde el inicio de la estructura MQSCO. Asegúrese de que este valor sea correcto y tenga un valor que pueda acomodarse dentro de un MQLONG (el lenguaje de programación más restrictivo es COBOL, para el cual el rango válido es de -999 999 999 a +999 999 999).

Considere utilizar *SCAIO* para lenguajes de programación que no admiten el tipo de datos puntero, o que implementan el tipo de datos puntero de una manera que no es portable a diferentes entornos (por ejemplo, el lenguaje de programación COBOL).

Independientemente de la técnica que elijas, solo se puede utilizar *SCAIO* una de *SCAIP* y; la llamada falla con el código de motivo RC2384 si ambas son distintas de cero.

Este es un campo de entrada. El valor inicial de este campo es el puntero nulo en aquellos lenguajes de programación que admiten punteros y, en los demás casos, una cadena de bytes totalmente nulos.

Nota: En plataformas donde el lenguaje de programación no admite el tipo de datos puntero, este campo se declara como una cadena de bytes de la longitud adecuada.

SCCERLBL (entero con signo de 10 dígitos)

Este campo proporciona detalles sobre la etiqueta del certificado que se está utilizando.

IBM MQ Inicializa el valor del campo SCCERLBL como en blanco. Introduzca el valor requerido o acepte el valor predeterminado.

ibmwebspheremquser_id es un valor válido para este campo en todas las versiones del producto, y para las versiones de MQSCO anteriores a 5.0 es el único valor válido. Por lo tanto, el valor de este campo se interpreta en tiempo de ejecución y se modifica si es necesario. Si se especifica una versión MQSCO inferior a 5.0 o se acepta el valor predeterminado en blanco para el campo SCCERLBL, el sistema utiliza el valor *ibmwebspheremquser_id*.

Este es un campo de entrada.

SCCERTVPOL (entero con signo de 10 dígitos)

Este campo especifica qué tipo de política de validación de certificados se utiliza. El campo se puede configurar con uno de los siguientes valores:

MQ_CERT_VAL_POLICY_ANY

Aplica cada una de las políticas de validación de certificados compatibles con la biblioteca de sockets seguros. Acepte la cadena de certificados si alguna de las políticas considera que la cadena de certificados es válida.

MQ_CERT_VAL_POLICY_RFC5280

Aplique únicamente la política de validación de certificados compatible con RFC5280. Este valor proporciona una validación más estricta que el valor ANY, pero rechaza algunos certificados digitales más antiguos.

El valor inicial de este campo es MQ_CERT_VAL_POLICY_ANY

SCCH (entero con signo de 10 dígitos)

Este campo proporciona detalles de configuración para el hardware criptográfico conectado al sistema cliente.

Establezca el campo en una cadena con el siguiente formato, o déjelo en blanco o nulo:

```
GSK_PKCS11=the PKCS #11 driver path and file name;the PKCS #11 token label;the PKCS #11 token password;symmetric cipher setting>;
```

Para utilizar hardware criptográfico que cumpla con la interfaz del controlador de hardware criptográfico (PKCS11), por ejemplo, el IBM 4960 o IBM el 4963, especifique la ruta del controlador del hardware criptográfico (PKCS11), la etiqueta del token (PKCS11) y las cadenas de contraseña del token (PKCS11), cada una de ellas terminada con un punto y coma.

La ruta del controlador PKCS #11 es una ruta absoluta a la biblioteca compartida que proporciona soporte para la tarjeta PKCS #11. El nombre del archivo del controlador PKCS #11 es el nombre de la biblioteca compartida. Un ejemplo del valor requerido para la ruta PKCS #11 y el nombre del archivo es:

```
/usr/lib/pkcs11/PKCS11_API.so
```

La etiqueta del token PKCS #11 debe estar escrita íntegramente en minúsculas. Si ha configurado su hardware con una etiqueta de token en mayúsculas y minúsculas o solo en mayúsculas, vuelva a configurarlo con esta etiqueta en minúsculas.

Si no se requiere ninguna configuración de hardware criptográfico, deje el campo en blanco o nulo.

Si el valor es más corto que la longitud del campo, termine el valor con un carácter nulo o rellénelo con espacios en blanco hasta alcanzar la longitud del campo. Si el valor no es válido o provoca un error al utilizarse para configurar el hardware criptográfico, la llamada falla con el código de motivo RC2382.

Este es un campo de entrada. La longitud de este campo viene dada por LNSSCH. El valor inicial de este campo son caracteres en blanco.

SCEPSUITEB (entero con signo de 10 dígitos)

Este campo especifica si se utiliza criptografía compatible con Suite B y qué nivel de seguridad se emplea. El valor puede ser uno o varios de los siguientes:

- SCEPSUITEB0
No se utiliza criptografía compatible con Suite B.
- SCEPSUITEB1
Se utiliza seguridad Suite B de 128 bits.
- SCEPSUITEB2
Se utiliza seguridad Suite B de 192 bits.

Nota: El uso de SCEPSUITEB0 con cualquier otro valor en este campo no es válido.

SCFR (entero con signo de 10 dígitos)

IBM MQ Se puede configurar con hardware criptográfico para que los módulos criptográficos utilizados sean los proporcionados por el producto de hardware; estos pueden FIPS certificarse a un nivel determinado en función del producto de hardware criptográfico que se utilice.

Utilice este campo para especificar que solo se utilicen FIPS algoritmos certificados si la criptografía se proporciona en el software IBM MQ suministrado.

Cuando IBM MQ se instala, también se instala una implementación de la criptografía TLS, que proporciona algunos módulos FIPS certificados.

Los valores pueden ser:

MQSSL_FIPS_NO

Éste es el valor predeterminado. Cuando se establece en este valor:

- Se puede utilizar cualquier CipherSpec compatible con una plataforma concreta.
- Si se ejecuta sin utilizar hardware criptográfico, los siguientes CipherSpecs se ejecutan utilizando criptografía FIPS 140-2 certificada en las IBM MQ plataformas:
 - TLS_RSA_WITH_3DES_EDE_CBC_SHA
 - TLS_RSA_WITH_AES_128_CBC_SHA
 - TLS_RSA_WITH_AES_256_CBC_SHA

MQSSL_FIPS_YES

Cuando se establece este valor, a menos que se utilice hardware criptográfico para realizar la criptografía, se puede estar seguro de que

- Solo se pueden utilizar FIPS algoritmos criptográficos certificados en el CipherSpec que se aplica a esta conexión de cliente.
- Las conexiones entrantes y salientes del canal de cifrado de datos (TLS) solo se realizan correctamente si se utiliza una de las siguientes especificaciones de cifrado:
 - TLS_RSA_WITH_3DES_EDE_CBC_SHA
 - TLS_RSA_WITH_AES_128_CBC_SHA
 - TLS_RSA_WITH_AES_256_CBC_SHA

Notas:

1.  La CipherSpec TLS_RSA_WITH_3DES_EDE_CBC_SHA está en desuso.
2. Siempre que sea posible, si se configura « CipherSpecs » (Solo FIPS), el cliente de MQI rechazará las conexiones que especifiquen una dirección de correo electrónico CipherSpec withRC2393 que no sea FIPS. IBM MQ no garantiza que rechace todas las conexiones de este tipo y es su responsabilidad determinar si su IBM MQ configuración es FIPS conforme.

SCKEYPWL (entero con signo de 10 dígitos)

Esta es la longitud de la contraseña del repositorio de claves TLS.

La longitud máxima de la contraseña del repositorio de claves es de 128 caracteres. Si la frase de contraseña del repositorio de claves supera la longitud máxima permitida, la conexión falla con el error « RC2381 ».

Este es un campo de entrada. El valor inicial de este campo es 0.

SCKEYPWO (entero con signo de 10 dígitos)

Este es el desplazamiento en bytes de la frase de contraseña del repositorio de claves TLS. El desplazamiento puede ser positivo o negativo.

Puede utilizar o SCKEYPWPSCKEYPWO para especificar la frase de contraseña del repositorio de claves, pero no ambos. Para obtener más información, consulte la descripción del SCKEYPWP campo.

Este es un campo de entrada. El valor inicial de este campo es 0.

SCKEYPWP (puntero)

Esta es la dirección de la contraseña del repositorio de claves TLS.

Este es un campo de entrada. El valor inicial de este campo es el puntero nulo.

La contraseña del repositorio de claves se puede especificar como una cadena de texto sin formato o como una contraseña que se ha cifrado con la **runmqicred** utilidad.

La contraseña del repositorio de claves especificada en este campo anula cualquier contraseña del repositorio de claves especificada mediante la variable de entorno *MQKEYRPWD* o la *SSLKeyRepositoryPassword* propiedad en la sección SSL del archivo de configuración del cliente.

Puede utilizar o SCKEYPWPSCKEYPWO para especificar la frase de contraseña del repositorio de claves, pero no ambos.

SCKR (entero con signo de 10 dígitos)

Este campo especifica la ubicación del archivo de base de datos de claves en el que se almacenan las claves y los certificados. Si no se especifica la extensión del .kdb archivo, se añade automáticamente la extensión.

Cada archivo de base de datos clave puede tener un *archivo de almacenamiento de contraseñas* asociado. Contiene contraseñas cifradas que se utilizan para permitir el acceso programático a la base de datos de claves. El archivo de almacenamiento de contraseñas debe residir en el mismo directorio y tener el mismo nombre base que la base de datos de claves, y debe terminar con la extensión .sth.

Por ejemplo, si el archivo de base de datos de claves es /xxx/yyy/key.kdb, entonces el archivo de almacenamiento de contraseñas debe ser /xxx/yyy/key.sth, donde xxx y yyy representan nombres de directorios.

La contraseña de la base de datos clave también se puede especificar utilizando los campos *SCKEYPWP* *SCKEYPWO* o.

Si el valor es más corto que la longitud del campo, termine el valor con un carácter nulo o rellénelo con espacios en blanco hasta alcanzar la longitud del campo. El valor no se comprueba; si se produce un error al acceder al repositorio de claves, la llamada falla con el código de motivo RC2381.

Para ejecutar una conexión TLS desde un IBM MQ MQI client, establezca *SCKR* en un nombre de archivo de base de datos de claves válido.

Este es un campo de entrada. La longitud de este campo viene dada por LNSSKR. El valor inicial de este campo es un carácter en blanco.

SCSID (entero con signo de 10 dígitos)

Este es el identificador de estructura; el valor debe ser:

SCSIDV

Identificador para la estructura de opciones de configuración de TLS.

Este campo siempre es un campo de entrada. El valor inicial de este campo es SCSIDV.

SCVER (entero con signo de 10 dígitos)

Este es el número de versión de la estructura; el valor debe ser:

SCVER1

Version-1 TLS Estructura de opciones de configuración.

SCVER2

Version-2 TLS Estructura de opciones de configuración.

SCVER3

Version-3 TLS Estructura de opciones de configuración.

SCVER4

Version-4 TLS Estructura de opciones de configuración.

SCVER5

Version-5 TLS Estructura de opciones de configuración.

SCVER6

Version-6 TLS Estructura de opciones de configuración.

La siguiente constante especifica el número de versión de la versión actual:

SCVERC

Versión actual de la estructura de opciones de configuración de TLS.

Este campo siempre es un campo de entrada. El valor inicial de este campo es SCVER1.

Valores iniciales

Tabla 765. Campos en MQSCO		
Nombre de campo	Nombre de la constante	Valor de la constante
SCSID	SCSIDV	'SC0~'
SCVER	SCVER1	1
SCKR	Ninguna	Cadena nula o espacios en blanco
SCCH	Ninguna	Cadena nula o espacios en blanco
SCAIC	Ninguna	0
SCAIO	Ninguna	0
SCAIP	Ninguna	Puntero nulo o bytes nulos
SCKRC	Ninguna	Puntero nulo o bytes nulos
SCFR	Ninguna	Puntero nulo o bytes nulos

Tabla 765. Campos en MQSCO (continuación)

Nombre de campo	Nombre de la constante	Valor de la constante
SCEPSUITEB	Ninguna	Puntero nulo o bytes nulos
SCCERTVPOL	Ninguna	Puntero nulo o bytes nulos
SCCERLBL	Ninguna	Puntero nulo o bytes nulos
SCKEYPWP	Ninguna	Puntero nulo o bytes nulos
SCKEYPWO	Ninguna	0
SCKEYPWL	Ninguna	0
Notas: 1. El símbolo ~ representa un único carácter en blanco. 2. Consulte “Declaración RPG” en la página 1239 para ver las SCEPSUITEB opciones.		

Declaración RPG

```

D*.1....:....2....:....3....:....4....:....5....:....6....:....7..
D* MQSCO Structure
D*
D* Structure identifier
D  SCSID          1      4    INZ('SCO ')
D* Structure version number
D  SCVER          5      8I 0 INZ(1)
D* Location of TLS key repository
D  SCKR           9     264   INZ
D* Cryptographic hardware configuration string
D  SCCH          265     520   INZ
D* Number of MQAIR records present
D  SCAIC         521     524I 0 INZ(0)
D* Offset of first MQAIR record from start of MQSCO structure
D  SCAIO         525     528I 0 INZ(0)
D* Address of first MQAIR record
D  SCAIP         529     544*   INZ(*NULL)
D* Ver:1 **
D* Number of unencrypted bytes sent/received before secret key is
D* reset
D  SCKRC         545     548I 0 INZ(0)
D* Using FIPS certified algorithms
D  SCFR         549     552I 0 INZ(0)
D* Ver:2 **
* Use only Suite B cryptographic algorithms
D  SCEPSUITEB0
D  SCEPSUITEB1     553     556I 0 INZ(1)
D  SCEPSUITEB2     557     560I 0 INZ(0)
D  SCEPSUITEB3     561     564I 0 INZ(0)
D  SCEPSUITEB4     565     568I 0 INZ(0)
D  SCEPSUITEB      10I 0 DIM(4) OVERLAY(SCEPSUITEB0)
D* Ver:3 **
D* Certificate validation policy
D  SCCERTVPOL     569     572I 0 INZ(0)
D* Ver:4 **
    
```

IBM i MQSD (descriptor de suscripción) en IBM i

La estructura MQSD se utiliza para especificar detalles sobre la suscripción que se está realizando.

Visión general

Finalidad

La estructura es un parámetro de entrada/salida en la llamada MQSUB.

Suscripciones gestionadas

Si una aplicación no tiene una necesidad específica de utilizar una cola concreta como destino para las publicaciones que coinciden con su suscripción, puede utilizar la función de suscripción administrada. Si una aplicación opta por utilizar una suscripción gestionada, el gestor de colas informa al suscriptor sobre el destino al que se envían los mensajes publicados, proporcionando un identificador de objeto como salida de la llamada MQSUB. Para obtener más información, consulte [HOBJ](#) (entero con signo de 10 dígitos): entrada/salida.

Cuando se elimina la suscripción, el gestor de colas también se encarga de limpiar los mensajes que no se han recuperado del destino gestionado, en las siguientes situaciones:

- Cuando se elimina la suscripción, mediante el uso de MQCLOSE con CORMSB, y se cierra el Hobj gestionado.
- De forma implícita cuando se pierde la conexión con una aplicación que utiliza una suscripción no duradera (SONDUR)
- Por caducidad, cuando se elimina una suscripción porque ha caducado y se cierra el Hobj gestionado.

Debe utilizar suscripciones gestionadas con suscripciones no duraderas, para que se pueda realizar la limpieza y para que los mensajes de las suscripciones no duraderas cerradas no ocupen espacio en el gestor de colas. Las suscripciones duraderas también pueden utilizar destinos gestionados.

Conjunto de caracteres y codificación

Los datos en MQSD deben estar en el juego de caracteres proporcionado por el atributo del **CodedCharSetId** gestor de colas y la codificación del gestor de colas local proporcionada por ENNAT. Sin embargo, si la aplicación se ejecuta como un IBM MQ cliente, la estructura debe estar en el juego de caracteres y la codificación del cliente.

- “Campos” en la [página 1240](#)
- “Valores iniciales” en la [página 1253](#)
- “Declaración RPG” en la [página 1253](#)

Campos

La estructura MQSD contiene los siguientes campos, que se describen en orden alfabético:

SDAID (cadena de caracteres de 32 bytes)

Este valor se encuentra en el *MDAID* campo del descriptor de mensajes (MQMD) de todos los mensajes de publicación que coinciden con esta suscripción. *SDAID* forma parte del contexto de identidad del mensaje. Para obtener más información sobre el contexto del mensaje, consulte [Contexto de mensaje](#).

Para obtener más información, *MDAID* consulte [MDAID](#).

Si no se especifica la opción SOSETI, el *MDAID* que se establece en cada mensaje publicado para esta suscripción es en blanco, como información de contexto predeterminada.

Si se especifica la opción SOSETI, el *SDAID* lo genera el usuario y este campo es un campo de entrada que contiene el *MDAID* que se establecerá en cada publicación para esta suscripción.

La longitud de este campo viene determinada por LNAIDD. El valor inicial de este campo es 32 caracteres en blanco.

Si se modifica una suscripción existente utilizando la opción SOALT, se puede cambiar la *SDAID* de cualquier mensaje de publicación futuro.

Al regresar de una llamada MQSUB utilizando SORES, este campo se establece en el actual *MDAID* que se utiliza para la suscripción.

SDACC (cadena de caracteres de 32 bytes)

Este valor se encuentra en el *MDACC* campo del descriptor de mensajes (MQMD) de todos los mensajes de publicación que coinciden con esta suscripción. *MDACC* forma parte del contexto de identidad del mensaje. Para obtener más información sobre el contexto del mensaje, consulte [Contexto de mensaje](#).

Para obtener más información, *MDACC* consulte [MDACC](#).

Puede utilizar el siguiente valor especial para el *SDACC* campo:

ACNONE

No se especifica ningún token contable.

El valor es un cero binario para la longitud del campo.

Si no se especifica la opción *SOSETI*, el token de contabilidad es generado por el gestor de colas como información de contexto predeterminada y este campo es un campo de salida que contiene el *MDACC* que se establece en cada mensaje publicado para esta suscripción.

Si se especifica la opción *SOSETI*, el token contable lo genera el usuario y este campo es un campo de entrada que contiene el *MDACC* que se debe establecer en cada publicación para esta suscripción.

La longitud de este campo viene determinada por *LNACCT*. El valor inicial de este campo es *ACNONE*.

Si se modifica una suscripción existente utilizando la opción *SOALT*, se puede cambiar el valor de *MDACC* en cualquier mensaje de publicación futuro.

Al regresar de una llamada *MQSUB* utilizando *SORES*, este campo se establece en el actual *MDACC* que se utiliza para la suscripción.

SDASI (cadena de bits de 40 bytes)

Se trata de un identificador de seguridad que se transmite junto con el *SDAU* al servicio de autorización para permitir que se realicen las comprobaciones de autorización adecuadas.

SDASI Se utiliza solo si se especifica *SOALTU* y el *SDAU* campo no está completamente en blanco hasta el primer carácter nulo o el final del campo.

Al regresar de una llamada *MQSUB* utilizando *SORES*, este campo permanece sin cambios.

Consulte la descripción de [ODASI](#) en el tipo de datos *MQOD* para obtener más información.

SDAU (cadena de caracteres de 12 bytes)

Si se especifica *SOALTU*, este campo contiene un identificador de usuario alternativo que se utiliza para comprobar la autorización de la suscripción y para la salida a la cola de destino (especificada en el **Hobj** parámetro de la llamada *MQSUB*), en lugar del identificador de usuario con el que se está ejecutando actualmente la aplicación.

Si tiene éxito, el identificador de usuario especificado en este campo se registra como el identificador de usuario propietario de la suscripción en lugar del identificador de usuario con el que se está ejecutando actualmente la aplicación.

Si se especifica *SOALTU* y este campo está completamente en blanco hasta el primer carácter nulo o el final del campo, la suscripción solo podrá realizarse correctamente si no se necesita autorización de usuario para suscribirse a este tema con las opciones especificadas o la cola de destino para la salida.

Si no se especifica *SOALTU*, este campo se ignora.

Al regresar de una llamada *MQSUB* utilizando *SORES*, este campo permanece sin cambios.

Este es un campo de entrada. La longitud de este campo viene dada por *LNUID*. El valor inicial de este campo es 12 caracteres en blanco.

SDCID (cadena de bits de 24 bytes)

Todas las publicaciones enviadas para coincidir con esta suscripción contienen este identificador de correlación en el descriptor del mensaje. Si varias suscripciones utilizan la misma cola para obtener sus publicaciones, el uso de MQGET por ID de correlación sólo permite obtener publicaciones para una suscripción específica. Este identificador de correlación puede ser generado por el gestor de colas o por el usuario.

Si no se especifica la opción SOSCID, el identificador de correlación lo genera el gestor de colas y este campo es un campo de salida que contiene el identificador de correlación establecido en cada mensaje publicado para esta suscripción.

Si se especifica la opción SOSCID, el identificador de correlación lo genera el usuario y este campo es un campo de entrada que contiene el identificador de correlación que se establecerá en cada publicación para esta suscripción. En este caso, si el campo contiene CINONE, el identificador de correlación que se establece en cada mensaje publicado para esta suscripción es el identificador de correlación que fue creado por la inserción original del mensaje.

Si se especifica la opción SOGRP y el identificador de correlación especificado es el mismo que el de una suscripción agrupada existente que utiliza la misma cola y una cadena de tema superpuesta, solo se proporciona una copia de la publicación a la suscripción más significativa del grupo.

La longitud de este campo viene dada por LNCID. El valor inicial de este campo es CINONE.

Si se modifica una suscripción existente utilizando la opción SOALT y este campo es un campo de entrada, se puede cambiar el ID de correlación de la suscripción, a menos que la suscripción se haya creado utilizando la opción SOGRP.

Al regresar de una llamada MQSUB utilizando SORES, este campo se establece en el ID de correlación actual para la suscripción.

SDEXP (entero con signo de 10 dígitos)

Es el tiempo expresado en décimas de segundo tras el cual caduca la suscripción. Una vez transcurrido este intervalo, no habrá más publicaciones que coincidan con esta suscripción. Esto también se utiliza como valor en el *MDEXP* campo del MQMD de las publicaciones enviadas a este suscriptor.

Se reconoce el siguiente valor especial:

EIULIM

La suscripción tiene una duración ilimitada.

Si se modifica una suscripción existente utilizando la opción SOALT, se puede cambiar la fecha de vencimiento de la suscripción.

Al regresar de una llamada MQSUB utilizando la opción SORES, este campo se establece en la fecha de vencimiento original de la suscripción y no en el tiempo de vencimiento restante.

SDON (cadena de caracteres de 48 bytes)

Este es el nombre del objeto de tema tal y como se define en el gestor de colas local.

El nombre puede contener los siguientes caracteres:

- Caracteres alfabéticos mayúsculos (de la A a la Z)
- Letras minúsculas (de la a a la z)
- Dígitos numéricos (del 0 al 9)
- Punto (.), barra inclinada (/), guión bajo (_), porcentaje (%)

El nombre no debe contener espacios en blanco al principio ni en el interior, pero puede contener espacios en blanco al final. Utilice un carácter nulo para indicar el final de los datos significativos en el nombre; el carácter nulo y cualquier carácter que le siga se tratan como espacios en blanco. Las restricciones son:

- En los sistemas que utilizan EBCDIC Katakana, no se pueden utilizar caracteres en minúscula.

- Los nombres que contengan caracteres en minúscula, barra inclinada o porcentaje deben escribirse entre comillas cuando se especifiquen en los comandos. Estas comillas no deben especificarse para nombres que aparecen como campos en estructuras o como parámetros en llamadas.

El *SDON* se utiliza para formar el nombre completo del tema.

El nombre completo del tema se puede construir a partir de dos campos diferentes: *SDON* y *SDOS*. Para obtener más información sobre cómo se utilizan estos dos campos, consulte [Combinación de cadenas de temas](#).

Al regresar de una llamada MQSUB utilizando la opción SORES, este campo permanece sin cambios.

La longitud de este campo viene dada por LNTOPN. El valor inicial de este campo es 48 caracteres en blanco.

Si se modifica una suscripción existente utilizando la opción SDALT, no se puede cambiar el nombre del objeto de tema suscrito. Este campo y *SDOS* pueden omitirse. Si se proporcionan, deben resolver al mismo nombre completo del tema o la llamada fallará con RC2510 .

SDOPT (entero con signo de 10 dígitos)

Debe especificar al menos una de las siguientes opciones:

- SOALT
- Llagas
- SOCRT

Los valores se pueden sumar. No añadas la misma constante más de una vez. La tabla muestra cómo se pueden combinar estas opciones: Las combinaciones que no son válidas están señaladas; cualquier otra combinación es válida.

Opciones de acceso o creación

Las opciones de acceso y creación controlan si se crea una suscripción o si se devuelve o modifica una suscripción existente. Debe especificar al menos una de estas opciones. La tabla muestra combinaciones válidas de opciones de acceso o creación.

<i>Tabla 766. Combinaciones válidas de opciones de acceso y creación</i>	
Combinación de opciones	Notas
SOCRT	Crea una suscripción si no existe ninguna; falla si la suscripción existe.
Llagas	Reanuda una suscripción existente; si no existe ninguna suscripción, se produce un error.
SOCRT + SORES	Crea una suscripción si no existe ninguna y reanuda una coincidente, si existe. Combinación útil si se utiliza en una aplicación que puede ejecutarse varias veces.
SORES + SOALT (véase la nota)	Reanuda una suscripción existente, modificando cualquier campo para que coincida con los especificados en el MQSD, falla si no existe ninguna suscripción.
SOCRT + SOALT (véase la nota)	Crea una suscripción si no existe ninguna y reanuda una que coincida, si existe, modificando los campos para que coincidan con los especificados en el MQSD. Combinación útil si se utiliza en una aplicación que desea asegurarse de que su suscripción se encuentra en un estado determinado antes de continuar.

Nota:

Las opciones que especifican SOALT también pueden especificar SORES, pero esta combinación no tiene ningún efecto adicional al especificar solo SOALT. SOALT implica SORES, porque llamar a MQSUB para modificar una suscripción implica que las suscripciones también se reanudan. Sin embargo, lo contrario no es cierto: reanudar una suscripción no implica que se vaya a modificar.

SOCRT

Crear una suscripción para el tema especificado. Si existe una suscripción que utiliza el mismo *SDSN*, la llamada falla con RC2432. Este fallo se puede evitar combinando la opción SOCRT con SORES. No siempre *SDSN* es necesario. Para obtener más detalles, consulte la descripción de ese campo.

La combinación de SOCRT con SORES comprueba primero si existe una suscripción para el especificado *SDSN*, si la hay, devuelve un identificador a esa suscripción preexistente; pero si no existe ninguna suscripción, se crea una nueva utilizando todos los campos proporcionados en el MQSD.

SOCRT también se puede combinar con SOALT para obtener un efecto similar (consulte los detalles sobre SOALT más adelante en este tema).

Llagas

Devuelve un identificador a una suscripción preexistente que coincide con las especificadas por *SDSN*. No se realizan cambios en los atributos de suscripción coincidentes, y estos se devuelven en la salida en la estructura MQSD. La mayor parte del contenido del MQSD no se utiliza: los campos que se utilizan son *SDSID*, *SDVER*, *SDOPT*, *SDAID* y *SDASI*, y *SDSN*.

La llamada falla con el código de motivo RC2428 si no existe una suscripción que coincida con el nombre completo de la suscripción. Este fallo se puede evitar combinando la opción SOCRT con SORES. Para obtener más información sobre SOCRT, consulte [SOCRT](#).

El ID de usuario de la suscripción es el ID de usuario que creó la suscripción o, si posteriormente ha sido modificado por otro ID de usuario, es el ID de usuario de la última modificación realizada con éxito. Si se utiliza *SDAID* un y se permite el uso de ID de usuario alternativos para ese usuario, *SDAID* se registra como el ID de usuario que creó la suscripción en lugar del ID de usuario con el que se realizó la suscripción.

El ID de usuario que creó la suscripción se registra como *SDAU* si se utiliza ese campo, y se permite el uso de ID de usuario alternativos para ese usuario.

Si existe una suscripción coincidente que se creó sin la opción SOAUID y el ID de usuario de la suscripción es diferente al de la aplicación que solicita un identificador para la suscripción, la llamada falla con el código de motivo RC2434.

Si existe una suscripción coincidente y actualmente está siendo utilizada por otra aplicación, la llamada falla con el código de motivo RC2429. Si actualmente está en uso por la misma conexión, la llamada no falla y se devuelve un identificador de la suscripción.

Si la suscripción indicada en SubName no es una suscripción válida para reanudar o modificar desde una aplicación, la llamada falla con RC2523.

SORES está implícito en SOALT, por lo que no es necesario combinarlo con esa opción; sin embargo, no es un error si se combinan ambas opciones.

SOALT

Devuelve un identificador a una suscripción preexistente con el nombre completo de la suscripción que coincide con los especificados en *SDSN*. Cualquier atributo de la suscripción que difiera de los especificados en el MQSD se modifica en la suscripción, a menos que no se permita la modificación de ese atributo. Los detalles se indican en la descripción de cada atributo y se resumen en la siguiente tabla. Si intenta modificar un atributo que no se puede cambiar, la llamada fallará con el código de motivo que se muestra en la siguiente tabla.

La llamada falla con el código de motivo RC2428 si no existe una suscripción que coincida con el nombre completo de la suscripción. Este fallo se puede evitar combinando la opción SOCRT con SOALT.

Al combinar SOCRT con SOALT, primero se comprueba si existe una suscripción para el nombre de suscripción completo especificado y, si existe, se devuelve un identificador a esa suscripción preexistente con las modificaciones realizadas anteriormente detalladas; pero si no existe ninguna suscripción, se crea una nueva utilizando todos los campos proporcionados en el MQSD.

El ID de usuario de la suscripción es el ID de usuario que creó la suscripción o, si posteriormente ha sido modificado por otro ID de usuario, es el ID de usuario de la última modificación realizada con éxito. Si SDAU se utiliza (y se permite el uso de ID de usuario alternativos para ese usuario), entonces el ID de usuario alternativo se registra como el ID de usuario que creó la suscripción, en lugar del ID de usuario con el que se realizó la suscripción.

Si existe una suscripción coincidente que se creó sin la opción SOAUID y el ID de usuario de la suscripción es diferente al de la aplicación que solicita un identificador para la suscripción, la llamada falla con el código de motivo RC2434 .

Si existe una suscripción coincidente y actualmente está siendo utilizada por otra aplicación, la llamada falla con RC2429 . Si actualmente está siendo utilizado por la misma conexión, la llamada no falla y se devuelve un identificador de la suscripción.

Si la suscripción indicada en SubName no es una suscripción válida para reanudar o modificar desde una aplicación, la llamada falla con RC2523 .

Las siguientes tablas muestran los atributos de suscripción que pueden ser modificados por SOALT.

<i>Tabla 767. Atributos en MQSD y MQSUB que se pueden modificar</i>			
Descriptor de tipo de datos o llamada a función	Nombre de campo	¿Se puede modificar este atributo utilizando SOALT?	Código de razón
MQSD	Opciones de durabilidad	No	RC2509
MQSD	Opciones de destino	Sí	Ninguna
MQSD	Opciones de registro	Sí (véase la nota 1)	RC2515 si intentas modificar SOGRP
MQSD	Opciones de publicación	Sí (véase la nota 2)	Ninguna
MQSD	Opciones comodín	No	RC2510
MQSD	Otras opciones	No (véase la nota 3)	Ninguna
MQSD	ObjectName	No	RC2510
MQSD	SDAU	No (véase la nota 4)	Ninguna
MQSD	SDASI	No (véase la nota 4)	Ninguna
MQSD	SDEXP	Sí	Ninguna
MQSD	SDOS	No	RC2510
MQSD	Red de Soluciones para el Desarrollo Sostenible (SDSN)	No (véase la nota 5)	Ninguna
MQSD	SDSUD	Sí	Ninguna
MQSD	SDCID	Sí (véase la nota 6)	RC2515 cuando se trata de una suscripción agrupada
MQSD	SDPRI	Sí	Ninguna
MQSD	SDACC	Sí	Ninguna

Tabla 767. Atributos en MQSD y MQSUB que se pueden modificar (continuación)			
Descriptor de tipo de datos o llamada a función	Nombre de campo	¿Se puede modificar este atributo utilizando SOALT?	Código de razón
MQSD	SDAID	Sí	Ninguna
MQSD	SDSL	No	RC2512
MQSUB	Hobj	Sí (véase la nota 6)	RC2515 cuando se trata de una suscripción agrupada

Notas:

1. El SOGRP no se puede modificar.
2. SONEWP no se puede modificar porque no forma parte de la suscripción
3. Estas opciones no forman parte de la suscripción
4. Este atributo no forma parte de la suscripción
5. Este atributo es la identidad de la suscripción que se está modificando
6. Modificable excepto cuando forma parte de un subgrupo (SOGRP)

Opciones de durabilidad : Las siguientes opciones controlan la durabilidad de la suscripción. Solo puede especificar una de estas opciones. Si está modificando una suscripción existente utilizando la opción SOALT, no puede cambiar la duración de la suscripción. Al regresar de una llamada MQSUB utilizando SORES, se establece la opción de durabilidad adecuada.

SODUR

Solicite que la suscripción a este tema permanezca hasta que se elimine explícitamente mediante MQCLOSE con la opción CORMSB. Si esta suscripción no se elimina explícitamente, permanecerá incluso después de que se cierre la aplicación que se conecta al gestor de colas.

Si se solicita una suscripción duradera a un tema que está definido como no permitir suscripciones duraderas, la llamada falla con RC2436 .

SONDUR

Solicite que se elimine la suscripción a este tema cuando se cierre la conexión de la aplicación al gestor de colas, si aún no se ha eliminado explícitamente. SONDUR es lo contrario de la opción SODUR y está definido para facilitar la documentación del programa. Es el valor predeterminado si no se especifica ninguno.

Opciones de destino : Las siguientes opciones controlan el destino al que se envían las publicaciones de un tema al que se ha suscrito. Si se modifica una suscripción existente utilizando la opción SOALT, se puede cambiar el destino utilizado para las publicaciones de la suscripción. Al regresar de una llamada MQSUB utilizando SORES, esta opción se establece si procede.

SOMAN

Solicite que el destino al que se envían las publicaciones sea gestionado por el gestor de colas.

El identificador de objeto devuelto en HOBJ representa una cola gestionada por el gestor de colas y se utiliza con las llamadas MQGET, MQCB, MQINQ o MQCLOSE posteriores.

No se puede proporcionar un identificador de objeto devuelto de una llamada MQSUB anterior en el Hobj parámetro cuando no se especifica SOMAN.

Opciones de registro : Las siguientes opciones controlan los detalles del registro que se realiza en el administrador de colas para esta suscripción. Si se modifica una suscripción existente utilizando la opción SOALT, estas opciones de registro pueden cambiarse. Al regresar de una llamada MQSUB utilizando SORES, se configuran las opciones de registro adecuadas.

SOGRP

Esta suscripción se agrupa con otras suscripciones de la misma *SDSL* que utilizan la misma cola y especifican el mismo ID de correlación, de modo que cualquier publicación en temas que provoque que se envíe más de un mensaje de publicación al grupo de suscripciones, debido a un conjunto superpuesto de cadenas de temas utilizadas, solo provoca que se envíe un mensaje a la cola. Si no se utiliza esta opción, cada suscripción única (identificada por *SDSN*) que coincida recibirá una copia de la publicación, lo que podría significar que se colocarán más de una copia de la publicación en la cola compartida por varias suscripciones.

Solo la suscripción más importante del grupo recibe una copia de la publicación. La suscripción más significativa se basa en el nombre completo del tema hasta el punto en que se encuentra un comodín. Si se utiliza una combinación de esquemas de comodines dentro del grupo, solo es importante la posición del comodín. Se recomienda no combinar diferentes esquemas de comodines dentro de un grupo de suscripciones que comparten la misma cola.

Al crear una nueva suscripción agrupada, esta debe seguir teniendo un nombre único *SDSN*, pero si coincide con el nombre completo del tema de una suscripción existente en el grupo, la llamada falla con RC2514 .

Si la suscripción más significativa del grupo también especifica *SONOLC* y se trata de una publicación de la misma aplicación, entonces no se entrega ninguna publicación a la cola.

Al modificar una suscripción realizada con esta opción, los campos que implican la agrupación, *Hobj* en la llamada *MQSUB* (que representa la cola y el nombre del gestor de colas), y el *SDCID* no se pueden cambiar. Si se intenta modificarlos, la llamada falla con RC2515 .

Esta opción debe combinarse con *SOSCID* con un *SDCID* que no esté establecido en *CINONE*, y no puede combinarse con *SOMAN*.

SOAUID

Cuando se especifica *SOAUID*, la identidad del suscriptor no se limita a un único ID de usuario. Esto permite que cualquier usuario modifique o reanude la suscripción cuando disponga de la autoridad adecuada. Solo un usuario puede tener la suscripción en un momento dado. Si se intenta reanudar el uso de una suscripción que ya está siendo utilizada por otra aplicación, la llamada fallará con RC2429 .

Para añadir esta opción a una suscripción existente, la llamada *MQSUB*, utilizando *SOALT*, debe provenir del mismo ID de usuario que la suscripción original.

Si una llamada *MQSUB* hace referencia a una suscripción existente con *SOAUID* establecido y el ID de usuario difiere del de la suscripción original, la llamada solo se realizará correctamente si el nuevo ID de usuario tiene autoridad para suscribirse al tema. Una vez completado con éxito, las futuras publicaciones para este suscriptor se colocan en la cola del suscriptor con el nuevo ID de usuario establecido en el mensaje de publicación.

No especifique tanto *SOAUID* como *SOFUID*. Si no se especifica ninguno, el valor predeterminado es *SOFUID*.

SOFUID

Cuando se especifica *SOFUID*, solo el último ID de usuario que modificó la suscripción puede modificarla o reanudarla. Si la suscripción no se ha modificado, es el ID de usuario que creó la suscripción.

Si un verbo *MQSUB* hace referencia a una suscripción existente con *SOAUID* establecido y modifica la suscripción utilizando *SOALT* para utilizar *SOFUID*, el ID de usuario de la suscripción queda fijado en este nuevo ID de usuario. La llamada sólo es satisfactoria si el nuevo ID de usuario tiene autoridad para suscribirse al tema.

Si un ID de usuario distinto al registrado como propietario de una suscripción intenta reanudar o modificar una suscripción *SOFUID*, la llamada falla con RC2434 . El ID de usuario propietario de una suscripción se puede ver utilizando el **DISPLAY SBSTATUS** comando.

No especifique tanto SOAUID como SOFUID. Si no se especifica ninguno, el valor predeterminado es SOFUID.

Opciones de publicación : Las siguientes opciones controlan la forma en que se envían las publicaciones a este suscriptor. Si se modifica una suscripción existente utilizando la opción SOALT, se pueden cambiar estas opciones de publicación.

SONOLC

Indica al agente que la aplicación no desea ver ninguna de sus propias publicaciones. Se considera que las publicaciones proceden de la misma aplicación si los identificadores de conexión son los mismos. Al regresar de una llamada MQSUB utilizando SORES, esta opción se establece si procede.

SONEWP

No se enviarán publicaciones actualmente retenidas, cuando se cree esta suscripción, solo nuevas publicaciones. Esta opción solo se aplica cuando se especifica SOCRE. Cualquier cambio posterior en una suscripción no altera el flujo de publicaciones, por lo que cualquier publicación que se haya conservado sobre un tema ya se ha enviado al suscriptor como nueva publicación. Si se especifica esta opción sin SOCRE, la llamada fallará con RC2046 . Al regresar de una llamada MQSUB utilizando SORES , esta opción no se establece aunque la suscripción se haya creado utilizando esta opción.

Si no se utiliza esta opción, los mensajes retenidos anteriormente se envían a la cola de destino proporcionada. Si esta acción falla debido a un error, ya sea RC2525 o RC2526 , la creación de la suscripción falla.

Esta opción no es válida en combinación con SOPUBR.

SOPUBR

Al seleccionar esta opción, se indica que el suscriptor solicita información específicamente cuando es necesario. El gestor de colas no envía mensajes no solicitados al suscriptor. La publicación retenida (o posiblemente varias publicaciones si se especifica un comodín en el tema) se envía al suscriptor cada vez que se realiza una llamada MQSUBRQ utilizando el identificador Hsub de una llamada MQSUB anterior. No se envían publicaciones como resultado de la llamada MQSUB utilizando esta opción. Al regresar de una llamada MQSUB utilizando SORES, esta opción se establece si procede.

Esta opción no es válida en combinación con SONEWP.

Opciones de comodines : Las siguientes opciones controlan cómo se interpretan los comodines en la cadena proporcionada en el SDOS campo del MQSD. Solo puede especificar una de estas opciones. Si se modifica una suscripción existente utilizando la opción SOALT, estas opciones comodín no se pueden cambiar. Al regresar de una llamada MQSUB utilizando SORES, se establece la opción comodín adecuada.

SOWCHR

Los comodines solo funcionan con los caracteres que se encuentran dentro de la cadena del tema. El campo SOWCHR trata la barra inclinada (/) como un carácter más sin ningún significado especial.

El comportamiento definido por SOWCHR se muestra en la siguiente tabla:

Tabla 768. Cómo se interpretan los comodines	
Carácter especial	Comportamiento
*	Comodín, cero o más caracteres
?	Comodín, un carácter
%	Carácter de escape para permitir que los caracteres «*», «?» o «%» se utilicen en una cadena y no se interpreten como caracteres especiales, por ejemplo, «%*», «%?» o «%%».

Por ejemplo, publicar sobre el siguiente tema:

```
/level0/level1/level2/level3/level4
```

empareja a los suscriptores utilizando los siguientes temas:

```
*  
/*  
/ level0/level1/level2/level3/*  
/ level0/level1/*/level3/level4  
/ level0/level1/le?el2/level3/level4
```

Nota: Este uso de comodines proporciona exactamente el significado que se indica en IBM MQ V6 y WebSphere MB V6 cuando se utilizan mensajes con formato MQRFH1 para publicar/suscribirse. Se recomienda no utilizarlo en aplicaciones de nueva creación, sino solo en aquellas que ya funcionaban con esa versión y no se han modificado para utilizar el comportamiento comodín predeterminado, tal y como se describe en SOWTOP.

SOWTOP

Los comodines solo funcionan en elementos temáticos dentro de la cadena temática. Este es el comportamiento predeterminado si no se elige ninguno.

El comportamiento requerido por SOWTOP se muestra en la siguiente tabla:

Tabla 769. Cómo se interpretan los comodines	
Carácter especial	Comportamiento
/	Separador de nivel de tema
#	Comodín: nivel de tema múltiple
+	Comodín: nivel de tema único

Nota:

Los caracteres «+» y «#» no se tratan como comodines si se mezclan con otros caracteres (incluidos ellos mismos) dentro de un nivel de tema. En la siguiente cadena, los caracteres «#» y «+» se tratan como caracteres normales.

```
level0/level1/#+/level3/level#
```

Por ejemplo, publicar sobre el siguiente tema:

```
/level0/level1/level2/level3/level4
```

empareja a los suscriptores utilizando los siguientes temas:

```
#  
/#  
/ level0/level1/level2/level3/#  
/ level0/level1+/level3/level4
```

Nota: Este uso de comodines proporciona el significado indicado en WebSphere Message Broker 6 al utilizar mensajes con formato MQRFH2 para publicar/suscribirse.

Otras opciones : Las siguientes opciones controlan la forma en que se emite la llamada a la API, en lugar de la suscripción. Al regresar de una llamada MQSUB utilizando SORES, estas opciones no cambian.

SOALTU

El campo SDAU contiene un identificador de usuario que se utiliza para validar esta llamada MQSUB. La llamada solo puede tener éxito si esta SDAU está autorizada para abrir el objeto con las opciones de acceso especificadas, independientemente de si el identificador de usuario con el que se ejecuta la aplicación está autorizado para hacerlo.

SOSCID

La suscripción consiste en utilizar el identificador de correlación proporcionado en el *SDCID* campo. Si no se especifica esta opción, el gestor de colas crea automáticamente un identificador de correlación en el momento de la suscripción y lo devuelve a la aplicación en el *SDCID* campo. Consulte [SDCID \(cadena de bits de 24 bytes\)](#) para obtener más información.

SOSETI

La suscripción consiste en utilizar el token de contabilidad y los datos de identidad de la aplicación proporcionados en los campos *SDACC* *SDAID* y.

Si se especifica esta opción, se lleva a cabo la misma comprobación de autorización que si se accediera a la cola de destino mediante una llamada *MQOPEN* con *00SETI*, excepto en el caso de que también se utilice la opción *SOMAN*, en cuyo caso no se realiza ninguna comprobación de autorización en la cola de destino.

Si no se especifica esta opción, las publicaciones enviadas a este suscriptor tendrán asociada la siguiente información contextual predeterminada:

Tabla 770. Información contextual predeterminada para las publicaciones enviadas a este suscriptor	
Campo de MQMD	Valor utilizado
<i>MDUID</i>	El ID de usuario asociado a la suscripción en el momento en que se realizó la suscripción.
<i>MDACC</i>	Se determina a partir del entorno si es posible; si no, se establece en <i>ACNONE</i> .
<i>MDAID</i>	Establecer como en blanco

Esta opción solo es válida con *SOCRE* y *SOALT*. Si se utiliza con *SORES*, los campos *SDACC* *SDAID* y se ignoran, por lo que esta opción no tiene ningún efecto.

Si se modifica una suscripción sin utilizar esta opción en la que previamente la suscripción había facilitado información de contexto de identidad, se genera información del contexto predeterminado para la suscripción modificada.

Si una suscripción que permite que diferentes ID de usuario la utilicen con la opción *SOAUID* es reanudada por un ID de usuario diferente, se genera un contexto de identidad predeterminado para el nuevo ID de usuario que ahora es propietario de la suscripción y todas las publicaciones posteriores se entregan con el nuevo contexto de identidad.

SOFIQ

La llamada *MQSUB* falla si el gestor de colas se encuentra en estado de inactividad. En *z/OS*, para una aplicación *CICS IMS* o, esta opción también fuerza el fallo de la llamada *MQSUB* si la conexión se encuentra en estado de inactividad.

SDAU (cadena de caracteres de 12 bytes)

Si se especifica *SOALTU*, este campo contiene un identificador de usuario alternativo que se utiliza para comprobar la autorización de la suscripción y para la salida a la cola de destino (especificada en el **Hobj** parámetro de la llamada *MQSUB*), en lugar del identificador de usuario con el que se está ejecutando actualmente la aplicación.

Si tiene éxito, el identificador de usuario especificado en este campo se registra como el identificador de usuario propietario de la suscripción en lugar del identificador de usuario con el que se está ejecutando actualmente la aplicación.

Si se especifica *SOALTU* y este campo está completamente en blanco hasta el primer carácter nulo o el final del campo, la suscripción solo podrá realizarse correctamente si no se requiere autorización de usuario para suscribirse a este tema con las opciones especificadas o la cola de destino para la salida.

Si no se especifica *SOALTU*, este campo se ignora.

Al regresar de una llamada MQSUB utilizando SORES, este campo permanece sin cambios.

Este es un campo de entrada. La longitud de este campo viene dada por LNUID. El valor inicial de este campo es 12 caracteres en blanco.

SDPRI (entero con signo de 10 dígitos)

Este es el valor que se encuentra en el *MQPRI* campo del descriptor de mensajes (MQMD) de todos los mensajes de publicación que coinciden con esta suscripción. Para obtener más información sobre el *MQPRI* campo en el MQMD, consulte [MDPRI](#).

El valor debe ser mayor o igual que cero, donde cero es la prioridad más baja. También se pueden utilizar los valores especiales siguientes:

PRQDEF

Cuando se proporciona una cola de suscripción en el campo Hobj de la llamada MQSUB y no es un identificador gestionado, la prioridad del mensaje se toma del **DefPriority** atributo de esta cola. Si la cola así identificada es una cola de clúster o hay más de una definición en la ruta de resolución del nombre de la cola, la prioridad se determina cuando el mensaje de publicación se coloca en la cola, tal y como se describe para [MDPRI](#).

Si la llamada MQSUB utiliza un identificador gestionado, la prioridad del mensaje se toma del **DefPriority** atributo de la cola modelo asociada al tema suscrito.

PRPUB

La prioridad del mensaje es la prioridad de la publicación original. Este es el valor inicial del campo.

Si se modifica una suscripción existente utilizando la opción SOALT, se puede cambiar la *MQPRI* de cualquier mensaje de publicación futuro.

Al regresar de una llamada MQSUB utilizando SORES, este campo se establece en la prioridad actual que se utiliza para la suscripción.

SDRO (MQCHARV)

SDRO es el nombre largo del objeto después de que el gestor de colas resuelve el nombre proporcionado en *SDON*.

Si se proporciona el nombre largo del objeto en *SDOS* y no se proporciona nada en *SDON*, el valor devuelto en este campo es el mismo que el proporcionado en *SDOS*.

Si se omite este campo (es decir, si *SDRO.VSBufSize* es cero), no se devuelve *SDRO* el, pero se devuelve la longitud en *SDRO.VSLength*. Si la longitud es menor que la completa *SDRO*, se trunca y devuelve tantos caracteres de la derecha como quepan en la longitud proporcionada.

Si *SDRO* se especifica incorrectamente, según la descripción de cómo utilizar la estructura [MQCHARV](#), o si supera la longitud máxima, la llamada falla con el código de motivo RC2520.

SDSID (cadena de caracteres de 4 bytes)

Este es el identificador de estructura; el valor debe ser:

SDSIDV

Identificador para la estructura del descriptor de suscripción.

Este campo siempre es un campo de entrada. El valor inicial de este campo es SDSIDV.

SDSL (entero con signo de 10 dígitos)

Éste es el nivel asociado a la suscripción. Las publicaciones solo se entregan a esta suscripción si se encuentra en el conjunto de suscripciones con el valor *SDSL* más alto inferior o igual al *PubLevel* e utilizado en el momento de la publicación.

El valor debe estar comprendido entre cero y 9. Cero sería el nivel más bajo.

El valor inicial de este campo es 1.

Si se modifica una suscripción existente utilizando la opción SOALT, entonces *SDSL* no se puede cambiar.

SDSN (MQCHARV)

SDSN especifica el nombre de la suscripción.

Este campo solo es obligatorio si *SDOPT* especifica la opción SODUR, pero si se proporciona, el gestor de colas también lo utiliza para SONDUR. Si se especifica, *SDSN* debe ser único dentro del gestor de colas, ya que es el campo utilizado para identificar las suscripciones.

La longitud máxima de *SDSN* es 10240.

Este campo tiene dos funciones. En el caso de una suscripción SODUR, es el medio por el que se identifica una suscripción para reanudarla después de haberla creado, si se ha cerrado el identificador de la suscripción (utilizando la opción COKPSB) o se ha desconectado del gestor de colas. Para identificar una suscripción y eliminarla después de haberla creado, se utiliza la llamada MQSUB con la opción SORES. El campo SDSN también se muestra en la vista de administración de suscripciones en el *SDSN* campo en DISPLAY SBSTATUS.

Si *SDSN* se especifica incorrectamente, según la descripción de cómo utilizar la estructura MQCHARV, o si supera la longitud máxima, o si se omite cuando es necesario (es decir *SDSN*,. *VCHRL* es cero), o si supera la longitud máxima, la llamada falla con el código de motivo RC2440 .

Este es un campo de entrada. Los valores iniciales de los campos de esta estructura son los mismos que los de la estructura MQCHARV.

Si se modifica una suscripción existente utilizando la SOALT opción, no se puede cambiar el nombre de la suscripción, ya que es el campo que se utiliza para identificarla. No se modifica en la salida de una llamada MQSUB con la opción SORES.

SDSS (MQCHARV)

SDSS es la cadena que proporciona los criterios de selección utilizados al suscribirse a los mensajes de un tema.

Este campo de longitud variable se devuelve en la salida de una llamada MQSUB utilizando la opción SORES, si se proporciona un búfer y si también hay una longitud de búfer positiva en VSBufSize. Si no se proporciona ningún búfer en la llamada, solo se devuelve la longitud de la cadena de selección en el campo VSLength de MQCHARV. Si el búfer proporcionado es inferior al espacio necesario para devolver el campo, solo se devuelven VSBufSize bytes en el búfer.

Si *SDSS* se especifica incorrectamente, según la descripción de cómo utilizar la estructura MQCHARV, o si supera la longitud máxima, la llamada falla con el código de motivo RC2519 .

SDSUD (MQCHARV)

Los datos proporcionados en la suscripción en este campo se incluyen como la propiedad del mensaje « mq.SubUserData » (Suscripción de correo electrónico) de cada publicación enviada a esta suscripción.

La longitud máxima de *SDSUD* es 10240.

Si *SDSUD* se especifica incorrectamente, según la descripción de cómo utilizar la estructura MQCHARV, o si supera la longitud máxima, la llamada falla con el código de motivo RC2431.

Este es un campo de entrada. Los valores iniciales de los campos de esta estructura son los mismos que los de la estructura MQCHARV.

Si se modifica una suscripción existente utilizando la opción SOALT, se pueden cambiar los datos de usuario de la suscripción.

Este campo de longitud variable se devuelve en la salida de una llamada MQSUB utilizando la opción SORES, si se proporciona un búfer y hay una longitud de búfer positiva en *VSBuflen*. Si no se proporciona ningún búfer en la llamada, solo se devuelve la longitud de los datos de usuario de la suscripción en el *VCHRL* campo del MQCHARV. Si el búfer proporcionado es más pequeño

que el espacio necesario para devolver el campo, solo se *VSBuflen* devuelven bytes en el búfer proporcionado.

SDVER (entero con signo de 10 dígitos)

Este es el número de versión de la estructura; el valor debe ser:

SDVER1

Version-1 Estructura del descriptor de suscripción.

La siguiente constante especifica el número de versión de la versión actual:

SDVERC

Versión actual de la estructura del descriptor de suscripción.

Este campo siempre es un campo de entrada. El valor inicial del campo es SDVER1.

Valores iniciales

Tabla 771. Campos en MQSD		
Nombre de campo	Nombre de la constante	Valor de la constante
SDSID	SDSIDV	'SD¬¬'
SDVER	SDVER1	1
SDOPT	SONDUR	0
SDON	Ninguna	Espacios en blanco
SDAU	Ninguna	Espacios en blanco
SDASI	SINONE	Nulos
SDEXP	EIULIM	-1
SDOS	Nombres y valores tal y como se definen para MQCHARV	
SDSN	Nombres y valores tal y como se definen para MQCHARV	
SDSUD	Nombres y valores tal y como se definen para MQCHARV	
SDCID	CINONE	Nulos
SDPRI	PRQDEF	-3
SDACC	ACNONE	Nulos
SDAID	Ninguna	Espacios en blanco
SDSL	Ninguna	1
SDRO	Nombres y valores tal y como se definen en MQCHARV	
Nota:		
1. El símbolo ¬ representa un único carácter en blanco.		

Declaración RPG

```
D*..1.....2.....3.....4.....5.....6.....7..
D* MQSD Structure
D*
D* Structure identifier
```

D	SDSID	1	4
D*	Structure version number		
D	SDVER	5	8I 0
D*	Options associated with subscribing		
D	SDOPT	9	12I 0
D*	Object name		
D	SDON	13	60
D*	Alternate user identifier		
D	SDAU	61	72
D*	Alternate security identifier		
D	SDASI	73	112
D*	Expiry of Subscription		
D	SDEXP	113	116I 0
D*	Object Long name		
D	SDOSP	117	132*
D	SDOSO	133	136I 0
D	SDOSS	137	140I 0
D	SDOSL	141	144I 0
D	SDOSC	145	148I 0
D*	Subscription name		
D	SDSNP	149	164*
D	SDSNO	165	168I 0
D	SDSNS	169	172I 0
D	SDSNL	173	176I 0
D	SDSNC	177	180I 0
D*	Subscription User data		
D	SDSUDP	181	196*
D	SDSUDO	197	200I 0
D	SDSUDS	201	204I 0
D	SDSUDL	205	208I 0
D	SDSUDC	209	212I 0
D*	Correlation Id related to this subscription		
D	SDCID	213	236
D*	Priority set in publications		
D	SDPRI	237	240I 0
D*	Accounting Token set in publications		
D	SDACC	241	272
D*	Appl Identity Data set in publications		
D	SDAID	273	304
D*	Message Selector		
D	SDSSP	305	320*
D	SDSSO	321	324I 0
D	SDSSS	325	328I 0
D	SDSSL	329	332I 0
D	SDSSC	333	336
D*	Subscription level		
D	SDSL	337	340 0
D*	Resolved Long object name		
D	SDROP	341	356*
D	SDROO	357	360I 0
D	SDROS	361	364I 0
D	SDROL	365	368I 0
D	SDROC	369	372I 0

MQSMPO (Establecer opciones de propiedad del mensaje) activado

IBM i

La **MQSMPO** estructura permite a las aplicaciones especificar opciones que controlan cómo se establecen las propiedades de los mensajes.

Visión general

Propósito : La estructura es un parámetro de entrada en la **MQSETMP** llamada.

Conjunto de caracteres y codificación : Los datos en **MQSMPO** deben estar en el conjunto de caracteres de la aplicación y la codificación de la aplicación (ENNAT).

- “Campos” en la [página 1255](#)
- “Valores iniciales” en la [página 1256](#)
- “Declaración RPG” en la [página 1256](#)

Campos

La estructura MQSMPO contiene los siguientes campos, descritos en **orden alfabético** :

SPOPT (entero con signo de 10 dígitos)

Opciones de ubicación: Las siguientes opciones se refieren a la ubicación relativa de la propiedad en comparación con el cursor de la propiedad:

SPSETF

Establece el valor de la primera propiedad que coincide con el nombre especificado o, si no existe, añade una nueva propiedad después de todas las demás propiedades con una jerarquía coincidente.

SPSETC

Establece el valor de la propiedad señalada por el cursor de propiedad. La propiedad señalada por el cursor de propiedad es la última que se consultó utilizando la opción IPINQF o IPINQN.

El cursor de propiedad se restablece cuando se reutiliza el identificador de mensaje o cuando se especifica el identificador de mensaje en el *HMSG* campo de la estructura MQGMO en una llamada MQGET o en la estructura MQPMO en una llamada MQPUT.

Si se utiliza esta opción cuando aún no se ha establecido el cursor de propiedad o si se ha eliminado la propiedad a la que apunta el cursor de propiedad, la llamada falla con el código de finalización CCFAIL y el código de motivo RC2471.

SPSETA

Establece una nueva propiedad después de la propiedad señalada por el cursor de propiedad. La propiedad señalada por el cursor de propiedad es la última consultada mediante la opción IPINQF o IPINQO.

El cursor de propiedad se restablece cuando se reutiliza el identificador de mensaje o cuando se especifica el identificador de mensaje en el *HMSG* campo de la estructura MQGMO en una llamada MQGET o en la estructura MQPMO en una llamada MQPUT.

Si se utiliza esta opción cuando aún no se ha establecido el cursor de propiedad o si se ha eliminado la propiedad a la que apunta el cursor de propiedad, la llamada falla con el código de finalización CCFAIL y el código de motivo RC2471.

Si no necesita ninguna de las opciones descritas, utilice la siguiente opción:

SPNONE

No se ha especificado ninguna opción.

Este campo siempre es un campo de entrada. El valor inicial de este campo es SPSETF.

SPSID (entero con signo de 10 dígitos)

Este es el identificador de estructura; el valor debe ser:

SPSIDV

Identificador para la estructura de opciones de propiedades de mensajes establecidos.

Este campo siempre es un campo de entrada. El valor inicial de este campo es **SPSIDV**.

SPVAKCSI (entero con signo de 10 dígitos)

El conjunto de caracteres del valor de la propiedad que se va a establecer si el valor es una cadena de caracteres.

Este campo siempre es un campo de entrada. El valor inicial de este campo es **CSAPL**.

SPVALENC (entero con signo de 10 dígitos)

La codificación del valor de la propiedad que se establecerá si el valor es numérico.

Este campo siempre es un campo de entrada. El valor inicial de este campo es **ENNAT**.

SPVER (entero con signo de 10 dígitos)

Este es el número de versión de la estructura; el valor debe ser:

SPVER1

Version-1 Establecer la estructura de opciones de propiedades del mensaje.

La siguiente constante especifica el número de versión de la versión actual:

SPVERC

Estructura actual de las opciones de configuración de la propiedad del mensaje.

Este campo siempre es un campo de entrada. El valor inicial de este campo es **SPVER1**.

Valores iniciales

Tabla 772. Campos en MQSMPO		
Nombre de campo	Nombre de la constante	Valor de la constante
SPSID	SPSIDV	'SMPO'
SPVER	SPVER1	1
SPOPT	SPNONE	0
SPVALENC	ENNAT	Depende del entorno
SPVALCSI	CSAPL	-3

Declaración RPG

```
D* MQSMPO Structure
D*
D*
D* Structure identifier
D  SPSID              1      4      INZ('SMPO')
D*
D* Structure version number
D  SPVER              5      8I 0  INZ(1)
D*
** Options that control the action of
D* MQSETMP
D  SPOPT              9      12I 0  INZ(0)
D*
D* Encoding of Value
D  SPVALENC          13      16I 0  INZ(273)
D*
D* Character set identifier of Value
D  SPVALCSI          17      20I 0  INZ(-3)
```

IBM i MQSRO (Opciones de solicitud de suscripción) en IBM i

La estructura MQSRO permite a la aplicación especificar opciones que controlan cómo se realiza una solicitud de suscripción.

Visión general

Propósito : La estructura es un parámetro de entrada/salida en la llamada MQSUBRQ.

Versión : La versión actual de MQSRO es SRVER1.

- [“Campos” en la página 1257](#)
- [“Valores iniciales” en la página 1257](#)
- [“Declaración RPG” en la página 1258](#)

Campos

La estructura MQSRO contiene los siguientes campos, descritos en **orden alfabético** :

SRNMP (entero con signo de 10 dígitos)

Este es un campo de salida, devuelto a la aplicación para indicar el número de publicaciones enviadas a la cola de suscripción como resultado de esta llamada. Aunque se ha enviado este número de publicaciones como resultado de esta llamada, no hay garantía de que la aplicación pueda obtener tantos mensajes, especialmente si se trata de mensajes no persistentes.

Puede haber más de una publicación si el tema suscrito contiene un comodín. Si no había comodines en la cadena del tema cuando se creó la *HSUB* suscripción representada por, entonces se envía como máximo una publicación como resultado de esta llamada.

SROPT (entero con signo de 10 dígitos)

Se debe especificar una de las siguientes opciones. Solo se puede especificar una opción.

Otras opciones: La siguiente opción controla lo que ocurre cuando el gestor de colas está inactivo:

SRFIQ

La llamada MQSUBRQ falla si el gestor de colas se encuentra en estado de inactividad.

Opción predeterminada: Si no se requiere la opción descrita anteriormente, se debe utilizar la siguiente opción:

SRNONE

Utilice este valor para indicar que no se han especificado otras opciones; todas las opciones toman sus valores predeterminados.

SRNONE ayuda a documentar el programa. Aunque no está previsto que esta opción se utilice con ninguna otra, dado que su valor es cero, este uso no puede detectarse.

SRSID (cadena de caracteres de 4 bytes)

Este es el identificador de estructura; el valor debe ser:

SRSIDV

Identificador para la estructura SROPT de solicitud de suscripción.

Este campo siempre es un campo de entrada. El valor inicial de este campo es SRSIDV.

SRVER (entero con signo de 10 dígitos)

Este es el número de versión de la estructura; el valor debe ser:

SRVER1

Version-1 Estructura de opciones de solicitud de suscripción.

La siguiente constante especifica el número de versión de la versión actual:

SRVERC

Versión actual de la estructura de opciones de solicitud de suscripción.

Este campo siempre es un campo de entrada. El valor inicial de este campo es SRVER1.

Valores iniciales

Tabla 773. Campos en MQSRO		
Nombre de campo	Nombre de la constante	Valor de la constante
SRSID	SRSIDV	'SRO~'
SRVER	SRVER1	1
SROPT	SRNONE	0

Tabla 773. Campos en MQSRO (continuación)

Nombre de campo	Nombre de la constante	Valor de la constante
SRNMP	Ninguna	0
Notas: 1. El símbolo ~ representa un único carácter en blanco. 2. El valor «Cadena nula o espacios en blanco» denota la cadena nula en C y los caracteres en blanco en otros lenguajes de programación.		

Declaración RPG

```

D*.1....2....3....4....5....6....7..
D* MQSRO Structure
D*
D* Structure identifier
D  SRSID          1          4
D* Structure version number
D  SRVER          5          8I 0
D* Options that control the action of MQSUBRQ
D  SROPT          9          12I 0
D* Number of publications sent
D  SRNMP         13          16I 0
  
```

IBM i MQSTS (Estructura de notificación de estado) en IBM i

La estructura MQSTS describe los datos de la estructura de estado devuelta por el comando MQSTAT.

Visión general

Conjunto de caracteres y codificación : Los datos de caracteres en MQSTS están en el conjunto de caracteres del gestor de colas local; esto viene dado por el atributo del *CodedCharSetId* gestor de colas. Los datos numéricos en MQSTS están en la codificación nativa de la máquina; esto lo proporciona *ENNAT*.

Uso : El comando MQSTAT se utiliza para recuperar información sobre el estado. Esta información se devuelve en una estructura MQSTS. Para obtener información sobre MQSTAT, consulte [“MQSTAT \(Recuperar información de estado\) en IBM i” en la página 1390](#).

- [“Campos” en la página 1258](#)
- [“Valores iniciales” en la página 1261](#)
- [“Declaración RPG” en la página 1262](#)

Campos

La estructura MQSTS contiene los siguientes campos, que se describen en **orden alfabético** :

STSCC (entero con signo de 10 dígitos)

Este es el código de finalización resultante del primer error notificado en la estructura MQSTS.

Este campo siempre es de salida. El valor inicial de este campo es CCOK.

STSFC (entero con signo de 10 dígitos)

Este es el número de llamadas asíncronas put que han fallado.

Se trata de un campo de salida. El valor inicial de este campo es 0.

STSOBJN (cadena de caracteres de 48 bytes)

Este es el nombre local del objeto involucrado en el primer fallo.

Se trata de un campo de salida. El valor inicial de este campo es 48 caracteres en blanco.

STSOQMGR (cadena de caracteres de 48 bytes)

Este es el nombre del gestor de colas en el que se define el *STSOBJN* objeto. Un nombre que está completamente en blanco hasta el primer carácter nulo o el final del campo indica el gestor de colas al que está conectada la aplicación (el gestor de colas local).

Se trata de un campo de salida. El valor inicial de este campo es 48 caracteres en blanco.

STS00 (entero con signo de 10 dígitos)

Se STS00 utiliza para abrir el objeto sobre el que se informa. Presente solo en la versión 2 de MQSTS o superior.

El valor de STS00 depende del valor del parámetro MQSTAT**STYPE** .

STATAPT

Cero.

STATREC

Cero.

STATRER

El STS00 utilizado cuando se produjo el fallo. El motivo del fallo se indica en los campos *STSCC* *STSRC* y de la MQSTS estructura.

STS00 es un campo de salida. Su valor inicial es cero.

STSOS (MQCHARV)

Nombre largo del objeto defectuoso sobre el que se informa. Presente solo en la versión 2 de MQSTS o superior.

STSOS es un campo MQCHARV con una longitud máxima de 10240. Consulte [MQCHARV](#) para obtener una descripción de cómo utilizar la estructura MQCHARV.

La interpretación de STSOS depende del valor del parámetro MQSTAT**STYPE** .

STATAPT

Este es el nombre completo del objeto de la cola o tema utilizado en la MQPUT operación que ha fallado.

STATREC

Serie de caracteres de longitud cero

STATRER

Este es el nombre completo del objeto que provocó el fallo de la reconexión.

STSOS es un campo de salida. Su valor inicial es una cadena de longitud cero.

STSOT (entero con signo de 10 dígitos)

El tipo de objeto que se nombra en *ObjectName*. Los valores posibles son:

OTALSQ

Cola alias.

OTLOCQ

Cola local.

OTMODQ

Cola modelo.

OTQ

Cola.

OTREMQ

Cola remota.

OTTOP

Este campo siempre es de salida. El valor inicial de este campo es OTQ.

STSRC (entero con signo de 10 dígitos)

Este es el código de motivo resultante del primer error notificado en la estructura MQSTS

Este campo siempre es de salida. El valor inicial de este campo es RCNONE.

STSRBJN (cadena de caracteres de 48 bytes)

Este es el nombre de la cola de destino especificada en *STSRBJN* después de que el gestor de colas local resuelva el nombre. El nombre devuelto es el nombre de una cola que existe en el gestor de colas identificado por *STSRQMGR*.

Solo se devuelve un valor distinto de blanco si el objeto es una cola única abierta para exploración, entrada o salida (o cualquier combinación). Si el objeto abierto es cualquiera de los siguientes, *STSRBJN* se establece en espacios en blanco:

- Un tema
- Una cola, pero no abierta para exploración, entrada o salida

Se trata de un campo de salida. El valor inicial de este campo es 48 caracteres en blanco.

STSRQMGR (cadena de caracteres de 48 bytes)

Este es el nombre del gestor de colas de destino después de que el gestor de colas local resuelva el nombre. El nombre devuelto es el nombre del gestor de colas propietario de la cola identificada por *STSRBJN*. *STSRQMGR* puede ser el nombre del administrador de colas local.

Si *STSRBJN* es una cola compartida que pertenece al grupo de uso compartido de colas al que pertenece el gestor de colas local, *STSRQMGR* es el nombre del grupo de uso compartido de colas. Si la cola pertenece a otro grupo de uso compartido de colas, *STSRBJN* puede ser el nombre del grupo de uso compartido de colas o el nombre de un gestor de colas que sea miembro del grupo de uso compartido de colas (la naturaleza del valor devuelto viene determinada por las definiciones de colas que existen en el gestor de colas local).

Solo se devuelve un valor distinto de blanco si el objeto es una cola única abierta para exploración, entrada o salida (o cualquier combinación). Si el objeto abierto es cualquiera de los siguientes, *STSRQMGR* se establece en espacios en blanco:

- Un tema
- Una cola, pero no abierta para exploración, entrada o salida
- Una cola de clúster con OOBNDN especificado (o con OOBNDQ en vigor cuando el **DefBind** atributo de cola tiene el valor OOBNDN)

Se trata de un campo de salida. El valor inicial de este campo es 48 caracteres en blanco.

STSSC (entero con signo de 10 dígitos)

Este es el número de llamadas asíncronas put que se han realizado correctamente.

Se trata de un campo de salida. El valor inicial de este campo es 0.

STSSID (cadena de caracteres de 4 bytes)

Este es el identificador de la estructura. El valor debe ser:

STSSID

Identificador de la estructura de notificación de estado.

El valor inicial de este campo es STSSID.

STSSO (entero con signo de 10 dígitos)

Se STSSO utilizaba para abrir la suscripción caducada. Presente solo en la versión 2 de MQSTS o superior.

La interpretación de STSS0 depende del valor del parámetro MQSTAT**STYPE** .

STATAPT

Cero.

STATREC

Cero.

STATRER

El STSS0 utilizado cuando se produjo el fallo. El motivo del fallo se indica en los campos STSCC STSRC y de la MQSTS estructura. Si el error no está relacionado con la suscripción a un tema, el valor devuelto es cero.

STSS0 es un campo de salida. Su valor inicial es cero.

STSSUN (MQCHARV)

El nombre de la suscripción fallida. Presente solo en la versión 2 de MQSTS o superior.

STSSUN es un campo MQCHARV con una longitud máxima de 10240. Consulte [MQCHARV](#) para obtener una descripción de cómo utilizar la estructura MQCHARV.

La interpretación de STSSUN depende del valor del parámetro MQSTAT**STYPE** .

STATAPT

Cadena de longitud cero.

STATREC

Cadena de longitud cero.

STATRER

El nombre de la suscripción que provocó el fallo de la reconexión. Si no hay ningún nombre de suscripción disponible, o si el error no está relacionado con una suscripción, se trata de una cadena de longitud cero.

STSSUN es un campo de salida. Su valor inicial es una cadena de longitud cero.

STSVR (entero con signo de 10 dígitos)

Este es el número de versión de la estructura. El valor debe ser:

STSVR1

Número de versión para la estructura de informes de estado.

La siguiente constante especifica el número de versión de la versión actual:

STSVRC

Versión actual de la estructura de informes de estado.

El valor inicial de este campo es STSVR1.

STSWC (entero con signo de 10 dígitos)

Este es el número de llamadas asíncronas put que se completaron con una advertencia.

Se trata de un campo de salida. El valor inicial de este campo es 0.

Valores iniciales

Tabla 774. Campos en MQSTS		
Nombre de campo	Nombre de la constante	Valor de la constante
STSSID	STSID	
STSVR	STSVRC	STSVR1

Tabla 774. Campos en MQSTS (continuación)		
Nombre de campo	Nombre de la constante	Valor de la constante
STSCC	CCOK	0
STSRC	RCNONE	0
STSSC	Ninguna	0
STSWC	Ninguna	0
STSFC	Ninguna	0
STSOT	Ninguna	0
STSOBJN	Ninguna	Espacios en blanco
STSOQMGR	Ninguna	Espacios en blanco
STSR OBJN	Ninguna	Espacios en blanco
STSRQMGR	Ninguna	Espacios en blanco
STSOS	Nombres y valores tal y como se definen para MQCHARV	
STSSUN	Nombres y valores tal y como se definen para MQCHARV	
STS00	Ninguna	0
STSS0	Ninguna	0

Declaración RPG

```

D*.1.....2.....3.....4.....5.....6.....7..
D* MQSTS Structure
D*
D* Structure identifier
D STSSID 1 4
D* Structure version number
D STSVER 5 8I 0
D* Completion code
D STSCC 9 12I 0
D* Reason code
D STSRC 13 16I 0
D* Success count
D STSSC 17 20I 0
D* Warning count
D STSWC 21 24I 0
D* Failure count
D STSFC 25 28I 0
D* Object type
D STSOT 29 32I 0
D* Object name
D STSOBJN 33 80
D* Object queue manager
D STSOQMGR 81 128
D* Resolved object name
D STSR OBJN 129 176
D* Resolved object queue manager name
D STSRQMGR 177 224
D* Ver:1 **
D* Failing object long name
D* Address of variable length string
D STSOSCHRP 225 240*
D* Offset of variable length string
D STSOSCHRO 241 244I 0
D* Size of buffer
D STSOSVSBS 245 248I 0
D* Length of variable length string
D STSOSCHRL 249 252I 0
D* CCSID of variable length string

```

```

D STSOSCHRC          253      256I 0
D* Failing subscription name
D* Address of variable length string
D STSSUNCHRP         257      272*
D* Offset of variable length string
D STSSUNCHRO         273      276I 0
D* Size of buffer
D STSSUNVSBS         277      280I 0
D* Length of variable length string
D STSSUNCHRL         281      284I 0
D* CCSID of variable length string
D STSSUNCHRC         285      288I 0
D* Failing open options
D STS00              289      292I 0
D* Failing subscription options
D STSS0              293      296I 0
D* Ver:2 **

```

MQTM - Mensaje de activación

La estructura MQTM describe los datos del mensaje de activación que envía el gestor de colas a una aplicación de supervisión de activadores cuando se produce un evento de activación para una cola.

Visión general

Propósito : Esta estructura forma parte de la Interfaz del IBM MQ monitor de disparadores (TMI), que es una de las IBM MQ interfaces del marco.

Nombre del formato : FMTM.

Conjunto de caracteres y codificación : Los datos de caracteres en MQTM se encuentran en el conjunto de caracteres del gestor de colas que genera el MQTM. Los datos numéricos en MQTM están en la codificación de máquina del gestor de colas que genera el MQTM.

El conjunto de caracteres y la codificación del MQTM vienen dados por los campos *MDCSI MDENC* y en:

- El MQMD (si la estructura MQTM se encuentra al inicio de los datos del mensaje), o
- La estructura de encabezado que precede a la estructura MQTM (en todos los demás casos).

Uso : Es posible que una aplicación de monitorización de desencadenantes necesite pasar parte o toda la información del mensaje de desencadenante a la aplicación que inicia la aplicación de monitorización de desencadenantes. La información que puede necesitar la aplicación iniciada incluye *TMQN*, *TMTD*, y *TMUD*. La aplicación de monitorización de disparadores puede pasar la estructura MQTM directamente a la aplicación iniciada, o pasar una estructura e MQTMC2, dependiendo de lo que permita el entorno y resulte conveniente para la aplicación iniciada. Para obtener información sobre MQTMC2, consulte [“MQTMC2 \(Mensaje de activación 2 - formato de caracteres\) activado IBM i”](#) en la página 1267.

- En IBM i, la aplicación de supervisión de desencadenantes proporcionada con IBM MQ pasa una estructura MQTMC2 a la aplicación iniciada.

Para obtener información sobre los desencadenantes, consulte [Requisitos previos para la activación](#).

- [“MQMD para un mensaje de activación”](#) en la página 1263
- [“Campos”](#) en la página 1264
- [“Valores iniciales”](#) en la página 1267
- [“Declaración RPG”](#) en la página 1267

MQMD para un mensaje de activación

Tabla 775. Configuración de los campos en el MQMD de un mensaje de activación generado por el gestor de colas

Campo de MQMD	Valor utilizado
MDSID	MDSIDV

Tabla 775. Configuración de los campos en el MQMD de un mensaje de activación generado por el gestor de colas (continuación)

Campo de MQMD	Valor utilizado
MDVER	MDVER1
MDREP	RONONE
MDMT	MTDGRM
MDEXP	EIULIM
MDFB	FBNONE
MDENC	ENNAT
MDCSI	Atributo CodedCharSetId del gestor de colas
MDFMT	FMTM
MDPRI	Atributo DefPriority de la cola de inicio
MDPER	PENPER
MDMID	Un valor único
MDCID	CINONE
MDBOC	0
MDRQ	Espacios en blanco
MDRM	Nombre de gestor de colas
MDUID	Espacios en blanco
MDACC	ACNONE
MDAID	Espacios en blanco
MDPAT	ATQM, o según corresponda para el agente del canal de mensajes
MDPAN	Primeros 28 bytes del nombre del gestor de colas
MDPD	Fecha en la que se envía el mensaje de activación
MDPT	Momento en el que se envía el mensaje de activación
MDAOD	Espacios en blanco

Se recomienda que una aplicación que genere un mensaje de activación establezca valores similares, excepto en los siguientes casos:

- El **MDPRI** campo se puede establecer en PRQDEF (el gestor de colas lo cambiará a la prioridad predeterminada para la cola de inicio cuando se envíe el mensaje).
- El **MDRM** campo se puede dejar en blanco (el gestor de colas lo cambiará por el nombre del gestor de colas local cuando se envíe el mensaje).
- Los campos de contexto deben configurarse según sea apropiado para la aplicación.

Campos

La estructura MQTM contiene los siguientes campos, que se describen en **orden alfabético** :

TMAI (cadena de caracteres de 256 bytes)

Identificador de la aplicación.

Se trata de una cadena de caracteres que identifica la aplicación que se va a iniciar y que utiliza la aplicación de supervisión de activadores que recibe el mensaje de activación. El gestor de colas

inicializa este campo con el valor del **App1Id** atributo del objeto de proceso identificado por el *TMPN* campo; consulte [“Atributos para definiciones de proceso en IBM i”](#) en la página 1430 para obtener más detalles sobre este atributo. El contenido de estos datos no tiene importancia para el gestor de colas.

El significado de *TMAI* lo determina la aplicación de monitorización de desencadenantes. El monitor de activación proporcionado por IBM MQ requiere *TMAI* que sea el nombre de un programa ejecutable.

La longitud de este campo viene dada por LNPROA. El valor inicial de este campo es 256 caracteres en blanco.

TMAT (entero con signo de 10 dígitos)

Tipo de aplicación.

Esto identifica la naturaleza del programa que se va a iniciar y es utilizado por la aplicación de monitorización de disparadores que recibe el mensaje de disparo. El gestor de colas inicializa este campo con el valor del **App1Type** atributo del objeto de proceso identificado por el *TMPN* campo; consulte [“Atributos para definiciones de proceso en IBM i”](#) en la página 1430 para obtener más detalles sobre este atributo. El contenido de estos datos no tiene importancia para el gestor de colas.

TMAT puede tener uno de los siguientes valores estándar. También se pueden utilizar tipos definidos por el usuario, pero deben restringirse a valores comprendidos entre ATUFST y ATULST:

a la(s)CICS

CICS transacción.

ATVSE

CICS/VSE transacción.

AT400

IBM i .

ATUFST

Valor más bajo para el tipo de aplicación definido por el usuario.

ATULST

Valor más alto para el tipo de aplicación definido por el usuario.

El valor inicial de este campo es 0.

TMED (cadena de caracteres de 128 bytes)

Datos medioambientales.

Se trata de una cadena de caracteres que contiene información relacionada con el entorno de la aplicación que se va a iniciar y que es utilizada por la aplicación de supervisión de activadores que recibe el mensaje de activación. El gestor de colas inicializa este campo con el valor del **EnvData** atributo del objeto de proceso identificado por el *TMPN* campo; consulte [“Atributos para definiciones de proceso en IBM i”](#) en la página 1430 para obtener más detalles sobre este atributo. El contenido de estos datos no tiene importancia para el gestor de colas.

La longitud de este campo viene dada por LNPROE. El valor inicial de este campo es 128 caracteres en blanco.

TMPN (cadena de caracteres de 48 bytes)

Nombre del objeto de proceso.

Este es el nombre del objeto del proceso del gestor de colas especificado para la cola activada, y puede ser utilizado por la aplicación de supervisión de activadores que recibe el mensaje de activación. El gestor de colas inicializa este campo con el valor del **ProcessName** atributo de la cola identificada por el *TMQN* campo; consulte [“Atributos para colas”](#) en la página 1399 para obtener más información sobre este atributo.

Los nombres que son más cortos que la longitud definida del campo siempre se rellenan a la derecha con espacios en blanco; no terminan prematuramente con un carácter nulo.

La longitud de este campo viene dada por LNPRON. El valor inicial de este campo es 48 caracteres en blanco.

TMQN (cadena de caracteres de 48 bytes)

Nombre de la cola activada.

Este es el nombre de la cola para la que se produjo un evento desencadenante, y lo utiliza la aplicación iniciada por la aplicación de supervisión de desencadenantes. El gestor de colas inicializa este campo con el valor del **QName** atributo de la cola activada; consulte [“Atributos para colas” en la página 1399](#) para obtener más información sobre este atributo.

Los nombres que son más cortos que la longitud definida del campo se rellenan a la derecha con espacios en blanco; no terminan prematuramente con un carácter nulo.

La longitud de este campo viene dada por LNQN. El valor inicial de este campo es 48 caracteres en blanco.

TMSID (cadena de caracteres de 4 bytes)

Identificador de estructura.

El valor debe ser:

TMSIDV

Identificador de la estructura del mensaje desencadenante.

El valor inicial de este campo es TMSIDV.

TMTD (cadena de caracteres de 64 bytes)

Datos del desencadenante.

Se trata de datos de formato libre para su uso por parte de la aplicación de monitorización de disparadores que recibe el mensaje de disparo. El gestor de colas inicializa este campo con el valor del **TriggerData** atributo de la cola identificada por el *TMQN* campo; consulte [“Atributos para colas” en la página 1399](#) para obtener más información sobre este atributo. El contenido de estos datos no tiene importancia para el gestor de colas.

La longitud de este campo viene dada por LNTRGD. El valor inicial de este campo es 64 caracteres en blanco.

TMUD (cadena de caracteres de 128 bytes)

Datos de usuario.

Se trata de una cadena de caracteres que contiene información del usuario relevante para la aplicación que se va a iniciar y que es utilizada por la aplicación de supervisión de activadores que recibe el mensaje de activación. El gestor de colas inicializa este campo con el valor del **UserData** atributo del objeto de proceso identificado por el *TMPN* campo; consulte [“Atributos para definiciones de proceso en IBM i” en la página 1430](#) para obtener más detalles sobre este atributo. El contenido de estos datos no tiene importancia para el gestor de colas.

La longitud de este campo viene dada por LNPROU. El valor inicial de este campo es 128 caracteres en blanco.

TMVER (entero con signo de 10 dígitos)

Número de versión de la estructura.

El valor debe ser:

TMVER1

Número de versión para la estructura del mensaje de activación.

La siguiente constante especifica el número de versión de la versión actual:

TMVERC

Versión actual de la estructura del mensaje de activación.

El valor inicial de este campo es TMVER1.

Valores iniciales

Tabla 776. Campos en MQTM		
Nombre de campo	Nombre de la constante	Valor de la constante
TMSID	TMSIDV	' TM↵↵ '
TMVER	TMVER1	1
TMQN	Ninguna	Espacios en blanco
TMPN	Ninguna	Espacios en blanco
TMTD	Ninguna	Espacios en blanco
TMAT	Ninguna	0
TMAI	Ninguna	Espacios en blanco
TMED	Ninguna	Espacios en blanco
TMUD	Ninguna	Espacios en blanco
Notas:		
1. El símbolo ↵ representa un único carácter en blanco.		

Declaración RPG

```
D*..1.....2.....3.....4.....5.....6.....7..
D*
D* MQTM Structure
D*
D* Structure identifier
D  TMSID          1          4  INZ(' TM ')
D* Structure version number
D  TMVER          5          8I 0  INZ(1)
D* Name of triggered queue
D  TMQN          9          56  INZ
D* Name of process object
D  TMPN         57         104  INZ
D* Trigger data
D  TMTD        105         168  INZ
D* Application type
D  TMAT        169        172I 0  INZ(0)
D* Application identifier
D  TMAI        173         428  INZ
D* Environment data
D  TMED        429         556  INZ
D* User data
D  TMUD        557         684  INZ
```

IBM i MQTMC2 (Mensaje de activación 2 - formato de caracteres) activado IBM i

Cuando una aplicación de monitorización de desencadenantes recupera un mensaje de desencadenante (MQTM) de una cola de inicio, es posible que el monitor de desencadenantes tenga que pasar parte o toda la información del mensaje de desencadenante a la aplicación iniciada por el monitor de desencadenantes.

Visión general

Propósito : La información que puede necesitar la aplicación iniciada incluye *TC2QN*, *TC2TD*, y *TC2UD*. La aplicación del monitor de activación puede pasar la estructura MQTM directamente a la aplicación iniciada, o pasar una estructura e MQTMC2 e en su lugar, dependiendo de lo que permita el entorno y resulte conveniente para la aplicación iniciada.

Esta estructura forma parte de la Interfaz del IBM MQ monitor de activadores (TMI), que es una de las IBM MQ interfaces del marco.

Conjunto de caracteres y codificación : Los datos de caracteres en MQTMC2 están en el conjunto de caracteres del gestor de colas local; esto viene dado por el atributo del **CodedCharSetId** gestor de colas.

Uso : La estructura MQTMC2 es similar al formato de la estructura MQTM. La diferencia es que los campos que no son de caracteres en MQTM se cambian en MQTMC2 a campos de caracteres de la misma longitud, y el nombre del gestor de colas se añade al final de la estructura.

- En IBM i, la aplicación de monitorización de desencadenantes proporcionada con IBM MQ pasa una estructura MQTMC2 a la aplicación iniciada.
- “Campos” en la [página 1268](#)
- “Valores iniciales” en la [página 1269](#)
- “Declaración RPG” en la [página 1269](#)

Campos

La estructura MQTMC2 contiene los siguientes campos, que se describen en **orden alfabético** :

TC2AI (cadena de caracteres de 256 bytes)

Identificador de la aplicación.

Véase el *TMAI* campo en la estructura MQTM.

TC2AT (cadena de caracteres de 4 bytes)

Tipo de aplicación.

Este campo siempre contiene espacios en blanco, independientemente del valor del *TMAT* campo en la estructura MQTM del mensaje de activación original.

TC2ED (cadena de caracteres de 128 bytes)

Datos medioambientales.

Véase el *TMED* campo en la estructura MQTM.

TC2PN (cadena de caracteres de 48 bytes)

Nombre del objeto de proceso.

Véase el *TPPN* campo en la estructura MQTM.

TC2QMN (cadena de caracteres de 48 bytes)

Nombre del gestor de colas.

Este es el nombre del gestor de colas en el que se produjo el evento desencadenante.

TC2QN (cadena de caracteres de 48 bytes)

Nombre de la cola activada.

Véase el *TMQN* campo en la estructura MQTM.

TC2SID (cadena de caracteres de 4 bytes)

Identificador de estructura.

El valor debe ser:

TCSIDV

Identificador de la estructura del mensaje desencadenante (formato de caracteres).

TC2TD (cadena de caracteres de 64 bytes)

Datos del desencadenante.

Véase el *TMTD* campo en la estructura MQTM.

TC2UD (cadena de caracteres de 128 bytes)

Datos de usuario.

Véase el *TMUD* campo en la estructura MQTM.

TC2VER (cadena de caracteres de 4 bytes)

Número de versión de la estructura.

El valor debe ser:

TCVER2

Estructura del mensaje de activación de la versión 2 (formato de caracteres).

La siguiente constante especifica el número de versión de la versión actual:

TCVERC

Estructura de la versión actual del mensaje de activación (formato de caracteres).

Valores iniciales

Tabla 777. Campos en MQTMC2		
Nombre de campo	Nombre de la constante	Valor de la constante
TC2SID	TCSIDV	'TMC¬'
TC2VER	TCVER2	'¬¬¬2'
TC2QN	Ninguna	Espacios en blanco
TC2PN	Ninguna	Espacios en blanco
TC2TD	Ninguna	Espacios en blanco
TC2AT	Ninguna	Espacios en blanco
TC2AI	Ninguna	Espacios en blanco
TC2ED	Ninguna	Espacios en blanco
TC2UD	Ninguna	Espacios en blanco
TC2QMN	Ninguna	Espacios en blanco
Notas:		
1. El símbolo ¬ representa un único carácter en blanco.		

Declaración RPG

```
D*.1.....2.....3.....4.....5.....6.....7..
D* MQTMC2 Structure
D*
D* Structure identifier
D TC2SID 1 4
D* Structure version number
D TC2VER 5 8
D* Name of triggered queue
```

D	TC2QN	9	56
D*	Name of process object		
D	TC2PN	57	104
D*	Trigger data		
D	TC2TD	105	168
D*	Application type		
D	TC2AT	169	172
D*	Application identifier		
D	TC2AI	173	428
D*	Environment data		
D	TC2ED	429	556
D*	User data		
D	TC2UD	557	684
D*	Queue manager name		
D	TC2QMN	685	732

MQWIH (Cabecera de información de trabajo) en IBM i

La estructura MQWIH describe la información que debe estar presente al inicio de un mensaje que va a ser gestionado por el gestor z/OS de carga de trabajo.

Visión general

Nombre del formato : FMWIH.

Conjunto de caracteres y codificación : Los campos de la estructura MQWIH están en el conjunto de caracteres y la codificación indicados por los campos *MDCSI MDENC* y de la estructura de encabezado que precede a MQWIH, o por los campos de la estructura MQMD si MQWIH se encuentra al principio de los datos del mensaje de la aplicación.

El juego de caracteres debe ser uno que tenga caracteres de un solo byte para los caracteres que son válidos en los nombres de las colas.

Uso : Si un mensaje debe ser procesado por el gestor z/OS de carga de trabajo, el mensaje debe comenzar con una estructura MQWIH.

- [“Campos” en la página 1270](#)
- [“Valores iniciales” en la página 1272](#)
- [“Declaración RPG” en la página 1273](#)

Campos

La estructura MQWIH contiene los siguientes campos, que se describen en **orden alfabético** :

WICSI (entero con signo de 10 dígitos)

Identificador del conjunto de caracteres de los datos que siguen a MQWIH.

Especifica el identificador del conjunto de caracteres de los datos que siguen a la estructura MQWIH; no se aplica a los datos de caracteres de la propia estructura MQWIH.

En la llamada MQPUT o MQPUT1, la aplicación debe establecer este campo con el valor adecuado para los datos. Se puede utilizar el siguiente valor especial:

CSINHT

Heredar el identificador del conjunto de caracteres de esta estructura.

Los datos de caracteres en los datos *que siguen* esta estructura están en el mismo conjunto de caracteres que esta estructura.

El gestor de colas cambia este valor en la estructura enviada en el mensaje al identificador real del juego de caracteres de la estructura. Si no se produce ningún error, la llamada MQGET no devuelve el valor CSINHT.

CSINHT no se puede utilizar si el valor del *MDPAT* campo en MQMD es ATBRKR.

El valor inicial de este campo es CSUNDF.

WIENC (entero con signo de 10 dígitos)

Codificación numérica de datos que sigue a MQWIH.

Esto especifica la codificación numérica de los datos que siguen a la estructura MQWIH; no se aplica a los datos numéricos de la propia estructura MQWIH.

En la llamada MQPUT o MQPUT1, la aplicación debe establecer este campo con el valor adecuado para los datos.

El valor inicial de este campo es 0.

WIFLG (entero con signo de 10 dígitos)

Flags

El valor debe ser:

WINONE

Sin banderas.

El valor inicial de este campo es WINONE.

WIFMT (cadena de caracteres de 8 bytes)

Formatear el nombre de los datos que siguen a MQWIH.

Especifica el nombre del formato de los datos que siguen a la estructura MQWIH.

En la llamada MQPUT o MQPUT1, la aplicación debe establecer este campo con el valor adecuado para los datos. Las reglas para codificar este campo son las mismas que para el *MDFMT* campo en MQMD.

La longitud de este campo viene dada por LNFMT. El valor inicial de este campo es FMNONE.

WILEN (entero con signo de 10 dígitos)

Longitud de la estructura MQWIH.

El valor debe ser:

WILEN1

Longitud de la estructura del encabezado de información de trabajo de version-1.

La siguiente constante especifica la longitud de la versión actual:

WILENC

Longitud de la estructura actual del encabezado de la información del trabajo.

El valor inicial de este campo es WILEN1.

WIRSV (cadena de caracteres de 32 bytes)

Reservado.

Este es un campo reservado; debe dejarse en blanco.

WISID (cadena de caracteres de 4 bytes)

Identificador de estructura.

El valor debe ser:

WISIDV

Identificador de la estructura del encabezado de la información del trabajo.

El valor inicial de este campo es WISIDV.

WISNM (cadena de caracteres de 32 bytes)

Nombre del servicio.

Este es el nombre del servicio que procesará el mensaje.

La longitud de este campo viene dada por LNSVNM. El valor inicial de este campo es 32 caracteres en blanco.

WISST (cadena de caracteres de 8 bytes)

Nombre del paso del servicio.

Este es el nombre del paso al *WISNM* que se refiere el mensaje.

La longitud de este campo viene dada por LNSVST. El valor inicial de este campo es 8 caracteres en blanco.

WITOK (cadena de bits de 16 bytes)

Token de mensaje.

Se trata de un token de mensaje que identifica de forma única el mensaje.

Para las llamadas MQPUT y MQPUT1, este campo se ignora. La longitud de este campo viene dada por LNMTOK. El valor inicial de este campo es MTKNON.

WIVER (entero con signo de 10 dígitos)

Número de versión de la estructura.

El valor debe ser:

WIVER1

Version-1 Estructura del encabezado de la información del trabajo.

La siguiente constante especifica el número de versión de la versión actual:

WIVERC

Versión actual de la estructura del encabezado de la información del trabajo.

El valor inicial de este campo es WIVER1.

Valores iniciales

Tabla 778. Campos en MQWIH		
Nombre de campo	Nombre de la constante	Valor de la constante
<i>WISID</i>	WISIDV	'WIH¬'
<i>WIVER</i>	WIVER1	1
<i>WILEN</i>	WILEN1	120
<i>WIENC</i>	Ninguna	0
<i>WICSI</i>	CSUNDF	0
<i>WIFMT</i>	FMNONE	Espacios en blanco
<i>WIFLG</i>	WINONE	0
<i>WISNM</i>	Ninguna	Espacios en blanco
<i>WISST</i>	Ninguna	Espacios en blanco
<i>WITOK</i>	MTKNON	Nulos
<i>WIRSV</i>	Ninguna	Espacios en blanco
Notas:		
1. El símbolo ¬ representa un único carácter en blanco.		

Declaración RPG

```
D*..1.....2.....3.....4.....5.....6.....7..
D*
D* MQWIH Structure
D*
D* Structure identifier
D  WISID          1          4      INZ('WIH ')
D* Structure version number
D  WIVER          5          8I 0 INZ(1)
D* Length of MQWIH structure
D  WILEN          9         12I 0 INZ(120)
D* Numeric encoding of data that followsMQWIH
D  WIENC         13         16I 0 INZ(0)
D* Character-set identifier of data thatfollows MQWIH
D  WICSI         17         20I 0 INZ(0)
D* Format name of data that followsMQWIH
D  WIFMT         21         28      INZ('      ')
D* Flags
D  WIFLG         29         32I 0 INZ(0)
D* Service name
D  WISNM        33         64      INZ
D* Service step name
D  WISST        65         72      INZ
D* Message token
D  WITOK        73         88      INZ(X'0000000000000000-
D                                     0000000000000000')
D* Reserved
D  WIRSV        89        120      INZ
```

IBM i MQXQH (encabezado de cola de transmisión) en IBM i

La estructura MQXQH describe la información que se antepone a los datos del mensaje de aplicación de los mensajes cuando se encuentran en colas de transmisión.

Visión general

Propósito : Una cola de transmisión es un tipo especial de cola local que almacena temporalmente mensajes destinados a colas remotas (es decir, destinados a colas que no pertenecen al administrador de colas local). Una cola de transmisión se indica mediante el **Usage** atributo que se indica con el valor USTRAN.

Nombre del formato : FMXQH.

Conjunto de caracteres y codificación : Los datos en MQXQH deben estar en el conjunto de caracteres dado por el atributo del **CodedCharSetId** gestor de colas y la codificación del gestor de colas local dada por ENNAT para el lenguaje de programación C.

El juego de caracteres y la codificación del MQXQH deben configurarse en los campos *MDCSI MDENC* y en:

- El MQMD independiente (si la estructura MQXQH se encuentra al principio de los datos del mensaje), o
- La estructura de encabezado que precede a la estructura MQXQH (en todos los demás casos).

Uso : Un mensaje que se encuentra en una cola de transmisión tiene *dos* descriptores de mensaje:

- Un descriptor de mensaje se almacena por separado de los datos del mensaje; se denomina *descriptor de mensaje independiente* y lo genera el gestor de colas cuando el mensaje se coloca en la cola de transmisión. Algunos de los campos del descriptor de mensajes independiente se copian del descriptor de mensajes proporcionado por la aplicación en la llamada MQPUT o MQPUT1.

El descriptor de mensaje independiente es el que se devuelve a la aplicación en el **MSGDSC** parámetro de la llamada MQGET cuando el mensaje se elimina de la cola de transmisión.

- Se almacena un segundo descriptor de mensaje dentro de la estructura MQXQH como parte de los datos del mensaje; este se denomina *descriptor de mensaje incrustado* y es una copia del descriptor de mensaje proporcionado por la aplicación en la llamada MQPUT o MQPUT1 (con pequeñas variaciones).

El descriptor de mensaje incrustado es siempre un MQMD (version-1). Si el mensaje enviado por la aplicación tiene valores no predeterminados para uno o más de los campos version-2 en el MQMD, una

estructura MQMDE sigue al MQXQH y, a su vez, es seguida por los datos del mensaje de la aplicación (si los hay). El MQMDE es:

- Generado por el gestor de colas (si la aplicación utiliza un MQMD version-2 para colocar el mensaje), o
- Ya presente al inicio de los datos del mensaje de la aplicación (si la aplicación utiliza un MQMD version-1 para enviar el mensaje).

El descriptor de mensaje incrustado es el que se devuelve a la aplicación en el **MSGDSC** parámetro de la llamada MQGET cuando el mensaje se elimina de la cola de destino final.

- [“Campos en el descriptor de mensaje independiente” en la página 1274](#)
- [“Campos del descriptor de mensaje incrustado” en la página 1275](#)
- [“Transferir mensajes a colas remotas” en la página 1276](#)
- [“Colocar mensajes directamente en las colas de transmisión” en la página 1276](#)
- [“Obtener mensajes de las colas de transmisión” en la página 1276](#)
- [“Campos” en la página 1276](#)
- [“Valores iniciales” en la página 1277](#)
- [“Declaración RPG” en la página 1278](#)

Campos en el descriptor de mensaje independiente

Los campos del descriptor de mensajes independiente son establecidos por el gestor de colas, tal y como se muestra en la siguiente lista. Si el gestor de colas no admite el MQMD « version-2 », se utiliza un MQMD « version-1 » sin pérdida de funcionalidad.

Tabla 779. Campos en el descriptor de mensaje separado y valores utilizados

Campo en MQMD separado	Valor utilizado
MDSID	MDSIDV
MDVER	MDVER2
MDREP	Copiado del descriptor de mensaje incrustado, pero con los bits identificados por ROAUXM establecidos en cero. (Esto evita que se genere un mensaje de informe COA o COD cuando se coloca o se elimina un mensaje de una cola de transmisión)
MDMT	Copiado del descriptor de mensaje incrustado.
MDEXP	Copiado del descriptor de mensaje incrustado.
MDFB	Copiado del descriptor de mensaje incrustado.
MDENC	ENNAT
MDCSI	Atributo CodedCharSetId del gestor de colas.
MDFMT	FMXQH
MDPRI	Copiado del descriptor de mensaje incrustado.
MDPER	Copiado del descriptor de mensaje incrustado.
MDMID	El gestor de colas genera un nuevo valor. Este identificador de mensaje es diferente del <i>MDMID</i> que el gestor de colas puede haber generado para el descriptor de mensaje incrustado (véase descrito anteriormente).
MDCID	El <i>MDMID</i> del descriptor de mensaje incrustado.
MDBOC	0
MDRQ	Copiado del descriptor de mensaje incrustado.

Tabla 779. Campos en el descriptor de mensaje separado y valores utilizados (continuación)

Campo en MQMD separado	Valor utilizado
<i>MDRM</i>	Copiado del descriptor de mensaje incrustado.
<i>MDUID</i>	Copiado del descriptor de mensaje incrustado.
<i>MDACC</i>	Copiado del descriptor de mensaje incrustado.
<i>MDAID</i>	Copiado del descriptor de mensaje incrustado.
<i>MDPAT</i>	ATQM
<i>MDPAN</i>	Los primeros 28 bytes del nombre del gestor de colas.
<i>MDPD</i>	Fecha en la que el mensaje se colocó en la cola de transmisión.
<i>MDPT</i>	Hora en la que el mensaje se colocó en la cola de transmisión.
<i>MDAOD</i>	Espacios en blanco
<i>MDGID</i>	GINONE
<i>MDSEQ</i>	1
<i>MDOFF</i>	0
<i>MDMFL</i>	MFNONE
<i>MDOLN</i>	OLUNDF

Campos del descriptor de mensaje incrustado

Los campos del descriptor de mensaje incrustado tienen los mismos valores que los del **MSGDSC** parámetro de la llamada MQPUT o MQPUT1, excepto en los siguientes casos:

- El *MDVER* campo siempre tiene el valor « MDVER1 ».
- Si el *MDPRI* campo tiene el valor PRQDEF, se sustituye por el valor del atributo **DefPriority** de la cola.
- Si el *MDPER* campo tiene el valor PEQDEF, se sustituye por el valor del atributo **DefPersistence** de la cola.
- Si el *MDMID* campo tiene el valor MINONE, o se ha especificado la opción PMNMID, o el mensaje es un mensaje de lista de distribución, *MDMID* se sustituye por un nuevo identificador de mensaje generado por el gestor de colas.

Cuando un mensaje de lista de distribución se divide en mensajes de lista de distribución más pequeños colocados en diferentes colas de transmisión, el *MDMID* campo de cada uno de los nuevos descriptores de mensajes incrustados es el mismo que el del mensaje de lista de distribución original.

- Si se especificó la opción PMNCID, *MDCID* se sustituye por un nuevo identificador de correlación generado por el gestor de colas.
- Los campos de contexto se establecen según lo indicado por las opciones PM* especificadas en el **PMO** parámetro; los campos de contexto son:

- *MDACC*
- *MDAID*
- *MDAOD*
- *MDPAN*
- *MDPAT*
- *MDPD*
- *MDPT*
- *MDUID*

- Los campos « version-2 » (si estaban presentes) se eliminan del MQMD y se trasladan a una estructura MQMDE, si uno o varios de los campos « version-2 » tienen un valor distinto al predeterminado.

Transferir mensajes a colas remotas

: Cuando una aplicación coloca un mensaje en una cola remota (ya sea especificando directamente el nombre de la cola remota o utilizando una definición local de la cola remota), el gestor de colas local:

- Crea una estructura MQXQH que contiene el descriptor de mensaje incrustado
- Añade un MQMDE si es necesario y aún no existe
- Añade los datos del mensaje de la aplicación
- Coloca el mensaje en una cola de transmisión adecuada

Colocar mensajes directamente en las colas de transmisión

También es posible que una aplicación coloque un mensaje directamente en una cola de transmisión. En este caso, la aplicación debe anteponer los datos del mensaje de la aplicación con una estructura MQXQH e inicializar los campos con los valores adecuados. Además, el *MDFMT* campo del **MSGDSC** parámetro de la llamada MQPUT o MQPUT1 debe tener el valor FMXQH.

Los datos de caracteres de la estructura MQXQH creada por la aplicación deben estar en el juego de caracteres del gestor de colas local (definido por el **CodedCharSetId** atributo queue manager), y los datos enteros deben estar en la codificación nativa de la máquina. Además, los datos de caracteres en la estructura MQXQH deben rellenarse con espacios en blanco hasta alcanzar la longitud definida del campo; los datos no deben terminar prematuramente utilizando un carácter nulo, ya que el gestor de colas no convierte el carácter nulo y los caracteres siguientes en espacios en blanco en la estructura MQXQH.

Sin embargo, tenga en cuenta que el gestor de colas no comprueba si existe una estructura MQXQH ni si se han especificado valores válidos para los campos.

Obtener mensajes de las colas de transmisión

Las aplicaciones que reciben mensajes de una cola de transmisión deben procesar la información de la estructura MQXQH de forma adecuada. La presencia de la estructura MQXQH al principio de los datos del mensaje de aplicación se indica mediante el valor FMXQH que se devuelve en el *MDFMT* campo del **MSGDSC** parámetro de la llamada MQGET. Los valores devueltos en los campos *MDCSI MDENC* y del **MSGDSC** parámetro indican el juego de caracteres y la codificación de los datos de caracteres e enteros en la estructura MQXQH. El conjunto de caracteres y la codificación de los datos del mensaje de la aplicación se definen mediante los campos *MDCSI MDENC* y del descriptor de mensajes incrustado.

Campos

La estructura MQXQH contiene los siguientes campos, que se describen en **orden alfabético** :

XQMD (MQMD1)

Descriptor del mensaje original.

Este es el descriptor de mensaje incrustado, y es una copia casi exacta del descriptor de mensaje MQMD que se especificó como parámetro **MSGDSC** en la llamada MQPUT o MQPUT1 cuando el mensaje se colocó originalmente en la cola remota.

Nota: Este es un MQMD de version-1.

Los valores iniciales de los campos de esta estructura son los mismos que los de la estructura MQMD.

XQRQ (cadena de caracteres de 48 bytes)

Nombre de la cola de destino.

Este es el nombre de la cola de mensajes que es el destino final aparente del mensaje (esto puede no ser el destino final real si, por ejemplo, esta cola se define en *XQRQM* como una definición local de otra cola remota).

Si el mensaje es un mensaje de lista de distribución (es decir, el *MDFMT* campo del descriptor de mensaje incrustado es *FMDH*), *XQRQ* está en blanco.

La longitud de este campo viene dada por *LNQN*. El valor inicial de este campo es 48 caracteres en blanco.

XQRQM (cadena de caracteres de 48 bytes)

Nombre del gestor de colas de destino.

Este es el nombre del administrador de colas o del grupo de uso compartido de colas propietario de la cola que es el destino final aparente del mensaje.

Si el mensaje es un mensaje de lista de distribución, *XQRQM* está en blanco.

La longitud de este campo viene dada por *LNQMN*. El valor inicial de este campo es 48 caracteres en blanco.

XQSID (cadena de caracteres de 4 bytes)

Identificador de estructura.

El valor debe ser:

XQSIDV

Identificador para la estructura de encabezado de la cola de transmisión.

El valor inicial de este campo es *XQSIDV*.

XQVER (entero con signo de 10 dígitos)

Número de versión de la estructura.

El valor debe ser:

XQVER1

Número de versión para la estructura del encabezado de la cola de transmisión.

La siguiente constante especifica el número de versión de la versión actual:

XQVERC

Versión actual de la estructura del encabezado de la cola de transmisión.

El valor inicial de este campo es *XQVER1*.

Valores iniciales

Tabla 780. Campos en MQXQH		
Nombre de campo	Nombre de la constante	Valor de la constante
<i>XQSID</i>	<i>XQSIDV</i>	'XQH¬'
<i>XQVER</i>	<i>XQVER1</i>	1
<i>XQRQ</i>	Ninguna	Espacios en blanco
<i>XQRQM</i>	Ninguna	Espacios en blanco
<i>XQMD</i>	Mismos nombres y valores que <i>MQMD</i> ; véase Tabla 748 en la página 1175	-
Notas: 1. El símbolo ¬ representa un único carácter en blanco.		

Declaración RPG

```
D*..1.....2.....3.....4.....5.....6.....7..
D*
D* MQXQH Structure
D*
D* Structure identifier
D  XQSID          1          4      INZ('XQH ')
D* Structure version number
D  XQVER          5          8I 0    INZ(1)
D* Name of destination queue
D  XQRQ           9          56      INZ
D* Name of destination queue manager
D  XQRQM          57         104      INZ
D* Original message descriptor
D  XQ1SID         105        108      INZ('MD ')
D  XQ1VER         109        112I 0    INZ(1)
D  XQ1REP         113        116I 0    INZ(0)
D  XQ1MT          117        120I 0    INZ(8)
D  XQ1EXP         121        124I 0    INZ(-1)
D  XQ1FB          125        128I 0    INZ(0)
D  XQ1ENC         129        132I 0    INZ(273)
D  XQ1CSI         133        136I 0    INZ(0)
D  XQ1FMT         137        144      INZ(' ')
D  XQ1PRI         145        148I 0    INZ(-1)
D  XQ1PER         149        152I 0    INZ(2)
D  XQ1MID         153        176      INZ(X'00000000000000-
D                                     00000000000000000000-
D                                     000000000000')
D  XQ1CID         177        200      INZ(X'00000000000000-
D                                     00000000000000000000-
D                                     000000000000')
D  XQ1BOC         201        204I 0    INZ(0)
D  XQ1RQ         205        252      INZ
D  XQ1RM         253        300      INZ
D  XQ1UID         301        312      INZ
D  XQ1ACC         313        344      INZ(X'00000000000000-
D                                     00000000000000000000-
D                                     00000000000000000000-
D                                     000000')
D  XQ1AID         345        376      INZ
D  XQ1PAT         377        380I 0    INZ(0)
D  XQ1PAN         381        408      INZ
D  XQ1PD         409        416      INZ
D  XQ1PT         417        424      INZ
D  XQ1AOD         425        428      INZ
```

IBM i Llamadas de función en IBM i

Utilice esta información para obtener información sobre las llamadas de función disponibles en la programación de IBM i.

Convenciones utilizadas en las descripciones de llamada en IBM i

Para cada llamada, esta colección de temas proporciona una descripción de los parámetros y el uso de la llamada. Esto va seguido de invocaciones típicas de la llamada, y declaraciones típicas de sus parámetros, en el lenguaje de programación RPG.

Importante: Al codificar llamadas de API de IBM MQ, debe asegurarse de que se proporcionan todos los parámetros relevantes (tal como se describe en las secciones siguientes). No hacerlo puede producir resultados imprevisibles.

La descripción de cada llamada contiene las secciones siguientes:

Nombre de llamada

El nombre de la llamada, seguido de una breve descripción de la finalidad de la llamada.

Parámetros

Para cada parámetro, el nombre va seguido de su tipo de datos entre paréntesis () y su dirección; por ejemplo:

CMPCOD (entero decimal de 9 dígitos)-salida

Hay más información sobre los tipos de datos de estructura en [“Tipos de datos elementales en IBM i” en la página 1016](#).

La dirección del parámetro puede ser:

Entrada

Usted (el programador) debe proporcionar este parámetro.

Salida

La llamada devuelve este parámetro.

Entrada/salida

Debe proporcionar este parámetro, pero la llamada lo modifica.

También hay una breve descripción de la finalidad del parámetro, junto con una lista de los valores que puede tomar el parámetro.

Los dos últimos parámetros de cada llamada son un código de terminación y un código de razón. El código de terminación indica si la llamada se ha completado correctamente, parcialmente o no. En el código de razón se proporciona más información sobre el éxito parcial o el fracaso de la llamada.

Notas de uso

Información adicional sobre la llamada, describiendo cómo utilizarla y cualquier restricción sobre su uso.

Invocación de RPG

Invocación típica de la llamada, y declaración de sus parámetros, en RPG.

Otras convenciones notariales son:

Constantes

Los nombres de constantes se muestran en mayúsculas; por ejemplo, OOOUT.

Matrices

En algunas llamadas, los parámetros son matrices de series de caracteres con un tamaño que no es fijo. En las descripciones de estos parámetros, una *n* en minúsculas representa una constante numérica. Cuando codifique la declaración para ese parámetro, sustituya *n* por el valor numérico que necesita.

MQBACK (Deshacer cambios) activado IBM i

La llamada MQBACK indica al gestor de colas que se deben revertir todas las operaciones de obtención y envío de mensajes que se hayan producido desde el último punto de sincronización. Los mensajes incluidos como parte de una unidad de trabajo se eliminan; los mensajes recuperados como parte de una unidad de trabajo se restablecen en la cola.

- Esta llamada es compatible con los siguientes entornos:

-  AIX
-  IBM i
-  Windows

- [“Sintaxis” en la página 1279](#)
- [“Notas de uso” en la página 1280](#)
- [“Parámetros” en la página 1281](#)
- [“Declaración RPG” en la página 1282](#)

Sintaxis

MQBACK (*Hconn*, *CompCode*, *Reason*)

Notas de uso

Tenga en cuenta estas notas de uso al utilizar MQBACK.

1. Esta llamada solo se puede utilizar cuando el propio gestor de colas coordina la unidad de trabajo. Se trata de una unidad de trabajo local, donde los cambios solo IBM MQ afectan a los recursos.
2. En entornos en los que el gestor de colas no coordina la unidad de trabajo, se debe utilizar la llamada de retroceso adecuada en lugar de MQBACK. El entorno también puede admitir una salida implícita causada por el cierre anormal de la aplicación.
 - En IBM i, esta llamada se puede utilizar para unidades de trabajo locales coordinadas por el gestor de colas. Esto significa que no debe existir una definición de compromiso a nivel de trabajo, es decir, no debe haberse emitido el STRCMTCTL comando con el **CMTSCOPE(*JOB)** parámetro para el trabajo.
3. Si una aplicación finaliza con cambios no confirmados en una unidad de trabajo, la disposición de dichos cambios depende de si la aplicación finaliza de forma normal o anómala. Consulte las notas de uso en [“MQDISC \(Desconectar gestor de colas\) en IBM i”](#) en la página 1319 para obtener más detalles.
4. Cuando una aplicación coloca u obtiene mensajes en grupos o segmentos de mensajes lógicos, el gestor de colas conserva la información relativa al grupo de mensajes y al mensaje lógico de las últimas llamadas MQPUT y MQGET realizadas con éxito. Esta información está asociada al identificador de cola e incluye datos tales como:
 - Los valores de los campos *MDGID*, *MDSEQ*, *MDMFL* *MDOFF*, y en MQMD.
 - Si el mensaje forma parte de una unidad de trabajo.
 - Para la llamada MQPUT: si el mensaje es persistente o no persistente.

El gestor de colas mantiene *tres* conjuntos de información sobre grupos y segmentos, uno para cada uno de los siguientes:

- La última llamada MQPUT exitosa (esto puede formar parte de una unidad de trabajo).
- La última llamada MQGET exitosa que eliminó un mensaje de la cola (esto puede formar parte de una unidad de trabajo).
- La última llamada MQGET exitosa que exploró un mensaje en la cola (esto no puede formar parte de una unidad de trabajo).

Si la aplicación envía o recibe los mensajes como parte de una unidad de trabajo y, a continuación, decide revertir la unidad de trabajo, la información del grupo y del segmento se restaura al valor que tenía anteriormente:

- La información asociada a la llamada MQPUT se restaura al valor que tenía antes de la primera llamada MQPUT correcta para ese identificador de cola en la unidad de trabajo actual.
- La información asociada a la llamada MQGET se restaura al valor que tenía antes de la primera llamada MQGET correcta para ese identificador de cola en la unidad de trabajo actual.

Las colas que fueron actualizadas por la aplicación después de que la unidad de trabajo se hubiera iniciado, pero fuera del ámbito de la unidad de trabajo, no recuperan su información de grupo y segmento si se revierte la unidad de trabajo.

Al restaurar la información del grupo y del segmento a su valor anterior cuando se revierte una unidad de trabajo, la aplicación puede distribuir un grupo de mensajes grande o un mensaje lógico grande compuesto por muchos segmentos entre varias unidades de trabajo, y reiniciar en el punto correcto del grupo de mensajes o del mensaje lógico si falla una de las unidades de trabajo. El uso de varias unidades de trabajo puede resultar ventajoso si el gestor de colas local solo dispone de un almacenamiento de colas limitado. Sin embargo, la aplicación debe conservar suficiente información para poder reiniciar el envío o la recepción de mensajes en el punto correcto si se produce un fallo del sistema. Para obtener más información sobre cómo reiniciar en el punto correcto tras un fallo del sistema, consulte la opción PMLOGO descrita en [“MQPMO \(Opciones de mensajes de envío\) activado IBM i”](#) en la página 1198 y la opción GMLOGO descrita en [“MQGMO \(Opciones de obtención de mensajes\) activado IBM i”](#) en la página 1096.

Las notas de uso restantes solo se aplican cuando el gestor de colas coordina las unidades de trabajo:

1. Una unidad de trabajo tiene el mismo alcance que un identificador de conexión. Esto significa que todas IBM MQ las llamadas que afecten a una unidad de trabajo concreta deben realizarse utilizando el mismo identificador de conexión. Las llamadas realizadas utilizando un identificador de conexión diferente (por ejemplo, llamadas realizadas por otra aplicación) afectan a una unidad de trabajo diferente. Consulte el **HCONN** parámetro descrito en “MQCONN (Conectar gestor de colas) en IBM i” en la página 1306 para obtener información sobre el ámbito de los identificadores de conexión.
2. Solo los mensajes que se colocaron o recuperaron como parte de la unidad de trabajo actual se ven afectados por esta llamada.
3. Una aplicación de larga duración que emite llamadas MQGET, MQPUT o MQPUT1 dentro de una unidad de trabajo, pero que nunca emite una llamada de confirmación o retroceso, puede provocar que las colas se llenen de mensajes que no están disponibles para otras aplicaciones. Para protegerse contra esta posibilidad, el administrador debe establecer el atributo del **MaxUncommittedMsgs** gestor de colas en un valor lo suficientemente bajo como para evitar que las aplicaciones descontroladas llenen las colas, pero lo suficientemente alto como para permitir que las aplicaciones de mensajería esperadas funcionen correctamente.

Parámetros

La llamada MQBACK tiene los siguientes parámetros:

HCONN (entero con signo de 10 dígitos) - entrada

Descriptor de conexión.

Este manejador representa la conexión con el gestor de colas. El valor de *HCONN* fue devuelto por una llamada anterior a MQCONN o MQCONNEX.

CMPCOD (entero con signo de 10 dígitos) - salida

Código de terminación.

Es uno de los siguientes:

CCOK

Realización satisfactoria.

CCFAIL

La llamada no se ha realizado satisfactoriamente.

RAZÓN (entero con signo de 10 dígitos) - salida

Código de motivo calificativo *COMCOD*.

Si *COMCOD* es CCOK:

RCNONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *COMCOD* es CCFail:

RC2219

(2219, X'8AB') Se ha vuelto a introducir la llamada « MQI » antes de que finalizara la llamada anterior.

RC2009

(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

RC2018

(2018, X'7E2') El manejador de conexión no es válido.

RC2101

(2101, X'835') El objeto se ha dañado.

RC2123

(2123, X'84B') El resultado de la operación de confirmación o retroceso es mixto.

RC2162

(2162, X'872') El gestor de colas está concluyendo.

RC2102

(2102, X'836') No hay disponibles suficientes recursos del sistema.

RC2071

(2071, X'817') No hay suficiente almacenamiento disponible.

RC2195

(2195, X'893') Se ha producido un error inesperado.

Declaración RPG

```
C*.1.....2.....3.....4.....5.....6.....7..
C                                CALLP      MQBACK(HCONN : COMCOD : REASON)
```

La definición del prototipo para la llamada es:

```
D*.1.....2.....3.....4.....5.....6.....7..
D*MQBACK                PR              EXTPROC('MQBACK')
D* Connection handle
D HCONN                  10I 0 VALUE
D* Completion code
D COMCOD                 10I 0
D* Reason code qualifying COMCOD
D REASON                 10I 0
```

MQBEGIN (Inicio de la unidad de trabajo) en IBM i

La llamada MQBEGIN inicia una unidad de trabajo coordinada por el gestor de colas, que puede implicar a gestores de recursos externos.

- Esta llamada es compatible con los siguientes entornos:

-  AIX
-  IBM i
-  Windows

- “Sintaxis” en la [página 1282](#)
- “Notas de uso” en la [página 1282](#)
- “Parámetros” en la [página 1283](#)
- “Declaración RPG” en la [página 1285](#)

Sintaxis

MQBEGIN (*HCONN*, *BEGOP*, *CMPCOD*, *REASON*)

Notas de uso

1. La llamada MQBEGIN se puede utilizar para iniciar una unidad de trabajo coordinada por el gestor de colas y que puede implicar cambios en recursos propiedad de otros gestores de recursos. El gestor de colas admite tres tipos de unidades de trabajo:

Unidad de trabajo local coordinada por el gestor de colas

Se trata de una unidad de trabajo en la que el gestor de colas es el único gestor de recursos que participa, por lo que el gestor de colas actúa como coordinador de la unidad de trabajo.

- Para iniciar este tipo de unidad de trabajo, se debe especificar la opción PMSYP o GMSYP en la primera llamada MQPUT, MQPUT1 o MQGET de la unidad de trabajo.

No es necesario que la aplicación emita la llamada MQBEGIN para iniciar la unidad de trabajo, pero si se utiliza MQBEGIN, la llamada se completa con CCWARN y el código de motivo RC2121.

- Para confirmar o cancelar este tipo de unidad de trabajo, se debe utilizar la llamada MQCMIT o MQBACK.

Unidad de trabajo global coordinada por el gestor de colas

Se trata de una unidad de trabajo en la que el gestor de colas actúa como coordinador de la unidad de trabajo, tanto para IBM MQ los recursos *como* para los recursos que pertenecen a otros gestores de recursos. Esos administradores de recursos cooperan con el administrador de colas para garantizar que todos los cambios en los recursos de la unidad de trabajo se confirmen o se deshagan juntos.

- Para iniciar este tipo de unidad de trabajo, se debe utilizar la llamada MQBEGIN.
- Para confirmar o cancelar este tipo de unidad de trabajo, se deben utilizar las llamadas MQCMIT y MQBACK.

Unidad de trabajo global coordinada externamente

Se trata de una unidad de trabajo en la que el gestor de colas es un participante, pero no actúa como coordinador de la unidad de trabajo. En su lugar, hay un coordinador externo de unidades de trabajo con el que coopera el gestor de colas.

- Para iniciar este tipo de unidad de trabajo, se debe utilizar la llamada correspondiente proporcionada por el coordinador externo de la unidad de trabajo.

Si se utiliza la llamada MQBEGIN para intentar iniciar la unidad de trabajo, la llamada falla con el código de motivo RC2012.

- Para confirmar o deshacer este tipo de unidad de trabajo, se deben utilizar las llamadas de confirmación y deshacer proporcionadas por el coordinador externo de unidades de trabajo.

Si se utiliza la llamada MQCMIT o MQBACK para intentar confirmar o deshacer la unidad de trabajo, la llamada falla con el código de motivo RC2012.

2. Si la aplicación finaliza con cambios no confirmados en una unidad de trabajo, la disposición de dichos cambios depende de si la aplicación finaliza de forma normal o anómala. Consulte las notas de uso en [“MQDISC \(Desconectar gestor de colas\) en IBM i”](#) en la página 1319 para obtener más detalles.
3. Una aplicación solo puede participar en una unidad de trabajo a la vez. La llamada MQBEGIN falla con el código de motivo RC2128 si ya existe una unidad de trabajo para la aplicación, independientemente del tipo de unidad de trabajo que sea.
4. La llamada MQBEGIN no es válida en un entorno IBM MQ cliente. El intento de utilizar la llamada falla con el código de motivo RC2012.
5. Cuando el gestor de colas actúa como coordinador de unidades de trabajo para unidades de trabajo globales, los gestores de recursos que pueden participar en la unidad de trabajo se definen en el archivo de configuración del gestor de colas.
6. En IBM i, se admiten los tres tipos de unidades de trabajo de la siguiente manera:
 - **Las unidades de trabajo locales coordinadas por el gestor de colas** solo se pueden utilizar cuando no existe una definición de compromiso a nivel de trabajo, es decir, no se debe haber emitido el STRCMTCTL comando con el **CMTSCOPE(*JOB)** parámetro para el trabajo.
 - No se admiten **unidades de trabajo globales coordinadas por el gestor de colas**.
 - **Las unidades de trabajo globales coordinadas externamente** solo se pueden utilizar cuando existe una definición de compromiso a nivel de trabajo, es decir, se debe haber emitido el STRCMTCTL comando con el **CMTSCOPE(*JOB)** parámetro para el trabajo. Si esto se ha hecho, las IBM iCOMMIT operaciones ROLLBACK y se aplican tanto a IBM MQ los recursos como a los recursos que pertenecen a otros gestores de recursos participantes.

Parámetros

La llamada MQBEGIN tiene los siguientes parámetros:

HCONN (entero con signo de 10 dígitos) - entrada

Descriptor de conexión.

Este manejador representa la conexión con el gestor de colas. El valor de *HCONN* fue devuelto por una llamada anterior a MQCONN o MQCONNX.

BEGOP (MQBO) - entrada/salida

Opciones que controlan la acción de MQBEGIN.

Consulte “MQBO (Opciones de inicio) activado IBM i” en la [página 1036](#) para obtener los detalles.

Si no se requieren opciones, los programas escritos en C o en ensambladore S/390 e pueden especificar una dirección de parámetro nula, en lugar de especificar la dirección de una estructura MQBO.

CMPCOD (entero con signo de 10 dígitos) - salida

Código de terminación.

Es uno de los siguientes:

CCOK

Realización satisfactoria.

CCWARN

Aviso (finalización parcial).

CCFAIL

La llamada no se ha realizado satisfactoriamente.

RAZÓN (entero con signo de 10 dígitos) - salida

Código de motivo calificativo *CMPCOD*.

Si *CMPCOD* es CCOK:

RCNONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CMPCOD* es CCWARN:

RC2121

(2121, X'849') No hay gestores de recursos participantes registrados.

RC2122

(2122, X'84A') Gestor de recursos participante no disponible.

Si *CMPCOD* es CCFail:

RC2134

(2134, X'856') Estructura de opciones de inicio no válida.

RC2219

(2219, X'8AB') Se ha vuelto a introducir la llamada « MQI » antes de que finalizara la llamada anterior.

RC2009

(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

RC2012

(2012, X'7DC') Llamada no válida en el entorno.

RC2018

(2018, X'7E2') El manejador de conexión no es válido.

RC2046

(2046, X'7FE') Las opciones no son válidas o no son coherentes.

RC2162

(2162, X'872') El gestor de colas está concluyendo.

RC2102

(2102, X'836') No hay disponibles suficientes recursos del sistema.

RC2071

(2071, X'817') No hay suficiente almacenamiento disponible.

RC2195

(2195, X'893') Se ha producido un error inesperado.

RC2128

(2128, X'850') Unidad de trabajo ya iniciada.

Declaración RPG

```
C*..1.....2.....3.....4.....5.....6.....7..
C                                CALLP      MQBEGIN(HCONN : BEGOP : CMPCOD :
C                                REASON)
```

La definición del prototipo para la llamada es:

```
D*..1.....2.....3.....4.....5.....6.....7..
D*MQBEGIN          PR          EXTPROC('MQBEGIN')
D* Connection handle
D HCONN            10I 0 VALUE
D* Options that control the action of MQBEGIN
D BEGOP            12A
D* Completion code
D CMPCOD            10I 0
D* Reason code qualifying CMPCOD
D REASON            10I 0
```

MQBUFMH (Convertir almacenamiento intermedio en descriptor de mensaje) en IBM i

La llamada de función MQBUFMH convierte un almacenamiento intermedio en un manejador de mensajes y es el inverso de la llamada MQMHBUF.

Esta llamada toma un descriptor de mensaje y las propiedades MQRFH2 del almacenamiento intermedio y las hace disponibles a través de un descriptor de mensaje. Las propiedades MQRFH2 de los datos del mensaje se eliminan, opcionalmente. Los campos *Encoding*, *CodedCharSetIdy Format* del descriptor de mensaje se actualizan, si es necesario, para describir correctamente el contenido del almacenamiento intermedio después de que se hayan eliminado las propiedades.

- [“Sintaxis” en la página 1285](#)
- [“Notas de uso” en la página 1285](#)
- [“Parámetros” en la página 1285](#)
- [“Declaración RPG” en la página 1287](#)

Sintaxis

MQBUFMH (*Hconn*, *Hmsg*, *BufMsgHOpts*, *MsgDesc*, *Buffer*, *BufferLength*, *DataLength*, *CompCode*, *Reason*)

Notas de uso

Las llamadas MQBUFMH no se pueden interceptar mediante salidas de API: un almacenamiento intermedio se convierte en un manejador de mensajes en el espacio de aplicación; la llamada no llega al gestor de colas.

Parámetros

La llamada MQBUFMH tiene los parámetros siguientes:

HCONN (entero con signo de 10 dígitos)-entrada

Este manejador representa la conexión con el gestor de colas. El valor de *HCONN* debe coincidir con el descriptor de conexión que se ha utilizado para crear el descriptor de mensaje especificado en el parámetro **Hmsg**.

Si el descriptor de mensaje se ha creado utilizando HCUNAS, se debe establecer una conexión válida en la hebra convirtiendo un almacenamiento intermedio en un descriptor de mensaje. Si no se establece una conexión válida, la llamada falla con RC2009.

HMSG (entero con signo de 20 dígitos)-entrada

Este descriptor de contexto es el descriptor de contexto de mensaje para el que se necesita un almacenamiento intermedio. El valor ha sido devuelto por una llamada MQCRTMH anterior.

BMHOPT (MQBMHO)-entrada

La estructura MQBMHO permite a las aplicaciones especificar opciones que controlan cómo se generan los manejadores de mensajes a partir de los almacenamientos intermedios.

Consulte “MQBMHO (Opciones de búfer para el controlador de mensajes) activado IBM i” en la [página 1035](#) para obtener los detalles.

MSGDSC (MQMD)-entrada/salida

La estructura *MSGDSC* contiene las propiedades del descriptor de mensaje y describe el contenido del área de almacenamiento intermedio.

En la salida de la llamada, las propiedades se eliminan opcionalmente del área de almacenamiento intermedio y, en este caso, el descriptor de mensaje se actualiza para describir correctamente el área de almacenamiento intermedio.

Los datos de esta estructura deben estar en el juego de caracteres y la codificación de la aplicación.

BUFLEN (entero con signo de 10 dígitos)-entrada

BUFLEN es la longitud del área de almacenamiento intermedio, en bytes.

Un *BUFLEN* de cero bytes es válido e indica que el área de almacenamiento intermedio no contiene datos.

BUFFER (serie de bits de 1 byte x BUFLEN)-entrada/salida

BUFFER define el área que contiene el almacenamiento intermedio de mensajes. Para la mayoría de los datos, debe alinear el almacenamiento intermedio en un límite de 4 bytes.

Si *BUFFER* contiene datos numéricos o de caracteres, establezca los campos *CodedCharSetId* y *Encoding* del parámetro **MSGDSC** en los valores adecuados para los datos; esto permite convertir los datos, si es necesario.

Si se encuentran propiedades en el almacenamiento intermedio de mensajes, se eliminan de forma opcional; posteriormente pasan a estar disponibles desde el descriptor de contexto de mensaje al devolver la llamada.

En el lenguaje de programación C, el parámetro se declara como un puntero a void, lo que significa que la dirección de cualquier tipo de datos se puede especificar como parámetro.

Si el parámetro **BUFLEN** es cero, no se hace referencia a *BUFFER*. En este caso, la dirección de parámetro pasada por los programas escritos en C o System/390 assembler puede ser nula.

DATLEN (entero con signo de 10 dígitos)-salida

DATLEN es la longitud, en bytes, del almacenamiento intermedio que puede tener las propiedades eliminadas.

CMPCOD (entero con signo de 10 dígitos)-salida

CCOK

Realización satisfactoria.

CCFAIL

La llamada no se ha realizado satisfactoriamente.

REASON (entero con signo de 10 dígitos)-salida

El código de razón que califica *CMPCOD*.

Si *CMPCOD* es CCOK:

RCNONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CMPCOD* es CCFail:

RC2204

(2204, X'089C') Adaptador no disponible.

RC2130

(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

RC2157

(2157, X'86D') Los ASID primario e inicial varían.

RC2489

(2489, X'09B9') La estructura de opciones del búfer al manejador de mensajes no es válida.

RC2004

(2004, X'07D4') El parámetro del búfer no es válido.

RC2005

(2005, X'07D5') El parámetro de longitud del búfer no es válido.

RC2219

(2219, X'08AB') MQI : se ha iniciado una llamada antes de que finalizara la anterior.

RC2009

(2009, X'07D9') Se ha perdido la conexión con el gestor de colas.

RC2460

(2460, X'099C') Manejador de mensaje no válido.

RC2026

(2026, X'07EA') Descriptor de mensaje no válido.

RC2499

(2499, X'09C3') Manejador de mensaje ya en uso.

RC2046

(2046, X'07FE') Opciones no válidas o no coherentes.

RC2334

(2334, X'091E') Estructura MQRFH2 no válida.

RC2421

(2421, X'0975 ') No se ha podido analizar una carpeta MQRFH2 que contiene propiedades.

RC2195

(2195, X'893') Se ha producido un error inesperado.

Declaración RPG

```

C*..1.....2.....3.....4.....5.....6.....7..
C                                CALLP      MQBUFMH(HCONN : HMSG : BMHOPT :
                                MSGDSC : BUFLN : BUFFER :
                                DATLEN : CMPCOD : REASON)

```

La definición de prototipo para la llamada es:

```

DMQBUFMH      PR      EXTPROC('MQBUFMH')
D* Connection handle
D HCONN      10I 0
D* Message handle

```

```

D HMSG                                10I 0
D* Options that control the action of MQBUFMH
D BMHOPT                             12A  VALUE
D* Message descriptor
D MSGDSC                             364A
D* Length in bytes of the Buffer area
D BUFLN                               10I 0
D* Area to contain the message buffer
D BUFFER                             *  VALUE
D* Length of the output buffer
D DATLN                               10I 0
D* Completion code
D CMPCOD                             10I 0
D* Reason code qualifying CompCode
D REASON                             10I 0

```

MQCB (Gestionar devolución de llamada) activado IBM i

La llamada MQCB vuelve a registrar una devolución de llamada para el identificador de objeto especificado y controla la activación y los cambios en la devolución de llamada.

Una devolución de llamada es un fragmento de código (especificado como el nombre de una función que se puede vincular dinámicamente o como un puntero de función) que se invoca cuando IBM MQ se producen determinados eventos.

Para utilizar MQCB y MQCTL en un cliente V7, debe estar conectado a un servidor V7 y el **SHARECNV** parámetro del canal debe tener un valor distinto de cero.

Los tipos de devolución de llamada que se pueden definir son:

Consumidor de mensajes

La función de devolución de llamada del consumidor de mensajes se invoca cuando hay un mensaje disponible en un identificador de objeto que cumple los criterios de selección especificados.

Solo se puede registrar una función de devolución de llamada para cada identificador de objeto. Si se va a leer una sola cola con múltiples criterios de selección, entonces la cola debe abrirse varias veces y debe registrarse una función de consumo en cada controlador.

Manejador de sucesos

El controlador de eventos se invoca para condiciones que afectan a todo el entorno de devolución de llamada.

La función se invoca cuando se produce una condición de evento, por ejemplo, cuando se detiene o se suspende un gestor de colas o una conexión.

La función no se invoca para condiciones específicas de un único consumidor de mensajes, por ejemplo, RC2016;. Sin embargo, se invoca si una función de devolución de llamada no finaliza normalmente.

- [“Sintaxis” en la página 1288](#)
- [“Notas de uso para MQCB” en la página 1288](#)
- [“Parámetros para MQCB” en la página 1290](#)
- [“Declaración RPG” en la página 1296](#)

Sintaxis

MQCB (*HCONN, OPERATN, HOBJ, CBDSC, MSGDSC, GMO, CMPCOD, REASON*)

Notas de uso para MQCB

1. MQCB se utiliza para definir la acción que se invocará para cada mensaje, que coincida con los criterios especificados, disponible en la cola. Cuando se procesa la acción, el mensaje se elimina de la cola y se pasa al consumidor de mensajes definido, o se proporciona un token de mensaje, que se utiliza para recuperar el mensaje.

2. MQCB se puede utilizar para definir rutinas de devolución de llamada antes de iniciar el consumo con MQCTL o se puede utilizar desde dentro de una rutina de devolución de llamada.
3. Para utilizar MQCB desde fuera de una rutina de devolución de llamada, primero debe suspender el consumo de mensajes mediante MQCTL y reanudarlo posteriormente.

Secuencia de devolución de llamada del consumidor de mensajes

Puede configurar un consumidor para que invoque una devolución de llamada en puntos clave durante el ciclo de vida del consumidor. Por ejemplo:

- cuando el consumidor se registra por primera vez,
- cuando se inicia la conexión,
- cuando se interrumpe la conexión y
- cuando el consumidor se da de baja, ya sea de forma explícita o implícita mediante un MQCLOSE.

Tabla 781. Definiciones de los verbos MQCTL	
Verbo	Significado
MQCTL(INICIO)	Llamada MQCTL mediante la operación CTLSR
MQCTL (STOP)	Llamada MQCTL mediante la operación CTLSP
MQCTL(WAIT)	Llamada MQCTL mediante la operación CTLSW

Permite al consumidor mantener el estado asociado al consumidor. Cuando una aplicación solicita una devolución de llamada, las reglas para la invocación del consumidor son las siguientes:

REGISTRAR

Siempre es el primer tipo de invocación de la devolución de llamada.

Siempre se invoca en el mismo subproceso que la invocación MQCB(CBREG).

START

Siempre se llama de forma sincrónica con el verbo MQCTL(START).

- Todas las devoluciones de llamada START se completan antes de que devuelva el verbo MQCTL(START).

Está en el mismo hilo que la entrega del mensaje si se solicita CTLTHR.

La llamada con start no está garantizada si, por ejemplo, una devolución de llamada anterior emite MQCTL(STOP) durante MQCTL(START).

STOP

Después de esta llamada, no se entregarán más mensajes ni eventos hasta que se reinicie la conexión.

Se garantiza una parada (STOP) si la aplicación fue llamada previamente para un inicio (START), un mensaje o un evento.

DEREGISTER

Siempre es el último tipo de invocación de la devolución de llamada.

Asegúrese de que su aplicación realice la inicialización y limpieza basadas en subprocesos en las devoluciones de llamada START y STOP. Puede realizar la inicialización y limpieza sin hilos con las devoluciones de llamada REGISTER y DEREGISTER.

No haga ninguna suposición sobre la vida útil y la disponibilidad del hilo que no sea lo que se indica. Por ejemplo, no confíe en que un subproceso permanezca activo más allá de la última llamada a DEREGISTER. Del mismo modo, cuando haya optado por no utilizar CTLTHR, no dé por sentado que el subproceso existe cada vez que se inicia la conexión.

Si su aplicación tiene requisitos específicos en cuanto a las características del subproceso, siempre puede crear un subproceso acorde con ellos y, a continuación, utilizar MQCTL(WAIT). Este paso *devuelve* el hilo a IBM MQ para la entrega asíncrona de mensajes.

Uso de la conexión del consumidor de mensajes

Normalmente, cuando una aplicación realiza otra llamada `MQI` mientras hay una pendiente, la llamada falla con el código de motivo «RC2219».

Sin embargo, hay casos especiales en los que la aplicación debe realizar una nueva llamada a `MQI` antes de que haya finalizado la llamada anterior. Por ejemplo, el consumidor puede invocarse durante una llamada MQCB con CBRE.

En tal caso, cuando, como consecuencia de la emisión por parte de la aplicación de un verbo MQCB o MQCTL, se vuelve a llamar a la aplicación, se le permite realizar una nueva llamada `MQI`. Esta instancia significa que puede emitir, por ejemplo, una llamada MQOPEN, en la función de consumidor cuando se llama con un tipo CBCCALLT de CBCTRC. Se permite cualquier llamada a `MQI`, excepto `MQDISC`.

Parámetros para MQCB

La llamada MQCB tiene los siguientes parámetros:

HCONN (entero con signo de 10 dígitos) - entrada

Gestionar función de devolución de llamada: parámetro HCONN.

Este manejador representa la conexión con el gestor de colas. El valor de *HCONN* fue devuelto por una llamada anterior a MQCONN o MQCONNX.

OPERATN (entero con signo de 10 dígitos) - entrada

Gestionar función de devolución de llamada: parámetro OPERATN.

La operación que se está procesando en la devolución de llamada definida para el identificador de objeto especificado. Debe especificar una de las siguientes opciones; si se requiere más de una opción, los valores se pueden añadir (no añada la misma constante más de una vez) o combinar mediante la operación OR bit a bit (si el lenguaje de programación admite operaciones bit a bit).

Las combinaciones que no son válidas se indican; todas las demás combinaciones son válidas.

CBREG

Defina la función de devolución de llamada para el identificador de objeto especificado. Esta operación define la función que se va a llamar y los criterios de selección que se van a utilizar.

Si ya se ha definido una función de devolución de llamada para el identificador del objeto, se sustituye la definición. Si se detecta un error al reemplazar la devolución de llamada, la función se da de baja.

Si se registra una devolución de llamada en la misma función de devolución de llamada en la que se había dado de baja anteriormente, esto se trata como una operación de sustitución; no se invocan las llamadas iniciales ni finales.

Puede utilizar CBREG con CTLSU o CTLRE.

CBUNR

Detiene el consumo de mensajes para el identificador del objeto y elimina el identificador de los que son elegibles para una devolución de llamada.

Una devolución de llamada se da de baja automáticamente si se cierra el identificador asociado.

Si se llama a CBUNR desde dentro de un consumidor y la devolución de llamada tiene definida una llamada de detención, se invoca al regresar del consumidor.

Si esta operación se emite contra un *Hobj* sin consumidor registrado, la llamada devuelve un error RC2448.

CTLSU

Suspende el consumo de mensajes para el identificador del objeto.

Si esta operación se aplica a un controlador de eventos, el controlador de eventos no recibe eventos mientras está suspendido, y los eventos perdidos mientras se encuentra en estado suspendido no se proporcionan a la operación cuando se reanuda.

Mientras está suspendida, la función de consumo sigue recibiendo las devoluciones de llamada del tipo de control.

CTLRE

Reanuda el consumo de mensajes para el identificador del objeto.

Si esta operación se aplica a un controlador de eventos, el controlador de eventos no recibe eventos mientras está suspendido, y los eventos perdidos mientras se encuentra en estado suspendido no se proporcionan a la operación cuando se reanuda.

CBDSC (MQCBD) - entrada

Gestionar función de devolución de llamada - Parámetro CBDSC.

Esta es una estructura que identifica la función de devolución de llamada que está registrando la aplicación y las opciones utilizadas al registrarla.

Consulte “MQCBD - Descriptor de devolución de llamada” en la página 236 para obtener más información sobre la estructura.

El descriptor de devolución de llamada solo es necesario para la opción CBREG; si el descriptor no es necesario, la dirección del parámetro pasada puede ser nula.

HOBJ (entero con signo de 10 dígitos) - entrada

Gestionar función de devolución de llamada: parámetro HOBJ.

Este identificador representa el acceso que se ha establecido al objeto desde el que se va a consumir un mensaje. Se trata de un identificador que ha sido devuelto por una llamada anterior a MQOPEN o MQSUB (en el **HOBJ** parámetro).

HOBJ No es necesario al definir una rutina de controlador de eventos (CBTEH) y debe especificarse como HONONE.

Si se ha *Hobj* devuelto esto desde una llamada MQOPEN, la cola debe haberse abierto con una o varias de las siguientes opciones:

- OOINPS
- OOINPX
- OOINPQ
- OOBROW

MSGDSC (MQMD) - entrada

Gestionar función de devolución de llamada: parámetro MSGDSC.

Esta estructura describe los atributos del mensaje necesario y los del mensaje recuperado.

El **MsgDesc** parámetro define los atributos de los mensajes requeridos por el consumidor y la versión del MQMD que se pasará al consumidor del mensaje.

Los *MsgId*, *CorrelId*, *GroupId*, *MsgSeqNumber*, y *Offset* del MQMD se utilizan para la selección de mensajes, dependiendo de las opciones especificadas en el **GetMsgOpts** parámetro.

Se utilizan *Encoding CodedCharSetId* y para la conversión de mensajes si se especifica la opción GMCONV.

Consulte MQMD para obtener más detalles.

MsgDesc Se utiliza solo para CBREG y, si necesita valores distintos de los predeterminados para cualquier campo. *MsgDesc* no se utiliza para un controlador de eventos.

Si el descriptor no es necesario, la dirección del parámetro pasada puede ser nula.

Tenga en cuenta que si hay varios consumidores registrados en la misma cola con selectores que se superponen, el consumidor elegido para cada mensaje es indefinido.

OGM (MQGMO) - entrada

Gestionar función de devolución de llamada - Parámetro GMO.

Opciones que controlan cómo el consumidor de mensajes recibe los mensajes.

Todas las opciones tienen el significado descrito en [“MQGMO \(Opciones de obtención de mensajes\) activado IBM i”](#) en la página 1096, cuando se utilizan en una llamada MQGET, excepto:

GMSSIG

Esta opción no está permitida.

GMBRWF, GMBRWN, GMMBH, GMMBC

El orden de los mensajes entregados a un consumidor que navega viene dictado por las combinaciones de estas opciones. Las combinaciones significativas son:

GMBRWF

El primer mensaje de la cola se entrega repetidamente al consumidor. Esto resulta útil cuando el consumidor consume de forma destructiva el mensaje en la devolución de llamada. Utilice esta opción con precaución.

GMBRWN

El consumidor recibe cada mensaje de la cola, desde la posición actual del cursor hasta llegar al final de la cola.

GMBRWF + GMBRWN

El cursor se restablece al inicio de la cola. A continuación, se envía cada mensaje al consumidor hasta que el cursor llega al final de la cola.

GMBRWF + GMMBH o GMMBC

Comenzando por el principio de la cola, se le da al consumidor el primer mensaje no marcado de la cola, que luego se marca para este consumidor. Esta combinación garantiza que el consumidor pueda recibir nuevos mensajes añadidos detrás del punto actual del cursor.

GMBRWN + GMMBH o GMMBC

A partir de la posición del cursor, se le da al consumidor el siguiente mensaje no marcado de la cola, que luego se marca para este consumidor. Utilice esta combinación con precaución, ya que los mensajes se pueden añadir a la cola detrás de la posición actual del cursor.

GMBRWF + GMBRWN + GMMBH o GMMBC

Esta combinación no está permitida; si se utiliza, la llamada devuelve un error RC2046.

GMNWT, GMWT y GMWI

Estas opciones controlan cómo se invoca al consumidor.

GMNWT

El consumidor nunca recibe llamadas des RC2033 es. El consumidor solo se invoca para mensajes y eventos

GMWT con un GMWI cero

El código « RC2033 » solo se transmite al consumidor cuando no hay mensajes y

- El consumidor ha sido iniciado
- Se ha entregado al consumidor al menos un mensaje desde el último código de motivo «sin mensajes».

Esto evita que el consumidor realice sondeos en un bucle ocupado cuando se especifica un intervalo de espera cero.

GMWT y un GMWI positivo

El usuario es invocado tras el intervalo de espera especificado con el código de motivo « RC2033 ». Esta llamada se realiza independientemente de si se han entregado mensajes al consumidor. Esto permite al usuario realizar procesamiento de tipo heartbeat o por lotes.

GMWT y GMWI de WIULIM

Esto especifica una espera infinita antes de devolver un RC2033 e. El consumidor nunca recibe llamadas des RC2033 es.

GMO Se utiliza solo para CBREG y, si necesita valores distintos de los predeterminados para cualquier campo. *GMO* no se utiliza para un controlador de eventos.

Si las opciones no son necesarias, la dirección del parámetro pasada puede ser nula.

Si se proporciona un identificador de propiedades de mensaje en la estructura MQGMO, se proporciona una copia en la estructura MQGMO que se pasa a la devolución de llamada del consumidor. Al regresar de la llamada MQCB, la aplicación puede eliminar el identificador de propiedades del mensaje.

CMPCOD (entero con signo de 10 dígitos) - salida

Gestionar función de devolución de llamada: parámetro CMPCOD.

El código de terminación; es uno de los siguientes:

CCOK

Realización satisfactoria.

CCWARN

Aviso (finalización parcial).

CCFAIL

La llamada no se ha realizado satisfactoriamente.

RAZÓN (entero con signo de 10 dígitos) - salida

Gestionar función de devolución de llamada: parámetro REASON.

Los siguientes códigos de motivo son los códigos que el gestor de colas puede devolver para el **REASON** parámetro.

Si *CMPCOD* es CCOK:

RCNONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es CCFail:

RC2204

(2204, X'89C') El adaptador no está disponible.

RC2133

(2133, X'855') No se han podido cargar módulos de servicio de conversión de datos.

RC2130

(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

RC2374

(2374, X'946') La salida de la API ha fallado.

RC2183

(2183, X'887') No se ha podido cargar la salida de la API.

RC2157

(2157, X'86D') Los ASID primario e inicial varían.

RC2005

(2005, X'7D5') El parámetro de longitud de almacenamiento intermedio no es válido.

RC2219

(2219, X'8AB') La llamada MQI ha entrado antes de completar la llamada anterior.

RC2487

(2487, X'9B7') Campo de tipo de devolución de llamada incorrecto.

RC2448

(2448, X'990') No se puede dar de baja, suspender o reanudar porque no hay ninguna devolución de llamada registrada.

RC2486

(2486, X'9B6') Se debe *CallbackFunctionCallbackName* especificar o, pero no ambos.

RC2483

(2483, X'9B3') Campo de tipo de devolución de llamada incorrecto.

RC2484

(2484, X'9B4') Campo de opciones MQCBD incorrecto.

RC2140

(2140, X'85C') Solicitud de espera rechazada por CICS.

RC2009

(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

RC2217

(2217, X'8A9') No tiene autorización para la conexión.

RC2202

(2202, X'89A') Conexión en fase de inmovilización.

RC2203

(2203, X'89B') La conexión está concluyendo.

RC2207

(2207, X'89F') Error de identificador de correlación.

RC2010

(2010, X'7DA') El parámetro longitud de datos no es válido.

RC2016

(2016, X'7E0') Se han inhibido las obtenciones para la cola.

RC2351

(2351, X'92F') Conflicto de unidades de trabajo global.

RC2186

(2186, X'88A') No es válida la estructura de las opciones de obtención de mensaje.

RC2353

(2353, X'931') Descriptor de contexto que se utiliza para la unidad de trabajo global.

RC2018

(2018, X'7E2') El manejador de conexión no es válido.

RC2019

(2019, X'7E3') El manejador de objeto no es válido.

RC2259

(2259, X'8D3') La especificación de examinar es incoherente.

RC2245

(2245, X'8C5') Especificación incoherente de unidad de trabajo.

RC2246

(2246, X'8C6') El mensaje bajo cursor no es válido para recuperación.

RC2352

(2352, X'930') La unidad de trabajo global entra en conflicto con la unidad de trabajo local.

RC2247

(2247, X'8C7') Las opciones de coincidencia no son válidas.

RC2485

(2485, X'9B4') Campo *MaxMsgLength* incorrecto.

RC2026

(2026, X'7EA') El descriptor de mensaje no es válido.

RC2497

(2497, X'9C1') No se ha encontrado el punto de entrada de la función especificada en el módulo.

RC2496

(2496, X'9C0') Se ha encontrado el módulo, pero es del tipo incorrecto; no es de 32 bits, 64 bits ni una biblioteca de enlaces dinámicos válida.

RC2495

(2495, X'9BF') Módulo no encontrado en la ruta de búsqueda o sin autorización para cargarse.

RC2250

(2250, X'8CA') El número de secuencia de mensaje no es válido.

RC2331

(2331, X'91B') El uso del símbolo de mensaje no es válido.

RC2033

(2033, X'7F1') No hay ningún mensaje disponible.

RC2034

(2034, X'7F2') El cursor para examinar no está situado en el mensaje.

RC2036

(2036, X'7F4') La cola no se ha abierto para examen.

RC2037

(2037, X'7F5') La cola no se ha abierto para entrada.

RC2041

(2041, X'7F9') La definición de objeto ha cambiado desde que se ha abierto.

RC2101

(2101, X'835') El objeto se ha dañado.

RC2206

(2206, X'89E') Código de operación incorrecto en la llamada a la API.

RC2046

(2046, X'7FE') Las opciones no son válidas o no son coherentes.

RC2193

(2193, X'891') Error al acceder al conjunto de datos del conjunto de páginas.

RC2052

(2052, X'804') La cola se ha suprimido.

RC2394

(2394, X'95A') La cola tiene un tipo de índice incorrecto.

RC2058

(2058, X'80A') El nombre del gestor de colas no es válido o no se conoce.

RC2059

(2059, X'80B') El gestor de colas no está disponible para efectuar la conexión.

RC2161

(2161, X'871') El gestor de colas se está desactivando temporalmente.

RC2162

(2162, X'872') El gestor de colas está concluyendo.

RC2102

(2102, X'836') No hay disponibles suficientes recursos del sistema.

RC2069

(2069, X'815') Símbolo pendiente para este descriptor de contexto.

RC2071

(2071, X'817') No hay suficiente almacenamiento disponible.

RC2109

(2109, X'83D') El programa de salida ha suprimido la llamada.

RC2024

(2024, X'7E8') No pueden manejarse más mensajes dentro de la unidad de trabajo actual.

RC2072

(2072, X'818') No hay soporte para Syncpoint.

RC2195

(2195, X'893') Se ha producido un error inesperado.

RC2354

(2354, X'932') Ha fallado el listado en la unidad de trabajo global.

RC2355

(2355, X'933') No se soporta la mezcla de llamadas de unidad de trabajo.

RC2255

(2255, X'8CF') La unidad de trabajo no está disponible para ser utilizada por el gestor de colas.

RC2090

(2090, X'82A') El intervalo de espera de MQGMO no es válido.

RC2256

(2256, X'8D0') Se ha suministrado una versión equivocada de MQGMO.

RC2257

(2257, X'8D1') La versión del MQMD suministrado es incorrecta.

RC2298

(2298, X'8FA') La función solicitada no está disponible en el entorno actual.

Declaración RPG

```

C*..1.....2.....3.....4.....5.....6.....7..
C                                CALLP      MQCB(HCONN : OPERATN : CBDSC :
                                HOBJ : MSGDSC : GMO :
                                DATLEN : CMPCOD : REASON)

```

La definición del prototipo para la llamada es:

```

DMQCB          PR          EXTPROC('MQCB')
D* Connection handle
D HCONN                10I 0 VALUE
D* Operation
D OPERATN              10I 0 VALUE
D* Callback descriptor
D CBDSC                180A
D* Object handle
D HOBJ                10I 0 VALUE
D* Message Descriptor
D MSGDSC              364A
D* Get options
D GMO                112A
D* Completion code
D CMPCOD              10I 0
* Reason code qualifying CompCode
D REASON              10I 0

```

MQCLOSE (Cerrar objeto) en IBM i

La llamada MQCLOSE renuncia al acceso a un objeto y es la inversa de la llamada MQOPEN.

- [“Sintaxis” en la página 1297](#)
- [“Notas de uso” en la página 1297](#)
- [“Parámetros” en la página 1298](#)
- [“Declaración RPG” en la página 1303](#)

Sintaxis

MQCLOSE (*HCONN, HOBJ, OPTS, CMPCOD, REASON*)

Notas de uso

1. Cuando una aplicación emite la llamada MQDISC, o finaliza de forma normal o anómala, los objetos abiertos por la aplicación y que siguen abiertos se cierran automáticamente con la opción CONONE.
2. Los puntos siguientes se aplican si el objeto que se está cerrando es una *cola*:
 - Si las operaciones en la cola se realizan como parte de una unidad de trabajo, la cola se puede cerrar antes o después de que se produzca el punto de sincronización sin afectar al resultado del punto de sincronización.
 - Si la cola se ha abierto con la opción OOBW, el cursor para examinar se destruye. Si la cola se vuelve a abrir posteriormente con la opción OOBW, se crea un nuevo cursor para examinar (consulte la opción OOBW descrita en MQOPEN).
 - Si un mensaje está bloqueado actualmente para este descriptor de contexto en el momento de la llamada MQCLOSE, el bloqueo se libera (consulte la opción GMLK descrita en [“MQGMO \(Opciones de obtención de mensajes\) activado IBM i”](#) en la página 1096).
3. Los puntos siguientes se aplican si el objeto que se está cerrando es una *cola dinámica* (permanente o temporal):

- Para una cola dinámica, se pueden especificar las opciones CODEL o COPURG independientemente de las opciones especificadas en la llamada MQOPEN correspondiente.
- Cuando se suprime una cola dinámica, se cancelan todas las llamadas MQGET con la opción GMWT que están pendientes en la cola y se devuelve el código de razón RC2052. Consulte la opción GMWT descrita en [“MQGMO \(Opciones de obtención de mensajes\) activado IBM i”](#) en la página 1096.

Después de que se haya suprimido una cola dinámica, cualquier llamada (distinta de MQCLOSE) que intente hacer referencia a la cola utilizando un descriptor de contexto *HOBJ* adquirido anteriormente falla con el código de razón RC2052.

Tenga en cuenta que aunque las aplicaciones no pueden acceder a una cola suprimida, la cola no se elimina del sistema y los recursos asociados no se liberan, hasta que se hayan cerrado todos los descriptors de contexto que hacen referencia a la cola y todas las unidades de trabajo que afectan a la cola se hayan confirmado o restituido.

- Cuando se suprime una cola dinámica permanente, si el descriptor de contexto *HOBJ* especificado en la llamada MQCLOSE no es el que ha devuelto la llamada MQOPEN que ha creado la cola, se comprueba que el identificador de usuario que se ha utilizado para validar la llamada MQOPEN esté autorizado para suprimir la cola. Si se ha especificado la opción OOALTU en la llamada MQOPEN, el identificador de usuario seleccionado es *ODAU*.

Esta comprobación no se realiza si:

- El descriptor de contexto especificado es el devuelto por la llamada MQOPEN que ha creado la cola.
- La cola que se está suprimiendo es una cola dinámica temporal.
- Cuando se cierra una cola dinámica temporal, si el descriptor de contexto *HOBJ* especificado en la llamada MQCLOSE es el que ha devuelto la llamada MQOPEN que ha creado la cola, se suprime la cola. Esto ocurre independientemente de las opciones de cierre especificadas en la llamada MQCLOSE. Si hay mensajes en la cola, se descartan; no se generan mensajes de informe.

Si hay unidades de trabajo no confirmadas que afectan a la cola, la cola y sus mensajes se siguen suprimiendo, pero esto no hace que fallen las unidades de trabajo. Sin embargo, como se ha descrito anteriormente, los recursos asociados con las unidades de trabajo no se liberan hasta que se haya confirmado o restituido cada una de las unidades de trabajo.

4. Los puntos siguientes se aplican si el objeto que se está cerrando es una *lista de distribución*:

- La única opción de cierre válida para una lista de distribución es CONONE; la llamada falla con el código de razón RC2046 o RC2045 si se especifica alguna otra opción.
- Cuando se cierra una lista de distribución, los códigos de terminación y los códigos de razón individuales no se devuelven para las colas de la lista; sólo los parámetros **CMPCOD** y **REASON** de la llamada están disponibles para fines de diagnóstico.

Si se produce una anomalía al cerrar una de las colas, el gestor de colas continúa el proceso e intenta cerrar las colas restantes de la lista de distribución. A continuación, los parámetros **CMPCOD** y **REASON** de la llamada se establecen para devolver información que describe la anomalía. Por lo tanto, es posible que el código de terminación sea CCFAIL, aunque la mayoría de las colas se hayan cerrado correctamente. La cola que ha encontrado el error no está identificada.

Si hay una anomalía en más de una cola, no se define qué anomalía se notifica en los parámetros **CMPCOD** y **REASON**.

Parámetros

La llamada MQCLOSE tiene los parámetros siguientes:

HCONN (entero con signo de 10 dígitos)-entrada

Descriptor de conexión.

Este manejador representa la conexión con el gestor de colas. El valor de *HCONN* ha sido devuelto por una llamada MQCONN o MQCONNX anterior.

HOBJ (entero con signo de 10 dígitos)-entrada/salida

El manejador del objeto.

Este descriptor de contexto representa el objeto que se está cerrando. El objeto puede ser de cualquier tipo. El valor de *HOBJ* ha sido devuelto por una llamada MQOPEN anterior.

Al finalizar correctamente la llamada, el gestor de colas establece este parámetro en un valor que no es un descriptor de contexto válido para el entorno. Este valor es:

HOUNUH

Descriptor de objeto inutilizable.

OPTS (entero con signo de 10 dígitos)-entrada

Opciones que controlan la acción de MQCLOSE.

El parámetro **OPTS** controla cómo se cierra el objeto. Sólo las colas dinámicas permanentes y las suscripciones se pueden cerrar de más de una forma. Las colas dinámicas permanentes se pueden retener o suprimir; son colas con un atributo **DefinitionType** que tiene el valor QDPERM (consulte el atributo **DefinitionType** descrito en [“Atributos para colas”](#) en la página 1399). Las opciones de cierre se resumen en una tabla más adelante en este tema.

Las suscripciones duraderas se pueden conservar o eliminar; estas se crean utilizando la llamada MQSUB con la opción SODUR.

Al cerrar el descriptor de contexto en un destino gestionado (es decir, el parámetro **Hobj** devuelto en una llamada MQSUB que ha utilizado la opción SOMAN), el gestor de colas limpiará las publicaciones no recuperadas cuando también se haya eliminado la suscripción asociada. Esto se realiza utilizando la opción CORMSB en el parámetro **Hsub** devuelto en una llamada MQSUB. Tenga en cuenta que CORMSB es el comportamiento predeterminado en MQCLOSE para una suscripción no duradera.

Al cerrar un descriptor de contexto en un destino no gestionado, es responsable de limpiar la cola donde se envían las publicaciones. Se recomienda cerrar primero la suscripción utilizando CORMSB y, a continuación, procesar los mensajes fuera de la cola hasta que no quede ninguno.

Se debe especificar uno (y sólo uno) de los siguientes:

Opciones de cierre de cola dinámica

Estas opciones controlan cómo se cierran las colas dinámicas permanentes:

MODELO

Suprima la cola.

La cola se suprime si se cumple alguna de las condiciones siguientes:

- Es una cola dinámica permanente, creada por una llamada MQOPEN anterior, y no hay mensajes en la cola y no hay solicitudes de obtención o colocación no confirmadas pendientes para la cola (para la tarea actual o para cualquier otra tarea).
- Es la cola dinámica temporal que ha creado la llamada MQOPEN que ha devuelto HOBJ. En este caso, se depuran todos los mensajes de la cola.

En todos los demás casos, incluido el caso en el que se ha devuelto Hobj en una llamada MQSUB, la llamada falla con el código de razón RC2045y el objeto no se suprime.

COPURG

Suprima la cola, depurando los mensajes que haya en ella.

La cola se suprime si se cumple alguna de las condiciones siguientes:

- Es una cola dinámica permanente, creada por una llamada MQOPEN anterior, y no hay solicitudes get o put no confirmadas pendientes para la cola (ni para la tarea actual ni para ninguna otra tarea).
- Es la cola dinámica temporal que ha creado la llamada MQOPEN que ha devuelto HOBJ.

En todos los demás casos, incluido el caso en el que se ha devuelto Hobj en una llamada MQSUB, la llamada falla con el código de razón RC2045y el objeto no se suprime.

La tabla siguiente muestra qué opciones de cierre son válidas y si el objeto se conserva o se suprime.

Tabla 782. Opciones de cierre válidas para utilizar con objetos retenidos o suprimidos			
Tipo de objeto o cola	CONONA	MODELO	COPURG
Objeto que no es una cola	Retenido	No válido	No válido
Cola predefinida	Retenido	No válido	No válido
cola dinámica permanente	Retenido	Suprimido si está vacío y no hay actualizaciones pendientes	Mensajes suprimidos; cola suprimida si no hay actualizaciones pendientes
Cola dinámica temporal (llamada emitida por el creador de la cola)	Suprimida	Suprimida	Suprimida
Cola dinámica temporal (llamada no emitida por el creador de la cola)	Retenido	No válido	No válido
Lista de distribución	Retenido	No válido	No válido
Destino de suscripción gestionada	Retenido	No válido	No válido
Lista de distribución (se ha eliminado la suscripción)	Mensajes suprimidos; cola suprimida	No válido	No válido

Opciones de cierre de suscripción

Estas opciones controlan si se eliminan las suscripciones duraderas cuando se cierra el descriptor de contexto y si se limpian las publicaciones que todavía están a la espera de ser leídas por la aplicación. Estas opciones sólo son válidas para su uso con un descriptor de objeto devuelto en el parámetro HSUB de una llamada MQSUB.

COKPSB

El descriptor de contexto de la suscripción se cierra, pero la suscripción realizada se mantiene. Las publicaciones se seguirán enviando al destino especificado en la suscripción. Esta opción sólo es válida si la suscripción se ha realizado con la opción SODUR. COKPSB es el valor predeterminado si la suscripción es duradera

CORMSB

La suscripción se elimina y se cierra el descriptor de contexto de la suscripción.

El parámetro **Hobj** de la llamada MQSUB no se invalida mediante el cierre del parámetro **Hsub** y puede seguir utilizándose para MQGET o MQCB para recibir las publicaciones restantes. Cuando el parámetro **Hobj** de la llamada MQSUB también se cierra, si era un destino gestionado, se eliminarán las publicaciones no recuperadas.

CORMSB es el valor predeterminado si la suscripción no es duradera.

Estas opciones de cierre de suscripción se resumen en las tablas siguientes:

Para cerrar un descriptor de contexto de suscripción duradera pero dejar la suscripción alrededor, utilice las siguientes opciones de cierre de suscripción:

Tabla 783. Opciones de tarea para cerrar un descriptor de contexto de suscripción duradera y dejar la suscripción alrededor	
Tarea	Opción de cierre de suscripción
Mantener publicaciones en un descriptor de contexto MQOPENed	COKPSB
Eliminar publicaciones en un descriptor de contexto MQOPENed	Acción no permitida
Mantener publicaciones en un descriptor de contexto con SOMAN	COKPSB
Eliminar publicaciones en un descriptor de contexto con SOMAN	Acción no permitida

Para anular la suscripción, ya sea cerrando un descriptor de contexto de suscripción duradera y cancelándola o cerrando un descriptor de contexto de suscripción no duradera, utilice las siguientes opciones de cierre de suscripción:

Tabla 784. Opciones de tarea para anular la suscripción	
Tarea	Opción de cierre de suscripción
Mantener publicaciones en un descriptor de contexto MQOPENed	CORMSB
Eliminar publicaciones en un descriptor de contexto MQOPENed	Acción no permitida
Mantener publicaciones en un descriptor de contexto con SOMAN	CORMSB
Eliminar publicaciones en un descriptor de contexto con SOMAN	COPGSB

Opciones de lectura anticipada

Las opciones siguientes controlan lo que sucede con los mensajes no persistentes que se han enviado al cliente antes de que una aplicación los solicitara y que todavía no han sido consumidos por la aplicación. Estos mensajes se almacenan en el almacenamiento intermedio de lectura anticipada del cliente a la espera de ser solicitados por la aplicación y se pueden descartar o consumir de la cola antes de que se complete MQCLOSE.

CIMM

El objeto se cierra inmediatamente y los mensajes que se han enviado al cliente antes de que una aplicación los solicitara se descartan y no están disponibles para que los consuma ninguna aplicación. Éste es el valor predeterminado.

COQSC

Se realiza una solicitud para cerrar el objeto, pero si alguno de los mensajes que se han enviado al cliente antes de que una aplicación los solicitara, sigue residiendo en el almacenamiento intermedio de lectura anticipada del cliente, la llamada MQCLOSE devolverá un código de aviso de RC2458y el descriptor de contexto del objeto seguirá siendo válido.

A continuación, la aplicación puede seguir utilizando el descriptor de objeto para recuperar mensajes hasta que no haya más disponibles y, a continuación, vuelva a cerrar el objeto. No se enviarán más mensajes al cliente antes de que se solicite una aplicación, la lectura anticipada está ahora desactivada.

Se recomienda a las aplicaciones que utilicen COQSC en lugar de intentar llegar a un punto en el que no haya más mensajes en el almacenamiento intermedio de lectura anticipada del cliente, ya que podría llegar un mensaje entre la última llamada MQGET y el siguiente MQCLOSE que se descartaría si se utilizara COIMM.

Si se emite un MQCLOSE con COQSC desde una función de devolución de llamada asíncrona, se aplica el mismo comportamiento de los mensajes de lectura anticipada. Si se devuelve el código de aviso RC2458 , se llamará a la función de devolución de llamada al menos una vez más. Cuando el último mensaje restante que se ha leído con anticipación se ha pasado a la función de devolución de llamada, el campo CBCFLG se establece en CBCFBE.

Opción predeterminada

Si no necesita ninguna de las opciones descritas anteriormente, puede utilizar la opción siguiente:

CONONA

No es necesario ningún proceso de cierre opcional.

Debe especificarse para:

- Objetos que no son colas
- Colas predefinidas
- Colas dinámicas temporales (pero sólo en los casos en los que *HOBJ* no es el manejador devuelto por la llamada MQOPEN que ha creado la cola).
- Listas de distribución

En todos los casos anteriores, el objeto se conserva y no se suprime.

Si se especifica esta opción para una cola dinámica temporal:

- La cola se suprime, si la ha creado la llamada MQOPEN que ha devuelto *HOBJ* ; los mensajes que están en la cola se depuran.
- En todos los demás casos, la cola (y los mensajes que contiene) se retienen.

Si se especifica esta opción para una cola dinámica permanente, la cola se conserva y no se suprime.

CMPCOD (entero con signo de 10 dígitos)-salida

Código de terminación.

Es uno de los siguientes:

CCOK

Realización satisfactoria.

CCWARN

Aviso (finalización parcial).

CCFAIL

La llamada no se ha realizado satisfactoriamente.

REASON (entero con signo de 10 dígitos)-salida

Código de razón que califica *CMPCOD*.

Si *CMPCOD* es CCOK:

RCNONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CMPCOD* es CCWARN:

RC2241

(2241, X'8C1') El grupo de mensajes no está completo.

RC2242

(2242, X'8C2') El mensaje lógico no está completo.

Si *CMPCOD* es CCFAIL:

RC2219

(2219, X'8AB') Se ha vuelto a introducir la llamada « MQI » antes de que finalizara la llamada anterior.

RC2009

(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

RC2018

(2018, X'7E2') El manejador de conexión no es válido.

RC2019

(2019, X'7E3') El manejador de objeto no es válido.

RC2035

(2035, X'7F3') Acceso no autorizado.

RC2101

(2101, X'835') El objeto se ha dañado.

RC2045

(2045, X'7FD') Opción no válida para el tipo de objeto.

RC2046

(2046, X'7FE') Las opciones no son válidas o no son coherentes.

RC2058

(2058, X'80A') El nombre del gestor de colas no es válido o no se conoce.

RC2059

(2059, X'80B') El gestor de colas no está disponible para efectuar la conexión.

RC2162

(2162, X'872') El gestor de colas está concluyendo.

RC2055

(2055, X'807') La cola contiene uno o más mensajes o solicitudes put u get no confirmadas.

RC2102

(2102, X'836') No hay disponibles suficientes recursos del sistema.

RC2063

(2063, X'80F') Se ha producido un error de seguridad.

RC2071

(2071, X'817') No hay suficiente almacenamiento disponible.

RC2195

(2195, X'893') Se ha producido un error inesperado.

Declaración RPG

```
C*.1.....2.....3.....4.....5.....6.....7..  
C          CALLP      MQCLOSE(HCONN : HOBJ : OPTS :  
C                                CMPCOD : REASON)
```

La definición de prototipo para la llamada es:

```
D*.1.....2.....3.....4.....5.....6.....7..  
DMQCLOSE      PR          EXTPROC('MQCLOSE')  
D* Connection handle  
D HCONN              10I 0 VALUE  
D* Object handle  
D HOBJ              10I 0  
D* Options that control the action of MQCLOSE  
D OPTS              10I 0 VALUE  
D* Completion code  
D CMPCOD            10I 0  
D* Reason code qualifying CMPCOD  
D REASON            10I 0
```

MQCMIT (Confirmar cambios) en IBM i

La llamada MQCMIT indica al gestor de colas que la aplicación ha alcanzado un punto de sincronización y que todas las operaciones de obtención y envío de mensajes que se hayan producido desde el último punto de sincronización deben hacerse permanentes. Los mensajes incluidos como parte de una unidad de trabajo se ponen a disposición de otras aplicaciones; los mensajes recuperados como parte de una unidad de trabajo se eliminan.

- [“Sintaxis” en la página 1303](#)
- [“Notas de uso” en la página 1303](#)
- [“Parámetros” en la página 1304](#)
- [“Declaración RPG” en la página 1305](#)

Sintaxis

MQCMIT (*HCONN*, *COMCOD*, *REASON*)

Notas de uso

Tenga en cuenta estas notas de uso al utilizar MQCMIT.

1. Esta llamada solo se puede utilizar cuando el propio gestor de colas coordina la unidad de trabajo. Se trata de una unidad de trabajo local, donde los cambios solo IBM MQ afectan a los recursos.
2. En entornos en los que el gestor de colas no coordina la unidad de trabajo, se debe utilizar la llamada de confirmación adecuada en lugar de MQCMIT. El entorno también puede admitir una confirmación implícita causada por el cierre normal de la aplicación.
 - En IBM i, esta llamada se puede utilizar para unidades de trabajo locales coordinadas por el gestor de colas. Esto significa que no debe existir una definición de compromiso a nivel de trabajo, es decir, no debe haberse emitido el STRCMTCTL comando con el **CMTSCOPE(*JOB)** parámetro para el trabajo.
3. Si una aplicación finaliza con cambios no confirmados en una unidad de trabajo, la disposición de dichos cambios depende de si la aplicación finaliza de forma normal o anómala. Consulte las notas de uso en [“MQDISC \(Desconectar gestor de colas\) en IBM i” en la página 1319](#) para obtener más detalles.
4. Cuando una aplicación coloca u obtiene mensajes en grupos o segmentos de mensajes lógicos, el gestor de colas conserva la información relativa al grupo de mensajes y al mensaje lógico de las últimas llamadas MQPUT y MQGET realizadas con éxito. Esta información está asociada al identificador de cola e incluye datos tales como:

- Los valores de los campos *MDGID*, *MDSEQ*, *MDMFL* *MDOFF*, y en *MQMD*.
- Si el mensaje forma parte de una unidad de trabajo.
- Para la llamada *MQPUT*: si el mensaje es persistente o no persistente.

Cuando se confirma una unidad de trabajo, el gestor de colas conserva la información del grupo y del segmento, y la aplicación puede seguir enviando o recibiendo mensajes en el grupo de mensajes actual o en el mensaje lógico.

Conservar la información del grupo y del segmento cuando se confirma una unidad de trabajo permite a la aplicación distribuir un grupo de mensajes grande o un mensaje lógico grande compuesto por muchos segmentos entre varias unidades de trabajo. El uso de varias unidades de trabajo puede resultar ventajoso si el gestor de colas local solo dispone de un almacenamiento de colas limitado. Sin embargo, la aplicación debe conservar suficiente información para poder reiniciar el envío o la recepción de mensajes en el punto correcto si se produce un fallo del sistema. Para obtener más información sobre cómo reiniciar en el punto correcto tras un fallo del sistema, consulte la opción *PMLOGO* descrita en “*MQPMO (Opciones de mensajes de envío) activado IBM i*” en la página 1198 y la opción *GMLOGO* descrita en “*MQGMO (Opciones de obtención de mensajes) activado IBM i*” en la página 1096.

5. Cuando una aplicación ha incluido la inserción de un mensaje persistente dentro de la unidad de trabajo, y la llamada *MQPUT* o *MQPUT1* ha utilizado la opción *MQPMO_ASYNC_RESPONSE*, y la inserción ha fallado, entonces *MQCMIT* devuelve *MQCC_FAILED*, *MQRC_BACKED_OUT*. Consulte la descripción de *MQPMO_ASYNC_RESPONSE* en [Detalles del campo para MQPMO](#).

En un escenario similar, para un mensaje no persistente dentro de la unidad de trabajo, *MQCMIT* devuelve *MQCC_OK*, *MQRC_NONE*.

En el caso de que haya varias operaciones de envío (*put*) y se haya utilizado *MQPMO_ASYNC_RESPONSE*, si falla el envío de cualquier mensaje persistente, la llamada *MQCMIT* devolverá *MQCC_FAILED*, *MQRC_BACKED_OUT*.

Las notas de uso restantes solo se aplican cuando el gestor de colas coordina las unidades de trabajo:

1. Una unidad de trabajo tiene el mismo alcance que un identificador de conexión. Esto significa que todas IBM MQ las llamadas que afecten a una unidad de trabajo concreta deben realizarse utilizando el mismo identificador de conexión. Las llamadas realizadas utilizando un identificador de conexión diferente (por ejemplo, llamadas realizadas por otra aplicación) afectan a una unidad de trabajo diferente. Consulte el **HCONN** parámetro descrito en *MQCONN* para obtener información sobre el ámbito de los identificadores de conexión.
2. Solo los mensajes que se colocaron o recuperaron como parte de la unidad de trabajo actual se ven afectados por esta llamada.
3. Una aplicación de larga duración que emite llamadas *MQGET*, *MQPUT* o *MQPUT1* dentro de una unidad de trabajo, pero que nunca emite una llamada de confirmación o de retroceso, puede provocar que las colas se llenen de mensajes que no están disponibles para otras aplicaciones. Para protegerse contra esta posibilidad, el administrador debe establecer el atributo del **MaxUncommittedMsgs** gestor de colas en un valor lo suficientemente bajo como para evitar que las aplicaciones descontroladas llenen las colas, pero lo suficientemente alto como para permitir que las aplicaciones de mensajería esperadas funcionen correctamente.

Parámetros

La llamada *MQCMIT* tiene los siguientes parámetros:

HCONN (entero con signo de 10 dígitos) - entrada

Descriptor de conexión.

Este manejador representa la conexión con el gestor de colas. El valor de *HCONN* fue devuelto por una llamada anterior a *MQCONN* o *MQCONN*.

COMCOD (entero con signo de 10 dígitos) - salida

Código de terminación.

Es uno de los siguientes:

CCOK

Realización satisfactoria.

CCWARN

Aviso (finalización parcial).

CCFAIL

La llamada no se ha realizado satisfactoriamente.

RAZÓN (entero con signo de 10 dígitos) - salida

Código de motivo calificativo *COMCOD*.

Si *COMCOD* es CCOK:

RCNONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *COMCOD* es CCWARN:

RC2003

(2003, X'7D3') Unidad de trabajo restituida.

RC2124

(2124, X'84C') El resultado de la operación de confirmación está pendiente.

Si *COMCOD* es CCFAIL:

RC2219

(2219, X'8AB') Se ha vuelto a introducir la llamada « MQI » antes de que finalizara la llamada anterior.

RC2009

(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

RC2018

(2018, X'7E2') El manejador de conexión no es válido.

RC2101

(2101, X'835') El objeto se ha dañado.

RC2123

(2123, X'84B') El resultado de la operación de confirmación o retroceso es mixto.

RC2162

(2162, X'872') El gestor de colas está concluyendo.

RC2102

(2102, X'836') No hay disponibles suficientes recursos del sistema.

RC2071

(2071, X'817') No hay suficiente almacenamiento disponible.

RC2195

(2195, X'893') Se ha producido un error inesperado.

Declaración RPG

```
C*..1.....2.....3.....4.....5.....6.....7..  
C                                CALLP      MQCMIT(HCONN : COMCOD : REASON)
```

La definición del prototipo para la llamada es:

```
D*..1.....2.....3.....4.....5.....6.....7..  
DMQCMIT      PR                                EXTPROC('MQCMIT')  
D* Connection handle
```

D HCONN	10I 0 VALUE
D* Completion code	
D COMCOD	10I 0
D* Reason code qualifying COMCOD	
D REASON	10I 0

IBM i MQCONN (Conectar gestor de colas) en IBM i

La llamada MQCONN conecta un programa de aplicación a un gestor de colas. Proporciona un descriptor de conexión del gestor de colas, que utiliza la aplicación en las llamadas de cola de mensajes posteriores.

- Las aplicaciones deben utilizar la llamada MQCONN o MQCONNX para conectarse al gestor de colas y la llamada MQDISC para desconectarse del gestor de colas.

En IBM MQ for Multiplatforms, cada hebra de una aplicación puede conectarse a distintos gestores de colas. En otros sistemas, todas las conexiones simultáneas dentro de un proceso deben estar en el mismo gestor de colas.

- “Sintaxis” en la [página 1306](#)
- “Notas de uso” en la [página 1306](#)
- “Parámetros” en la [página 1307](#)
- “Declaración RPG” en la [página 1309](#)

Sintaxis

MQCONN (QMNAME, HCONN, CMPCOD, REASON)

Notas de uso

- El gestor de colas con el que se realiza la conexión utilizando la llamada MQCONN se denomina *gestor de colas local*.
- Las colas que son propiedad del gestor de colas local aparecen en la aplicación como colas locales. Es posible colocar mensajes y obtener mensajes de estas colas.

Las colas compartidas que son propiedad del grupo de compartición de colas al que pertenece el gestor de colas local aparecen en la aplicación como colas locales. Es posible colocar mensajes y obtener mensajes de estas colas.

Las colas que son propiedad de gestores de colas remotos aparecen como colas remotas. Es posible colocar mensajes en estas colas, pero no es posible obtener mensajes de estas colas.

- Si el gestor de colas falla mientras una aplicación se está ejecutando, la aplicación debe volver a emitir la llamada MQCONN para obtener un nuevo descriptor de conexión que utilizar en las llamadas IBM MQ posteriores. La aplicación puede emitir la llamada MQCONN periódicamente hasta que la llamada sea satisfactoria.

Si una aplicación no está segura de si está conectada al gestor de colas, la aplicación puede emitir de forma segura una llamada MQCONN para obtener un descriptor de conexión. Si la aplicación ya está conectada, el descriptor de contexto devuelto es el mismo que el devuelto por la llamada MQCONN anterior, pero con el código de terminación CCWARN y el código de razón RC2002.

- Cuando la aplicación haya terminado de utilizar las llamadas IBM MQ, la aplicación debe utilizar la llamada MQDISC para desconectarse del gestor de colas.
- En IBM i, los programas que finalizan de forma anómala no se desconectan automáticamente del gestor de colas. Por lo tanto, las aplicaciones deben escribirse para permitir la posibilidad de que la llamada MQCONN o MQCONNX devuelva el código de terminación CCWARN y el código de razón RC2002. El descriptor de conexión devuelto en esta situación se puede utilizar como normal.

Parámetros

La llamada MQCONN tiene los parámetros siguientes:

QMNAME (serie de caracteres de 48 bytes)-entrada

Nombre del gestor de colas.

Es el nombre del gestor de colas al que la aplicación desea conectarse. El nombre puede contener los siguientes caracteres:

- Caracteres alfabéticos en mayúsculas (de la A a la Z)
- Caracteres alfabéticos en minúsculas (de la a a la z)
- Dígitos numéricos (de 0 a 9)
- Punto (.), barra inclinada (/), subrayado (_), porcentaje (%)

El nombre no debe contener espacios en blanco iniciales o intercalados, pero puede contener espacios en blanco finales. Se puede utilizar un carácter nulo para indicar el final de datos significativos en el nombre; el nulo y cualquier carácter que lo siga se tratan como espacios en blanco. Las restricciones siguientes se aplican en los entornos indicados:

- En IBM i, los nombres que contienen caracteres en minúsculas, barras inclinadas o porcentaje deben especificarse entre comillas cuando se especifican en mandatos. Estas comillas no se deben especificar en el parámetro **QMNAME**.

Si el nombre consta por completo de espacios en blanco, se utiliza el nombre del gestor de colas *por omisión*.

El nombre especificado para **QMNAME** debe ser el nombre de un gestor de colas *conectable*.

Grupos de compartición de colas: En sistemas donde existen varios gestores de colas y están configurados para formar un grupo de compartición de colas, el nombre del grupo de compartición de colas se puede especificar para **QMNAME** en lugar del nombre de un gestor de colas. Esto permite a la aplicación conectarse a *cualquier* gestor de colas que esté disponible en el grupo de compartición de colas. El sistema también se puede configurar para que un **QMNAME** en blanco provoque la conexión con el grupo de compartición de colas en lugar de con el gestor de colas predeterminado.

Si **QMNAME** especifica el nombre del grupo de compartición de colas, pero también hay un gestor de colas con ese nombre en el sistema, se establece una conexión con el último en lugar del primero. Sólo si la conexión falla, se intenta la conexión con uno de los gestores de colas del grupo de compartición de colas.

Si la conexión es satisfactoria, el descriptor de contexto devuelto por la llamada MQCONN o MQCONNX se puede utilizar para acceder a *todos* los recursos (compartidos y no compartidos) que pertenecen al gestor de colas concreto con el que se ha realizado la conexión. El acceso a estos recursos está sujeto a los controles de autorización típicos.

Si la aplicación emite dos llamadas MQCONN o MQCONNX para establecer conexiones simultáneas, y una o ambas llamadas especifica el nombre del grupo de compartición de colas, la segunda llamada puede devolver el código de terminación CCWARN y el código de razón RC2002. Esto ocurre cuando la segunda llamada se conecta al mismo gestor de colas que la primera llamada.

Los grupos de compartición de colas solo se admiten en z/OS. La conexión a un grupo de compartición de colas sólo está soportada en los entornos por lotes, por lotes RRS y TSO.

IBM MQ aplicaciones cliente: Para aplicaciones IBM MQ MQI client, se intenta una conexión para cada definición de canal de conexión de cliente con el nombre de gestor de colas especificado, hasta que una sea satisfactoria. Sin embargo, el gestor de colas debe tener el mismo nombre que el nombre especificado. Si se especifica un nombre todo en blanco, se intenta cada canal de conexión de cliente con un nombre de gestor de colas todo en blanco hasta que uno sea satisfactorio; en este caso, no hay ninguna comprobación con respecto al nombre real del gestor de colas.

Grupos de gestores de colas de cliente IBM MQ: si el nombre especificado empieza con un asterisco (*), el gestor de colas real con el que se realiza la conexión puede tener un nombre distinto del

especificado por la aplicación. El nombre especificado (sin el asterisco) define un *grupo* de gestores de colas que son elegibles para la conexión. La implementación selecciona uno del grupo intentando cada uno a su vez, en orden alfabético, hasta que se encuentra uno al que se puede realizar una conexión. Si ninguno de los gestores de colas del grupo está disponible para la conexión, la llamada falla. Cada gestor de colas se intenta una sola vez. Si se especifica un asterisco solo para el nombre, se utiliza un grupo de gestores de colas predeterminado definido por la implementación.

Los grupos de gestores de colas sólo están soportados para las aplicaciones que se ejecutan en un entorno de cliente MQ; la llamada falla si una aplicación no cliente especifica un nombre de gestor de colas que empieza con un asterisco. Un grupo se define proporcionando varias definiciones de canal de conexión de cliente con el mismo nombre de gestor de colas (el nombre especificado sin el asterisco), para comunicarse con cada uno de los gestores de colas del grupo. El grupo predeterminado se define proporcionando una o más definiciones de canal de conexión de cliente, cada una con un nombre de gestor de colas en blanco (por lo tanto, especificar un nombre en blanco tiene el mismo efecto que especificar un solo asterisco para el nombre de una aplicación cliente).

Después de conectarse a un gestor de colas de un grupo, una aplicación puede especificar espacios en blanco de la forma habitual en los campos de nombre de gestor de colas en los descriptores de mensaje y objeto para indicar el nombre del gestor de colas al que se ha conectado realmente la aplicación (el *gestor de colas local*). Si la aplicación necesita conocer este nombre, se puede emitir la llamada MQINQ para consultar el atributo del gestor de colas **QMGrName**.

El prefijo de un asterisco al nombre de conexión implica que la aplicación no depende de la conexión con un gestor de colas determinado del grupo. Las aplicaciones adecuadas serían:

- Aplicaciones que colocan mensajes pero no obtienen mensajes.
- Aplicaciones que colocan mensajes de solicitud y, a continuación, obtienen los mensajes de respuesta de una cola *dinámica temporal*.

Las aplicaciones no adecuadas serían aquellas que necesitan obtener mensajes de una cola determinada en un gestor de colas determinado; dichas aplicaciones no deben añadir un asterisco al nombre.

Tenga en cuenta que si se especifica un asterisco, la longitud máxima del resto del nombre es de 47 caracteres.

La longitud de este parámetro la proporciona LNQMNM.

HCONN (entero con signo de 10 dígitos)-salida

Descriptor de conexión.

Este manejador representa la conexión con el gestor de colas. Debe especificarse en todas las llamadas de cola de mensajes posteriores emitidas por la aplicación. Deja de ser válido cuando se emite la llamada MQDISC, o cuando termina la unidad de proceso que define el ámbito del descriptor de contexto.

El ámbito del descriptor de contexto está restringido a la unidad más pequeña de proceso paralelo soportado por la plataforma en la que se ejecuta la aplicación; el descriptor de contexto no es válido fuera de la unidad de proceso paralelo desde la que se ha emitido la llamada MQCONN.

- En IBM i, el ámbito del descriptor de contexto es el trabajo que emite la llamada.

CMPCOD (entero con signo de 10 dígitos)-salida

Código de terminación.

Es uno de los siguientes:

CCOK

Realización satisfactoria.

CCWARN

Aviso (finalización parcial).

CCFAIL

La llamada no se ha realizado satisfactoriamente.

REASON (entero con signo de 10 dígitos)-salida

Código de razón que califica *CMPCOD*.

Si *CMPCOD* es CCOK:

RCNONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CMPCOD* es CCWARN:

RC2002

(2002, X'7D2') Aplicación ya conectada.

Si *CMPCOD* es CCFAIL:

RC2219

(2219, X'8AB') Se ha vuelto a introducir la llamada « MQI » antes de que finalizara la llamada anterior.

RC2267

(2267, X'8DB') No se ha podido cargar la carga de trabajo del clúster exit.

RC2009

(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

RC2018

(2018, X'7E2') El manejador de conexión no es válido.

RC2035

(2035, X'7F3') Acceso no autorizado.

RC2137

(2137, X'859') El objeto no se ha abierto correctamente.

RC2058

(2058, X'80A') El nombre del gestor de colas no es válido o no se conoce.

RC2059

(2059, X'80B') El gestor de colas no está disponible para efectuar la conexión.

RC2161

(2161, X'871') El gestor de colas se está desactivando temporalmente.

RC2162

(2162, X'872') El gestor de colas está concluyendo.

RC2102

(2102, X'836') No hay disponibles suficientes recursos del sistema.

RC2063

(2063, X'80F') Se ha producido un error de seguridad.

RC2071

(2071, X'817') No hay suficiente almacenamiento disponible.

RC2195

(2195, X'893') Se ha producido un error inesperado.

Declaración RPG

```
C*..1.....2.....3.....4.....5.....6.....7..  
C                                CALLP      MQCONN(QMNAME : HCONN : CMPCOD :  
C                                REASON)
```

La definición de prototipo para la llamada es:

```
D*..1.....2.....3.....4.....5.....6.....7..  
DMQCONN      PR      EXTPROC('MQCONN')  
D* Name of queue manager  
D QMNAME      48A  
D* Connection handle
```

D HCONN	10I 0
D* Completion code	
D CMPCOD	10I 0
D* Reason code qualifying CMPCOD	
D REASON	10I 0

IBM i MQCONN (Conectar gestor de colas (ampliado)) en IBM i

La llamada MQCONN conecta un programa de aplicación a un gestor de colas. Proporciona un descriptor de conexión del gestor de colas, que utiliza la aplicación en las llamadas IBM MQ posteriores.

La llamada MQCONN es como la llamada MQCONN, excepto que MQCONN permite especificar opciones para controlar la forma en que funciona la llamada.

En IBM MQ for Multiplatforms, cada hebra de una aplicación puede conectarse a distintos gestores de colas. En otros sistemas, todas las conexiones simultáneas dentro de un proceso deben estar en el mismo gestor de colas.

- [“Sintaxis” en la página 1310](#)
- [“Parámetros” en la página 1310](#)
- [“Declaración RPG” en la página 1311](#)

Sintaxis

MQCONN (*QMNAME*, *CNOPT*, *HCONN*, *CMPCOD*, *REASON*)

Parámetros

La llamada MQCONN tiene los parámetros siguientes:

QMNAME (serie de caracteres de 48 bytes)-entrada

Nombre del gestor de colas.

Consulte el parámetro **QMNAME** descrito en [“MQCONN \(Conectar gestor de colas\) en IBM i” en la página 1306](#) para obtener más detalles.

CNOPT (MQCNO)-entrada/salida

Opciones que controlan la acción de MQCONN.

Consulte [“MQCNO \(Opciones de conexión\) activado IBM i” en la página 1067](#) para obtener los detalles.

HCONN (entero con signo de 10 dígitos)-salida

Descriptor de conexión.

Consulte el parámetro **HCONN** descrito en [“MQCONN \(Conectar gestor de colas\) en IBM i” en la página 1306](#) para obtener más detalles.

CMPCOD (entero con signo de 10 dígitos)-salida

Código de terminación.

Consulte el parámetro **CMPCOD** descrito en [“MQCONN \(Conectar gestor de colas\) en IBM i” en la página 1306](#) para obtener más detalles.

REASON (entero con signo de 10 dígitos)-salida

Código de razón que califica *CMPCOD*.

Consulte el parámetro **REASON** descrito en [“MQCONN \(Conectar gestor de colas\) en IBM i” en la página 1306](#) para obtener detalles de los posibles códigos de razón.

La llamada MQCONN puede devolver los siguientes códigos de razón adicionales:

Si *CMPCOD* es CCFAIL:

RC2278

(2278, X'8E6') Los campos de conexión del cliente no son válidos.

RC2139

(2139, X'85B') La estructura Connect-options no es válida.

RC2046

(2046, X'7FE') Las opciones no son válidas o no son coherentes.

Declaración RPG

```
C*..1.....2.....3.....4.....5.....6.....7..  
C                                CALLP      MQCONN(QMNAME : HCONN : CMPCOD :  
C                                REASON)
```

La definición de prototipo para la llamada es:

```
D*..1.....2.....3.....4.....5.....6.....7..  
DMQCONN      PR      EXTPROC('MQCONN')  
D* Name of queue manager  
D QMNAME      48A  
D* Options that control the action of MQCONN  
D HCONN      224A  
D* Connection handle  
D HCONN      10I 0  
D* Completion code  
D CMPCOD      10I 0  
D* Reason code qualifying CMPCOD  
D REASON      10I 0
```

MQCRTMH (Crear manejador de mensajes) en IBM i

La llamada MQCRTMH devuelve un descriptor de mensaje.

Una aplicación puede utilizarla en llamadas de cola de mensajes posteriores:

- Utilice la llamada [MQSETMP](#) para establecer una propiedad del manejador de mensajes.
- Utilice la llamada [MQINQMP](#) para consultar el valor de una propiedad del manejador de mensajes.
- Utilice la llamada [MQDLTMP](#) para suprimir una propiedad del manejador de mensajes.

El descriptor de mensaje se puede utilizar en las llamadas MQPUT y MQPUT1 para asociar las propiedades del descriptor de mensaje con las propiedades del mensaje que se está colocando. De forma similar, al especificar un descriptor de mensaje en la llamada MQGET, se puede acceder a las propiedades del mensaje que se está recuperando utilizando el descriptor de mensaje cuando se completa la llamada MQGET.

Utilice [MQDLTMH](#) para suprimir el descriptor de mensaje.

- [“Sintaxis” en la página 1311](#)
- [“Parámetros” en la página 1311](#)
- [“Declaración RPG” en la página 1313](#)

Sintaxis

MQCRTMH (*Hconn*, *CrtMsgHOpts*, *Hmsg*, *CompCode*, *Reason*)

Parámetros

La llamada MQCRTMH tiene los parámetros siguientes:

HCONN (entero con signo de 10 dígitos)-entrada

Este manejador representa la conexión con el gestor de colas. El valor de *HCONN* ha sido devuelto por una llamada MQCONN o MQCONNEX anterior. Si la conexión con el gestor de colas deja de ser válida y no hay ninguna llamada IBM MQ operativa en el manejador de mensajes, se llama implícitamente a MQDLTMH para suprimir el mensaje.

De forma alternativa, puede especificar el valor siguiente:

HCUNAS

El descriptor de conexión no representa una conexión con ningún gestor de colas en particular.

Cuando se utiliza este valor, el descriptor de mensaje debe suprimirse con una llamada explícita a MQDLTMH para liberar cualquier almacenamiento asignado a él; IBM MQ nunca suprime implícitamente el descriptor de mensaje.

Debe haber al menos una conexión válida con un gestor de colas establecido en la hebra que crea el manejador de mensajes; de lo contrario, la llamada falla con RC2018.

CRTOPT (MQCMHO)-entrada

Las opciones que controlan la acción de MQCRTMH. Consulte MQCMHO para obtener detalles.

HMSG (entero con signo de 20 dígitos)-salida

En la salida, se devuelve un descriptor de mensaje que se puede utilizar para establecer, consultar y suprimir propiedades del descriptor de mensaje. Inicialmente, el manejador de mensajes no contiene propiedades.

Un descriptor de mensaje también tiene un descriptor de mensaje asociado. Inicialmente, este descriptor de mensaje contiene los valores predeterminados. Los valores de los campos de descriptor de mensaje asociados se pueden establecer y consultar utilizando las llamadas MQSETMP y MQINQMP. La llamada MQDLTMP restablece un campo del descriptor de mensaje a su valor predeterminado.

Si se especifica el parámetro *HCONN* como el valor HCUNAS, el descriptor de mensaje devuelto se puede utilizar en llamadas MQGET, MQPUT o MQPUT1 con cualquier conexión dentro de la unidad de proceso, pero sólo puede estar en uso por una llamada IBM MQ cada vez. Si el descriptor de contexto está en uso cuando una segunda llamada IBM MQ intenta utilizar el mismo descriptor de contexto de mensaje, la segunda llamada IBM MQ falla con el código de razón RC2499.

Si el parámetro *HCONN* no es HCUNAS, el descriptor de mensaje devuelto sólo se puede utilizar en la conexión especificada.

Se debe utilizar el mismo valor del parámetro *HCONN* en las siguientes llamadas a `MQI` en las que se utilice este identificador de mensaje:

- MQDLTMH
- MQSETMP
- MQINQMP
- MQDLTMP
- MQMHBUFF
- MQBUFMH

El descriptor de mensaje devuelto deja de ser válido cuando se emite la llamada MQDLTMH para el descriptor de mensaje, o cuando termina la unidad de proceso que define el ámbito del descriptor de contexto. MQDLTMH se llama implícitamente si se proporciona una conexión específica cuando se crea el descriptor de mensaje y la conexión con el gestor de colas deja de ser válida, por ejemplo, si se llama a MQDBC.

CMPCOD (entero con signo de 10 dígitos)-salida

El código de terminación; es uno de los siguientes:

CCOK

Realización satisfactoria.

CCFAIL

La llamada no se ha realizado satisfactoriamente.

REASON (entero con signo de 10 dígitos)-salida

El código de razón que califica *CMPCOD*.

Si *CMPCOD* es CCOK:

RCNONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CMPCOD* es CCFAIL:

RC2204

(2204, X'089C') Adaptador no disponible.

RC2130

(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

RC2157

(2157, X'86D') Los ASID primario e inicial varían.

RC2219

(2219, X'08AB') Se ha iniciado una llamada al MQI antes de que finalizara la llamada anterior.

RC2461

(2461, X'099D') La estructura de opciones de creación de mensajes no es válida.

RC2273

(2273, X'7D9') Se ha perdido la conexión con el gestor de colas.

RC2017

(2017, X'07E1') No hay más asas disponibles.

RC2018

(2018, X'7E2') El manejador de conexión no es válido.

RC2460

(2460, X'099C') El puntero del gestor de mensajes no es válido.

RC2046

(2046, X'07FE') Opciones no válidas o no coherentes.

RC2071

(2071, X'817') No hay suficiente almacenamiento disponible.

RC2195

(2195, X'893') Se ha producido un error inesperado.

Consulte [“Códigos de retorno para IBM i \(ILE RPG\)”](#) en la página 1447 para obtener más detalles.

Declaración RPG

```
C*..1.....2.....3.....4.....5.....6.....7..
C                                CALLP      MQCRTMH(HCONN : CRTOPT : HMSG :
                                           CMPCOD : REASON)
```

La definición de prototipo para la llamada es:

```
DMQCRTMH          PR          EXTPROC('MQCRTMH')
D* Connection handle
D HCONN           10I 0 VALUE
D* Options that control the action of MQCRTMH
D CRTOPT          12A
D* Message handle
D HMSG            20I 0
D* Completion code
D CMPCOD          10I 0
```

MQCTL (devolución de llamada de control) en IBM i

La llamada MQCTL realiza acciones de control en los manejadores de objetos abiertos para una conexión.

- [“Sintaxis” en la página 1314](#)
- [“Notas de uso” en la página 1314](#)
- [“Parámetros” en la página 1314](#)
- [“Declaración RPG” en la página 1319](#)

Sintaxis

MQCTL (*Hconn, Operation, ControlOpts, CompCode, Reason*)

Notas de uso

1. Las rutinas de devolución de llamada deben comprobar las respuestas de todos los servicios que invocan y, si la rutina detecta una condición que no se puede resolver, debe emitir un mandato MQCB (CBREG) para evitar llamadas repetidas a la rutina de devolución de llamada.

Parámetros

La llamada MQCTL tiene los parámetros siguientes:

HCONN (entero con signo de 10 dígitos)-entrada

Este manejador representa la conexión con el gestor de colas. El valor de *HCONN* ha sido devuelto por una llamada MQCONN o MQCONNEX anterior.

OPERATN (entero con signo de 10 dígitos)-entrada

La operación que se está procesando en la devolución de llamada definida para el descriptor de objeto especificado. Debe especificar una, y sólo una, de las opciones siguientes:

CTLSR

Iniciar el consumo de mensajes para todas las funciones de consumidor de mensajes definidas para el descriptor de conexión especificado.

Las devoluciones de llamada se ejecutan en una hebra iniciada por el sistema, que es diferente de cualquiera de las hebras de aplicación.

Esta operación proporciona el control del descriptor de conexión proporcionado al sistema. Las únicas llamadas a `MQI` que puede realizar un hilo distinto del hilo consumidor son:

- MQCTL con operación CTLSP
- MQCTL con operación CTLSU
- MQDISC-Realiza MQCTL con la operación CTLSP antes de desconectar el HConn.

Se devuelve RC2500 si se emite una llamada de API IBM MQ mientras se inicia el descriptor de conexión y la llamada no se origina en una función de consumidor de mensajes.

Si una conexión falla, esto detiene la conversación tan pronto como sea posible. Por lo tanto, es posible que se emita una llamada de API de IBM MQ en la hebra principal para recibir el código de retorno RC2500 durante un tiempo, seguido del código de retorno RC2009 cuando la conexión vuelve al estado detenido.

Esto se puede emitir en una función de consumidor. Para la misma conexión que la rutina de devolución de llamada, su única finalidad es cancelar una operación CTLSP emitida anteriormente.

Esta opción no está soportada si la aplicación está enlazada con una biblioteca IBM MQ sin hebras.

CTLSW

Iniciar el consumo de mensajes para todas las funciones de consumidor de mensajes definidas para el descriptor de conexión especificado.

Los consumidores de mensajes se ejecutan en la misma hebra y el control no se devuelve al interlocutor de MQCTL hasta que:

- Liberado por el uso de las operaciones MQCTL CTLSP o CTLSU, o
- Todas las rutinas de consumidor se han desregistrado o suspendido.

Si se anula el registro o se suspenden todos los consumidores, se emite una operación CTLSP implícita.

Esta opción no se puede utilizar desde una rutina de devolución de llamada, ya sea para el descriptor de conexión actual o cualquier otro descriptor de conexión. Si se intenta la llamada, se devuelve con RC2012.

Si, en algún momento durante una operación CTLSW no hay ningún consumidor registrado, no suspendido, la llamada falla con un código de razón de RC2446.

Si, durante una operación CTLSW, la conexión se suspende, la llamada MQCTL devuelve un código de razón de aviso de RC2521; la conexión permanece 'iniciada'.

La aplicación puede elegir emitir CTLSP o CTLRE. En este caso, la operación CTLRE se bloquea.

Esta opción no está soportada en un cliente de una sola hebra.

CTLSP

Detenga el consumo de mensajes y espere a que todos los consumidores completen sus operaciones antes de que se complete esta opción. Esta operación libera el descriptor de conexión.

Si se emite desde dentro de una rutina de devolución de llamada, esta opción no entra en vigor hasta que se cierra la rutina. No se llama a más rutinas de consumidor de mensajes después de que se hayan completado las rutinas de consumidor para los mensajes que ya se han leído, y después de que se hayan realizado llamadas de detención (si se han solicitado) a rutinas de devolución de llamada.

Si se emite fuera de una rutina de devolución de llamada, el control no vuelve al llamante hasta que se hayan completado las rutinas del consumidor para los mensajes ya leídos y después de que se hayan realizado las llamadas de detención (si se han solicitado) a las devoluciones de llamada. Sin embargo, las devoluciones de llamada permanecen registradas.

Esta función no tiene ningún efecto en los mensajes de lectura anticipada. Debe asegurarse de que los consumidores ejecuten MQCLOSE (COQSC), desde dentro de la función de devolución de llamada, para determinar si hay más mensajes disponibles para entregar.

CTLSU

Pausar el consumo de mensajes. Esta operación libera el descriptor de conexión.

Esto no afecta a la lectura anticipada de los mensajes para la aplicación. Si tiene previsto dejar de consumir mensajes durante un periodo largo, considere la posibilidad de cerrar la cola y volver a abrirla cuando el consumo deba continuar.

Si se emite desde dentro de una rutina de devolución de llamada, no entra en vigor hasta que se cierra la rutina. No se llamarán más rutinas de consumidor de mensajes después de las salidas de rutina actuales.

Si se emite fuera de una devolución de llamada, el control no vuelve al interlocutor hasta que se haya completado la rutina de consumidor actual y no se llame a más.

CTLRE

Reanude el consumo de mensajes.

Esta opción se emite normalmente desde la hebra de aplicación principal, pero también se puede utilizar desde una rutina de devolución de llamada para cancelar una solicitud de suspensión anterior emitida en la misma rutina.

Si se utiliza CTLRE para reanudar un CTLSW, la operación se bloquea.

PCTLOP (MQCTLO)-entrada

Opciones que controlan la acción de MQCTL

Consulte MQCTLO para obtener detalles de la estructura.

CMPCOD (entero con signo de 10 dígitos)-salida

El código de terminación; es uno de los siguientes:

CCOK

Realización satisfactoria.

CCWARN

Aviso (finalización parcial).

CCFAIL

La llamada no se ha realizado satisfactoriamente.

REASON (entero con signo de 10 dígitos)-salida

Los siguientes códigos de razón son los que el gestor de colas puede devolver para el parámetro **Reason**.

Si *CMPCOD* es CCOK:

RCNONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CMPCOD* es CCFail:

RC2133

(2133, X'855') No se han podido cargar módulos de servicio de conversión de datos.

RC2204

(2204, X'89C') El adaptador no está disponible.

RC2130

(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

RC2374

(2374, X'946') La salida de la API ha fallado.

RC2183

(2183, X'887') No se ha podido cargar la salida de la API.

RC2157

(2157, X'86D') Los ASID primario e inicial varían.

RC2005

(2005, X'7D5') El parámetro de longitud de almacenamiento intermedio no es válido.

RC2487

(2487, X'9B7') No se ha podido llamar a la rutina de devolución de llamada

RC2448

(2448, X' 990 ') No se puede anular el registro, suspender o reanudar porque no hay ninguna devolución de llamada registrada

RC2486

(2486, X'9B6') O bien se han especificado tanto CallbackFunction como CallbackName en una llamada CBREG, o bien se ha especificado una de CallbackFunction o CallbackName pero no coincide con la función de callback registrada actualmente.

RC2483

(2483, X'9B3') Campo CallBackType incorrecto.

RC2219

(2219, X'8AB') La llamada MQI ha entrado antes de completar la llamada anterior.

RC2444

(2444, X'98C') El bloque de opciones es incorrecto.

RC2484

(2484, X'9B4') Campo de opciones MQCBD incorrecto.

RC2140

(2140, X'85C') Solicitud de espera rechazada por ' CICS.

RC2009

(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

RC2217

(2217, X'8A9') No tiene autorización para la conexión.

RC2202

(2202, X'89A') Conexión en fase de inmovilización.

RC2203

(2203, X'89B') La conexión está concluyendo.

RC2207

(2207, X'89F') Error de identificador de correlación.

RC2016

(2016, X'7E0') Se han inhibido las obtenciones para la cola.

RC2351

(2351, X'92F') Conflicto de unidades de trabajo global.

RC2186

(2186, X'88A') No es válida la estructura de las opciones de obtención de mensaje.

RC2353

(2353, X'931') Descriptor de contexto que se utiliza para la unidad de trabajo global.

RC2018

(2018, X'7E2') El manejador de conexión no es válido.

RC2019

(2019, X'7E3') El manejador de objeto no es válido.

RC2259

(2259, X'8D3') La especificación de examinar es incoherente.

RC2245

(2245, X'8C5') Especificación incoherente de unidad de trabajo.

RC2246

(2246, X'8C6') El mensaje bajo cursor no es válido para recuperación.

RC2352

(2352, X'930') La unidad de trabajo global entra en conflicto con la unidad de trabajo local.

RC2247

(2247, X'8C7') Las opciones de coincidencia no son válidas.

RC2485

(2485, X'9B5') Campo MaxMsgLength incorrecto

RC2026

(2026, X'7EA') El descriptor de mensaje no es válido.

RC2497

(2497, X'9C1') El punto de entrada de la función especificada no se encontró en el módulo.

RC2496

(2496, X'9C0') Se ha encontrado el módulo, pero es del tipo incorrecto (32 o 64 bits) o no es una dll válida.

RC2495

(2495, X'9BF') Módulo no encontrado en la ruta de búsqueda o no autorizado a cargar.

RC2206

(2206, X'89E') Error de identificador de mensaje.

RC2250

(2250, X'8CA') El número de secuencia de mensaje no es válido.

RC2331

(2331, X'91B') El uso del símbolo de mensaje no es válido.

RC2036

(2036, X'7F4') La cola no se ha abierto para examen.

RC2037

(2037, X'7F5') La cola no se ha abierto para entrada.

RC2041

(2041, X'7F9') La definición de objeto ha cambiado desde que se ha abierto.

RC2101

(2101, X'835') El objeto se ha dañado.

RC2488

(2488, X'9B8') Código de operación incorrecto en la llamada API

RC2046

(2046, X'7FE') Las opciones no son válidas o no son coherentes.

RC2193

(2193, X'891') Error al acceder al conjunto de datos del conjunto de páginas.

RC2052

(2052, X'804') La cola se ha suprimido.

RC2394

(2394, X'95A') La cola tiene un tipo de índice incorrecto.

RC2058

(2058, X'80A') El nombre del gestor de colas no es válido o no se conoce.

RC2059

(2059, X'80B') El gestor de colas no está disponible para efectuar la conexión.

RC2161

(2161, X'871') El gestor de colas se está desactivando temporalmente.

RC2162

(2162, X'872') El gestor de colas está concluyendo.

RC2102

(2102, X'836') No hay disponibles suficientes recursos del sistema.

RC2069

(2069, X'815') Símbolo pendiente para este descriptor de contexto.

RC2071

(2071, X'817') No hay suficiente almacenamiento disponible.

RC2109

(2109, X'83D') El programa de salida ha suprimido la llamada.

RC2072

(2072, X'818 ') El soporte de punto de sincronización no está disponible.

RC2195

(2195, X'893') Se ha producido un error inesperado.

RC2354

(2354, X'932') Ha fallado el listado en la unidad de trabajo global.

RC2355

(2355, X'933') No se soporta la mezcla de llamadas de unidad de trabajo.

RC2255

(2255, X'8CF') La unidad de trabajo no está disponible para ser utilizada por el gestor de colas.

RC2090

(2090, X'82A') El intervalo de espera de MQGMO no es válido.

RC2256

(2256, X'8D0') Se ha suministrado una versión equivocada de MQGMO.

RC2257

(2257, X'8D1') La versión del MQMD suministrado es incorrecta.

RC2298

(2298, X'8FA') La función solicitada no está disponible en el entorno actual.

Declaración RPG

```

C*..1.....2.....3.....4.....5.....6.....7..
C                                CALLP      MQCTL(HCONN : OPERATN : PCTLOP :
                                           CMPCOD : REASON)

```

La definición de prototipo para la llamada es:

```

DMQCTL          PR              EXTPROC('MQCTL')
D* Connection handle
D HCONN          10I 0 VALUE
D* Operation
D OPERATN        10I 0 VALUE
D* Control options
D PCTLOP          32A
D* Completion code
D CMPCOD          10I 0
D* Reason code qualifying CompCode
D REASON          10I 0

```

MQDISC (Desconectar gestor de colas) en IBM i

La llamada MQDISC interrumpe la conexión entre el gestor de colas y el programa de aplicación, y es la inversa de la llamada MQCONN o MQCONNX.

- [“Sintaxis” en la página 1319](#)
- [“Notas de uso” en la página 1319](#)
- [“Parámetros” en la página 1320](#)
- [“Declaración RPG” en la página 1321](#)

Sintaxis

MQDISC (*HCONN*, *CMPCOD*, *REASON*)

Notas de uso

1. Si se emite una llamada MQDISC cuando la aplicación todavía tiene objetos abiertos, el gestor de colas cierra esos objetos, con las opciones de cierre establecidas en CONONE.
2. Si la aplicación finaliza con cambios no confirmados en una unidad de trabajo, la disposición de dichos cambios depende de cómo finalice la aplicación:
 - a. Si la aplicación emite la llamada MQDISC antes de finalizar:
 - Para una unidad de trabajo coordinada del gestor de colas, el gestor de colas emite la llamada MQCMIT en nombre de la aplicación. La unidad de trabajo se confirma si es posible y se restituye si no es así.

- Para una unidad de trabajo coordinada externamente, no hay ningún cambio en el estado de la unidad de trabajo; sin embargo, el gestor de colas indicará que la unidad de trabajo debe confirmarse, cuando lo solicite el coordinador de la unidad de trabajo.
- b. Si la aplicación finaliza normalmente pero sin emitir la llamada MQDISC, la unidad de trabajo se restituye.
- c. Si la aplicación finaliza *anormalmente* sin emitir la llamada MQDISC, la unidad de trabajo se restituye.

Parámetros

La llamada MQDISC tiene los parámetros siguientes:

HCONN (entero con signo de 10 dígitos)-entrada/salida

Descriptor de conexión.

Este manejador representa la conexión con el gestor de colas. El valor de *HCONN* ha sido devuelto por una llamada MQCONN o MQCONNEX anterior.

Al finalizar correctamente la llamada, el gestor de colas establece *HCONN* en un valor que no es un descriptor de contexto válido para el entorno. Este valor es:

HCUNUH

Descriptor de conexión inutilizable.

CMPCOD (entero con signo de 10 dígitos)-salida

Código de terminación.

Es uno de los siguientes:

CCOK

Realización satisfactoria.

CCWARN

Aviso (finalización parcial).

CCFAIL

La llamada no se ha realizado satisfactoriamente.

REASON (entero con signo de 10 dígitos)-salida

Código de razón que califica *CMPCOD*.

Si *CMPCOD* es CCOK:

RCNONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CMPCOD* es CCFAIL:

RC2219

(2219, X'8AB') Se ha vuelto a introducir la llamada « MQI » antes de que finalizara la llamada anterior.

RC2009

(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

RC2018

(2018, X'7E2') El manejador de conexión no es válido.

RC2058

(2058, X'80A') El nombre del gestor de colas no es válido o no se conoce.

RC2059

(2059, X'80B') El gestor de colas no está disponible para efectuar la conexión.

RC2162

(2162, X'872') El gestor de colas está concluyendo.

RC2102

(2102, X'836') No hay disponibles suficientes recursos del sistema.

RC2071

(2071, X'817') No hay suficiente almacenamiento disponible.

RC2195

(2195, X'893') Se ha producido un error inesperado.

Declaración RPG

```
C*.1.....2.....3.....4.....5.....6.....7..
C                                CALLP      MQDISC(HCONN : CMPCOD : REASON)
```

La definición de prototipo para la llamada es:

```
D*.1.....2.....3.....4.....5.....6.....7..
DMQDISC          PR          EXTPROC('MQDISC')
D* Connection handle
D HCONN                      10I 0
D* Completion code
D CMPCOD                      10I 0
D* Reason code qualifying CMPCOD
D REASON                      10I 0
```

MQDLTMH (Suprimir descriptor de mensaje) en IBM i

La llamada MQDLTMH suprime un manejador de mensajes y es el inverso de la llamada MQCRTMH.

- [“Sintaxis” en la página 1321](#)
- [“Notas de uso” en la página 1321](#)
- [“Parámetros” en la página 1323](#)
- [“Declaración RPG” en la página 1324](#)

Sintaxis

MQDLTMH ((*Hconn*, *Hmsg*, *DltMsgHOpts*, *CompCode*, *Reason*))

Notas de uso

1. Sólo puede utilizar esta llamada cuando el propio gestor de colas coordina la unidad de trabajo. Este puede ser:
 - Unidad de trabajo local, donde los cambios sólo afectan a los recursos de IBM MQ .
 - Unidad de trabajo global, donde los cambios pueden afectar a los recursos que pertenecen a otros gestores de recursos, así como a los recursos de IBM MQ .

Para obtener más detalles sobre las unidades de trabajo locales y globales, consulte [“MQBEGIN \(Inicio de la unidad de trabajo\) en IBM i” en la página 1282](#).
2. En entornos en los que el gestor de colas no coordina la unidad de trabajo, utilice la llamada de restitución adecuada en lugar de MQBACK. El entorno también puede dar soporte a una devolución implícita causada por la terminación anómala de la aplicación.
 - En z/OS, utilice las llamadas siguientes:
 - Los programas por lotes (incluidos los programas IMS batch DL/I) pueden utilizar la llamada MQBACK si la unidad de trabajo sólo afecta a los recursos de IBM MQ . Sin embargo, si la unidad de trabajo afecta tanto a los recursos de IBM MQ como a los recursos pertenecientes a otros gestores de recursos (por ejemplo, Db2), utilice la llamada SRRBACK proporcionada por el servicio de recursos recuperables (RRS) de z/OS . La llamada SRRBACK restituye los cambios en los recursos que pertenecen a los gestores de recursos que se han habilitado para la coordinación RRS.

- Las aplicaciones CICS deben utilizar el mandato EXEC CICS SYNCPOINT ROLLBACK para restituir la unidad de trabajo. No utilice la llamada MQBACK para aplicaciones CICS .
 - Las aplicaciones IMS (que no sean programas DL/I por lotes) deben utilizar llamadas IMS como ROLB para restituir la unidad de trabajo. No utilice la llamada MQBACK para aplicaciones IMS (que no sean programas DL/I por lotes).
 - En IBM i, utilice esta llamada para las unidades de trabajo locales coordinadas por el gestor de colas. Esto significa que no debe existir una definición de compromiso a nivel de trabajo, es decir, el mandato STRCMTCTL con el parámetro **CMTSCOPE(*JOB)** no debe haberse emitido para el trabajo.
3. Si una aplicación finaliza con cambios no confirmados en una unidad de trabajo, la disposición de dichos cambios depende de si la aplicación finaliza de forma normal o anómala. Consulte las notas de uso en [“MQDISC \(Desconectar gestor de colas\) en IBM i”](#) en la [página 1319](#) para obtener más detalles.
4. Cuando una aplicación coloca u obtiene mensajes en grupos o segmentos de mensajes lógicos, el gestor de colas conserva información relacionada con el grupo de mensajes y el mensaje lógico para las últimas llamadas MQPUT y MQGET satisfactorias. Esta información está asociada con el descriptor de contexto de cola e incluye cosas como:

- Los valores de los campos *GroupId*, *MsgSeqNumber*, *Offset* y *MsgFlags* en MQMD.
- Indica si el mensaje forma parte de una unidad de trabajo.
- Para la llamada MQPUT: si el mensaje es persistente o no persistente.

El gestor de colas mantiene tres conjuntos de información de grupo y segmento, un conjunto para cada uno de los siguientes:

- La última llamada MQPUT satisfactoria (puede formar parte de una unidad de trabajo).
- La última llamada MQGET satisfactoria que ha eliminado un mensaje de la cola (esto puede formar parte de una unidad de trabajo).
- La última llamada MQGET satisfactoria que ha examinado un mensaje en la cola (no puede formar parte de una unidad de trabajo).

Si la aplicación coloca u obtiene los mensajes como parte de una unidad de trabajo y, a continuación, la aplicación restituye la unidad de trabajo, la información de grupo y segmento se restaura al valor que tenía anteriormente:

- La información asociada con la llamada MQPUT se restaura al valor que tenía antes de la primera llamada MQPUT satisfactoria para ese manejador de cola en la unidad de trabajo actual.
- La información asociada con la llamada MQGET se restaura al valor que tenía antes de la primera llamada MQGET satisfactoria para ese manejador de cola en la unidad de trabajo actual.

Las colas actualizadas por la aplicación después de que se iniciara la unidad de trabajo, pero fuera del ámbito de la unidad de trabajo, no tienen la información de grupo y segmento restaurada si se restituye la unidad de trabajo.

La restauración de la información de grupo y segmento a su valor anterior cuando se restituye una unidad de trabajo permite a la aplicación distribuir un grupo de mensajes grande o un mensaje lógico grande que consta de muchos segmentos entre varias unidades de trabajo, y reiniciar en el punto correcto del grupo de mensajes o mensaje lógico si falla una de las unidades de trabajo. El uso de varias unidades de trabajo puede ser ventajoso si el gestor de colas local sólo tiene un almacenamiento de cola limitado. Sin embargo, la aplicación debe mantener suficiente información para poder reiniciar la colocación u obtención de mensajes en el punto correcto si se produce una anomalía del sistema.

Para obtener detalles sobre cómo reiniciar en el punto correcto después de una anomalía del sistema, consulte la opción PMLOGO descrita en [PMOPT \(entero con signo de 10 dígitos\)](#) y la opción GMLOGO descrita en [GMOPT \(entero con signo de 10 dígitos\)](#).

Las notas de uso restantes sólo se aplican cuando el gestor de colas coordina las unidades de trabajo:

5. Una unidad de trabajo tiene el mismo ámbito que un descriptor de conexión. Todas las llamadas de IBM MQ que afectan a una unidad de trabajo determinada se deben realizar utilizando el mismo descriptor de conexión. Las llamadas emitidas utilizando un descriptor de conexión diferente (por

ejemplo, las llamadas emitidas por otra aplicación) afectan a una unidad de trabajo diferente. Consulte HCONN (entero con signo de 10 dígitos)-salida para obtener información sobre el ámbito de los manejadores de conexión.

6. Esta llamada sólo afecta a los mensajes que se han colocado o recuperado como parte de la unidad de trabajo actual.
7. Una aplicación de larga ejecución que emite llamadas MQGET, MQPUT o MQPUT1 dentro de una unidad de trabajo, pero que nunca emite una llamada de confirmación o restitución, puede llenar las colas con mensajes que no están disponibles para otras aplicaciones. Para protegerse de esta posibilidad, el administrador debe establecer el atributo de gestor de colas **MaxUncommittedMsgs** en un valor lo suficientemente bajo como para evitar que las aplicaciones desbocadas llenen las colas, pero lo suficientemente alto como para permitir que las aplicaciones de mensajería esperadas funcionen correctamente.

Parámetros

La llamada MQDLTMH tiene los parámetros siguientes:

HCONN (entero con signo de 10 dígitos)-entrada

Este manejador representa la conexión con el gestor de colas.

El valor debe coincidir con el descriptor de conexión que se ha utilizado para crear el descriptor de mensaje especificado en el parámetro **HMSG**.

Si el descriptor de contexto de mensaje se ha creado utilizando HCUNAS, se debe establecer una conexión válida en la hebra suprimiendo el descriptor de contexto de mensaje; de lo contrario, la llamada falla con RC2009.

HMSG (entero con signo de 20 dígitos)-entrada/salida

Este es el descriptor de contexto de mensaje que se va a suprimir. El valor ha sido devuelto por una llamada MQCRTMH anterior.

Al finalizar correctamente la llamada, el descriptor de contexto se establece en un valor no válido para el entorno. Este valor es:

HMUNUH

Manejador de mensajes inutilizable.

El descriptor de mensaje no se puede suprimir si hay otra llamada IBM MQ en curso a la que se ha pasado el mismo descriptor de mensaje.

DLTOPT (MQDMHO)-entrada

Consulte MQDMHO para obtener más detalles.

CMPCOD (entero con signo de 10 dígitos)-salida

El código de terminación; es uno de los siguientes:

CCOK

Realización satisfactoria.

CCFAIL

La llamada no se ha realizado satisfactoriamente.

REASON (entero con signo de 10 dígitos)-salida

El código de razón que califica *CMPCOD*.

Si *CMPCOD* es CCOK:

RCNONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CMPCOD* es CCFAIL:

RC2204

(2204, X'089C') Adaptador no disponible.

RC2130

(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

RC2157

(2157, X'86D') Los ASID primario e inicial varían.

RC2219

(2219, X'08AB') Se ha iniciado una llamada al MQI antes de que finalizara la llamada anterior.

RC2009

(2009, X'07D9') Se ha perdido la conexión con el gestor de colas.

RC2462

(2462, X'099E') La estructura de las opciones del gestor de mensajes no es válida.

RC2460

(2460, X'099C') El puntero del gestor de mensajes no es válido.

RC2499

(2499, X'09C3') Manejador de mensaje ya en uso.

RC2046

(2046, X'07FE') Opciones no válidas o no coherentes.

RC2071

(2071, X'817') No hay suficiente almacenamiento disponible.

RC2195

(2195, X'893') Se ha producido un error inesperado.

Consulte [“Códigos de retorno para IBM i \(ILE RPG\)”](#) en la [página 1447](#) para obtener más detalles.

Declaración RPG

```

C*..1.....2.....3.....4.....5.....6.....7..
C                                CALLP      MQDLTMH(HCONN : HMSG : DLTOPT :
                                           CMPCOD : REASON)

```

La definición de prototipo para la llamada es:

```

DMQDLTMH          PR                EXTPROC('MQDLTMH')
D* Connection handle
D HCONN              10I 0 VALUE
D* Message handle
D HMSG              20I 0
D* Options that control the action of MQDLTMH
D DLTOPT             12A
D* Completion code
D CMPCOD             10I 0
D* Reason code qualifying CompCode
D REASON             10I 0

```

MQDLTMP-Suprimir propiedad de mensaje

La llamada MQDLTMP suprime una propiedad de un manejador de mensajes y es la inversa de la llamada MQSETMP.

- [“Sintaxis”](#) en la [página 1324](#)
- [“Parámetros”](#) en la [página 1325](#)
- [“Declaración RPG”](#) en la [página 1326](#)

Sintaxis

MQDLTMP (Hconn, Hmsg, DltPropOpts, Name, CompCode, Reason)

Parámetros

La llamada MQDLTMP tiene los parámetros siguientes:

HCONN (entero con signo de 10 dígitos)-Entrada

Este manejador representa la conexión con el gestor de colas. El valor debe coincidir con el descriptor de conexión que se ha utilizado para crear el descriptor de mensaje especificado en el parámetro **HMSG**.

Si el descriptor de mensaje se ha creado utilizando HCUNAS, se debe establecer una conexión válida en la hebra suprimiendo el descriptor de mensaje, de lo contrario la llamada fallará con RC2009.

HMSG (entero con signo de 20 dígitos)-entrada

Este es el descriptor de contexto de mensaje que contiene la propiedad que se va a suprimir. El valor ha sido devuelto por una llamada MQCRTMH anterior.

DLTOPT (MQDMPO)-Entrada

Consulte el tipo de datos [MQDMPO](#) para obtener detalles.

PRNAME (MQCHARV)-entrada

El nombre de la propiedad que se va a suprimir. Consulte [Nombres de propiedad](#) para obtener más información sobre los nombres de propiedad.

Los comodines no están permitidos en el nombre de propiedad.

CMPCOD (entero con signo de 10 dígitos)-salida

El código de terminación; es uno de los siguientes:

CCOK

Realización satisfactoria.

CCWARN

Aviso (finalización parcial).

CCFAIL

La llamada no se ha realizado satisfactoriamente.

REASON (entero con signo de 10 dígitos)-salida

El código de razón que califica *CMPCOD*.

Si *CMPCOD* es CCOK:

RCNONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CMPCOD* es CCWARN:

RC2471

(2471, X'09A7') Propiedad no disponible.

RC2421

(2421, X'0975 ') No se ha podido analizar una carpeta MQRFH2 que contiene propiedades.

Si *CMPCOD* es CCFAIL:

RC2204

(2204, X'089C') Adaptador no disponible.

RC2130

(2130, X'0852 ') No se puede cargar el módulo de servicio del adaptador.

RC2157

(2157, X'086D') Los ASID primario y de origen difieren.

RC2219

(2219, X'08AB') MQI : se ha iniciado una llamada antes de que finalizara la anterior.

RC2009

(2009, X'07D9') Se ha perdido la conexión con el gestor de colas.

RC2481

(2481, X'09B1') La estructura de opciones de la propiedad Borrar mensaje no es válida.

RC2460

(2460, X'099C') Manejador de mensaje no válido.

RC2499

(2499, X'09C3') Manejador de mensaje ya en uso.

RC2046

(2046, X'07FE') Opciones no válidas o no coherentes.

RC2442

(2442, X'098A') Nombre de propiedad no válido.

RC2111

(2111, X'083F') El identificador del juego de caracteres codificado del nombre de la propiedad no es válido.

RC2195

(2195, X'0893 ') Se ha producido un error inesperado.

Para obtener más información sobre estos códigos, consulte [Códigos de terminación y razón de la API](#).

Declaración RPG

```
C*.1.....2.....3.....4.....5.....6.....7..
C                                CALLP      MQDLTMP(HCONN : HMSG : DLTOPT :
                                PRNAME : CMPCOD : REASON)
```

La definición de prototipo para la llamada es:

```
DMQDLTMP          PR          EXTPROC('MQDLTMP')
D* Connection handle
D HCONN           10I 0 VALUE
D* Message handle
D HMSG           20I 0 VALUE
D* Options that control the action of MQDLTMP
D DLTOPT          12A
D* Property name
D PRNAME          32A
D* Completion code
D CMPCOD          10I 0
D* Reason code qualifying CompCode
D REASON          10I 0
```

MQGET (Obtener mensaje) en IBM i

La llamada MQGET recupera un mensaje de una cola local que se ha abierto utilizando la llamada MQOPEN.

- [“Sintaxis” en la página 1326](#)
- [“Notas de uso” en la página 1327](#)
- [“Parámetros” en la página 1330](#)
- [“Declaración RPG” en la página 1334](#)

Sintaxis

MQGET (*HCONN*, *HOBJ*, *MSGDSC*, *GMO*, *BUFLEN*, *BUFFER*, *DATLEN*, *CMPCOD*, *REASON*)

Notas de uso

1. El mensaje recuperado se suprime normalmente de la cola. Esta supresión puede producirse como parte de la propia llamada MQGET o como parte de un punto de sincronización. La supresión de mensajes no se produce si se especifica una opción GMBRWF o GMBRWN en el parámetro **GMO** (consulte el campo *GMOPT* descrito en [“MQGMO \(Opciones de obtención de mensajes\) activado IBM i”](#) en la página 1096).
2. Si se especifica la opción GMLK con una de las opciones de examen, el mensaje examinado se bloquea para que sea visible sólo para este descriptor de contexto.

Si se especifica la opción GMUNLK, se desbloquea un mensaje bloqueado anteriormente. No se recupera ningún mensaje en este caso y los parámetros **MSGDSC**, **BUFLN**, **BUFFER** y **DATLEN** no se comprueban ni modifican.

3. Si la aplicación que emite la llamada MQGET se ejecuta como un IBM MQ MQI client, es posible que el mensaje recuperado se pierda si durante el proceso de la llamada MQGET el IBM MQ MQI client termina de forma anómala o se interrumpe la conexión de cliente. Esto se debe a que el sustituto que se ejecuta en la plataforma del gestor de colas y que emite la llamada MQGET en nombre del cliente no puede detectar la pérdida del cliente hasta que el sustituto está a punto de devolver el mensaje al cliente; esto es después de que el mensaje se haya eliminado de la cola. Esto puede ocurrir tanto para mensajes persistentes como para mensajes no persistentes.

El riesgo de perder mensajes de esta forma se puede eliminar recuperando siempre mensajes dentro de las unidades de trabajo (es decir, especificando la opción GMSYP en la llamada MQGET y utilizando las llamadas MQCMIT o MQBACK para confirmar o restituir la unidad de trabajo cuando se complete el proceso del mensaje). Si se especifica GMSYP y el cliente termina de forma anómala o se interrumpe la conexión, el sustituto restituye la unidad de trabajo en el gestor de colas y el mensaje se restablece en la cola.

En principio, se puede producir la misma situación con las aplicaciones que se están ejecutando en la plataforma del gestor de colas, pero en este caso la ventana durante la que se puede perder un mensaje es pequeña. Sin embargo, al igual que con IBM MQ MQI clients , el riesgo se puede eliminar recuperando el mensaje dentro de una unidad de trabajo.

4. Si una aplicación pone una secuencia de mensajes en una determinada cola dentro de una única unidad de trabajo y, a continuación, confirma esa unidad de trabajo de forma satisfactoria, los mensajes están disponibles para la recuperación tal como se indica a continuación:
 - Si la cola es una *cola no compartida* (es decir, una cola local), todos los mensajes de la unidad de trabajo pasan a estar disponibles al mismo tiempo.
 - Si la cola es una *cola compartida*, los mensajes de la unidad de trabajo pasan a estar disponibles en el orden en el que se colocaron, pero no todos al mismo tiempo. Cuando el sistema está muy cargado, es posible que el primer mensaje de la unidad de trabajo se recupere correctamente, pero que la llamada MQGET para el segundo o posterior mensaje de la unidad de trabajo falle con RC2033. Si esto ocurre, la aplicación debe esperar un poco y luego volver a intentar la operación.
5. Si una aplicación coloca una secuencia de mensajes en la misma cola sin utilizar grupos de mensajes, el orden de dichos mensajes se conserva si se cumplen determinadas condiciones. Consulte las notas de uso en la descripción de la llamada MQPUT para obtener detalles. Si las condiciones se cumplen, los mensajes se presentan a la aplicación receptora en el orden en que fueron enviados, si:
 - Sólo un receptor está obteniendo mensajes de la cola.

Si hay dos o más aplicaciones que obtienen mensajes de la cola, debe ponerse de acuerdo con el remitente sobre el mecanismo que se utilizará para identificar los mensajes que pertenecen a una secuencia. Por ejemplo, el remitente puede establecer todos los campos MDCID de los mensajes de una secuencia en un valor que sea exclusivo para esa secuencia de mensajes.

- El receptor no cambia deliberadamente el orden de recuperación, por ejemplo, especificando un MDMID o MDCID determinado.

Si la aplicación emisora coloca los mensajes como un grupo de mensajes, los mensajes se presentan a la aplicación receptora en el orden correcto si la aplicación receptora especifica la opción GMLOGO en la llamada MQGET. Para obtener más información sobre los grupos de mensajes, consulte:

- MDMFL Campo de MQMD
 - Opción PMLOGO en MQPMO
 - Opción GMLOGO en MQGMO
6. Las aplicaciones prueban el código de comentarios FBQUIT en el campo MDFB del parámetro **MSGDSC**. Si se encuentra este valor, la aplicación finaliza. Consulte el campo MDFB descrito en [“MQMD \(descriptor de mensaje\) en IBM i”](#) en la página 1131 para obtener más información.
 7. Si la cola identificada por H0BJ se ha abierto con la opción OOSAVA, y el código de finalización de la llamada MQGET es CCOK o CCWARN, el contexto asociado con el descriptor de contexto de cola H0BJ se establece en el contexto del mensaje que se ha recuperado (a menos que se establezca la opción GMBRWF o GMBRWN, en cuyo caso el contexto se marca como no disponible). Este contexto se puede utilizar en una llamada MQPUT o MQPUT1 posterior especificando las opciones PMPASI o PMPASA. Esto permite que el contexto del mensaje recibido se transfiera en su totalidad o en parte a otro mensaje (por ejemplo, cuando el mensaje se reenvía a otra cola). Para obtener más información sobre el contexto de mensaje, consulte [Contexto de mensaje y Control de la información de contexto](#).
 8. Si la opción GMCONV se incluye en el parámetro **GMO**, los datos del mensaje de aplicación se convierten a la representación solicitada por la aplicación receptora, antes de que los datos se coloquen en el parámetro **BUFFER**:
 - El campo MDFMT de la información de control del mensaje identifica la estructura de los datos de aplicación y los campos MDCSI y MDENC de la información de control del mensaje especifican su identificador y codificación de juego de caracteres.
 - La aplicación que emite la llamada MQGET especifica en los campos MDCSI y MDENC del parámetro **MSGDSC** el identificador de juego de caracteres y la codificación a los que se deben convertir los datos del mensaje de aplicación.

Cuando es necesaria la conversión de los datos del mensaje, la conversión la realiza el propio gestor de colas o una salida escrita por el usuario, en función del valor del campo MDFMT en la información de control del mensaje:

- El gestor de colas convierte automáticamente los formatos siguientes; estos formatos se denominan formatos "incorporados":

FMADMN	FMMDE
FMCICS	FMPCF
FMCM1	HMRM
FMCM2	FMRFH
FMDLH	FMRFH2
FMDH	FMSTR
FMEVNT	FMTM
FMIMS	FMXQH
FMIMVS	

- El nombre de formato FMNONE es un valor especial que indica que la naturaleza de los datos del mensaje no está definida. Como consecuencia, el gestor de colas no intenta la conversión cuando se recupera el mensaje de la cola.

Nota: Si se especifica GMCONV en la llamada MQGET para un mensaje que tiene un nombre de formato FMNONE, y el juego de caracteres o la codificación del mensaje difiere del especificado en el parámetro **MSGDSC**, el mensaje se devuelve en el parámetro **BUFFER** (suponiendo que no haya otros errores), pero la llamada se completa con el código de terminación CCWARN y el código de razón RC2110.

FMNONE puede utilizarse cuando la naturaleza de los datos del mensaje significa que no requiere conversión, o cuando las aplicaciones de envío y recepción han acordado entre sí el formulario en el que deben enviarse los datos del mensaje.

- Todos los demás nombres de formato hacen que el mensaje se pase a una salida escrita por el usuario para su conversión. La salida tiene el mismo nombre que el formato, aparte de las adiciones específicas del entorno. Los nombres de formato especificados por el usuario no deben empezar por las letras "MQ", ya que estos nombres pueden entrar en conflicto con los nombres de formato soportados en el futuro.

Los datos de usuario del mensaje se pueden convertir entre cualquier juego de caracteres y codificación admitidos. Sin embargo, tenga en cuenta que si el mensaje contiene una o más estructuras de cabecera IBM MQ, el mensaje no se puede convertir de o a un juego de caracteres que tenga caracteres de doble byte o de varios bytes para cualquiera de los caracteres válidos en los nombres de cola. El código de razón RC2111 o RC2115 da como resultado si se intenta y el mensaje se devuelve sin convertir. El juego de caracteres Unicode UTF-16 es un ejemplo de este tipo de juego de caracteres.

En la devolución de la llamada MQGET, el código de razón siguiente indica que el mensaje se ha convertido correctamente:

- RCNONE

El siguiente código de razón indica que el mensaje puede haberse convertido correctamente; la aplicación debe comprobar los campos MDCSI y MDENC en el parámetro **MSGDSC** para averiguarlo:

- RC2079

Los demás códigos de razón indican que el mensaje no se ha convertido.

Nota: La interpretación del código de razón descrito en este ejemplo es verdadera para las conversiones realizadas por salidas escritas por el usuario sólo si la salida se ajusta a las directrices de proceso.

9. Para los formatos incorporados listados anteriormente, el gestor de colas puede realizar la conversión predeterminada de series de caracteres en el mensaje cuando se especifica la opción GMCONV. La conversión predeterminada permite que, para convertir datos de series de caracteres, el gestor de colas utilice un juego de caracteres predeterminado especificado por la instalación que se aproxima al juego de conjunto de caracteres real. Como resultado, la llamada MQGET puede ejecutarse correctamente con el código de terminación CCOK, en lugar de completarse con CCWARN y el código de razón RC2111 o RC2115.

Nota: El resultado de utilizar un juego de caracteres aproximado para convertir datos de series de caracteres es que es posible que algunos se conviertan de forma incorrecta. Esto se puede evitar utilizando en la serie sólo caracteres que son comunes tanto al juego de caracteres real como al juego de caracteres predeterminado.

La conversión predeterminada se aplica a los datos de los mensajes de la aplicación y a los campos de tipo carácter de las estructuras MQMD y MQMDE:

- La conversión predeterminada de los datos de mensaje de aplicación sólo se produce cuando se cumplen todas las sentencias siguientes:
 - La aplicación especifica GMCONV.
 - El mensaje contiene datos que se deben convertir de o a un juego de caracteres que no está soportado.
 - La conversión predeterminada no estaba activada cuando se ha instalado o reiniciado el gestor de colas.
- La conversión predeterminada de los campos de caracteres de las estructuras MQMD y MQMDE se produce según convenga, si dicha conversión predeterminada está habilitada para el gestor de colas. La conversión se realiza incluso si la aplicación no especifica la opción GMCONV en la llamada MQGET.

10. El parámetro **BUFFER** que se muestra en el ejemplo de programación RPG se declara como una serie; esto restringe la longitud máxima del parámetro a 256 bytes. Si se necesita un almacenamiento intermedio más grande, el parámetro debe declararse en su lugar como una estructura o como un campo en un archivo físico.

Declarar el parámetro como una estructura aumenta la longitud máxima posible a 9999 bytes, mientras que declarar el parámetro como un campo en un archivo físico aumenta la longitud máxima posible a aproximadamente 32 KB.

Parámetros

La llamada MQGET tiene los parámetros siguientes:

HCONN (entero con signo de 10 dígitos)-entrada

Descriptor de conexión.

Este manejador representa la conexión con el gestor de colas. El valor de HCONN ha sido devuelto por una llamada MQCONN o MQCONNEX anterior.

HOBJ (entero con signo de 10 dígitos)-entrada

El manejador del objeto.

Este manejador representa la cola de la que debe recuperarse el mensaje. El valor de HOBJ ha sido devuelto por una llamada MQOPEN anterior. La cola debe haberse abierto con una o más de las opciones siguientes (consulte [“MQOPEN \(Abrir objeto\) en IBM i”](#) en la página 1351 para obtener más detalles):

- OOINPS
- OOINPX
- OOINPQ
- OOBROW

MSGDSC (MQMD)-entrada/salida

Descriptor de mensaje.

Esta estructura describe los atributos del mensaje necesario y los del mensaje recuperado. Consulte [“MQMD \(descriptor de mensaje\) en IBM i”](#) en la página 1131 para obtener los detalles.

Si BUFLen es menor que la longitud del mensaje, el gestor de colas sigue especificando MSGDSC , si se ha especificado GMATM en el parámetro **GMO** (consulte el campo GMOPT descrito en [“MQGMO \(Opciones de obtención de mensajes\) activado IBM i”](#) en la página 1096).

Si la aplicación proporciona un MQMD version-1 , el mensaje devuelto tiene un MQMDE con el prefijo de los datos del mensaje de aplicación, pero sólo si uno o varios de los campos de MQMDE tienen un valor no predeterminado. Si todos los campos de MQMDE tienen valores predeterminados, se omite MQMDE. Un nombre de formato FMMDE en el campo MDFMT en MQMD indica que hay un MQMDE presente.

GMO (MQGMO)-entrada/salida

Opciones que controlan la acción de MQGET.

Consulte [“MQGMO \(Opciones de obtención de mensajes\) activado IBM i”](#) en la página 1096 para obtener los detalles.

BUFLen (entero con signo de 10 dígitos)-entrada

Longitud en bytes del área BUFFER .

Se puede especificar cero para los mensajes que no tienen datos, o si el mensaje se va a eliminar de la cola y los datos descartados (en este caso, se debe especificar GMATM).

Nota: La longitud del mensaje más largo que es posible leer de la cola la proporciona el atributo de cola **MaxMsgLength** ; consulte [“Atributos para colas”](#) en la página 1399.

BUFFER (serie de bits de 1 byte x BUFLen)-salida

Área para contener los datos del mensaje.

El almacenamiento intermedio debe estar alineado en un límite adecuado a la naturaleza de los datos del mensaje. La alineación de 4 bytes debe ser adecuada para la mayoría de los mensajes (incluidos los mensajes que contienen estructuras de cabecera IBM MQ), pero algunos mensajes pueden requerir una alineación más estricta. Por ejemplo, un mensaje que contiene un entero binario de 64 bits puede requerir una alineación de 8 bytes.

Si BUFLen es menor que la longitud del mensaje, la mayor parte posible del mensaje se mueve a BUFFER; esto sucede si se especifica GMATM en el parámetro **GMO** (consulte el campo GMOPT descrito en [“MQGMO \(Opciones de obtención de mensajes\) activado IBM i”](#) en la página 1096 para obtener más información).

El juego de caracteres y la codificación de los datos en **BUFFER** se proporcionan mediante los campos MDCSI y MDENC devueltos en el parámetro **MSGDSC**. Si estos valores son distintos de los valores que necesita el destinatario, éste deberá convertir los datos de mensajes de la aplicación en el juego de caracteres y en la codificación necesarios. La opción GMCONV se puede utilizar con una salida escrita por el usuario para realizar la conversión de los datos del mensaje (consulte [“MQGMO \(Opciones de obtención de mensajes\) activado IBM i”](#) en la página 1096 para obtener detalles de esta opción).

Nota: Todos los demás parámetros de la llamada MQGET están en el juego de caracteres y la codificación del gestor de colas local (proporcionados por el atributo de gestor de colas **CodedCharSetId** y ENNAT).

Si falla la llamada, el contenido del almacenamiento también se podría haber modificado.

DATLEN (entero con signo de 10 dígitos)-salida

Longitud del mensaje.

Es la longitud en bytes de los datos de aplicación del mensaje. Si esta longitud de mensaje es mayor que BUFLen, sólo se devuelven BUFLen bytes en el parámetro **BUFFER** (es decir, el mensaje se trunca). Si el valor es cero, significa que el mensaje no contiene datos de aplicación.

Si BUFLen es menor que la longitud del mensaje, el gestor de colas sigue especificando DATLEN, si se ha especificado GMATM en el parámetro **GMO** (consulte el campo GMOPT descrito en [“MQGMO \(Opciones de obtención de mensajes\) activado IBM i”](#) en la página 1096 para obtener más información). Esto permite a la aplicación determinar el tamaño del almacenamiento intermedio necesario para introducir los datos del mensaje y después reemitir la llamada con un almacenamiento intermedio que tenga el tamaño adecuado.

Sin embargo, si se especifica la opción GMCONV y los datos de mensaje convertidos son demasiado largos para caber en BUFFER, el valor devuelto para DATLEN es:

- La longitud de los datos no convertidos, para los formatos definidos por el gestor de colas.

En este caso, si la naturaleza de los datos hace que se expanda durante la conversión, la aplicación debe asignar un almacenamiento intermedio mayor que el valor devuelto por el gestor de colas para DATLEN.

- El valor devuelto por la salida de conversión de datos, para formatos definidos por la aplicación.

CMPCOD (entero con signo de 10 dígitos)-salida

Código de terminación.

Es uno de los siguientes:

CCOK

Realización satisfactoria.

CCWARN

Aviso (finalización parcial).

CCFAIL

La llamada no se ha realizado satisfactoriamente.

REASON (entero con signo de 10 dígitos)-salida

Código de razón que califica CMPCOD.

Los siguientes códigos de razón son los que el gestor de colas puede devolver para el parámetro **REASON**. Si la aplicación especifica la opción GMCONV y se invoca una salida escrita por el usuario para convertir algunos o todos los datos del mensaje, es la salida la que decide qué valor se devuelve para el parámetro **REASON**. Como resultado, los valores que no sean los documentados más adelante en esta sección son posibles.

Si CMPCOD es CCOK:

RCNONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si CMPCOD es CCWARN:

RC2120

(2120, X'848') Los datos convertidos son demasiado grandes para el almacenamiento intermedio.

RC2190

(2190, X'88E') La serie convertida es demasiado grande para el campo.

RC2150

(2150, X'866') La serie DBCS no es válida.

RC2110

(2110, X'83E') El formato de mensaje no válido.

RC2243

(2243, X'8C3') Los segmentos del mensaje tienen distintos CCSID.

RC2244

(2244, X'8C4') Los segmentos del mensaje tienen distintas codificaciones.

RC2209

(2209, X'8A1') No hay ningún mensaje bloqueado.

RC2119

(2119, X'847') Los datos del mensaje no se han convertido.

RC2272

(2272, X'8E0') Los datos del mensaje se han convertido parcialmente.

RC2145

(2145, X'861') El parámetro de almacenamiento intermedio de origen no es válido.

RC2111

(2111, X'83F') El identificador de juego de caracteres codificado origen no es válido.

RC2113

(2113, X'841') Codificación de decimal empaquetado en el mensaje no reconocida.

RC2114

(2114, X'842') Codificación de coma flotante en el mensaje no reconocida.

RC2112

(2112, X'840') Codificación de entero de origen no reconocida.

RC2143

(2143, X'85F') Parámetro de longitud de origen no válido.

RC2146

(2146, X'862') Parámetro de almacenamiento intermedio de destino no válido.

RC2115

(2115, X'843') El identificador de juego de caracteres codificado de destino no es válido.

RC2117

(2117, X'845') Codificación de decimal empaquetado especificada por receptor no reconocida.

RC2118

(2118, X'846') Codificación de coma flotante especificada por receptor no reconocida.

RC2116

(2116, X'844') Codificación de entero de destino no reconocida.

RC2079

(2079, X'81F') Se ha devuelto un mensaje truncado (el proceso se ha completado).

RC2080

(2080, X'820') Se ha devuelto un mensaje truncado (el proceso no se ha completado).

Si CMPCOD es CCFAIL:

RC2004

(2004, X'7D4') El parámetro almacenamiento intermedio no es válido.

RC2005

(2005, X'7D5') El parámetro de longitud de almacenamiento intermedio no es válido.

RC2219

(2219, X'8AB') Se ha vuelto a introducir la llamada « MQI » antes de que finalizara la llamada anterior.

RC2009

(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

RC2010

(2010, X'7DA') El parámetro longitud de datos no es válido.

RC2016

(2016, X'7E0') Se han inhibido las obtenciones para la cola.

RC2186

(2186, X'88A') No es válida la estructura de las opciones de obtención de mensaje.

RC2018

(2018, X'7E2') El manejador de conexión no es válido.

RC2019

(2019, X'7E3') El manejador de objeto no es válido.

RC2241

(2241, X'8C1') El grupo de mensajes no está completo.

RC2242

(2242, X'8C2') El mensaje lógico no está completo.

RC2259

(2259, X'8D3') La especificación de examinar es incoherente.

RC2245

(2245, X'8C5') Especificación incoherente de unidad de trabajo.

RC2246

(2246, X'8C6') El mensaje bajo cursor no es válido para recuperación.

RC2247

(2247, X'8C7') Las opciones de coincidencia no son válidas.

RC2026

(2026, X'7EA') El descriptor de mensaje no es válido.

RC2250

(2250, X'8CA') El número de secuencia de mensaje no es válido.

RC2033

(2033, X'7F1') No hay ningún mensaje disponible.

RC2034

(2034, X'7F2') El cursor para examinar no está situado en el mensaje.

RC2036

(2036, X'7F4') La cola no se ha abierto para examen.

RC2037

(2037, X'7F5') La cola no se ha abierto para entrada.

RC2041

(2041, X'7F9') La definición de objeto ha cambiado desde que se ha abierto.

RC2101

(2101, X'835') El objeto se ha dañado.

RC2046

(2046, X'7FE') Las opciones no son válidas o no son coherentes.

RC2052

(2052, X'804') La cola se ha suprimido.

RC2058

(2058, X'80A') El nombre del gestor de colas no es válido o no se conoce.

RC2059

(2059, X'80B') El gestor de colas no está disponible para efectuar la conexión.

RC2161

(2161, X'871') El gestor de colas se está desactivando temporalmente.

RC2162

(2162, X'872') El gestor de colas está concluyendo.

RC2102

(2102, X'836') No hay disponibles suficientes recursos del sistema.

RC2071

(2071, X'817') No hay suficiente almacenamiento disponible.

RC2024

(2024, X'7E8') No pueden manejarse más mensajes dentro de la unidad de trabajo actual.

RC2072

(2072, X'818 ') El soporte de punto de sincronización no está disponible.

RC2195

(2195, X'893') Se ha producido un error inesperado.

RC2255

(2255, X'8CF') La unidad de trabajo no está disponible para ser utilizada por el gestor de colas.

RC2090

(2090, X'82A') El intervalo de espera de MQGMO no es válido.

RC2256

(2256, X'8D0') Se ha suministrado una versión equivocada de MQGMO.

RC2257

(2257, X'8D1') La versión del MQMD suministrado es incorrecta.

Declaración RPG

```

C*.1.....2.....3.....4.....5.....6.....7..
C                                CALLP      MQGET(HCONN : HOBJ : MSGDSC : GMO :
C                                BUFLN : BUFFER : DATLEN :
C                                CMPCOD : REASON)

```

La definición de prototipo para la llamada es:

```

D*.1.....2.....3.....4.....5.....6.....7..
D*MQGET          PR          EXTPROC('MQGET')
D* Connection handle
D HCONN          10I 0 VALUE
D* Object handle
D HOBJ          10I 0 VALUE
D* Message descriptor
D MSGDSC          364A
D* Options that control the action of MQGET
D GMO          112A

```

```

D* Length in bytes of the Buffer area
D BUFLN      10I 0 VALUE
D* Area to contain the message data
D BUFFER      *   VALUE
D* Length of the message
D DATLEN      10I 0
D* Completion code
D CMPCOD      10I 0
D* Reason code qualifying CMPCOD
D REASON      10I 0

```

IBM i MQINQ (Consultar atributos de objeto) en IBM i

La llamada MQINQ devuelve una matriz de números enteros y un conjunto de cadenas de caracteres que contienen los atributos de un objeto.

Los siguientes tipos de objetos son válidos:

- Cola
- Lista de nombres
- Definición de proceso
- Gestor de colas
- [“Sintaxis” en la página 1335](#)
- [“Notas de uso” en la página 1335](#)
- [“Parámetros” en la página 1336](#)
- [“Declaración RPG” en la página 1343](#)

Sintaxis

MQINQ (*HCONN*, *HOBJ*, *SELCNT*, *SELS*, *IACNT*, *INTATR*, *CALEN*, *CHRATR*, *CMPCOD*, *REASON*)

Notas de uso

1. Los valores devueltos son una instantánea de los atributos seleccionados. No hay garantía de que los atributos no se modifiquen antes de que la aplicación pueda actuar sobre los valores devueltos.
2. Cuando se abre una cola de modelos, se crea una cola local dinámica. Esto es así incluso si se abre la cola del modelo para consultar sus atributos.

Los atributos de la cola dinámica (con ciertas excepciones) son los mismos que los de la cola modelo en el momento en que se crea la cola dinámica. Si luego utiliza la llamada MQINQ en esta cola, el gestor de colas devuelve los atributos de la cola dinámica, y no los de la cola modelo. Consulte la [Tabla 1](#) para obtener más información sobre qué atributos de la cola modelo se heredan en la cola dinámica.

3. Si el objeto consultado es una cola alias, los valores de atributo devueltos por la llamada MQINQ son los de la cola alias, y no los de la cola base a la que resuelve el alias.
4. Si el objeto consultado es una cola de clúster, los atributos que se pueden consultar dependen de cómo se abra la cola:
 - Si la cola del clúster se abre para consultar más uno o más de los siguientes: entrada, exploración o configuración, debe existir una instancia local de la cola del clúster para que la apertura se realice correctamente. En este caso, los atributos que se pueden consultar son los válidos para las colas locales.
 - Si la cola del clúster se abre solo para consulta, o para consulta y salida, solo se pueden consultar los siguientes atributos; el **QType** atributo tiene el valor QTCLUS en este caso:
 - CAQD
 - CAQN
 - IADBND
 - IADPER

- IADPRI
- IAIPUT
- IAQTYP

Si la cola del clúster se abre sin un enlace fijo (es decir, se especifica OOBNDN en la llamada MQOPEN o se especifica OOBNDQ cuando el **DefBind** atributo tiene el valor BNDNOT), las llamadas MQINQ sucesivas para la cola pueden consultar diferentes instancias de la cola del clúster, aunque normalmente todas las instancias tienen los mismos valores de atributo.

Para obtener más información sobre las colas de clúster, consulte [Configuración de un clúster de gestores de colas](#).

5. Si se deben consultar varios atributos y, a continuación, algunos de ellos se deben establecer mediante la llamada MQSET, puede resultar conveniente colocar al principio de las matrices del selector los atributos que se deben establecer, de modo que se puedan utilizar las mismas matrices (con recuentos reducidos) para MQSET.
6. Si se produce más de una de las situaciones de advertencia (véase el **CMPCOD** parámetro), el código de motivo devuelto es el *primero* de la siguiente lista que sea aplicable:
 - a. RC2068
 - b. RC2022
 - c. RC2008
7. Para obtener más información sobre los atributos de los objetos, consulte:
 - [“Atributos para colas” en la página 1399](#)
 - [“Atributos de las listas de nombres” en la página 1429](#)
 - [“Atributos para definiciones de proceso en IBM i” en la página 1430](#)
 - [“Atributos del gestor de colas en IBM i” en la página 1432](#)
8. Se utiliza una nueva cola local SYSTEM.ADMIN.COMMAND.EVENT para poner en cola los mensajes que se generan cada vez que se emiten comandos. Los mensajes se colocan en esta cola para la mayoría de los comandos, dependiendo de cómo esté configurado el atributo del gestor de colas CMDEV:
 - HABILITADO: se generan mensajes de evento de comando y se colocan en la cola para todos los comandos correctos.
 - NODISPLAY: se generan mensajes de evento de comando y se colocan en la cola para todos los comandos correctos, excepto el comando DISPLAY (MQSC) y el comando Inquire (PCF).
 - DESACTIVADO: no se generan mensajes de evento de comando (este es el valor predeterminado inicial del gestor de colas).

Parámetros

La llamada MQINQ tiene los siguientes parámetros:

HCONN (entero con signo de 10 dígitos) - entrada

Descriptor de conexión.

Este manejador representa la conexión con el gestor de colas. El valor de *HCONN* fue devuelto por una llamada anterior a MQCONN o MQCONNX.

HOBJ (entero con signo de 10 dígitos) - entrada

El manejador del objeto.

Este identificador representa el objeto (de cualquier tipo) con los atributos requeridos. El identificador debe haber sido devuelto por una llamada MQOPEN anterior que especificara la opción OOINQ.

SELCNT (entero con signo de 10 dígitos) - entrada

Recuento de selectores.

Este es el recuento de selectores que se proporcionan en la *SELS* matriz. Es el número de atributos que se van a devolver. Cero es un valor válido. El número máximo permitido es 256.

SELS (entero con signo de 10 dígitos x SELCNT) - entrada

Matriz de selectores de atributos.

Se trata de una matriz de **SELCNT** selectores de atributos; cada selector identifica un atributo (entero o carácter) con un valor que es obligatorio.

Cada selector debe ser válido para el tipo de objeto que *HOBJ* representa; de lo contrario, la llamada falla con el código de finalización CCFAIL y el código de motivo RC2067.

En el caso especial de las colas:

- Si el selector no es válido para colas de *ningún* tipo, la llamada falla con el código de finalización CCFAIL y el código de motivo RC2067.
- Si el selector solo es aplicable a colas de un tipo o tipos distintos al del objeto, la llamada se realiza correctamente con el código de finalización CCWARN y el código de motivo RC2068.
- Si la cola consultada es una cola de clúster, los selectores válidos dependen de cómo se haya resuelto la cola; consulte la nota de uso 4 para obtener más detalles.

Los selectores se pueden especificar en cualquier orden. Los valores de los atributos que corresponden a selectores de atributos enteros (selectores IA*) se devuelven en *INTATR* en el mismo orden en que aparecen estos selectores en *SELS*. Los valores de los atributos que corresponden a los selectores de atributos de caracteres (selectores CA*) se devuelven en *CHRATR* el mismo orden en que aparecen dichos selectores. Los selectores IA* pueden intercalarse con los selectores CA*; solo es importante el orden relativo dentro de cada tipo.

Nota:

1. Los selectores de atributos de enteros y caracteres se asignan dentro de dos rangos diferentes; los selectores IA* residen dentro del rango IAFRST a IALAST, y los selectores CA* dentro del rango CAFRST a CALAST.

Para cada rango, las constantes IALSTU y CALSTU definen el valor más alto que acepta el gestor de colas.

2. Si todos los selectores IA* aparecen primero, se pueden utilizar los mismos números de elemento para direccionar los elementos correspondientes en las matrices *SELS INTATR* y.

Los atributos que se pueden consultar se enumeran en las siguientes tablas. Para los selectores CA*, la constante que define la longitud en bytes de la cadena resultante en *CHRATR* se indica entre paréntesis.

<i>Tabla 785. Selektores de atributos MQINQ para colas</i>		
Selector	Descripción	Nota
CAALTD	Fecha de la última modificación (LNDATE).	1
CAALTT	Hora de la última modificación (LNTIME).	1
CABRQN	Nombre de cola de espera excesiva (LNQN).	5
CABASQ	Nombre de la cola a la que resuelve el alias (LNQN).	
CACFSN	Nombre de la estructura de la instalación de acoplamiento (LNCFSN).	3
CACLN	Nombre del clúster (LNCLUN).	1
CACLNL	Lista de nombres de clústeres (LNNLN).	1
CACRTD	Fecha de creación de la cola (LNCRTD).	
CACRTT	Tiempo de creación de la cola (LNCRTT).	
CAINIQ	Nombre de la cola de iniciación (LNQN).	

Tabla 785. Selektores de atributos MQINQ para colas (continuación)		
Selector	Descripción	Nota
CAPRON	Nombre de la definición del proceso (LNPRON).	
CAQD	Descripción de la cola (LNQD).	
CAQN	Nombre de la cola (LNQN).	
CARQMN	Nombre del gestor de colas remotas (LNQMN).	
CARQN	Nombre de la cola remota tal y como se conoce en el gestor de colas remotas (LNQN).	
CATRGD	Datos desencadenantes (LNTRGD).	5
CAXQN	Nombre de la cola de transmisión (LNQN).	
IABTHR	El umbral de restituciones.	5
IACDEP	Número de mensajes en la cola.	
IADBND	Enlace predeterminado.	1
IADINP	Opción predeterminada abierta para entrada.	5
IADPER	Persistencia de mensajes predeterminada.	
IADPRI	Prioridad de mensaje predeterminada.	5
IADEFT	El tipo de definición de la cola.	
IADIST	Compatibilidad con listas de distribución.	2
IAHGB	Si se debe endurecer el recuento de retroceso.	5
IAIGET	Si se permiten las operaciones get.	
IAIPUT	Si se permiten las operaciones de inserción.	
IAMLEN	La longitud máxima de mensajes.	
IAMDEP	Número máximo de mensajes permitidos en la cola.	
IAMDS	Si la prioridad del mensaje es relevante.	5
IAOIC	Número de llamadas MQOPEN que tienen la cola abierta para la entrada.	
IAOOC	Número de llamadas MQOPEN que tienen la cola abierta para salida.	
IAQDHE	Atributo de control para eventos de profundidad de cola alta.	4, 5
IAQDHL	Límite máximo para la profundidad de la cola.	4, 5
IAQDLE	Atributo de control para eventos de profundidad de cola baja.	4, 5
IAQDLL	Límite inferior para la profundidad de la cola.	4, 5
IAQDME	Atributo de control para eventos máximos de profundidad de cola.	4, 5
IAQSI	Límite para el intervalo de servicio de cola.	4, 5
IAQSIE	Atributo de control para eventos de intervalo de servicio de cola.	4, 5
IAQTYP	El tipo de cola.	
IAQSGD	Disposición del grupo de uso compartido de colas.	3
IARINT	Intervalo de retención de la cola.	5
IASCOP	Ámbito de definición de la cola.	4, 5

Tabla 785. Selektors de atributos MQINQ para colas (continuación)		
Selector	Descripción	Nota
IASHAR	Si la cola se puede compartir para la entrada.	
IATRG	Control de desencadenante.	
IATRGD	Profundidad del desencadenante.	5
IATRGP	Umbral de prioridad del mensaje para activaciones.	5
IATRGT	Tipo de desencadenante.	
IAUSAG	Uso.	
CLWLUSEQ	Utilice colas remotas.	

Nota:

1. Compatible con las siguientes plataformas:

- ▶ **AIX** AIX
- ▶ **IBM i** IBM i
- ▶ **Windows** Windows
- ▶ **z/OS** z/OS

y para IBM MQ MQI clientes conectados a estos sistemas.

2. Compatible con las siguientes plataformas:

- ▶ **AIX** AIX
- ▶ **IBM i** IBM i
- ▶ **Windows** Windows

y para IBM MQ los clientes conectados a estos sistemas.

3. ▶ **z/OS** Compatible con z/OS.

4. ▶ **z/OS** No compatible con z/OS.

5. No compatible con VSE/ESA.

Tabla 786. Selektors de atributos MQINQ para listas de nombres		
Selector	Descripción	Nota
CAALTD	Fecha de la última modificación (LNDATE)	1
CAALTT	Hora de la última modificación (LNTIME)	1
CALSTD	Descripción de la lista de nombres (LNNLD)	1
CALSTN	Nombre del objeto de la lista de nombres (LNNLN)	1
CANAMS	Nombres en la lista de nombres (LNQN x <i>Número de nombres en la lista</i>)	1
IANAMC	Número de nombres en la lista de nombres	1
IAQSGD	Disposición de grupo de uso compartido de colas	3

<i>Tabla 787. Selektors de atributos MQINQ para definiciones de procesos</i>		
Selector	Descripción	Nota
CAALTD	Fecha de la última modificación (LNDATE)	1
CAALTT	Hora de la última modificación (LNTIME)	1
CAAPPI	Identificador de la aplicación (LNPROA)	5
CAENV	Datos medioambientales (LNPROE)	5
CAPROD	Descripción de la definición del proceso (LNPROD)	5
CAPRON	Nombre de la definición del proceso (LNPRON)	5
CAUSR	Datos de usuario (LNPROU)	5
IAAPPT	Tipo de aplicación	5
IAQSGD	Disposición de grupo de uso compartido de colas	3

<i>Tabla 788. Selektors de atributos MQINQ para el gestor de colas</i>		
Selector	Descripción	Nota
CAALTD	Fecha de la última modificación (LNDATE)	1
CAALTT	Hora de la última modificación (LNTIME)	1
CACADX	Nombre de salida de definición automática de canal (LNEXN)	1
CACLWD	Datos transferidos a la salida de carga de trabajo del clúster (LNEXDA)	1
CACLWX	Nombre de la salida de carga de trabajo del clúster (LNEXN)	1
CACMDQ	Nombre de la cola de entrada de comandos del sistema (LNQN)	5
CADLQ	Nombre de la cola de mensajes no entregados (LNQN)	5
CADXQN	Nombre predeterminado de la cola de transmisión (LNQN)	5
CAQMD	Descripción del gestor de colas (LNQMD)	5
CAQMID	Identificador del gestor de colas (LNQMID)	1
CAQMN	Nombre del administrador de colas local (LNQMN)	5
CAQSGN	Nombre del grupo de uso compartido de colas (LNQSGN)	3
CARPN	Nombre del clúster para el que el gestor de colas proporciona servicios de repositorio (LNQMN)	1
CARPNL	Nombre del objeto de la lista de nombres que contiene los nombres de los clústeres para los que el gestor de colas proporciona servicios de repositorio (LNNLN)	1
CMDEV	Atributo de control que determina si los mensajes generados cuando se emiten comandos se colocan en una cola	8
IAAUTE	Atributo de control para eventos de autoridad	4, 5
IACAD	Atributo de control para la definición automática de canales	2
IACADE	Atributo de control para eventos de definición automática de canales	2
IACLXQ	Tipo de cola de transmisión de clúster predeterminado.	4
IACLWL	Longitud de la carga de trabajo de clúster	1

Tabla 788. Selektores de atributos MQINQ para el gestor de colas (continuación)		
Selector	Descripción	Nota
IACCSI	Identificador de juego de caracteres codificado	5
IACMDL	Nivel de comando compatible con el gestor de colas	5
IACFGE	Atributo de control para eventos de configuración	3
IADIST	Soporte de lista de distribución	2
IAINHE	Atributo de control para inhibir eventos	4, 5
IALCLE	Atributo de control para eventos locales	4, 5
IAMHND	Número máximo de asas	5
IAMLEN	Longitud máxima de mensaje	5
IAMPRI	Prioridad máxima	5
IAMUNC	Número máximo de mensajes no confirmados dentro de una unidad de trabajo	5
IAPFME	Atributo de control para eventos de rendimiento	4, 5
IAPLAT	Plataforma en la que reside el gestor de colas	5
IARMTE	Atributo de control para eventos remotos	4, 5
IASSE	Atributo de control para eventos de inicio y parada	4, 5
IASYNC	Disponibilidad del punto de sincronización	5
IATRLFT	Vida útil de los temas no administrativos no utilizados	
IATRGI	Activar intervalo	5

IACNT (entero con signo de 10 dígitos) - entrada

Recuento de atributos enteros.

Este es el número de elementos de la INTATR matriz. Cero es un valor válido.

Si este es al menos el número de selectores IA* en el **SELS** parámetro, se devuelven todos los atributos enteros solicitados.

INTATR (entero con signo de 10 dígitos x IACNT) - salida

Matriz de atributos enteros.

Se trata de una matriz de valores de IACNT atributos enteros.

Los valores de los atributos enteros se devuelven en el mismo orden que los selectores IA* en el **SELS** parámetro. Si la matriz contiene más elementos que el número de selectores IA*, los elementos sobrantes no se modifican.

Si HOBJ representa una cola, pero un selector de atributos no es aplicable a ese tipo de cola, se devuelve el valor específico IAVNA para el elemento correspondiente en la INTATR matriz.

CALEN (entero con signo de 10 dígitos) - entrada

Longitud del búfer de atributos de caracteres.

Esta es la longitud en bytes del **CHRATR** parámetro.

Esto debe ser al menos la suma de las longitudes de los atributos de caracteres solicitados (véase SELS). Cero es un valor válido.

CHRATR (cadena de caracteres de 1 byte x CALEN) - salida

Atributos de los personajes.

Este es el búfer en el que se devuelven los atributos de los caracteres, concatenados entre sí. La longitud del búfer viene dada por el **CALEN** parámetro.

Los atributos de los caracteres se devuelven en el mismo orden que los selectores CA* en el **SELS** parámetro. La longitud de cada cadena de atributos es fija para cada atributo (véase SELS), y el valor que contiene se rellena a la derecha con espacios en blanco si es necesario. Si el búfer es mayor que el necesario para contener todos los atributos de caracteres solicitados (incluido el relleno), los bytes más allá del último valor de atributo devuelto no se modifican.

Si HOBJ representa una cola, pero un selector de atributos no es aplicable a ese tipo de cola, se devuelve una cadena de caracteres compuesta íntegramente por asteriscos (*) como valor de ese atributo en CHRATR.

CMPCOD (entero con signo de 10 dígitos) - salida

Código de terminación.

Es uno de los siguientes:

CCOK

Realización satisfactoria.

CCWARN

Aviso (finalización parcial).

CCFAIL

La llamada no se ha realizado satisfactoriamente.

RAZÓN (entero con signo de 10 dígitos) - salida

Código de motivo calificativo CMPCOD.

Si CMPCOD es CCOK:

RCNONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si CMPCOD es CCWARN:

RC2008

(2008, X'7D8') No hay suficiente espacio para los atributos de los personajes.

RC2022

(2022, X'7E6') No hay suficiente espacio para los atributos enteros.

RC2068

(2068, X'814') Selector no aplicable al tipo de cola.

Si CMPCOD es CCFAIL:

RC2219

(2219, X'8AB') Se ha vuelto a introducir la llamada « MQI » antes de que finalizara la llamada anterior.

RC2006

(2006, X'7D6') Longitud de los atributos de los caracteres no válida.

RC2007

(2007, X'7D7') Cadena de atributos de carácter no válida.

RC2009

(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

RC2018

(2018, X'7E2') El manejador de conexión no es válido.

RC2019

(2019, X'7E3') El manejador de objeto no es válido.

RC2021

(2021, X'7E5') Recuento de atributos enteros no válido.

RC2023

(2023, X'7E7') Matriz de atributos enteros no válida.

RC2038

(2038, X'7F6') La cola no está abierta para consultas.

RC2041

(2041, X'7F9') La definición de objeto ha cambiado desde que se ha abierto.

RC2101

(2101, X'835') El objeto se ha dañado.

RC2052

(2052, X'804') La cola se ha suprimido.

RC2058

(2058, X'80A') El nombre del gestor de colas no es válido o no se conoce.

RC2059

(2059, X'80B') El gestor de colas no está disponible para efectuar la conexión.

RC2162

(2162, X'872') El gestor de colas está concluyendo.

RC2102

(2102, X'836') No hay disponibles suficientes recursos del sistema.

RC2065

(2065, X'811') Recuento de selectores no válido.

RC2067

(2067, X'813') Selector de atributos no válido.

RC2066

(2066, X'812') Número de selectores demasiado grande.

RC2071

(2071, X'817') No hay suficiente almacenamiento disponible.

RC2195

(2195, X'893') Se ha producido un error inesperado.

Declaración RPG

```

C*..1.....2.....3.....4.....5.....6.....7..
C          CALLP      MQINQ(HCONN : HOBJ : SELCNT :
C                      SELS(1) : IACNT : INTATR(1) :
C                      CALEN : CHRATR : CMPCOD :
C                      REASON)

```

La definición del prototipo para la llamada es:

```

D*..1.....2.....3.....4.....5.....6.....7..
DMQINQ      PR          EXTPROC('MQINQ')
D* Connection handle
D HCONN          10I 0 VALUE
D* Object handle
D HOBJ          10I 0 VALUE
D* Count of selectors
D SELCNT        10I 0 VALUE
D* Array of attribute selectors
D SELS          10I 0
D* Count of integer attributes
D IACNT          10I 0 VALUE
D* Array of integer attributes
D INTATR        10I 0
D* Length of character attributes buffer
D CALEN          10I 0 VALUE
D* Character attributes
D CHRATR          *   VALUE
D* Completion code
D CMPCOD        10I 0

```

IBM i MQINQMP (propiedad Consultar mensaje) en IBM i

La llamada MQINQMP devuelve el valor de una propiedad de un mensaje.

- [“Sintaxis” en la página 1344](#)
- [“Parámetros” en la página 1344](#)
- [“Declaración RPG” en la página 1348](#)

Sintaxis

MQINQMP (*Hconn*, *Hmsg*, *InqPropOpts*, *Name*, *PropDesc*, *Type*, *ValueLength*, *Value*, *DataLength*, *CompCode*, *Reason*)

Parámetros

La llamada MQINQMP tiene los parámetros siguientes:

HCONN (entero con signo de 10 dígitos)-entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* debe coincidir con el descriptor de conexión que se ha utilizado para crear el descriptor de mensaje especificado en el parámetro **Hmsg**.

Si el descriptor de contexto de mensaje se ha creado utilizando HCUNAS, se debe establecer una conexión válida en la hebra consultando una propiedad del descriptor de contexto de mensaje; de lo contrario, la llamada falla con RC2009.

HMSG (entero con signo de 20 dígitos)-entrada

Este es el manejador de mensajes que se debe consultar. El valor ha sido devuelto por una llamada **MQCRTMH** anterior.

INQOPT (MQIMPO)-entrada

Consulte el tipo de datos [MQIMPO](#) para obtener más detalles.

PRNAME (MQCHARV)-entrada

Describe el nombre de la propiedad a consultar.

Si no se encuentra ninguna propiedad con este nombre, la llamada falla con la razón RC2471.

Puede utilizar el carácter de signo de porcentaje (%) al final del nombre de propiedad. El comodín coincide con cero o más caracteres, incluido el carácter de punto (.). Esto permite a una aplicación consultar el valor de muchas propiedades. Llame a MQINQMP con la opción IPINQF para obtener la primera propiedad coincidente y de nuevo con la opción IPINQN para obtener la siguiente propiedad coincidente. Cuando no hay más propiedades coincidentes disponibles, la llamada falla con RC2471. Si el campo 'ReturnedName' de la estructura InqPropOpts se inicializa con una dirección o un offset para el nombre devuelto de la propiedad, éste se completa a la vuelta de MQINQMP con el nombre de la propiedad que ha coincidido. Si el campo 'VSBuFSize' de 'ReturnedName' en la estructura InqPropOpts es menor que la longitud del nombre de propiedad devuelto, el código de finalización es CCFAIL con la razón RC2465.

Las propiedades que tienen sinónimos conocidos se devuelven de la siguiente manera:

1. Propiedades con el prefijo "mqps." se devuelven con el nombre de propiedad IBM MQ. Por ejemplo, "MQTopicString" es el nombre devuelto en lugar de mqps.Top".
2. Propiedades con el prefijo "jms." o "mcd." se devuelven como nombre de campo de cabecera JMS. Por ejemplo, "JMSExpiration" es el nombre devuelto en lugar de "jms.Exp".

3. Propiedades con el prefijo "usr." se devuelven sin ese prefijo. Por ejemplo, se devuelve "Color" en lugar de "usr.Color".

Las propiedades con sinónimos sólo se devuelven una vez.

En el lenguaje de programación RPG, las variables de macro siguientes se definen para consultar todas las propiedades y todas las propiedades que empiezan por "usr.":

INQALL

Consultar todas las propiedades del mensaje.

INQUSR

Consultar todas las propiedades del mensaje que inician "usr.". El nombre devuelto se devuelve sin el "usr." .

Si se especifica IPINQN pero el nombre ha cambiado desde la llamada anterior o ésta es la primera llamada, entonces está implícito IPINQF.

Consulte [Nombres de propiedad](#) y [Restricciones de nombre de propiedad](#) para obtener más información sobre el uso de nombres de propiedad.

PRPDSC (MQPD)-salida

Esta estructura se utiliza para definir los atributos de una propiedad, incluido lo que sucede si la propiedad no está soportada, a qué contexto de mensaje pertenece la propiedad y en qué mensajes debe copiarse la propiedad. Consulte [MQPD](#) para obtener detalles de esta estructura.

TYPE (entero con signo de 10 dígitos)-entrada/salida

Al volver de la llamada MQINQMP, este parámetro se establece en el tipo de datos *Valor*. El tipo de datos puede ser cualquiera de los siguientes:

TIPBOL

Un booleano.

TYPBST

una serie de bytes.

TYPI8

Un entero con signo de 8 bits.

TYPI16

Un entero con signo de 16 bits.

TYPI32

Un entero con signo de 32 dígitos.

TYPI64

Un entero con signo de 64 bits.

TYPF32

Un número de coma flotante de 32 bits.

TYPF64

Un número de coma flotante de 64 bits.

TIPSTR

Una serie de caracteres.

TYPNUL

La propiedad existe pero tiene un valor nulo.

Si el tipo de datos del valor de propiedad no se reconoce, se devuelve TYPSTR y se coloca una representación de serie del valor en el área *Valor* . Se puede encontrar una representación de serie del tipo de datos en el campo *IPITYP* del parámetro *IPOPT* . Se devuelve un código de finalización de aviso con la razón RC2467.

Además, si se especifica la opción IPCTYP, se solicita la conversión del valor de propiedad. Utilice *Tipo* como entrada para especificar el tipo de datos con el que desea que se devuelva la propiedad.

Consulte la descripción de la opción IPCTYP de “MQIMPO (Opciones de propiedad del mensaje de consulta) activado IBM i” en la página 1124 para obtener detalles de la conversión de tipo de datos.

Si no solicita la conversión de tipo, puede utilizar el valor siguiente en la entrada:

TYPASTO

El valor de la propiedad se devuelve sin convertir su tipo de datos.

VALLEN (entero con signo de 10 dígitos)-entrada

Longitud en bytes del área Valor.

Especifique cero para las propiedades para las que no necesita el valor devuelto. Estas podrían ser propiedades diseñadas por una aplicación para tener un valor nulo o una serie vacía. Especifique también cero si se ha especificado la opción IPQLEN; en este caso, no se devuelve ningún valor.

VALUE (bit de 1 byte stringxVALLEN)-salida

Este es el área que contiene el valor de propiedad consultado. El almacenamiento intermedio debe estar alineado en un límite adecuado para el valor que se devuelve. Si no lo hace, es posible que se produzca un error cuando se acceda más tarde al valor.

Si *VALLEN* es menor que la longitud del valor de propiedad, la mayor parte posible del valor de propiedad se mueve a *VALUE* y la llamada falla con el código de terminación CCFAIL y la razón RC2469.

El juego de caracteres de los datos en *VALUE* se proporciona mediante el campo IPRETCSI en el parámetro INQOPT. La codificación de los datos en *VALUE* la proporciona el campo IPRETENC en el parámetro INQOPT.

Si el parámetro *VALLEN* es cero, no se hace referencia a *VALUE*.

DATLEN (entero con signo de 10 dígitos)-salida

Es la longitud en bytes del valor de propiedad real tal como se devuelve en el área *Valor*.

Si *DataLength* es menor que la longitud del valor de la propiedad, *DataLength* se sigue entrando al devolver de la llamada MQINQMP. Esto permite a la aplicación determinar el tamaño del almacenamiento intermedio necesario para acomodar el valor de propiedad y, a continuación, volver a emitir la llamada con un almacenamiento intermedio del tamaño adecuado.

También se pueden devolver los valores siguientes.

Si el parámetro *Tipo* se establece en TYPSTR o TYPBST:

VLEMP

La propiedad existe pero no contiene caracteres ni bytes.

CMPCOD (entero con signo de 10 dígitos)-salida

El código de terminación; es uno de los siguientes:

CCOK

Realización satisfactoria.

CCWARN

Aviso (finalización parcial).

CCFAIL

La llamada no se ha realizado satisfactoriamente.

REASON (entero con signo de 10 dígitos)-salida

El código de razón que califica a *CompCode*.

Si *CMPCOD* es CCOK:

RCNONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es CCWARN:

RC2492

(2492, X'09BC') Nombre de propiedad devuelto no convertido.

RC2466

(2466, X'09A2') Valor de propiedad no convertido.

RC2467

(2467, X'09A3') No se admite el tipo de datos de la propiedad.

RC2421

(2421, X'0975 ') No se ha podido analizar una carpeta MQRFH2 que contiene propiedades.

Si *CMPCOD* es CCFAIL:

RC2204

(2204, X'089C') Adaptador no disponible.

RC2130

(2130, X'0852 ') No se puede cargar el módulo de servicio del adaptador.

RC2157

(2157, X'086D') Los ASID primario y de origen difieren.

RC2004

(2004, X'07D4') Parámetro de valor no válido.

RC2005

(2005, X'07D5') Parámetro de longitud de valor no válido.

RC2219

(2219, X'08AB') Se ha iniciado una llamada al MQI antes de que finalizara la llamada anterior.

RC2009

(2009, X'07D9') Se ha perdido la conexión con el gestor de colas.

RC2010

(2010, X'07DA') El parámetro de longitud de datos no es válido.

RC2464

(2464, X'09A0') La estructura de opciones de la propiedad del mensaje de consulta no es válida.

RC2460

(2460, X'099C') Manejador de mensaje no válido.

RC2499

(2499, X'09C3') Manejador de mensaje ya en uso.

RC2064

(2046, X'07F8') Opciones no válidas o no coherentes.

RC2482

(2482, X'09B2') La estructura del descriptor de propiedad no es válida.

RC2470

(2470, X'09A6') No se admite la conversión del tipo de datos real al solicitado.

RC2442

(2442, X'098A') Nombre de propiedad no válido.

RC2465

(2465, X'09A1') El nombre de la propiedad es demasiado grande para el búfer de nombres devuelto.

RC2471

(2471, X'09A7) Propiedad no disponible.

RC2469

(2469, X'09A5') Valor de propiedad demasiado grande para el área Valor.

RC2472

(2472, X'09A8') Se ha encontrado un error de formato numérico en los datos de valor.

RC2473

(2473, X'09A9') Tipo de propiedad solicitado no válido.

RC2111

(2111, X'083F') El identificador del juego de caracteres codificado del nombre de la propiedad no es válido.

RC2071

(2071, X'0871 ') Almacenamiento insuficiente disponible.

RC2195

(2195, X'0893 ') Se ha producido un error inesperado.

Para obtener información detallada sobre estos códigos, consulte:

- [IBM MQ for z/OS Mensajes, finalización y códigos de motivo para IBM MQ for z/OS](#)
- [Mensajes y códigos de motivo para todas las demás plataformas IBM MQ](#)

Declaración RPG

```
C*..1.....2.....3.....4.....5.....6.....7..
C                                CALLP      MQINQMP(HCONN : HMSG : INQOPT :
                                           PRNAME : PRPDSC : TYPE :
                                           VALLEN : VALUE : DATLEN :
                                           CMPCOD : REASON)
```

La definición de prototipo para la llamada es:

```
DMQINQMP          PR          EXTPROC('MQINQMP')
D* Connection handle
D HCONN              10I 0 VALUE
D* Message handle
D HMSG              20I 0 VALUE
D* Options that control the action of MQINQMP
D INQOPT              72A
D* Property name
D PRNAME              32A
D* Property descriptor
D PRPDSC              24A
D* Property data type
D TYPE              10I 0
D* Length in bytes of the Value area
D VALLEN              10I 0 VALUE
D* Property value
D VALUE              * VALUE
D* Length of the property value
D DATLEN              10I 0
D* Completion code
D CMPCOD              10I 0
D* Reason code qualifying CompCode
D REASON              10I 0
```

MQMHBUFF (Convertir descriptor de mensaje en almacenamiento intermedio) en IBM i

MQMHBUFF convierte un manejador de mensajes en un almacenamiento intermedio y es el inverso de la llamada MQBUFMH.

- [“Sintaxis” en la página 1348](#)
- [“Notas de uso” en la página 1349](#)
- [“Parámetros” en la página 1349](#)
- [“Declaración RPG” en la página 1351](#)

Sintaxis

MQMHBUFF (*Hconn*, *Hmsg*, *MsgHBufOpts*, *Name*, *MsgDesc*, *BufferLength*, *Buffer*, *DataLength*, *CompCode*, *Reason*)

Notas de uso

MQMHBUF convierte un descriptor de mensaje en un almacenamiento intermedio.

Puede utilizarlo con una salida de API MQGET para acceder a determinadas propiedades, utilizando las API de propiedades de mensajes y, a continuación, volver a pasar estas propiedades en un almacenamiento intermedio a una aplicación diseñada para utilizar cabeceras MQRFH2 en lugar de manejadores de mensajes.

Esta llamada es la inversa de la llamada MQBUFMH, que puede utilizar para analizar propiedades de mensaje de un almacenamiento intermedio en un descriptor de mensaje.

Parámetros

La llamada MQMHBUF tiene los parámetros siguientes:

HCONN (entero con signo de 10 dígitos)-entrada

Este manejador representa la conexión con el gestor de colas.

El valor de *HCONN* debe coincidir con el descriptor de conexión que se ha utilizado para crear el descriptor de mensaje especificado en el parámetro **HMSG**.

Si el descriptor de mensaje se ha creado utilizando HCUNAS, se debe establecer una conexión válida en la hebra suprimiendo el descriptor de mensaje. Si no se establece una conexión válida, la llamada falla con RC2009.

HMSG (entero con signo de 20 dígitos)-entrada

Este descriptor de contexto es el descriptor de contexto de mensaje para el que se necesita un almacenamiento intermedio.

El valor ha sido devuelto por una llamada MQCRTMH anterior.

MHBOPT (MQMHBO)-entrada

La estructura MQMHBO permite a las aplicaciones especificar opciones que controlan cómo se producen los almacenamientos intermedios a partir de los manejadores de mensajes.

Consulte [“MQBMHO \(Opciones de búfer para el controlador de mensajes\) activado IBM i”](#) en la página 1035 para obtener los detalles.

PRNAME (MQCHARV)-entrada

El nombre de la propiedad o propiedades que se van a poner en el almacenamiento intermedio.

Si no se encuentra ninguna propiedad que coincida con el nombre, la llamada falla con RC2471.

Comodines

Puede utilizar un comodín para colocar más de una propiedad en el almacenamiento intermedio. Para ello, utilice el signo de porcentaje (%) al final del nombre de propiedad. Este comodín coincide con cero o más caracteres, incluido el carácter de punto (.).

Consulte [Nombres de propiedad y Restricciones de nombre de propiedad](#) para obtener más información sobre el uso de nombres de propiedad.

MSGDSC (MQMD)-entrada/salida

La estructura *MSGDSC* describe el contenido del área de almacenamiento intermedio.

En la salida, los campos *Encoding*, *CodedCharSetId* y *Format* se establecen para describir correctamente la codificación, el identificador de juego de caracteres y el formato de los datos en el área de almacenamiento intermedio tal como los graba la llamada.

Los datos de esta estructura están en el juego de caracteres y la codificación de la aplicación.

BUFLEN (entero con signo de 10 dígitos)-entrada

BUFLEN es la longitud del área de almacenamiento intermedio, en bytes.

BUFFER (serie de bits de 1 byte x BUFLen)-entrada/salida

BUFFER define el área que contiene el almacenamiento intermedio de mensajes. Para la mayoría de los datos, debe alinear el almacenamiento intermedio en un límite de 4 bytes.

Si *BUFFER* contiene datos numéricos o de caracteres, establezca los campos *CodedCharSetId* y *Encoding* del parámetro **MSGDSC** en los valores adecuados para los datos; esto permite convertir los datos, si es necesario.

Si se encuentran propiedades en el almacenamiento intermedio de mensajes, se eliminan de forma opcional; posteriormente pasan a estar disponibles desde el descriptor de contexto de mensaje al devolver la llamada.

En el lenguaje de programación C, el parámetro se declara como un puntero a void, lo que significa que la dirección de cualquier tipo de datos se puede especificar como parámetro.

Si el parámetro **BUFLen** es cero, no se hace referencia a *BUFFER*. En este caso, la dirección de parámetro pasada por los programas escritos en C o System/390 assembler puede ser nula.

DATLen (entero con signo de 10 dígitos)-salida

DATLen es la longitud, en bytes, de las propiedades devueltas en el almacenamiento intermedio. Si el valor es cero, ninguna propiedad coincide con el valor proporcionado en *PRNAME* y la llamada falla con el código de razón RC2471.

Si *BUFLen* es menor que la longitud necesaria para almacenar las propiedades en el almacenamiento intermedio, la llamada MQMHBUFF falla con RC2469, pero se sigue entrando un valor en *DATLen*. Esto permite a la aplicación determinar el tamaño del almacenamiento intermedio necesario para acomodar las propiedades y, a continuación, volver a emitir la llamada con el *BUFLen* necesario.

CMPCOD (entero con signo de 10 dígitos)-salida

El código de terminación; es uno de los siguientes:

CCOK

Realización satisfactoria.

CCFAIL

La llamada no se ha realizado satisfactoriamente.

REASON (entero con signo de 10 dígitos)-salida

El código de razón que califica *CMPCOD*.

Si *CMPCOD* es CCOK:

RCNONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CMPCOD* es CCFail:

RC2204

(2204, X'089C') Adaptador no disponible.

RC2130

(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

RC2157

(2157, X'86D') Los ASID primario e inicial varían.

RC2501

(2501, X'095C') El manejador del mensaje a la estructura de opciones del buffer no es válido.

RC2004

(2004, X'07D4') El parámetro del búfer no es válido.

RC2005

(2005, X'07D5') El parámetro de longitud del búfer no es válido.

RC2219

(2219, X'08AB') Se ha iniciado una llamada al MQI antes de que finalizara la llamada anterior.

RC2009

(2009, X'07D9') Se ha perdido la conexión con el gestor de colas.

RC2010

(2010, X'07DA') El parámetro de longitud de datos no es válido.

RC2460

(2460, X'099C') Manejador de mensaje no válido.

RC2026

(2026, X'07EA') Descriptor de mensaje no válido.

RC2499

(2499, X'09C3') Manejador de mensaje ya en uso.

RC2046

(2046, X'07FE') Opciones no válidas o no coherentes.

RC2442

(2442, X'098A') El nombre de la propiedad no es válido.

RC2471

(2471, X'09A7') Propiedad no disponible.

RC2469

(2469, X'09A5') El valor de BufferLength es demasiado pequeño para contener las propiedades especificadas.

RC2195

(2195, X'893') Se ha producido un error inesperado.

Declaración RPG

```

C*..1.....2.....3.....4.....5.....6.....7..
C                                CALLP      MQMHBUFF(HCONN : HMSG : MHBOPT :
                                           PRNAME : MSGDSC : BUFLen :
                                           BUFFER : DATLEN :
                                           CMPCOD : REASON)

```

La definición de prototipo para la llamada es:

```

DMQMHBUFF      PR      EXTPROC('MQMHBUFF')
D* Connection handle
D HCONN              10I 0 VALUE
D* Message handle
D HMSG              20I 0 VALUE
D* Options that control the action of MQMHBUFF
D MHBOPT            12A
D* Property name
D PRNAME            32A
D* Message descriptor
D MSGDSC            364A
D* Length in bytes of the Buffer area
D BUFLen            10I 0 VALUE
D* Area to contain the properties
D BUFFER            *    VALUE
D* Length of the properties
D DATLEN            10I 0
D* Completion code
D CMPCOD            10I 0
D* Reason code qualifying CompCode
D REASON            10I 0

```

MQOPEN (Abrir objeto) en IBM i

La llamada MQOPEN establece el acceso a un objeto.

Son válidos los siguientes tipos de objeto:

- Cola (incluidas las listas de distribución)
- Lista de nombres

- Definición de proceso
- Gestor de colas
- Tema

Índice

- [“Sintaxis” en la página 1352](#)
- [“Notas de uso” en la página 1352](#)
- [“Parámetros” en la página 1356](#)
- [“Declaración RPG” en la página 1363](#)

Sintaxis

`MQOPEN (HCONN, OBJDSC, OPTS, HOBJ, CMPCOD, REASON)`

Notas de uso

1. El objeto abierto es uno de los siguientes:

- Una cola, para:
 - Obtener o examinar mensajes (utilizando la llamada MQGET)
 - Transferir mensajes (utilizando la llamada MQPUT)
 - Consultar los atributos de la cola (utilizando la llamada MQINQ)
 - Establecer los atributos de la cola (utilizando la llamada MQSET)

Si la cola especificada es una cola modelo, se crea una cola local dinámica.

Una lista de distribución es un tipo especial de objeto de cola que contiene una lista de colas. Se puede abrir para transferir mensajes, pero no para obtener o examinar mensajes, ni para consultar o establecer atributos. Consulte la nota de uso 8 para obtener más detalles.

Una cola que tiene QSGDISP (GROUP) es un tipo especial de definición de cola que no se puede utilizar con las llamadas MQOPEN o MQPUT1 .

- Una lista de nombres, para:
 - Consultar los nombres de las colas de la lista (utilizando la llamada MQINQ).
 - Una definición de proceso, para:
 - Consultar sobre los atributos de proceso (utilizando la llamada MQINQ).
 - El gestor de colas, para:
 - Consultar los atributos del gestor de colas local (utilizando la llamada MQINQ).
2. Es válido que una aplicación abra el mismo objeto más de una vez. Se devuelve un descriptor de objeto diferente para cada apertura. Cada descriptor de contexto que se devuelve se puede utilizar para las funciones para las que se ha realizado la apertura correspondiente.
3. Si el objeto que se está abriendo es una cola pero no una cola de clúster, toda la resolución de nombres dentro del gestor de colas local tiene lugar en el momento de la llamada MQOPEN. Esto puede incluir uno o más de los siguientes para una llamada MQOPEN determinada:
- Resolución de alias para el nombre de una cola base
 - Resolución del nombre de una definición local de una cola remota al nombre del gestor de colas remoto y el nombre por el que se conoce la cola en el gestor de colas remoto
 - Resolución del nombre del gestor de colas remoto en el nombre de una cola de transmisión local
- Sin embargo, tenga en cuenta que las llamadas MQINQ o MQSET posteriores para el descriptor de contexto se relacionan únicamente con el nombre que se ha abierto y no con el objeto resultante después de que se haya producido la resolución de nombres. Por ejemplo, si el objeto abierto es un

alias, los atributos devueltos por la llamada MQINQ son los atributos del alias, no los atributos de la cola base en la que se resuelve el alias. Sin embargo, la comprobación de resolución de nombres se sigue realizando, independientemente de lo que se especifique para el parámetro **OPTS** en el MQOPEN correspondiente.

Si el objeto que se está abriendo es una cola de clúster, la resolución de nombres se puede producir en el momento de la llamada MQOPEN, o se puede aplazar hasta más adelante. El punto en el que se produce la resolución está controlado por las opciones OOBND* especificadas en la llamada MQOPEN:

- OOBND0
- OOBNDN
- OOBNDQ

Consulte [Resolución de nombres](#) para obtener más información sobre la resolución de nombres para colas de clúster.

4. Los atributos de un objeto pueden cambiar mientras una aplicación tiene el objeto abierto. En muchos casos, la aplicación no se da cuenta de ello, pero para determinados atributos el gestor de colas marca el descriptor de contexto como ya no válido. Son las siguientes:

- Cualquier atributo que afecte a la resolución de nombres del objeto. Esto se aplica independientemente de las opciones de apertura utilizadas e incluye lo siguiente:
 - Un cambio en el atributo **BaseQName** de una cola alias que está abierta.
 - Un cambio en los atributos de cola **RemoteQName** o **RemoteQMGrName** , para cualquier descriptor de contexto que esté abierto para esta cola, o para una cola que se resuelva a través de esta definición como un alias de gestor de colas.
 - Cualquier cambio que haga que un descriptor de contexto abierto actualmente para una cola remota se resuelva en una cola de *transmisión* diferente, o que no se pueda resolver en una en absoluto. Por ejemplo, esto puede incluir:
 - Un cambio en el atributo **XmitQName** de la definición local de una cola remota, si la definición se está utilizando para una cola o para un alias de gestor de colas.

Hay una excepción a esto, a saber, la creación de una nueva cola de transmisión. Un descriptor de contexto que se habría resuelto en esta cola si estuviera presente cuando se abrió el descriptor de contexto, pero que en su lugar se hubiera resuelto en la cola de transmisión predeterminada, no se ha convertido en no válido.

- Un cambio en el atributo del gestor de colas **DefXmitQName** . En este caso, todos los descriptores de contexto abiertos que se han resuelto en la cola especificada anteriormente (que se ha resuelto sólo porque era la cola de transmisión predeterminada) se marcan como no válidos. Los manejadores que se han resuelto en esta cola por otras razones no se ven afectados.
- El atributo de cola **Shareability** , si hay dos o más descriptores de contexto que proporcionan actualmente acceso OOINPS para esta cola, o para una cola que se resuelve en esta cola. Si es así, *todos* los descriptores de contexto que están abiertos para esta cola, o para una cola que se resuelve en esta cola, se marcan como no válidos, independientemente de las opciones de apertura.
- El atributo de cola **Usage** , para todos los descriptores de contexto que están abiertos para esta cola, o para una cola que se resuelve en esta cola, independientemente de las opciones de apertura.

Cuando un descriptor de contexto se marca como no válido, todas las llamadas posteriores (distintas de MQCLOSE) que utilizan este descriptor de contexto fallan con el código de razón RC2041; la aplicación debe emitir una llamada MQCLOSE (utilizando el descriptor de contexto original) y, a continuación, volver a abrir la cola. Las actualizaciones no confirmadas en el descriptor de contexto antiguo de las llamadas satisfactorias anteriores se pueden seguir confirmando o restituyendo, según lo requiera la lógica de la aplicación.

Si el cambio de un atributo hace que esto suceda, se debe utilizar una versión especial "force" del mandato.

5. El gestor de colas realiza comprobaciones de seguridad cuando se emite una llamada MQOPEN, para verificar que el identificador de usuario bajo el que se ejecuta la aplicación tiene el nivel de autorización adecuado antes de que se permita el acceso. La comprobación de autorización se realiza en el nombre del objeto que se está abriendo, y no en el nombre o nombres, lo que resulta después de que se haya resuelto un nombre.

Si el objeto que se está abriendo es una cola modelo, el gestor de colas realiza una comprobación de seguridad completa con el nombre de la cola modelo y el nombre de la cola dinámica que se crea. Si la cola dinámica resultante se abre entonces de forma explícita, se realiza una comprobación de seguridad de recursos adicional con respecto al nombre de la cola dinámica.

6. Una cola remota se puede especificar de una de dos maneras en el parámetro **OBJDSC** de esta llamada (consulte los campos *ODON* y *ODMN* descritos en [“MQOD \(Descriptor de objeto\) activado IBM i”](#) en la página 1183):

- Especificando para *ODON* el nombre de una definición local de la cola remota. En este caso, *ODMN* hace referencia al gestor de colas local y se puede especificar como espacios en blanco.

La validación de seguridad realizada por el gestor de colas local verifica que el usuario tiene autorización para abrir la definición local de la cola remota.

- Especificando para *ODON* el nombre de la cola remota tal como la conoce el gestor de colas remoto. En este caso, *ODMN* es el nombre del gestor de colas remoto.

La validación de seguridad realizada por el gestor de colas local verifica que el usuario está autorizado a enviar mensajes a la cola de transmisión como resultado del proceso de resolución de nombres.

En cualquier caso:

- El gestor de colas local no envía mensajes al gestor de colas remoto para comprobar que el usuario está autorizado a colocar mensajes en la cola.
- Cuando llega un mensaje al gestor de colas remoto, el gestor de colas remoto puede rechazarlo porque el usuario que ha originado el mensaje no está autorizado.

7. Una llamada MQOPEN con la opción OOBROW establece un cursor de examen, para su uso con llamadas MQGET que especifican el descriptor de objeto y una de las opciones de examen. Esto permite explorar la cola sin alterar su contenido. Un mensaje que se ha encontrado al examinar se puede eliminar posteriormente de la cola utilizando la opción GMMUC.

Varios cursores de examen pueden estar activos para una sola aplicación emitiendo varias solicitudes MQOPEN para la misma cola.

8. Las siguientes notas se aplican al uso de listas de distribución.

- Los campos de la estructura MQOD deben establecerse de la forma siguiente al abrir una lista de distribución:
 - *ODVER* debe ser ODVER2 o superior.
 - *ODOT* debe ser OTQ.
 - *ODON* debe estar en blanco o la serie nula.
 - *ODMN* debe estar en blanco o la serie nula.
 - *ODREC* Tiene que ser mayor que cero.
 - Uno de *ODORO* y *ODORP* debe ser cero y el otro distinto de cero.
 - No más de uno de *ODRRO* y *ODRRP* puede ser distinto de cero.
 - Debe haber *ODREC* registros de objeto, direccionado por *ODORO* o *ODORP*. Los registros de objeto deben establecerse en los nombres de las colas de destino que se van a abrir.

- Si uno de *ODRR0* y *ODRRP* es distinto de cero, debe haber *ODREC* registros de respuesta presentes. Los establece el gestor de colas si la llamada se completa con el código de razón RC2136.

Un MQOD version-2 también se puede utilizar para abrir una única cola que no esté en una lista de distribución, asegurándose de que *ODREC* sea cero.

- Sólo las siguientes opciones de apertura son válidas en el parámetro **OPTS** :

- OOOOUT
- OOPA*
- OOSSET*
- OOALTU
- OOFIQ

- Las colas de destino de la lista de distribución pueden ser colas locales, alias o remotas, pero no pueden ser colas modelo. Si se especifica una cola modelo, dicha cola no se puede abrir, con el código de razón RC2057. Sin embargo, esto no impide que otras colas de la lista se abran correctamente.

- Los parámetros de código de terminación y código de razón se establecen de la forma siguiente:

- Si las operaciones abiertas para las colas de la lista de distribución son satisfactorias o fallan de la misma forma, los parámetros de código de terminación y código de razón se establecen para describir el resultado común. Los registros de respuesta MQRR (si los proporciona la aplicación) no se establecen en este caso.

Por ejemplo, si cada apertura se realiza correctamente, el código de terminación se establece en CCOK y el código de razón es RCNONE; si cada apertura falla porque no existe ninguna de las colas, los parámetros se establecen en CCFAIL y RC2085.

- Si las operaciones abiertas para las colas de la lista de distribución no son todas satisfactorias o fallan de la misma forma:

- El parámetro de código de terminación se establece en CCWARN si al menos una apertura se ha realizado correctamente, y en CCFAIL si todo ha fallado.
- El parámetro de código de razón se establece en RC2136.
- Los registros de respuesta (si los proporciona la aplicación) se establecen en los códigos de terminación individuales y los códigos de razón para las colas de la lista de distribución.

- Cuando una lista de distribución se ha abierto correctamente, el descriptor de contexto *HOB*J devuelto por la llamada se puede utilizar en llamadas MQPUT posteriores para colocar mensajes en las colas de la lista de distribución y en una llamada MQCLOSE para renunciar al acceso a la lista de distribución. La única opción de cierre válida para una lista de distribución es CONONE.

La llamada MQPUT1 también se puede utilizar para colocar un mensaje en una lista de distribución; la estructura MQOD que define las colas de la lista se especifica como un parámetro en dicha llamada.

- Cada destino abierto correctamente en la lista de distribución cuenta como un descriptor de contexto *separado* al comprobar si la aplicación ha superado el número máximo permitido de descriptors de contexto (consulte el atributo de gestor de colas **MaxHandles**). Esto es cierto incluso cuando dos o más de los destinos de la lista de distribución se resuelven realmente en la misma cola física. Si la llamada MQOPEN o MQPUT1 para una lista de distribución haría que el número de descriptors de contexto utilizados por la aplicación excediera de *MaxHandles*, la llamada falla con el código de razón RC2017.
- Cada destino que se abre correctamente tiene el valor de su atributo **OpenOutputCount** incrementado en uno. Si dos o más de los destinos de la lista de distribución se resuelven realmente en la misma cola física, el atributo **OpenOutputCount** de dicha cola se incrementa en el número de destinos de la lista de distribución que se resuelven en dicha cola.
- Cualquier cambio en las definiciones de cola que hubiera hecho que un descriptor de contexto no fuera válido si las colas se hubieran abierto individualmente (por ejemplo, un cambio en la vía

de acceso de resolución), no hace que el descriptor de contexto de lista de distribución no sea válido. Sin embargo, da como resultado una anomalía para esa cola concreta cuando se utiliza el descriptor de contexto de lista de distribución en una llamada MQPUT posterior.

- Es válido que una lista de distribución contenga sólo un destino.

9. Las notas siguientes se aplican al uso de colas de clúster.

- Cuando una cola de clúster se abre por primera vez, y el gestor de colas local no es un gestor de colas de depósito completo, el gestor de colas local obtiene información sobre la cola de clúster de un gestor de colas de depósito completo. Cuando la red está ocupada, el gestor de colas local puede tardar varios segundos en recibir la información necesaria del gestor de colas de repositorio. Como resultado, es posible que la aplicación que emite la llamada MQOPEN tenga que esperar hasta 10 segundos antes de que se devuelva el control de la llamada MQOPEN. Si el gestor de colas local no recibe la información necesaria sobre la cola de clúster dentro de este tiempo, la llamada falla con el código de razón RC2189.
- Cuando se abre una cola de clúster y hay varias instancias de la cola en el clúster, la instancia realmente abierta depende de las opciones especificadas en la llamada MQOPEN:

- Si las opciones especificadas incluyen alguna de las siguientes:

- OOBROW
- OOINPQ
- OOINPX
- OOINPS
- OOSSET

la instancia de la cola de clúster abierta debe ser la instancia local. Si no hay ninguna instancia local de la cola, la llamada MQOPEN falla.

- Si las opciones especificadas no incluyen ninguna de las opciones anteriores, pero sí incluyen una o ambas de las siguientes:

- OOINQ
- OOOOUT

la instancia abierta es la instancia local si hay una y, de lo contrario, una instancia remota. Sin embargo, la instancia elegida por el gestor de colas puede ser alterada por una salida de carga de trabajo de clúster (si la hay).

Para obtener más información sobre las colas de clúster, consulte [Colas de clúster](#).

10. A las aplicaciones iniciadas por un supervisor desencadenante se les pasa el nombre de la cola asociada con la aplicación cuando se inicia la aplicación. Este nombre de cola se puede especificar en el parámetro **OBJDSC** para abrir la cola. Consulte la descripción de la estructura MQTMC para obtener más detalles.
11. Cuando se utiliza la opción OORLOQ, ya se devuelve la cola local cuando se abre una cola local, de alias o de modelo, pero no ocurre lo mismo cuando, por ejemplo, se abre una cola remota o una cola de clúster no local; el ResolvedQName y el ResolvedQMGrName se introducen con el RemoteQName y el RemoteQMGrName que se encuentran en la definición de la cola remota, o de forma similar con la cola de clúster remota elegida. Si se especifica OORLOQ al abrir, por ejemplo, una cola remota, ResolvedQName será ahora la cola de transmisión a la que se colocarán los mensajes. En ResolvedQMGrName se introducirá el nombre del gestor de colas local que aloja la cola de transmisión. Si un usuario tiene autorización para examinar, entrar o salir en una cola, tiene la autorización necesaria para especificar este distintivo en la llamada MQOPEN. No se necesita ninguna autorización especial.

Parámetros

La llamada MQOPEN tiene los parámetros siguientes:

HCONN (entero con signo de 10 dígitos)-entrada

Descriptor de conexión.

Este manejador representa la conexión con el gestor de colas. El valor de *HCONN* ha sido devuelto por una llamada MQCONN o MQCONNX anterior.

OBJDSC (MQOD)-entrada/salida

Descriptor de objeto.

Esta es una estructura que identifica el objeto que se va a abrir; consulte [“MQOD \(Descriptor de objeto\) activado IBM i”](#) en la página 1183 para obtener más detalles.

Si el campo *ODON* del parámetro **OBJDSC** es el nombre de una cola modelo, una cola local dinámica se crea con los atributos de la cola modelo; esto sucede independientemente de las opciones de apertura especificadas por el parámetro **OPTS**. Las operaciones posteriores que utilizan *HOBJ* devueltas por la llamada MQOPEN se realizan en la nueva cola dinámica y no en la cola modelo. Esto es cierto incluso para las llamadas MQINQ y MQSET. El nombre de la cola modelo en el parámetro **OBJDSC** se sustituye por el nombre de la cola dinámica creada. El tipo de la cola dinámica viene determinado por el valor del atributo **DefinitionType** de la cola modelo (consulte [“Atributos para colas”](#) en la página 1399). Para obtener información sobre las opciones de cierre aplicables a las colas dinámicas, consulte la descripción de la llamada MQCLOSE.

OPTS (entero con signo de 10 dígitos)-entrada

Opciones que controlan la acción de MQOPEN.

Se debe especificar al menos una de las opciones siguientes:

- OOBW
- OOINP* (sólo uno de ellos)
- OOINQ
- OOOOT
- OOSSET
- OORLQ

Se pueden especificar otras opciones según sea necesario. Si se necesita más de una opción, se pueden añadir los valores (no añadir la misma constante más de una vez). Las combinaciones que no son válidas se anotan; todas las demás combinaciones son válidas. Sólo se permiten las opciones que son aplicables al tipo de objeto especificado por *OBJDSC* (consulte [Opciones MQOPEN válidas para cada tipo de cola](#)).

Opciones de acceso: las opciones siguientes controlan el tipo de operaciones que se pueden realizar en el objeto:

OOINPQ

Abra la cola para obtener mensajes utilizando el valor predeterminado definido por la cola.

La cola se abre para su uso con las llamadas MQGET posteriores. El tipo de acceso es compartido o exclusivo, en función del valor del atributo de cola **DefInputOpenOption**; consulte [“Atributos para colas”](#) en la página 1399 para obtener más detalles.

Esta opción sólo es válida para colas locales, alias y modelo; no es válida para colas remotas, listas de distribución y objetos que no son colas.

OOINPS

Abra la cola para obtener mensajes con acceso compartido.

La cola se abre para su uso con las llamadas MQGET posteriores. La llamada puede realizarse correctamente si la cola está abierta actualmente por esta u otra aplicación con OOIINPS, pero falla con el código de razón RC2042 si la cola está abierta actualmente con OOIINPX.

Esta opción sólo es válida para colas locales, alias y modelo; no es válida para colas remotas, listas de distribución y objetos que no son colas.

OOINPX

Abra la cola para obtener mensajes con acceso exclusivo.

La cola se abre para su uso con las llamadas MQGET posteriores. La llamada falla con el código de razón RC2042 si la cola está abierta actualmente por esta u otra aplicación para entrada de cualquier tipo (OOINPS u OOIINPX).

Esta opción sólo es válida para colas locales, alias y modelo; no es válida para colas remotas, listas de distribución y objetos que no son colas.

Las siguientes notas se aplican a estas opciones:

- Sólo se puede especificar una de estas opciones.
- Una llamada MQOPEN con una de estas opciones puede ser satisfactoria incluso si el atributo de cola **InhibitGet** se establece en QAGETI (aunque las llamadas MQGET posteriores fallarán mientras el atributo se establece en este valor).
- Si la cola está definida como no compartible (es decir, el atributo de cola **Shareability** tiene el valor QANSHR), los intentos de abrir la cola para acceso compartido se tratan como intentos de abrir la cola con acceso exclusivo.
- Si se abre una cola alias con una de estas opciones, la prueba de uso exclusivo (o si otra aplicación tiene uso exclusivo) se realiza en la cola base en la que se resuelve el alias.
- Estas opciones no son válidas si *ODMN* es el nombre de un alias de gestor de colas; esto es cierto incluso si el valor del atributo **RemoteQMGrName** en la definición local de una cola remota utilizada para el alias de gestor de colas es el nombre del gestor de colas local.

OOBRW

Abra la cola para examinar los mensajes.

La cola se abre para su uso con llamadas MQGET posteriores con una de las opciones siguientes:

- GMBRWF
- GMBRWN
- GMBRWC

Esto se permite incluso si la cola está abierta actualmente para OOIINPX. Una llamada MQOPEN con la opción OOBRW establece un cursor de examen y lo sitúa lógicamente antes del primer mensaje en la cola; consulte el campo *GMOPT* descrito en [“MQGMO \(Opciones de obtención de mensajes\) activado IBM i”](#) en la página 1096 para obtener más información.

Esta opción sólo es válida para colas locales, alias y modelo; no es válida para colas remotas, listas de distribución y objetos que no son colas. Tampoco es válido si *ODMN* es el nombre de un alias de gestor de colas; esto es cierto incluso si el valor del atributo **RemoteQMGrName** en la definición local de una cola remota utilizada para el alias de gestor de colas es el nombre del gestor de colas local.

OOOUT

Abra la cola para transferir mensajes, o un tema o serie de tema para publicar mensajes.

La cola se abre para su uso con llamadas MQPUT posteriores.

Una llamada MQOPEN con esta opción puede ser satisfactoria incluso si el atributo de cola **InhibitPut** se establece en QAPUTI (aunque las llamadas MQPUT posteriores fallarán mientras el atributo se establezca en este valor).

Esta opción es válida para todos los tipos de cola, incluidas las listas de distribución y los temas.

OOINQ

Abrir objeto para consultar atributos.

La cola, la lista de nombres, la definición de proceso o el gestor de colas se abre para su uso con llamadas MQINQ posteriores.

Esta opción es válida para todos los tipos de objetos que no sean listas de distribución. No es válido si *ODMN* es el nombre de un alias de gestor de colas; esto es cierto incluso si el valor del

atributo **RemoteQMgrName** en la definición local de una cola remota utilizada para la asignación de alias de gestor de colas es el nombre del gestor de colas local.

OOSET

Abra la cola para establecer atributos.

La cola se abre para utilizarla con las llamadas MQSET posteriores.

Esta opción es válida para todos los tipos de cola que no sean listas de distribución. No es válido si *ODMN* es el nombre de una definición local de una cola remota; esto es cierto incluso si el valor del atributo **RemoteQMgrName** en la definición local de una cola remota utilizada para el alias de gestor de colas es el nombre del gestor de colas local.

Opciones de enlace: Las opciones siguientes se aplican cuando el objeto que se está abriendo es una cola de clúster; estas opciones controlan el enlace del descriptor de contexto de cola a una instancia de la cola de clúster:

OOBNDQ

Manejador de enlace a destino cuando se abre la cola.

Esto hace que el gestor de colas local enlace el descriptor de contexto de cola a una instancia de la cola de destino cuando se abre la cola. Como resultado, todos los mensajes colocados utilizando este descriptor de contexto se envían a la misma instancia de la cola de destino y por la misma ruta.

Esta opción solo es válida para colas y solo afecta a colas de clúster. Si se especifica en una cola que no sea de clúster, la opción se pasará por alto.

OOBNDN

No enlazar a un destino específico.

Esto detiene el gestor de colas local que enlaza el descriptor de contexto de cola con una instancia de la cola de destino. Como resultado, las llamadas MQPUT sucesivas que utilizan este descriptor de contexto pueden hacer que los mensajes se envíen a *diferentes* instancias de la cola de destino, o que se envíen a la misma instancia pero mediante rutas diferentes. También permite que el gestor de colas local, un gestor de colas remoto o un agente de canal de mensajes (MCA) cambien la instancia seleccionada más tarde, según las condiciones de red.

Nota: Las aplicaciones cliente y servidor que necesitan intercambiar una *serie* de mensajes para completar una transacción no deben utilizar OOBNDN (u OOBNDQ cuando *DefBind* tiene el valor BNDNOT), porque los mensajes sucesivos de la serie se pueden enviar a diferentes instancias de la aplicación de servidor.

Si se especifica OOBROW o una de las opciones OOINP* para una cola de clúster, el gestor de colas se fuerza a seleccionar la instancia local de la cola de clúster. Como resultado, el enlace del descriptor de contexto de cola es fijo, incluso si se especifica OOBNDN.

Si se especifica OOINQ con OOBNDN, las llamadas MQINQ sucesivas que utilizan ese manejador pueden consultar distintas instancias de la cola de clúster, aunque normalmente todas las instancias tienen los mismos valores de atributo.

OOBNDN sólo es válido para las colas y sólo afecta a las colas de clúster. Si se especifica en una cola que no sea de clúster, la opción se pasará por alto.

OOBNDQ

Utilice el enlace predeterminado para la cola.

Esto hace que el gestor de colas local enlace el descriptor de contexto de cola de la forma definida por el atributo de cola **DefBind**. El valor de este atributo es BNDOPN o BNDNOT.

OOBNDQ es el valor por omisión si no se especifican OOBNDQ y OOBNDN.

OOBNDQ se define para ayudar a la documentación del programa. No está previsto que esta opción se utilice con cualquiera de las otras dos opciones de vinculación, pero debido a que su valor es cero, no se puede detectar dicho uso.

Opciones de contexto: Las opciones siguientes controlan el proceso del contexto de mensaje:

OOSAVA

Guardar contexto cuando se recupere el mensaje.

La información de contexto está asociada con este descriptor de contexto de cola. Esta información se establece a partir del contexto de cualquier mensaje recuperado utilizando este descriptor de contexto. Para obtener más información sobre el contexto de mensaje, consulte [Contexto de mensaje](#) y [Control de la información de contexto](#).

Esta información de contexto se puede pasar a un mensaje que se coloca posteriormente en una cola utilizando las llamadas MQPUT o MQPUT1 . Consulte las opciones PMPASI y PMPASA descritas en [“MQPMO \(Opciones de mensajes de envío\) activado IBM i”](#) en la página 1198.

Hasta que un mensaje se haya recuperado correctamente, el contexto no se puede pasar a un mensaje que se está colocando en una cola.

Un mensaje recuperado utilizando una de las opciones de examen GMBRW* no tiene guardada su información de contexto (aunque los campos de contexto del parámetro **MSGDSC** se establecen después de un examen).

Esta opción sólo es válida para colas locales, alias y modelo; no es válida para colas remotas, listas de distribución y objetos que no son colas. Debe especificarse una de las opciones OOINP*.

OOPASI

Permitir que se pase el contexto de identidad.

Esto permite especificar la opción PMPASI en el parámetro **PMO** cuando se coloca un mensaje en una cola; esto proporciona al mensaje la información de contexto de identidad de una cola de entrada que se ha abierto con la opción OOSAVA. Para obtener más información sobre el contexto de mensaje, consulte [Contexto de mensaje](#) y [Control de la información de contexto](#).

Debe especificarse la opción OOOUT.

Esta opción es válida para todos los tipos de cola, incluidas las listas de distribución.

OOPASA

Permitir que se pase todo el contexto.

Esto permite especificar la opción PMPASA en el parámetro **PMO** cuando se coloca un mensaje en una cola; esto proporciona al mensaje la información de contexto de identidad y origen de una cola de entrada que se ha abierto con la opción OOSAVA. Para obtener más información sobre el contexto de mensaje, consulte [Contexto de mensaje](#) y [Control de la información de contexto](#).

Esta opción implica OOPASI, que por lo tanto no es necesario especificar. Debe especificarse la opción OOOUT.

Esta opción es válida para todos los tipos de cola, incluidas las listas de distribución.

OOOSETI

Permitir que se establezca el contexto de identidad.

Esto permite especificar la opción PMSETI en el parámetro **PMO** cuando se coloca un mensaje en una cola; esto proporciona al mensaje la información de contexto de identidad contenida en el parámetro **MSGDSC** especificado en la llamada MQPUT o MQPUT1 . Para obtener más información sobre el contexto de mensaje, consulte [Contexto de mensaje](#) y [Control de la información de contexto](#).

Esta opción implica OOPASI, que por lo tanto no es necesario especificar. Debe especificarse la opción OOOUT.

Esta opción es válida para todos los tipos de cola, incluidas las listas de distribución.

OOSETA

Permitir que se establezca todo el contexto.

Esto permite especificar la opción PMSETA en el parámetro **PMO** cuando se coloca un mensaje en una cola; esto proporciona al mensaje la información de contexto de identidad y origen contenida en el parámetro **MSGDSC** especificado en la llamada MQPUT o MQPUT1 . Para obtener

más información sobre el contexto de mensaje, consulte [Contexto de mensaje](#) y [Control de la información de contexto](#).

Esta opción implica las siguientes opciones, por lo que no es necesario especificarlas:

- OOPASI
- OOPASA
- OOOSETI

Debe especificarse la opción OOOOUT.

Esta opción es válida para todos los tipos de cola, incluidas las listas de distribución.

Otras opciones: Las opciones siguientes controlan la comprobación de autorización y lo que sucede cuando el gestor de colas se desactiva temporalmente:

OOALTU

Validar con el identificador de usuario especificado.

Esto indica que el campo *ODAU* del parámetro **OBJDSC** contiene un identificador de usuario que se va a utilizar para validar esta llamada MQOPEN. La llamada sólo puede realizarse correctamente si este *ODAU* está autorizado para abrir el objeto con las opciones de acceso especificadas, independientemente de si el identificador de usuario bajo el que se ejecuta la aplicación está autorizado para hacerlo. Sin embargo, esto no se aplica a las opciones de contexto especificadas, que siempre se comprueban con el identificador de usuario bajo el que se ejecuta la aplicación.

Esta opción es válida para todos los tipos de objeto.

OOFIQ

Fallar si el gestor de colas se está desactivando temporalmente.

Esta opción fuerza a que la llamada MQOPEN falle si el gestor de colas está en estado de desactivación temporal.

Esta opción es válida para todos los tipos de objeto.

OORLQ

Especifique el nombre de la cola local que se ha abierto.

Esta opción especifica que el ResolvedQName de la estructura MQOD (si está disponible) debe especificarse con el nombre de la cola local que se ha abierto. En ResolvedQMGrName se introducirá de forma similar el nombre del gestor de colas local que aloja la cola local.

Tabla 789. Opciones MQOPEN válidas para cada tipo de cola						
Opción	Alias ("1" en la página 1362)	Local y modelo	Remoto	Clúster no local	Lista de distribución	Tema
OOINPQ	✓	✓	-	-	-	-
OOINPS	✓	✓	-	-	-	-
OOINPX	✓	✓	-	-	-	-
OOBRW	✓	✓	-	-	-	-
OOOUT	✓	✓	✓	✓	✓	✓
OOINQ	✓	✓	"2" en la página 1362	✓	-	-
OOSET	✓	✓	"2" en la página 1362	-	-	-

Tabla 789. Opciones MQOPEN válidas para cada tipo de cola (continuación)						
Opción	Alias ("1" en la página 1362)	Local y modelo	Remoto	Clúster no local	Lista de distribución	Tema
OOBNDQ ("3" en la página 1362)	✓	✓	✓	✓	✓	-
OOBNDN ("3" en la página 1362)	✓	✓	✓	✓	✓	-
OOBNDQ ("3" en la página 1362)	✓	✓	✓	✓	✓	-
OOSAVA	✓	✓	-	-	-	-
OOPASI	✓	✓	✓	✓	✓	"5" en la página 1362
OOPASA	✓	✓	✓	✓	✓	"5" en la página 1362
OOOSETI	✓	✓	✓	✓	✓	"5" en la página 1362
OOSETA	✓	✓	✓	✓	✓	"5" en la página 1362
OOALTU	✓	✓	✓	✓	✓	✓
OOFIQ	✓	✓	✓	✓	✓	✓
OORLQ	✓	✓	✓	✓	-	-

Notas:

1. La validez de las opciones para los alias depende de la validez de la opción para la cola en la que se resuelve el alias.
2. Esta opción sólo es válida para la definición local de una cola remota.
3. Esta opción se puede especificar para cualquier tipo de cola, pero se ignora si la cola no es una cola de clúster.
4. Este atributo se ignora para un tema.
5. Estos atributos se pueden utilizar con un tema, pero sólo afectan al contexto establecido para el mensaje retenido, no a los campos de contexto enviados a cualquier suscriptor.

HOBJ (entero con signo de 10 dígitos)-salida

El manejador del objeto.

Este descriptor de contexto representa el acceso que se ha establecido al objeto. Debe especificarse en las llamadas de cola de mensajes posteriores que operan en el objeto. Deja de ser válido cuando se emite la llamada MQCLOSE o cuando termina la unidad de proceso que define el ámbito del descriptor de contexto.

El ámbito del descriptor de contexto está restringido a la unidad más pequeña de proceso paralelo soportado por la plataforma en la que se ejecuta la aplicación; el descriptor de contexto no es válido fuera de la unidad de proceso paralelo desde la que se ha emitido la llamada MQOPEN:

- En IBM i, el ámbito del descriptor de contexto es el trabajo que emite la llamada.

CMPCOD (entero con signo de 10 dígitos)-salida

Código de terminación.

Es uno de los siguientes:

CCOK

Realización satisfactoria.

CCWARN

Aviso (finalización parcial).

CCFAIL

La llamada no se ha realizado satisfactoriamente.

Declaración RPG

```
C*.1.....2.....3.....4.....5.....6.....7..
C                                CALLP      MQOPEN(HCONN : OBJDSC : OPTS :
C                                HOBJ : CMPCOD : REASON)
```

La definición de prototipo para la llamada es:

```
D*.1.....2.....3.....4.....5.....6.....7..
DMQOPEN          PR          EXTPROC('MQOPEN')
D* Connection handle
D HCONN          10I 0 VALUE
D* Object descriptor
D OBJDSC          468A
D* Options that control the action of MQOPEN
D OPTS          10I 0 VALUE
D* Object handle
D HOBJ          10I 0
D* Completion code
D CMPCOD          10I 0
D* Reason code qualifying CMPCOD
D REASON          10I 0
```

MQPUT (transferir mensaje) en IBM i

La llamada MQPUT coloca un mensaje en una cola, lista de distribución o en un tema. La cola, la lista de distribución o el tema ya deben estar abiertos.

- [“Sintaxis” en la página 1363](#)
- [“Notas de uso” en la página 1363](#)
 - [“Temas” en la página 1364](#)
 - [“MQPUT y MQPUT1” en la página 1364](#)
 - [“Colas de destino” en la página 1364](#)
 - [“Listas de distribución” en la página 1365](#)
 - [“Cabeceras” en la página 1367](#)
 - [“Almacenamiento intermedio” en la página 1367](#)
- [“Parámetros” en la página 1368](#)
- [“Declaración RPG” en la página 1373](#)

Sintaxis

MQPUT (*HCONN*, *HOBJ*, *MSGDSC*, *PMO*, *BUFLEN*, *BUFFER*, *CMPCOD*, *REASON*)

Notas de uso

Temas

Las notas siguientes se aplican al uso de temas:

1. Cuando se utiliza MQPUT para publicar mensajes sobre un tema, donde uno o más suscriptores de ese tema no pueden recibir la publicación debido a un problema con su cola de suscriptores (por ejemplo, está llena), el código de razón devuelto a la llamada MQPUT y el comportamiento de entrega depende del valor de los atributos PMSGDLV o NMSGDLV en el TOPIC. Tenga en cuenta que la entrega de una publicación a la cola de mensajes no entregados cuando se especifica RODLQ, o descartar el mensaje cuando se especifica RODISC, se considera una entrega satisfactoria del mensaje. Si no se entrega ninguna de las publicaciones, MQPUT volverá con RC2502. Esto puede ocurrir en los siguientes casos:
 - Un mensaje se publica en un TOPIC con PMSGDLV o NMSGDLV (dependiendo de la persistencia del mensaje) establecido en ALL y cualquier suscripción (duradera o no) tiene una cola que no puede recibir la publicación.
 - Un mensaje se publica en un TOPIC con PMSGDLV o NMSGDLV (dependiendo de la persistencia del mensaje) establecido en ALLDUR y una suscripción duradera tiene una cola que no puede recibir la publicación.
- MQPUT puede volver con RCNONE aunque las publicaciones no se hayan podido entregar a algunos suscriptores en los casos siguientes:
- Un mensaje se publica en un TOPIC con PMSGDLV o NMSGDLV (dependiendo de la persistencia del mensaje) establecido en ALLAVAIL y cualquier suscripción, duradera o no, tiene una cola que no puede recibir la publicación.
 - Un mensaje se publica en un TOPIC con PMSGDLV o NMSGDLV (en función de la persistencia del mensaje) establecido en ALLDUR y una suscripción no duradera tiene una cola que no puede recibir la publicación.
2. Si no hay suscriptores para el tema que se está utilizando, el mensaje publicado no se envía a ninguna cola y se descarta. No hace ninguna diferencia si este mensaje es persistente o no persistente, o si tiene una caducidad ilimitada o algún tiempo de caducidad pequeño, todavía se descarta si no hay suscriptores. La excepción a esto es si el mensaje se va a retener, en cuyo caso, aunque no se envía a las colas de ningún suscriptor, se almacena en el tema que se va a entregar a cualquier suscripción nueva o a cualquier suscriptor que solicite publicaciones retenidas utilizando MQSUBRQ.

MQPUT y MQPUT1

Las llamadas MQPUT y MQPUT1 se pueden utilizar para colocar mensajes en una cola; la llamada que se debe utilizar depende de las circunstancias

- La llamada MQPUT debe utilizarse cuando se van a colocar varios mensajes en la *misma* cola.

En primer lugar se emite una llamada MQOPEN que especifica la opción OOOOUT, seguida de una o varias solicitudes MQPUT para añadir mensajes a la cola; finalmente, la cola se cierra con una llamada MQCLOSE. Esto proporciona un mejor rendimiento que el uso repetido de la llamada MQPUT1 .

- La llamada MQPUT1 debe utilizarse cuando sólo se va a colocar *un* mensaje en una cola.

Esta llamada encapsula las llamadas MQOPEN, MQPUT y MQCLOSE en una sola llamada, minimizando el número de llamadas que se deben emitir.

Colas de destino

Si una aplicación coloca una secuencia de mensajes en la misma cola sin utilizar grupos de mensajes, el orden de estos mensajes se conserva si se cumplen las condiciones siguientes. Algunas condiciones se aplican tanto a las colas de destino locales como a las remotas; otras condiciones se aplican sólo a las colas de destino remotas.

Condiciones para colas de destino locales y remotas

- Todas las llamadas MQPUT están dentro de la misma unidad de trabajo, o ninguna de ellas está dentro de una unidad de trabajo.

Cuando los mensajes se colocan en una cola determinada dentro de una sola unidad de trabajo, los mensajes de otras aplicaciones pueden intercalarse con la secuencia de mensajes de la cola.

- Todas las llamadas MQPUT se realizan utilizando el mismo descriptor de objeto *HOBJ*.

En algunos entornos, la secuencia de mensajes también se conserva cuando se utilizan distintos manejadores de objetos, siempre que las llamadas se realicen desde la misma aplicación. El significado de "misma aplicación" viene determinado por el entorno:

– En IBM i, la aplicación es el trabajo.

- Los mensajes tienen todos la misma prioridad.

Condiciones adicionales para colas de destino remotas

- Sólo hay una vía de acceso desde el gestor de colas emisor al gestor de colas de destino.

Si existe la posibilidad de que algunos mensajes de la secuencia vayan a una vía de acceso diferente (por ejemplo, debido a la reconfiguración, el equilibrio de tráfico o la selección de vía de acceso en función del tamaño del mensaje), no se puede garantizar el orden de los mensajes en el gestor de colas de destino.

- Los mensajes no se colocan temporalmente en colas de mensajes no entregados en los gestores de colas de envío, intermedios o de destino.

Si uno o varios de los mensajes se colocan temporalmente en una cola de mensajes no entregados (por ejemplo, porque una cola de transmisión o la cola de destino está llena temporalmente), los mensajes pueden llegar a la cola de destino fuera de secuencia.

- Los mensajes son todos persistentes o todos no persistentes.

Si un canal de la ruta entre los gestores de colas de envío y de destino tiene su atributo **CDNPM** establecido en NPFAS, los mensajes no persistentes pueden saltar por delante de los mensajes persistentes, lo que da como resultado que el orden de los mensajes persistentes relativos a los mensajes no persistentes no se conserve. Sin embargo, se conserva el orden de los mensajes persistentes relativos entre sí y de los mensajes no persistentes relativos entre sí.

Si no se cumplen estas condiciones, se pueden utilizar grupos de mensajes para conservar el orden de los mensajes, pero tenga en cuenta que esto requiere que las aplicaciones emisora y receptora utilicen el soporte de agrupación de mensajes. Para obtener más información sobre los grupos de mensajes, consulte:

- *MDMFL* Campo de MQMD
- Opción PMLOGO en MQPMO
- Opción GMLOGO en MQGMO

Listas de distribución

Las siguientes notas se aplican al uso de listas de distribución.

1. Los mensajes se pueden transferir a una lista de distribución utilizando version-1 o version-2 MQPMO. Si se utiliza una MQPMO version-1 (o una MQPMO version-2 con *PMREC* igual a cero), la aplicación no puede proporcionar registros de mensajes de colocación ni registros de respuesta. Esto significa que no será posible identificar las colas que encuentran errores, si el mensaje se envía correctamente a algunas colas de la lista de distribución y no a otras.

Si la aplicación proporciona registros de mensajes de colocación o registros de respuesta, el campo *PMVER* debe establecerse en *PMVER2*.

Un MQPMO version-2 también se puede utilizar para enviar mensajes a una sola cola que no esté en una lista de distribución, asegurándose de que *PMREC* sea cero.

2. Los parámetros de código de terminación y código de razón se establecen de la forma siguiente:
 - Si todas las colocaciones en las colas de la lista de distribución son satisfactorias o fallan de la misma forma, el código de terminación y los parámetros de código de razón se establecen para

describir el resultado común. Los registros de respuesta MQRR (si los proporciona la aplicación) no se establecen en este caso.

Por ejemplo, si cada colocación es satisfactoria, el código de terminación se establece en CCOK y el código de razón es RCNONE; si cada colocación falla porque todas las colas están inhibidas para las colocaciones, los parámetros se establecen en CCFAIL y RC2051.

- Si las colocaciones en las colas de la lista de distribución no son todas satisfactorias o fallan de la misma forma:
 - El parámetro de código de terminación se establece en CCWARN si al menos una operación put se ha realizado correctamente, y en CCFAIL si todo ha fallado.
 - El parámetro de código de razón se establece en RC2136.
 - Los registros de respuesta (si los proporciona la aplicación) se establecen en los códigos de terminación individuales y los códigos de razón para las colas de la lista de distribución.

Si la colocación en un destino falla porque la apertura para dicho destino ha fallado, los campos del registro de respuesta se establecen en CCFAIL y RC2137; dicho destino se incluye en *PMIDC*.

3. Si un destino de la lista de distribución se resuelve en una cola local, el mensaje se coloca en esa cola en formato normal (es decir, no como un mensaje de lista de distribución). Si más de un destino se resuelve en la misma cola local, se coloca un mensaje en la cola para cada destino.

Si un destino de la lista de distribución se resuelve en una cola remota, se coloca un mensaje en la cola de transmisión adecuada. Cuando varios destinos se resuelvan en la misma cola de transmisión, podrá colocarse en la cola de transmisión un único mensaje de lista de distribución que contenga dichos destinos, incluso si dichos destinos no estuvieran adyacentes en la lista de destinos proporcionada por la aplicación. Sin embargo, esto sólo se puede hacer si la cola de transmisión da soporte a mensajes de lista de distribución (consulte el atributo de cola **DistLists** descrito en [“Atributos para colas”](#) en la página 1399).

Si la cola de transmisión no soporta listas de distribución, se coloca una copia del mensaje en formato normal en la cola de transmisión para cada destino que utiliza dicha cola de transmisión.

Si una lista de distribución con los datos de mensaje de aplicación es demasiado grande para una cola de transmisión, el mensaje de lista de distribución se divide en mensajes de lista de distribución más pequeños, cada uno de los cuales contiene menos destinos. Si los datos del mensaje de aplicación solo se ajustan a la cola, los mensajes de lista de distribución no se pueden utilizar en absoluto, y el gestor de colas genera una copia del mensaje en formato normal para cada destino que utiliza dicha cola de transmisión.

Si destinos diferentes tienen una prioridad de mensaje o persistencia de mensaje diferentes (esto puede ocurrir cuando la aplicación específica PRQDEF o PEQDEF), los mensajes no se conservan en el mismo mensaje de lista de distribución. En su lugar, el gestor de colas genera tantos mensajes de lista de distribución como sean necesarios para acomodar los diferentes valores de prioridad y persistencia.

4. Una colocación en una lista de distribución puede dar como resultado:

- Un único mensaje de lista de distribución, o
- Un número de mensajes de lista de distribución más pequeños, o
- Una combinación de mensajes de lista de distribución y mensajes normales, o
- Sólo mensajes normales.

Cuál de las situaciones anteriores depende de si:

- Los destinos de la lista son locales, remotos o una mezcla.
- Los destinos tienen la misma prioridad de mensaje y persistencia de mensaje.
- Las colas de transmisión pueden contener mensajes de lista de distribución.
- Las longitudes máximas de mensajes de las colas de transmisión son lo suficientemente grandes para acomodar el mensaje en formato de lista de distribución.

Sin embargo, independientemente de cuál de las situaciones anteriores, cada mensaje *físico* resultante (es decir, cada mensaje normal o mensaje de lista de distribución resultante de la operación de transferir) cuenta como un solo mensaje *uno* cuando:

- Comprobación de si la aplicación ha superado el número máximo permitido de mensajes en una unidad de trabajo (consulte el atributo del gestor de colas **MaxUncommittedMsgs**).
 - Comprobando si se cumplen las condiciones de desencadenamiento.
 - Incrementando las profundidades de cola y comprobando si se superaría la profundidad máxima de cola de las colas.
5. Cualquier cambio en las definiciones de cola que hubiera hecho que un descriptor de contexto no fuera válido si las colas se hubieran abierto individualmente (por ejemplo, un cambio en la vía de acceso de resolución), no hace que el descriptor de contexto de lista de distribución no sea válido. Sin embargo, da como resultado una anomalía para esa cola concreta cuando se utiliza el descriptor de contexto de lista de distribución en una llamada MQPUT posterior.

Cabeceras

Si un mensaje se coloca con una o más estructuras de cabecera IBM MQ al principio de los datos del mensaje de aplicación, el gestor de colas realiza determinadas comprobaciones en las estructuras de cabecera para verificar que son válidas. Si el gestor de colas detecta un error, la llamada falla con un código de razón adecuado. Las comprobaciones realizadas varían según las estructuras particulares que estén presentes. Además, las comprobaciones sólo se realizan si se utiliza un MQMD version-2 o posterior en la llamada MQPUT o MQPUT1 ; las comprobaciones no se realizan si se utiliza un MQMD version-1 , incluso si hay un MQMDE al principio de los datos del mensaje de aplicación.

El gestor de colas valida completamente las siguientes estructuras de cabecera IBM MQ : MQDH, MQMDE.

Para otras estructuras de cabecera de IBM MQ , el gestor de colas realiza alguna validación, pero no comprueba cada campo. Las estructuras que no están soportadas por el gestor de colas local y las estructuras que siguen a la primera MQDLH del mensaje no se validan.

Además de las comprobaciones generales en los campos de las estructuras IBM MQ , deben cumplirse las condiciones siguientes:

- Una estructura IBM MQ no debe dividirse en dos o más segmentos; la estructura debe estar totalmente contenida en un segmento.
- La suma de las longitudes de las estructuras de un mensaje PCF debe ser igual a la longitud especificada por el parámetro **BUFLEN** en la llamada MQPUT o MQPUT1 . Un mensaje PCF es un mensaje que tiene uno de los siguientes nombres de formato:
 - FMADMN
 - FMEVNT
 - FMPCF
- Las estructuras IBM MQ no se deben truncar, excepto en las situaciones siguientes en las que se permiten estructuras truncadas:
 - Mensajes que son mensajes de informe.
 - Mensajes PCF.
 - Mensajes que contienen una estructura MQDLH. (Las estructuras *que siguen* a la primera MQDLH se pueden truncar; las estructuras que preceden a la MQDLH no se pueden truncar.)

Almacenamiento intermedio

El parámetro **BUFFER** que se muestra en el ejemplo de programación RPG se declara como una serie; esto restringe la longitud máxima del parámetro a 256 bytes. Si se necesita un almacenamiento intermedio más grande, el parámetro debe declararse en su lugar como una estructura o como un campo en un archivo físico. Esto aumentará la longitud máxima posible a aproximadamente 32 KB.

Parámetros

La llamada MQPUT tiene los parámetros siguientes:

HCONN (entero con signo de 10 dígitos)-entrada

Descriptor de conexión.

Este manejador representa la conexión con el gestor de colas. El valor de *HCONN* ha sido devuelto por una llamada MQCONN o MQCONNX anterior.

HOBJ (entero con signo de 10 dígitos)-entrada

El manejador del objeto.

Este descriptor de contexto representa la cola a la que se añade el mensaje o el tema en el que se publica el mensaje. El valor de *HOBJ* lo ha devuelto una llamada MQOPEN anterior que ha especificado la opción OOOUT.

MSGDSC (MQMD)-entrada/salida

Descriptor de mensaje.

Esta estructura describe los atributos del mensaje que se envía y recibe información sobre el mensaje una vez completada la solicitud de colocación. Consulte [“MQMD \(descriptor de mensaje\) en IBM i”](#) en la página 1131 para obtener los detalles.

Si la aplicación proporciona un MQMD version-1, los datos del mensaje pueden tener como prefijo una estructura MQMDE para especificar valores para los campos que existen en el MQMD version-2 pero no en el MQMD version-1. El campo *MDfmt* en MQMD debe establecerse en FMMDE para indicar que hay un MQMDE presente. Consulte [“MQMDE \(Extensión del descriptor de mensajes\) en IBM i”](#) en la página 1177 para obtener más detalles.

PMO (MQPMO)-entrada/salida

Opciones que controlan la acción de MQPUT.

Consulte [“MQPMO \(Opciones de mensajes de envío\) activado IBM i”](#) en la página 1198 para obtener los detalles.

BUFLEN (entero con signo de 10 dígitos)-entrada

Longitud del mensaje en *BUFFER*.

Cero es válido e indica que el mensaje no contiene datos de aplicación. El límite superior para *BUFLEN* depende de varios factores:

- Si la cola de destino es una cola compartida, el límite superior es de 63 KB (64 512 bytes).
- Si el destino es una cola local o se resuelve en una cola local (pero no es una cola compartida), el límite superior depende de si:
 - El gestor de colas local da soporte a la segmentación.
 - La aplicación emisora especifica el distintivo que permite al gestor de colas segmentar el mensaje. Este distintivo es MFSEGA, y se puede especificar en un MQMD de version-2 o en un MQMDE utilizado con un MQMD de version-1.

Si se cumplen ambas condiciones, *BUFLEN* no puede exceder de 999 999 999 menos el valor del campo *MDOFF* en MQMD. El mensaje lógico más largo que se puede transferir es, por lo tanto, 999 999 999 bytes (cuando *MDOFF* es cero). Sin embargo, las restricciones de recursos impuestas por el sistema operativo o el entorno en el que se ejecuta la aplicación pueden dar como resultado un límite inferior.

Si una o ambas de las condiciones descritas anteriormente no se cumplen, *BUFLEN* no puede exceder el atributo **MaxMsgLength** y el atributo **MaxMsgLength** del gestor de colas.

- Si el destino es una cola remota o se resuelve en una cola remota, se aplican las condiciones para las colas locales, *pero en cada gestor de colas a través del cual debe pasar el mensaje para llegar a la cola de destino*; en particular:

1. La cola de transmisión local utilizada para almacenar el mensaje temporalmente en el gestor de colas local
2. Colas de transmisión intermedias (si las hay) utilizadas para almacenar el mensaje en los gestores de colas de la ruta entre los gestores de colas local y de destino
3. La cola de destino en el gestor de colas de destino

Por lo tanto, el mensaje más largo que se puede transferir está gobernado por el más restrictivo de estas colas y gestores de colas.

Cuando un mensaje está en una cola de transmisión, la información adicional reside en los datos del mensaje, y esto reduce la cantidad de datos de aplicación que pueden transportarse. En esta situación, se recomienda que los bytes LNMHD se resten de los valores *MaxMsgLength* de las colas de transmisión al determinar el límite para *BUFLEN*.

Nota: Sólo el incumplimiento de la condición 1 se puede diagnosticar de forma síncrona (con el código de razón RC2030 o RC2031) cuando se coloca el mensaje. Si no se cumplen las condiciones 2 o 3, el mensaje se redirige a una cola de mensajes no entregados, ya sea en un gestor de colas intermedio o en el gestor de colas de destino. Si esto sucede, se genera un mensaje de informe si el remitente lo ha solicitado.

BUFFER (serie de bits de 1 byte x BUFLEN)-entrada

Los datos del mensaje.

Se trata de un almacenamiento intermedio que contiene los datos de aplicación que se van a enviar. El almacenamiento intermedio debe estar alineado en un límite adecuado a la naturaleza de los datos del mensaje. La alineación de 4 bytes debe ser adecuada para la mayoría de los mensajes (incluidos los mensajes que contienen estructuras de cabecera MQ), pero algunos mensajes pueden requerir una alineación más estricta. Por ejemplo, un mensaje que contiene un entero binario de 64 bits puede requerir una alineación de 8 bytes.

Si *BUFFER* contiene datos de tipo carácter, datos numéricos o ambos, los campos *MDCSI* y *MDENC* del parámetro **MSGDSC** deben establecerse en los valores adecuados para los datos; esto permitirá al receptor del mensaje convertir los datos (si es necesario) en el juego de caracteres y la codificación utilizados por el receptor.

Nota: Todos los demás parámetros de la llamada MQPUT deben estar en el juego de caracteres proporcionado por el atributo de gestor de colas **CodedCharSetId** y la codificación del gestor de colas local proporcionada por ENNAT.

CMPCOD (entero con signo de 10 dígitos)-salida

Código de terminación.

Es uno de los siguientes:

CCOK

Realización satisfactoria.

CCWARN

Aviso (finalización parcial).

CCFAIL

La llamada no se ha realizado satisfactoriamente.

REASON (entero con signo de 10 dígitos)-salida

Código de razón que califica *CMPCOD*.

Si *CMPCOD* es CCOK:

RCNONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CMPCOD* es CCWARN:

RC2104

(2104, X'838 ') No se reconoce la opción de informe en el descriptor de mensaje.

RC2136

(2136, X'858 ') Se han devuelto varios códigos de razón.

Si *CMPCOD* es CCFAIL:

RC2004

(2004, X'7D4') El parámetro almacenamiento intermedio no es válido.

RC2005

(2005, X'7D5') El parámetro de longitud de almacenamiento intermedio no es válido.

RC2009

(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

RC2013

(2013, X'7DD') Hora de caducidad no válida.

RC2014

(2014, X'7DE') Código de respuesta no válido.

RC2018

(2018, X'7E2') El manejador de conexión no es válido.

RC2019

(2019, X'7E3') El manejador de objeto no es válido.

RC2024

(2024, X'7E8') No pueden manejarse más mensajes dentro de la unidad de trabajo actual.

RC2026

(2026, X'7EA') El descriptor de mensaje no es válido.

RC2027

(2027, X'7EB') Falta cola de respuesta.

RC2029

(2029, X'7ED') El tipo de mensaje en el descriptor de mensaje no es válido.

RC2030

(2030, X'7EE') Longitud de mensaje superior a la máxima para la cola.

RC2031

(2031, X'7EF') Longitud del mensaje superior a la máxima para el gestor de colas.

RC2039

(2039, X'7F7') Cola no abierta para salida.

RC2041

(2041, X'7F9') La definición de objeto ha cambiado desde que se ha abierto.

RC2046

(2046, X'7FE') Las opciones no son válidas o no son coherentes.

RC2047

(2047, X'7FF') Persistencia no válida.

RC2048

(2048, X'800 ') La cola no admite mensajes persistentes.

RC2050

(2050, X'802 ') La prioridad del mensaje no es válida.

RC2051

(2051, X'803 ') Llamadas de colocación inhibidas para la cola.

RC2052

(2052, X'804') La cola se ha suprimido.

RC2053

(2053, X'805 ') La cola ya contiene el número máximo de mensajes.

RC2056

(2056, X'808 ') No hay espacio disponible en disco para la cola.

RC2058

(2058, X'80A') El nombre del gestor de colas no es válido o no se conoce.

RC2059

(2059, X'80B') El gestor de colas no está disponible para efectuar la conexión.

RC2061

(2061, X'80D') Las opciones de informe del descriptor de mensaje no son válidas.

RC2071

(2071, X'817') No hay suficiente almacenamiento disponible.

RC2072

(2072, X'818 ') El soporte de punto de sincronización no está disponible.

RC2093

(2093, X'82D') Cola no abierta para pasar todo el contexto.

RC2094

(2094, X'82E') Cola no abierta para pasar contexto de identidad.

RC2095

(2095, X'82F') Cola no abierta para establecer todo el contexto.

RC2096

(2096, X'830 ') La cola no está abierta para el contexto de identidad establecido.

RC2097

(2097, X'831 ') El descriptor de contexto de cola al que se hace referencia no guarda el contexto.

RC2098

(2098, X'832 ') Contexto no disponible para el descriptor de contexto de cola al que se hace referencia.

RC2101

(2101, X'835') El objeto se ha dañado.

RC2102

(2102, X'836') No hay disponibles suficientes recursos del sistema.

RC2135

(2135, X'857 ') La estructura de cabecera de distribución no es válida.

RC2136

(2136, X'858 ') Se han devuelto varios códigos de razón.

RC2137

(2137, X'859 ') El objeto no se ha abierto correctamente.

RC2149

(2149, X'865 ') Estructuras PCF no válidas.

RC2154

(2154, X'86A') Número de registros presentes no válido.

RC2156

(2156, X'86C') Los registros de respuesta no son válidos.

RC2158

(2158, X'86E') Las banderas de registro de mensajes no son válidas.

RC2159

(2159, X'86F') Los registros de mensajes no son válidos.

RC2161

(2161, X'871') El gestor de colas se está desactivando temporalmente.

RC2162

(2162, X'872') El gestor de colas está concluyendo.

RC2173

(2173, X'87D') La estructura de opciones de venta no es válida.

RC2185

(2185, X'889 ') Especificación de persistencia incoherente.

RC2188

(2188, X'88C') Llamada rechazada por la salida de la carga de trabajo del cluster.

RC2189

(2189, X'88D') La resolución de nombres de clúster ha fallado.

RC2195

(2195, X'893') Se ha producido un error inesperado.

RC2219

(2219, X'8AB') Se ha vuelto a introducir la llamada « MQI » antes de que finalizara la llamada anterior.

RC2241

(2241, X'8C1') El grupo de mensajes no está completo.

RC2242

(2242, X'8C2') El mensaje lógico no está completo.

RC2245

(2245, X'8C5') Especificación incoherente de unidad de trabajo.

RC2248

(2248, X'8C8') La extensión del descriptor de mensaje no es válida.

RC2249

(2249, X'8C9') Los indicadores de mensaje no son válidos.

RC2250

(2250, X'8CA') El número de secuencia de mensaje no es válido.

RC2251

(2251, X'8CB') Desplazamiento de segmento de mensaje no válido.

RC2252

(2252, X'8CC') Longitud original no válida.

RC2253

(2253, X'8CD') La longitud de los datos del segmento de mensaje es cero.

RC2255

(2255, X'8CF') La unidad de trabajo no está disponible para ser utilizada por el gestor de colas.

RC2257

(2257, X'8D1') La versión del MQMD suministrado es incorrecta.

RC2258

(2258, X'8D2') Identificador de grupo no válido.

RC2266

(2266, X'8DA') Fallo en la salida de la carga de trabajo del cluster.

RC2269

(2269, X'8DD') Error de recurso de cluster.

RC2270

(2270, X'8DE') No hay colas de destino disponibles.

RC2420

(2420) Se ha emitido una llamada MQPUT, pero los datos del mensaje contienen una estructura MQEPH que no es válida.

RC2479

(2479, X'9AF') No se ha podido retener la publicación.

RC2480

(2480, X'9B0') El tipo de destino ha cambiado: el alias queue se refería a una cola pero ahora se refiere a un tema.

RC2502

(2502, X'9C6') La publicación ha fallado y no se ha entregado a ningún suscriptor

RC2551

(2551, X'9F7') La cadena de selección especificada no está disponible.

RC2554

(2554, X'9FA') No se ha podido analizar el contenido del mensaje para determinar si debe entregarse a un abonado con un selector de mensajes ampliado.

Declaración RPG

```
C*.1.....2.....3.....4.....5.....6.....7..
C                                CALLP      MQPUT(HCONN : HOBJ : MSGDSC : PMO :
C                                BUFLN : BUFFER : CMPCOD :
C                                REASON)
```

La definición de prototipo para la llamada es:

```
D*.1.....2.....3.....4.....5.....6.....7..
DMQPUT      PR                      EXTPROC('MQPUT')
D* Connection handle
D HCONN                      10I 0 VALUE
D* Object handle
D HOBJ                      10I 0 VALUE
D* Message descriptor
D MSGDSC                      364A
D* Options that control the action of MQPUT
D PMO                      200A
D* Length of the message in Buffer
D BUFLN                      10I 0 VALUE
D* Message data
D BUFFER                      *    VALUE
D* Completion code
D CMPCOD                      10I 0
D* Reason code qualifying CMPCOD
D REASON                      10I 0
```

MQPUT1 (Colocar un mensaje) en IBM i

La llamada MQPUT1 coloca un mensaje en una cola o lista de distribución, o en un tema. No es necesario que la cola, la lista de distribución o el tema estén abiertos.

- [“Sintaxis” en la página 1373](#)
- [“Notas de uso” en la página 1373](#)
- [“Parámetros” en la página 1374](#)
- [“Declaración RPG” en la página 1379](#)

Sintaxis

MQPUT1 (*HCONN*, *OBJDSC*, *MSGDSC*, *PMO*, *BUFLN*, *BUFFER*, *CMPCOD*, *REASON*)

Notas de uso

1. Las llamadas MQPUT y MQPUT1 se pueden utilizar para colocar mensajes en una cola; la llamada que se debe utilizar depende de las circunstancias:
 - La llamada MQPUT debe utilizarse cuando se van a colocar varios mensajes en la *misma* cola.

En primer lugar se emite una llamada MQOPEN que especifica la opción OOOUT, seguida de una o varias solicitudes MQPUT para añadir mensajes a la cola; finalmente, la cola se cierra con una llamada MQCLOSE. Esto proporciona un mejor rendimiento que el uso repetido de la llamada MQPUT1.

- La llamada MQPUT1 debe utilizarse cuando sólo se va a colocar *un* mensaje en una cola.

Esta llamada encapsula las llamadas MQOPEN, MQPUT y MQCLOSE en una sola llamada, minimizando el número de llamadas que se deben emitir.

2. Si una aplicación coloca una secuencia de mensajes en la misma cola sin utilizar grupos de mensajes, el orden de dichos mensajes se conserva si se cumplen determinadas condiciones. Sin embargo, en la mayoría de los entornos, la llamada MQPUT1 no satisface estas condiciones, por lo que no conserva el orden de los mensajes. En su lugar, se debe utilizar la llamada MQPUT en estos entornos. Consulte las notas de uso en la descripción de la llamada MQPUT para obtener detalles.
3. La llamada MQPUT1 se puede utilizar para transferir mensajes a listas de distribución. Para obtener información general sobre esto, consulte las notas de uso para las llamadas MQOPEN y MQPUT.

Las diferencias siguientes se aplican cuando se utiliza la llamada MQPUT1 :

- a. Si la aplicación proporciona registros de respuesta MQRR, se deben proporcionar utilizando la estructura MQOD; no se pueden proporcionar utilizando la estructura MQPMO.
- b. MQPUT1 nunca devuelve el código de razón RC2137 en los registros de respuesta; si una cola no se puede abrir, el registro de respuesta para dicha cola contiene el código de razón real resultante de la operación de apertura.

Si una operación de apertura para una cola es satisfactoria con un código de terminación de CCWARN, el código de terminación y el código de razón del registro de respuesta para dicha cola se sustituyen por los códigos de terminación y razón resultantes de la operación de transferencia.

Al igual que con las llamadas MQOPEN y MQPUT, el gestor de colas establece los registros de respuesta (si se proporcionan) sólo cuando el resultado de la llamada no es el mismo para todas las colas de la lista de distribución; esto se indica mediante la llamada que se completa con el código de razón RC2136.

4. Si se utiliza la llamada MQPUT1 para colocar un mensaje en una cola de clúster, la llamada se comporta como si se hubiera especificado OOBNDN en la llamada MQOPEN.
5. Si un mensaje se coloca con una o más estructuras de cabecera IBM MQ al principio de los datos del mensaje de aplicación, el gestor de colas realiza determinadas comprobaciones en las estructuras de cabecera para verificar que son válidas. Para obtener más información sobre esto, consulte las notas de uso para la llamada MQPUT.
6. Si surge más de una de las situaciones de aviso (consulte el parámetro **CMPCOD**), el código de razón devuelto es el *primero* de la lista siguiente que se aplica:
 - a. RC2136
 - b. RC2242
 - c. RC2241
 - d. RC2049 o RC2104
7. El parámetro **BUFFER** que se muestra en el ejemplo de programación RPG se declara como una serie; esto restringe la longitud máxima del parámetro a 256 bytes. Si se necesita un almacenamiento intermedio más grande, el parámetro debe declararse en su lugar como una estructura o como un campo en un archivo físico. Esto aumentará la longitud máxima posible a aproximadamente 32 KB.

Parámetros

La llamada MQPUT1 tiene los parámetros siguientes:

HCONN (entero con signo de 10 dígitos)-entrada

Descriptor de conexión.

Este manejador representa la conexión con el gestor de colas. El valor de *HCONN* ha sido devuelto por una llamada MQCONN o MQCONNEX anterior.

OBJDSC (MQOD)-entrada/salida

Descriptor de objeto.

Es una estructura que identifica la cola a la que se añade el mensaje. Consulte [“MQOD \(Descriptor de objeto\) activado IBM i”](#) en la página 1183 para obtener los detalles.

El usuario debe tener autorización para abrir la cola para salida. La cola **no** debe ser una cola modelo.

MSGDSC (MQMD)-entrada/salida

Descriptor de mensaje.

Esta estructura describe los atributos del mensaje que se envía y recibe información de retorno una vez completada la solicitud de colocación. Consulte [“MQMD \(descriptor de mensaje\) en IBM i”](#) en la página 1131 para obtener los detalles.

Si la aplicación proporciona un MQMD version-1, los datos del mensaje pueden tener como prefijo una estructura MQMDE para especificar valores para los campos que existen en el MQMD version-2 pero no en el MQMD version-1. El campo *MDFMT* en MQMD debe establecerse en FMMDE para indicar que hay un MQMDE presente. Consulte [“MQMDE \(Extensión del descriptor de mensajes\) en IBM i”](#) en la página 1177 para obtener más detalles.

PMO (MQPMO)-entrada/salida

Opciones que controlan la acción de MQPUT1.

Consulte [“MQPMO \(Opciones de mensajes de envío\) activado IBM i”](#) en la página 1198 para obtener los detalles.

BUFLEN (entero con signo de 10 dígitos)-entrada

Longitud del mensaje en *BUFFER*.

Cero es válido e indica que el mensaje no contiene datos de aplicación. El límite superior depende de varios factores; consulte la descripción del parámetro **BUFLEN** de la llamada MQPUT para obtener más detalles.

BUFFER (serie de bits de 1 byte x BUFLEN)-entrada

Los datos del mensaje.

Es un almacenamiento intermedio que contiene los datos de mensaje de aplicación que se van a enviar. El almacenamiento intermedio debe estar alineado en un límite adecuado a la naturaleza de los datos del mensaje. La alineación de 4 bytes debe ser adecuada para la mayoría de los mensajes (incluidos los mensajes que contienen estructuras de cabecera IBM MQ), pero algunos mensajes pueden requerir una alineación más estricta. Por ejemplo, un mensaje que contiene un entero binario de 64 bits puede requerir una alineación de 8 bytes.

Si *BUFFER* contiene datos de tipo carácter, datos numéricos o ambos, los campos *MDCSI* y *MDENC* del parámetro **MSGDSC** deben establecerse en los valores adecuados para los datos; esto permitirá al receptor del mensaje convertir los datos (si es necesario) en el juego de caracteres y la codificación utilizados por el receptor.

Nota: Todos los demás parámetros de la llamada MQPUT1 deben estar en el juego de caracteres proporcionado por el atributo de gestor de colas **CodedCharSetId** y la codificación del gestor de colas local proporcionada por ENNAT.

CMPCOD (entero con signo de 10 dígitos)-salida

Código de terminación.

Es uno de los siguientes:

CCOK

Realización satisfactoria.

CCWARN

Aviso (finalización parcial).

CCFAIL

La llamada no se ha realizado satisfactoriamente.

REASON (entero con signo de 10 dígitos)-salida

Código de razón que califica *CMPCOD*.

Si *CMPCOD* es CCOK:

RCNONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CMPCOD* es CCWARN:

RC2104

(2104, X'838 ') No se reconoce la opción de informe en el descriptor de mensaje.

RC2136

(2136, X'858 ') Se han devuelto varios códigos de razón.

RC2049

(2049, X'801 ') La prioridad de mensaje sobrepasa el valor máximo soportado.

RC2241

(2241, X'8C1') El grupo de mensajes no está completo.

RC2242

(2242, X'8C2') El mensaje lógico no está completo.

Si *CMPCOD* es CCFAIL:

RC2001

(2001, X'7D1') Alias cola base no es un tipo válido.

RC2004

(2004, X'7D4') El parámetro almacenamiento intermedio no es válido.

RC2005

(2005, X'7D5') El parámetro de longitud de almacenamiento intermedio no es válido.

RC2009

(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

RC2013

(2013, X'7DD') Hora de caducidad no válida.

RC2014

(2014, X'7DE') Código de respuesta no válido.

RC2017

(2017, X'7E1') No hay más asas disponibles.

RC2018

(2018, X'7E2') El manejador de conexión no es válido.

RC2024

(2024, X'7E8') No pueden manejarse más mensajes dentro de la unidad de trabajo actual.

RC2026

(2026, X'7EA') El descriptor de mensaje no es válido.

RC2027

(2027, X'7EB') Falta cola de respuesta.

RC2029

(2029, X'7ED') El tipo de mensaje en el descriptor de mensaje no es válido.

RC2030

(2030, X'7EE') Longitud de mensaje superior a la máxima para la cola.

RC2031

(2031, X'7EF') Longitud del mensaje superior a la máxima para el gestor de colas.

RC2035

(2035, X'7F3') Acceso no autorizado.

RC2042

(2042, X'7FA') Objeto ya abierto con opciones conflictivas.

RC2043

(2043, X'7FB') Tipo de objeto no válido.

RC2044

(2044, X'7FC') La estructura del descriptor de objeto no es válida.

RC2046

(2046, X'7FE') Las opciones no son válidas o no son coherentes.

RC2047

(2047, X'7FF') Persistencia no válida.

RC2048

(2048, X'800 ') La cola no admite mensajes persistentes.

RC2050

(2050, X'802 ') La prioridad del mensaje no es válida.

RC2051

(2051, X'803 ') Llamadas de colocación inhibidas para la cola.

RC2052

(2052, X'804') La cola se ha suprimido.

RC2053

(2053, X'805 ') La cola ya contiene el número máximo de mensajes.

RC2056

(2056, X'808 ') No hay espacio disponible en disco para la cola.

RC2057

(2057, X'809 ') Tipo de cola no válido.

RC2058

(2058, X'80A') El nombre del gestor de colas no es válido o no se conoce.

RC2059

(2059, X'80B') El gestor de colas no está disponible para efectuar la conexión.

RC2061

(2061, X'80D') Las opciones de informe del descriptor de mensaje no son válidas.

RC2063

(2063, X'80F') Se ha producido un error de seguridad.

RC2071

(2071, X'817') No hay suficiente almacenamiento disponible.

RC2072

(2072, X'818 ') El soporte de punto de sincronización no está disponible.

RC2082

(2082, X'822 ') Cola base alias desconocida.

RC2085

(2085, X'825 ') Nombre de objeto desconocido.

RC2086

(2086, X'826 ') Gestor de colas de objetos desconocido.

RC2087

(2087, X'827 ') Gestor de colas remoto desconocido.

RC2091

(2091, X'82B') Cola de transmisión no local.

RC2092

(2092, X'82C') Cola de transmisión con uso incorrecto.

RC2097

(2097, X'831 ') El descriptor de contexto de cola al que se hace referencia no guarda el contexto.

RC2098

(2098, X'832 ') Contexto no disponible para el descriptor de contexto de cola al que se hace referencia.

RC2101

(2101, X'835') El objeto se ha dañado.

RC2102

(2102, X'836') No hay disponibles suficientes recursos del sistema.

RC2135

(2135, X'857 ') La estructura de cabecera de distribución no es válida.

RC2136

(2136, X'858 ') Se han devuelto varios códigos de razón.

RC2149

(2149, X'865 ') Estructuras PCF no válidas.

RC2154

(2154, X'86A') Número de registros presentes no válido.

RC2155

(2155, X'86B') Los registros del objeto no son válidos.

RC2156

(2156, X'86C') Los registros de respuesta no son válidos.

RC2158

(2158, X'86E') Las banderas de registro de mensajes no son válidas.

RC2159

(2159, X'86F') Los registros de mensajes no son válidos.

RC2161

(2161, X'871') El gestor de colas se está desactivando temporalmente.

RC2162

(2162, X'872') El gestor de colas está concluyendo.

RC2173

(2173, X'87D') La estructura de opciones de venta no es válida.

RC2184

(2184, X'888 ') El nombre de cola remota no es válido.

RC2188

(2188, X'88C') Llamada rechazada por la salida de la carga de trabajo del cluster.

RC2189

(2189, X'88D') La resolución de nombres de clúster ha fallado.

RC2195

(2195, X'893') Se ha producido un error inesperado.

RC2196

(2196, X'894 ') Cola de transmisión desconocida.

RC2197

(2197, X'895 ') Cola de transmisión predeterminada desconocida.

RC2198

(2198, X'896 ') La cola de transmisión predeterminada no es local.

RC2199

(2199, X'897 ') Error de uso de cola de transmisión predeterminado.

RC2258

(2258, X'8D2') Identificador de grupo no válido.

RC2248

(2248, X'8C8') La extensión del descriptor de mensaje no es válida.

RC2219

(2219, X'8AB') Se ha vuelto a introducir la llamada « MQI » antes de que finalizara la llamada anterior.

RC2249

(2249, X'8C9') Los indicadores de mensaje no son válidos.

RC2250

(2250, X'8CA') El número de secuencia de mensaje no es válido.

RC2251

(2251, X'8CB') Desplazamiento de segmento de mensaje no válido.

RC2252

(2252, X'8CC') Longitud original no válida.

RC2253

(2253, X'8CD') La longitud de los datos del segmento de mensaje es cero.

RC2255

(2255, X'8CF') La unidad de trabajo no está disponible para ser utilizada por el gestor de colas.

RC2257

(2257, X'8D1') La versión del MQMD suministrado es incorrecta.

RC2266

(2266, X'8DA') Fallo en la salida de la carga de trabajo del cluster.

RC2269

(2269, X'8DD') Error de recurso de cluster.

RC2270

(2270, X'8DE') No hay colas de destino disponibles.

RC2420

(2420) Se ha emitido una llamada MQPUT1 , pero los datos del mensaje contienen una estructura MQEPH que no es válida.

RC2551

(2551, X'9F7') La cadena de selección especificada no está disponible.

RC2554

(2554, X'9FA') No se ha podido analizar el contenido del mensaje para determinar si debe entregarse a un abonado con un selector de mensajes ampliado.

Declaración RPG

```

C*..1.....2.....3.....4.....5.....6.....7..
C                                CALLP      MQPUT1(HCONN : OBJDSC : MSGDSC :
C                                PMO : BUFLN : BUFFER :
C                                CMPCOD : REASON)

```

La definición de prototipo para la llamada es:

```

D*..1.....2.....3.....4.....5.....6.....7..
DMQPUT1      PR      EXTPROC('MQPUT1')
D* Connection handle
D HCONN      10I 0 VALUE
D* Object descriptor
D OBJDSC      468A
D* Message descriptor
D MSGDSC      364A
D* Options that control the action of MQPUT1
D PMO      200A
D* Length of the message in BUFFER
D BUFLN      10I 0 VALUE

```

```

D* Message data
D BUFFER                      *   VALUE
D* Completion code
D CMPCOD                      10I 0
D* Reason code qualifying CMPCOD
D REASON                      10I 0

```

MQSET (Establecer atributos de objeto) en IBM i

La llamada MQSET se utiliza para cambiar los atributos de un objeto representado por un identificador. El objeto debe ser una cola.

- [“Sintaxis” en la página 1380](#)
- [“Notas de uso” en la página 1380](#)
- [“Parámetros” en la página 1380](#)
- [“Declaración RPG” en la página 1384](#)

Sintaxis

MQSET (*HCONN, HOBJ, SELCNT, SELS, IACNT, INTATR, CALEN, CHRATR, CMPCOD, REASON*)

Notas de uso

1. Mediante esta llamada, la aplicación puede especificar una matriz de atributos enteros, una colección de cadenas de atributos de caracteres, o ambos. Si no se producen errores, todos los atributos especificados se establecen simultáneamente. Si se produce un error (por ejemplo, si un selector no es válido o se intenta establecer un atributo con un valor que no es válido), la llamada falla y no se establece ningún atributo.
2. Los valores de los atributos se pueden determinar utilizando la llamada MQINQ; consulte [“MQINQ \(Consultar atributos de objeto\) en IBM i” en la página 1335](#) para obtener más detalles.
Nota: No todos los atributos cuyos valores se pueden consultar mediante la llamada MQINQ pueden cambiar sus valores mediante la llamada MQSET. Por ejemplo, con esta llamada no se pueden establecer atributos de objetos de proceso ni de gestores de colas.
3. Los cambios en los atributos se conservan tras los reinicios del gestor de colas (excepto las modificaciones en las colas dinámicas temporales, que no se conservan tras los reinicios del gestor de colas).
4. No se pueden cambiar los atributos de una cola de modelos utilizando la llamada MQSET. Sin embargo, si abre una cola de modelos utilizando la llamada MQOPEN con la opción MQOO_SET, puede utilizar la llamada MQSET para establecer los atributos de la cola local dinámica creada por la llamada MQOPEN.
5. Si el objeto que se está configurando es una cola de clúster, debe existir una instancia local de la cola de clúster para que la apertura se realice correctamente.

Para obtener más información sobre los atributos de los objetos, consulte:

- [“Atributos para colas” en la página 1399](#)
- [“Atributos de las listas de nombres” en la página 1429](#)
- [“Atributos para definiciones de proceso en IBM i” en la página 1430](#)
- [“Atributos del gestor de colas en IBM i” en la página 1432](#)

Parámetros

La llamada MQSET tiene los siguientes parámetros:

HCONN (entero con signo de 10 dígitos) - entrada

Descriptor de conexión.

Este manejador representa la conexión con el gestor de colas. El valor de HCONN fue devuelto por una llamada anterior a MQCONN o MQCONNX.

HOBJ (entero con signo de 10 dígitos) - entrada

El manejador del objeto.

Este identificador representa el objeto de cola con los atributos que se van a establecer. El identificador fue devuelto por una llamada MQOPEN anterior que especificaba la opción OOSSET.

SELCNT (entero con signo de 10 dígitos) - entrada

Recuento de selectores.

Este es el recuento de selectores que se proporcionan en la SELS matriz. Es el número de atributos que se van a establecer. Cero es un valor válido. El número máximo permitido es 256.

SELS (entero con signo de 10 dígitos x SELCNT) - entrada

Matriz de selectores de atributos.

Se trata de una matriz de **SELCNT** selectores de atributos; cada selector identifica un atributo (entero o carácter) con un valor que se va a establecer.

Cada selector debe ser válido para el tipo de cola que HOBJ representa. Solo se permiten determinados valores de IA* y CA*; estos valores se enumeran más adelante en esta sección.

Los selectores se pueden especificar en cualquier orden. Los valores de los atributos que corresponden a selectores de atributos enteros (selectores IA*) deben especificarse en INTATR en el mismo orden en que aparecen estos selectores en SELS. Los valores de los atributos que corresponden a los selectores de atributos de caracteres (selectores CA*) deben especificarse en CHRATR el mismo orden en que aparecen dichos selectores. Los selectores IA* pueden intercalarse con los selectores CA*; solo es importante el orden relativo dentro de cada tipo.

No es un error especificar el mismo selector más de una vez; si se hace, el último valor especificado para un selector concreto es el que surte efecto.

Nota:

1. Los selectores de atributos de enteros y caracteres se asignan dentro de dos rangos diferentes; los selectores IA* residen dentro del rango IAFRST a IALAST, y los selectores CA* dentro del rango CAFRST a CALAST.

Para cada rango, las constantes IALSTU y CALSTU definen el valor más alto que aceptará el gestor de colas.

2. Si todos los selectores IA* aparecen primero, se pueden utilizar los mismos números de elemento para direccionar los elementos correspondientes en las matrices SELS INTATR y.

Los atributos que se pueden configurar se enumeran en la siguiente tabla. No se pueden establecer otros atributos mediante esta llamada. Para los selectores de atributos CA*, la constante que define la longitud en bytes de la cadena que se requiere en CHRATR se proporciona entre paréntesis.

Tabla 790. Seletores de atributos MQSET para colas		
Selector	Descripción	Nota
CATRGD	Datos desencadenantes (LNTRGD).	“2” en la página 1382
IADIST	Compatibilidad con listas de distribución.	“1” en la página 1382

Tabla 790. Selektors de atributos MQSET para colas (continuación)

Selector	Descripción	Nota
IAIGET	Si se permiten las operaciones get.	
IAIPUT	Si se permiten las operaciones de inserción.	
IATRGC	Control de desencadenante.	“2” en la página 1382
IATRGD	Profundidad del desencadenante.	“2” en la página 1382
IATRGP	Umbral de prioridad del mensaje para activaciones.	“2” en la página 1382
IATRGT	Tipo de desencadenante.	“2” en la página 1382

Notas:

1. Solo compatible con las siguientes plataformas:

-  AIX
-  IBM i
-  Windows

y para IBM MQ los clientes conectados a estos sistemas.

2. No compatible con VSE/ESA.

IACNT (entero con signo de 10 dígitos) - entrada

Recuento de atributos enteros.

Este es el número de elementos de la INTATR matriz, y debe ser al menos igual al número de selectores IA* del **SELS** parámetro. Cero es un valor válido si no hay ninguno.

INTATR (rxIACNT de enteros con signo de 10 dígitos) - entrada

Matriz de atributos enteros.

Se trata de una matriz de valores de IACNT atributos enteros. Estos valores de atributo deben estar en el mismo orden que los selectores IA* en la SELS matriz.

CALEN (entero con signo de 10 dígitos) - entrada

Longitud del búfer de atributos de caracteres.

Esta es la longitud en bytes del **CHRATR** parámetro, y debe ser al menos la suma de las longitudes de los atributos de caracteres especificados en la SELS matriz. Cero es un valor válido si no hay selectores CA* en SELS.

CHRATR (cadena de caracteres de 1 byte x CALEN) - entrada

Atributos de los personajes.

Este es el búfer que contiene los valores de los atributos de los caracteres, concatenados entre sí. La longitud del búfer viene dada por el **CALEN** parámetro.

Los atributos de los caracteres deben especificarse en el mismo orden que los selectores CA* en la SELS matriz. La longitud de cada atributo de carácter es fija (véase SELS). Si el valor que se va a establecer para un atributo contiene menos caracteres no en blanco que la longitud definida del atributo, el valor en CHRATR debe rellenarse a la derecha con espacios en blanco para que el valor del atributo coincida con la longitud definida del atributo.

CMPCOD (entero con signo de 10 dígitos) - salida

Código de terminación.

Es uno de los siguientes:

CCOK

Realización satisfactoria.

CCFAIL

La llamada no se ha realizado satisfactoriamente.

RAZÓN (entero con signo de 10 dígitos) - salida

Código de motivo calificativo CMPCOD.

Si CMPCOD es CCOK:

RCNONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si CMPCOD es CCFAIL:

RC2219

(2219, X'8AB') Se ha vuelto a introducir la llamada « MQI » antes de que finalizara la llamada anterior.

RC2006

(2006, X'7D6') Longitud de los atributos de los caracteres no válida.

RC2007

(2007, X'7D7') Cadena de atributos de carácter no válida.

RC2009

(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

RC2018

(2018, X'7E2') El manejador de conexión no es válido.

RC2019

(2019, X'7E3') El manejador de objeto no es válido.

RC2020

(2020, X'7E4') El valor del atributo de cola inhibit-get o inhibit-put no es válido.

RC2021

(2021, X'7E5') Recuento de atributos enteros no válido.

RC2023

(2023, X'7E7') Matriz de atributos enteros no válida.

RC2040

(2040, X'7F8') La cola no está abierta para el conjunto.

RC2041

(2041, X'7F9') La definición de objeto ha cambiado desde que se ha abierto.

RC2101

(2101, X'835') El objeto se ha dañado.

RC2052

(2052, X'804') La cola se ha suprimido.

RC2058

(2058, X'80A') El nombre del gestor de colas no es válido o no se conoce.

RC2059

(2059, X'80B') El gestor de colas no está disponible para efectuar la conexión.

RC2162

(2162, X'872') El gestor de colas está concluyendo.

RC2102

(2102, X'836') No hay disponibles suficientes recursos del sistema.

RC2065

(2065, X'811') Recuento de selectores no válido.

RC2067

(2067, X'813') Selector de atributos no válido.

RC2066

(2066, X'812') Número de selectores demasiado grande.

RC2071

(2071, X'817') No hay suficiente almacenamiento disponible.

RC2075

(2075, X'81B') El valor del atributo de control de activación no es válido.

RC2076

(2076, X'81C') El valor del atributo trigger-depth no es válido.

RC2077

(2077, X'81D') El valor del atributo trigger-message-priority no es válido.

RC2078

(2078, X'81E') El valor del atributo de tipo trigger no es válido.

RC2195

(2195, X'893') Se ha producido un error inesperado.

Declaración RPG

```

C*..1.....2.....3.....4.....5.....6.....7..
C          CALLP      MQSET(HCONN : HOBJ : SELCNT :
C                      SELS(1) : IACNT : INTATR(1) :
C                      CALEN : CHRATR : CMPCOD :
C                      REASON)

```

La definición del prototipo para la llamada es:

```

D*..1.....2.....3.....4.....5.....6.....7..
DMQSET      PR          EXTPROC('MQSET')
D* Connection handle
D HCONN          10I 0 VALUE
D* Object handle
D HOBJ          10I 0 VALUE
D* Count of selectors
D SELCNT        10I 0 VALUE
D* Array of attribute selectors
D SELS          10I 0
D* Count of integer attributes
D IACNT          10I 0 VALUE
D* Array of integer attributes
D INTATR        10I 0
D* Length of character attributes buffer
D CALEN          10I 0 VALUE
D* Character attributes
D CHRATR          *   VALUE
D* Completion code
D CMPCOD        10I 0

```

MQSETMP (Establecer propiedad de manejador de mensajes) en IBM i

La llamada MQSETMP establece o modifica una propiedad de un manejador de mensajes.

- [“Sintaxis” en la página 1385](#)
- [“Notas de uso” en la página 1385](#)
- [“Parámetros” en la página 1387](#)
- [“Declaración RPG” en la página 1389](#)

Sintaxis

MQSETMP (*Hconn, Hmsg, SetPropOpts, Name, PropDesc, Type, ValueLength, Value, CompCode, Reason*)

Notas de uso

- Sólo puede utilizar esta llamada cuando el propio gestor de colas coordina la unidad de trabajo. Este puede ser:
 - Unidad de trabajo local, donde los cambios sólo afectan a los recursos de IBM MQ .
 - Unidad de trabajo global, donde los cambios pueden afectar a los recursos que pertenecen a otros gestores de recursos, así como a los recursos de IBM MQ .
- Para obtener más detalles sobre las unidades de trabajo locales y globales, consulte [“MQBEGIN \(Inicio de la unidad de trabajo\) en IBM i” en la página 1282](#).
- En entornos en los que el gestor de colas no coordina la unidad de trabajo, utilice la llamada de restitución adecuada en lugar de MQBACK. El entorno también puede dar soporte a una devolución implícita causada por la terminación anómala de la aplicación.
 - En z/OS, utilice las llamadas siguientes:
 - Los programas por lotes (incluidos los programas IMS batch DL/I) pueden utilizar la llamada MQBACK si la unidad de trabajo sólo afecta a los recursos de IBM MQ . Sin embargo, si la unidad de trabajo afecta tanto a los recursos de IBM MQ como a los recursos pertenecientes a otros gestores de recursos (por ejemplo, Db2), utilice la llamada SRRBACK proporcionada por el servicio de recursos recuperables (RRS) de z/OS . La llamada SRRBACK restituye los cambios en los recursos que pertenecen a los gestores de recursos que se han habilitado para la coordinación RRS.
 - Las aplicaciones CICS deben utilizar el mandato EXEC CICS SYNCPOINT ROLLBACK para restituir la unidad de trabajo. No utilice la llamada MQBACK para aplicaciones CICS .
 - Las aplicaciones IMS (que no sean programas DL/I por lotes) deben utilizar llamadas IMS como ROLB para restituir la unidad de trabajo. No utilice la llamada MQBACK para aplicaciones IMS (que no sean programas DL/I por lotes).
 - En IBM i, utilice esta llamada para las unidades de trabajo locales coordinadas por el gestor de colas. Esto significa que no debe existir una definición de compromiso a nivel de trabajo, es decir, el mandato STRCMTCTL con el parámetro **CMTSCOPE(*JOB)** no debe haberse emitido para el trabajo.
- Si una aplicación finaliza con cambios no confirmados en una unidad de trabajo, la disposición de dichos cambios depende de si la aplicación finaliza de forma normal o anómala. Consulte las notas de uso en [“MQDISC \(Desconectar gestor de colas\) en IBM i” en la página 1319](#) para obtener más detalles.
- Cuando una aplicación coloca u obtiene mensajes en grupos o segmentos de mensajes lógicos, el gestor de colas conserva información relacionada con el grupo de mensajes y el mensaje lógico para las últimas llamadas MQPUT y MQGET satisfactorias. Esta información está asociada con el descriptor de contexto de cola e incluye cosas como:

- Los valores de los campos *GroupId*, *MsgSeqNumber*, *Offset* y *MsgFlags* en MQMD.
- Indica si el mensaje forma parte de una unidad de trabajo.
- Para la llamada MQPUT: si el mensaje es persistente o no persistente.

El gestor de colas mantiene tres conjuntos de información de grupo y segmento, un conjunto para cada uno de los siguientes:

- La última llamada MQPUT satisfactoria (puede formar parte de una unidad de trabajo).
- La última llamada MQGET satisfactoria que ha eliminado un mensaje de la cola (esto puede formar parte de una unidad de trabajo).
- La última llamada MQGET satisfactoria que ha examinado un mensaje en la cola (no puede formar parte de una unidad de trabajo).

Si la aplicación coloca u obtiene los mensajes como parte de una unidad de trabajo y, a continuación, la aplicación decide restituir la unidad de trabajo, la información de grupo y segmento se restaura al valor que tenía anteriormente:

- La información asociada con la llamada MQPUT se restaura al valor que tenía antes de la primera llamada MQPUT satisfactoria para ese manejador de cola en la unidad de trabajo actual.
- La información asociada con la llamada MQGET se restaura al valor que tenía antes de la primera llamada MQGET satisfactoria para ese manejador de cola en la unidad de trabajo actual.

Las colas actualizadas por la aplicación después de que se iniciara la unidad de trabajo, pero fuera del ámbito de la unidad de trabajo, no tienen la información de grupo y segmento restaurada si se restituye la unidad de trabajo.

La restauración de la información de grupo y segmento a su valor anterior cuando se restituye una unidad de trabajo permite a la aplicación distribuir un grupo de mensajes grande o un mensaje lógico grande que consta de muchos segmentos entre varias unidades de trabajo, y reiniciar en el punto correcto del grupo de mensajes o mensaje lógico si falla una de las unidades de trabajo.

El uso de varias unidades de trabajo puede ser ventajoso si el gestor de colas local sólo tiene un almacenamiento de cola limitado. Sin embargo, la aplicación debe mantener suficiente información para poder reiniciar la colocación u obtención de mensajes en el punto correcto si se produce una anomalía del sistema.

Para obtener detalles sobre cómo reiniciar en el punto correcto después de una anomalía del sistema, consulte la opción PMLOGO descrita en [PMOPT \(entero con signo de 10 dígitos\)](#) y la opción GMLOGO descrita en [GMOPT \(entero con signo de 10 dígitos\)](#).

Las notas de uso restantes sólo se aplican cuando el gestor de colas coordina las unidades de trabajo:

- Una unidad de trabajo tiene el mismo ámbito que un descriptor de conexión. Todas las llamadas de IBM MQ que afectan a una unidad de trabajo determinada se deben realizar utilizando el mismo descriptor de conexión. Las llamadas emitidas utilizando un descriptor de conexión diferente (por ejemplo, las llamadas emitidas por otra aplicación) afectan a una unidad de trabajo diferente. Consulte [HCONN \(entero con signo de 10 dígitos\)-salida](#) para obtener información sobre el ámbito de los manejadores de conexión.
- Esta llamada sólo afecta a los mensajes que se han colocado o recuperado como parte de la unidad de trabajo actual.
- Una aplicación de larga ejecución que emite llamadas MQGET, MQPUT o MQPUT1 dentro de una unidad de trabajo, pero que nunca emite una llamada de confirmación o restitución, puede llenar las colas con mensajes que no están disponibles para otras aplicaciones. Para protegerse de esta posibilidad, el administrador debe establecer el atributo de gestor de colas **MaxUncommittedMsgs** en un valor lo suficientemente bajo como para evitar que las aplicaciones desbocadas llenen las colas, pero lo suficientemente alto como para permitir que las aplicaciones de mensajería esperadas funcionen correctamente.

Parámetros

La llamada MQSETMP tiene los parámetros siguientes:

HCONN (entero con signo de 10 dígitos)-entrada

Este manejador representa la conexión con el gestor de colas.

El valor debe coincidir con el descriptor de conexión que se ha utilizado para crear el descriptor de mensaje especificado en el parámetro **HMSG**.

Si el descriptor de mensaje se ha creado utilizando HCUNAS, se debe establecer una conexión válida en la hebra estableciendo una propiedad del descriptor de mensaje; de lo contrario, la llamada falla con el código de razón RC2009.

HMSG (entero con signo de 20 dígitos)-entrada

Este es el manejador de mensajes que se va a modificar. El valor ha sido devuelto por una llamada MQCRTMH anterior.

SETOPT (MQSMPO)-entrada

Controlar cómo se establecen las propiedades de mensaje.

Esta estructura permite a las aplicaciones especificar opciones que controlan cómo se establecen las propiedades de mensaje. La estructura es un parámetro de entrada en la llamada MQSETMP. Consulte [MQSMPO](#) para obtener más información.

PRNAME (MQCHARV)-entrada

Es el nombre de la propiedad que se va a establecer.

Consulte [Nombres de propiedad y Restricciones de nombre de propiedad](#) para obtener más información sobre el uso de nombres de propiedad.

PRPDSC (MQPD)-entrada/salida

Esta estructura se utiliza para definir los atributos de una propiedad, incluyendo:

- qué sucede si la propiedad no está soportada
- a qué contexto de mensaje pertenece la propiedad
- en qué mensajes se copia la propiedad a medida que fluye

Consulte [MQPD](#) para obtener más información sobre esta estructura.

TYPE (entero con signo de 10 dígitos)-entrada

El tipo de datos de la propiedad que se está definiendo. Puede ser uno de los siguientes:

TIPBOL

Un booleano. *ValueLength* debe ser 4.

TYPBST

Una serie de bytes. *ValueLength* debe ser cero o mayor.

TYPI8

Un entero con signo de 8 bits. *ValueLength* debe ser 1.

TYPI16

Un entero con signo de 16 bits. *ValueLength* debe ser 2.

TYPI32

Un entero con signo de 32 bits. *ValueLength* debe ser 4.

TYPI64

Un entero con signo de 64 bits. *ValueLength* debe ser 8.

TYPEF32

Un número de coma flotante de 32 bits. *ValueLength* debe ser 4.

TYPF64

Un número de coma flotante de 64 bits. *ValueLength* debe ser 8.

TIPSTR

Una serie de caracteres. *ValueLength* debe ser cero o mayor, o el valor especial VLNUL.

TYPNUL

La propiedad existe pero tiene un valor nulo. *ValueLength* debe ser cero.

VALLEN (entero con signo de 10 dígitos)-entrada

Longitud en bytes del valor de propiedad en el parámetro *Valor*.

Cero sólo es válido para valores nulos o para series o series de bytes. Cero indica que la propiedad existe pero que el valor no contiene caracteres ni bytes.

El valor debe ser mayor o igual que cero o el siguiente valor especial si el parámetro *Tipo* tiene establecido TYPSTR:

VLNUL

El valor está delimitado por el primer nulo encontrado en la serie. El valor nulo no se incluye como parte de la serie. Este valor no es válido si no se ha establecido también TYPSTR.

Nota: El carácter nulo utilizado para terminar una serie si se establece VLNUL es un nulo del juego de caracteres del valor.

VALUE (serie de bits de 1 byte x VALLEN)-entrada

El valor de la propiedad que se va a establecer. El almacenamiento intermedio debe estar alineado en un límite adecuado a la naturaleza de los datos del valor.

En el lenguaje de programación C, el parámetro se declara como puntero a void; la dirección de cualquier tipo de datos se puede especificar como parámetro.

Si *ValueLength* es cero, no se hace referencia a *Valor*. En este caso, la dirección de parámetro pasada por los programas escritos en C o System/390 assembler puede ser nula.

CMPCOD (entero con signo de 10 dígitos)-salida

El código de terminación; es uno de los siguientes:

CCOK

Realización satisfactoria.

CCFAIL

La llamada no se ha realizado satisfactoriamente.

REASON (entero con signo de 10 dígitos)-salida

El código de razón que califica *CMPCOD*.

Si *CMPCOD* es CCOK:

RCNONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CMPCOD* es CCWARN:

RC2421

(2421, X'0975 ') No se ha podido analizar una carpeta MQRFH2 que contiene propiedades.

Si *CMPCOD* es CCFail:

RC2204

(2204, X'089C') Adaptador no disponible.

RC2130

(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

RC2157

(2157, X'86D') Los ASID primario e inicial varían.

RC2004

(2004, X'07D4') Parámetro de valor no válido.

RC2005

(2005, X'07D5') Parámetro de longitud de valor no válido.

RC2219

(2219, X'08AB') MQI : se ha iniciado una llamada antes de que finalizara la anterior.

RC2460

(2460, X'099C') El puntero del gestor de mensajes no es válido.

RC2499

(2499, X'09C3') Manejador de mensaje ya en uso.

RC2046

(2046, X'07FE') Opciones no válidas o no coherentes.

RC2482

(2482, X'09B2') La estructura del descriptor de propiedad no es válida.

RC2442

(2442, X'098A') Nombre de propiedad no válido.

RC2473

(2473, X'09A9') Tipo de datos de propiedad no válido.

RC2472

(2472, X'09A8') Se ha encontrado un error de formato numérico en los datos de valor.

RC2463

(2463, X'099F') La estructura de opciones de propiedades de mensajes no es válida.

RC2111

(2111, X'083F') El identificador del juego de caracteres codificado del nombre de la propiedad no es válido.

RC2071

(2071, X'817') No hay suficiente almacenamiento disponible.

RC2195

(2195, X'893') Se ha producido un error inesperado.

Consulte [“Códigos de retorno para IBM i \(ILE RPG\)”](#) en la página 1447 para obtener más detalles.

Declaración RPG

```

C*.1.....2.....3.....4.....5.....6.....7..
C                                CALLP      MQSETMP(HCONN : HMSG : SETOPT :
                                           PRNAME : PRPDSC :
                                           TYPE : VALLEN : VALUE :
                                           CMPCOD : REASON)

```

La definición de prototipo para la llamada es:

```

DMQSETMP          PR                EXTPROC('MQSETMP')
D* Connection handle
D HCONN              10I 0 VALUE
D* Message handle
D HMSG              10I 0 VALUE
D* Options that control the action of MQSETMP
D SETOPT            20A
D* Property name
D PRNAME            32A
D* Property descriptor
D PRPDSC            24A
D* Property data type
D TYPE              10I 0 VALUE
D* Length of the Value area
D VALLEN            10I 0 VALUE
D* Property value
D VALUE              *    VALUE
D* Completion code

```

D CMPCOD	10I 0
D* Reason code qualifying CompCode	
D REASON	10I 0

MQSTAT (Recuperar información de estado) en IBM i

Utilice la llamada MQSTAT para recuperar información de estado. El tipo de información de estado devuelta viene determinado por el valor STYPE especificado en la llamada.

- [“Sintaxis” en la página 1390](#)
- [“Notas de uso” en la página 1390](#)
- [“Parámetros” en la página 1390](#)
- [“Declaración RPG” en la página 1391](#)

Sintaxis

MQSTAT (*HCONN*, *STYPE*, *STAT*, *CMPCOD*, *REASON*)

Notas de uso

1. Una llamada a MQSTAT que especifica un tipo de STATAPT devuelve información sobre operaciones MQPUT y MQPUT1 asíncronas anteriores. La estructura MQSTAT pasada en la llamada se completa con la primera información de error o aviso asíncrono registrada para dicha conexión. Si hay más errores o avisos a continuación del primero, normalmente no alteran estos valores. Sin embargo, si se produce un error con un código de terminación de CCWARN, en su lugar se devuelve un error posterior con un código de terminación de CCFAIL.
2. Si no se han producido errores desde que se estableció la conexión o desde la última llamada a MQSTAT, se devuelve un CMPCOD de CCOK y REASON de RCNONE.
3. Los recuentos del número de llamadas asíncronas que se han procesado bajo el descriptor de conexión se devuelven utilizando tres contadores: STSPSC, STSPWC y STSPFC. Estos contadores son incrementados por el gestor de colas cada vez que una operación asíncrona se procesa correctamente, tiene un aviso o falla (tenga en cuenta que, a efectos de contabilidad, una transferencia a una lista de distribución cuenta una vez por cola de destino en lugar de una vez por lista de distribución).
4. Una llamada satisfactoria a MQSTAT da como resultado que se restablezca cualquier información de error o recuento anterior.

Parámetros

La llamada MQSTAT tiene los parámetros siguientes:

Hconn (MQHCONN)-entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* ha sido devuelto por una llamada MQCONN o MQCONNEX anterior.

STYPE (entero con signo de 10 dígitos)-entrada

Tipo de información de estado que se está solicitando. El único valor válido es:

STATAPT

Devuelve información sobre operaciones de transferencia asíncronas anteriores.

STS (MQSTS)-entrada/salida

Estructura de información de estado. Consulte [“MQSTS \(Estructura de notificación de estado\) en IBM i” en la página 1258](#) para obtener los detalles.

CMPCOD (entero con signo de 10 dígitos)-salida

El código de terminación; es uno de los siguientes:

CCOK

Realización satisfactoria.

CCFAIL

La llamada no se ha realizado satisfactoriamente.

REASON (entero con signo de 10 dígitos)-salida

El código de razón que califica *CMPCOD*.

Si *CMPCOD* es CCOK:

RCNONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CMPCOD* es CCFAIL:

RC2374

(2374, X' 946 ') La salida de API ha fallado

RC2183

(2183, X'887') No se ha podido cargar la salida de la API.

RC2219

(2219, X'8AB') La llamada MQI ha entrado antes de completar la llamada anterior.

RC2009

(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

RC2203

(2203, X'89B') La conexión está concluyendo.

RC2018

(2018, X'7E2') El manejador de conexión no es válido.

RC2162

(2162, X'872 ') Se está deteniendo el gestor de colas

RC2102

(2102, X'836') No hay disponibles suficientes recursos del sistema.

RC2430

(2430, X'97E') Error con el tipo MQSTAT.

RC2071

(2071, X'817') No hay suficiente almacenamiento disponible.

RC2424

(2424, X' 978 ') Error con estructura MQSTS

RC2195

(2195, X'893') Se ha producido un error inesperado.

RC2298

(2298, X'8FA') La función solicitada no está disponible en el entorno actual.

Para obtener información detallada sobre estos códigos, consulte:

- [Mensajes y códigos de motivo](#)

Declaración RPG

```

C*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
C          CALLP      MQSTAT(HCONN : ETYPE : ERR :
C                                CMPCOD : REASON)

```

La definición de prototipo para la llamada es:

```

D.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
DMQSTAT      PR          EXTPROC('MQSTAT')
D* Connection handle
D HCONN          10I 0 VALUE

```

D* Status information type	
D STYPE	10I 0 VALUE
D* Status information	
D STATUS	296A
D* Completion code	
D CMPCOD	10I 0
D* Reason code qualifying CompCode	
D REASON	10I 0

IBM i MQSUB (Registrar suscripción) en IBM i

La llamada MQSUB registra la suscripción de aplicaciones a un tema determinado.

- “Sintaxis” en la [página 1392](#)
- “Notas de uso” en la [página 1392](#)
- “Parámetros” en la [página 1394](#)
- “Declaración RPG” en la [página 1397](#)

Sintaxis

MQSUB (*HCONN*, *SUBDSC*, *HOBJ*, *HSUB*, *CMPCOD*, *REASON*)

Notas de uso

- La suscripción se realiza a un tema, denominado utilizando el nombre abreviado de un objeto de tema predefinido, el nombre completo de la serie de tema o está formado por la concatenación de dos partes, tal como se describe en [Combinación de series de tema](#).
- El gestor de colas efectúa comprobaciones de seguridad cuando se emite una llamada MQSUB para verificar que el identificador de usuario con el que se está ejecutando la aplicación tiene el nivel de autorización adecuado antes de permitir el acceso. El objeto de tema adecuado se localiza mediante un nombre abreviado que se proporciona en la llamada o el objeto de nombre abreviado más cercano en la jerarquía de temas que se encuentra si se proporciona un nombre largo. Se realiza una comprobación de autorización sobre este objeto de tema para asegurarse de que se ha establecido la autorización para suscribirse y sobre la cola de destino para asegurarse de que se ha establecido la autorización para la salida. Si se utiliza la opción SDMAN, esto significa que se realiza una comprobación de autorización en el nombre de cola gestionada asociado a este objeto de tema, y si se proporciona una cola no gestionada, esto significa que se realiza una comprobación de autorización en la cola representada por el parámetro **HOBJ**.
- El **HOBJ** devuelto en la llamada MQSUB cuando se utiliza la opción SOMAN, se puede consultar para averiguar atributos como, por ejemplo, el umbral de restitución y el nombre de reposición en cola de restitución excesiva. También puede consultar el nombre de la cola gestionada, pero no debe intentar abrir directamente esta cola.
- Las suscripciones se pueden agrupar, lo que permite que se entregue una única publicación al grupo de suscripciones, incluso si más de una suscripción del grupo coincidía con la publicación. Las suscripciones se agrupan utilizando la opción SOGRP y para agrupar suscripciones deben:
 - utilizar la misma cola con nombre (que no utiliza la opción SOMAN) en el mismo gestor de colas representado por el parámetro **HOBJ** en la llamada MQSUB
 - compartir el mismo *SDCID*
 - ser del mismo *SDSL*

Estos atributos definen el conjunto de suscripciones que se tienen en cuenta para formar parte de un grupo y que también son los atributos que no se pueden alterar si se agrupa una suscripción. La modificación de los resultados de *SDSL* en RC2512, y la modificación de cualquiera de los demás (que se puede cambiar si una suscripción no está agrupada) da como resultado RC2515.

- Los campos en MQSD se completan al volver de una llamada MQSUB que utiliza la opción SORES. El MQSD devuelto se puede pasar directamente a una llamada MQSUB que utiliza la opción SOALT con

los cambios que necesite realizar en la suscripción aplicada al MQSD. Algunos campos tienen algunas consideraciones especiales, tal como se indica en la tabla.

Tabla 791. Salida MQSD de MQSUB	
Nombre de campo en MQSD	Consideraciones especiales
Opciones de acceso o creación	Ninguna de estas opciones se establece al volver de la llamada MQSUB. Si más adelante reutiliza el MQSD en una llamada MQSUB, la opción que necesita debe establecerse explícitamente.
Opciones de durabilidad, Opciones de destino, Opciones de registro & Opciones de comodín	Estas opciones se establecerán según corresponda
Opciones de publicación	Estas opciones se establecerán según corresponda, excepto para SONEWP que solo es aplicable a SOCRE.
Otras opciones	Estas opciones no se modifican en la devolución de una llamada MQSUB. Controlan cómo se emite la llamada de API y no se almacenan con la suscripción. Deben establecerse de la forma que sea necesaria en cualquier llamada MQSUB subsiguiente que vuelva a utilizar MQSD.
ObjectName	Este campo de sólo entrada no se modifica en la devolución de una llamada MQSUB.
ObjectString	Este campo de sólo entrada no se modifica en la devolución de una llamada MQSUB. El nombre de tema completo utilizado se devuelve en el campo SDR0 , si se proporciona un almacenamiento intermedio.
AlternateUserId y AlternateSecurityId	Estos campos de sólo entrada no se modifican en la devolución de una llamada MQSUB. Controlan cómo se emite la llamada de API y no se almacenan con la suscripción. Deben establecerse de la forma que sea necesaria en cualquier llamada MQSUB subsiguiente que vuelva a utilizar MQSD.
SubExpiry	Al volver de una llamada MQSUB utilizando la opción SORES, este campo se establecerá en la caducidad original de la suscripción y no en el tiempo de caducidad restante. Si, a continuación, reutiliza el MQSD en una llamada MQSUB utilizando la opción SOALT, restablecerá la caducidad de la suscripción para empezar a contar de nuevo.
SubName	Este campo es un campo de entrada en una llamada MQSUB y no se modifica en la salida.
SubUserData y SelectionString	Estos campos de longitud variable se devolverán en la salida de una llamada MQSUB utilizando la opción SORES, si se proporciona un almacenamiento intermedio, y también una longitud de almacenamiento intermedio positiva en VCHRP. Si no se proporciona ningún almacenamiento intermedio, sólo se devolverá la longitud en el campo VCHRL de MQCHARV. If el almacenamiento intermedio proporcionado es menor que el espacio necesario para devolver el campo, sólo se devuelven VCHRP bytes en el almacenamiento intermedio proporcionado. Si posteriormente reutiliza el MQSD en una llamada MQSUB utilizando la opción SOALT y no se proporciona un almacenamiento intermedio, pero se proporciona un VCHRL distinto de cero, si esa longitud coincide con la longitud existente del campo, no se realizará ninguna modificación en el campo.
SubCorrelId y PubAccountingToken	Si no utiliza SOSCID, el gestor de colas generará SDCID . Si no utiliza SOSETI, el gestor de colas generará SDACC . Estos campos se devolverán en MQSD desde una llamada MQSUB utilizando la opción SORES. Si los genera el gestor de colas, el valor generado se devolverá en una llamada MQSUB utilizando la opción SOCRE o SOALT.
PubPriority, SubLevel & PubApplIdentityData	Estos campos se devolverán en MQSD.
ResObjectString	Este campo de sólo salida se devolverá en MQSD si se proporciona un almacenamiento intermedio.

Parámetros

La llamada MQSUB tiene los parámetros siguientes:

HCONN (entero con signo de 10 dígitos)-entrada

Este manejador representa la conexión con el gestor de colas. El valor de *HCONN* ha sido devuelto por una llamada MQCONN o MQCONNX anterior.

SUBDSC (MQSD)-entrada/salida

Se trata de una estructura que identifica el objeto con uso que está registrando la aplicación. Consulte [“MQSD \(descriptor de suscripción\) en IBM i” en la página 1239](#) para obtener más información.

HOBJ (entero con signo de 10 dígitos)-entrada/salida

Este descriptor de contexto representa el acceso que se ha establecido para obtener los mensajes enviados a esta suscripción. Estos mensajes se pueden almacenar en una cola específica o se puede solicitar al gestor de colas que gestione su almacenamiento sin necesidad de una cola específica.

El manejador del objeto.

Si se va a utilizar una cola específica, debe estar asociada a la suscripción en el momento de la creación. Esto se puede hacer de dos maneras:

- Proporcionando este descriptor de contexto al llamar a MQSUB con la opción SDCRT. Si este descriptor de contexto se proporciona como parámetro de entrada en la llamada, debe ser un descriptor de contexto de objeto válido devuelto de una llamada MQOPEN anterior de una cola utilizando al menos una de las opciones OOINP*, OOOOUT (si es una cola remota, por ejemplo) u OOBW. Si este no es el caso, la llamada falla con RC2019. No puede ser un descriptor de contexto de objeto para una cola alias que se resuelve en un objeto de tema. Si es así, la llamada falla con RC2019
- Utilizando el mandato DEFINE SUB MQSC y proporcionando a ese mandato el nombre de un objeto de cola.

Si el gestor de colas va a gestionar el almacenamiento de los mensajes enviados a esta suscripción, debe indicarlo cuando se crea la suscripción, utilizando la opción SOMAN y estableciendo el valor del parámetro en HONONE. El gestor de colas devuelve el descriptor de contexto como un parámetro de salida en la llamada, y el descriptor de contexto que se devuelve se conoce como descriptor de contexto gestionado. Si se especifica HONONE y no se especifica también SOMAN, la llamada falla con RC2019.

Un descriptor de contexto gestionado devuelto por el gestor de colas se puede utilizar en una llamada MQGET o MQCB, con o sin opciones de examen, en una llamada MQINQ o en MQCLOSE. No se puede utilizar en MQPUT, MQSET o en una MQSUB posterior; el intento de hacerlo falla con RC2039 para MQPUT, RC2040 para MQSET o RC2038 para MQSUB.

Si se utiliza la opción SORES en el campo *OPTS* de la estructura MQSD para reanudar esta suscripción, el descriptor de contexto se puede devolver a la aplicación en este parámetro si se especifica HONONE. Puede utilizar esto tanto si la suscripción está utilizando un descriptor de contexto gestionado como si no. Puede ser útil para suscripciones creadas utilizando DEFINE SUB si desea el descriptor de contexto para la cola de suscripción definida en el mandato DEFINE SUB. En el caso de que se reanude una suscripción creada administrativamente, la cola se abre con OOINPQ y OOBW. Si se necesitan otras opciones, la aplicación debe abrir la cola de suscripción de forma explícita y proporcionar el descriptor de objeto en la llamada. Si hay un problema al abrir la cola, la llamada fallará con RC2522. Si se proporciona *HOBJ*, debe ser equivalente a *HOBJ* en la llamada MQSUB original. Esto significa que si se proporciona un descriptor de contexto de objeto devuelto desde una llamada MQOPEN, el descriptor de contexto debe estar en la misma cola que se ha utilizado anteriormente o la llamada falla con RC2019.

Si se está alterando esta suscripción, utilizando la opción SOALT en el campo *OPTS* de la estructura MQSD, se puede proporcionar un *HOBJ* diferente. Las publicaciones que se hayan entregado a la cola identificada anteriormente a través de este parámetro permanecen en dicha cola y es responsabilidad de la aplicación recuperar estos mensajes si el parámetro **HOBJ** representa ahora una cola diferente.

El uso de este parámetro con diversas opciones de suscripción se resume en la tabla siguiente:

Tabla 792. Utilización de <i>hobj</i> con varias opciones de suscripción		
Opciones	Hobj	Descripción
SOCRT + SOMAN	Ignorado en la salida	Crea una suscripción con almacenamiento de mensajes gestionado por el gestor de colas.
SOCRT	Descriptor de contexto de objeto válido	Crea una suscripción que facilita una cola específica como destino de los mensajes.
Llagas	HONONE	Reanuda una suscripción creada anteriormente (gestionada o no) y hace que el gestor de colas devuelva el descriptor de contexto de objeto para que lo utilice la aplicación.
Llagas	Válido, coincidente, descriptor de contexto de objeto	Reanuda una suscripción creada anteriormente que utiliza una cola específica como destino de los mensajes y utiliza un descriptor de contexto de objeto con opciones de apertura específicas.
SOALT + SOMAN	HONONE	Altera una suscripción existente que anteriormente utilizaba una cola específica, para que ahora se gestione.
SOALT	Descriptor de contexto de objeto válido	Altera una suscripción existente para utilizar una cola específica (ya sea de gestionada o de una cola específica diferente).

Tanto si se ha proporcionado como si se ha devuelto, *HOBJ* debe especificarse en las llamadas MQGET posteriores que necesite para recibir las publicaciones.

El descriptor de contexto *HOBJ* deja de ser válido cuando se emite la llamada MQCLOSE en él, o cuando termina la unidad de proceso que define el ámbito del descriptor de contexto. El ámbito del descriptor de contexto de objeto devuelto es el mismo que el del descriptor de conexión especificado en la llamada. Consulte [HCONN](#) para obtener información sobre el ámbito de descriptor de contexto. Un MQCLOSE del descriptor de contexto *HOBJ* no tiene ningún efecto en el descriptor de contexto *HSUB*.

HSUB (entero con signo de 10 dígitos)-salida

Este descriptor de contexto representa la suscripción que se ha realizado. Se puede utilizar para otras dos operaciones:

- Se puede utilizar en una llamada MQSUBRQ posterior para solicitar que se envíen publicaciones cuando se haya utilizado la opción SOPUBR al realizar la suscripción.
- Se puede utilizar en una llamada MQCLOSE subsiguiente para eliminar la suscripción que se ha realizado. El descriptor de contexto *HSUB* deja de ser válido cuando se emite la llamada MQCLOSE o cuando termina la unidad de proceso que define el ámbito del descriptor de contexto. El ámbito del descriptor de contexto de objeto devuelto es el mismo que el del descriptor de conexión especificado en la llamada. Un MQCLOSE del descriptor de contexto *HSUB* no tiene ningún efecto en el descriptor de contexto *HOBJ*.

Este descriptor de contexto no se puede pasar a una llamada MQGET o MQCB. Debe utilizar el parámetro **HOBJ**. Si se pasa este descriptor de contexto a cualquier otra llamada IBM MQ, se genera RC2019.

CMPCOD (entero con signo de 10 dígitos)-salida

El código de terminación; es uno de los siguientes:

CCOK

Realización satisfactoria

CCWARN

Aviso (terminación parcial)

CCFAIL

Llamada fallida

REASON (entero con signo de 10 dígitos)-salida

El código de razón que califica *CMPCOD*.

Si *CMPCOD* es CCOK:

RCNONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CMPCOD* es CCFAIL:

RC2019

(2019 X'07E3') Manipulador de objeto no válido

RC2046

(2046 X'07FE') Opciones no válidas o no coherentes

RC2085

(2085 X'0825 ') No se puede encontrar el objeto identificado

RC2161

(2161 X'0871 ') Desactivación temporal del gestor de colas

RC2298

(2298 X'08FA') Función no soportada.

RC2424

(2424 X'0978 ') Descriptor de suscripción (MQSD) no válido

RC2425

(2441 X' 979 ') Serie de tema no válida

RC2428

(2428 X'097C') El nombre de suscripción especificado no coincide con las suscripciones existentes

RC2429

(2429 X'097D') El nombre de la suscripción existe y está siendo utilizado por otra aplicación

RC2431

(2431 X'097F') El campo SubUserData no es válido

RC2432

(2432 X'0980 ') La suscripción existe

RC2434

(2434 X'0982 ') El nombre de suscripción coincide con la suscripción existente

RC2440

(2440 X'0988 ') El campo SubName no es válido

RC2441

(2441 X'0989 ') El campo de serie de objeto no es válido

RC2435

(2435 X'0983 ') El atributo no se puede cambiar utilizando SDALT, o se ha creado la suscripción con SDIMM.

RC2436

(2436 X'0984 ') Opción SODUR no válida

RC2459

(2459, X'99B') Error de sintaxis en la cadena de selección.

RC2503

(2503 X'09C7') Las llamadas MQSUB están actualmente inhibidas para los temas suscritos.

RC2519

(2519, X'9D7') La cadena de selección no es la especificada en la descripción de cómo utilizar una estructura MQCHARV.

RC2551

(2551, X'9F7') La cadena de selección especificada no está disponible.

Declaración RPG

```
C*.1.....2.....3.....4.....5.....6.....7..  
C                                CALLP      MQSUB(HCONN : SUBDSC : HOBJ :  
C                                HSUB : CMPCOD : REASON)
```

La definición de prototipo para la llamada es:

```
D*.1.....2.....3.....4.....5.....6.....7..  
DMQSUB          PR          EXTPROC('MQSUB')  
D* Connection handle  
D HCONN          10I 0 VALUE  
D* Subscription descriptor  
D SUBDSC          400A  
D* Object handle for queue  
D HOBJ          10I 0  
D* Subscription object handle  
D HSUB          10I 0  
D* Completion code  
D CMPCOD          10I 0  
D* Reason code qualifying CompCode  
D REASON          10I 0
```

IBM i MQSUBRQ (Solicitud de suscripción) en IBM i

La llamada MQSUBRQ realiza una solicitud sobre una suscripción.

- “Sintaxis” en la [página 1397](#)
- “Notas de uso” en la [página 1397](#)
- “Parámetros” en la [página 1398](#)
- “Declaración RPG” en la [página 1399](#)

Sintaxis

MQSUBRQ (*HCONN*, *HSUB*, *ACTION*, *SUBROPT*, *CMPCOD*, *REASON*)

Notas de uso

Las siguientes notas de uso se aplican al uso de SRAPUB:

1. Si este verbo se completa correctamente, las publicaciones retenidas que coinciden con la suscripción especificada se han enviado a la suscripción y se pueden recibir utilizando MQGET o MQCB con el HOBJ devuelto en el verbo MQSUB original que creó la suscripción.
2. Si el tema suscrito por el verbo MQSUB original que creó la suscripción contenía un comodín, es posible que se envíen más de una publicación retenida. El número de publicaciones enviadas como resultado de esta convocatoria se registra en el *SRNMP* campo de la estructura SBROPT.
3. Si este verbo se completa con un código de motivo « RC2437 », significa que no hay publicaciones retenidas actualmente para el tema especificado.
4. Si este verbo se completa con un código de motivo « RC2525 » (no se ha podido entregar) o « RC2526 » (no se ha podido entregar), significa que actualmente hay publicaciones retenidas para el tema especificado, pero se ha producido un error que ha impedido su entrega.
5. La aplicación debe tener una suscripción vigente al tema antes de poder realizar esta llamada. Si la suscripción se realizó en una instancia anterior de la aplicación y no hay disponible un identificador válido para la suscripción, la aplicación debe llamar primero a MQSUB con la opción SORES para obtener un identificador que se utilizará en esta llamada.

6. Las publicaciones se envían al destino registrado para su uso con la suscripción actual de esta aplicación. Si las publicaciones deben enviarse a otra dirección, primero hay que modificar la suscripción utilizando la llamada MQSUB con la opción SOALT.

Parámetros

La llamada MQSUBRQ tiene los siguientes parámetros:

HCONN (entero con signo de 10 dígitos) - entrada

Este manejador representa la conexión con el gestor de colas. El valor de *HCONN* fue devuelto por una llamada anterior a MQCONN o MQCONNX.

En z/OS para CICS aplicaciones, se puede omitir la llamada MQCONN y especificar el siguiente valor para *HCONN*:

HCDEFH

Manejador de conexión predeterminado.

HSUB (entero con signo de 10 dígitos) - entrada

Este identificador representa la suscripción para la que se va a solicitar una actualización. El valor de *HSUB* se devolvió desde una llamada MQSUB anterior.

ACCIÓN (entero con signo de 10 dígitos) - entrada

Este parámetro controla la acción concreta que se solicita en la suscripción. Se debe especificar uno (y solo uno) de los siguientes:

SRAPUB

Esta acción solicita que se envíe una publicación actualizada sobre el tema especificado. Esto se utiliza normalmente si el suscriptor especificó la opción SOPUBR en la llamada MQSUB cuando realizó la suscripción. Si el gestor de colas tiene una publicación retenida para el tema, esta se envía al suscriptor. Si no es así, la llamada falla. Si se envía una solicitud a una publicación que se ha retenido, esto se indica mediante la propiedad del mensaje « MQIsRetained » (Solicitud retenida) de dicha publicación.

Dado que el tema de la suscripción existente representado por el **HSUB** parámetro puede contener comodines, el suscriptor podría recibir varias publicaciones retenidas.

SBROPT (MQSRO) - entrada/salida

Estas opciones controlan la acción de MQSUBRQ. Consulte [“MQSRO - Opciones de solicitud de suscripción”](#) en la [página 582](#) para obtener más detalles.

CMPCOD (entero con signo de 10 dígitos) - salida

El código de terminación; es uno de los siguientes:

CCOK

Realización satisfactoria

CCWARN

Aviso (terminación parcial)

CCFAIL

Llamada fallida

Razón (entero con signo de 10 dígitos) - salida

El código de motivo que califica *CMPCOD*.

Si *CMPCOD* es CCOK:

RCNONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CMPCOD* es CCFail:

RC2298

2298 (X'08FA') La función solicitada no está disponible en el entorno actual.

RC2437

2437 (X'0985') Actualmente no hay publicaciones conservadas almacenadas para este tema.

RC2046

2046 (X'07FE') El parámetro o campo Options contiene opciones que no son válidas, o una combinación de opciones que no es válida.

RC2161

2161 (X'0871') El gestor de colas se está deteniendo

RC2438

2438 (X'0986') En la llamada MQSUBRQ, las opciones de solicitud de suscripción MQSRO no son válidas.

Declaración RPG

```
C*..1.....2.....3.....4.....5.....6.....7..
C          CALLP      MQSUBRQ(HCONN : HSUB : ACTION :
C                      SBROPT : CMPCOD : REASON)
```

La definición del prototipo para la llamada es:

```
D*..1.....2.....3.....4.....5.....6.....7..
DMQSUBRQ      PR          EXTPROC('MQSUBRQ')
D* Connection handle
D HCONN              10I 0 VALUE
D* Subscription handle
D HSUB              10I 0 VALUE
D* Action requested on the subscription
D ACTION            10I 0 VALUE
D* Subscription Request Options
D SBROPT              16A
D* Completion code
D CMPCOD              10I 0
D* Reason code qualifying CompCode
D REASON              10I 0
```

Atributos de objetos en IBM i

Esta colección de temas lista sólo los objetos IBM MQ que pueden ser objeto de una llamada a función MQINQ, y proporciona detalles de los atributos que se pueden consultar y los selectores que se van a utilizar.

Atributos para colas

Utilice esta información para obtener información sobre los distintos tipos de definiciones de cola y los atributos soportados por cada uno.

Tipos de cola: el gestor de colas da soporte a los siguientes tipos de definición de cola:

Cola local

Se trata de una cola física que almacena mensajes. La cola existe en el gestor de colas local.

Las aplicaciones conectadas al gestor de colas local pueden colocar mensajes y eliminar mensajes de colas de este tipo. El valor del atributo de cola **QType** es QTLOC.

Cola compartida

Se trata de una cola física que almacena mensajes. La cola existe en un repositorio compartido al que pueden acceder todos los gestores de colas que pertenecen al grupo de compartición de colas propietario del repositorio compartido.

Las aplicaciones conectadas a cualquier gestor de colas del grupo de compartición de colas pueden colocar mensajes y eliminar mensajes de colas de este tipo. Estas colas son efectivamente las mismas que las colas locales. El valor del atributo de cola **QType** es QTLOC.

- Las colas compartidas solo están soportadas en z/OS.

Cola de clúster

Se trata de una cola física que almacena mensajes. La cola existe en el gestor de colas local o en uno o varios de los gestores de colas que pertenecen al mismo clúster que el gestor de colas local.

Las aplicaciones conectadas al gestor de colas local pueden colocar mensajes en colas de este tipo, independientemente de la ubicación de la cola. Si existe una instancia de la cola en el gestor de colas local, la cola se comporta de la misma forma que una cola local, y las aplicaciones conectadas al gestor de colas local pueden eliminar mensajes de la cola. El valor del atributo de cola **QType** es QTCLUS.

Cola alias

Esto no es una cola física, es un nombre alternativo para una cola local. El nombre de la cola local en el que se resuelve el alias forma parte de la definición de la cola alias.

Las aplicaciones conectadas al gestor de colas local pueden colocar mensajes y eliminarlos de las colas alias: los mensajes se colocan y se eliminan de la cola local en la que se resuelve el alias. El valor del atributo de cola **QType** es QTALS.

Cola remota

No es una cola física, es la definición local de una cola que existe en un gestor de colas remoto. La definición local de la cola remota contiene información que indica al gestor de colas local cómo direccionar los mensajes al gestor de colas remoto.

Las aplicaciones conectadas al gestor de colas local pueden colocar mensajes en colas remotas: los mensajes se colocan en la cola de transmisión local que se utiliza para direccionar los mensajes al gestor de colas remoto. Las aplicaciones no pueden eliminar mensajes de colas remotas. El valor del atributo de cola **QType** es QTREM.

También se puede utilizar una definición de cola remota para:

- Alias de cola de respuestas

En este caso, el nombre de la definición es el nombre de una cola de respuestas. Para obtener más información, consulte [Definiciones de alias de cola de respuesta](#).

- Alias de gestor de colas

En este caso, el nombre de la definición es un alias para un gestor de colas, no el nombre de una cola. Para obtener más información, consulte [Definiciones de alias de gestor de colas](#).

Cola modelo

No es una cola física, es un conjunto de atributos de cola a partir de los cuales se puede crear una cola local.

Los mensajes no se pueden almacenar en colas de este tipo.

Algunos atributos de cola se aplican a todos los tipos de cola; otros atributos de cola se aplican sólo a determinados tipos de cola. Los tipos de cola a los que se aplica un atributo se indican con una "X" en [Tabla 793 en la página 1401](#) y en las tablas siguientes.

La [Tabla 793 en la página 1401](#) resume los atributos específicos de las colas. Los atributos se describen en orden alfabético.

Los nombres de los atributos mostrados en la tabla son los nombres utilizados con las llamadas MQINQ y MQSET. Cuando se utilizan comandos MQSC para definir, modificar o mostrar atributos, se utilizan nombres abreviados alternativos; consulte [Comandos MQSC](#) para obtener más detalles.

En la siguiente tabla, las columnas se aplican del siguiente modo:

- La columna para colas locales también se aplica a colas compartidas.
- La columna para colas modelo indica qué atributos hereda la cola local creada a partir de la cola modelo.
- La columna para colas de clúster indica los atributos que se pueden consultar cuando la cola de clúster se abre solo para consulta, o para consulta y salida. Si la cola de clúster se abre para realizar consultas

más una o más entradas, examinar o establecer, en su lugar se aplica la columna correspondiente a las colas locales.

Tabla 793. Atributos para colas						
Atributo	Descripción	Local	Modelo	Alias	Remoto	Clúster
AlterationDate	Fecha en que se modificó por última vez la definición	X		X	X	
AlterationTime	Hora a la que se modificó por última vez la definición	X		X	X	
BackoutRequeueQName	Nombre de cola de reposición en cola de restitución excesivo	X	X			
BackoutThreshold	Umbral de restituciones	X	X			
BaseQName	Nombre de cola al que se resuelve el alias			X		
ClusterChannelName	Nombre de canal de clúster emisor	✓	✓			
ClusterName	Nombre del clúster al que pertenece la cola	X		X	X	
ClusterNameList	Nombre del objeto de lista de nombres que contiene nombres de clústeres a los que pertenece la cola	X		X	X	
CreationDate	Fecha de creación de la cola	X				
CreationTime	Hora en que se creó la cola	X				
CurrentQDepth	Profundidad de cola actual	X				
DefBind	Enlace predeterminado	X		X	X	X
DefinitionType	Tipo de definición de cola	X	X			
DefInputOpenOption	Opción abierta de entrada predeterminada	X	X			
DefPersistence	Persistencia de mensajes predeterminada	X	X	X	X	X
DefPriority	Prioridad de mensajes predeterminada	X	X	X	X	X

Tabla 793. Atributos para colas (continuación)						
Atributo	Descripción	Local	Modelo	Alias	Remoto	Clúster
DistLists	Soporte de lista de distribución	X	X			
HardenGetBackout	Si se debe mantener un recuento de restituciones preciso	X	X			
InhibitGet	Controla si se permiten operaciones get para la cola	X	X	X		
InhibitPut	Controla si se permiten operaciones de colocación para la cola	X	X	X	X	X
InitiationQName	Nombre de cola de inicio	X	X			
MaxMsgLength	Longitud máxima de mensaje en bytes	X	X			
MaxQDepth	Profundidad máxima de la cola	X	X			
MediaLog	Identidad de la extensión de registro más antigua (o del receptor de diario más antiguo en IBM i) necesario para la recuperación de medios de una cola especificada	✓	✓			
MsgDeliverySequence	Secuencia de entrega de mensajes	X	X			
OpenInputCount	Número de aperturas para entrada	X				
OpenOutputCount	Número de aperturas para salida	X				
ProcessName	Nombre de proceso	X	X			
QDepthHighEvent	Controla si se generan sucesos de profundidad de cola alta	X	X			
QDepthHighLimit	Límite alto para profundidad de cola	X	X			
QDepthLowEvent	Controla si se generan sucesos de profundidad de cola baja	X	X			

Tabla 793. Atributos para colas (continuación)						
Atributo	Descripción	Local	Modelo	Alias	Remoto	Clúster
QDepthLowLimit	Límite bajo para profundidad de cola	X	X			
QDepthMaxEvent	Controla si se generan sucesos de cola llena	X	X			
QDesc	Descripción de la cola	X	X	X	X	X
QName	Nombre de cola	X		X	X	X
QServiceInterval	Destino para intervalo de servicio de cola	X	X			
QServiceIntervalEvent	Controla si se generan sucesos de intervalo de servicio alto o de intervalo de servicio correcto	X	X			
QType	Tipo de cola	X		X	X	X
RemoteQMgrName	Nombre del gestor de colas remoto				X	
RemoteQName	Nombre de cola remota				X	
RetentionInterval	Intervalo de retención	X	X			
Ámbito	Controla si una entrada para la cola también existe en un directorio de célula	X		X	X	
Posibilidad de compartición	Compatibilidad de cola	X	X			
TriggerControl	Activar control	X	X			
TriggerData	Datos desencadenantes	X	X			
TriggerDepth	Profundidad de desencadenante	X	X			
TriggerMsgPriority	Prioridad de mensaje de umbral para desencadenantes	X	X			
TriggerType	Tipo de desencadenante	X	X			
Uso	Uso de cola	X	X			
XmitQName	Nombre de cola de transmisión				X	

 ***AlterationDate (serie de caracteres de 12 bytes) en IBM i***
Fecha en la que se cambió por última vez la definición.

Tabla 794. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X		X	X	

Es la fecha en la que se modificó por última vez la definición. El formato de la fecha es YYYY-MM-DD, relleno con dos espacios en blanco al final para que la longitud sea de 12 bytes (por ejemplo, 1992-09-23 , donde representa dos caracteres en blanco).

Los valores de determinados atributos (por ejemplo, *CurrentQDepth*) cambian a medida que opera el gestor de colas. Los cambios en estos atributos no afectan a *AlterationDate*.

Para determinar el valor de este atributo, utilice el selector CAALTD con la llamada MQINQ. La longitud de este atributo viene dada por LNDATE.

IBM i ***AlterationTime (serie de caracteres de 8 bytes) en IBM i***

Hora a la que se modificó por última vez la definición.

Tabla 795. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X		X	X	

Es la hora en que se modificó por última vez la definición. El formato de la hora es HH.MM.SS utilizando el reloj de 24 horas, con un cero inicial si la hora es inferior a 10 (por ejemplo, 09.10.20). La hora es local.

Los valores de determinados atributos (por ejemplo, *CurrentQDepth*) cambian a medida que opera el gestor de colas. Los cambios en estos atributos no afectan a *AlterationTime*.

Para determinar el valor de este atributo, utilice el selector CAALTT con la llamada MQINQ. La longitud de este atributo la proporciona LNTIME.

IBM i ***BackoutRequeueQName (cadena de caracteres de 48 bytes) en IBM i***

Nombre de cola de reposición en cola de restitución excesivo.

Tabla 796. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X			

Las aplicaciones que se ejecutan dentro de WebSphere Application Server y las que utilizan IBM MQ Application Server Facilities utilizan este atributo para determinar dónde deben ir los mensajes que se han restituido. Para todas las demás aplicaciones, aparte de permitir que se consulte su valor, el gestor de colas no realiza ninguna acción basada en el valor del atributo.

Para determinar el valor de este atributo, utilice el selector CABRQN con la llamada MQINQ. La longitud de este atributo la proporciona LNQN.

IBM i ***BackoutThreshold (entero con signo de 10 dígitos) en IBM i***

El umbral de restituciones.

Tabla 797. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remoto	Clúster
X	X			

Las aplicaciones que se ejecutan dentro de WebSphere Application Server y las que utilizan IBM MQ Application Server Facilities utilizan este atributo para determinar si se debe restituir un mensaje. Para todas las demás aplicaciones, aparte de permitir que se consulte su valor, el gestor de colas no realiza ninguna acción basada en el valor del atributo.

Para determinar el valor de este atributo, utilice el selector IABTHR con la llamada MQINQ.

IBM i **BaseQName (serie de caracteres de 48 bytes) en IBM i**

Es el nombre de la cola a la que resuelve el alias.

Tabla 798. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remoto	Clúster
		X		

Es el nombre de una cola definida en el gestor de colas local. (Para obtener más información sobre los nombres de cola, consulte la descripción del campo *ODON* en MQOD. La cola es de uno de los tipos siguientes:

QTLOC

Cola local.

QTREM

Definición local de una cola remota.

QTCLUS

Cola de clúster.

Para determinar el valor de este atributo, utilice el selector CABASQ con la llamada MQINQ. La longitud de este atributo la proporciona LNQN.

IBM i **BaseType (estructura de parámetros enteros) en IBM i**

El tipo de objeto en el que se resuelve el alias.

Tabla 799. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remoto	Clúster
		X		

Este atributo puede tener uno de los siguientes valores:

OTQ

El tipo de objeto base es una cola

OTTOP

El tipo de objeto base es un tema

IBM i **CFStrucName (serie de caracteres de 12 bytes) en IBM i**

Nombre de estructura de recurso de acoplamiento.

Tabla 800. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X			

Es el nombre de la estructura del recurso de acoplamiento donde se almacenan los mensajes de la cola. El primer carácter del nombre está en el rango de A a Z, y los caracteres restantes están en el rango de A a Z, de 0 a 9 o en blanco.

El nombre completo de la estructura en el recurso de acoplamiento se obtiene sufriendo el valor del atributo de gestor de colas **QSGName** con el valor del atributo de cola **CFStrucName**.

Este atributo sólo se aplica a las colas compartidas; se ignora si *QSGDisp* no tiene el valor QSGDSH.

Para determinar el valor de este atributo, utilice el selector CACFSN con la llamada MQINQ. La longitud de este atributo la proporciona LNCFSN.

 Este atributo sólo está soportado en z/OS.

ClusterChannelName (cadena de caracteres de 20 bytes)

ClusterChannelName es el nombre genérico de los canales de envío del clúster que utilizan esta cola como cola de transmisión. El atributo especifica los canales de clúster emisor han enviado mensajes a un canal de clúster receptor desde esta cola de transmisión de clúster.

Tabla 801. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X			

La configuración del gestor de colas predeterminado es para todos los canales de clúster emisor para enviar mensajes desde una sola cola de transmisión, SYSTEM.CLUSTER.TRANSMIT.QUEUE. La configuración predeterminada se puede cambiar modificando el atributo del gestor de colas, **DefClusterXmitQueueType**. El valor predeterminado del atributo es SCTQ. Puede cambiar el valor a CHANNEL. Si establece el atributo **DefClusterXmitQueueType** en CHANNEL, cada canal de clúster emisor utiliza de forma predeterminada una cola de transmisión de clúster específica, SYSTEM.CLUSTER.TRANSMIT.ChannelName.

También puede establecer el atributo de cola de transmisión ClusterChannelName en un canal de clúster emisor manualmente. Los mensajes destinados al gestor de colas conectado por el canal de clúster emisor se almacenan en la cola de transmisión que identifica el canal de clúster emisor. No se almacenan en la cola de transmisión de clúster predeterminada. Si establece el atributo ClusterChannelName en blancos, el canal conmuta a la cola de transmisión de clúster predeterminada cuando se reinicia el canal. La cola predeterminada es SYSTEM.CLUSTER.TRANSMIT.ChannelName o SYSTEM.CLUSTER.TRANSMIT.QUEUE, en función del valor del atributo DefClusterXmitQueueType del gestor de colas.

Al especificar asteriscos, "*", en **ClusterChannelName**, puede asociar una cola de transmisión con un conjunto de canales de clúster emisor. Los asteriscos pueden estar al principio, al final o en cualquier posición intermedia de la serie de nombre de canal. **ClusterChannelName** está limitado a una longitud de 20 caracteres: MQ_CHANNEL_NAME_LENGTH.

ClusterName (serie de caracteres de 48 bytes) en IBM i

Nombre del clúster al que pertenece la cola.

Tabla 802. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remoto	Clúster
X		X	X	

Es el nombre del clúster al que pertenece la cola. Si la cola pertenece a más de un clúster, *ClusterNameList* especifica el nombre de un objeto de lista de nombres que identifica los clústeres y *ClusterName* está en blanco. Al menos uno de *ClusterName* y *ClusterNameList* debe estar en blanco.

Para determinar el valor de este atributo, utilice el selector CACLN con la llamada MQINQ. La longitud de este atributo la proporciona LNCLUN.

IBM i **ClusterNameList (serie de caracteres de 48 bytes) en IBM i**

Nombre del objeto de lista de nombres que contiene nombres de clústeres a los que pertenece la cola.

Tabla 803. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remoto	Clúster
X		X	X	

Es el nombre de un objeto de lista de nombres que contiene los nombres de los clústeres a los que pertenece esta cola. Si la cola sólo pertenece a un clúster, el objeto de lista de nombres sólo contiene un nombre. De forma alternativa, se puede utilizar *ClusterName* para especificar el nombre del clúster, en cuyo caso *ClusterNameList* está en blanco. Al menos uno de *ClusterName* y *ClusterNameList* debe estar en blanco.

Para determinar el valor de este atributo, utilice el selector CACLNL con la llamada MQINQ. La longitud de este atributo la proporciona LNNLN.

IBM i **CreationDate (serie de caracteres de 12 bytes) en IBM i**

Fecha de creación de la cola.

Tabla 804. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remoto	Clúster
X				

Es la fecha en la que se creó la cola. El formato de la fecha es YYYY-MM-DD, rellenado con dos espacios en blanco al final para que la longitud sea de 12 bytes (por ejemplo, 1992-09-23--), donde -- representa dos caracteres en blanco).

- En IBM i, la fecha de creación de una cola puede diferir de la de la entidad del sistema operativo subyacente (archivo o espacio de usuario) que representa la cola.

Para determinar el valor de este atributo, utilice el selector CACRTD con la llamada MQINQ. La longitud de este atributo la proporciona LNCRTD.

IBM i **CreationTime (serie de caracteres de 8 bytes) en IBM i**

Hora a la que se ha creado la cola.

Tabla 805. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remoto	Clúster
X				

Es la hora en que se ha creado la cola. El formato de la hora es HH.MM.SS utilizando el reloj de 24 horas, con un cero inicial si la hora es inferior a 10 (por ejemplo, 09.10.20). La hora es local.

- En IBM i, la hora de creación de una cola puede diferir de la de la entidad del sistema operativo subyacente (archivo o espacio de usuario) que representa la cola.

Para determinar el valor de este atributo, utilice el selector CACRTT con la llamada MQINQ. La longitud de este atributo la proporciona LNCRTT.

IBM i **CurrentQDepth** (entero con signo de 10 dígitos) en IBM i

Profundidad de cola actual.

Tabla 806. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remoto	Clúster
X				

Es el número de mensajes que hay en la cola actualmente. Se incrementa durante una llamada MQPUT y durante la restitución de una llamada MQGET. Se reduce durante una llamada MQGET no examinada y durante la restitución de una llamada MQPUT. El efecto de esto es que el recuento incluye los mensajes que se han colocado en la cola dentro de una unidad de trabajo, pero que todavía no se han confirmado, aunque no sean elegibles para ser recuperados por la llamada MQGET. De forma similar, excluye los mensajes que se han recuperado dentro de una unidad de trabajo utilizando la llamada MQGET, pero que todavía no se han confirmado.

El recuento también incluye los mensajes que han pasado su hora de caducidad pero que todavía no se han descartado, aunque estos mensajes no son elegibles para ser recuperados. Consulte el campo *MDEXP* descrito en “MQMD (descriptor de mensaje) en IBM i” en la página 1131.

El proceso de unidad de trabajo y la segmentación de mensajes pueden hacer que *CurrentQDepth* supere *MaxQDepth*. Sin embargo, esto no afecta a la capacidad de recuperación de los mensajes- todos los mensajes de la cola se pueden recuperar utilizando la llamada MQGET de la forma normal.

El valor de este atributo fluctúa a medida que opera el gestor de colas.

Para determinar el valor de este atributo, utilice el selector IACDEP con la llamada MQINQ.

IBM i **DefBind** (entero con signo de 10 dígitos) en IBM i

Enlace predeterminado.

Tabla 807. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remoto	Clúster
X		X	X	X

Este atributo es el enlace predeterminado que se utiliza cuando se especifica OOBNDQ en la llamada MQOPEN y la cola es una cola de clúster. DefBind puede tener uno de los valores siguientes:

BNDOPN

Enlace arreglado por la llamada MQOPEN.

BNDNO

Enlace no arreglado.

BNDGRP

El enlace no se arregla mediante la llamada MQOPEN, sino que se arregla en MQPUT para todos los mensajes de un grupo lógico.

Para determinar el valor de este atributo, utilice el selector IADBND con la llamada MQINQ.

IBM i **DefinitionType** (entero con signo de 10 dígitos) en IBM i

El tipo de definición de la cola.

Tabla 808. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X			

Indica cómo se ha definido la cola. El valor puede ser uno de los siguientes:

QDPRE

Cola permanente predefinida.

La cola es una cola permanente creada por el administrador del sistema; sólo el administrador del sistema puede suprimirla.

Las colas predefinidas se crean utilizando el mandato MQSC de DEFINE y solo se pueden suprimir utilizando el mandato MQSC de DELETE . Las colas predefinidas no se pueden crear a partir de colas modelo.

Los mandatos pueden ser emitidos por un operador o por un usuario autorizado que envía un mensaje de mandato a la cola de entrada de mandatos (consulte el atributo **CommandInputQName** descrito en [“Atributos del gestor de colas en IBM i” en la página 1432](#)).

QDPERM

Cola permanente definida dinámicamente.

La cola es una cola permanente creada por una aplicación que emite una llamada MQOPEN con el nombre de una cola modelo especificada en el descriptor de objeto MQOD. La definición de cola modelo tenía el valor QDPERM para el atributo **DefinitionType** .

Este tipo de cola se puede suprimir utilizando la llamada MQCLOSE. Consulte [“MQCLOSE \(Cerrar objeto\) en IBM i” en la página 1296](#) para obtener más detalles.

El valor del atributo **QSGDisp** para una cola dinámica permanente es QSGDQM.

QDTEMP

Cola temporal definida dinámicamente.

La cola es una cola temporal creada por una aplicación que emite una llamada MQOPEN con el nombre de una cola modelo especificada en el descriptor de objeto MQOD. La definición de cola modelo tenía el valor QDTEMP para el atributo **DefinitionType** .

Este tipo de cola se suprime automáticamente mediante la llamada MQCLOSE cuando la aplicación que la ha creado la cierra.

El valor del atributo **QSGDisp** para una cola dinámica temporal es QSGDQM.

QDSHAR

Cola compartida definida dinámicamente.

La cola es una cola permanente compartida creada por una aplicación que emite una llamada MQOPEN con el nombre de una cola modelo especificada en el descriptor de objeto MQOD. La definición de cola modelo tenía el valor QDSHAR para el atributo **DefinitionType** .

Este tipo de cola se puede suprimir utilizando la llamada MQCLOSE. Consulte [“MQCLOSE \(Cerrar objeto\) en IBM i”](#) en la página 1296 para obtener más detalles.

El valor del atributo **QSGDisp** para una cola dinámica compartida es QSGDSH.

Este atributo en una definición de cola modelo no indica cómo se ha definido la cola modelo, porque las colas modelo siempre están predefinidas. En su lugar, el valor de este atributo en la cola modelo se utiliza para determinar el *DefinitionType* de cada una de las colas dinámicas creadas a partir de la definición de cola modelo utilizando la llamada MQOPEN.

Para determinar el valor de este atributo, utilice el selector IADEFT con la llamada MQINQ.

DefInputOpenOption (entero con signo de 10 dígitos) en IBM i

Opción de apertura de entrada predeterminada.

Tabla 809. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X			

Esta es la forma predeterminada en la que se debe abrir la cola para entrada. Se aplica si se especifica la opción OOINPQ en la llamada MQOPEN cuando se abre la cola. Puede tener uno de los valores siguientes:

OOINPX

Abra la cola para obtener mensajes con acceso exclusivo.

La cola se abre para su uso con las llamadas MQGET posteriores. La llamada falla con el código de razón RC2042 si la cola está abierta actualmente por esta u otra aplicación para entrada de cualquier tipo (OOINPS u OOIINPX).

OOINPS

Abra la cola para obtener mensajes con acceso compartido.

La cola se abre para su uso con las llamadas MQGET posteriores. La llamada puede realizarse correctamente si la cola está abierta actualmente por esta u otra aplicación con OOINPS, pero falla con el código de razón RC2042 si la cola está abierta actualmente con OOIINPX.

Para determinar el valor de este atributo, utilice el selector IADINP con la llamada MQINQ.

DefPersistence (entero con signo de 10 dígitos) en IBM i

Persistencia de mensajes predeterminada.

Tabla 810. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X	X	X	X

Es la persistencia predeterminada de los mensajes en la cola. Se aplica si se especifica PEQDEF en el descriptor de mensaje cuando se coloca el mensaje.

Si hay más de una definición en la vía de acceso de resolución de nombre de cola, la persistencia predeterminada se toma del valor de este atributo en la *primera* definición de la vía de acceso en el momento de la llamada MQPUT o MQPUT1. Puede ser lo siguiente:

- una cola alias
- Una cola local
- Una definición local de una cola remota
- Un alias de gestor de colas

- Una cola de transmisión (por ejemplo, la cola *DefXmitQName*)

Puede tener uno de los valores siguientes:

PEPER

El mensaje es persistente.

Esto significa que el mensaje sobrevive a las anomalías del sistema y a los reinicios del gestor de colas. Los mensajes persistentes no se pueden colocar en:

- Colas dinámicas temporales
- Colas compartidas

Los mensajes persistentes se pueden colocar en colas dinámicas permanentes y colas predefinidas.

PENDIENTE

El mensaje no es persistente.

Esto significa que el mensaje normalmente no sobrevive a las anomalías del sistema o a los reinicios del gestor de colas. Esto se aplica incluso si se encuentra una copia intacta del mensaje en el almacenamiento auxiliar durante el reinicio del gestor de colas.

En el caso especial de las colas compartidas, los mensajes no persistentes *sí* sobreviven a los reinicios de los gestores de colas en el grupo de compartición de colas, pero no sobreviven a las anomalías del recurso de acoplamiento utilizado para almacenar mensajes en las colas compartidas.

Los mensajes persistentes y no persistentes pueden existir en la misma cola.

Para determinar el valor de este atributo, utilice el selector IADPER con la llamada MQINQ.

 **DefPriority (entero con signo de 10 dígitos) en IBM i**

Prioridad de mensaje predeterminada.

Tabla 811. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X	X	X	X

Esta es la prioridad predeterminada para los mensajes de la cola. Esto se aplica si se especifica PRQDEF en el descriptor de mensaje cuando el mensaje se coloca en la cola.

Si hay más de una definición en la vía de acceso de resolución de nombre de cola, la prioridad predeterminada para el mensaje se toma del valor de este atributo en la *primera* definición de la vía de acceso en el momento de la operación de colocación. Puede ser lo siguiente:

- una cola alias
- Una cola local
- Una definición local de una cola remota
- Un alias de gestor de colas
- Una cola de transmisión (por ejemplo, la cola *DefXmitQName*)

La forma en que se coloca un mensaje en una cola depende del valor del atributo

MsgDeliverySequence de la cola:

- Si el atributo **MsgDeliverySequence** es MSPRIO, la posición lógica en la que se coloca un mensaje en la cola depende del valor del campo *MDPRI* en el descriptor de mensaje.
- Si el atributo **MsgDeliverySequence** es MSFIFO, los mensajes se colocan en la cola como si tuvieran una prioridad igual a *DefPriority* de la cola resuelta, independientemente del valor del campo *MDPRI* en el descriptor de mensaje. Sin embargo, el campo *MDPRI* conserva el valor especificado por la aplicación que ha colocado el mensaje. Consulte el atributo **MsgDeliverySequence** descrito en [“Atributos para colas” en la página 1399](#) para obtener más información.

Las prioridades están en el rango de cero (menor) a *MaxPriority* (mayor); consulte el atributo **MaxPriority** descrito en “Atributos del gestor de colas en IBM i” en la página 1432.

Para determinar el valor de este atributo, utilice el selector IADPRI con la llamada MQINQ.

DefReadAhead (entero con signo de 10 dígitos) en IBM i

Especifica el comportamiento de lectura anticipada predeterminado para los mensajes no persistentes entregados al cliente.

Tabla 812. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X	X		

DefReadAhead puede establecerse en uno de los siguientes valores:

RRAHNO

Los mensajes no persistentes no se envían al cliente antes de que una aplicación los solicite. Puede perderse un mensaje no persistente como máximo, si el cliente finaliza de forma anómala.

RAYA

Los mensajes no persistentes se envían al cliente antes de que una aplicación los solicite. Los mensajes no persistentes se pueden perder si el cliente finaliza de forma anómala o si el cliente no consume todos los mensajes que se envían.

SRAHDIS

Lectura anticipada de mensajes no persistentes en no habilitados para esta cola. Los mensajes no se envían al cliente independientemente de si la aplicación cliente solicita la lectura anticipada.

Para determinar el valor de este atributo, utilice el selector IADRAH con la llamada MQINQ.

DefPResp (entero con signo de 10 dígitos) en IBM i

El atributo de tipo de respuesta put por defecto (DEFPRESP) define el valor utilizado por las aplicaciones cuando el PutResponseType dentro de MQPMO se ha establecido en PMRASQ. Este atributo es válido para todos los tipos de cola.

Tabla 813. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X	X	X	X

Puede tener uno de los valores siguientes:

SYNC

La operación de colocación se emite de forma síncrona devolviendo una respuesta.

ASYNC

La operación de transferencia se emite de forma asíncrona, devolviendo un subconjunto de campos MQMD.

Para determinar el valor de este atributo, utilice el selector IADPRT con la llamada MQINQ.

DistLists (entero con signo de 10 dígitos) en IBM i

Soporte de lista de distribución.

Tabla 814. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X			

Indica si los mensajes de lista de distribución se pueden colocar en la cola. El atributo lo establece un agente de canal de mensajes (MCA) para informar al gestor de colas local si el gestor de colas del otro extremo del canal da soporte a listas de distribución. Este último gestor de colas (denominado "gestor de colas asociado") es el que recibe a continuación el mensaje, después de que un MCA emisor lo haya eliminado de la cola de transmisión local.

El MCA emisor establece el atributo siempre que establece una conexión con el MCA receptor en el gestor de colas asociado. De este modo, el MCA emisor puede hacer que el gestor de colas local coloque en la cola de transmisión sólo los mensajes que el gestor de colas asociado puede procesar correctamente.

Este atributo se utiliza principalmente con colas de transmisión, pero el proceso descrito se realiza independientemente del uso definido para la cola (consulte el atributo **Usage**).

Puede tener uno de los valores siguientes:

DLSUPP

Listas de distribución soportadas.

Esto indica que los mensajes de lista de distribución pueden almacenarse en la cola y transmitirse al gestor de colas asociado en ese formato. Esto reduce la cantidad de proceso necesaria para enviar el mensaje a varios destinos.

DLNSUP

Listas de distribución no soportadas.

Esto indica que los mensajes de lista de distribución no se pueden almacenar en la cola, porque el gestor de colas asociado no da soporte a las listas de distribución. Si una aplicación coloca un mensaje de lista de distribución y ese mensaje se va a colocar en esta cola, el gestor de colas divide el mensaje de lista de distribución y coloca los mensajes individuales en la cola. Esto aumenta la cantidad de proceso necesaria para enviar el mensaje a varios destinos, pero garantiza que el gestor de colas asociado procesará correctamente los mensajes.

Para determinar el valor de este atributo, utilice el selector IADIST con la llamada MQINQ. Para cambiar el valor de este atributo, utilice la llamada MQSET.

HardenGetBackout (entero con signo de 10 dígitos) en IBM i

Indica si se debe mantener un recuento de restituciones preciso.

Tabla 815. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X			

Para cada mensaje, se mantiene un recuento del número de veces que una llamada MQGET recupera el mensaje dentro de una unidad de trabajo y dicha unidad de trabajo se restituye posteriormente. Este recuento está disponible en el campo *MDBOC* del descriptor de mensaje después de que se haya completado la llamada MQGET.

El recuento de restituciones de mensajes sobrevive cuando se reinicia el gestor de colas. Sin embargo, para asegurarse de que el recuento es preciso, la información debe "guardarse" (grabarse en disco u otro dispositivo de almacenamiento permanente) cada vez que una llamada MQGET recupera un mensaje dentro de una unidad de trabajo para esta cola. Si esto no se hace, y se produce una anomalía del gestor de colas junto con la restitución de la llamada MQGET, es posible que el recuento no se incremente.

Sin embargo, la información de protección para cada llamada MQGET dentro de una unidad de trabajo impone un coste de rendimiento, y el atributo **HardenGetBackout** debe establecerse en QABH sólo si el recuento debe ser preciso.

- En IBM i, el recuento de restituciones de mensajes siempre está protegido, independientemente del valor de este atributo.

Son posibles los siguientes valores:

QABH

Recuento de restituciones recordado.

El refuerzo se utiliza para asegurarse de que el recuento de restituciones para los mensajes de esta cola es preciso.

QABNH

Es posible que no se recuerde el recuento de restituciones.

El refuerzo no se utiliza para asegurarse de que el recuento de restituciones para los mensajes de esta cola es preciso. Por lo tanto, el recuento puede ser menor de lo que debería ser.

Para determinar el valor de este atributo, utilice el selector IAHGB con la llamada MQINQ.

InhibitGet (entero con signo de 10 dígitos) en IBM i

Controla si se permiten operaciones get para esta cola.

Tabla 816. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X	X		

Si la cola es una cola alias, se deben permitir operaciones get para el alias y la cola base en el momento de la operación get, para que la llamada MQGET sea satisfactoria. El valor puede ser uno de los siguientes:

QAGETI

Las operaciones de obtención están inhibidas.

Las llamadas MQGET fallan con el código de razón RC2016. Esto incluye las llamadas MQGET que especifican GMBRWF o GMBRWN.

Nota: Si una llamada MQGET que opera dentro de una unidad de trabajo se completa correctamente, el cambio del valor del atributo **InhibitGet** después de QAGETI no impide que se confirme la unidad de trabajo.

QAGETA

Las operaciones de obtención están permitidas.

Para determinar el valor de este atributo, utilice el selector IAIGET con la llamada MQINQ. Para cambiar el valor de este atributo, utilice la llamada MQSET.

InhibitPut (entero con signo de 10 dígitos) en IBM i

Controla si se permiten operaciones de colocación para esta cola.

Tabla 817. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X	X	X	X

Si hay más de una definición en la vía de acceso de resolución de nombre de cola, se deben permitir operaciones de colocación para *cada* definición en la vía de acceso (incluidas las definiciones de alias de gestor de colas) en el momento de la operación de colocación, para que la llamada MQPUT o MQPUT1 sea satisfactoria. Puede tener uno de los valores siguientes:

QAPUTI

Las operaciones de colocación están inhibidas.

Las llamadas MQPUT y MQPUT1 fallan con el código de razón RC2051.

Nota: Si una llamada MQPUT que opera dentro de una unidad de trabajo se completa correctamente, el cambio del valor del atributo **InhibitPut** posteriormente a QAPUTI no impide que se confirme la unidad de trabajo.

QAPUTA

Las operaciones de colocación están permitidas.

Para determinar el valor de este atributo, utilice el selector IAIPUT con la llamada MQINQ. Para cambiar el valor de este atributo, utilice la llamada MQSET.

IBM i **InitiationQName** (serie de caracteres de 48 bytes) en IBM i

Es el nombre de la cola de inicio.

Tabla 818. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X			

Es el nombre de una cola definida en el gestor de colas local; la cola debe ser de tipo QTLOC. El gestor de colas envía un mensaje desencadenante a la cola de inicio cuando el inicio de la aplicación es necesario como resultado de un mensaje que llega a la cola a la que pertenece este atributo. La cola de inicio debe ser supervisada por una aplicación de supervisor desencadenante que iniciará la aplicación adecuada después de recibir el mensaje desencadenante.

Para determinar el valor de este atributo, utilice el selector CAINIQ con la llamada MQINQ. La longitud de este atributo la proporciona LNQN.

IBM i **MaxMsgLength** (entero con signo de 10 dígitos) en IBM i

Longitud máxima de mensaje en bytes.

Tabla 819. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X			

Este es un límite superior para la longitud del mensaje *físico* más largo que se puede colocar en la cola. Sin embargo, debido a que el atributo de cola **MaxMsgLength** se puede establecer independientemente del atributo de gestor de colas **MaxMsgLength**, el límite superior real para la longitud del mensaje físico más largo que se puede colocar en la cola es el menor de estos dos valores.

Si el gestor de colas da soporte a la segmentación, es posible que una aplicación coloque un mensaje *lógico* que sea más largo que el menor de los dos atributos **MaxMsgLength**, pero sólo si la aplicación especifica el distintivo MFSEGA en MQMD. Si se especifica ese distintivo, el límite superior para la longitud de un mensaje lógico es de 999.999.999 bytes, pero normalmente, las restricciones de recursos impuestas por el sistema operativo o por el entorno en el que se ejecuta la aplicación, dan como resultado un límite inferior.

Un intento de colocar en la cola un mensaje demasiado largo falla con el código de razón:

- RC2030 si el mensaje es demasiado grande para la cola
- RC2031 si el mensaje es demasiado grande para el gestor de colas, pero no demasiado grande para la cola

El límite inferior para el atributo **MaxMsgLength** es cero. El límite superior lo determina el entorno:

- En IBM i, la longitud máxima del mensaje es de 100 MB (104 857 600 bytes).

Para obtener más información, consulte el parámetro **BUFLN** descrito en “MQPUT (transferir mensaje) en IBM i” en la página 1363.

Para determinar el valor de este atributo, utilice el selector IAMLEN con la llamada MQINQ.

MaxQDepth (entero con signo de 10 dígitos) en IBM i

Es la profundidad máxima de la cola.

Tabla 820. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X			

Es el límite superior definido para el número de mensajes físicos que pueden existir en la cola en cualquier momento. Un intento de colocar un mensaje en una cola que ya contiene mensajes *MaxQDepth* falla con el código de razón RC2053.

El proceso de unidad de trabajo y la segmentación de mensajes pueden hacer que el número real de mensajes físicos en la cola supere *MaxQDepth*. Sin embargo, esto no afecta a la capacidad de recuperación de los mensajes- todos los mensajes de la cola se pueden recuperar utilizando la llamada MQGET de la forma normal.

El valor de este atributo es cero o mayor. El límite superior lo determina el entorno.

Nota: Es posible que el espacio de almacenamiento disponible para la cola se agote incluso si hay menos de *MaxQDepth* mensajes en la cola.

Para determinar el valor de este atributo, utilice el selector IAMDEP con la llamada MQINQ.

MediaLog (entero con signo de 10 dígitos) en IBM i

Identidad de la extensión de registro (o receptor de diario en IBM i) necesario para la recuperación de soporte de una cola determinada.

Tabla 821. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X			

En los gestores de colas donde el registro circular está en uso, el valor se devuelve como una serie nula.

MsgDeliverySequence (entero con signo de 10 dígitos) en IBM i

Secuencia de entrega de mensajes.

Tabla 822. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X			

Esto determina el orden en el que la llamada MQGET devuelve los mensajes a la aplicación:

MSFIFO

Los mensajes se devuelven en orden FIFO (primero en entrar, primero en salir).

Esto significa que una llamada MQGET devolverá el *primer* mensaje que cumpla los criterios de selección especificados en la llamada, independientemente de la prioridad del mensaje.

MSPRIO

Los mensajes se devuelven en orden de prioridad.

Esto significa que una llamada MQGET devolverá el mensaje de *prioridad más alta* que satisface los criterios de selección especificados en la llamada. Dentro de cada nivel de prioridad, los mensajes se devuelven en orden FIFO (primero en entrar, primero en salir).

Si los atributos relevantes se cambian mientras hay mensajes en la cola, la secuencia de entrega es la siguiente:

- El orden en el que la llamada MQGET devuelve los mensajes viene determinado por los valores de los atributos **MsgDeliverySequence** y **DefPriority** en vigor para la cola en el momento en que el mensaje llega a la cola:
 - Si *MsgDeliverySequence* es MSFIFO cuando llega el mensaje, el mensaje se coloca en la cola como si su prioridad fuera *DefPriority*. Esto no afecta al valor del campo *MDPRI* en el descriptor de mensaje del mensaje; dicho campo conserva el valor que tenía cuando se colocó el mensaje por primera vez.
 - Si *MsgDeliverySequence* es MSPRIO cuando llega el mensaje, el mensaje se coloca en la cola en el lugar adecuado a la prioridad proporcionada por el campo *MDPRI* en el descriptor de mensaje.

Si el valor del atributo **MsgDeliverySequence** cambia mientras hay mensajes en la cola, el orden de los mensajes en la cola no cambia.

Si el valor del atributo **DefPriority** se cambia mientras hay mensajes en la cola, los mensajes no se entregarán necesariamente en orden FIFO, aunque el atributo **MsgDeliverySequence** se establezca en MSFIFO; los que se colocaron en la cola con la prioridad más alta se entregarán primero.

Para determinar el valor de este atributo, utilice el selector IAMDS con la llamada MQINQ.

IBM i **OpenInputCount (entero con signo de 10 dígitos) en IBM i**

Es el número de aperturas para entrada.

Tabla 823. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X				

Es el número de descriptores de contexto que son válidos actualmente para eliminar mensajes de la cola con la llamada MQGET. Es el número total de manejadores de este tipo conocidos por el gestor de colas *local*. Si la cola es una cola compartida, el recuento no incluye las aperturas para la entrada que se han realizado para la cola en otros gestores de colas del grupo de compartición de colas al que pertenece el gestor de colas local.

El recuento incluye los descriptores de contexto en los que se ha abierto una cola alias que se resuelve en esta cola para entrada. El recuento no incluye los descriptores de contexto en los que se ha abierto la cola para acciones que no incluían entrada (por ejemplo, una cola abierta sólo para examinar).

El valor de este atributo fluctúa a medida que opera el gestor de colas.

Para determinar el valor de este atributo, utilice el selector IAOIC con la llamada MQINQ.

IBM i **OpenOutputCount (entero con signo de 10 dígitos) en IBM i**

Es el número de aperturas para salida.

Tabla 824. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remoto	Clúster
X				

Es el número de descriptores de contexto que son válidos actualmente para añadir mensajes a la cola con la llamada MQPUT. Es el número total de manejadores de este tipo conocidos por el gestor de colas *local*; no incluye las aperturas para salida que se han realizado para esta cola en los gestores de colas remotos. Si la cola es una cola compartida, el recuento no incluye las aperturas para la salida que se han realizado para la cola en otros gestores de colas del grupo de compartición de colas al que pertenece el gestor de colas local.

El recuento incluye los descriptores de contexto en los que se ha abierto para salida una cola alias que se resuelve en esta cola. El recuento no incluye los descriptores de contexto en los que se ha abierto la cola para acciones que no incluían salida (por ejemplo, una cola abierta sólo para consulta).

El valor de este atributo fluctúa a medida que opera el gestor de colas.

Para determinar el valor de este atributo, utilice el selector IAOC con la llamada MQINQ.

IBM i **ProcessName (serie de caracteres de 48 bytes) en IBM i**

Nombre del proceso.

Tabla 825. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remoto	Clúster
X	X			

Es el nombre de un objeto de proceso definido en el gestor de colas local. El objeto de proceso identifica un programa que puede dar servicio a la cola.

Para determinar el valor de este atributo, utilice el selector CAPRON con la llamada MQINQ. La longitud de este atributo la proporciona LNPRON.

IBM i **QDepthHighEvent (entero con signo de 10 dígitos) en IBM i**

Controla si se generan sucesos de profundidad de cola alta.

Tabla 826. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remoto	Clúster
X	X			

Un suceso de profundidad de cola alta indica que una aplicación ha colocado un mensaje en una cola, lo que ha hecho que el número de mensajes de la cola sea mayor o igual que el umbral de profundidad de cola alta (consulte el atributo **QDepthHighLimit**).

Nota: El valor de este atributo puede cambiar dinámicamente.

QDepthHighEvent puede tener uno de dos valores:

EVREDIS

Informes de sucesos inhabilitados.

EVRENA

Informes de sucesos habilitados.

Para obtener más información sobre los sucesos, consulte [Supervisión de sucesos](#).

Para determinar el valor de este atributo, utilice el selector IAQDHE con la llamada MQINQ.

IBM i **QDepthHighLimit (entero con signo de 10 dígitos) en IBM i**
Límite alto para la profundidad de cola.

Tabla 827. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X			

Es el umbral con el que se compara la profundidad de cola para generar un suceso Profundidad de cola alta. Este suceso indica que una aplicación ha colocado un mensaje en una cola, y esto ha hecho que el número de mensajes en la cola sea mayor o igual que el umbral alto de profundidad de cola. Consulte el atributo **QDepthHighEvent**.

El valor se expresa como un porcentaje de la profundidad máxima de cola (atributo **MaxQDepth**) y está en el rango de cero a 100. El valor predeterminado es 80.

Para determinar el valor de este atributo, utilice el selector IAQDHL con la llamada MQINQ.

IBM i **QDepthLowEvent (entero con signo de 10 dígitos) en IBM i**
Controla si se generan sucesos de Profundidad de cola baja.

Tabla 828. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X			

Un suceso Profundidad de cola baja indica que una aplicación ha recuperado un mensaje de una cola, lo que ha hecho que el número de mensajes de la cola sea menor o igual que el umbral de profundidad de cola baja (consulte el atributo **QDepthLowLimit**).

Nota: El valor de este atributo puede cambiar dinámicamente.

QDepthLowEvent puede tener uno de los siguientes valores:

EVRDIS

Informes de sucesos inhabilitados.

EVRENA

Informes de sucesos habilitados.

Para obtener más información sobre los sucesos, consulte [Supervisión de sucesos](#).

Para determinar el valor de este atributo, utilice el selector IAQDLE con la llamada MQINQ.

IBM i **QDepthLowLimit (entero con signo de 10 dígitos) en IBM i**
Límite bajo para profundidad de cola.

Tabla 829. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X			

Es el umbral con el que se compara la profundidad de cola para generar un suceso Profundidad de cola baja. Este suceso indica que una aplicación ha recuperado un mensaje de una cola y esto ha hecho que el

número de mensajes de la cola sea menor o igual que el umbral bajo de profundidad de cola. Consulte el atributo **QDepthLowEvent**.

El valor se expresa como un porcentaje de la profundidad máxima de cola (atributo **MaxQDepth**) y está en el rango de cero a 100. El valor predeterminado es 20.

Para determinar el valor de este atributo, utilice el selector IAQDLL con la llamada MQINQ.

QDepthMaxEvent (entero con signo de 10 dígitos) en IBM i

Controla si se generan sucesos Cola llena.

Tabla 830. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X			

Un suceso Cola llena indica que una colocación en una cola se ha rechazado porque la cola está llena, es decir, la profundidad de cola ya ha alcanzado su valor máximo.

Nota: El valor de este atributo puede cambiar dinámicamente.

Puede tener uno de los valores siguientes:

EVRDIS

Informes de sucesos inhabilitados.

EVRENA

Informes de sucesos habilitados.

Para obtener más información sobre los sucesos, consulte [Supervisión de sucesos](#).

Para determinar el valor de este atributo, utilice el selector IAQDME con la llamada MQINQ.

QDesc (serie de caracteres de 64 bytes) en IBM i

Descripción de cola.

Tabla 831. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X	X	X	X

Este es un campo que se puede utilizar para comentarios descriptivos. El contenido del campo no es significativo para el gestor de colas, pero es posible que el gestor de colas requiera que el campo sólo contenga caracteres que se puedan visualizar. No puede contener ningún carácter nulo; si es necesario, se rellena a la derecha con espacios en blanco. En una instalación DBCS, el campo puede contener caracteres DBCS (sujetos a una longitud máxima de campo de 64 bytes).

Nota: Si este campo contiene caracteres que no están en el juego de caracteres del gestor de colas (tal como lo define el atributo de gestor de colas **CodedCharSetId**), estos caracteres podrían traducirse incorrectamente si este campo se envía a otro gestor de colas.

Para determinar el valor de este atributo, utilice el selector CAQD con la llamada MQINQ. La longitud de este atributo la proporciona LNQD.

QName (serie de caracteres de 48 bytes) en IBM i

Nombre de cola.

Tabla 832. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remoto	Clúster
X		X	X	X

Es el nombre de una cola definida en el gestor de colas local. Para obtener más información sobre los nombres de cola, consulte Reglas para la denominación de objetos de IBM MQ. Todas las colas definidas en un gestor de colas comparten el mismo espacio de nombres de cola. Por lo tanto, una cola QTLOC y una cola QTALS no pueden tener el mismo nombre.

Para determinar el valor de este atributo, utilice el selector CAQN con la llamada MQINQ. La longitud de este atributo la proporciona LNQN.

IBM i **QServiceInterval** (entero con signo de 10 dígitos) en IBM i

Destino del intervalo de servicio de cola.

Tabla 833. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remoto	Clúster
X	X			

Este es el intervalo de servicio utilizado para la comparación para generar sucesos de intervalo de servicio alto y de intervalo de servicio correcto. Consulte el atributo **QServiceIntervalEvent**.

El valor está en unidades de milisegundos y está en el rango de cero a 999.999.999.

Para determinar el valor de este atributo, utilice el selector IAQSI con la llamada MQINQ.

IBM i **QServiceIntervalEvent** (entero con signo de 10 dígitos) en IBM i

Controla si se generan sucesos de intervalo de servicio alto o de intervalo de servicio correcto.

Tabla 834. Tipos de cola a los que se aplica este atributo

Local	Modelo	Alias	Remoto	Clúster
X	X			

- Se genera un suceso de intervalo de servicio alto cuando una comprobación indica que no se ha recuperado ningún mensaje de la cola durante al menos el tiempo indicado por el atributo **QServiceInterval**.
- Se genera un suceso de intervalo de servicio correcto cuando una comprobación indica que los mensajes se han recuperado de la cola dentro del tiempo indicado por el atributo **QServiceInterval**.

Nota: El valor de este atributo puede cambiar dinámicamente.

Este atributo puede tener uno de los siguientes valores:

QSIEHI

Sucesos de intervalo de servicio de cola alto habilitados.

- Los sucesos de intervalo de servicio de cola alto están **habilitados** y
- Los sucesos de intervalo de servicio de cola correcto están **inhabilitados**.

QSIEOK

Sucesos de intervalo de servicio de cola correcto habilitados.

- Los sucesos de intervalo de servicio de cola alto están **inhabilitados** y

- Los sucesos de intervalo de servicio de cola correcto están **habilitados**.

QSIENO

No hay sucesos de intervalo de servicio de cola habilitados.

- Los sucesos de intervalo de servicio de cola alto están **inhabilitados** y
- Los sucesos de intervalo de servicio de cola correcto también están **inhabilitados**.

Para las colas compartidas, se ignora el valor de este atributo; se asume el valor QSIENO.

Para obtener más información sobre los sucesos, consulte [Supervisión de sucesos](#).

Para determinar el valor de este atributo, utilice el selector IAQSIE con la llamada MQINQ.

IBM i QSGDisp (entero con signo de 10 dígitos) en IBM i

Disposición del grupo de compartición de colas.

Tabla 835. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X		X	X	

Especifica la disposición de la cola. El valor puede ser uno de los siguientes:

QSGDQM

Disposición del gestor de colas.

El objeto tiene disposición de gestor de colas. Esto significa que la definición de objeto sólo es conocida por el gestor de colas local; la definición no es conocida por otros gestores de colas del grupo de compartición de colas.

Es posible que cada gestor de colas del grupo de compartición de colas tenga un objeto con el mismo nombre y tipo que el objeto actual, pero estos son objetos separados y no hay ninguna correlación entre ellos. Sus atributos no están restringidos a ser iguales entre sí.

QSGDCP

Disposición de objeto copiado.

El objeto es una copia local de una definición de objeto maestro que existe en el repositorio compartido. Cada gestor de colas del grupo de compartición de colas puede tener su propia copia del objeto. Inicialmente, todas las copias tienen los mismos atributos, pero utilizando mandatos MQSC, cada copia se puede modificar para que sus atributos difieran de los de las otras copias. Los atributos de las copias se resincronizan cuando se modifica la definición maestra en el repositorio compartido.

QSGDSH

Disposición compartida.

El objeto tiene una disposición compartida. Esto significa que existe en el repositorio compartido una única instancia del objeto que es conocida por todos los gestores de colas del grupo de compartición de colas. Cuando un gestor de colas del grupo accede al objeto, accede a la única instancia compartida del objeto.

Para determinar el valor de este atributo, utilice el selector IAQSGD con la llamada MQINQ.

z/OS Este atributo sólo está soportado en z/OS.

IBM i QType (entero con signo de 10 dígitos) en IBM i

El tipo de cola.

Tabla 836. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X		X	X	X

Este atributo tiene uno de los siguientes valores:

QTALS

Definición de cola alias.

QTCLUS

Cola de clúster.

QTLOC

Cola local.

QTREM

Definición local de una cola remota.

Para determinar el valor de este atributo, utilice el selector IAQTYP con la llamada MQINQ.

 RemoteQMGrName (cadena de caracteres de 48 bytes) en IBM i

Nombre del gestor de colas remoto.

Tabla 837. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
			X	

Es el nombre del gestor de colas remoto en el que se define la cola *RemoteQName*. Si la cola *RemoteQName* tiene un valor *QSGDisp* de QSGDCP o QSGDSH, *RemoteQMGrName* puede ser el nombre del grupo de compartición de colas propietario de *RemoteQName*.

Si una aplicación abre la definición local de una cola remota, *RemoteQMGrName* no debe estar en blanco y no debe ser el nombre del gestor de colas local. Si *XmitQName* está en blanco, se utiliza la cola local con el mismo nombre que *RemoteQMGrName* como cola de transmisión. Si no hay ninguna cola con el nombre *RemoteQMGrName*, se utiliza la cola identificada por el atributo de gestor de colas **DefXmitQName**.

Si esta definición se utiliza para un alias de gestor de colas, *RemoteQMGrName* es el nombre del gestor de colas que se está alias. Puede ser el nombre del gestor de colas local. De lo contrario, si *XmitQName* está en blanco cuando se produce la apertura, debe haber una cola local con el mismo nombre que *RemoteQMGrName*; esta cola se utiliza como cola de transmisión.

Si esta definición se utiliza para un alias de respuesta, este nombre es el nombre del gestor de colas que debe ser *MDRM*.

Nota: No se realiza ninguna validación en el valor especificado para este atributo cuando se crea o modifica la definición de cola.

Para determinar el valor de este atributo, utilice el selector CARQMN con la llamada MQINQ. La longitud de este atributo la proporciona LNQMNI.

 RemoteQName (serie de caracteres de 48 bytes) en IBM i

Nombre de la cola remota.

Tabla 838. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
			X	

Es el nombre de la cola tal como se conoce en el gestor de colas remoto *RemoteQMGrName*.

Si una aplicación abre la definición local de una cola remota, cuando se produce la apertura, *RemoteQName* no debe estar en blanco.

Si esta definición se utiliza para una definición de alias de gestor de colas, cuando se produzca la apertura, *RemoteQName* debe estar en blanco.

Si la definición se utiliza para un alias de respuesta, este nombre es el nombre de la cola que va a ser *MDRQ*.

Nota: No se realiza ninguna validación en el valor especificado para este atributo cuando se crea o modifica la definición de cola.

Para determinar el valor de este atributo, utilice el selector CARQN con la llamada MQINQ. La longitud de este atributo la proporciona LNQN.

RetentionInterval (entero con signo de 10 dígitos) en IBM i

Intervalo de retención.

Tabla 839. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X			

Es la hora en la que se debe retener la cola. Una vez transcurrido este tiempo, la cola es apta para su supresión.

El tiempo se mide en horas, contando desde la fecha y hora en que se creó la cola. La fecha de creación de la cola se registra en *CreationDate* y la hora de creación de la cola se registra en el atributo **CreationTime**.

Esta información se proporciona para permitir que una aplicación de mantenimiento o el operador identifique y suprima colas que ya no son necesarias.

Nota: El gestor de colas nunca intenta suprimir colas basadas en este atributo, o impedir la supresión de colas con un intervalo de retención que no ha caducado; es responsabilidad del usuario hacer que se lleve a cabo cualquier acción necesaria.

Se debe utilizar un intervalo de retención realista para evitar la acumulación de colas dinámicas permanentes (consulte *DefinitionType*). Sin embargo, este atributo también se puede utilizar con colas predefinidas.

Para determinar el valor de este atributo, utilice el selector IARINT con la llamada MQINQ.

Ámbito (entero con signo de 10 dígitos) en IBM i

Controla si también existe una entrada para esta cola en un directorio de celdas.

Tabla 840. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X		X	X	

Un servicio de nombres instalable proporciona un directorio de célula. Puede tener uno de los valores siguientes:

SCOQM

Ámbito de gestor de colas.

La definición de cola tiene ámbito de gestor de colas. Esto significa que la definición de la cola no se extiende más allá del gestor de colas que la posee. Para abrir la cola para salida de algún otro gestor de colas, debe especificarse el nombre del gestor de colas propietario o el otro gestor de colas debe tener una definición local de la cola.

SCOCEL

Ámbito de célula.

La definición de cola tiene ámbito de célula. Esto significa que la definición de cola también se coloca en un directorio de célula disponible para todos los gestores de colas de la célula. La cola se puede abrir para salida de cualquiera de los gestores de colas de la célula simplemente especificando el nombre de la cola; no es necesario especificar el nombre del gestor de colas propietario de la cola. Sin embargo, la definición de cola no está disponible para ningún gestor de colas de la célula que también tenga una definición local de una cola con ese nombre, ya que la definición local tiene prioridad.

El directorio de células lo proporciona un servicio de nombres instalable, como LDAP (Protocolo ligero de acceso a directorios). Tenga en cuenta que IBM MQ ya no es compatible con el servicio de nombres DCE (Distributed Computing Environment), que antes se utilizaba para insertar definiciones de colas en un directorio DCE (que tampoco es compatible ya).

El modelo y las colas dinámicas no pueden tener ámbito de célula.

Este valor sólo es válido si se ha configurado un servicio de nombres que da soporte a un directorio de célula.

Para determinar el valor de este atributo, utilice el selector IASCOPI con la llamada MQINQ.

El soporte para este atributo está sujeto a las restricciones siguientes:

- En IBM i, el atributo está soportado, pero sólo SCOQM es válido.

Compartibilidad (entero con signo de 10 dígitos) en IBM i

Si la cola se puede compartir para entrada.

Tabla 841. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X			

Esto indica si la cola se puede abrir para entrada varias veces simultáneamente. Puede tener uno de los valores siguientes:

QASHR

La cola es compartible.

Se permiten varias aperturas con la opción OOINPS.

QANSHR

La cola no se puede compartir.

Una llamada MQOPEN con la opción OOINPS se trata como OOINPX.

Para determinar el valor de este atributo, utilice el selector IASHAR con la llamada MQINQ.

TriggerControl (entero con signo de 10 dígitos) en IBM i

Control de desencadenante.

Tabla 842. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X			

Esto controla si los mensajes desencadenantes se graban en una cola de inicio, para hacer que una aplicación se inicie para dar servicio a la cola. Es uno de los siguientes:

TCOFF

Los mensajes desencadenantes no son necesarios.

No se deben escribir mensajes desencadenantes para esta cola. El valor de *TriggerType* es irrelevante en este caso.

TCON

Mensajes desencadenantes necesarios.

Los mensajes desencadenantes deben escribirse para esta cola, cuando se produzcan los sucesos desencadenantes adecuados.

Para determinar el valor de este atributo, utilice el selector IATRGC con la llamada MQINQ. Para cambiar el valor de este atributo, utilice la llamada MQSET.

IBM i **TriggerData (serie de caracteres de 64 bytes) en IBM i**

Datos del desencadenante.

Tabla 843. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X			

Son datos de formato libre que el gestor de colas inserta en el mensaje desencadenante cuando un mensaje que llega a esta cola hace que se grave un mensaje desencadenante en la cola de inicio.

El contenido de estos datos no es significativo para el gestor de colas. Es significativo para la aplicación de supervisor desencadenante que procesa la cola de inicio o para la aplicación iniciada por el supervisor desencadenante.

La serie de caracteres no puede contener nulos. Se rellena a la derecha con espacios en blanco si es necesario.

Para determinar el valor de este atributo, utilice el selector CATRGD con la llamada MQINQ. Para cambiar el valor de este atributo, utilice la llamada MQSET. La longitud de este atributo la proporciona LNTRGD.

IBM i **TriggerDepth (entero con signo de 10 dígitos) en IBM i**

Profundidad del desencadenante.

Tabla 844. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X			

Es el número de mensajes de prioridad *TriggerMsgPriority* o superior que deben estar en la cola antes de que se escriba un mensaje desencadenante. Esto se aplica cuando *TriggerType* se establece en TTDPTH. El valor de *TriggerDepth* es uno o mayor. De lo contrario, este atributo no se utiliza.

Para determinar el valor de este atributo, utilice el selector IATRGD con la llamada MQINQ. Para cambiar el valor de este atributo, utilice la llamada MQSET.

TriggerMsgPriority (entero con signo de 10 dígitos) en IBM i

Prioridad de mensajes de umbral para desencadenantes en IBM MQ for IBM i.

Tabla 845. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X			

Esta es la prioridad de mensaje por debajo de la cual los mensajes no contribuyen a la generación de mensajes desencadenantes (es decir, el gestor de colas ignora estos mensajes al determinar si se debe generar un mensaje desencadenante). *TriggerMsgPriority* puede estar en el rango de cero (más bajo) a *MaxPriority* (más alto; consulte [“Atributos del gestor de colas en IBM i” en la página 1432](#)); un valor de cero hace que todos los mensajes contribuyan a la generación de mensajes desencadenantes.

Para determinar el valor de este atributo, utilice el selector IATRGP con la llamada MQINQ. Para cambiar el valor de este atributo, utilice la llamada MQSET.

TriggerType (entero con signo de 10 dígitos) en IBM i

Tipo de desencadenante.

Tabla 846. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X			

Esto controla las condiciones en las que se graban los mensajes desencadenantes como resultado de los mensajes que llegan a esta cola. El valor puede ser uno de los siguientes:

TTNONA

No hay mensajes desencadenantes.

No se graban mensajes desencadenantes como resultado de los mensajes de esta cola. Esto tiene el mismo efecto que establecer *TriggerControl* en TCOFF.

TTFRST

Desencadenar mensaje cuando la profundidad de cola va de 0 a 1.

Se graba un mensaje desencadenante siempre que el número de mensajes de prioridad *TriggerMsgPriority* o superior en la cola cambia de 0 a 1.

TTEVRY

Mensaje desencadenante para cada mensaje.

Se graba un mensaje desencadenante siempre que llega a la cola un mensaje de prioridad *TriggerMsgPriority* o superior.

TTDPH

Desencadenar mensaje cuando se supere el umbral de profundidad.

Un mensaje desencadenante se graba siempre que el número de mensajes de prioridad *TriggerMsgPriority* o superior en la cola es igual o superior a *TriggerDepth*. Después de que se haya escrito el mensaje desencadenante, *TriggerControl* se establece en TCOFF para evitar que se desencadene más hasta que se vuelva a activar explícitamente.

Para determinar el valor de este atributo, utilice el selector IATRGT con la llamada MQINQ. Para cambiar el valor de este atributo, utilice la llamada MQSET.

► IBM i **Uso (entero con signo de 10 dígitos) en IBM i**

Uso de cola.

Tabla 847. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
X	X			

Indica para qué se utiliza la cola. El valor puede ser uno de los siguientes:

USNORM

Uso normal.

Se trata de una cola que las aplicaciones normales utilizan al transferir y obtener mensajes; la cola no es una cola de transmisión.

USTRAN

Cola de transmisión.

Es una cola utilizada para contener mensajes destinados a gestores de colas remotos. Cuando una aplicación normal envía un mensaje a una cola remota, el gestor de colas local almacena el mensaje temporalmente en la cola de transmisión adecuada en un formato especial. A continuación, un agente de canal de mensajes lee el mensaje de la cola de transmisión y lo transporta al gestor de colas remoto. Para obtener más información sobre las colas de transmisión, consulte [Colas de transmisión](#).

Sólo las aplicaciones con privilegios pueden abrir una cola de transmisión para que OOOOUT coloque mensajes en ella directamente. Normalmente sólo se espera que las aplicaciones de programa de utilidad lo hagan. Debe tener cuidado de que el formato de datos del mensaje sea correcto (consulte [“MQXQH \(encabezado de cola de transmisión\) en IBM i” en la página 1273](#)); de lo contrario, se pueden producir errores durante el proceso de transmisión. El contexto no se pasa ni se establece a menos que se especifique una de las opciones de contexto PM*.

Para determinar el valor de este atributo, utilice el selector IAUSAG con la llamada MQINQ.

► IBM i **XmitQName (serie de caracteres de 48 bytes) en IBM i**

Nombre de la cola de transmisión.

Tabla 848. Tipos de cola a los que se aplica este atributo				
Local	Modelo	Alias	Remoto	Clúster
			X	

Si este atributo no está en blanco cuando se produce una apertura, ya sea para una cola remota o para una definición de alias de gestor de colas, especifica el nombre de la cola de transmisión local que se utilizará para reenviar el mensaje.

Si *XmitQName* está en blanco, se utiliza la cola local con el mismo nombre que *RemoteQMGrName* como cola de transmisión. Si no hay ninguna cola con el nombre *RemoteQMGrName*, se utiliza la cola identificada por el atributo de gestor de colas **DefXmitQName**.

Este atributo se ignora si la definición se está utilizando como alias de gestor de colas y *RemoteQMGrName* es el nombre del gestor de colas local. Este atributo también se ignora si la definición se utiliza como definición de alias de cola de respuestas.

Para determinar el valor de este atributo, utilice el selector CAXQN con la llamada MQINQ. La longitud de este atributo la proporciona LNQN.

Atributos de las listas de nombres

En este tema se resumen los atributos específicos de las listas de nombres. Los atributos se describen en orden alfabético.

Nota: Los nombres de los atributos mostrados son los nombres utilizados con las llamadas MQINQ y MQSET.

Descripciones de los atributos

Un objeto de lista de nombres tiene los atributos siguientes:

AlterationDate (serie de caracteres de 12 bytes)

Fecha en la que se cambió por última vez la definición.

Es la fecha en la que se modificó por última vez la definición. El formato de la fecha es YYYY-MM-DD, rellenado con dos espacios en blanco finales para que la longitud sea de 12 bytes.

Para determinar el valor de este atributo, utilice el selector CAALTD con la llamada MQINQ. La longitud de este atributo viene dada por LNDATE.

AlterationTime (serie de caracteres de 8 bytes)

Hora a la que se modificó por última vez la definición.

Es la hora en que se modificó por última vez la definición. El formato de la hora es HH.MM.SS.

Para determinar el valor de este atributo, utilice el selector CAALTT con la llamada MQINQ. La longitud de este atributo la proporciona LNTIME.

NameCount (entero con signo de 10 dígitos)

Número de nombres en la lista de nombres.

Es mayor o igual que cero. Se define el valor siguiente:

NCMXNL

Número máximo de nombres en una lista de nombres.

Para determinar el valor de este atributo, utilice el selector IANAMC con la llamada MQINQ.

NamelistDesc (serie de caracteres de 64 bytes)

Descripción de lista de nombres.

Se trata de un campo que se puede utilizar para comentarios descriptivos; su valor se establece mediante el proceso de definición. El contenido del campo no es significativo para el gestor de colas, pero es posible que el gestor de colas requiera que el campo sólo contenga caracteres que se puedan visualizar. No puede contener ningún carácter nulo; si es necesario, se rellena a la derecha con espacios en blanco. En una instalación DBCS, este campo puede contener caracteres DBCS (sujetos a una longitud máxima de campo de 64 bytes).

Nota: Si este campo contiene caracteres que no están en el juego de caracteres del gestor de colas (tal como lo define el atributo de gestor de colas **CodedCharSetId**), estos caracteres podrían traducirse incorrectamente si este campo se envía a otro gestor de colas.

Para determinar el valor de este atributo, utilice el selector CALSTD con la llamada MQINQ.

La longitud de este atributo la proporciona LNNLD.

NamelistName (serie de caracteres de 48 bytes)

Nombre de lista de nombres.

Es el nombre de una lista de nombres definida en el gestor de colas local.

Cada lista de nombres tiene un nombre que es diferente de los nombres de otras listas de nombres que pertenecen al gestor de colas, pero puede duplicar los nombres de otros objetos del gestor de colas de tipos diferentes (por ejemplo, colas).

Para determinar el valor de este atributo, utilice el selector CALSTN con la llamada MQINQ.

La longitud de este atributo la proporciona LNNLN.

Nombres (serie de caracteres de 48 bytes x NameCount)

Una lista de nombres de *NameCount*.

Cada nombre es el nombre de un objeto definido en el gestor de colas local. Para obtener más información sobre los nombres de objeto, consulte [Denominación de objetos IBM MQ](#).

Para determinar el valor de este atributo, utilice el selector CANAMS con la llamada MQINQ.

La longitud de cada nombre de la lista la proporciona LNOBJN.

Atributos para definiciones de proceso en IBM i

En este tema se resumen los atributos específicos de las definiciones de proceso. Los atributos se describen en orden alfabético.

Nota: Los nombres de los atributos mostrados son los nombres utilizados con las llamadas MQINQ y MQSET. Cuando se utilizan mandatos MQSC para definir, modificar o visualizar atributos, se utilizan nombres abreviados alternativos; consulte [Mandatos MQSC](#) para obtener más detalles.

Descripciones de los atributos

Un objeto de definición de proceso tiene los atributos siguientes:

AlterationDate (serie de caracteres de 12 bytes)

Fecha en la que se cambió por última vez la definición.

Es la fecha en la que se modificó por última vez la definición. El formato de la fecha es YYYY-MM-DD, rellenado con dos espacios en blanco finales para que la longitud sea de 12 bytes.

Para determinar el valor de este atributo, utilice el selector CAALTD con la llamada MQINQ. La longitud de este atributo viene dada por LNDATE.

AlterationTime (serie de caracteres de 8 bytes)

Hora a la que se modificó por última vez la definición.

Es la hora en que se modificó por última vez la definición. El formato de la hora es HH.MM.SS.

Para determinar el valor de este atributo, utilice el selector CAALTT con la llamada MQINQ. La longitud de este atributo la proporciona LNTIME.

ApplId (serie de caracteres de 256 bytes)

Identificador de aplicación.

Es una serie de caracteres que identifica la aplicación que se va a iniciar. Esta información la utiliza una aplicación de supervisor desencadenante que procesa mensajes en la cola de inicio; la información se envía a la cola de inicio como parte del mensaje desencadenante.

El significado de *ApplId* lo determina la aplicación de supervisor desencadenante. El supervisor desencadenante proporcionado por IBM MQ requiere que *ApplId* sea el nombre de un programa ejecutable.

La serie de caracteres no puede contener nulos. Se rellena a la derecha con espacios en blanco si es necesario.

Para determinar el valor de este atributo, utilice el selector CAAPPI con la llamada MQINQ. La longitud de este atributo la proporciona LNPROA.

ApplType (entero con signo de 10 dígitos)

Tipo de aplicación.

Identifica la naturaleza del programa que se va a iniciar en respuesta a la recepción de un mensaje desencadenante. Esta información la utiliza una aplicación de supervisor desencadenante que procesa mensajes en la cola de inicio; la información se envía a la cola de inicio como parte del mensaje desencadenante.

ApplType puede tener cualquier valor. Puede utilizar los valores siguientes para los tipos estándar; los tipos de aplicación definidos por el usuario están restringidos a los valores del rango ATUFST a ATULST:

a la(s)CICS

Transacción CICS .

AT400

IBM i .

ATUFST

Valor más bajo para el tipo de aplicación definido por el usuario.

ATULST

Valor más alto para el tipo de aplicación definido por el usuario.

Para determinar el valor de este atributo, utilice el selector IAAPPT con la llamada MQINQ.

EnvData (serie de caracteres de 128 bytes)

Datos de entorno.

Es una serie de caracteres que contiene información relacionada con el entorno perteneciente a la aplicación que se va a iniciar. Esta información la utiliza una aplicación de supervisor desencadenante que procesa mensajes en la cola de inicio; la información se envía a la cola de inicio como parte del mensaje desencadenante.

El significado de *EnvData* lo determina la aplicación de supervisor desencadenante. El supervisor desencadenante proporcionado por IBM MQ añade *EnvData* a la lista de parámetros pasada a la aplicación iniciada. La lista de parámetros consta de la estructura MQTMC2 , seguida de un espacio en blanco, seguida de *EnvData* con los espacios en blanco finales eliminados.

La serie de caracteres no puede contener nulos. Se rellena a la derecha con espacios en blanco si es necesario.

Para determinar el valor de este atributo, utilice el selector CAENVD con la llamada MQINQ. La longitud de este atributo la proporciona LNPROE.

ProcessDesc (serie de caracteres de 64 bytes)

Descripción de proceso.

Este es un campo que se puede utilizar para comentarios descriptivos. El contenido del campo no es significativo para el gestor de colas, pero es posible que el gestor de colas requiera que el campo contenga sólo caracteres que se puedan visualizar. No puede contener ningún carácter nulo; si es necesario, se rellena a la derecha con espacios en blanco. En una instalación DBCS, el campo puede contener caracteres DBCS (sujetos a una longitud máxima de campo de 64 bytes).

Nota: Si este campo contiene caracteres que no están en el juego de caracteres del gestor de colas (tal como lo define el atributo de gestor de colas **CodedCharSetId**), estos caracteres podrían traducirse incorrectamente si este campo se envía a otro gestor de colas.

Para determinar el valor de este atributo, utilice el selector CAPROD con la llamada MQINQ.

La longitud de este atributo la proporciona LNPROD.

ProcessName (serie de caracteres de 48 bytes)

Nombre del proceso.

Es el nombre de una definición de proceso definida en el gestor de colas local.

Cada definición de proceso tiene un nombre que es diferente de los nombres de otras definiciones de proceso que pertenecen al gestor de colas. Pero el nombre de la definición de proceso puede ser el mismo que los nombres de otros objetos de gestor de colas de tipos diferentes (por ejemplo, colas).

Para determinar el valor de este atributo, utilice el selector CAPRON con la llamada MQINQ.

La longitud de este atributo la proporciona LNPRON.

UserData (serie de caracteres de 128 bytes)

Datos de usuario.

Es una serie de caracteres que contiene información de usuario perteneciente a la aplicación que se va a iniciar. Esta información es para que la utilice una aplicación de supervisor desencadenante que procesa mensajes en la cola de inicio, o la aplicación que inicia el supervisor desencadenante. La información se envía a la cola de inicio como parte del mensaje desencadenante.

El significado de *UserData* lo determina la aplicación de supervisor desencadenante. El supervisor desencadenante proporcionado por IBM MQ pasa *UserData* a la aplicación iniciada como parte de la lista de parámetros. La lista de parámetros consta de la estructura MQTMC2 (que contiene *UserData*), seguida de un espacio en blanco, seguida de *EnvData* con los espacios en blanco finales eliminados.

La serie de caracteres no puede contener nulos. Se rellena a la derecha con espacios en blanco si es necesario.

Para determinar el valor de este atributo, utilice el selector CAUSRD con la llamada MQINQ. La longitud de este atributo la proporciona LNPROU.

Atributos del gestor de colas en IBM i

Resumen de los atributos del gestor de colas.

Algunos atributos de gestor de colas son fijos para implementaciones concretas, mientras que otros se pueden cambiar utilizando el mandato MQSC ALTER QMGR. Los atributos también se pueden visualizar utilizando el mandato DISPLAY QMGR. La mayoría de los atributos del gestor de colas se pueden consultar abriendo un objeto OTQM especial y utilizando la llamada MQINQ con el descriptor de contexto devuelto.

En la tabla siguiente se resumen los atributos que son específicos del gestor de colas. Los atributos se describen en orden alfabético.

Nota: Los nombres de los atributos que se muestran en esta sección son los nombres utilizados con las llamadas MQINQ y MQSET. Cuando se utilizan mandatos MQSC para definir, modificar o visualizar atributos, se utilizan nombres abreviados alternativos; consulte [Mandatos MQSC](#) para obtener más información.

Tabla 849. Atributos para el gestor de colas	
Atributo	Descripción
AlterationDate	Fecha en que se modificó por última vez la definición
AlterationTime	Hora a la que se modificó por última vez la definición
AuthorityEvent	Controla si se generan sucesos de autorización (no autorizados)
BridgeEvent	Controla si se generan sucesos de puente IMS
ChannelAutoDef	Controla si se permite la definición de canal automática
ChannelAutoDefEvent	Controla si se generan sucesos de definición automática de canal
ChannelAutoDefExit	Nombre de la salida de usuario para la definición automática de canal
ChannelEvent	Controla si se generan sucesos de canal

Tabla 849. Atributos para el gestor de colas (continuación)	
Atributo	Descripción
ClusterCacheType	Controla si la memoria caché de clúster es de tamaño fijo o de tamaño dinámico
ClusterWorkloadData	Datos de usuario para salida de carga de trabajo de clúster
ClusterWorkloadExit	Nombre de salida de usuario para la gestión de carga de trabajo de clúster
ClusterWorkloadLength	Longitud máxima de datos de mensaje pasados a salida de carga de trabajo de clúster
CodedCharSetId	Identificador de juego de caracteres codificado
CommandEvent	Controla si los mensajes de suceso de mandato están en cola
CommandInputQName	Nombre de cola de entrada de mandatos
CommandLevel	Nivel de mandato
ConfigurationEvent	Suceso de configuración
DeadLetterQName	Nombre de la cola de mensajes no entregados
DefClusterXmitQueueType	Tipo de cola de transmisión de clúster predeterminado.
DefXmitQName	Nombre de cola de transmisión predeterminado
DistLists	Soporte de lista de distribución
InhibitEvent	Controla si se generan sucesos de inhibición (inhibir obtención e inhibir colocación)
LocalEvent	Controla si se generan sucesos de error locales
LoggerEvent	Controla si se generan sucesos de registro de recuperación
MaxHandles	Número máximo de descriptores de contexto
MaxMsgLength	Longitud máxima de mensaje en bytes
MaxPriority	Prioridad máxima
MaxUncommittedMsgs	Número máximo de mensajes no confirmados dentro de una unidad de trabajo
PerformanceEvent	Controla si se generan sucesos relacionados con el rendimiento
Plataforma	Plataforma en la que se ejecuta el gestor de colas
PubSubMode	Si el motor de publicación/suscripción y la interfaz de publicación/suscripción en cola se están ejecutando
QMgrDesc	Descripción del gestor de colas
QMgrIdentifier	Identificador exclusivo generado internamente del gestor de colas
QMgrName	Nombre del gestor de colas
RemoteEvent	Controla si se generan sucesos de error remotos
RepositoryName	Nombre del clúster para el que este gestor de colas proporciona servicios de repositorio

Tabla 849. Atributos para el gestor de colas (continuación)	
Atributo	Descripción
RepositoryNamelist	Nombre del objeto de lista de nombres que contiene nombres de clústeres para los que este gestor de colas proporciona servicios de repositorio
SSLCRLNamelist	Nombre del objeto de lista de nombres que contiene nombres de objetos de información de autenticación (consulte la nota 1)
SSLEvent	Determina si se generan eventos de « TLS »
SSLKeyRepository	Ubicación del repositorio de claves de TLS (véase la nota 1)
SSLKeyResetCount	Determina el número de bytes no cifrados enviados y recibidos en una conversación de « TLS » antes de que se renegocie la clave de cifrado
StartStopEvent	Controla si se generan sucesos de inicio y detención
SyncPoint	Disponibilidad de punto de sincronismo
TraceRouteRecording	Controla el registro de la información de ruta de rastreo para los mensajes
TreeLifeTime	El tiempo de vida, en segundos, de los temas no administrativos
TriggerInterval	Desencadenante-intervalo de mensaje
Notas: 1. Este atributo no se puede consultar utilizando la llamada MQINQ y no se describe en esta sección. Para obtener más información sobre este atributo, consulte Cambiar gestor de colas .	

► IBM i **AlterationDate (serie de caracteres de 12 bytes) en IBM i**

Fecha en la que se cambió por última vez la definición.

Es la fecha en la que se modificó por última vez la definición. El formato de la fecha es YYYY-MM-DD, rellenado con dos espacios en blanco finales para que la longitud sea de 12 bytes.

Para determinar el valor de este atributo, utilice el selector CAALTD con la llamada MQINQ. La longitud de este atributo viene dada por LNDATE.

► IBM i **AlterationTime (serie de caracteres de 8 bytes) en IBM i**

Hora a la que se modificó por última vez la definición.

Es la hora en que se modificó por última vez la definición. El formato de la hora es HH.MM.SS.

Para determinar el valor de este atributo, utilice el selector CAALTT con la llamada MQINQ. La longitud de este atributo la proporciona LNTIME.

► IBM i **AuthorityEvent (entero con signo de 10 dígitos) en IBM i**

Controla si se generan sucesos de autorización (no autorizados).

El atributo AuthorityEvent debe establecerse en uno de los valores siguientes:

EVRDIS

Informes de sucesos inhabilitados.

EVRENA

Informes de sucesos habilitados.

Para obtener más información sobre los sucesos, consulte [Supervisión de sucesos](#).

Para determinar el valor de este atributo, utilice el selector IAAUTE con la llamada MQINQ.

BridgeEvent (serie de caracteres) en IBM i

Este atributo determina si los mensajes de sucesos de puente IMS se colocan en el SYSTEM.ADMIN.CHANNEL.EVENT . Sólo está soportado en z/OS.

ChannelAutoDef (entero con signo de 10 dígitos) en IBM i

Controla si se permite la definición automática de canal.

Este atributo controla la definición automática de canales de tipo CTCVR y CTSVCN. Tenga en cuenta que la definición automática de canales CTCLSD siempre está habilitada. Puede tener uno de los valores siguientes:

CHADDI

Definición automática de canal inhabilitada.

CHADEN

Definición automática de canal habilitada.

Para determinar el valor de este atributo, utilice el selector IACAD con la llamada MQINQ.

ChannelAutoDefEvent (entero con signo de 10 dígitos) en IBM i

Controla si se generan sucesos de definición automática de canal.

Esto se aplica a los canales de tipo CTCVR, CTSVCN y CTCLSD. Puede tener uno de los valores siguientes:

EVRDIS

Informes de sucesos inhabilitados.

EVRENA

Informes de sucesos habilitados.

Para obtener más información sobre los sucesos, consulte [Supervisión y rendimiento](#).

Para determinar el valor de este atributo, utilice el selector IACADE con la llamada MQINQ.

ChannelAutoDefExit (cadena de caracteres de 20 bytes) en IBM i

Nombre de la salida de usuario para la definición automática de canal.

Si este nombre no está en blanco y *ChannelAutoDef* tiene el valor CHADEN, se llama a la salida cada vez que el gestor de colas está a punto de crear una definición de canal. Esto se aplica a los canales de tipo CTCVR, CTSVCN y CTCLSD. A continuación, la salida puede realizar una de las acciones siguientes:

- Permitir la creación de la definición de canal para continuar sin cambios.
- Modifique los atributos de la definición de canal que se crea.
- Suprimir la creación del canal por completo.

Para determinar el valor de este atributo, utilice el selector CACADX con la llamada MQINQ. La longitud de este atributo la proporciona LNEXT.

ChannelEvent (serie de caracteres) en IBM i

Determina si se generan mensajes de sucesos de canal.

Este atributo determina si los mensajes de sucesos de canal se colocan en SYSTEM.ADMIN.CHANNEL.EVENT y, si es así, qué tipo de mensajes se ponen en cola (por ejemplo, 'canal iniciado', 'canal detenido', 'canal no activado'). Antes de la implementación de este atributo, la única forma de impedir que los mensajes de sucesos de canal se colocaran en cola era suprimir la cola de destino.

Este atributo también le permite recopilar sólo sucesos de puente IMS (porque ahora puede desactivar los sucesos de canal, no se colocan en la misma cola). Lo mismo ocurre con los eventos « TLS », que también se pueden recopilar sin necesidad de recopilar además los eventos de canal.

Este atributo también le permite recopilar sólo sucesos significativos (por ejemplo, cuando los canales tienen errores, no cuando se inician y se detienen normalmente).

El valor del atributo ChannelEvent puede ser uno de los siguientes:

- EVREXP (solo se generan los siguientes sucesos de canal: RC2279, RC2283, RC2284, RC2295, RC2296).
- EVRENA (se generan todos los sucesos de canal; es decir, además de los sucesos generados por EVREXP, también se generan los sucesos RC2282y RC2283).
- EVRDIS (no se generan sucesos de canal; este es el valor predeterminado inicial del gestor de colas).

Para determinar el valor de este atributo, utilice el selector IACHNE con la llamada MQINQ.

ClusterCacheType (cadena de caracteres de 32 bytes) en IBM i

Controla si la memoria caché de clúster es de tamaño fijo o de tamaño dinámico.

Es una serie de caracteres de 32 bytes definida por el usuario que se pasa a la salida de carga de trabajo del clúster cuando se llama. Si no hay datos para pasar a la salida, la serie está en blanco.

Para determinar el valor de este atributo, utilice el selector CACLWD con la llamada MQINQ.

ClusterWorkloadData (cadena de caracteres de 32 bytes) en IBM i

Datos de usuario para salida de carga de trabajo de clúster.

Es una serie de caracteres de 32 bytes definida por el usuario que se pasa a la salida de carga de trabajo del clúster cuando se llama. Si no hay datos para pasar a la salida, la serie está en blanco.

Para determinar el valor de este atributo, utilice el selector CACLWD con la llamada MQINQ.

ClusterWorkloadExit (cadena de caracteres de 20 bytes) en IBM i

Nombre de salida de usuario para la gestión de carga de trabajo de clúster.

Si este nombre no está en blanco, se llama a la salida cada vez que se transfiere un mensaje a una cola de clúster o se mueve de una cola de clúster emisor a otra. A continuación, la salida puede aceptar la instancia de cola seleccionada por el gestor de colas como destino del mensaje o seleccionar otra instancia de cola.

Para determinar el valor de este atributo, utilice el selector CACLWX con la llamada MQINQ. La longitud de este atributo la proporciona LNEXN.

ClusterWorkloadLength (entero con signo de 10 dígitos) en IBM i

Longitud máxima de datos de mensaje pasados a la salida de carga de trabajo de clúster.

Es la longitud máxima de los datos de mensaje que se pasan a la salida de carga de trabajo del clúster. La longitud real de los datos pasados a la salida es el mínimo de lo siguiente:

- Longitud del mensaje.
- El atributo **MaxMsgLength** del gestor de colas.
- El atributo **ClusterWorkloadLength** .

Para determinar el valor de este atributo, utilice el selector IACLWL con la llamada MQINQ.

CodedCharSetId (entero con signo de 10 dígitos) en IBM i

Identificador de juego de caracteres codificados.

Esto define el juego de caracteres que utiliza el gestor de colas para todos los campos de cadena de caracteres definidos en el directorio de objetos (MQI), como los nombres de los objetos y la fecha y hora de creación de las colas. El juego de caracteres debe ser uno que tenga caracteres de un solo byte para los caracteres que son válidos en los nombres de objeto. No se aplica a los datos de aplicación transportados en el mensaje. El valor depende del entorno:

- En IBM i, el valor es el que se establece en el entorno cuando se crea por primera vez el gestor de colas.

Para determinar el valor de este atributo, utilice el selector IACCSI con la llamada MQINQ.

► IBM i **CommandEvent (entero) en IBM i**

Controla si los mensajes se colocan en una cola local cuando se emiten mandatos.

Esto controla si los mensajes se graban en una cola de sucesos nueva, SYSTEM.ADMIN.COMMAND.EVENT, siempre que se emitan mandatos. Esta característica es útil para la notificación de seguimiento de mandatos y para el diagnóstico de problemas. Para consultar sobre el atributo de gestor de colas CommandEvent, utilice el nuevo selector de atributos iacev con uno de los valores siguientes:

- EVRENA-los mensajes de suceso de mandato se generan y se colocan en la cola para todos los mandatos satisfactorios.
- Los mensajes de suceso de mandato EVND se generan y se colocan en la cola para todos los mandatos satisfactorios que no sean el mandato DISPLAY (MQSC) y el mandato Inquire (PCF).
- EVRDIS-los mensajes de suceso de mandato no se generan ni se colocan en la cola (este es el valor predeterminado inicial del gestor de colas).

Para determinar el valor de este atributo, utilice el selector CMDEV con la llamada MQINQ.

► IBM i **CommandInputQName (cadena de caracteres de 48 bytes) en IBM i**

Nombre de cola de entrada de mandatos.

CommandInputQName es el nombre de la cola de entrada de comandos definida en el gestor de colas local. Es una cola a la que los usuarios pueden enviar mandatos, si están autorizados a hacerlo. El nombre de la cola depende del entorno:

- En IBM i, el nombre de la cola es SYSTEM.ADMIN.COMMAND.QUEUEy solo se le pueden enviar mandatos PCF. Sin embargo, se puede enviar un mandato MQSC a esta cola si el mandato MQSC está entre un mandato PCF de tipo CMESC. Para obtener más información sobre el mandato Escape, consulte [Escape](#).

Para determinar el valor de este atributo, utilice el selector CACMDQ con la llamada MQINQ. La longitud de este atributo la proporciona LNQN.

► IBM i **CommandLevel (entero con signo de 10 dígitos) en IBM i**

Nivel de comando. Esto indica el nivel de comandos de control del sistema que admite el gestor de colas.

El nivel es uno de los siguientes valores:

CML800

Nivel 800 de comandos de control del sistema.

Este valor es devuelto por las siguientes aplicaciones:

- IBM MQ for IBM i
 - versión 8.0

CML900

Nivel 900 de comandos de control del sistema.

Este valor es devuelto por las siguientes aplicaciones:

- IBM MQ for IBM i
 - Versión 9.0

CML910

Nivel 910 de comandos de control del sistema.

Este valor es devuelto por las siguientes aplicaciones:

- IBM MQ for IBM i
 - versión 9.1

CML920

Nivel 920 de comandos de control del sistema.

Este valor es devuelto por las siguientes aplicaciones:

- IBM MQ for IBM i
 - Versión 9.2

CML930

Nivel 930 de comandos de control del sistema.

Este valor es devuelto por las siguientes aplicaciones:

- IBM MQ for IBM i
 - Versión 9.3

El conjunto de comandos de control del sistema que corresponde a un valor concreto del **CommandLevel** atributo varía en función del valor del **Platform** atributo; ambos deben utilizarse para decidir qué comandos de control del sistema son compatibles.

Para determinar el valor de este atributo, utilice el selector IACMDL con la llamada MQINQ.

ConfigurationEvent en IBM i

Controla si los sucesos de configuración se generan y se envían a SYSTEM.ADMIN.CONFIG.EVENT EVENT.

El atributo ConfigurationEvent puede ser uno de los valores siguientes:

- EVRENA
- EVRDIS

Si el atributo ConfigurationEvent se establece en EVRENA y runmqsc o PCF emiten correctamente determinados mandatos, los sucesos de configuración se generan y se envían al SYSTEM.ADMIN.CONFIG.EVENT . Se emiten sucesos para los mandatos siguientes, incluso si un mandato alter no cambia el objeto implicado. Los mandatos para los que se generan y envían sucesos de configuración son:

- DEFINE/ALTER AUTHINFO
- DEFINE/ALTER CHANNEL
- DEFINE/ALTER NAMELIST
- DEFINE/ALTER PROCESS
- DEFINE/ALTER QLOCAL (a menos que sea una cola dinámica temporal)
- DEFINE/ALTER QMODEL/QALIAS/QREMOTE
- DELETE AUTHINFO
- DELETE CHANNEL
- DELETE NAMELIST
- DELETE PROCESS
- DELETE QLOCAL (a menos que sea una cola dinámica temporal)
- DELETE QMODEL/QALIAS/QREMOTE
- ALTER QMGR (a menos que el atributo CONFIGEV esté inhabilitado y no se cambie a habilitado)
- REFRESH QMGR
- Una llamada MQSET, que no sea para una cola dinámica temporal.

Los sucesos no se generan (si están habilitados) en las circunstancias siguientes:

- El mandato o la llamada MQSET falla.
- El gestor de colas no puede colocar el mensaje de suceso en la cola de sucesos. El mandato debería completarse correctamente.

- Colas dinámicas temporales.
- Cambios de atributos internos realizados directa o implícitamente (no mediante MQSET o mandato); esto afecta a TRIGGER, CURDEPTH, IPPROCS, OPPROCS, QDPHIEV, QDPLOEV, QDPMAXEV, QSVCI EV.
- Cuando se cambia la cola de sucesos de configuración, aunque se generará un mensaje de suceso para ese cambio cuando se solicite una renovación.
- Agrupación en clúster de cambios mediante los mandatos REFRESH/RESET CLUSTER y RESUME/SUSPEND QMGR.
- Creación o supresión de un gestor de colas.

DeadLetterQName (cadena de caracteres de 48 bytes) en IBM i

Nombre de la cola de mensajes no entregados.

Es el nombre de una cola definida en el gestor de colas local. Los mensajes se envían a esta cola si no pueden direccionarse a su destino correcto.

Por ejemplo, los mensajes se colocan en esta cola cuando:

- Llega un mensaje a un gestor de colas, destinado a una cola que todavía no está definida en ese gestor de colas
- Un mensaje llega a un gestor de colas, pero la cola a la que está destinado no puede recibirlo porque, posiblemente:
 - La cola está llena
 - Las solicitudes de colocación están inhibidas
 - El nodo emisor no tiene autorización para colocar mensajes en la cola

Las aplicaciones también pueden colocar mensajes en la cola de mensajes no entregados.

Los mensajes de informe se tratan de la misma forma que los mensajes ordinarios; si el mensaje de informe no se puede entregar a su cola de destino (normalmente la cola especificada por el campo *MDRQ* en el descriptor de mensaje del mensaje original), el mensaje de informe se coloca en la cola de mensajes no entregados (mensaje no entregado).

Nota: Mensajes que han pasado su tiempo de caducidad (consulte el campo *MDEXP* descrito en “MQMD (descriptor de mensaje) en IBM i” en la [página 1131](#)) **no** se transfieren a esta cola cuando se descartan. Sin embargo, todavía se genera un mensaje de informe de caducidad (ROEXP) y se envía a la cola *MDRQ*, si lo solicita la aplicación emisora.

Los mensajes no se colocan en la cola de mensajes no entregados cuando la aplicación que ha emitido la solicitud de colocación ha recibido una notificación síncrona del problema con el código de razón devuelto por la llamada MQPUT o MQPUT1 (por ejemplo, un mensaje colocado en una cola local para el que se inhiben las solicitudes de colocación).

Los mensajes de la cola de mensajes no entregados a veces tienen sus datos de mensaje de aplicación con el prefijo de una estructura MQDLH. Esta estructura contiene información adicional que indica por qué el mensaje se ha colocado en la cola de mensajes no entregados (undelivered-message). Consulte “MQDLH (Encabezado de carta muerta) en IBM i” en la [página 1084](#) para obtener más detalles de esta estructura.

Esta cola debe ser una cola local, con un atributo **Usage** de USNORM.

Si un gestor de colas no soporta una cola de mensajes no entregados (undelivered-message) o no se ha definido una, el nombre estará en blanco. Todos los gestores de colas de IBM MQ dan soporte a una cola de mensajes no entregados (undelivered-message), pero de forma predeterminada no está definida.

Si la cola de mensajes no entregados (undelivered-message) no está definida, o está llena, o inutilizable por alguna otra razón, un mensaje que un agente de canal de mensajes le habría transferido se retiene en su lugar en la cola de transmisión.

Para determinar el valor de este atributo, utilice el selector CADLQ con la llamada MQINQ. La longitud de este atributo la proporciona LNQN.

DefClusterXmitQueueType (entero con signo de 10 dígitos)

El atributo « DefClusterXmitQueueType » controla qué cola de transmisión seleccionan por defecto los canales «cluster-sender» para recibir mensajes y enviarlos a los canales «cluster-receiver».

Los valores de **DefClusterXmitQueueType** son MQCLXQ_SCTQ o MQCLXQ_CHANNEL.

MQCLXQ_SCTQ

Todos los canales de clúster emisor envían mensajes de SYSTEM.CLUSTER.TRANSMIT.QUEUE. El correlID de los mensajes colocados en la cola de transmisión identifica el canal de clúster emisor al que va destinado el mensaje.

La variable SCTQ se establece al definir un gestor de colas.

MQCLXQ_CHANNEL

Cada canal de clúster emisor envía mensajes desde una cola de transmisión diferente.
Cada cola de transmisión se crea como una cola dinámica permanente de la cola modelo SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE.

Si el atributo del gestor de colas, DefClusterXmitQueueType, está establecido en CHANNEL, la configuración predeterminada cambia a los canales de clúster emisor que se están asociando con colas de transmisión de clúster individuales. SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUELas colas de transmisión son colas dinámicas permanentes creadas a partir de la cola modelo. Cada cola de transmisión está asociada a un canal de clúster emisor. Cuando un canal de clúster emisor presta servicio a una cola de transmisión de clúster, la cola de transmisión contiene mensajes únicamente para un gestor de colas de un clúster. Puede configurar clústeres de modo que cada gestor de colas de un clúster sólo contenga una cola de clúster. En este caso, el tráfico de mensajes de un gestor de colas a cada cola de clúster se transfiere por separado de los mensajes a otras colas.

Para consultar el valor, llame a MQINQo envíe un mandato Consultar gestor de colas (MQCMD_INQUIRE_Q_MGR) PCF, estableciendo el selector MQIA_DEF_CLUSTER_XMIT_Q_TYPE . Para cambiar el valor, envíe un mandato PCF Cambiar gestor de colas (MQCMD_CHANGE_Q_MGR), estableciendo el selector MQIA_DEF_CLUSTER_XMIT_Q_TYPE .

Referencia relacionada

[Cambiar gestor de colas](#)

[Consultar gestor de colas](#)

“MQINQ (Consultar atributos de objeto) en IBM i” en la página 1335

La llamada MQINQ devuelve una matriz de números enteros y un conjunto de cadenas de caracteres que contienen los atributos de un objeto.

IBM i DefXmitQName (cadena de caracteres de 48 bytes) en IBM i

Nombre de cola de transmisión predeterminado.

Es el nombre de la cola de transmisión que se utiliza para la transmisión de mensajes a gestores de colas remotos, si no hay ninguna otra indicación de qué cola de transmisión utilizar.

Si no hay ninguna cola de transmisión predeterminada, el nombre está totalmente en blanco. El valor inicial de este atributo está en blanco.

Para determinar el valor de este atributo, utilice el selector CADXQN con la llamada MQINQ. La longitud de este atributo la proporciona LNQN.

IBM i DistLists (entero con signo de 10 dígitos) en IBM i

Soporte de lista de distribución.

Esto indica si el gestor de colas local da soporte a listas de distribución en las llamadas MQPUT y MQPUT1 . Puede tener uno de los valores siguientes:

DLSUPP

Listas de distribución soportadas.

DLNSUP

Listas de distribución no soportadas.

Para determinar el valor de este atributo, utilice el selector IADIST con la llamada MQINQ.

InhibitEvent (entero con signo de 10 dígitos) en IBM i

Controla si se generan sucesos de inhibición (inhibir obtención e inhibir colocación).

Puede tener uno de los valores siguientes:

EVRDIS

Informes de sucesos inhabilitados.

EVRENA

Informes de sucesos habilitados.

Para obtener más información sobre los sucesos, consulte [Supervisión y rendimiento](#).

Para determinar el valor de este atributo, utilice el selector IAINHE con la llamada MQINQ.

LocalEvent (entero con signo de 10 dígitos) en IBM i

Controla si se generan sucesos de error locales.

El valor puede ser uno de los siguientes:

EVRDIS

Informes de sucesos inhabilitados.

EVRENA

Informes de sucesos habilitados.

Para obtener más información sobre los sucesos, consulte [Supervisión de sucesos](#).

Para determinar el valor de este atributo, utilice el selector IALCLE con la llamada MQINQ.

LoggerEvent (entero con signo de 10 dígitos) en IBM i

Controla si se generan sucesos de registrador de recuperación.

Puede tener uno de los valores siguientes:

ENABLED

Se generan sucesos de registrador.

DISABLED

Los sucesos de registrador no se generan. Este es el valor predeterminado inicial de los gestores de colas.

Para obtener más información sobre los sucesos, consulte [Supervisión y rendimiento](#).

MaxHandles (entero con signo de 10 dígitos) en IBM i

Número máximo de descriptores de contexto.

Es el número máximo de descriptores de contexto abiertos que cualquier tarea puede utilizar simultáneamente. Cada llamada MQOPEN satisfactoria para una sola cola (o para un objeto que no es una cola) utiliza un descriptor de contexto. Ese descriptor de contexto pasa a estar disponible para su reutilización cuando se cierra el objeto. Sin embargo, cuando se abre una lista de distribución, a cada cola de la lista de distribución se le asigna un descriptor de contexto independiente, y de modo que la llamada MQOPEN utiliza tantos descriptores de contexto como colas hay en la lista de distribución. Esto debe tenerse en cuenta al decidir un valor adecuado para *MaxHandles*.

La llamada MQPUT1 realiza una llamada MQOPEN como parte de su proceso; como resultado, MQPUT1 utiliza tantos manejadores como MQOPEN lo haría, pero los manejadores sólo se utilizan durante la propia llamada MQPUT1.

El valor está en el rango de 1 a 999.999.999. En IBM i, el valor predeterminado es 256.

Para determinar el valor de este atributo, utilice el selector IAMHND con la llamada MQINQ.

MaxMsgLength (entero con signo de 10 dígitos) en IBM i

Longitud máxima de mensaje en bytes.

Es la longitud del mensaje *físico* más largo que puede manejar el gestor de colas. Sin embargo, debido a que el atributo de gestor de colas **MaxMsgLength** se puede establecer independientemente del atributo de cola **MaxMsgLength**, el mensaje físico más largo que se puede colocar en una cola es el menor de estos dos valores.

Si el gestor de colas da soporte a la segmentación, es posible que una aplicación coloque un mensaje *lógico* que sea más largo que el menor de los dos atributos **MaxMsgLength**, pero sólo si la aplicación especifica el distintivo MFSEGA en MQMD. Si se especifica ese distintivo, el límite superior para la longitud de un mensaje lógico es de 999.999.999 bytes, pero normalmente, las restricciones de recursos impuestas por el sistema operativo o por el entorno en el que se ejecuta la aplicación darán como resultado un límite inferior.

El límite inferior para el atributo **MaxMsgLength** es de 32 KB (32.768 bytes). En IBM i, la longitud máxima del mensaje es de 100 MB (104 857 600 bytes).

Para determinar el valor de este atributo, utilice el selector IAMLEN con la llamada MQINQ.

MaxPriority (entero con signo de 10 dígitos) en IBM i

Prioridad máxima.

Esta es la prioridad de mensaje máxima soportada por el gestor de colas. Las prioridades van de cero (más bajo) a *MaxPriority* (más alto).

Para determinar el valor de este atributo, utilice el selector IAMPRI con la llamada MQINQ.

MaxUncommittedMsgs (entero con signo de 10 dígitos) en IBM i

Número máximo de mensajes no confirmados dentro de una unidad de trabajo.

Es el número máximo de mensajes no confirmados que pueden existir en una unidad de trabajo. El número de mensajes no confirmados es la suma de los siguientes mensajes desde el inicio de la unidad de trabajo actual:

- Mensajes colocados por la aplicación con la opción PMSYP
- Mensajes recuperados por la aplicación con la opción GMSYP
- Mensajes desencadenantes y mensajes de informe COA generados por el gestor de colas para mensajes colocados con la opción PMSYP
- Mensajes de informe COD generados por el gestor de colas para mensajes recuperados con la opción GMSYP

Los mensajes siguientes no se cuentan como no confirmados:

- Mensajes colocados o recuperados por la aplicación fuera de una unidad de trabajo
- Mensajes desencadenantes o mensajes de informe COA/COD generados por el gestor de colas como resultado de mensajes colocados o recuperados fuera de una unidad de trabajo
- Mensajes de informe de caducidad generados por el gestor de colas (aunque la llamada provoque el mensaje de informe de caducidad especificado GMSYP)
- Mensajes de suceso generados por el gestor de colas (aunque la llamada provoque el mensaje de suceso especificado PMSYP o GMSYP)

Nota:

1. Los mensajes de informe de excepción los genera el agente de canal de mensajes (MCA) o la aplicación, por lo que se tratan de la misma forma que los mensajes ordinarios colocados o recuperados por la aplicación.

2. Cuando se coloca un mensaje o segmento con la opción PMSYP, el número de mensajes no confirmados se incrementa en uno independientemente de cuántos mensajes físicos resulten realmente de la colocación. (Puede producirse más de un mensaje físico si el gestor de colas necesita subdividir el mensaje o segmento.)
3. Cuando se coloca una lista de distribución con la opción PMSYP, el número de mensajes no confirmados se incrementa en un *para cada mensaje físico que se genera*. Esto puede ser tan pequeño como uno, o tan grande como el número de destinos en la lista de distribución.

El límite inferior para este atributo es 1; el límite superior es 999 999 999.

Para determinar el valor de este atributo, utilice el selector IAMUNC con la llamada MQINQ.

PerformanceEvent (entero con signo de 10 dígitos) en IBM i

Controla si se generan sucesos relacionados con el rendimiento.

PerformanceEvent puede tener uno de los valores siguientes:

EVRDIS

Informes de sucesos inhabilitados.

EVRENA

Informes de sucesos habilitados.

Para obtener más información sobre los sucesos, consulte [Supervisión de sucesos](#).

Para determinar el valor de este atributo, utilice el selector IAPFME con la llamada MQINQ.

Plataforma (entero con signo de 10 dígitos) en IBM i

Plataforma en la que se ejecuta el gestor de colas.

Indica el sistema operativo en el que se ejecuta el gestor de colas. El valor es:

PL400

IBM i.

PubSubMode (entero con signo de 10 dígitos) en IBM i

Si el motor de publicación/suscripción y la interfaz de publicación/suscripción en cola se están ejecutando, permitiendo por lo tanto que las aplicaciones publiquen/suscriban utilizando la interfaz de programación de aplicaciones y las colas que están siendo supervisadas por la interfaz de publicación/suscripción en cola.

Puede tener uno de los valores siguientes:

PSMCP

El motor de publicación/suscripción está ejecutándose. Por lo tanto, es posible publicar/suscribirse utilizando la interfaz de programación de aplicaciones. La interfaz de publicación/suscripción en cola no se está ejecutando, por lo tanto, no se actúa sobre ningún mensaje que se coloque en las colas supervisadas por la interfaz de publicación/suscripción en cola. Este valor se utiliza para la compatibilidad con WebSphere Message Broker V6 o versiones anteriores que utilizan este gestor de colas, porque debe leer las mismas colas de las que normalmente lee la interfaz de publicación/suscripción en cola.

PSMDS

El motor de publicación/suscripción y la interfaz de publicación/suscripción en cola no están ejecutándose. Por lo tanto, no es posible publicar/suscribirse utilizando la interfaz de programación de aplicaciones. No se actúa sobre los mensajes de publicación/suscripción que se colocan en las colas supervisadas por la interfaz de publicación/suscripción en cola.

PAMÉN

El motor de publicación/suscripción y la interfaz de publicación/suscripción en cola están ejecutándose. Por lo tanto, es posible publicar/suscribirse utilizando la interfaz de programación de aplicaciones y las colas supervisadas por la interfaz de publicación/suscripción en cola. Este es el valor predeterminado inicial del gestor de colas.

Para determinar el valor de este atributo, utilice el selector PSMODE con la llamada MQINQ.

QMGrDesc (serie de caracteres de 64 bytes) en IBM i

Descripción del gestor de colas.

Este es un campo que se puede utilizar para comentarios descriptivos. El contenido del campo no es significativo para el gestor de colas, pero es posible que el gestor de colas requiera que el campo contenga sólo caracteres que se puedan visualizar. No puede contener ningún carácter nulo; si es necesario, se rellena a la derecha con espacios en blanco. En una instalación DBCS, este campo puede contener caracteres DBCS (sujetos a una longitud máxima de campo de 64 bytes).

Nota: Si este campo contiene caracteres que no están en el juego de caracteres del gestor de colas (tal como lo define el atributo de gestor de colas **CodedCharSetId**), estos caracteres podrían traducirse incorrectamente si este campo se envía a otro gestor de colas.

En IBM i, el valor por omisión es blancos.

Para determinar el valor de este atributo, utilice el selector CAQMD con la llamada MQINQ. La longitud de este atributo la proporciona LNQMD.

QMGrIdentifier (serie de caracteres de 48 bytes) en IBM i

Identificador exclusivo generado internamente del gestor de colas.

Es un nombre exclusivo generado internamente para el gestor de colas.

Para determinar el valor de este atributo, utilice el selector CAQMID con la llamada MQINQ. La longitud de este atributo la proporciona LNQMID.

QMGrName (serie de caracteres de 48 bytes) en IBM i

Nombre del gestor de colas.

Es el nombre del gestor de colas local, es decir, el nombre del gestor de colas al que está conectada la aplicación.

Los primeros 12 caracteres del nombre se utilizan para construir un identificador de mensaje exclusivo (consulte el campo *MDMID* descrito en “MQMD (descriptor de mensaje) en IBM i” en la página 1131). Por lo tanto, los gestores de colas que pueden intercomunicarse deben tener nombres que difieran en los primeros 12 caracteres, para que los identificadores de mensajes sean exclusivos en la red de gestores de colas.

Para determinar el valor de este atributo, utilice el selector CAQMN con la llamada MQINQ. La longitud de este atributo la proporciona LNQMN.

RemoteEvent (entero con signo de 10 dígitos) en IBM i

Controla si se generan sucesos de error remotos.

El valor puede ser uno de los siguientes:

EVRDIS

Informes de sucesos inhabilitados.

EVRENA

Informes de sucesos habilitados.

Para obtener más información sobre los sucesos, consulte [Supervisión de sucesos](#).

Para determinar el valor de este atributo, utilice el selector IARMTE con la llamada MQINQ.

RepositoryName (serie de caracteres de 48 bytes) en IBM i

Nombre del clúster para el que este gestor de colas proporciona servicios de repositorio.

Es el nombre de un clúster para el que este gestor de colas proporciona un servicio de gestor de repositorios. Si el gestor de colas proporciona este servicio para más de un clúster, *RepositoryNameList* especifica el nombre de un objeto de lista de nombres que identifica

los clústeres y *RepositoryName* está en blanco. Al menos uno de *RepositoryName* y *RepositoryNameList* debe estar en blanco.

Para determinar el valor de este atributo, utilice el selector CARPN con la llamada MQINQ. La longitud de este atributo la proporciona LNQM.

IBM i *RepositoryNameList (serie de caracteres de 48 bytes) en IBM i*

Nombre del objeto de lista de nombres que contiene nombres de clústeres para los que este gestor de colas proporciona servicios de repositorio.

Es el nombre de un objeto de lista de nombres que contiene los nombres de los clústeres para los que este gestor de colas proporciona un servicio de gestor de repositorios. Si el gestor de colas proporciona este servicio sólo para un clúster, el objeto de lista de nombres sólo contiene un nombre. De forma alternativa, se puede utilizar *RepositoryName* para especificar el nombre del clúster, en cuyo caso *RepositoryNameList* está en blanco. Al menos uno de *RepositoryName* y *RepositoryNameList* debe estar en blanco.

Para determinar el valor de este atributo, utilice el selector CARPNL con la llamada MQINQ. La longitud de este atributo la proporciona LNNL.

IBM i *SSLEvent (serie de caracteres) en IBM i*

Determina si se generan eventos de tipo « TLS ».

El valor puede ser uno de los siguientes:

- EVRENA (MQINQ/PCF/evento de configuración) ACTIVADO (MQSC): se generan eventos « TLS » (es decir, se genera el evento « RC2371 »).
- EVRDIS (MQINQ/PCF/evento de configuración) DESACTIVADO (MQSC): no se generan eventos « TLS ». Este es el valor predeterminado inicial del gestor de colas.

Para determinar el valor de este atributo, utilice el selector IASSLE con la llamada MQINQ.

IBM i *SSLKeyResetCount (entero) en IBM i*

Determina el número total de bytes no cifrados que se envían y reciben en una conversación de « TLS », antes de que se renegocie la clave secreta. El número de bytes incluye información de control enviada por el agente de canal de mensajes (MCA).

Este valor solo lo utilizan los MCA de canal de tipo « TLS » que inician la comunicación desde este gestor de colas (es decir, el MCA de canal emisor en un par de canales emisor y receptor).

Si el valor de este atributo es mayor que 0, y las pulsaciones de canal están habilitadas para un canal, la clave secreta también se renegocia antes de que se envíen o reciban los datos después de una pulsación de canal. El recuento de bytes hasta la siguiente renegociación de clave secreta se restablece después de cada renegociación satisfactoria.

El valor puede estar en el rango de 0 a 999.999.999. Un valor de 0 para este atributo indica que la clave secreta nunca se renegocia. Si especifica una cuenta de restablecimiento de clave secreta TLS entre 1 byte y 32 KB, los canales TLS utilizarán una cuenta de restablecimiento de clave secreta de 32 KB. El objetivo es evitar el coste de procesamiento que supondría un número excesivo de restablecimientos de claves, lo cual ocurriría con valores de restablecimiento de la clave secreta de TLS demasiado pequeños.

Cuando el servidor de SSL actúa como IBM MQ gestor de colas y están habilitados tanto el restablecimiento de la clave secreta como los latidos del canal, la renegociación se produce inmediatamente después de cada latido del canal.

Para determinar el valor de este atributo, utilice el selector IASSRC con la llamada MQINQ.

IBM i *StartStopEvent (entero con signo de 10 dígitos) en IBM i*

Controla si se generan sucesos de inicio y detención.

Este atributo puede tener uno de los siguientes valores:

EVRDIS

Informes de sucesos inhabilitados.

EVRENA

Informes de sucesos habilitados.

Para obtener más información sobre los sucesos, consulte [Supervisión de sucesos](#).

Para determinar el valor de este atributo, utilice el selector IASSE con la llamada MQINQ.

SyncPoint (entero con signo de 10 dígitos) en IBM i

Disponibilidad de punto de sincronismo.

Esto indica si el gestor de colas local soporta unidades de trabajo y sincronización con las llamadas MQGET, MQPUT y MQPUT1.

SAVL

Unidades de trabajo y sincronización disponibles.

SPNAVL

Unidades de trabajo y sincronización no disponibles.

Para determinar el valor de este atributo, utilice el selector IASYNC con la llamada MQINQ.

TraceRouteRecording (entero con signo de 10 dígitos) en IBM i

Esto controla si la información sobre los mensajes se registra a medida que fluyen a través de un gestor de colas.

El valor puede ser uno de los siguientes:

- RECD: no se permite ninguna adición para rastrear mensajes de ruta
- RECDQ: los mensajes se colocan en una cola con nombre fijo
- RECDM: determinar utilizando mensaje (este es el valor predeterminado inicial)

Para evitar que el mensaje de ruta de rastreo permanezca en el sistema, establezca un valor de caducidad que sea mayor que cero y especifique la opción de informe RODISC. Para evitar que los mensajes de informe o respuesta permanezcan en el sistema, establezca la opción de informe ROPDAE. Para obtener más información, consulte [“Opciones de informe y distintivos de mensaje en IBM i”](#) en la [página 1454](#).

Para determinar el valor de este atributo, utilice el selector IATRGI con la llamada MQINQ.

TreeLifeTime (entero con signo de 10 dígitos) en IBM i

El tiempo de vida, en segundos, de los temas no administrativos.

Los temas no administrativos son los que se crean cuando una aplicación publica en, o se suscribe como, una serie de tema que no existe como nodo administrativo. Cuando este nodo no administrativo ya no tiene ninguna suscripción activa, este parámetro determina cuánto tiempo esperará el gestor de colas antes de eliminar ese nodo. Sólo los temas no administrativos utilizados por una suscripción duradera permanecen después de que se reinicie el gestor de colas.

Especifique un valor en el rango de 0 a 604 000. El valor 0 significa que el gestor de colas no elimina los temas no administrativos. El valor predeterminado inicial del gestor de colas es 1800.

Para determinar el valor de este atributo, utilice el selector IATRLFT con la llamada MQINQ.

TriggerInterval (entero con signo de 10 dígitos) en IBM i

Intervalo de mensaje de desencadenante.

Es un intervalo de tiempo (en milisegundos) utilizado para restringir el número de mensajes desencadenantes. Esto sólo es relevante cuando *TriggerType* es TTFIRST. En este caso, los mensajes desencadenantes normalmente se generan sólo cuando llega un mensaje adecuado a la cola, y la cola estaba vacía anteriormente. En determinadas circunstancias, sin embargo, se puede generar un mensaje

desencadenante adicional con el desencadenamiento TTFRST incluso si la cola no estaba vacía. Estos mensajes desencadenantes adicionales no se generan con más frecuencia que cada *TriggerInterval* milisegundos.

Para obtener más información sobre el desencadenamiento, consulte [Desencadenamiento de canales](#).

El valor está en el rango de cero a 999.999.999. El valor predeterminado es 999.999.999.

Para determinar el valor de este atributo, utilice el selector IATRGI con la llamada MQINQ.

Códigos de retorno para IBM i (ILE RPG)

Esta información describe los códigos de respuesta asociados a la interfaz de programación de aplicaciones (MQI) y a MQAI.

Los códigos de retorno asociados con:

- Los mandatos PCF (Programmable Command Format) se listan en [Referencia de formatos de comandos programables](#).
- Las llamadas C++ se listan en [Usando C++](#).

Por cada llamada, el gestor de colas o una rutina de salida devuelven un código de terminación y un código de razón para indicar que la llamada se ha realizado de forma satisfactoria o bien con errores.

Las aplicaciones no deben depender de los errores que se comprueban en un orden específico, excepto cuando se indique específicamente. Si puede surgir más de un código de terminación o código de razón de una llamada, el error concreto notificado depende de la implementación.

Códigos de terminación para IBM i (ILE RPG)

El parámetro de código de terminación (*CMPCOD*) permite al interlocutor ver rápidamente si la llamada se ha completado correctamente, se ha completado parcialmente o ha fallado.

CCOK

(MQCC_OK en otras plataformas)

Realización satisfactoria.

La llamada se ha completado del todo; se han establecido todos los parámetros de salida. El parámetro **REASON** siempre tiene el valor RCNONE en este caso.

CCWARN

(MQCC_WARN en otras plataformas)

Aviso (finalización parcial).

La llamada se ha completado parcialmente. Es posible que algunos parámetros de salida se hayan establecido además de los parámetros de salida *CMPCOD* y *REASON*. El parámetro **REASON** proporciona información adicional sobre la finalización parcial.

CCFAIL

(MQCC_FAIL en otras plataformas)

La llamada no se ha realizado satisfactoriamente.

El procesamiento de la llamada no se ha completado y el estado del gestor de colas no suele variar; se indican excepciones concretas. Se han establecido los parámetros de salida *CMPCOD* y *REASON*; otros parámetros no se modifican, excepto cuando se indique lo contrario.

La razón puede ser un error en el programa de aplicación, o puede ser un resultado de alguna situación externa al programa, por ejemplo, es posible que se haya revocado la autorización del usuario. El parámetro **REASON** proporciona información adicional sobre el error.

Códigos de razón para IBM i (ILE RPG)

El parámetro de código de razón (*REASON*) es una calificación para el parámetro de código de terminación (*CMPCOD*).

Si no hay ninguna razón especial para informar, se devuelve RCNONE. Una llamada satisfactoria devuelve CCOK y RCNONE.

Si el código de terminación es CCWARN o CCFAIL, el gestor de colas siempre informa de una razón de calificación; los detalles se proporcionan bajo cada descripción de llamada.

Allá donde las salidas de usuario establezcan códigos de terminación y de razón, habrán de atenerse a estas reglas. Además, los valores de razón especiales definidos por salidas de usuario han de ser menores que cero para garantizar que no entran en conflicto con los valores definidos por el gestor de colas. Las salidas pueden establecer razones ya definidas por el gestor de colas cuando procedan.

También aparecen códigos de razón en:

- El campo *DLREA* de la estructura MQDLH
- El campo *MDFB* de la estructura MQMD

Para obtener la lista completa de códigos de razón, consulte [Códigos de terminación y razón de la API](#).

Para encontrar el código de razón IBM i en esa lista, elimine el "RC" de la parte frontal, por ejemplo, RC2002 pasa a ser 2002. También los códigos de terminación allí se muestran como están en otras plataformas:

Tabla 850. Nombres de código de razón en IBM i y en otras plataformas	
IBM i	Otras plataformas
CCOK	MQCC_OK
CCWARN	MQCC_ADVERTENCIA
CCFAIL	MQCC_FAIL

Normas para validar las opciones de « MQI » en IBM i (ILE RPG)

Este tema proporciona información sobre las situaciones que generan un código de razón RC2046 a partir de una llamada MQOPEN, MQPUT, MQPUT1, MQGET o MQCLOSE.

Llamada MQOPEN en IBM i

Para las opciones de la llamada MQOPEN:

- Se debe especificar *Al menos uno* de los siguientes:
 - OOBROW
 - OOINPQ
 - OOINPX
 - OOINPS
 - OOINQ
 - OOOUT
 - OOSSET
- Solo se permite *uno* de los siguientes:
 - OOINPQ
 - OOINPX
 - OOINPS

- Solo se permite *uno* de los siguientes:

- OOBNDO
- OOBNDN
- OOBNDQ

Nota: Las opciones listadas anteriormente son mutuamente excluyentes. Sin embargo, debido a que el valor de OOBNDQ es cero, si se especifica con cualquiera de las otras dos opciones de enlace no se genera el código de razón RC2046. Se proporciona OOBNDQ para ayudar a la documentación del programa.

- Si se especifica OOSAVA, también debe especificarse una de las opciones OOINP*.
- Si se especifica una de las opciones OOSSET* u OOPAS*, también debe especificarse OOSOUT.

Llamada MQPUT en IBM i

Para las opciones de colocación de mensajes:

- La combinación de PMSYP y PMNSYP no está permitida.
- Solo se permite *uno* de los siguientes:
 - PMDEFC
 - PNOC
 - PMPASA
 - PMPASI
 - PMSETA
 - PMSETI
- PMALTU no está permitido (sólo es válido en la llamada MQPUT1).

Llamada MQPUT1 en IBM i

Para las opciones de transferencia de mensajes, las reglas son las mismas que para la llamada MQPUT, excepto para las opciones siguientes:

- PMALTU está permitido.
- PMLOGO no está permitido.

Llamada MQGET en IBM i

Para las opciones get-message:

- Solo se permite *una* de las opciones siguientes:
 - GMNSYP
 - GMSYP
 - GMPSYP
- Solo se permite *una* de las opciones siguientes:
 - GMBRWF
 - GMBRWC
 - GMBRWN
 - GMMUC
- GMSYP no está permitido con ninguna de las opciones siguientes:
 - GMBRWF
 - GMBRWC

- GMBRWN
- GMLK
- GMUNLK
- GMPSTP no está permitido con ninguna de las opciones siguientes:
 - GMBRWF
 - GMBRWC
 - GMBRWN
 - GMCMPM
 - GMUNLK
- Si se especifica GMLK, también debe especificarse una de las opciones siguientes:
 - GMBRWF
 - GMBRWC
 - GMBRWN
- Si se especifica GMUNLK, solo se permiten las opciones siguientes:
 - GMNSTP
 - GMNWT

Llamada MQCLOSE en IBM i

- Para las opciones de la llamada MQCLOSE. La combinación de CODEL y COPURG no está permitida.
- Sólo se permite uno de los siguientes:
 - COKPSB
 - CORMSB

Llamada MQSUB en IBM i

Para las opciones de la llamada MQSUB:

- Se debe especificar al menos uno de los siguientes:
 - Se debe especificar al menos uno de los siguientes:
 - SOALT
 - Llagas
 - SOCRT
 - Sólo se permite uno de los siguientes:
 - SODUR
 - SONDUR
- Nota:** Las opciones listadas anteriormente son mutuamente excluyentes. Sin embargo, como el valor de SONDUR es cero, si se especifica con SODUR no se genera el código de razón RC2046. SONDUR se proporciona para ayudar a la documentación del programa.
- La combinación de SOGRP y SOMAN no está permitida.
 - SOGRP requiere que se especifique SOSCID.
 - Solo se permite uno de los siguientes: SOAUID SOFUID
 - La combinación de SONEWP y SOPUBR no está permitida.
 - SONEWP solo está permitido en combinación con SOCRT.
 - Sólo se permite uno de los siguientes:
 - SOCHR

Codificaciones de máquina en IBM i

Utilice esta información para obtener información sobre la estructura del campo *MDENC* en el descriptor de mensaje.

Para obtener más información sobre el descriptor de mensaje, consulte [“MQMD \(descriptor de mensaje\) en IBM i”](#) en la página 1131.

El campo *MDENC* es un entero de 32 bits que se divide en cuatro subcampos separados; estos subcampos identifican:

- La codificación utilizada para enteros binarios
- La codificación utilizada para enteros de decimal empaquetado
- La codificación utilizada para números de coma flotante
- Bits reservados

Cada subcampo es identificado por una máscara de bit que tiene 1-bits en las posiciones correspondientes al subcampo, y 0-bits en otra parte. Los bits están numerados de tal manera que el bit 0 es el bit más significativo, y el bit 31 el bit menos significativo. Se definen las máscaras siguientes:

ESIMSK

Máscara para codificación de enteros binarios.

Este subcampo ocupa las posiciones de bits 28 a 31 dentro del campo *MDENC*.

ENDMSK

Máscara para codificación de enteros decimales empaquetados.

Este subcampo ocupa las posiciones de bits 24 a 27 dentro del campo *MDENC*.

ENFMSK

Máscara para codificación de coma flotante.

Este subcampo ocupa las posiciones de bits 20 a 23 dentro del campo *MDENC*.

ENRMSK

Máscara para bits reservados.

Este subcampo ocupa las posiciones de bits 0 a 19 dentro del campo *MDENC*.

Codificación de enteros binarios en IBM i

Valores válidos para la codificación de enteros binarios.

Los valores siguientes son válidos para la codificación de entero binario:

ENIUND

Codificación de enteros no definida.

Los enteros binarios se representan utilizando una codificación que no está definida.

ENINOR

Codificación de enteros normal.

Los enteros binarios se representan de la forma convencional:

- El byte menos significativo del número tiene la dirección más alta de cualquiera de los bytes del número; el byte más significativo tiene la dirección más baja.
- El bit menos significativo de cada byte está junto al byte con la siguiente dirección superior; el bit más significativo de cada byte está junto al byte con la siguiente dirección inferior.

ENIREV

Codificación de enteros invertida.

Los enteros binarios se representan de la misma forma que ENINOR, pero con los bytes ordenados en orden inverso. Los bits dentro de cada byte se disponen de la misma forma que ENINOR.

Codificación de enteros decimales empaquetados en IBM i

Valores válidos para la codificación de enteros decimales empaquetados

Los valores siguientes son válidos para la codificación de enteros decimales empaquetados:

FIN

Codificación de decimal empaquetado no definida.

Los enteros de decimal empaquetado se representan utilizando una codificación que no está definida.

ENDNOR

Codificación de decimal empaquetado normal.

Los enteros decimales empaquetados se representan de la forma convencional:

- Cada dígito decimal en la forma imprimible del número se representa en decimal empaquetado por un dígito hexadecimal único en el rango de X' 0 'a X' 9'. Cada dígito hexadecimal ocupa 4 bits, por lo que cada byte del número decimal empaquetado representa dos dígitos decimales en el formato imprimible del número.
- El byte menos significativo en el número decimal empaquetado es el byte que contiene el dígito decimal menos significativo. Dentro de ese byte, los 4 bits más significativos contienen el dígito decimal menos significativo, y los 4 bits menos significativos contienen el signo. El signo es X'C '(positivo), X'D' (negativo) o X'F ' (sin signo).
- El byte menos significativo del número tiene la dirección más alta de cualquiera de los bytes del número; el byte más significativo tiene la dirección más baja.
- El bit menos significativo de cada byte está junto al byte con la siguiente dirección superior; el bit más significativo de cada byte está junto al byte con la siguiente dirección inferior.

ENDREV

Codificación de decimal empaquetado invertida.

Los enteros decimales empaquetados se representan de la misma forma que ENDNOR, pero con los bytes ordenados en orden inverso. Los bits dentro de cada byte se disponen de la misma forma que ENDNOR.

Codificación de coma flotante en IBM i

Valores válidos para la codificación de coma flotante

Los valores siguientes son válidos para la codificación de coma flotante:

ENFUNDAR

Codificación de coma flotante no definida.

Los números de coma flotante se representan utilizando una codificación que no está definida.

ENFNOR

Codificación normal de flotador IEEE (Instituto de Ingenieros Eléctricos y Electrónicos).

Los números de coma flotante se representan utilizando el formato de coma flotante IEEE estándar, con los bytes ordenados de la siguiente manera:

- El byte menos significativo de la mantisa tiene la dirección más alta de cualquiera de los bytes del número; el byte que contiene el exponente tiene la dirección más baja
- El bit menos significativo de cada byte está junto al byte con la siguiente dirección superior; el bit más significativo de cada byte está junto al byte con la siguiente dirección inferior

Los detalles de la codificación flotante IEEE pueden encontrarse en el estándar IEEE 754.

ENFREV

Codificación flotante IEEE invertida.

Los números de coma flotante se representan de la misma forma que ENFNOR, pero con los bytes ordenados en orden inverso. Los bits dentro de cada byte se disponen de la misma forma que ENFNOR.

ENF390

Codificación flotante de la arquitectura System/390.

Los números de coma flotante se representan utilizando el formato de coma flotante estándar System/390 ; también lo utiliza System/370.

IBM i Construcción de codificaciones en IBM i

Para construir un valor para el campo *MDENC* en MQMD, se deben añadir las constantes relevantes que describen las codificaciones necesarias.

Asegúrese de combinar sólo una de las codificaciones ENI* con una de las codificaciones END* y una de las codificaciones ENF*.

IBM i Análisis de codificaciones en IBM i

El campo *MDENC* contiene subcampos; debido a esto, las aplicaciones que necesitan examinar la codificación entera, decimal empaquetado o flotante deben utilizar la técnica descrita en este tema.

Utilización aritmética

Los pasos siguientes se deben realizar utilizando aritmética de enteros:

1. Seleccione uno de los valores siguientes, según el tipo de codificación necesario:

- 1 para la codificación de enteros binarios
- 16 para la codificación de enteros decimales empaquetados
- 256 para la codificación de coma flotante

Llame al valor A.

2. Divida el valor del campo *MDENC* por A ; llamar al resultado B.

3. Divida B por 16; llame al resultado C.

4. Multiplique C por 16 y reste de B ; llamar al resultado D.

5. Multiplique D por A ; llamar al resultado E.

6. E es la codificación necesaria y se puede probar la igualdad con cada uno de los valores válidos para ese tipo de codificación.

IBM i Resumen de codificaciones de arquitectura de máquina en IBM i

Una tabla que resume las codificaciones para arquitecturas de máquina.

Las codificaciones para arquitecturas de máquina se muestran en [Tabla 851 en la página 1453](#).

Tabla 851. Resumen de codificaciones para arquitecturas de máquina			
Arquitectura de máquina	Codificación de enteros binarios	Codificación de enteros de decimal empaquetado	Codificación de coma flotante
IBM i	normal	normal	IEEE estándar
Intel x86	reversed	reversed	IEEE invertido

Tabla 851. Resumen de codificaciones para arquitecturas de máquina (continuación)

Arquitectura de máquina	Codificación de enteros binarios	Codificación de enteros de decimal empaquetado	Codificación de coma flotante
PowerPC	normal	normal	IEEE estándar
System/390	normal	normal	System/390



Opciones de informe y distintivos de mensaje en IBM i

Este tema se refiere a los campos *MDREP* y *MDMFL* que forman parte del descriptor de mensaje MQMD especificado en las llamadas MQGET, MQPUT y MQPUT1.

Para obtener más información sobre el descriptor de mensaje, consulte [“MQMD \(descriptor de mensaje\) en IBM i”](#) en la página 1131. Esta información describe:

- La estructura del campo de informe y cómo lo procesa el gestor de colas
- Cómo debe analizar una aplicación el campo de informe
- La estructura del campo de distintivos de mensaje

Estructura del campo de informe

El campo *MDREP* es un entero de 32 bits que se divide en tres subcampos separados.

Estos subcampos identifican:

- Opciones de informe que se rechazan si el gestor de colas local no las reconoce
- Opciones de informe que siempre se aceptan, incluso si el gestor de colas local no las reconoce
- Opciones de informe que sólo se aceptan si se cumplen otras condiciones

Cada subcampo es identificado por una máscara de bit que tiene 1-bits en las posiciones correspondientes al subcampo, y 0-bits en otra parte. Tenga en cuenta que los bits de un subcampo no son necesariamente adyacentes. Los bits están numerados de tal manera que el bit 0 es el bit más significativo, y el bit 31 el bit menos significativo. Las máscaras siguientes están definidas para identificar los subcampos:

RORO

Máscara para opciones de informe no soportadas que se rechazan.

Esta máscara identifica las posiciones de bits dentro del campo *MDREP* donde las opciones de informe que no están soportadas por el gestor de colas local harán que la llamada MQPUT o MQPUT1 falle con el código de terminación CCFAIL y el código de razón RC2061.

Este subcampo ocupa las posiciones de bits 3 y 11 a 13.

ROAUM

Enmascarar las opciones de informe no soportadas que se aceptan.

Esta máscara identifica las posiciones de bits dentro del campo *MDREP* donde las opciones de informe que no están soportadas por el gestor de colas local se aceptarán en las llamadas MQPUT o MQPUT1. En este caso, se devuelve el código de terminación CCWARN con el código de razón RC2104.

Este subcampo ocupa las posiciones de bits 0 a 2, 4 a 10 y 24 a 31.

En este subcampo se incluyen las siguientes opciones de informe:

- RCMTCT
- RODLQ
- DISCO
- ROEXC

- ROEXCD
- ROEXCF
- ROEXP
- ROEXPD
- ROEXPF
- RONAN
- RONMI
- RONONA
- ROPAN
- ROPCI
- ROPMI

ROAUXM

Enmascarar las opciones de informe no soportadas que sólo se aceptan en determinadas circunstancias.

Esta máscara identifica las posiciones de bits dentro del campo *MDREP* donde las opciones de informe que no están soportadas por el gestor de colas local se aceptarán en las llamadas MQPUT o MQPUT1 *siempre que* se cumplan las dos condiciones siguientes:

- El mensaje está destinado a un gestor de colas remoto.
- La aplicación no coloca el mensaje directamente en una cola de transmisión local (es decir, la cola identificada por los campos *ODMN* y *ODON* en el descriptor de objeto especificado en la llamada MQOPEN o MQPUT1 no es una cola de transmisión local).

El código de terminación CCWARN con el código de razón RC2104 se devuelve si se cumplen estas condiciones y CCFAIL con el código de razón RC2061 si no es así.

Este subcampo ocupa las posiciones de bits 14 a 23.

En este subcampo se incluyen las siguientes opciones de informe:

- ROCOA
- ROCOAD
- ROCOAF
- ROCOD
- ROCODD
- ROCODF

Si hay alguna opción especificada en el campo *MDREP* que el gestor de colas no reconoce, el gestor de colas comprueba cada subcampo a su vez utilizando la operación AND a nivel de bit para combinar el campo *MDREP* con la máscara para ese subcampo. Si el resultado de esa operación no es cero, se devuelven el código de terminación y los códigos de razón descritos anteriormente.

Si se devuelve CCWARN, no se define qué código de razón se devuelve si existen otras condiciones de aviso.

La posibilidad de especificar y tener opciones de informe aceptadas que no son reconocidas por el gestor de colas local es útil cuando es necesario enviar un mensaje con una opción de informe que será reconocida y procesada por un gestor de colas *remoto*.

Análisis del campo de informe en IBM i

El campo MDREP contiene subcampos. Debido a esto, algunas aplicaciones necesitan comprobar si el remitente del mensaje ha solicitado un informe determinado. Estas aplicaciones deben utilizar la técnica descrita en este tema.

Utilización aritmética

Los pasos siguientes se deben realizar utilizando aritmética de enteros:

1. Seleccione uno de los valores siguientes, de acuerdo con el tipo de informe que se va a comprobar:

- Informe ROCOA para COA
- Informe ROCOD para COD
- ROEXC para informe de excepción
- ROEXP para informe de caducidad

Llame al valor A.

2. Divida el campo *MDREP* por A ; llamar al resultado B.

3. Divida B por 8 ; llamar al resultado C.

4. Multiplique C por 8 y resta de B ; llamar al resultado D.

5. Multiplique D por A ; llamar al resultado E.

6. Pruebe E para la igualdad con cada uno de los valores posibles para ese tipo de informe.

Por ejemplo, si A es ROEXC, pruebe la igualdad de E con cada uno de los siguientes para determinar qué ha especificado el remitente del mensaje:

- RONONA
- ROEXC
- ROEXCD
- ROEXCF

Las pruebas se pueden realizar en el orden que sea más conveniente para la lógica de la aplicación.

El pseudocódigo siguiente ilustra esta técnica para los mensajes de informe de excepción:

```
A = ROEXC
B = Report/A
C = B/8
D = B - C*8
E = D*A
```

Se puede utilizar un método similar para probar las opciones ROPMI o ROPCI; seleccione como valor A cualquiera de estas dos constantes y, a continuación, continúe como se ha descrito anteriormente, pero sustituyendo el valor 8 en los pasos anteriores por el valor 2.

Estructura del campo de distintivos de mensaje en IBM i

El campo *MDMFL* es un entero de 32 bits que se divide en tres subcampos separados.

Estos subcampos identifican:

- Distintivos de mensaje que se rechazan si el gestor de colas local no los reconoce
- Distintivos de mensajes que siempre se aceptan, incluso si el gestor de colas local no los reconoce
- Distintivos de mensaje que se aceptan sólo si se cumplen otras condiciones

Nota: Todos los subcampos de *MDMFL* están reservados para que los utilice el gestor de colas.

Cada subcampo es identificado por una máscara de bit que tiene 1-bits en las posiciones correspondientes al subcampo, y 0-bits en otra parte. Los bits están numerados de tal manera que el bit 0 es el bit más significativo, y el bit 31 el bit menos significativo. Las máscaras siguientes están definidas para identificar los subcampos:

MFRO

Máscara para distintivos de mensaje no soportados que se rechazan.

Esta máscara identifica las posiciones de bits dentro del campo *MDMFL* donde los distintivos de mensaje que no están soportados por el gestor de colas local harán que la llamada MQPUT o MQPUT1 falle con el código de terminación CCFAIL y el código de razón RC2249.

Este subcampo ocupa las posiciones de bits 20 a 31.

En este subcampo se incluyen los distintivos de mensaje siguientes:

- MFLMIG
- MFLSEG
- MFMIG
- MFSEG
- MFSEGA
- MFSEGI

MFAUM

Máscara para distintivos de mensaje no soportados que se aceptan.

Esta máscara identifica las posiciones de bits dentro del campo *MDMFL* donde los distintivos de mensaje que no están soportados por el gestor de colas local se aceptarán en las llamadas MQPUT o MQPUT1. El código de terminación es CCOK.

Este subcampo ocupa las posiciones de bits 0 a 11.

MFAUXM

Máscara para distintivos de mensaje no soportados que sólo se aceptan en determinadas circunstancias.

Esta máscara identifica las posiciones de bits dentro del campo *MDMFL* donde los distintivos de mensaje no soportados por el gestor de colas local se aceptarán en las llamadas MQPUT o MQPUT1 *siempre que* se cumplan las dos condiciones siguientes:

- El mensaje está destinado a un gestor de colas remoto.
- La aplicación no coloca el mensaje directamente en una cola de transmisión local (es decir, la cola identificada por los campos *ODMN* y *ODON* en el descriptor de objeto especificado en la llamada MQOPEN o MQPUT1 no es una cola de transmisión local).

Se devuelve el código de terminación CCOK si se cumplen estas condiciones y CCFAIL con el código de razón RC2249 si no es así.

Este subcampo ocupa las posiciones de bits 12 a 19.

Si hay distintivos especificados en el campo *MDMFL* que el gestor de colas no reconoce, el gestor de colas comprueba cada subcampo a su vez utilizando la operación AND a nivel de bit para combinar el campo *MDMFL* con la máscara para ese subcampo. Si el resultado de esa operación no es cero, se devuelven el código de terminación y los códigos de razón descritos anteriormente.

Conversión de datos en IBM i

En este tema se describe la interfaz para la salida de conversión de datos y el proceso realizado por el gestor de colas cuando es necesaria la conversión de datos.

La salida de conversión de datos se invoca como parte del proceso de la llamada MQGET. Se utiliza para convertir los datos de mensaje de aplicación a la representación que necesita la aplicación receptora. La conversión de los datos del mensaje de aplicación es opcional y requiere que se especifique la opción GMCONV en la llamada MQGET.

Se describen los siguientes aspectos de la conversión de datos:

- El proceso realizado por el gestor de colas en respuesta a la opción GMCONV; consulte [“Proceso de conversión en IBM i”](#) en la página 1458.

- Convenios de proceso utilizados por el gestor de colas al procesar un formato incorporado; estos convenios también se recomiendan para las salidas escritas por el usuario. Consulte [“Convenios de proceso en IBM i”](#) en la página 1459.
- Consideraciones especiales para la conversión de mensajes de informe; consulte [“Conversión de mensajes de informe en IBM i”](#) en la página 1463.
- Los parámetros pasados a la salida de conversión de datos; consulte [“MQCONVX \(Salida de conversión de datos\) en IBM i”](#) en la página 1474.
- Llamada que se puede utilizar desde la salida para convertir datos de tipo carácter entre distintas representaciones; consulte [“MQXCNV \(Convertir caracteres\) en IBM i”](#) en la página 1469.
- El parámetro data-structure que es específico de la salida; consulte [“MQDXP \(parámetro de salida de conversión de datos\) en IBM i”](#) en la página 1464.

IBM i Proceso de conversión en IBM i

Esta información describe el proceso realizado por el gestor de colas en respuesta a la opción GMCONV.

El gestor de colas realiza las acciones siguientes si se especifica la opción GMCONV en la llamada MQGET y hay un mensaje que se debe devolver a la aplicación:

1. Si se cumple una o más de las condiciones siguientes, no es necesaria ninguna conversión:
 - Los datos del mensaje ya están en el juego de caracteres y la codificación que necesita la aplicación que emite la llamada MQGET. La aplicación debe establecer los campos *MDCSI* y *MDENC* en el parámetro **MSGDSC** de la llamada MQGET en los valores necesarios, antes de emitir la llamada.
 - La longitud de los datos del mensaje es cero.
 - La longitud del parámetro **BUFFER** de la llamada MQGET es cero.

En estos casos, el mensaje se devuelve sin conversión a la aplicación que emite la llamada MQGET; los valores *MDCSI* y *MDENC* del parámetro **MSGDSC** se establecen en los valores de la información de control del mensaje y la llamada se completa con una de las siguientes combinaciones de código de terminación y código de razón:

Código de terminación
Código de razón

CCOK
RCNONE

CCWARN
RC2079

CCWARN
RC2080

Los pasos siguientes sólo se realizan si el juego de caracteres o la codificación de los datos del mensaje difiere del valor correspondiente en el parámetro **MSGDSC** y hay datos que se deben convertir:

1. Si el campo *MDFMT* de la información de control del mensaje tiene el valor FMNONE, el mensaje se devuelve sin convertir, con el código de terminación CCWARN y el código de razón RC2110.
En todos los demás casos, el proceso de conversión continúa.
2. El mensaje se elimina de la cola y se coloca en un almacenamiento intermedio temporal que tiene el mismo tamaño que el parámetro **BUFFER**. Para las operaciones de examen, el mensaje se copia en el almacenamiento intermedio temporal, en lugar de eliminarse de la cola.
3. Si el mensaje debe truncarse para que quepa en el almacenamiento intermedio, se realiza lo siguiente:
 - Si no se ha especificado la opción GMATM, el mensaje se devuelve sin convertir, con el código de terminación CCWARN y el código de razón RC2080.
 - Si se ha especificado la opción de GMATM, el código de terminación se establece en CCWARN, el código de razón se establece en RC2079 y el proceso de conversión continúa.

4. Si el mensaje se puede acomodar en el almacenamiento intermedio sin truncamiento, o se ha especificado la opción GMATM, se realiza lo siguiente:

- Si el formato es un formato incorporado, el almacenamiento intermedio se pasa al servicio de conversión de datos del gestor de colas.
- Si el formato no es un formato incorporado, el almacenamiento intermedio se pasa a una salida escrita por el usuario que tiene el mismo nombre que el formato. Si no se puede encontrar la salida, el mensaje se devuelve sin convertir, con el código de terminación CCWARN y el código de razón RC2110.

Si no se produce ningún error, la salida del servicio de conversión de datos o de la salida escrita por el usuario es el mensaje convertido, más el código de terminación y el código de razón que se devolverá a la aplicación que emite la llamada MQGET.

5. Si la conversión es satisfactoria, el gestor de colas devuelve el mensaje convertido a la aplicación. En este caso, el código de terminación y el código de razón devueltos por la llamada MQGET normalmente serán una de las combinaciones siguientes:

Código de terminación

Código de razón

CCOK

RCNONE

CCWARN

RC2079

Sin embargo, si la conversión la realiza una salida escrita por el usuario, se pueden devolver otros códigos de razón, incluso cuando la conversión es satisfactoria.

Si la conversión falla (por cualquier motivo), el gestor de colas devuelve el mensaje sin convertir a la aplicación, con los campos *MDCSI* y *MDENC* del parámetro **MSGDSC** establecidos en los valores de la información de control del mensaje y con el código de terminación CCWARN.



Convenios de proceso en IBM i

Al convertir un formato incorporado, el gestor de colas sigue los convenios de proceso descritos en este tema.

Considere la posibilidad de aplicar estos convenios a las salidas escritas por el usuario, aunque el gestor de colas no lo aplique. Los formatos incorporados convertidos por el gestor de colas son los siguientes:

- FMADMN
- FMMDE
- FMCICS
- FMPCF
- FMCMD1
- HMRM
- FMCMD2
- FMRFH
- FMDLH
- FMRFH2
- FMDH
- FMSTR
- FMEVNT
- FMTM
- FMIMS
- FMXQH

- FMIMVS

1. Si el mensaje se expande durante la conversión y supera el tamaño del parámetro **BUFFER**, se realiza lo siguiente:

- Si no se ha especificado la opción GMATM, el mensaje se devuelve sin convertir, con el código de terminación CCWARN y el código de razón RC2120.
- Si se ha especificado la opción de GMATM, el mensaje se trunca, el código de terminación se establece en CCWARN, el código de razón se establece en RC2079y el proceso de conversión continúa.

2. Si se produce un truncamiento (antes o durante la conversión), es posible que el número de bytes válidos devueltos en el parámetro **BUFFER** sea *menor que* la longitud del almacenamiento intermedio.

Esto puede ocurrir, por ejemplo, si un entero de 4 bytes o un carácter DBCS se encuentra entre los extremos del almacenamiento intermedio. El elemento de información incompleto no se convierte y, por lo tanto, los bytes del mensaje devuelto no contienen información válida. Esto también puede ocurrir si un mensaje que se ha truncado antes de la conversión se reduce durante la conversión.

Si el número de bytes válidos devueltos es menor que la longitud del almacenamiento intermedio, los bytes no utilizados al final del almacenamiento intermedio se establecen en nulos.

3. Si una matriz o serie se encuentra entre el final del almacenamiento intermedio, se convierte la mayor parte posible de los datos; sólo el elemento de matriz o carácter DBCS concreto que está incompleto no se convierte-se convierten los elementos o caracteres de matriz anteriores.
4. Si se produce un truncamiento (antes o durante la conversión), la longitud devuelta para el parámetro **DATLEN** es la longitud del mensaje *unconvert* antes del truncamiento.
5. Cuando las series se convierten entre juegos de caracteres de un solo byte (SBCS), juegos de caracteres de doble byte (DBCS) o juegos de caracteres de varios bytes (MBCS), las series pueden expandirse o contraerse.

- En los formatos PCF FMADMN, FMEVNT y FMPCF, las series de las estructuras MQCFST y MQCFSL se expanden o se contraen según sea necesario para acomodar la serie después de la conversión.

Para la estructura de lista de series MQCFSL, las series de la lista pueden expandirse o contraerse en cantidades diferentes. Si esto sucede, el gestor de colas rellena las series más cortas con espacios en blanco para que tengan la misma longitud que la serie más larga después de la conversión.

- En el formato FMRMH, las series a las que se dirigen los campos RMSEO, RMSNO, RMDEOy RMDNO se expanden o contraen según sea necesario para acomodar las series después de la conversión.
- En el formato FMRFH, el campo RFNVS se expande o contrae según sea necesario para acomodar los pares nombre-valor después de la conversión.
- En estructuras con tamaños de campo fijos, el gestor de colas permite que las series se expandan o se contraigan dentro de sus campos fijos, si no se pierde información significativa. En este sentido, los espacios en blanco finales y los caracteres que siguen al primer carácter nulo del campo se tratan como insignificantes.
 - Si la serie se expande, pero sólo es necesario descartar los caracteres insignificantes para acomodar la serie convertida en el campo, la conversión se realiza correctamente y la llamada se completa con CCOK y el código de razón RCNONE (suponiendo que no haya otros errores).
 - Si la serie se expande, pero la serie convertida requiere que se descarten caracteres significativos para que quepan en el campo, el mensaje se devuelve sin convertir y la llamada se completa con CCWARN y el código de razón RC2190.

Nota: El código de razón RC2190 da como resultado en este caso si se ha especificado la opción GMATM.

- Si la serie se contrae, el gestor de colas rellena la serie con espacios en blanco hasta la longitud del campo.

6. Para los mensajes que constan de una o más estructuras de cabecera IBM MQ seguidas de datos de usuario, es posible convertir una o más de las estructuras de cabecera, mientras que el resto del mensaje no lo es. Sin embargo, con dos excepciones, los campos MDCSI y MDENC de cada estructura de cabecera siempre indican correctamente el juego de caracteres y la codificación de los datos que siguen a la estructura de cabecera.

Las dos excepciones son las estructuras MQCIH y MQIIH, donde los valores de los campos MDCSI y MDENC de esas estructuras no son significativos. Para esas estructuras, los datos que siguen a la estructura están en el mismo juego de caracteres y codificación que la propia estructura MQCIH o MQIIH.

7. Si los campos MDCSI o MDENC de la información de control del mensaje que se está recuperando, o en el parámetro **MSGDSC**, especifican valores que no están definidos o no están soportados, el gestor de colas puede ignorar el error si no es necesario utilizar el valor no definido o no soportado al convertir el mensaje.

Por ejemplo, si el campo MDENC del mensaje especifica una codificación flotante no soportada, pero el mensaje contiene sólo datos enteros, o contiene datos de coma flotante que no requieren conversión (porque las codificaciones flotantes de origen y destino son idénticas), el error puede o no diagnosticarse.

Si se diagnostica el error, el mensaje se devuelve sin convertir, con el código de terminación CCWARN y uno de los códigos de razón RC2111, RC2112, RC2113, RC2114 o RC2115, RC2116, RC2117, RC2118 (según corresponda); los campos MDCSI y MDENC del parámetro **MSGDSC** se establecen en los valores de la información de control del mensaje.

Si el error no se diagnostica y la conversión se completa correctamente, los valores devueltos en los campos MDCSI y MDENC en el parámetro **MSGDSC** son los especificados por la aplicación que emite la llamada MQGET.

8. En todos los casos, si el mensaje se devuelve a la aplicación sin convertir, el código de finalización se establece en CCWARN, y los campos MDCSI y MDENC del parámetro **MSGDSC** se establecen en los valores adecuados para los datos sin convertir. Esto también se hace para FMNONE.

El parámetro **REASON** se establece en un código que indica por qué no se ha podido llevar a cabo la conversión, a menos que el mensaje también se haya tenido que truncar; los códigos de razón relacionados con el truncamiento tienen prioridad sobre los códigos de razón relacionados con la conversión. (Para determinar si se ha convertido un mensaje truncado, compruebe los valores devueltos en los campos MDCSI y MDENC en el parámetro **MSGDSC**.)

Cuando se diagnostica un error, se devuelve un código de razón específico o el código de razón general RC2119. El código de razón devuelto depende de las prestaciones de diagnóstico del servicio de conversión de datos subyacente.

9. Si se devuelve el código de terminación CCWARN y hay más de un código de razón relevante, el orden de prioridad es el siguiente:
- a. La razón siguiente tiene prioridad sobre todas las demás:
 - RC2079
 - b. La siguiente en prioridad es la siguiente razón:
 - RC2110
 - c. El orden de prioridad dentro de los códigos de razón restantes no está definido.

10. Al finalizar la llamada MQGET:

- El siguiente código de razón indica que el mensaje se ha convertido correctamente:
 - RCNONE
- El siguiente código de razón indica que el mensaje *puede* haberse convertido correctamente (compruebe los campos MDCSI y MDENC en el parámetro **MSGDSC** para averiguarlo):
 - RC2079
- Los demás códigos de razón indican que el mensaje no se ha convertido.

El siguiente proceso es específico de los formatos incorporados; no es aplicable a los formatos definidos por el usuario:

1. Excepto para los formatos siguientes:

- FMADMN
- FMEVNT
- FMIMVS
- FMPCF
- FMSTR

ninguno de los formatos incorporados se puede convertir de o a conjuntos de caracteres que no tengan caracteres SBCS para los caracteres que son válidos en los nombres de cola. Si se intenta realizar una conversión de este tipo, el mensaje se devuelve sin convertir, con el código de terminación CCWARN y el código de razón RC2111 o RC2115, según corresponda.

El juego de caracteres Unicode UTF-16 es un ejemplo de un juego de caracteres que no tiene caracteres SBCS para los caracteres que son válidos en los nombres de cola.

2. Si los datos del mensaje para un formato incorporado se truncan, los campos dentro del mensaje que contienen longitudes de series, o recuentos de elementos o estructuras, no se ajustan para reflejar la longitud de los datos devueltos a la aplicación; los valores devueltos para dichos campos dentro de los datos del mensaje son los valores aplicables al mensaje antes del truncamiento.

Al procesar mensajes como, por ejemplo, un mensaje FMADMN truncado, debe asegurarse de que la aplicación no intenta acceder a los datos más allá del final de los datos devueltos.

3. Si el nombre de formato es FMDLH, los datos de mensaje empiezan por una estructura MQDLH, y esto puede ir seguido de cero o más bytes de datos de mensaje de aplicación. El formato, el juego de caracteres y la codificación de los datos del mensaje de aplicación se definen mediante los campos DLFMT, DLCSI y DLENC en la estructura MQDLH al principio del mensaje. Puesto que la estructura MQDLH y los datos del mensaje de aplicación pueden tener diferentes conjuntos de caracteres y codificaciones, es posible que uno, otro o ambos datos de la estructura MQDLH y del mensaje de aplicación requieran conversión.

El gestor de colas convierte primero la estructura MQDLH, según sea necesario. Si la conversión es satisfactoria, o la estructura MQDLH no requiere conversión, el gestor de colas comprueba los campos DLCSI y DLENC de la estructura MQDLH para ver si es necesaria la conversión de los datos del mensaje de aplicación. Si la conversión es necesaria, el gestor de colas invoca la salida escrita por el usuario con el nombre proporcionado por el campo DLFMT en la estructura MQDLH, o realiza la propia conversión (si DLFMT es el nombre de un formato incorporado).

Si la llamada MQGET devuelve un código de terminación de CCWARN, y el código de razón es uno de los que indica que la conversión no ha sido satisfactoria, se aplica uno de los siguientes:

- No se ha podido convertir la estructura MQDLH. En este caso, los datos del mensaje de aplicación tampoco se habrán convertido.
- La estructura MQDLH se ha convertido, pero los datos del mensaje de aplicación no.

La aplicación puede examinar los valores devueltos en los campos MDCSI y MDENC en el parámetro **MSGDSC**, y los de la estructura MQDLH, para determinar cuál de los anteriores se aplica.

4. Si el nombre de formato es FMXQH, los datos del mensaje empiezan con una estructura MQXQH, y esto puede ir seguido de cero o más bytes de datos adicionales. Estos datos adicionales son normalmente los datos de mensaje de aplicación (que pueden tener una longitud cero), pero también puede haber una o más estructuras de cabecera IBM MQ adicionales presentes, al principio de los datos adicionales.

La estructura MQXQH debe estar en el juego de caracteres y la codificación del gestor de colas. El formato, el juego de caracteres y la codificación de los datos que siguen a la estructura MQXQH se proporcionan mediante los campos MDFMT, MDCSI y MDENC en la estructura MQMD contenida en MQXQH. Para cada estructura de cabecera IBM MQ posterior presente, los campos MDFMT, MDCSI y

MDENC de la estructura describen los datos que siguen a esa estructura; esos datos son otra estructura de cabecera IBM MQ o los datos de mensaje de aplicación.

Si se especifica la opción GMCONV para un mensaje FMXQH, los datos del mensaje de aplicación y algunas de las estructuras de cabecera de MQ se convierten, pero los datos de la estructura MQXQH no. Al volver de la llamada MQGET, por lo tanto:

- Los valores de los campos MDFMT, MDCSI y MDENC del parámetro **MSGDSC**, describen los datos de la estructura MQXQH y no los datos del mensaje de aplicación; por lo tanto, los valores no serán los mismos que los especificados por la aplicación que ha emitido la llamada MQGET.

El efecto de esto es que una aplicación que obtiene repetidamente mensajes de una cola de transmisión con la opción GMCONV especificada debe restablecer los campos MDCSI y MDENC en el parámetro **MSGDSC** a los valores necesarios para los datos de mensaje de aplicación, antes de cada llamada MQGET.

- Los valores de los campos MDFMT, MDCSI y MDENC de la última estructura de cabecera de MQ presente describen los datos del mensaje de aplicación. Si no hay otras estructuras de cabecera IBM MQ presentes, los datos de mensaje de aplicación se describen mediante estos campos en la estructura MQMD dentro de la estructura MQXQH. Si la conversión es satisfactoria, los valores serán los mismos que los especificados en el parámetro **MSGDSC** por la aplicación que ha emitido la llamada MQGET.

Si el mensaje es un mensaje de lista de distribución, la estructura MQXQH va seguida de una estructura MQDH (más sus matrices de registros MQOR y MQPMR), que a su vez puede ir seguida de cero o más estructuras de cabecera IBM MQ adicionales y de cero o más bytes de datos de mensaje de aplicación. Al igual que la estructura MQXQH, la estructura MQDH debe estar en el juego de caracteres y la codificación del gestor de colas, y no se convierte en la llamada MQGET, incluso si se especifica la opción GMCONV.

El proceso de las estructuras MQXQH y MQDH descritas anteriormente están pensadas principalmente para que las utilicen los agentes de canal de mensajes cuando obtienen mensajes de las colas de transmisión.



Conversión de mensajes de informe en IBM i

Un mensaje de informe puede contener cantidades variables de datos de mensaje de aplicación, según las opciones de informe especificadas por el remitente del mensaje original.

En concreto, un mensaje de informe puede contener:

1. No hay datos de mensaje de aplicación
2. Algunos de los datos de mensaje de aplicación del mensaje original

Esto ocurre cuando el remitente del mensaje original especifica RO* D y el mensaje tiene más de 100 bytes.

3. Todos los datos de mensaje de aplicación del mensaje original

Esto ocurre cuando el remitente del mensaje original especifica RO* F, o especifica RO* D y el mensaje tiene 100 bytes o menos.

Cuando el gestor de colas o el agente de canal de mensajes genera un mensaje de informe, copia el nombre de formato del mensaje original en el campo *MDFMT* de la información de control del mensaje de informe. Por lo tanto, el nombre de formato del mensaje de informe puede implicar una longitud de datos diferente de la longitud presente en el mensaje de informe (casos 1 y 2 descritos anteriormente).

Si se especifica la opción GMCONV cuando se recupera el mensaje de informe:

- Para el caso 1 descrito anteriormente, no se invocará la salida de conversión de datos (porque el mensaje de informe no tendrá datos).
- Para el caso 3 descrito anteriormente, el nombre de formato implica correctamente la longitud de los datos del mensaje.

- Pero para el caso 2 descrito anteriormente, se invocará la salida de conversión de datos para convertir un mensaje que sea *más corto* que la longitud implícita por el nombre de formato.

Además, el código de razón pasado a la salida será normalmente RCNONE (es decir, el código de razón no indicará que el mensaje se ha truncado). Esto sucede porque los datos del mensaje han sido truncados por el *emisor* del mensaje de informe y no por el gestor de colas del receptor en respuesta a la llamada MQGET.

Debido a estas posibilidades, la salida de conversión de datos no debe utilizar el nombre de formato para deducir la longitud de los datos que se le pasan; en su lugar, la salida debe comprobar la longitud de los datos proporcionados y estar preparada para convertir menos datos que la longitud implícita en el nombre de formato. Si los datos se pueden convertir correctamente, la salida debe devolver el código de terminación CCOK y el código de razón RCNONE. La longitud de los datos de mensaje que se van a convertir se pasa a la salida como parámetro **INLEN**.

Interfaz de programación sensible al producto

Si un mensaje de informe contiene información sobre una actividad que ha tenido lugar, se conoce como informe de actividad. Ejemplos de actividades son:

- un MCA que envía un mensaje desde una cola hacia abajo en un canal
- un MCA que recibe un mensaje de un canal y lo coloca en una cola
- un mensaje no entregado MCA que pone en cola un mensaje no entregable
- un MCA que obtiene un mensaje de una cola y lo descarta
- un manejador de mensajes no entregados colocando un mensaje de nuevo en una cola
- el servidor de mandatos que procesa una solicitud PCF-un intermediario que procesa una solicitud de publicación
- una aplicación de usuario que obtiene un mensaje de una cola: una aplicación de usuario que examina un mensaje en una cola

Cualquier aplicación, incluido el gestor de colas, puede añadir algunos de los datos de mensaje al informe de actividad después de la cabecera del informe. La cantidad de datos que se deben proporcionar si se envían algunos no es fija y la decide la aplicación. La información devuelta debe ser útil para la aplicación que procesa el informe de actividad. Los informes de actividad del gestor de colas devolverán con ellos las estructuras de cabecera IBM MQ estándar (empezando por 'MQH') contenidas en el mensaje original. Esto incluye, por ejemplo, cualquier cabecera MQRFH2 que se haya incluido en el mensaje original. Además, el gestor de colas devolverá una cabecera MQCFH encontrada, pero no los parámetros PCF asociados a ella. Esto da a las aplicaciones de supervisión una idea de lo que era el mensaje.

IBM i

MQDXP (parámetro de salida de conversión de datos) en IBM i

Bloque de parámetros de salida de conversión de datos.

Visión general

Finalidad: La estructura MQDXP es un parámetro que el gestor de colas pasa a la salida de conversión de datos cuando se invoca la salida para convertir los datos del mensaje como parte del proceso de la llamada MQGET. Consulte la descripción de la llamada MQCONVX para obtener detalles de la salida de conversión de datos.

Conjunto de caracteres y codificación: los datos de caracteres de MQDXP están en el juego de caracteres del gestor de colas local; esto lo proporciona el atributo de gestor de colas **CodedCharSetId**. Los datos numéricos en MQDXP están en la codificación de máquina nativa; esto lo proporciona ENNAT.

Uso: solo la salida puede cambiar los campos *DXLEN*, *DXCC*, *DXREA* y *DXRES* en MQDXP; los cambios en otros campos se ignoran. Sin embargo, el campo *DXLEN* no se puede cambiar si el mensaje que se está convirtiendo es un segmento que sólo contiene parte de un mensaje lógico.

Cuando el control vuelve al gestor de colas desde la salida, el gestor de colas comprueba los valores devueltos en MQDXP. Si los valores devueltos no son válidos, el gestor de colas continúa el proceso como si la salida hubiera devuelto XRFAIL en DXRES ; sin embargo, el gestor de colas ignora los valores de los campos DXCC y DXREA devueltos por la salida en este caso, y utiliza en su lugar los valores que esos campos tenían en *entrada* para la salida. Los siguientes valores en MQDXP hacen que se produzca este proceso:

- DXRES campo no XROK y no XRFAIL
- DXCC campo no CCOK y no CCWARN
- DXLEN campo menor que cero, o DXLEN campo cambiado cuando el mensaje que se está convirtiendo es un segmento que sólo contiene parte de un mensaje lógico.
- “Campos” en la página 1465
- “Declaración RPG (copiar archivo CMQDXPH)” en la página 1469

Campos

La estructura MQDXP contiene los campos siguientes; los campos se describen en **orden alfabético**:

DXAOP (entero con signo de 10 dígitos)

Opciones de aplicación.

Se trata de una copia del campo *GMOPT* de la estructura MQGMO especificada por la aplicación que emite la llamada MQGET. Es posible que la salida tenga que examinarlos para determinar si se ha especificado la opción GMATM.

Es un campo de entrada para la salida.

DXCC (entero con signo de 10 dígitos)

Código de terminación.

Cuando se invoca la salida, contiene el código de terminación que se devolverá a la aplicación que ha emitido la llamada MQGET, si la salida elige no hacer nada. Siempre es CCWARN, porque el mensaje se ha truncado, o el mensaje requiere conversión y esto todavía no se ha realizado.

En la salida de la salida, este campo contiene el código de terminación que se devolverá a la aplicación en el parámetro **CMPCOD** de la llamada MQGET; sólo CCOK y CCWARN son válidos. Consulte la descripción del campo *DXREA* para obtener sugerencias sobre cómo la salida debe establecer este campo en la salida.

Es un campo de entrada/salida para la salida.

DXCSI (entero con signo de 10 dígitos)

Juego de caracteres necesario para la aplicación.

Este es el identificador de juego de caracteres codificado del juego de caracteres que necesita la aplicación que emite la llamada MQGET; consulte el campo *MDCSI* en la estructura MQMD para obtener más detalles. Si la aplicación especifica el valor especial CSQM en la llamada MQGET, el gestor de colas lo cambia por el identificador de juego de caracteres real del juego de caracteres utilizado por el gestor de colas, antes de invocar la salida.

Si la conversión es satisfactoria, la salida debe copiarla en el campo *MDCSI* del descriptor de mensaje.

Es un campo de entrada para la salida.

DXENC (entero con signo de 10 dígitos)

Codificación numérica necesaria para la aplicación.

Esta es la codificación numérica que necesita la aplicación que emite la llamada MQGET; consulte el campo *MDENC* en la estructura MQMD para obtener más detalles.

Si la conversión es satisfactoria, la salida debe copiarla en el campo *MDENC* del descriptor de mensaje.

Es un campo de entrada para la salida.

DXHCN (entero con signo de 10 dígitos)

Descriptor de conexión.

Se trata de un descriptor de conexión que se puede utilizar en la llamada MQXCNCV. Este descriptor de contexto no es necesariamente el mismo que el descriptor de contexto especificado por la aplicación que ha emitido la llamada MQGET.

DXLEN (entero con signo de 10 dígitos)

Longitud en bytes de datos de mensaje.

Cuando se invoca la salida, este campo contiene la longitud original de los datos del mensaje de aplicación. Si el mensaje se ha truncado para que quepa en el almacenamiento intermedio proporcionado por la aplicación, el tamaño del mensaje proporcionado a la salida será *menor* que el valor de **DXLEN**. El tamaño del mensaje proporcionado a la salida siempre lo proporciona el parámetro **INLEN** de la salida, independientemente de cualquier truncamiento que se haya podido producir.

El truncamiento se indica mediante el campo **DXREA** que tiene el valor RC2079 en la entrada de la salida.

La mayoría de las conversiones no tendrán que cambiar esta longitud, pero una salida puede hacerlo si es necesario; el valor establecido por la salida se devuelve a la aplicación en el parámetro **DATLEN** de la llamada MQGET. Sin embargo, esta longitud no se puede cambiar si el mensaje que se está convirtiendo es un segmento que sólo contiene parte de un mensaje lógico. Esto se debe a que cambiar la longitud haría que los desplazamientos de segmentos posteriores en el mensaje lógico fueran incorrectos.

Tenga en cuenta que, si la salida desea cambiar la longitud de los datos, tenga en cuenta que el gestor de colas ya ha decidido si los datos del mensaje se ajustarán al almacenamiento intermedio de la aplicación, basándose en la longitud de los datos *no convertidos*. Esta decisión determina si el mensaje se elimina de la cola (o se mueve el cursor para examinar, para una solicitud de examinar) y no se ve afectado por ningún cambio en la longitud de los datos causado por la conversión. Por este motivo, se recomienda que las salidas de conversión no provoquen un cambio en la longitud de los datos del mensaje de aplicación.

Si la conversión de caracteres implica un cambio de longitud, una serie se puede convertir en otra serie con la misma longitud en bytes, truncando los espacios en blanco finales o rellenando con espacios en blanco según sea necesario.

La salida no se invoca si el mensaje no contiene datos de mensaje de aplicación; por lo tanto, **DXLEN** siempre es mayor que cero.

Es un campo de entrada/salida para la salida.

DXREA (entero con signo de 10 dígitos)

Código de razón que califica **DXCC**.

Cuando se invoca la salida, contiene el código de razón que se devolverá a la aplicación que ha emitido la llamada MQGET, si la salida elige no hacer nada. Entre los valores posibles se encuentran RC2079, que indica que el mensaje se ha truncado para ajustarse al almacenamiento intermedio proporcionado por la aplicación, y RC2119, que indica que el mensaje requiere conversión pero que todavía no se ha realizado.

En la salida de la salida, este campo contiene la razón que debe devolverse a la aplicación en el parámetro **REASON** de la llamada MQGET; se recomienda lo siguiente:

- Si **DXREA** tenía el valor RC2079 en la entrada de la salida, los campos **DXREA** y **DXCC** no se deben modificar, independientemente de si la conversión se realiza correctamente o falla.

(Si el campo **DXCC** no es CCOK, la aplicación que recupera el mensaje puede identificar una anomalía de conversión comparando los valores **MDENC** y **MDCSI** devueltos en el descriptor de mensaje con los valores solicitados; por el contrario, la aplicación no puede distinguir un mensaje truncado de un mensaje que acaba de ajustarse al almacenamiento intermedio. Por este motivo,

RC2079 debe devolverse con preferencia a cualquiera de las razones que indican un error de conversión.)

- Si *DXREA* tenía cualquier otro valor en la entrada para la salida:
 - Si la conversión se realiza correctamente, *DXCC* debe establecerse en CCOK y *DXREA* debe establecerse en RCNONE.
 - Si la conversión falla, o el mensaje se expande y se tiene que truncar para que quepa en el almacenamiento intermedio, *DXCC* debe establecerse en CCWARN (o dejarse sin modificar), y *DXREA* establecerse en uno de los valores *i* de la lista siguiente, para indicar la naturaleza del error.

Tenga en cuenta que, si el mensaje después de la conversión es demasiado grande para el almacenamiento intermedio, sólo se debe truncar si la aplicación que ha emitido la llamada MQGET ha especificado la opción GMATM:

- Si ha especificado esta opción, se debe devolver la razón RC2079 .
- Si no ha especificado esta opción, el mensaje debe devolverse sin convertir, con el código de razón RC2120.

Los códigos de razón de la lista siguiente se recomiendan para que los utilice la salida para indicar la razón por la que ha fallado la conversión, pero la salida puede devolver otros valores del conjunto de códigos RC* si se considera apropiado. Además, el rango de valores RC0900 a RC0999 se asignan para que los utilice la salida para indicar las condiciones en las que la salida desea comunicarse con la aplicación que emite la llamada MQGET.

Nota: Si el mensaje no se puede convertir correctamente, la salida debe devolver XRFAIL en el campo *DXRES* , para que el gestor de colas devuelva el mensaje no convertido. Esto es cierto independientemente del código de razón devuelto en el campo *DXREA* .

RC0900

(900, X'384 ') Valor más bajo para el código de razón definido por la aplicación.

RC0999

(999, X'3E7') Valor más alto para el código de motivo definido por la aplicación.

RC2120

(2120, X'848 ') Datos convertidos demasiado grandes para almacenamiento intermedio.

RC2119

(2119, X'847') Los datos del mensaje no se han convertido.

RC2111

(2111, X'83F') El identificador de juego de caracteres codificado origen no es válido.

RC2113

(2113, X'841') Codificación de decimal empaquetado en el mensaje no reconocida.

RC2114

(2114, X'842') Codificación de coma flotante en el mensaje no reconocida.

RC2112

(2112, X'840') Codificación de entero de origen no reconocida.

RC2115

(2115, X'843') El identificador de juego de caracteres codificado de destino no es válido.

RC2117

(2117, X'845') Codificación de decimal empaquetado especificada por receptor no reconocida.

RC2118

(2118, X'846') Codificación de coma flotante especificada por receptor no reconocida.

RC2116

(2116, X'844') Codificación de entero de destino no reconocida.

RC2079

(2079, X'81F') Se ha devuelto un mensaje truncado (el proceso se ha completado).

Es un campo de entrada/salida para la salida.

DXRES (entero con signo de 10 dígitos)

Respuesta de la salida.

Esto lo establece la salida para indicar el éxito o no de la conversión. Debe ser uno de los siguientes:

XRC

La conversión ha sido satisfactoria.

Si la salida especifica este valor, el gestor de colas devuelve lo siguiente a la aplicación que ha emitido la llamada MQGET:

- El valor del campo *DXCC* en la salida de la salida
- El valor del campo *DXREA* en la salida de la salida
- El valor del campo *DXLEN* en la salida de la salida
- El contenido del almacenamiento intermedio de salida de la salida *OUTBUF*. El número de bytes devueltos es el menor del parámetro **OUTLEN** de la salida y el valor del campo *DXLEN* en la salida de la salida

Si los campos *MDENC* y *MDCSI* del parámetro de descriptor de mensaje de la salida *ambos* no se modifican, el gestor de colas devuelve:

- El valor de los campos *MDENC* y *MDCSI* en la estructura MQDXP en *entrada* a la salida

Si se ha cambiado uno o ambos campos *MDENC* y *MDCSI* en el parámetro de descriptor de mensaje de la salida, el gestor de colas devuelve:

- El valor de los campos *MDENC* y *MDCSI* en el parámetro de descriptor de mensaje de la salida en la salida de la salida
-

XRFAIL

La conversión no ha sido satisfactoria.

Si la salida especifica este valor, el gestor de colas devuelve lo siguiente a la aplicación que ha emitido la llamada MQGET:

- El valor del campo *DXCC* en la salida de la salida
- El valor del campo *DXREA* en la salida de la salida
- El valor del campo *DXLEN* en *entrada* a la salida
- El contenido del almacenamiento intermedio de entrada de la salida *INBUF*. El número de bytes devueltos lo proporciona el parámetro **INLEN**

Si la salida ha modificado *INBUF*, los resultados no están definidos.

DXRES es un campo de salida de la salida.

DXSID (serie de caracteres de 4 bytes)

Identificador de estructura.

El valor debe ser:

DXSIDV

Identificador de la estructura de parámetros de salida de conversión de datos.

Es un campo de entrada para la salida.

DXVER (entero con signo de 10 dígitos)

Número de versión de la estructura.

El valor debe ser:

DXVER1

Número de versión para la estructura de parámetros de salida de conversión de datos.

La constante siguiente especifica el número de versión de la versión actual:

DXVERC

Versión actual de la estructura de parámetros de salida de conversión de datos.

Nota: Cuando se introduce una nueva versión de esta estructura, el diseño de la parte existente no cambia. Por lo tanto, la salida debe comprobar que el campo *DXVER* es igual o mayor que la versión más baja que contiene los campos que la salida necesita utilizar.

Es un campo de entrada para la salida.

DXXOP (entero con signo de 10 dígitos)

Reservado.

Se trata de un campo reservado; su valor es 0.

Declaración RPG (copiar archivo CMQDXPH)

```
D*..1.....2.....3.....4.....5.....6.....7..
D* MQDXP Structure
D*
D* Structure identifier
D DXSID 1 4
D* Structure version number
D DXVER 5 8I 0
D* Reserved
D DXXOP 9 12I 0
D* Application options
D DXAOP 13 16I 0
D* Numeric encoding required by application
D DXENC 17 20I 0
D* Character set required by application
D DXCSI 21 24I 0
D* Length in bytes of message data
D DXLEN 25 28I 0
D* Completion code
D DXCC 29 32I 0
D* Reason code qualifying DXCC
D DXREA 33 36I 0
D* Response from exit
D DXRES 37 40I 0
D* Connection handle
D DXHCN 41 44I 0
```

IBM i

MQXCNCV (Convertir caracteres) en IBM i

La llamada MQXCNCV convierte los caracteres de un juego de caracteres a otro.

Esta llamada forma parte de la interfaz de conversión de datos (DCI) de IBM MQ , que es una de las interfaces de infraestructura de IBM MQ . Nota: Esta llamada sólo se puede utilizar desde una salida de conversión de datos.

- [“Sintaxis” en la página 1469](#)
- [“Parámetros” en la página 1470](#)
- [“Invocación de RPG \(ILE\)” en la página 1474](#)

Sintaxis

MQXCNCV HCONN, OPTS, SRCCSI, SRCLEN, SRCBUF, TGTCSI, TGTLEN, TGTBUF, DATLEN, CMPCOD, REASON)

Parámetros

La llamada MQXCNVN tiene los parámetros siguientes:

HCONN (entero con signo de 10 dígitos)-entrada

Descriptor de conexión.

Este manejador representa la conexión con el gestor de colas. Normalmente debe ser el descriptor de contexto pasado a la salida de conversión de datos en el campo DXHCN de la estructura MQDXP; este descriptor de contexto no es necesariamente el mismo que el descriptor de contexto especificado por la aplicación que ha emitido la llamada MQGET.

En IBM i, se puede especificar el siguiente valor especial para HCONN:

HCDEFH

Manejador de conexión predeterminado.

OPTS (entero con signo de 10 dígitos)-entrada

Opciones que controlan la acción de MQXCNVN.

Se pueden especificar cero o más de las opciones descritas más adelante en esta sección. Si se necesita más de uno, se pueden añadir los valores (no añadir la misma constante más de una vez).

Opción de conversión predeterminada: la siguiente opción controla el uso de la conversión de caracteres predeterminada:

DCCDEF

Conversión predeterminada.

Esta opción especifica que se puede utilizar la conversión de caracteres por omisión si uno o ambos de los juegos de caracteres especificados en la llamada no están soportados. Esto permite al gestor de colas utilizar un juego de caracteres predeterminado especificado por la instalación que se aproxima al juego de caracteres especificado, al convertir la serie.

Nota: El resultado de utilizar un juego de caracteres aproximado para convertir la serie es que algunos caracteres pueden convertirse incorrectamente. Esto se puede evitar utilizando en la serie sólo caracteres que son comunes tanto al juego de caracteres especificado como al juego de caracteres predeterminado.

Los juegos de caracteres predeterminados se definen mediante una opción de configuración cuando se instala o se reinicia el gestor de colas.

Si no se especifica DCCDEF, el gestor de colas sólo utiliza los juegos de caracteres especificados para convertir la serie, y la llamada falla si uno o ambos juegos de caracteres no están soportados.

Opción de relleno: la opción siguiente permite al gestor de colas rellenar la serie convertida con espacios en blanco o descartar caracteres finales insignificantes, para que la serie convertida se ajuste al almacenamiento intermedio de destino:

DCCFIL

Rellene el almacenamiento intermedio de destino.

Esta opción solicita que la conversión tenga lugar de forma que el almacenamiento intermedio de destino se llene completamente:

- Si la serie se contrae cuando se convierte, se añaden espacios en blanco finales para rellenar el almacenamiento intermedio de destino.
- Si la serie se expande cuando se convierte, los caracteres finales que no son significativos se descartan para que la serie convertida se ajuste al almacenamiento intermedio de destino. Si esto se puede realizar correctamente, la llamada se completa con CCOK y el código de razón RCNONE.

Si hay muy pocos caracteres finales insignificantes, la mayor parte de la serie que quepa se coloca en el almacenamiento intermedio de destino y la llamada se completa con CCWARN y el código de razón RC2120.

Los caracteres insignificantes son:

- Blancos de cola
- Caracteres que siguen al primer carácter nulo de la serie (pero excluyendo el propio primer carácter nulo)
- Si la serie, TGTCSI y TGTLEN son tales que el almacenamiento intermedio de destino no se puede establecer completamente con caracteres válidos, la llamada falla con CCFAIL y el código de razón RC2144. Esto puede ocurrir cuando TGTCSI es un juego de caracteres DBCS puro (como UTF-16), pero TGTLEN especifica una longitud que es un número impar de bytes.
- TGTLEN puede ser menor o mayor que SRCLEN. Al volver de MQXCNCV, DATLEN tiene el mismo valor que TGTLEN.

Si no se especifica esta opción:

- La serie puede contraerse o expandirse dentro del almacenamiento intermedio de destino según sea necesario. Los caracteres finales insignificantes no se añaden ni descartan.

Si la serie convertida se ajusta al almacenamiento intermedio de destino, la llamada se completa con CCOK y el código de razón RCNONE.

Si la serie convertida es demasiado grande para el almacenamiento intermedio de destino, la mayor parte de la serie que quepa se coloca en el almacenamiento intermedio de destino y la llamada se completa con CCWARN y el código de razón RC2120. Tenga en cuenta que en este caso se pueden devolver menos de *TGTLEN* bytes.

- TGTLEN puede ser menor o mayor que SRCLEN. Al volver de MQXCNCV, DATLEN es menor o igual que TGTLEN.

Opciones de codificación: se pueden utilizar las opciones siguientes para especificar las codificaciones de enteros de las series de origen y destino. La codificación relevante sólo se utiliza cuando el identificador de juego de caracteres correspondiente indica que la representación del juego de caracteres en el almacenamiento principal depende de la codificación utilizada para los enteros binarios. Esto sólo afecta a determinados juegos de caracteres multibyte (por ejemplo, juegos de caracteres UTF-16).

La codificación se ignora si el juego de caracteres es un juego de caracteres de un solo byte (SBCS), o un juego de caracteres de varios bytes con representación en el almacenamiento principal que no depende de la codificación de enteros.

Sólo debe especificarse uno de los valores DCCS*, combinado con uno de los valores DCCT*:

DCCSNA

La codificación de origen es el valor predeterminado para el entorno y el lenguaje de programación.

DCCSNO

La codificación de origen es normal.

DCCSRE

La codificación de origen se invierte.

DCCSUN

La codificación de origen no está definida.

DCCTNA

La codificación de destino es el valor predeterminado para el entorno y el lenguaje de programación.

DCCTNO

La codificación de destino es normal.

DCCTRE

La codificación de destino se invierte.

DCCTUN

La codificación de destino no está definida.

Los valores de codificación definidos anteriormente se pueden añadir directamente al campo OPTS . Sin embargo, si la codificación de origen o destino se obtiene del campo MDENC en el MQMD u otra estructura, se debe realizar el proceso siguiente:

1. La codificación de enteros debe extraerse del campo MDENC eliminando las codificaciones decimal flotante y empaquetado; consulte [“Análisis de codificaciones en IBM i”](#) en la página 1453 para obtener detalles sobre cómo hacerlo.
2. La codificación de enteros resultante del paso 1 debe multiplicarse por el factor adecuado antes de añadirse al campo OPTS . Estos factores son:

DCCSFA

Factor para codificación de origen

DCCTFA

Factor para codificación de destino

Si no se especifica, las opciones de codificación toman como valor predeterminado sin definir (DCC* UN). En la mayoría de los casos, esto no afecta a la finalización satisfactoria de la llamada MQXCNVC. Sin embargo, si el juego de caracteres correspondiente es un juego de caracteres multibyte con representación que depende de la codificación (por ejemplo, un juego de caracteres UTF-16), la llamada falla con el código de razón RC2112 o RC2116 según corresponda.

Opción predeterminada: si no se especifica ninguna de las opciones descritas anteriormente, se puede utilizar la opción siguiente:

DCCNON

No se ha especificado ninguna opción.

DCCNON está definido para ayudar a la documentación del programa. No se pretende que esta opción se utilice con ninguna otra, pero como su valor es cero, no se puede detectar dicho uso.

SRCCSI (entero con signo de 10 dígitos)-entrada

Identificador de juego de caracteres codificado de serie antes de la conversión.

Es el identificador de juego de caracteres codificado de la serie de entrada en SRCBUF.

SRCLLEN (entero con signo de 10 dígitos)-entrada

Longitud de la serie antes de la conversión.

Es la longitud en bytes de la serie de entrada en SRCBUF ; debe ser cero o mayor.

SRCBUF (serie de caracteres de 1 byte x SRCLLEN)-entrada

Serie que se va a convertir.

Es el almacenamiento intermedio que contiene la serie que se va a convertir de un juego de caracteres a otro.

TGTCSI (entero con signo de 10 dígitos)-entrada

Identificador de juego de caracteres codificado de la serie después de la conversión.

Es el identificador de juego de caracteres codificado del juego de caracteres al que se va a convertir SRCBUF .

TGTLEN (entero con signo de 10 dígitos)-entrada

Longitud del almacenamiento intermedio de salida.

Es la longitud en bytes del almacenamiento intermedio de salida TGTBUF ; debe ser cero o mayor. Puede ser menor o mayor que SRCLLEN.

TGTBUF (serie de caracteres de 1 byte x TGTLEN)-salida

Serie después de la conversión.

Esta es la serie después de que se haya convertido al juego de caracteres definido por TGTCSI. La serie convertida puede ser más corta o más larga que la serie no convertida. El parámetro **DATLEN** indica el número de bytes válidos devueltos.

DATLEN (entero con signo de 10 dígitos)-salida

Longitud de la serie de salida.

Es la longitud de la serie devuelta en el almacenamiento intermedio de salida TGTBUF. La serie convertida puede ser más corta o más larga que la serie no convertida.

CMPCOD (entero con signo de 10 dígitos)-salida

Código de terminación.

Es uno de los siguientes:

CCOK

Realización satisfactoria.

CCWARN

Aviso (finalización parcial).

CCFAIL

La llamada no se ha realizado satisfactoriamente.

REASON (entero con signo de 10 dígitos)-salida

Código de razón que califica CMPCOD.

Si CMPCOD es CCOK:

RCNONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si CMPCOD es CCWARN:

RC2120

(2120, X'848') Los datos convertidos son demasiado grandes para el almacenamiento intermedio.

Si CMPCOD es CCFAIL:

RC2010

(2010, X'7DA') El parámetro longitud de datos no es válido.

RC2150

(2150, X'866') La serie DBCS no es válida.

RC2018

(2018, X'7E2') El manejador de conexión no es válido.

RC2046

(2046, X'7FE') Las opciones no son válidas o no son coherentes.

RC2102

(2102, X'836') No hay disponibles suficientes recursos del sistema.

RC2145

(2145, X'861') El parámetro de almacenamiento intermedio de origen no es válido.

RC2111

(2111, X'83F') El identificador de juego de caracteres codificado origen no es válido.

RC2112

(2112, X'840') Codificación de entero de origen no reconocida.

RC2143

(2143, X'85F') Parámetro de longitud de origen no válido.

RC2071

(2071, X'817') No hay suficiente almacenamiento disponible.

RC2146

(2146, X'862') Parámetro de almacenamiento intermedio de destino no válido.

RC2115

(2115, X'843') El identificador de juego de caracteres codificado de destino no es válido.

RC2116

(2116, X'844') Codificación de entero de destino no reconocida.

RC2144

(2144, X'860 ') El parámetro de longitud de destino no es válido.

RC2195

(2195, X'893') Se ha producido un error inesperado.

Para obtener más información sobre estos códigos de razón, consulte [“Códigos de retorno para IBM i \(ILE RPG\)”](#) en la página 1447.

Invocación de RPG (ILE)

```
C*.1.....2.....3.....4.....5.....6.....7..
C                                CALLP      MQXCNCV(HCONN : OPTS : SRCCSI :
C                                SRCLN : SRCBUF : TGTCSI :
C                                TGTLEN : TGTFUF : DATLEN :
C                                CMPCOD : REASON)
```

La definición de prototipo para la llamada es:

```
D*.1.....2.....3.....4.....5.....6.....7..
DMQXCNCV      PR      EXTPROC('MQXCNCV')
D* Connection handle
D HCONN      10I 0 VALUE
D* Options that control the action of MQXCNCV
D OPTS      10I 0 VALUE
D* Coded character set identifier of string before conversion
D SRCCSI      10I 0 VALUE
D* Length of string before conversion
D SRCLN      10I 0 VALUE
D* String to be converted
D SRCBUF      *      VALUE
D* Coded character set identifier of string after conversion
D TGTCSI      10I 0 VALUE
D* Length of output buffer
D TGTLEN      10I 0 VALUE
D* String after conversion
D TGTFUF      *      VALUE
D* Length of output string
D DATLEN      10I 0
D* Completion code
D CMPCOD      10I 0
D* Reason code qualifying CMPCOD
D REASON      10I 0
```

IBM i MQCONVX (Salida de conversión de datos) en IBM i

Esta definición de llamada describe los parámetros que se pasan a la salida de conversión de datos.

El gestor de colas no proporciona ningún punto de entrada denominado MQCONVX (consulte la nota de uso [“11”](#) en la página 1476).

Esta definición forma parte de la interfaz de conversión de datos (DCI) de IBM MQ , que es una de las interfaces de infraestructura de IBM MQ .

- [“Sintaxis”](#) en la página 1475
- [“Notas de uso”](#) en la página 1475
- [“Parámetros”](#) en la página 1476
- [“Invocación de RPG \(ILE\)”](#) en la página 1478

Sintaxis

MQCONVX (MQDXP, MQMD, INLEN, INBUF, OUTLEN, OUTBUF)

Notas de uso

1. Una salida de conversión de datos es una salida escrita por el usuario que recibe el control durante el proceso de una llamada MQGET. La función realizada por la salida de conversión de datos la define el proveedor de la salida; sin embargo, la salida debe ajustarse a las reglas descritas aquí y en la estructura de parámetros asociada MQDXP.

El entorno determina los lenguajes de programación que se pueden utilizar para una salida de conversión de datos.

2. La salida sólo se invoca si se cumplen *todas* las sentencias siguientes:

- La opción GMCONV se especifica en la llamada MQGET
- El campo *MDFMT* del descriptor de mensaje no es FMNONE
- El mensaje no está ya en la representación necesaria; es decir, uno o ambos *MDCSI* y *MDENC* del mensaje son diferentes del valor especificado por la aplicación en el descriptor de mensaje proporcionado en la llamada MQGET
- El gestor de colas todavía no ha realizado la conversión correctamente
- La longitud del almacenamiento intermedio de la aplicación es mayor que cero
- La longitud de los datos del mensaje es mayor que cero
- El código de razón hasta ahora durante la operación MQGET es RCNONE o RC2079

3. Cuando se está escribiendo una salida, se debe tener en cuenta la codificación de la salida de una forma que le permita convertir los mensajes que se han truncado. Los mensajes truncados pueden aparecer de las siguientes maneras:

- La aplicación receptora proporciona un almacenamiento intermedio que es menor que el mensaje, pero especifica la opción GMATM en la llamada MQGET.

En este caso, el campo *DXREA* del parámetro **MQDXP** en la entrada de la salida tendrá el valor RC2079.

- El remitente del mensaje lo ha truncado antes de enviarlo. Esto puede suceder con los mensajes de informe, por ejemplo (consulte [“Conversión de mensajes de informe en IBM i”](#) en la página 1463 para obtener más detalles).

En este caso, el campo *DXREA* del parámetro **MQDXP** en la entrada de la salida tendrá el valor RCNONE (si la aplicación receptora ha proporcionado un almacenamiento intermedio lo suficientemente grande para el mensaje).

Por lo tanto, el valor del campo *DXREA* en la entrada a la salida no siempre se puede utilizar para decidir si el mensaje se ha truncado.

La característica distintiva de un mensaje truncado es que la longitud proporcionada a la salida en el parámetro **INLEN** será *menor que* la longitud implícita por el nombre de formato contenido en el campo *MDFMT* del descriptor de mensaje. Por lo tanto, la salida debe comprobar el valor de *INLEN* antes de intentar convertir cualquiera de los datos; la salida *no* debe suponer que se ha proporcionado la cantidad completa de datos implícita en el nombre de formato.

Si la salida no se ha grabado para convertir mensajes truncados, y **INLEN** es menor que el valor esperado, la salida debe devolver XRFAIL en el campo *DXRES* del parámetro **MQDXP**, con el campo *DXCC* establecido en CCWARN y el campo *DXREA* establecido en RC2110.

Si la salida *ha* se ha grabado para convertir mensajes truncados, la salida debe convertir la mayor cantidad posible de datos (consulte la siguiente nota de uso), teniendo cuidado de no intentar examinar o convertir datos más allá del final de *INBUF*. Si la conversión se completa correctamente, la salida debe dejar el campo *DXREA* en el parámetro **MQDXP** sin modificar. Esto devuelve RC2079

si el gestor de colas del receptor ha truncado el mensaje y RCNONE si el emisor del mensaje ha truncado el mensaje.

También es posible que un mensaje expanda *durante la conversión de* , hasta el punto en el que es mayor que *OUTBUF*. En este caso, la salida debe decidir si trunca el mensaje; el campo *DXAOP* del parámetro **MQDXP** indicará si la aplicación receptora ha especificado la opción GMATM.

4. Por lo general, se recomienda que se conviertan todos los datos del mensaje proporcionado a la salida en *INBUF* , o que no se convierta ninguno de ellos. Sin embargo, se produce una excepción a esto si el mensaje se trunca, ya sea antes de la conversión o durante la conversión; en este caso puede haber un elemento incompleto al final del almacenamiento intermedio (por ejemplo: un byte de un carácter de doble byte o 3 bytes de un entero de 4 bytes). En esta situación, se recomienda que se omita el elemento incompleto y que los bytes no utilizados en *OUTBUF* se establezcan en nulos. Sin embargo, los elementos o caracteres completos dentro de una matriz o serie *deben* convertirse.
5. Cuando se necesita una salida por primera vez, el gestor de colas intenta cargar un objeto que tiene el mismo nombre que el formato (aparte de las extensiones). El objeto cargado debe contener la salida que procesa los mensajes con ese nombre de formato. Se recomienda que el nombre de salida y el nombre del objeto que contiene la salida sean idénticos, aunque no todos los entornos lo requieran.
6. Se carga una nueva copia de la salida cuando una aplicación intenta recuperar el primer mensaje que utiliza ese *MDFMT* desde que la aplicación se conectó al gestor de colas. Una nueva copia también se puede cargar en otras ocasiones, si el gestor de colas ha descartado una copia cargada anteriormente. Por esta razón, una salida no debe intentar utilizar el almacenamiento estático para comunicar información de una invocación de la salida a la siguiente-la salida se puede descargar entre las dos invocaciones.
7. Si hay una salida proporcionada por el usuario con el mismo nombre que uno de los formatos incorporados soportados por el gestor de colas, la salida proporcionada por el usuario no sustituye la rutina de conversión incorporada. Las únicas circunstancias en las que se invoca una salida de este tipo son:
 - Si la rutina de conversión incorporada no puede manejar conversiones a o desde los *MDCSI* o *MDENC* implicados, o
 - Si la rutina de conversión incorporada no ha podido convertir los datos (por ejemplo, porque hay un campo o carácter que no se puede convertir).
8. El ámbito de la salida depende del entorno. Los nombres de *MDFMT* deben elegirse para minimizar el riesgo de conflictos con otros formatos. Se recomienda que empiecen por caracteres que identifiquen la aplicación que define el nombre de formato.
9. La salida de conversión de datos se ejecuta en un entorno como el del programa que ha emitido la llamada MQGET; el entorno incluye el espacio de direcciones y el perfil de usuario (si procede). El programa podría ser un agente de canal de mensajes que envía mensajes a un gestor de colas de destino que no soporta la conversión de mensajes. La salida no puede comprometer la integridad del gestor de colas, ya que no se ejecuta en el entorno del gestor de colas.
10. La única llamada de « MQI » que puede utilizar la salida es MQXCNCV; si se intenta utilizar otras llamadas de « MQI », se produce un error con el código de motivo « RC2219 » u otros errores impredecibles.
11. El gestor de colas no proporciona ningún punto de entrada denominado MQCONVX. El nombre de la salida debe ser el mismo que el nombre de formato (el nombre contenido en el campo *MDFMT* en MQMD), aunque esto no es necesario en todos los entornos.

Parámetros

La llamada MQCONVX tiene los parámetros siguientes:

MQDXP (MQDXP)-entrada/salida

Bloque de parámetros de salida de conversión de datos.

Esta estructura contiene información relacionada con la invocación de la salida. La salida establece información en esta estructura para indicar el resultado de la conversión. Consulte “MQDXP (parámetro de salida de conversión de datos) en IBM i” en la página 1464 para obtener detalles de los campos de esta estructura.

MQMD (MQMD)-entrada/salida

Descriptor de mensaje.

En la entrada a la salida, este es el descriptor de mensaje que se devolvería a la aplicación si no se realizara ninguna conversión. Por lo tanto, contiene *MDFMT*, *MDENC* y *MDCSI* del mensaje no convertido contenido en *INBUF*.

Nota: El parámetro **MQMD** pasado a la salida es siempre la versión más reciente de MQMD soportada por el gestor de colas que invoca la salida. Si la salida está pensada para ser portable entre distintos entornos, la salida debe comprobar el campo *MDVER* en *MQMD* para verificar que los campos a los que la salida necesita acceder están presentes en la estructura.

En IBM i, a la salida se le pasa un MQMD de version-2 .

En la salida, la salida debe cambiar los campos *MDENC* y *MDCSI* por los valores solicitados por la aplicación, si la conversión ha sido satisfactoria; estos cambios se reflejarán de nuevo en la aplicación. Cualquier otro cambio que realice la salida en la estructura se ignorará; no se reflejarán de nuevo en la aplicación.

Si la salida devuelve XROK en el campo *DXRES* de la estructura MQDXP, pero no cambia los campos *MDENC* o *MDCSI* del descriptor de mensaje, el gestor de colas devuelve para esos campos los valores que los campos correspondientes de la estructura MQDXP tenían en la entrada de la salida.

INLEN (entero con signo de 10 dígitos)-entrada

Longitud en bytes de *INBUF*.

Es la longitud del almacenamiento intermedio de entrada *INBUF* y especifica el número de bytes que debe procesar la salida. *INLEN* es la longitud menor de los datos del mensaje antes de la conversión y la longitud del almacenamiento intermedio proporcionado por la aplicación en la llamada MQGET.

El valor es siempre mayor que cero.

INBUF (serie de bits de 1 byte x INLEN)-entrada

Almacenamiento intermedio que contiene el mensaje no convertido.

Contiene los datos del mensaje antes de la conversión. Si la salida no puede convertir los datos, el gestor de colas devuelve el contenido de este almacenamiento intermedio a la aplicación una vez completada la salida.

Nota: La salida no debe alterar *INBUF* ; si se altera este parámetro, los resultados no están definidos.

OUTLEN (entero con signo de 10 dígitos)-entrada

Longitud en bytes de *OUTBUF*.

Es la longitud del almacenamiento intermedio de salida *OUTBUF* y es la misma que la longitud del almacenamiento intermedio proporcionado por la aplicación en la llamada MQGET.

El valor es siempre mayor que cero.

OUTBUF (serie de bits de 1 byte x OUTLEN)-salida

Almacenamiento intermedio que contiene el mensaje convertido.

En la salida de la salida, si la conversión ha sido satisfactoria (tal como indica el valor XROK en el campo *DXRES* del parámetro **MQDXP**), **OUTBUF** contiene los datos de mensaje que se van a entregar a la aplicación, en la representación solicitada. Si la conversión no ha sido satisfactoria, se ignoran los cambios que la salida ha realizado en este almacenamiento intermedio.

Invocación de RPG (ILE)

```
C*.1.....2.....3.....4.....5.....6.....7..  
C          CALLP      exitname(MQDXP : MQMD : INLEN :  
C                               INBUF : OUTLEN : OUTBUF)
```

La definición de prototipo para la llamada es:

```
D*.1.....2.....3.....4.....5.....6.....7..  
Dexitname      PR          EXTPROC('exitname')  
D* Data-conversion exit parameter block  
D MQDXP              44A  
D* Message descriptor  
D MQMD              364A  
D* Length in bytes of INBUF  
D INLEN              10I 0 VALUE  
D* Buffer containing the unconverted message  
D INBUF              *   VALUE  
D* Length in bytes of OUTBUF  
D OUTLEN             10I 0 VALUE  
D* Buffer containing the converted message  
D OUTBUF             *   VALUE
```

Fin de la interfaz de programación sensible al producto

Referencia de salidas de usuarios, salidas de API y servicios instalables

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

- [“Estructura MQIEP” en la página 1478](#)
- [“Referencia de salida de conversión de datos” en la página 1482](#)
- [“MQ_PUBLISH_EXIT - Salida de publicación” en la página 1486](#)
- [“Llamadas de salida de canal y estructuras de datos” en la página 1494](#)
- [“Referencia a la salida de la API” en la página 1587](#)
- [“Información de consulta sobre la interfaz de servicios instalables” en la página 1646](#)

Conceptos relacionados

[Salidas de usuario, salidas de API y servicios instalables de IBM MQ](#)

Tareas relacionadas

[Extensión de recursos del gestor de colas](#)

Estructura MQIEP

La estructura MQIEP contiene un punto de entrada para cada llamada de función que se permite realizar a las salidas.

Campos

StrucId

Tipo: MQCHAR4 - entrada

Identificador de estructura. El valor es el siguiente:

MQIEP_STRUC_ID

Versión

Tipo: MQLONG - entrada

Número de versión de la estructura. El valor es el siguiente:

MQIEP_VERSION_1

Número de versión de la estructura de la versión 1.

MQIEP_VERSION_ACTUAL

Versión actual de la estructura.

StrucLength

Tipo MQLONG

Tamaño de la estructura MQIEP en bytes. El valor es el siguiente:

MQIEP_LENGTH_1**Flags**

Tipo MQLONG

Proporciona información sobre las direcciones de las funciones. Se puede utilizar una bandera para indicar si la biblioteca es roscada con una bandera para indicar si la biblioteca es una biblioteca cliente o servidor.

El siguiente valor se utiliza para especificar que no hay información de la biblioteca:

MQIEPF_NINGUNO

Uno de los siguientes valores se utiliza para especificar si la biblioteca compartida es threaded o no threaded:

MQIEPF_BIBLIOTECA_NO_MULTITHREAD

Una biblioteca compartida sin hilos

MQIEPF_LIBRERÍA_MULTITHREAD

Una biblioteca compartida con subprocesos

Se utiliza uno de los siguientes valores para especificar si la biblioteca compartida es una biblioteca compartida de cliente o de servidor:

MQIEPF_CLIENT_LIBRARY

Una biblioteca compartida por el cliente

MQIEPF_BIBLIOTECA_LOCAL

Una biblioteca compartida de servidor

Reserved

Tipo: MQPTR

MQBACK_Call

Tipo: PMQ_BACK_CALL

Dirección de la llamada MQBACK.

MQBEGIN_Call

Tipo: PMQ_BEGIN_CALL

Dirección de la llamada MQBEGIN.

MQBUFMH_Llamada

Tipo: PMQ_BUFMH_CALL

Dirección de la llamada MQBUFMH.

MQCB_Llamada

Tipo: PMQ_CB_CALL

Dirección de la llamada MQCB.

MQCLOSE_Call

Tipo: PMQ_CLOSE_CALL

Dirección de la llamada MQCLOSE.

MQCMIT_Llamada

Tipo: PMQ_CMIT_CALL

Dirección de la llamada MQCMIT.

MQCONN_Call

Tipo: PMQ_CONN_CALL

Dirección de la llamada MQCONN.

MQCONNX_Call

Tipo: PMQ_CONNX_CALL

Dirección de la llamada MQCONNX.

MQCRTMH_Llamada

Tipo: PMQ_CRTMH_CALL

Dirección de la llamada MQCRTMH.

MQCTL_Call

Tipo: PMQ_CTL_CALL

Dirección de la llamada MQCTL.

MQDISC_Call

Tipo: PMQ_DISC_CALL

Dirección de la llamada MQDISC.

MQDLTMH_Llamada

Tipo: PMQ_DLTMH_CALL

Dirección de la llamada MQDLTMH.

MQDLTMP_Llamada

Tipo: PMQ_DLTMP_CALL

Dirección de la llamada MQDLTMP.

MQGET_Call

Tipo: PMQ_GET_CALL

Dirección de la llamada MQGET.

MQINQ_Llamada

Tipo: PMQ_INQ_CALL

Dirección de la llamada MQINQ.

MQINQMP_Llamada

Tipo: PMQ_INQMP_CALL

Dirección de la llamada MQINQMP.

MQMHBUF_Llamada

Tipo: PMQ_MHBUF_CALL

Dirección de la llamada MQMHBUF.

MQOPEN_Call

Tipo: PMQ_OPEN_CALL

Dirección de la llamada MQOPEN.

MQPUT_Call

Tipo: PMQ_PUT_CALL

Dirección de la llamada MQPUT.

MQPUT1_Call

Tipo: PMQ_PUT1_CALL

Dirección de la llamada MQPUT1.

MQSET_Call

Tipo: PMQ_SET_CALL

Dirección de la llamada MQSET.

MQSETMP_Call

Tipo: PMQ_SETMP_CALL

Dirección de la llamada MQSETMP.

MQSTAT_Call

Tipo: PMQ_STAT_CALL

Dirección de la llamada a MQSTAT.

MQSUB_Call

Tipo: PMQ_SUB_CALL

Dirección de la llamada MQSUB.

MQSUBRQ_Llamada

Tipo: PMQ_SUBRQ_CALL

Dirección de la llamada MQSUBRQ.

MQXCNVC_Llamada

Tipo: PMQ_XCNVC_CALL

Dirección de la llamada MQXCNVC.

MQXCLWLN_Llamada

Tipo: PMQ_XCLWLN_CALL

Dirección de la llamada MQXCLWLN.

MQXDX_Call

Tipo: PMQ_XDX_CALL

Dirección de la llamada MQXDX.

MQXEP_Call

Tipo: PMQ_XEP_CALL

Dirección de la llamada MQXEP.

MQZEP_Llamada

Tipo: PMQ_ZEP_CALL

Dirección de la llamada MQZEP.

Declaración C

```
struct tagMQIEP {
    MQCHAR4      StrucId;          /* Structure identifier */
    MQLONG       Version;          /* Structure version number */
    MQLONG       StrucLength;      /* Structure length */
    MQLONG       Flags;            /* Flags */
    MQPTR        Reserved;         /* Reserved */
    PMQ_BACK_CALL MQBACK_Call;     /* Address of MQBACK */
    PMQ_BEGIN_CALL MQBEGIN_Call;   /* Address of MQBEGIN */
    PMQ_BUFMH_CALL MQBUFMH_Call;   /* Address of MQBUFMH */
    PMQ_CB_CALL  MQCB_Call;        /* Address of MQCB */
    PMQ_CLOSE_CALL MQCLOSE_Call;   /* Address of MQCLOSE */
    PMQ_CMIT_CALL MQCMIT_Call;     /* Address of MQCMIT */
    PMQ_CONN_CALL MQCONN_Call;     /* Address of MQCONN */
    PMQ_CONNX_CALL MQCONNX_Call;   /* Address of MQCONNX */
    PMQ_CRTMH_CALL MQCRTMH_Call;   /* Address of MQCRTMH */
    PMQ_CTL_CALL  MQCTL_Call;      /* Address of MQCTL */
    PMQ_DISC_CALL MQDISC_Call;     /* Address of MQDISC */
    PMQ_DLTMH_CALL MQDLTMH_Call;   /* Address of MQDLTMH */
    PMQ_DLTMP_CALL MQDLTMP_Call;   /* Address of MQDLTMP */
    PMQ_GET_CALL  MQGET_Call;      /* Address of MQGET */
    PMQ_INQ_CALL  MQINQ_Call;      /* Address of MQINQ */
    PMQ_INQMP_CALL MQINQMP_Call;   /* Address of MQINQMP */
    PMQ_MHBUF_CALL MQMHBUF_Call;   /* Address of MQMHBUF */
    PMQ_OPEN_CALL MQOPEN_Call;     /* Address of MQOPEN */
}
```

```

PMQ_PUT_CALL      MQPUT_Call;      /* Address of MQPUT */
PMQ_PUT1_CALL     MQPUT1_Call;     /* Address of MQPUT1 */
PMQ_SET_CALL      MQSET_Call;      /* Address of MQSET */
PMQ_SETMP_CALL    MQSETMP_Call;    /* Address of MQSETMP */
PMQ_STAT_CALL     MQSTAT_Call;     /* Address of MQSTAT */
PMQ_SUB_CALL      MQSUB_Call;      /* Address of MQSUB */
PMQ_SUBRQ_CALL    MQSUBRQ_Call;    /* Address of MQSUBRQ */
PMQ_XCLWLN_CALL   MQXCLWLN_Call;   /* Address of MQXCLWLN */
PMQ_XCNVC_CALL    MQXCNVC_Call;    /* Address of MQXCNVC */
PMQ_XDX_CALL      MQXDX_Call;      /* Address of MQXDX */
PMQ_XEP_CALL      MQXEP_Call;      /* Address of MQXEP */
PMQ_ZEP_CALL      MQZEP_Call;      /* Address of MQZEP */
};

```

Referencia de salida de conversión de datos

Para z/OS, debe escribir salidas de conversión de datos en lenguaje ensamblador. Para otras plataformas, se recomienda utilizar el lenguaje de programación C.

Para ayudarle a crear un programa de salida de conversión de datos, se proporcionan los recursos siguientes:

- Un archivo de origen de esqueleto
- Una llamada de conversión de caracteres
- Un programa de utilidad que crea un fragmento de código que realiza la conversión de datos en estructuras de tipo de datos. Este programa de utilidad sólo toma la entrada C. En z/OS, genera código de ensamblador.

Para el procedimiento para escribir los programas, consulte:

-  [Escritura de un programa de salida de conversión de datos para IBM MQ for IBM i](#)
-  [Escritura de un programa de salida de conversión de datos para IBM MQ for z/OS](#)
- [Escritura de una salida de conversión de datos para sistemas IBM MQ for AIX or Linux](#)
- [Escritura de una salida de conversión de datos para IBM MQ for Windows](#)

Archivo fuente esqueleto

Estos pueden utilizarse como punto de partida a la hora de escribir un programa de salida para la conversión de datos.

Los archivos suministrados se enumeran en [Tabla 852 en la página 1482](#).

Tabla 852. Archivos fuente esqueleto	
Plataforma	Archivo
 AIX	amqsvfc0.c
 IBM i	QMOMSAMP/QCSRC (AMQSVFC4)
 Linux	amqsvfc0.c
Sistemas  Windows	amqsvfc0.c
 z/OS	CSQ4BAX8 (“1” en la página 1483) CSQ4BAX9 (“2” en la página 1483) CSQ4CAX9 (“3” en la página 1483)

Tabla 852. Archivos fuente esqueleto (continuación)

Plataforma	Archivo
Notas: 1. Ilustra la llamada MQXCVNC. 2. Un contenedor para los fragmentos de código generados por la utilidad para su uso en todos los entornos excepto CICS. 3. Un contenedor para los fragmentos de código generados por la utilidad para su uso en el CICS entorno.	

Llamada de conversión de caracteres

Utilice la llamada MQXCNVC (convertir caracteres) desde dentro de un programa de salida de conversión de datos para convertir datos de mensajes de caracteres de un juego de caracteres a otro. Para determinados juegos de caracteres multibyte (por ejemplo, juegos de caracteres UTF-16), se deben utilizar las opciones adecuadas.

No se pueden realizar otras llamadas de tipo « MQI » desde el interior de la salida; cualquier intento de realizar una llamada de este tipo dará lugar a un error con el código de motivo MQRC_CALL_IN_PROGRESS.

Consulte “MQXCNVC - Convertir caracteres” en la [página 927](#) para obtener más información sobre la llamada MQXCNVC y las opciones adecuadas.

Utilidad para crear código de salida de conversión

Utiliza esta información para obtener más información sobre cómo crear código de salida de conversión.

Los comandos para crear el código de salida de conversión son:

	IBM i CVTMQMDTA (Convertir tipo IBM MQ de datos)
	AIX, Linux, and Windows sistemas crtmqcvx (Crear IBM MQ salida de conversión)
	z/OS CSQUCVX

El comando para su plataforma produce un fragmento de código que realiza la conversión de datos en estructuras de tipos de datos, para su uso en su programa de salida de conversión de datos. El comando toma un archivo que contiene una o más definiciones de estructuras en lenguaje C.  A continuación z/OS, genera un conjunto de datos que contiene fragmentos de código ensamblador y funciones de conversión. En otras plataformas, genera un archivo con una función C para convertir cada definición de estructura. En z/OS, la utilidad requiere acceso a la biblioteca de tiempo de ejecución LE/370 SCEERUN.

Invocando la utilidad CSQUCVX en z/OS



[Figura 9 en la página 1484](#) muestra un ejemplo del JCL utilizado para invocar la utilidad CSQUCVX.

```
//CVX      EXEC PGM=CSQUCVX
//STEPLIB DD DISP=SHR,DSN=thlqual.SCSQANLE
//          DD DISP=SHR,DSN=thlqual.SCSQLOAD
//          DD DISP=SHR,DSN=le370qual.SCEERUN
//SYSPRINT DD SYSOUT=*
//CSQUINP  DD DISP=SHR,DSN=MY.MQSERIES.FORMATS(MSG1)
//CSQUOUT  DD DISP=OLD,DSN=MY.MQSERIES.EXIT(S(MSG1)
```

Figura 9. Ejemplo de JCL utilizado para invocar la utilidad CSQUCVX

z/OS sentencias de definición de datos



La utilidad CSQUCVX requiere sentencias DD con los siguientes nombres DD que se muestran en [Tabla 853 en la página 1484](#):

Tabla 853. Nombres y descripciones de las declaraciones de definición de datos	
Sentencia DD	Descripción
SYSPRINT	Especifica un conjunto de datos o una clase de cola de impresión para informes y mensajes de error.
CSQUINP	Especifica el conjunto de datos particionado que contiene las definiciones de las estructuras de datos que se van a convertir.
CSQUOUT	Especifica el conjunto de datos particionado donde se escribirán los fragmentos del código de conversión. La longitud lógica del registro (LRECL) debe ser 80 y el formato del registro (RECFM) debe ser FB.

Mensajes de error en AIX, Linux, and Windows sistemas

El `crtmqcvx` comando devuelve mensajes en el rango AMQ7953 hasta AMQ7970.

Estos mensajes se enumeran en [Mensajes y códigos de motivo IBM MQ Mensajes](#).

Hay dos tipos principales de errores:

- Los errores graves, como los errores de sintaxis, impiden continuar con el procesamiento.
En la pantalla aparece un mensaje que indica el número de línea del error en el archivo de entrada. Es posible que el archivo de salida se haya creado parcialmente.
- Otros errores cuando se muestra un mensaje indicando que se ha encontrado un problema, pero que el análisis de la estructura puede continuar.
El archivo de salida se ha creado y contiene información sobre los errores que se han producido. Esta información de error lleva el prefijo `#error` para que el código generado no sea aceptado por ningún compilador sin una intervención previa para rectificar los problemas.

Sintaxis válida

El archivo de entrada para el programa de utilidad debe ajustarse a la sintaxis del lenguaje C.

Si no está familiarizado con C, consulte el [ejemplo C](#) de este tema.

Además, tenga en cuenta las reglas siguientes:

- `typedef` sólo se reconoce antes de la palabra clave `struct`.
- Se necesita un código de estructura en las declaraciones de estructura.
- Puede utilizar corchetes vacíos `[]` para indicar una serie o matriz de longitud variable al final de un mensaje.

- Las matrices multidimensionales y las matrices de series no están soportadas.
- Se reconocen los siguientes tipos de datos adicionales:

- MQBOOL
- MQBYTE
- MQCHAR
- MQFLOAT32
- MQFLOAT64
- MQSHORT
- MQLONG
- MQINT8
- MQUINT8
- MQINT16
- MQUINT16
- MQINT32
- MQUINT32
- MQINT64
- MQUINT64

Los campos MQCHAR se convierten en página de códigos, pero MQBYTE, MQINT8 y MQUINT8 se dejan intactos. Si la codificación es diferente, MQSHORT, MQLONG, MQINT16, MQUINT16, MQINT32, MQUINT32, MQINT64, MQUINT64, MQFLOAT32, MQFLOAT64 y MQBOOL se convierten en consecuencia.

- No utilice los siguientes tipos de datos:
 - double
 - Punteros
 - campos de bits

Esto se debe a que el programa de utilidad para crear el código de salida de conversión no proporciona el recurso para convertir estos tipos de datos. Para superar esto, puedes escribir tus propias rutinas y llamarlas desde la salida.

Otros puntos a tener en cuenta:

- No utilice números de secuencia en el conjunto de datos de entrada.
- Si hay campos para los que desea proporcionar sus propias rutinas de conversión, declárelos como MQBYTE y, a continuación, sustituya las macros CMQXCFBA generadas por su propio código de conversión.

Ejemplo de C

```
struct TEST { MQLONG    SERIAL_NUMBER;
              MQCHAR     ID[5];
              MQINT16     VERSION;
              MQBYTE      CODE[4];
              MQLONG      DIMENSIONS[3];
              MQCHAR      NAME[24];
            } ;
```

Esto corresponde a las siguientes declaraciones en otros lenguajes de programación:

COBOL

```
10 TEST.
15 SERIAL-NUMBER PIC S9(9) BINARY.
```

```

15 ID          PIC X(5).
15 VERSION     PIC S9(4) BINARY.
* CODE IS NOT TO BE CONVERTED
15 CODE        PIC X(4).
15 DIMENSIONS  PIC S9(9) BINARY OCCURS 3 TIMES.
15 NAME        PIC X(24).

```

System/390

```

TEST          EQU *
SERIAL_NUMBER DS F
ID            DS CL5
VERSION       DS H
CODE          DS XL4
DIMENSIONS    DS 3F
NAME          DS CL24

```

PL/I

Soportado solo en z/OS

```

DCL 1 TEST,
      2 SERIAL_NUMBER FIXED BIN(31),
      2 ID             CHAR(5),
      2 VERSION        FIXED BIN(15),
      2 CODE           CHAR(4),      /* not to be converted */
      2 DIMENSIONS(3)  FIXED BIN(31),
      2 NAME           CHAR(24);

```

MQ_PUBLISH_EXIT - Salida de publicación

La llamada MQ_PUBLISH_EXIT puede inspeccionar y modificar los mensajes entregados a los suscriptores.

Finalidad

Utilice la salida de publicación para inspeccionar y modificar los mensajes entregados a los suscriptores:

- Examinar el contenido de un mensaje publicado en cada suscriptor.
- Modificar el contenido de un mensaje publicado en cada suscriptor.
- Modificar la cola en la que se coloca el mensaje.
- Detener la entrega de un mensaje a un suscriptor.

Esta salida no está disponible en IBM MQ for z/OS.

Sintaxis

MQ_PUBLISH_EXIT (*ExitParms*, *PubContext*, *SubContext*)

Parámetros

ExitParms (MQPSXP) - Input/Output

ExitParms contiene información sobre la invocación de la salida.

PubContext (MQPBC) - Input

PubContext contiene información contextual sobre el editor de la publicación.

SubContext (MQSBC) - Input/Output

SubContext contiene información contextual sobre el suscriptor que recibe la publicación.

MQPSXP-Estructura de datos de salida de publicación

La estructura MQPSXP describe la información que se pasa a y se devuelve de la salida de publicación.

[Tabla 854 en la página 1487](#) resume los campos de la estructura:

Tabla 854. Campos en MQPSXP	
Campo	Descripción
<u>StrucID</u>	Identificador de la estructura
<u>Version</u>	Número de versión de la estructura
<u>ExitId</u>	Tipo de salida que se está llamando
<u>ExitReason</u>	Razón para llamar a la salida
<u>ExitResponse</u>	Respuesta de la salida
<u>ExitResponse2</u>	Respuesta secundaria de salida
<u>Feedback</u>	Código de retroalimentación
<u>ExitUserArea</u>	Salir del área de usuario
<u>ExitData</u>	Datos de salida
<u>QMgrName</u>	Nombre del gestor de colas local
<u>Hconn</u>	Descriptor de contexto de conexión
<u>MsgDescPtr</u>	Dirección del descriptor de mensaje (MQMD)
<u>MsgHandle</u>	Descriptor de contexto para propiedades de mensaje (MQHMSG)
<u>MsgInPtr</u>	Dirección del mensaje de entrada
<u>MsgInLength</u>	Longitud del mensaje de entrada
<u>MsgOutPtr</u>	Dirección del mensaje de salida
<u>MsgOutLength</u>	Longitud del mensaje de salida
<u>pEntryPoints</u>	Dirección de la estructura MQIEP

Campos

StrucID (MQCHAR4)

StrucID es el identificador de estructura. El valor es el siguiente:

MQPSXP_STRUCID

MQPSXP_STRUCID es el identificador de la estructura de parámetros de salida de publicación. Para el lenguaje de programación C, la constante MQPSXP_STRUC_ID_ARRAY también está definida; tiene el mismo valor que MQPSXP_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

StrucID es un campo de entrada para la salida.

Version (MQLONG)

Version es el número de versión de la estructura. El valor es el siguiente:

MQPSXP_VERSION_1

MQPSXP_VERSION_1 es la estructura del parámetro de salida de publicación de la versión 1. La constante MQPSXP_CURRENT_VERSION también se define con el mismo valor.

Version es un campo de entrada para la salida.

ExitId (MQLONG)

ExitId es el tipo de salida que se está llamando. El valor es el siguiente:

MQXT_PUBLISH_EXIT

Salida de publicación.

ExitId es un campo de entrada para la salida.

***ExitReason* (MQLONG)**

ExitReason es la razón por la que se llama a la salida. Los valores posibles son:

MQXR_INIT

Se llama a la salida para esta conexión para la inicialización. La salida puede adquirir e inicializar los recursos que necesita; por ejemplo, el almacenamiento principal.

MQXR_TERM

Se llama a la salida para esta conexión porque la salida está a punto de detenerse. La salida debe liberar los recursos que haya adquirido desde que se inicializó; por ejemplo, el almacenamiento principal.

MQXR_PUBLICATION

El gestor de colas llama a la salida antes de colocar una publicación en una cola de mensajes de un suscriptor. La salida puede cambiar el mensaje, no colocar el mensaje en la cola o detener la publicación.

ExitReason es un campo de entrada para la salida.

***ExitResponse* (MQLONG)**

Establezca *ExitResponse* en la salida para especificar cómo debe continuar el proceso.

ExitResponse es uno de los valores siguientes:

MQXCC_OK

Establezca MQXCC_OK para continuar procesando normalmente. Establezca MQXCC_OK en respuesta a cualquier valor de *ExitReason*.

Si *ExitReason* tiene el valor MQXR_PUBLICATION, los campos *DestinationQName* y *DestinationQMgrName* de la estructura MQSBC identifican el destino al que se envía el mensaje.

MQXCC_FAILED

Establezca MQXCC_FAILED para detener la operación de publicación. El código de terminación MQCC_FAILED y el código de razón [2557 \(09FD\) \(RC2557\): MQRC_PUBLISH_EXIT_ERROR](#) se establece en el retorno de la salida.

MQXCC_SUPPRESS_FUNCTION

Establezca MQXCC_SUPPRESS_FUNCTION para detener el proceso normal del mensaje. Solo establezca MQXCC_SUPPRESS_FUNCTION si *ExitReason* tiene el valor MQXR_PUBLICATION.

El gestor de colas sigue procesando el mensaje de acuerdo con la opción MQRO_DISCARD_MSG del campo *Report* del descriptor de mensaje del mensaje.

- Si se especifica la opción MQRO_DISCARD_MSG, el mensaje no se entrega al suscriptor.
- Si no se especifica la opción MQRO_DISCARD_MSG, el mensaje se coloca en la cola de mensajes no entregados. Si no hay ninguna cola de mensajes no entregados o el mensaje no se puede colocar correctamente en la cola de mensajes no entregados, la publicación no se entrega al suscriptor. La entrega de la publicación a otros suscriptores depende de los valores de los atributos de objeto de tema PMSGDLV y NPMSGDLV. Para obtener una explicación de estos atributos, consulte las descripciones de los parámetros para el mandato [DEFINE TOPIC](#).

ExitResponse es un campo de salida de la salida.

***ExitResponse2* (MQLONG)**

ExitResponse2 está reservado para uso futuro.

***Feedback* (MQLONG)**

Feedback es el código de comentarios que se debe utilizar si la salida devuelve MQXCC_SUPPRESS_FUNCTION en *ExitResponse*.

En la entrada a la salida, *Feedback* siempre tiene el valor MQFB_NONE. Si la salida devuelve MQXCC_SUPPRESS_FUNCTION, establezca *Feedback* en el valor que se utilizará para el mensaje cuando el gestor de colas lo coloque en la cola de mensajes no entregados. Al volver de la salida, si *Feedback* tiene el valor original MQFB_NONE, el gestor de colas establece *Feedback* en MQFB_STOPPED_BY_PUBSUB_EXIT.

Feedback es un campo de entrada/salida para la salida.

ExitUserArea (MQBYTE16)

ExitUserArea es un campo que está disponible para que lo utilice la salida. Cada conexión tiene un *ExitUserArea* independiente. La longitud de *ExitUserArea* la proporciona MQ_EXIT_USER_AREA_LENGTH.

El campo *ExitReason* tiene el valor MQXR_INIT en la primera invocación de la salida. *ExitUserArea* se inicializa en MQXUA_NONE en la primera invocación de la salida para una conexión. Los cambios posteriores en *ExitUserArea* se conservan entre invocaciones de la salida.

ExitUserArea es un campo de entrada/salida para la salida.

ExitData (MQCHAR32)

ExitData son datos de salida fijos definidos por el parámetro **PublishExitData** de la stanza en el archivo de inicialización del gestor de colas. Los datos se rellenan con espacios en blanco hasta la longitud completa del campo. Si no hay datos de salida fijos definidos en el archivo de inicialización, *ExitData* está en blanco. La longitud de *ExitData* la proporciona MQ_EXIT_DATA_LENGTH.

ExitData es un campo de entrada para la salida.

QMgrName (MQCHAR48)

QMgrName es el nombre del gestor de colas local. El nombre se rellena con espacios en blanco hasta la longitud completa del campo. La longitud de este campo la proporciona MQ_Q_MGR_NAME_LENGTH.

QMgrName es un campo de entrada para la salida.

Hconn (MQHCONN)

Hconn es el descriptor de contexto que representa una conexión con el gestor de colas. Utilice sólo *Hconn* como parámetro para las llamadas a la función de propiedad de mensaje MQSETMP, MQINQMMPo MQDLTMP para trabajar con propiedades de mensaje.

Hconn es un campo de entrada para la salida.

MsgDescPtr (PMQMD)

MsgDescPtr es la dirección del descriptor de mensaje (MQMD) del mensaje que se está procesando y es una copia del MQMD devuelto por la llamada MQPUT. La salida puede cambiar el contenido del descriptor de mensaje. Cualquier cambio en el contenido del descriptor de mensaje debe realizarse con cuidado. En concreto, en el caso en el que el campo *SubType* de la estructura MQSBC es de valor MQSUBTYPE_PROXY, el campo *CorrelId* del descriptor de mensaje no debe cambiarse.

No se pasa ningún descriptor de mensaje a la salida si *ExitReason* es MQXR_INIT o MQXR_TERM; en estos casos, *MsgDescPtr* es el puntero nulo.

MsgDescPtr es un campo de entrada para la salida.

MsgHandle (MQHMSG)

MsgHandle es el descriptor de contexto de las propiedades de mensaje. Utilice sólo *MsgHandle* con las llamadas de función de propiedad de mensaje MQSETMP, MQINQMMPo MQDLTMP para trabajar con propiedades de mensaje.

MsgHandle es un campo de entrada para la salida.

MsgInPtr (PMQVOID)

MsgInPtr es la dirección de los datos del mensaje de entrada. El contenido del almacenamiento intermedio direccionado por *MsgInPtr* puede ser modificado por la salida; consulte [*MsgOutPtr*](#).

MsgInPtr es un campo de entrada para la salida.

MsgInLength (MQLONG)

MsgInLength es la longitud en bytes de los datos de mensaje pasados a la salida. La dirección de los datos la proporciona *MsgInPtr*.

MsgInLength es un campo de entrada para la salida.

MsgOutPtr (PMQVOID)

MsgOutPtr es la dirección de un almacenamiento intermedio que contiene datos de mensaje que se devuelven de la salida. En la entrada a la salida, *MsgOutPtr* es nulo. Al volver de la salida, si el valor

sigue siendo nulo, el gestor de colas envía el mensaje especificado por *MsgInPtr*, con la longitud proporcionada por *MsgInLength*.

Si la salida modifica los datos del mensaje, utilice uno de los procedimientos siguientes:

- Si la longitud de los datos no cambia, los datos se pueden modificar en el almacenamiento intermedio direccionado por *MsgInPtr*. En este caso, no cambie *MsgOutPtr* y *MsgOutLength*.
- Si los datos modificados son más cortos que los datos originales, los datos se pueden modificar en el almacenamiento intermedio direccionado por *MsgInPtr*. En este caso, *MsgOutPtr* debe establecerse en la dirección del almacenamiento intermedio de mensajes de entrada y *MsgOutLength* debe establecerse en la nueva longitud de los datos del mensaje.
- Si los datos modificados son, o pueden ser, más largos que los datos originales, la salida debe obtener un nuevo almacenamiento intermedio de mensajes. Copie los datos modificados en él. Establezca *MsgOutPtr* en la dirección del nuevo almacenamiento intermedio y establezca *MsgOutLength* en la longitud de los nuevos datos de mensaje. La salida es responsable de liberar el almacenamiento intermedio direccionado por *MsgOutPtr* cuando se llama a la salida a continuación.

Nota: *MsgOutPtr* es siempre el puntero nulo en la entrada a la salida, y no la dirección de un almacenamiento intermedio de mensajes obtenido anteriormente. Para liberar el almacenamiento intermedio obtenido anteriormente, la salida debe guardar su dirección y longitud. Guarde la información en *ExitUserArea* en un bloque de control que tenga su dirección guardada en *ExitUserArea*.

MsgOutPtr es un campo de entrada/salida para la salida.

MsgOutLength (MQLONG)

MsgOutLength es la longitud en bytes de los datos de mensaje devueltos por la salida. En la entrada a la salida, este campo es siempre cero. Al volver de la salida, este campo se ignora si *MsgOutPtr* es nulo. Consulte [MsgOutPtr](#) para obtener información sobre cómo modificar los datos del mensaje.

MsgOutLength es un campo de entrada/salida para la salida.

pEntryPoints (PMQIEP)

pEntryPoints es la dirección de una estructura MQIEP a través de la cual se pueden realizar llamadas de tipo « MQI » y « DCI ».

Declaración de lenguaje C-MQPSXP

```
typedef struct tagMQPSXP {
    MQCHAR4      StrucId;          /* Structure identifier */
    MQLONG       Version;         /* Structure version number */
    MQLONG       ExitId;          /* Type of exit */
    MQLONG       ExitReason;      /* Reason for invoking exit */
    MQLONG       ExitResponse;    /* Response from exit */
    MQLONG       ExitResponse2;   /* Reserved */
    MQLONG       Feedback;        /* Feedback code */
    MQBYTE16     ExitUserArea;    /* Exit user area */
    MQCHAR32     ExitData;        /* Exit data */
    MQCHAR48     QMgrName;        /* Name of local queue manager */
    MQHCONN      Hconn;          /* Connection handle */
    MQHMSG       MsgHandle;       /* Handle to message properties */
    PMQMD        MsgDescPtr;      /* Address of message descriptor */
    PMQV0ID      MsgInPtr;        /* Address of input message data */
    MQLONG       MsgInLength;     /* Length of input message data */
    PMQV0ID      MsgOutPtr;       /* Address of output message data */
    MQLONG       MsgOutLength;    /* Length of output message data */
    /* Ver:1 */
    PMQIEP       pEntryPoints;    /* Address of the MQIEP structure */
    /* Ver:2 */
} MQPSXP;
```

MQPBC-Estructura de datos de contexto de publicación

La estructura MQPBC contiene la información contextual, relacionada con el publicador de la publicación, que se pasa a la salida de publicación.

Tabla 855 en la página 1491 resume los campos de la estructura:

Tabla 855. Campos en MQPBC	
Campo	Descripción
<u>StrucID</u>	Identificador de la estructura
<u>Version</u>	Número de versión de la estructura
<u>PubTopicString</u>	Serie de tema de publicación
<u>MsgDescPtr</u>	Dirección del descriptor de mensaje (MQMD)

Campos

StrucID (MQCHAR4)

StrucID es el identificador de estructura. El valor es el siguiente:

MQPBC_STRUCID

MQPBC_STRUCID es el identificador de la estructura de contexto de publicación. Para el lenguaje de programación C, la constante MQPBC_STRUC_ID_ARRAY también está definida; tiene el mismo valor que MQPBC_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

StrucID es un campo de entrada para la salida.

Version (MQLONG)

Version es el número de versión de la estructura. El valor es el siguiente:

MQPBC_VERSION_1

MQPBC_VERSION_1 es la estructura del parámetro de salida de publicación de la versión 1.

MQPBC_VERSION_2

MQPBC_VERSION_2 es la estructura de parámetros de salida de publicación de la versión 2. La constante MQPBC_CURRENT_VERSION también se define con el mismo valor.

Version es un campo de entrada para la salida.

PubTopicString (MQCHARV)

PubTopicString es la serie de tema en la que se publica.

PubTopicString es un campo de entrada para la salida.

MsgDescPtr (PMQMD)

MsgDescPtr es la dirección de una copia del descriptor de mensaje (MQMD) para el mensaje que se está procesando.

MsgDescPtr es un campo de entrada para la salida.

Declaración de lenguaje C-MQPBC

```
typedef struct tagMQPBC {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQCHARV    PubTopicString;   /* Publish topic string */
    PMQMD      MsgDescPtr;       /* Address of message descriptor */
} MQPBC;
```

MQSBC - Estructura de datos del contexto de suscripción

La MQSBC estructura contiene la información contextual relacionada con el suscriptor que recibe la publicación, que se transfiere a la salida de publicación.

Tabla 856 en la página 1492 resume los campos de la estructura:

Tabla 856. Entrada de campos MQSBC

Campo	Descripción
<u>StrucID</u>	Identificador de la estructura
<u>Version</u>	Número de versión de la estructura
<u>DestinationQMgrName</u>	Nombre del gestor de colas de destino
<u>DestinationQName</u>	Nombre de la cola de destino
<u>SubType</u>	Tipo de suscripción
<u>SubOptions</u>	Opciones de suscripción
<u>ObjectName</u>	Nombre de objeto
<u>ObjectString</u>	Cadena de objetos
<u>SubTopicString</u>	Cadena de tema de suscripción
<u>SubName</u>	Nombre de suscripción
<u>SubId</u>	Identificador de suscripción
<u>SelectionString</u>	Dirección de la cadena de selección
<u>SubLevel</u>	Nivel de suscripción
<u>PSPProperties</u>	Propiedades de publicación/suscripción

Campos

StrucID (MQCHAR4)

Identificador de estructura. El valor es el siguiente:

MQSBC_STRUCID

MQSBC_STRUCID es el identificador de la estructura del parámetro de salida de publicación.

Para el lenguaje de programación C, también se MQSBC_STRUC_ID_ARRAY define la constante;

MQSBC_STRUC_ID_ARRAY tiene el mismo valor que MQSBC_STRUC_ID, pero es una matriz de caracteres en lugar de una cadena.

StrucID es un campo de entrada a la salida.

Version (MQLONG)

Número de versión de la estructura. El valor es el siguiente:

MQSBC_VERSION_1

Versión 1: publicar la estructura del parámetro de salida. La constante también

MQSBC_CURRENT_VERSION se define con el mismo valor.

Version es un campo de entrada a la salida.

DestinationQMgrName (MQCHAR48)

DestinationQMgrName es el nombre del gestor de colas al que se envía el mensaje. El nombre se rellena con espacios en blanco hasta completar la longitud del campo. El nombre puede modificarse al salir. La longitud de este campo viene dada por MQ_Q_MGR_NAME_LENGTH.

DestinationQMgrName es un campo de entrada/salida a la salida; véase [la nota](#).

DestinationQName (MQCHAR48)

DestinationQName es el nombre de la cola a la que se envía el mensaje. El nombre se rellena con espacios en blanco hasta completar la longitud del campo. El nombre puede modificarse al salir. La longitud de este campo viene dada por MQ_Q_NAME_LENGTH.

DestinationQName es un campo de entrada/salida a la salida; véase [la nota](#).

SubType (MQLONG)

SubType indica cómo se creó la suscripción. Los valores válidos son MQSUBTYPE_API, MQSUBTYPE_ADMIN y MQSUBTYPE_PROXY ; consulte [Consultar el estado de la suscripción \(respuesta\)](#).

SubType es un campo de entrada a la salida.

SubOptions (MQLONG)

SubOptions son las opciones de suscripción; consulte [“Opciones \(MQLONG\)”](#) en la página 564 para obtener una descripción de los valores que puede tomar este campo.

SubOptions es un campo de entrada a la salida.

ObjectName (MQCHAR48)

ObjectName es el nombre del objeto del tema tal y como se define en el gestor de colas local. La longitud de este campo viene dada por MQ_TOPIC_NAME_LENGTH. El nombre del objeto es el nombre del objeto temático administrativo que el gestor de colas ha asociado a la cadena temática. Incluso si el suscriptor proporcionó un objeto de tema como parte de la suscripción, el *ObjectName* podría ser un objeto de tema diferente. La asociación de un objeto temático con una suscripción depende de la resolución completa de *SubTopicString*.

ObjectName es un campo de entrada a la salida.

ObjectString (MQCHARV)

ObjectString es la cadena completa del tema de la publicación a la que se suscribió. Se resuelven todos los comodines de la cadena de suscripción original. Es diferente al campo MQSD de *ObjectString* suscripción descrito en [“ObjectString \(MQCHARV\)”](#) en la página 574, que puede contener comodines, y es exclusivo de cualquier nombre de objeto proporcionado por el suscriptor.

ObjectString es un campo de entrada a la salida.

SubTopicString (MQCHARV)

SubTopicString es la cadena completa del tema tal y como la ha proporcionado el suscriptor.

SubTopicString es la combinación de la cadena de tema definida en un objeto de tema y una cadena de tema. Un suscriptor debe proporcionar un objeto de tema, una cadena de tema o ambos. Si el suscriptor proporciona una cadena de tema, esta puede contener comodines.

SubTopicString es un campo de entrada a la salida.

SubName (MQCHARV)

SubName Es el nombre de la suscripción proporcionado por el suscriptor o un nombre generado.

SubName es un campo de entrada a la salida.

SubId (MQBYTE 24)

SubId es el identificador único de suscripción interno.

SubId es un campo de entrada a la salida.

SelectionString (MQCHARV)

SelectionString Es el criterio de selección utilizado al suscribirse a los mensajes de un tema; consulte [Selectores](#).

SelectionString es un campo de entrada a la salida.

SubLevel (MQLONG)

SubLevel es el nivel de interceptación asociado a la suscripción; consulte [“SubLevel \(MQLONG\)”](#) en la página 578 para obtener más detalles.

SubLevel es un campo de entrada a la salida.

PSPProperties (MQLONG)

PSPProperties son las propiedades de publicación/suscripción. Especifican cómo se añaden las propiedades de los mensajes relacionados con la publicación/suscripción a los mensajes enviados a esta suscripción. Los valores posibles son MQPSPROP_NONE, MQPSPROP_COMPAT, MQPSPROP_RFH2, MQPSPROP_MSGPROP. Consulte [Parámetros opcionales \(Cambiar, Copiar y Crear suscripción\)](#) para obtener una descripción de estos valores.

PSPProperties es un campo de entrada a la salida.

Nota: Las comprobaciones de autorización solo se realizan en los valores originales de *DestinationQMGrName* y *DestinationQName* antes de que se pasen a la salida de publicación. No se realizan nuevas comprobaciones de autorización cuando la salida cambia la cola de destino, ya sea cambiando *DestinationQMGrName* o *DestinationQName*.

Declaración en lenguaje C - MQSBC

```
typedef struct tagMQSBC {
    MQCHAR4      StructId;           /* Structure identifier */
    MQLONG       Version;            /* Structure version number */
    MQCHAR48     DestinationQMGrName; /* Destination queue manager */
    MQCHAR48     DestinationQName;   /* Destination queue name */
    MQLONG       SubType;            /* Type of subscription */
    MQLONG       SubOptions;         /* Subscription options */
    MQCHAR48     ObjectName;         /* Object name */
    MQCHARV      ObjectString;       /* Object string */
    MQCHARV      SubTopicString;     /* Subscription topic string */
    MQCHARV      SubName;            /* Subscription name */
    MQBYTE24     SubId;              /* Subscription identifier */
    MQCHARV      SelectionString;    /* Subscription selection string */
    MQLONG       SubLevel;           /* Subscription level */
    MQLONG       PSPProperties;       /* Publish/subscribe properties */
} MQSBC;
```

Llamadas de salida de canal y estructuras de datos

Esta colección de temas proporciona información de referencia sobre las llamadas IBM MQ especiales y las estructuras de datos que puede utilizar al escribir programas de salida de canal.

Esta información es información confidencial sobre la interfaz de programación del producto. Puede escribir salidas IBM MQ de usuario en los siguientes lenguajes de programación:

Tabla 857. IBM MQ Salidas de usuario: plataformas y lenguajes de programación	
Plataforma	Lenguajes de programación
IBM MQ for z/OS	Assembler y C (que deben ajustarse al entorno de programación del sistema C para las salidas del sistema, descrito en la <i>Guía de programación z/OS C/C++</i>)
IBM MQ for IBM i	ILE C, ILE COBOL e ILE RPG
Todas las demás IBM MQ plataformas	C    (solo 64 bits en arquitecturas de tipo « x86 »)

También puede escribir salidas de usuario en Java para su uso exclusivo con Java y JMS aplicaciones. Para obtener más información sobre cómo crear y utilizar salidas de canal con IBM MQ classes for Java, consulte [Uso de salidas de canal en IBM MQ classes for Java](#) y para IBM MQ classes for JMS, consulte [Uso de salidas de canal con IBM MQ classes for JMS](#).

No se pueden escribir salidas IBM MQ de usuario en TAL o Visual Basic. Sin embargo, en Visual Basic se proporciona una declaración para la estructura MQCD para su uso en la llamada MQCONN desde un IBM MQ MQI client programa.

En varios casos de las descripciones que siguen, los parámetros son matrices o cadenas de caracteres con un tamaño que no es fijo. Para estos parámetros, se utiliza una "n" minúscula para representar una constante numérica. Cuando se codifica la declaración de ese parámetro, la "n" debe sustituirse por el valor numérico requerido. Para obtener más información sobre las convenciones utilizadas en estas descripciones, consulte el ["Tipos de datos elementales"](#) en la página 177.

archivos de definición de datos

Los archivos de definición de datos se suministran con IBM MQ para cada uno de los lenguajes de programación compatibles. Para obtener más información sobre estos archivos, consulte [IBM MQ Archivos COPY, de encabezado, de inclusión y de módulo](#).

MQ_CHANNEL_EXIT-Salida de canal

La llamada MQ_CHANNEL_EXIT describe los parámetros que se pasan a cada una de las salidas de canal llamadas por el agente de canal de mensajes.

El gestor de colas no proporciona ningún punto de entrada denominado MQ_CHANNEL_EXIT; el nombre MQ_CHANNEL_EXIT no tiene ninguna significación especial puesto que los nombres de las salidas de canal se proporcionan en la definición de canal MQCD.

Hay cinco tipos de salida de canal:

- Salida de seguridad de canal
- Salida de mensajes de canal
- Salida de envío de canal
- Salida de recepción de canal
- Salida de reintento de mensajes de canal

Los parámetros son similares para cada tipo de salida, y la descripción que se proporciona aquí se aplica a todos ellos, excepto cuando se indique específicamente.

Sintaxis

MQ_CHANNEL_EXIT (*ChannelExitParms*, *ChannelDefinition*, *DataLength*, *AgentBufferLength*, *AgentBuffer*, *ExitBufferLength*, *ExitBufferAddr*)

Parámetros

La llamada MQ_CHANNEL_EXIT tiene los parámetros siguientes.

ChannelExitParms (MQCXP) - entrada/salida

Bloque de parámetros de salida de canal.

Esta estructura contiene información adicional relacionada con la invocación de la salida. La salida establece información en esta estructura para indicar cómo procede el MCA.

ChannelDefinition (MQCD)-entrada/salida

Definición de canal.

Esta estructura contiene parámetros establecidos por el administrador para controlar el comportamiento del canal.

DataLength (MQLONG)-entrada/salida

Longitud de los datos.

Los datos dependen del tipo de salida:

- Para una salida de seguridad de canal, cuando se invoca la salida, este parámetro contiene la longitud de cualquier mensaje de seguridad en el campo *AgentBuffer*, si *ExitReason* es MQXR_SEC_MSG. Es cero si no hay ningún mensaje. La salida debe establecer este campo en la longitud de cualquier mensaje de seguridad que se vaya a enviar a su asociado si establece *ExitResponse* en MQXCC_SEND_SEC_MSG o MQXCC_SEND_AND_REQUEST_SEC_MSG. Los datos del mensaje se encuentran en *AgentBuffer* o *ExitBufferAddr*.

El contenido de los mensajes de seguridad es responsabilidad exclusiva de las salidas de seguridad.

- Para una salida de mensajes de canal, cuando se invoca la salida, este parámetro contiene la longitud del mensaje (incluida la cabecera de cola de transmisión). La salida debe establecer este campo en la longitud del mensaje en *AgentBuffer* o *ExitBufferAddr* que va a continuar. Debe ser mayor o igual que la longitud de la cabecera de cola de transmisión (MQXQH).
- Para una salida de emisión o recepción de canal, cuando se invoca la salida, este parámetro contiene la longitud de la transmisión. La salida debe establecer este campo en la longitud de la transmisión en *AgentBuffer* o *ExitBufferAddr* que va a continuar.

Si una salida de seguridad envía un mensaje, y no hay ninguna salida de seguridad en el otro extremo del canal, o el otro extremo establece un *ExitResponse* de MQXCC_OK, la salida de inicio se vuelve a invocar con MQXR_SEC_MSG y una respuesta nula (*DataLength* = 0).

AgentBufferLength (MQLONG) - entrada

Longitud del almacenamiento intermedio del agente.

Este parámetro puede ser mayor que *DataLength* en la invocación.

Para las salidas de envío y recepción de mensajes de canal, la salida puede utilizar cualquier espacio no utilizado en la invocación para expandir los datos en su lugar. Si se hace esto, la salida debe establecer correctamente el parámetro **DataLength**.

En el lenguaje de programación C, este parámetro se pasa por dirección.

AgentBuffer (MQBYTE x AgentBufferLength) - entrada/salida

Almacenamiento intermedio de agente.

El contenido de este parámetro depende del tipo de salida:

- Para una salida de seguridad de canal, al invocar la salida contiene un mensaje de seguridad si *ExitReason* es MQXR_SEC_MSG. Para devolver un mensaje de seguridad, la salida puede utilizar este almacenamiento intermedio o su propio almacenamiento intermedio (*ExitBufferAddr*).
- Para una salida de mensajes de canal, al invocar la salida, este parámetro contiene:
 - La cabecera de cola de transmisión (MQXQH), que incluye el descriptor de mensaje (que a su vez contiene la información de contexto del mensaje), seguida inmediatamente de
 - Los datos del mensaje

Si el mensaje va a continuar, la salida puede realizar una de las acciones siguientes:

- Deje el contenido del almacenamiento intermedio intacto
- Modificar el contenido en su lugar (devolviendo la nueva longitud de los datos en *DataLength*; no debe ser mayor que *AgentBufferLength*)
- Copie el contenido en *ExitBufferAddr*, realizando los cambios necesarios

Los cambios que realiza la salida en la cabecera de cola de transmisión no se comprueban; sin embargo, las modificaciones erróneas pueden significar que el mensaje no se puede colocar en el destino.

- Para una salida de emisión o recepción de canal, al invocar la salida contiene los datos de transmisión. La salida puede realizar una de las acciones siguientes:
 - Deje el contenido del almacenamiento intermedio intacto
 - Modificar el contenido en su lugar (devolviendo la nueva longitud de los datos en *DataLength*; no debe ser mayor que *AgentBufferLength*)
 - Copie el contenido en *ExitBufferAddr*, realizando los cambios necesarios

La salida no debe cambiar los primeros 8 bytes de los datos.

ExitBufferLength (MQLONG) - entrada/salida

Longitud del almacenamiento intermedio de salida.

En la primera invocación de la salida, este parámetro se establece en cero. A partir de entonces, cualquier valor que la salida devuelva, en cada invocación, se presentará a la salida la próxima vez que se invoque. El MCA no utiliza el valor.

Nota: Este parámetro no lo deben utilizar las salidas escritas en lenguajes de programación que no soportan el tipo de datos de puntero.

ExitBufferAddr (MQPTR) - entrada/salida

Dirección del almacenamiento intermedio de salida.

Este parámetro es un puntero a la dirección de un almacenamiento intermedio de almacenamiento gestionado por la salida, donde puede elegir devolver datos de mensaje o transmisión (en función del tipo de salida) al agente si el almacenamiento intermedio del agente es o no es lo suficientemente grande, o si es más conveniente que la salida lo haga.

En la primera invocación de la salida, la dirección pasada a la salida es nula. A partir de entonces, cualquier dirección que la salida devuelva, en cada invocación, se presentará a la salida la próxima vez que se invoque.

Si ExitBufferAddr es nulo, los datos utilizados se toman del parámetro AgentBuffer.

Si ExitBufferAddr no es nulo, los datos utilizados se toman del búfer apuntado por el parámetro ExitBufferAddr.

Nota: Este parámetro no lo deben utilizar las salidas escritas en lenguajes de programación que no dan soporte al tipo de datos de puntero.

Invocación en C

```
exitname (&ChannelExitParms, &ChannelDefinition,  
&DataLength, &AgentBufferLength, AgentBuffer,  
&ExitBufferLength, &ExitBufferAddr);
```

Los parámetros pasados a la salida se declaran de la siguiente manera:

```
MQCXP  ChannelExitParms;    /* Channel exit parameter block */  
MQCD   ChannelDefinition;  /* Channel definition */  
MQLONG DataLength;         /* Length of data */  
MQLONG AgentBufferLength;  /* Length of agent buffer */  
MQBYTE AgentBuffer[n];     /* Agent buffer */  
MQLONG ExitBufferLength;   /* Length of exit buffer */  
MQPTR  ExitBufferAddr;     /* Address of exit buffer */
```

Invocación en COBOL

```
CALL 'exitname' USING CHANNELEXITPARMS, CHANNELDEFINITION,  
                      DATALENGTH, AGENTBUFFERLENGTH, AGENTBUFFER,  
                      EXITBUFFERLENGTH, EXITBUFFERADDR.
```

Los parámetros pasados a la salida se declaran de la siguiente manera:

```
** Channel exit parameter block  
01 CHANNELEXITPARMS.  
   COPY CMQCXPV.  
** Channel definition  
01 CHANNELDEFINITION.  
   COPY CMQCDV.  
** Length of data  
01 DATALENGTH          PIC S9(9) BINARY.  
** Length of agent buffer  
01 AGENTBUFFERLENGTH    PIC S9(9) BINARY.  
** Agent buffer  
01 AGENTBUFFER          PIC X(n).  
** Length of exit buffer  
01 EXITBUFFERLENGTH     PIC S9(9) BINARY.  
** Address of exit buffer  
01 EXITBUFFERADDR       POINTER.
```

Invocación de RPG (ILE)

```
C*.1.....2.....3.....4.....5.....6.....7..  
C          CALLP      exitname(MQCP : MQCD : DATLEN :  
C                               ABUFL : ABUF : EBUFL :  
C                               EBUF)
```

La definición de prototipo para la llamada es:

```
D*.1.....2.....3.....4.....5.....6.....7..  
Dexitname      PR          EXTPROC('exitname')  
D* Channel exit parameter block  
D MQCP          160A  
D* Channel definition  
D MQCD          1328A  
D* Length of data  
D DATLEN        10I 0  
D* Length of agent buffer  
D ABUFL          10I 0  
D* Agent buffer  
D ABUF          *    VALUE  
D* Length of exit buffer  
D EBUFL          10I 0  
D* Address of exit buffer  
D EBUF          *
```

Invocación de ensamblador System/390

```
CALL EXITNAME, (CHANNELEXITPARMS, CHANNELDEFINITION, DATALENGTH, X  
                AGENTBUFFERLENGTH, AGENTBUFFER, EXITBUFFERLENGTH, X  
                EXITBUFFERADDR)
```

Los parámetros pasados a la salida se declaran de la siguiente manera:

CHANNELEXITPARMS	CMQCPA	,	Channel exit parameter block
CHANNELDEFINITION	CMQCD	,	Channel definition
DATALENGTH	DS	F	Length of data
AGENTBUFFERLENGTH	DS	F	Length of agent buffer
AGENTBUFFER	DS	CL(n)	Agent buffer
EXITBUFFERLENGTH	DS	F	Length of exit buffer
EXITBUFFERADDR	DS	F	Address of exit buffer

Notas de uso

1. La función realizada por la salida de canal la define el proveedor de la salida. Sin embargo, la salida debe ajustarse a las reglas definidas aquí y en el bloque de control asociado, MQCP.
2. El parámetro **ChannelDefinition** pasado a la salida de canal puede ser una de varias versiones. Consulte el campo *Version* en la estructura MQCD para obtener más información.
3. Si la salida de canal recibe una estructura MQCD con el campo *Version* establecido en un valor mayor que MQCD_VERSION_1, la salida debe utilizar el campo *ConnectionName* en MQCD, en preferencia al campo *ShortConnectionName*.
4. En general, las salidas de canal pueden cambiar la longitud de los datos de mensaje. Esto puede surgir como resultado de que la salida añada datos al mensaje, o elimine datos del mensaje, o comprima o cifre el mensaje. Sin embargo, se aplican restricciones especiales si el mensaje es un segmento que sólo contiene parte de un mensaje lógico. En particular, no debe haber ningún cambio neto en la longitud del mensaje como resultado de las acciones de salidas complementarias de envío y recepción.

Por ejemplo, se permite que una salida de envío acorte el mensaje comprimiéndolo, pero la salida de recepción complementaria debe restaurar la longitud original del mensaje descomprimiéndolo, de modo que no haya ningún cambio neto en la longitud del mensaje.

Esta restricción surge porque cambiar la longitud de un segmento haría que los desplazamientos de segmentos posteriores del mensaje fueran incorrectos, y esto inhibiría la capacidad del gestor de colas de reconocer que los segmentos formaban un mensaje lógico completo.

MQ_CHANNEL_AUTO_DEF_EXIT-Salida de definición automática de canal

La llamada MQ_CHANNEL_AUTO_DEF_EXIT describe los parámetros que se pasan a la salida de definición automática de canal llamada por el agente de canal de mensajes.

El gestor de colas no proporciona ningún punto de entrada denominado MQ_CHANNEL_AUTO_DEF_EXIT; el nombre MQ_CHANNEL_AUTO_DEF_EXIT no tiene ninguna importancia especial porque los nombres de las salidas de definición automática se proporcionan en el gestor de colas.

Sintaxis

MQ_CHANNEL_AUTO_DEF_EXIT (*ChannelExitParms*, *ChannelDefinition*)

Parámetros

La llamada MQ_CHANNEL_AUTO_DEF_EXIT tiene los parámetros siguientes.

ChannelExitParms (MQCXP) - entrada/salida

Bloque de parámetros de salida de canal.

Esta estructura contiene información adicional relacionada con la invocación de la salida. La salida establece información en esta estructura para indicar cómo procede el MCA.

ChannelDefinition (MQCD)-entrada/salida

Definición de canal.

Esta estructura contiene parámetros establecidos por el administrador para controlar el comportamiento de los canales que se crean automáticamente. La salida establece información en esta estructura para modificar el comportamiento predeterminado establecido por el administrador.

La salida no debe modificar los campos MQCD listados:

- *ChannelName*
- *ChannelType*
- *StrucLength*
- *Version*

Si se cambian otros campos, el valor establecido por la salida debe ser válido. Si el valor no es válido, se graba un mensaje de error en el archivo de registro de errores o se visualiza en la consola (según corresponda al entorno).



Atención: Los canales autodefinidos creados mediante una salida de definición automática de canal (CHAD) no pueden establecer la etiqueta de certificado, ya que el reconocimiento TLS se produce en el momento en que se crea el canal. Establecer la etiqueta de certificado en una salida CHAD para los canales de entrada no tiene ningún efecto.

Invocación en C

```
exitname (&ChannelExitParms, &ChannelDefinition);
```

Los parámetros pasados a la salida se declaran de la siguiente manera:

```
MQCXP  ChannelExitParms;  /* Channel exit parameter block */
MQCD   ChannelDefinition; /* Channel definition */
```

Invocación en COBOL

```
CALL 'exitname' USING CHANNELEXITPARMS, CHANNELDEFINITION.
```

Los parámetros pasados a la salida se declaran de la siguiente manera:

```

** Channel exit parameter block
01 CHANNELEXITPARMS.
   COPY CMQXCPV.
** Channel definition
01 CHANNELDEFINITION.
   COPY CMQCDV.

```

Invocación de RPG (ILE)

```

C*.1.....2.....3.....4.....5.....6.....7..
C                                CALLP      exitname(MQCXP : MQCD)

```

La definición de prototipo para la llamada es:

```

D*.1.....2.....3.....4.....5.....6.....7..
Dexitname          PR              EXTPROC('exitname')
D* Channel exit parameter block
D MQCXP              160A
D* Channel definition
D MQCD              1328A

```

Invocación de ensamblador System/390

```
CALL EXITNAME, (CHANNELEXITPARMS, CHANNELDEFINITION)
```

Los parámetros pasados a la salida se declaran de la siguiente manera:

```

CHANNELEXITPARMS  CMQCXPA  , Channel exit parameter block
CHANNELDEFINITION CMQCDA   , Channel definition

```

Notas de uso

1. La función realizada por la salida de canal la define el proveedor de la salida. Sin embargo, la salida debe ajustarse a las reglas definidas aquí y en el bloque de control asociado, MQCXP.
2. El parámetro **ChannelExitParms** pasado a la salida de definición automática de canal es una estructura MQCXP. La versión de MQCXP pasada depende del entorno en el que se ejecuta la salida; consulte la descripción del campo *Version* en [“MQCXP-Parámetro de salida de canal”](#) en la [página 1544](#) para obtener más detalles.
3. El parámetro **ChannelDefinition** pasado a la salida de definición automática de canal es una estructura MQCD. La versión de MQCD pasada depende del entorno en el que se ejecuta la salida; consulte la descripción del campo *Version* en [“MQCD-Definición de canal”](#) en la [página 1502](#) para obtener más detalles.

MQXWAIT-Esperar en salida

La llamada MQXWAIT espera a que se produzca un suceso. Sólo se puede utilizar desde una salida de canal en z/OS.

El uso de MQXWAIT ayuda a evitar problemas de rendimiento que de otro modo podrían producirse si una salida de canal hace algo que provoca una espera. El evento que MQXWAIT está esperando es señalado por un MVS ECB (bloque de control de eventos). El ECB se describe en la descripción del bloque de control MQXWD.

 Para obtener más información sobre el uso de MQXWAIT y la grabación de programas de salida de canal, consulte [Escritura de programas de salida de canal en z/OS](#)

Sintaxis

MQXWAIT (*Hconn*, *WaitDesc*, *CompCode*, *Reason*)

Parámetros

La llamada MQXWAIT tiene los parámetros siguientes.

Hconn (MQHCONN)-entrada

Descriptor de conexión.

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* ha sido devuelto por una llamada MQCONN anterior emitida en la misma invocación o anterior de la salida.

WaitDesc (MQXWD)-entrada/salida

Descriptor de espera.

Este parámetro describe el suceso a esperar. Consulte [“MQXWD-Descriptor de espera de salida” en la página 1559](#) para obtener detalles de los campos de esta estructura.

CompCode (MQLONG)-salida

Código de terminación.

Es uno de los códigos siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón (MQLONG)-salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'89C') El adaptador no está disponible.

MQRC_OPTIONS_ERROR

(2046, X'7FE') Las opciones no son válidas o no son coherentes.

MQRC_XWAIT_CANCELADO

(2107, X'83B') Llamada MQXWAIT cancelada.

MQRC_XWAIT_ERROR

(2108, X'83C') La invocación de la llamada MQXWAIT no es válida.

Invocación en C

```
MQXWAIT (Hconn, &WaitDesc, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQHCONN  Hconn;      /* Connection handle */
MQXWD     WaitDesc;   /* Wait descriptor */
MQLONG    CompCode;   /* Completion code */
MQLONG    Reason;     /* Reason code qualifying CompCode */
```

Invocación de ensamblador System/390

```
CALL MQXWAIT,(HCONN,WAITDESC,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

```
HCONN      DS      F      Connection handle
WAITDESC   CMQXWDA  ,      Wait descriptor
```

COMPCODE	DS	F	Completion code
REASON	DS	F	Reason code qualifying COMPCODE

MQCD-Definición de canal

La estructura MQCD contiene los parámetros que controlan la ejecución de un canal. Se pasa a cada salida de canal que se llama desde un agente de canal de mensajes (MCA).

Si desea más información sobre las salidas de canal, consulte “MQ_CHANNEL_EXIT-Salida de canal” en la [página 1495](#). La descripción de este tema se refiere tanto a los canales de mensajes como a los canales de « MQI ».

Campos de nombre de salida

Cuando se llama a una salida, el campo relevante de *SecurityExit*, *MsgExit*, *SendExit*, *ReceiveExit* y *MsgRetryExit* contiene el nombre de la salida que se está invocando actualmente. El significado del nombre en estos campos depende del entorno en el que se ejecuta el MCA. Excepto donde se indique, el nombre se alinea a la izquierda dentro del campo, sin blancos intercalados; el nombre se rellena con blancos hasta la longitud del campo. En las descripciones siguientes, los corchetes ([]) indican información opcional:

AIX and Linux

El nombre de salida es el nombre de un módulo o biblioteca cargable dinámicamente, con el sufijo del nombre de una función que reside en dicha biblioteca. El nombre de función debe estar entre paréntesis. Opcionalmente, el nombre de biblioteca puede tener como prefijo una vía de acceso de directorio:

```
[ path ] library ( function )
```

El nombre está limitado a un máximo de 128 caracteres.

z/OS

El nombre de salida es el nombre de un módulo de carga que es válido para la especificación en el parámetro EP de la macro LINK o LOAD. El nombre está limitado a un máximo de ocho caracteres.

Windows

El nombre de salida es el nombre de una biblioteca de enlace dinámico, con el sufijo del nombre de una función que reside en dicha biblioteca. El nombre de función debe estar entre paréntesis. Opcionalmente, el nombre de biblioteca puede tener como prefijo una vía de acceso de directorio y una unidad:

```
[d:][ path ] library ( function )
```

El nombre está limitado a un máximo de 128 caracteres.

IBM i

El nombre de salida es un nombre de programa de 10 bytes seguido de un nombre de biblioteca de 10 bytes. Si los nombres tienen menos de 10 bytes de longitud, cada nombre se rellena con espacios en blanco para que tenga 10 bytes. El nombre de biblioteca puede ser *LIBL excepto cuando se llama a una salida de definición automática de canal, en cuyo caso se necesita un nombre completo.

Cambio de campos MQCD en una salida de canal

Una salida de canal pueden cambiar los campos del MQCD. El valor cambiado permanece en el MQCD y se pasa a las salidas restantes de una cadena de salida y a cualquier conversación que comparta la instancia de canal. El MQCD modificado también lo utiliza el MCA para su proceso normal durante el tiempo de vida continuo del canal.

La salida no debe modificar los siguientes campos MQCD:

- ChannelName
- ChannelType

- StrucLength
- Versión

Referencia relacionada

[“Campos” en la página 1503](#)

Este tema lista todos los campos de la estructura MQCD y describe cada campo.

[“Declaración C” en la página 1531](#)

Esta declaración es la declaración C para la estructura MQCD.

[“COBOL declaración” en la página 1533](#)

Esta declaración es la declaración « COBOL » para la estructura MQCD.

[“Declaración RPG \(ILE\)” en la página 1536](#)

Esta declaración es la declaración RPG para la estructura MQCD.

[“System/390 declaración del ensamblador” en la página 1538](#)

Esta declaración es la declaración del ensamblador System/390 para la estructura MQCD.

[“Declaración de Visual Basic” en la página 1540](#)

Esta declaración es la declaración de Visual Basic de la estructura MQCD.

[“Cambio de campos MQCD en una salida de canal” en la página 1541](#)

Una salida de canal pueden cambiar los campos del MQCD. Sin embargo, estos cambios normalmente no se realizan, excepto en las circunstancias listadas.

Campos

Este tema lista todos los campos de la estructura MQCD y describe cada campo.

BatchDataLimit (MQLONG)

Este campo especifica el límite, en kilobytes, de la cantidad de datos que se pueden enviar a través de un canal antes de tomar un punto de sincronización.

Un punto de sincronización se alcanza después de que el mensaje que haya provocado que se llegue al límite, haya fluído a través del canal.

El proceso por lotes finaliza cuando se cumple una de las condiciones siguientes:

- **BatchSize** Se han enviado mensajes.
- **BatchDataLimit** Se han enviado bytes.
- La cola de transmisión está vacía y **BatchInterval** se ha superado.

El valor debe estar comprendido entre 0 y 999999. El valor predeterminado es 5000.

Un valor de cero en este atributo significa que no se aplica ningún límite de datos a los lotes a través de este canal.

Este parámetro solo se aplica a canales con un *ChannelType* de MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSRCVR o MQCHT_CLUSSDR.

Este es un campo de entrada a la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_11.

BatchHeartbeat (MQLONG)

Este campo especifica el intervalo de tiempo que se utiliza para desencadenar una pulsación por lotes para el canal.

La pulsación por lotes permite a los canales emisores determinar si la instancia de canal remoto sigue activa antes de pasar a ser dudosa. Se produce una pulsación por lotes si un canal emisor no se ha comunicado con la instancia de canal remoto dentro del intervalo de tiempo especificado.

El valor está en el rango de 0 a 999 999; las unidades son milisegundos. Un valor de cero indica que la pulsación por lotes no está habilitada.

Este campo sólo es relevante para los canales que tienen un *ChannelType* de MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR o MQCHT_CLUSRCVR.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_7.

BatchInterval (MQLONG)

Este campo especifica el tiempo aproximado en milisegundos que un canal mantiene abierto un lote, si se han transmitido menos de *BatchSize* mensajes en el lote actual.

Si *BatchInterval* es mayor que cero, el lote termina por cualquiera de los sucesos siguientes que se produzcan primero:

- Se han enviado *BatchSize* mensajes, o
- Han transcurrido *BatchInterval* milisegundos desde el inicio del lote.

Si *BatchInterval* es cero, el lote termina por cualquiera de los sucesos siguientes que se produzcan primero:

- Se han enviado *BatchSize* mensajes, o
- la cola de transmisión pasa a estar vacía.

BatchInterval debe estar en el rango de cero a 999 999 999 999.

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR o MQCHT_CLUSRCVR.

Es un campo de entrada para la salida. El campo no está presente cuando *Version* es menor que MQCD_VERSION_4.

BatchSize (MQLONG)

Este campo especifica el número máximo de mensajes que se pueden enviar a través de un canal antes de sincronizar el canal.

Este campo no es relevante para los canales con un *ChannelType* de MQCHT_SVRCONN o MQCHT_CLNTCONN.

CertificateLabel (MQCHAR64)

Este campo proporciona información sobre las etiquetas de los certificados que se están utilizando.

 Se pueden configurar varias etiquetas de certificado, separadas por un  espacio o  una coma. Cuando se especifican varias etiquetas, IBM MQ el sistema probará cada una de ellas para determinar si el certificado seleccionado dará lugar a una selección válida de « CipherSuite » durante el protocolo de enlace « TLS ». Esto está pensado para utilizarse en aquellos casos en los que se pueda emplear TLS, 1.2, RSA o ECC CipherSuites.

 Cuando se indiquen varias etiquetas, utilice **una** coma o un espacio entre ellas. Los delimitadores no pertinentes provocarán errores.

 Por lo general, no se recomienda incluir comas ni espacios en las etiquetas de los certificados. Si la etiqueta de un certificado ya existente contiene comas o espacios, escápalas con una tecla de retroceso. Por ejemplo, si la etiqueta del certificado en el almacén de claves es *val1*, *val2* (coma-espacio), entonces el **CERTLABL** valor sería *val1*, *\ val2*. El carácter de escape indica al gestor de colas que la coma o el espacio forman parte de la etiqueta.  En IBM i, el carácter de espacio no es un separador compatible y no debe escaparse.

IBM MQ Inicializa el valor predeterminado del *CertificateLabel* campo como espacios en blanco.

Esto se interpreta en tiempo de ejecución como el valor predeterminado y es compatible con versiones anteriores.

Por ejemplo, especificar una versión MQCD inferior a 11, o utilizar el valor predeterminado de espacios en blanco para el *CertificateLabel* campo, significa que este campo se ignora.

La longitud de este campo viene dada por MQ_CERT_LABEL_LENGTH.

ChannelMonitoring (MQLONG)

Este campo especifica el nivel actual de recopilación de datos de supervisión para el canal.

Este campo no es relevante para los canales con un ChannelType de MQCHT_CLNTCONN.

Es uno de los valores siguientes:

- MQMON_OFF
- MQMON_LOW
- MQMON_MEDIO
- MQMON_HIGH

Es un campo de entrada para la salida. No está presente si *Version* es menor que MQCD_VERSION_8.

ChannelName (MQCHAR20)

Este campo especifica el nombre de definición de canal.

Debe haber una definición de canal del mismo nombre en la máquina remota para poder comunicarse.

El nombre debe utilizar sólo los caracteres:

- Mayúsculas A-Z
- Minúsculas a-z
- Números 0-9
- Punto (.)
- Barra inclinada (/)
- Subrayado (_)
- Signo de porcentaje (%)

y debe rellenarse a la derecha con espacios en blanco. No se permiten los espacios en blanco iniciales o intercalados.

La longitud de este campo la proporciona MQ_CHANNEL_NAME_LENGTH.

ChannelStatistics (MQLONG)

Este campo especifica el nivel actual de recopilación de datos de estadísticas para el canal.

Este campo no es relevante para los canales con un ChannelType de MQCHT_CLNTCONN o MQCHT_SVRCONN.

Es uno de los valores siguientes:

- MQMON_OFF
- MQMON_LOW
- MQMON_MEDIO
- MQMON_HIGH

Es un campo de entrada para la salida. No está presente si *Version* es menor que MQCD_VERSION_8.

ChannelType (MQLONG)

Este campo especifica el tipo de canal.

Es uno de los valores siguientes:

MQCHT_SENDER

Remitente.

MQCHT_SERVER

Servidor.

MQCHT_RECEIVER

Receptor.

MQCHT_REQUESTER

Solicitante.

MQCHT_CLNTCONN

Conexión de cliente.

MQCHT_SVRCONN

Conexión de servidor (para que lo utilicen los clientes).

MQCHT_CLUSSDR

Remitente de clúster.

MQCHT_CLUSRCVR

Receptor de clúster.

ClientChannelWeight (MQLONG)

Este campo especifica una ponderación para influir en qué definición de canal de conexión de cliente se utiliza.

El atributo ClientChannelWeight se utiliza para que las definiciones de canal cliente puedan seleccionarse aleatoriamente en función de su ponderación cuando se disponga de más de una definición adecuada. Cuando un cliente emite un MQCONN solicitando la conexión a un grupo de gestores de colas, especificando un nombre de gestor de colas que comienza con un asterisco, y hay más de una definición de canal adecuada disponible en la tabla de definición de canal de cliente (CCDT), la definición a utilizar se selecciona aleatoriamente en función de la ponderación, seleccionándose en primer lugar las definiciones aplicables ClientChannelWeight(0) en orden alfabético.

Especifique un valor entre 0 y 99. El valor predeterminado es 0.

El valor 0 indica que no se realiza ningún equilibrio de carga y que las definiciones aplicables se seleccionan en orden alfabético. Para habilitar el equilibrio de carga, elija un valor entre 1 y 99, donde 1 es el peso más bajo y 99 el más alto. La distribución de mensajes entre dos o más canales con ponderaciones distintas de cero es proporcional a la proporción de esas ponderaciones. Por ejemplo, tres canales con valores ClientChannelWeight de 2, 4 y 14 se seleccionan aproximadamente el 10%, el 20% y el 70% de las veces. Esta distribución no está garantizada.

Este atributo sólo es válido para el tipo de canal de conexión de cliente.

Es un campo de entrada para la salida. El campo no está presente si *Versión* es menor que MQCD_VERSION_9.

ClusterPtr (MQPTR)

Este campo especifica la dirección de una lista de nombres de clúster.

Si *ClustersDefined* es mayor que cero, esta dirección es la dirección de una lista de nombres de clúster. El canal pertenece a cada clúster listado.

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_CLUSSDR o MQCHT_CLUSRCVR.

Es un campo de entrada para la salida. El campo no está presente si *Versión* es menor que MQCD_VERSION_5.

ClustersDefined (MQLONG)

Este campo especifica el número de clústeres a los que pertenece el canal.

Este campo es el número de nombres de clúster a los que apunta *ClusterPtr*. Es cero o mayor.

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_CLUSSDR o MQCHT_CLUSRCVR.

Es un campo de entrada para la salida. El campo no está presente si *Versión* es menor que MQCD_VERSION_5.

CLWLChannelPriority (MQLONG)

Este campo especifica la prioridad del canal de carga de trabajo del clúster.

El algoritmo de selección del gestor de carga de trabajo selecciona un destino con la prioridad más alta del conjunto de destinos seleccionados en función del rango. Si hay dos gestores de colas de destino posibles, este atributo se puede utilizar para realizar una migración tras error de un gestor de colas en el otro gestor de colas. Todos los mensajes van al gestor de colas con la prioridad más alta hasta que finaliza, a continuación, los mensajes van al gestor de colas con la siguiente prioridad más alta.

El valor está en el rango de 0 a 9. El valor predeterminado es 0.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_8.

Para obtener más información, consulte [Configuración de un clúster de gestores de colas](#).

CLWLChannelRank (MQLONG)

Este campo especifica el rango de canal de carga de trabajo de clúster.

El algoritmo de selección del gestor de carga de trabajo selecciona un destino con el rango más alto. Cuando el destino final es un gestor de colas en un clúster diferente, puede establecer el rango de gestores de colas de pasarela intermedios (en la intersección de clústeres vecinos) para que el algoritmo de selección elija correctamente un gestor de colas de destino más próximo al destino final.

El valor está en el rango de 0 a 9. El valor predeterminado es 0.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_8.

Para obtener más información, consulte [Configuración de un clúster de gestores de colas](#).

CLWLChannelWeight (MQLONG)

Este campo especifica el peso del canal de carga de trabajo del clúster.

Peso de canal de carga de trabajo de clúster.

El algoritmo de selección del gestor de carga de trabajo utiliza el atributo "weight" del canal al desvío de la opción de destino para que se puedan enviar más mensajes a una máquina determinada. Por ejemplo, puede dar a un canal en un servidor UNIX grande un "peso" mayor que otro canal en un PC de escritorio pequeño, y el algoritmo de selección elige el servidor UNIX con más frecuencia que el PC.

El valor está en el rango de 1 a 99. El valor predeterminado es 50.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_8.

Para obtener más información, consulte [Configuración de un clúster de gestores de colas](#).

ConnectionAffinity (MQLONG)

Este campo especifica si las aplicaciones cliente que se conectan varias veces utilizando el mismo nombre de gestor de colas, utilizan el mismo canal de cliente.

Utilice este atributo cuando hay disponibles varias definiciones de canal aplicables.

El valor puede ser uno de los siguientes:

MQCAFTY_PREFERIDO

La primera conexión en un proceso que lee una tabla de definiciones de canal de cliente (CCDT) crea una lista de definiciones aplicables basadas en la ponderación con cualquier definición de CLNTWGHT (0) aplicable primero y en orden alfabético. Cada conexión del proceso intenta conectar utilizando la primera definición de la lista. Si una conexión no es satisfactoria, se utiliza la siguiente definición. Las definiciones no satisfactorias con valores CLNTWGHT distintos de 0 se mueven al final de la lista. Las definiciones CLNTWGHT(0) permanecen en el principio de la lista y se seleccionan en primer lugar para cada conexión.

Cada proceso de cliente con el mismo nombre de host siempre crea la misma lista.

Para las aplicaciones cliente escritas en C, C++ o la infraestructura de programación de .NET (incluida la .NET totalmente gestionada), la lista se actualiza si la CCDT se ha modificado desde que se creó la lista.

Este es el valor predeterminado.

MQCAFTY_NONE

La primera conexión de un proceso que lee una CCDT crea una lista de definiciones aplicables.

Todas las conexiones en un proceso seleccionan una definición aplicable según el peso con cualquier definición CLNTWGHT(0) aplicable seleccionada primero en orden alfabético.

Para las aplicaciones cliente escritas en C, C++ o la infraestructura de programación de .NET (incluida la .NET totalmente gestionada), la lista se actualiza si la CCDT se ha modificado desde que se creó la lista.

Este atributo sólo es válido para el tipo de canal de conexión de cliente.

Es un campo de entrada para la salida. El campo no está presente si *Versión* es menor que MQCD_VERSION_9.

ConnectionName (MQCHAR264)

Este campo especifica el nombre de conexión para el canal.

Para los canales de clúster receptor (cuando se especifica), CONNAME está relacionado con el gestor de colas local y para otros canales está relacionado con el gestor de colas de destino. El valor que especifique depende del protocolo de transmisión (*TransportType*) que se va a utilizar:

- Para MQXPT_LU62, es el nombre completo de la unidad lógica asociada.
- Para MQXPT_NETBIOS, es el nombre NetBIOS definido en la máquina remota.
- Para MQXPT_TCP, es el nombre de host, la dirección de red de la máquina remota especificada en formato IPv4 decimal con puntos o IPv6 hexadecimal, o la máquina local para canales de clúster receptor.
- Para MQXPT_SPX, es una dirección de estilo SPX que consta de una dirección de red de 4 bytes, una dirección de nodo de 6 bytes y un número de socket de 2 bytes.

Al definir un canal, este campo no es relevante para los canales con un *ChannelType* de MQCHT_SVRCONN o MQCHT_RECEIVER. Sin embargo, cuando la definición de canal se pasa a una salida, este campo contiene la dirección del socio, sea cual sea el tipo de canal.

La longitud de este campo la proporciona MQ_CONN_NAME_LENGTH. Este campo no está presente si *Versión* es menor que MQCD_VERSION_2.

DataConversion (MQLONG)

Este campo especifica si el agente de canal de mensajes emisor intenta la conversión de los datos de mensaje de aplicación si el agente de canal de mensajes receptor no puede realizar esta conversión.

Este campo sólo se aplica a los mensajes que no son segmentos de mensajes lógicos; el MCA nunca intenta convertir mensajes que son segmentos.

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR o MQCHT_CLUSRCVR. Es uno de los siguientes:

MQCDC_SENDER_CONVERSION

Conversión por remitente.

MQCDC_NO_CONVERSIÓN_DEL_REMITENTE

Sin conversión por remitente.

DefReconnect (MQLONG)

El DefReconnect atributo channel establece el valor predeterminado del atributo de reconexión para un canal de conexión de cliente.

La opción de reconexión de cliente automática predeterminada. Puede configurar un IBM MQ MQI client para que reconecte automáticamente una aplicación cliente. El IBM MQ MQI client intenta reconectarse a

un gestor de colas después de la anomalía de una anomalía de la conexión. Intenta reconectarse sin que el cliente de la aplicación emita una llamada MQCONN o MQCONNX MQI.

La reconexión es una MQCONNX opción. Mediante el uso del DefReconnect atributo channel, puede añadir un comportamiento de reconexión a las aplicaciones existentes que utilizan MQCONN. También puede cambiar el comportamiento de reconexión de las aplicaciones que utilizan MQCONNX.

También puede establecer el DefRecon valor desde el mqclient.ini archivo para configurar o modificar el comportamiento de reconexión. El DefRecon valor del archivo mqclient.ini tiene prioridad sobre el atributo DefReconnect del canal.

Syntax

DefReconnect (MQRCN_NO (default) |MQRCN_YES|MQRCN_Q_MGR|MQRCN_DISABLED)

Parámetros

MQRCN_NO

MQRCN_NO es el valor predeterminado.

A menos que se anule mediante **MQCONNX**, el cliente no se vuelve a conectar automáticamente.

MQRCN_YES

MQCONNXA menos que se especifique lo contrario, el cliente se vuelve a conectar automáticamente.

MQRCN_Q_MGR

A menos que se anule mediante **MQCONNX**, el cliente se vuelve a conectar automáticamente, pero solo al mismo gestor de colas. La opción QMGR tiene el mismo efecto que MQCNO_RECONNECT_Q_MGR.

MQRCN_DISABLED

La reconexión está desactivada, incluso si el programa cliente la solicita mediante la **MQCONNX** llamada ` MQI `.

IBM MQ classes for Java no da soporte a la reconexión automática del cliente.

Tabla 858. La reconexión automática depende de los valores establecidos en la aplicación y en la definición de canal				
DefReconnect	Opciones de reconexión establecidas en la aplicación			
	MQCNO_RECONNE CT	MQCNO_RECONNE CT_Q_MGR	MQCNO_RECONNE CT_AS_DEF	MQCNO_RECONNE CT_DISABLED
MQRCN_NO	SÍ	QMGR	NO	NO
MQRCN_YES	SÍ	QMGR	SÍ	NO
MQRCN_Q_MGR	SÍ	QMGR	QMGR	NO
MQRCN_DISABLED	NO	NO	NO	NO

Conceptos relacionados

[Reconexión de cliente automática](#)

[Reconexión de canal y cliente](#)

[Stanza CHANNELS del archivo de configuración de cliente](#)

Información relacionada

[“Opciones \(MQLONG\)” en la página 277](#)

Descripción (MQCHAR64)

Este campo se puede utilizar para comentarios descriptivos.

El contenido del campo no es significativo para los agentes de canal de mensajes. Sin embargo, sólo debe contener caracteres que se puedan visualizar. No puede contener ningún carácter nulo; si es necesario,

se rellena a la derecha con espacios en blanco. En una instalación DBCS, el campo puede contener caracteres DBCS (sujetos a una longitud máxima de campo de 64 bytes).

Nota: Si este campo contiene caracteres que no están en el juego de caracteres del gestor de colas (tal como se define en el atributo de gestor de colas **CodedCharSetId**), es posible que estos caracteres se conviertan incorrectamente si este campo se envía a otro gestor de colas.

La longitud de este campo la proporciona MQ_CHANNEL_DESC_LENGTH.

DiscInterval (MQLONG)

Este campo especifica el tiempo máximo en segundos durante el cual el canal espera a que llegue un mensaje a la cola de transmisión, antes de terminar el canal.

En otras palabras, especifica el intervalo de desconexión.

El valor A de cero hace que el MCA espere indefinidamente.

En el caso de los canales de conexión al servidor que utilizan el protocolo TCP, el intervalo representa el tiempo de inactividad del cliente tras el cual se produce la desconexión, expresado en segundos. Si una conexión de servidor no ha recibido ninguna comunicación de su cliente asociado durante este tiempo, termina la conexión. El intervalo de inactividad de conexión de servidor sólo se aplica entre llamadas de API de IBM MQ desde un cliente, por lo que ningún cliente se desconecta durante una MQGET de larga ejecución con llamada de espera.

Este atributo no es aplicable a los canales de conexión al servidor que utilicen protocolos distintos de TCP.

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR, MQCHT_CLUSRCVR o MQCHT_SVRCONN.

ExitDataLength (MQLONG)

Este campo especifica la longitud en bytes de cada uno de los elementos de datos de usuario en las listas de elementos de datos de usuario de salida a los que se dirigen los campos *MsgUserDataPtr*, *SendUserDataPtr* y *ReceiveUserDataPtr*.

Esta longitud no es necesariamente la misma que MQ_EXIT_DATA_LENGTH.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_4.

ExitNameLength (MQLONG)

Este campo especifica la longitud en bytes de cada uno de los nombres de las listas de nombres de salida a los que se dirigen los campos *MsgExitPtr*, *SendExitPtr* y *ReceiveExitPtr*.

Esta longitud no es necesariamente la misma que MQ_EXIT_NAME_LENGTH.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_4.

HdrCompList [2] (MQLONG)

Este campo especifica la lista de técnicas de compresión de datos de cabecera soportadas por el canal.

La lista contiene uno o varios de los valores siguientes:

MQCOMPRESS_NONE

No se lleva a cabo ninguna compresión de datos de cabecera.

MQCOMPRESS_SISTEMA

Se lleva a cabo la compresión de datos de cabecera.

MQCOMPRESS_NO_DISPONIBLE

Los valores no utilizados de la lista se establecen en este valor.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_8.

HeartbeatInterval (MQLONG)

Este campo especifica el tiempo en segundos entre los flujos de pulsaciones.

La interpretación de este campo depende del tipo de canal, tal como se indica a continuación:

- Para un tipo de canal de MQCHT_SENDER, MQCHT_SERVER, MQCHT_RECEIVER MQCHT_REQUESTER, MQCHT_CLUSSDR o MQCHT_CLUSRCVR, este campo es el tiempo en segundos entre los flujos de pulsaciones pasados desde el MCA emisor cuando no hay mensajes en la cola de transmisión. Esto da al MCA receptor la oportunidad de desactivar temporalmente el canal. Para ser útil, *HeartbeatInterval* debe ser menor que *DiscInterval*.
- Para un tipo de canal de MQCHT_CLNTCONN o MQCHT_SVRCONN con el campo Conversaciones de compartición de MQCD establecido en cero, este campo es el tiempo en segundos entre los flujos de latido pasados desde el MCA del servidor cuando dicho MCA ha emitido una llamada MQGET con la opción MQGMO_WAIT en nombre de una aplicación cliente. Esto permite al MCA del servidor manejar situaciones en las que la conexión de cliente falla durante un MQGET con MQGMO_WAIT.
- Para un tipo de canal de MQCHT_CLNTCONN o MQCHT_SVRCONN con el campo Conversaciones de compartición de MQCD establecido en un valor distinto de cero, este campo es el tiempo en segundos entre el flujo de pulsaciones cuando no hay flujos de datos enviados o recibidos. Esto permite que el canal se desactive temporalmente de forma eficiente.

El valor está en el rango de 0 a 999 999. El valor que se utiliza es el mayor de los valores especificados en el lado de envío y el lado de recepción a menos que se especifique un valor de 0 en cada lado, en cuyo caso no se produce ningún intercambio de pulsaciones.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_4.

KeepAliveInterval (MQLONG)

Este campo especifica el valor pasado a la pila de comunicaciones para la temporización de estado activo para el canal.

Este valor es aplicable a los protocolos de comunicación « TCP/IP » y «SPX», aunque no todas las implementaciones admiten este parámetro.

El valor está en el rango de 0 a 99 999; las unidades son segundos. Un valor de cero indica que la función «keepalive» del canal no está habilitada, aunque es posible que se produzca un «keepalive» si está habilitada la función «keepalive» de « TCP/IP » (en lugar de la del canal). También es válido el siguiente valor especial:

MQKAI_AUTO

Automático.

Este valor indica que el intervalo de estado activo se calcula a partir del intervalo de latido negociado, como se indica a continuación:

- Si el intervalo de pulsaciones negociado es mayor que cero, el intervalo de estado activo que se utiliza es el intervalo de pulsaciones más 60 segundos.
- Si el intervalo de pulsaciones negociado es cero, el intervalo de estado activo que se utiliza es cero.
- En TCP/IPz/OS, se produce un keepalive cuando se especifica TCPKEEP(YES) en el objeto del gestor de colas.
- En otros entornos, el mecanismo de mantenimiento de conexión de « TCP/IP » se activa cuando se especifica el **KEEPALIVE=YES** parámetro en la sección « TCP » del archivo de configuración de colas distribuidas.

Este campo sólo es relevante para los canales que tienen un *TransportType* de MQXPT_TCP o MQXPT_SPX.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_7.

LocalAddress (MQCHAR48)

Este campo especifica la dirección local TCP/IP definida para el canal en lo que respecta a las comunicaciones salientes.

Este campo queda en blanco si no se define ninguna dirección específica para las comunicaciones salientes. La dirección puede incluir opcionalmente un número de puerto o un rango de números de puerto. El formato de esta dirección es:

```
[ip-addr] [(low-port[,high-port])]
```

donde los corchetes ([]) indican información opcional, *ip-addr* se especifica en formato IPv4 decimal punteado, IPv6 hexadecimal o alfanumérico, y *low-port* y *high-port* son números de puerto entre paréntesis. Todos son opcionales.

Especificar una dirección IP, un puerto o un rango de puertos para las comunicaciones salientes resulta útil en situaciones de recuperación en las que se reinicia un canal en una pila de TCP/IP diferente.

LocalAddress es similar en forma a *ConnectionName*, pero no debe confundirse con él. *LocalAddress* especifica las características de las comunicaciones locales, mientras que *ConnectionName* especifica cómo llegar a un gestor de colas remoto.

Desde IBM MQ 9.3.0, la Interfaz Java de cola de mensajes (JMQUI) se ha actualizado para garantizar que el campo de dirección local se establezca en un objeto MQCD después de que se haya creado una instancia de canal y se haya conectado a un gestor de colas. Esto significa que cuando una salida de canal escrita en Java llama al método MQCD.*getLocalAddress()*, el método devuelve la dirección local que la instancia de canal está utilizando. Antes IBM MQ 9.3.0, la salida de seguridad del canal no podía acceder a la dirección local utilizada por la instancia del canal y el método MQCD.*getLocalAddress()* devolvía un valor nulo.

Este campo solo es relevante para canales con un *TransportType* de MQXPT_TCP y un *ChannelType* de MQCHT_SENDER, MQCHT_SERVER, MQCHT_REQUESTER, MQCHT_CLNTCONN, MQCHT_CLUSSDR o MQCHT_CLUSRCVR.

La longitud de este campo viene dada por MQ_LOCAL_ADDRESS_LENGTH. Este campo no aparece si *Version* es menor que MQCD_VERSION_7.

LongMCAUserIdLength (MQLONG)

Este campo especifica la longitud en bytes del identificador de usuario MCA completo al que apunta *LongMCAUserIdPtr*.

Este campo no es relevante para los canales con un *ChannelType* de MQCHT_CLNTCONN.

Es un campo de entrada/salida para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_6.

LongMCAUserIdPtr (MQPTR)

Este campo especifica la dirección del identificador de usuario de MCA largo.

Si *LongMCAUserIdLength* es mayor que cero, este campo es la dirección del identificador de usuario MCA completo. La longitud del identificador completo la proporciona *LongMCAUserIdLength*. Los primeros 12 bytes del identificador de usuario de MCA también están contenidos en el campo *MCAUserIdentifier*.

Consulte la descripción del campo *MCAUserIdentifier* para obtener detalles del identificador de usuario de MCA.

Este campo no es relevante para los canales con un *ChannelType* de MQCHT_SDR, MQCHT_SVR, MQCHT_CLNTCONN o MQCHT_CLUSSDR.

Es un campo de entrada/salida para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_6.

LongRemoteUserIdLength (MQLONG)

Este campo especifica la longitud en bytes del identificador de usuario remoto completo al que apunta *LongRemoteUserIdPtr*.

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_CLNTCONN o MQCHT_SVRCONN.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_6.

LongRemoteUserIdPtr (MQPTR)

Este campo especifica la dirección del identificador de usuario remoto largo.

Si *LongRemoteUserIdLength* es mayor que cero, este distintivo es la dirección del identificador de usuario remoto completo. La longitud del identificador completo la proporciona *LongRemoteUserIdLength*. Los primeros 12 bytes del identificador de usuario remoto también están contenidos en el campo *RemoteUserIdIdentifier*.

Consulte la descripción del campo *RemoteUserIdIdentifier* para obtener detalles del identificador de usuario remoto.

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_CLNTCONN o MQCHT_SVRCONN.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_6.

LongRetryCount (MQLONG)

Este campo especifica el recuento utilizado después de que se haya agotado el recuento especificado por *ShortRetryCount*.

Especifica el número máximo de intentos adicionales que se realizan para conectarse a la máquina remota, a intervalos especificados por *LongRetryInterval*, antes de registrar un error en el operador.

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR o MQCHT_CLUSRCVR.

LongRetryInterval (MQLONG)

Este campo especifica el número máximo de segundos que se debe esperar antes de volver a intentar la conexión con la máquina remota.

El intervalo entre reintentos se puede ampliar si el canal tiene que esperar a estar activo.

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR o MQCHT_CLUSRCVR.

MaxInstances (MQLONG)

Este campo especifica el número máximo de instancias simultáneas de un canal de conexión de servidor individual que se pueden iniciar.

Este campo sólo se utiliza en canales de conexión de servidor.

El campo puede tener un valor en el rango de 0 a 999.999.999. El valor cero impide el acceso de los clientes.

El valor por omisión de este campo es 999 999 999.

Si el valor de este campo se reduce a un número menor que el número de instancias del canal de conexión con el servidor que se están ejecutando actualmente, estas instancias en ejecución no se verán afectadas. Sin embargo, las nuevas instancias no se pueden iniciar hasta que hayan dejado de ejecutarse suficientes instancias existentes, de modo que el número de instancias actualmente en ejecución sea menor que el valor del campo.

MaxInstancesPerClient (MQLONG)

Este campo especifica el número máximo de instancias simultáneas de un canal de conexión de servidor individual que se pueden iniciar desde un único cliente.

En este contexto, las conexiones que se originan desde la misma dirección de red remota se consideran procedentes del mismo cliente.

Este campo sólo se utiliza en canales de conexión de servidor.

El campo puede tener un valor en el rango de 0 a 999.999.999. El valor cero impide el acceso de los clientes.

El valor por omisión de este campo es 999 999 999.

Si el valor de este campo se reduce a un número menor que el número de instancias del canal de conexión con el servidor que se están ejecutando actualmente desde clientes individuales, las instancias en ejecución no se verán afectadas. Sin embargo, las nuevas instancias de cualquiera de estos clientes no se pueden iniciar hasta que hayan dejado de ejecutarse suficientes instancias existentes de modo que el número de instancias actualmente en ejecución, que se originan en el cliente que intenta iniciar una nueva, sea menor que el valor del campo.

MaxMsgLength (MQLONG)

Este campo especifica la longitud máxima de mensaje que puede transmitirse en el canal.

Este valor se compara con el del canal remoto y el máximo real es el menor de los dos valores.

MCANombre (MQCHAR20)

Este campo es un campo reservado.

El valor de este campo está en blanco.

La longitud de este campo la proporciona MQ_MCA_NAME_LENGTH.

MCASecurityId (MQBYTE40)

Este campo especifica el identificador de seguridad para el MCA.

Este campo no es relevante para los canales con un *ChannelType* de MQCHT_CLNTCONN.

El siguiente valor especial indica que no hay ningún identificador de seguridad:

MQSID_NONE

No se ha especificado ningún identificador de seguridad.

El valor es cero binario para la longitud del campo.

Para el lenguaje de programación C, también se define la constante MQSID_NONE_ARRAY; esta constante tiene el mismo valor que MQSID_NONE, pero es una matriz de caracteres en lugar de una serie.

Es un campo de entrada/salida para la salida. La longitud de este campo la proporciona MQ_SECURITY_ID_LENGTH. Este campo no está presente si *Version* es menor que MQCD_VERSION_6.

MCAType (MQLONG)

Este campo especifica el tipo de programa de agente de canal de mensajes.

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_SENDER, MQCHT_SERVER, MQCHT_REQUESTER, MQCHT_CLUSSDR o MQCHT_CLUSRCVR.

El valor puede ser uno de los siguientes:

MQMCAT_PROCESO

proceso.

El agente de canal de mensajes se ejecuta como un proceso independiente.

HEBRA MQMCAT_THREAD

Hebra (Multiplataformas).

El agente de canal de mensajes se ejecuta como una hebra separada.

Este campo no está presente cuando *Versión* es menor que MQCD_VERSION_2.

MCAUserIdentifier (MQCHAR12)

Este campo especifica el identificador de usuario para el agente de canal de mensajes (MCA).

Este campo utiliza los primeros 12 bytes del identificador de usuario de MCA y puede ser establecido por un agente de seguridad.

Hay dos campos que contienen el identificador de usuario de MCA:

- *MCAUserIdentifier* contiene los primeros 12 bytes del identificador de usuario de MCA y se rellena con espacios en blanco si el identificador es inferior a 12 bytes. *MCAUserIdentifier* puede estar en blanco.
- *LongMCAUserIdPtr* apunta al identificador de usuario de MCA completo, que puede tener más de 12 bytes. Su longitud la proporciona *LongMCAUserIdLength*. El identificador completo no contiene blancos de cola y no termina en nulo. Si el identificador está en blanco, *LongMCAUserIdLength* es cero y el valor de *LongMCAUserIdPtr* no está definido.

Nota: *LongMCAUserIdPtr* no está presente si *Versión* es menor que MQCD_VERSION_6.

Si el identificador de usuario de MCA no está en blanco, especifica el identificador de usuario que utilizará el agente de canal de mensajes para la autorización para acceder a los recursos de IBM MQ. Para los tipos de canal MQCHT_REQUESTER, MQCHT_RECEIVER y MQCHT_CLUSRCVR, si PutAuthority es MQPA_DEFAULT, este es el identificador de usuario utilizado para las comprobaciones de autorización para la operación de colocación en colas de destino.

Si el identificador de usuario de MCA está en blanco, el agente de canal de mensajes utiliza su identificador de usuario predeterminado.

El identificador de usuario de MCA se puede establecer mediante una salida de seguridad para indicar el identificador de usuario que debe utilizar el agente de canal de mensajes. La salida puede cambiar *MCAUserIdentifier* la serie a la que apunta *LongMCAUserIdPtr*. Si ambos se cambian pero difieren entre sí, el MCA utiliza *LongMCAUserIdPtr* en preferencia a *MCAUserIdentifier*. Si la salida cambia la longitud de la serie a la que se refiere *LongMCAUserIdPtr*, *LongMCAUserIdLength* debe establecerse de forma correspondiente. Si la salida aumenta la longitud del identificador, la salida debe asignar el almacenamiento de la longitud necesaria, establecer dicho almacenamiento en el identificador necesario y colocar la dirección de dicho almacenamiento en *LongMCAUserIdPtr*. La salida es responsable de liberar ese almacenamiento cuando la salida se invoca más tarde con la razón MQXR_TERM.

Para los canales con un *ChannelType* de MQCHT_SVRCONN, si *MCAUserIdentifier* en la definición de canal está en blanco, cualquier identificador de usuario transferido desde el cliente se copia en él. Este identificador de usuario (después de cualquier modificación por parte de la salida de seguridad en el servidor) es el que se supone que ejecuta la aplicación cliente.

El identificador de usuario de MCA no es relevante para los canales con un *ChannelType* de MQCHT_SDR, MQCHT_SVR, MQCHT_CLNTCONN, MQCHT_CLUSSDR.

Es un campo de entrada/salida para la salida. La longitud de este campo la proporciona MQ_USER_ID_LENGTH. Este campo no está presente cuando *Versión* es menor que MQCD_VERSION_2.

ModeName (MQCHAR8)

Este campo especifica el nombre de modalidad de LU 6.2.

Este campo sólo es relevante si el protocolo de transmisión (*TransportType*) es MQXPT_LU62y *ChannelType* no es MQCHT_SVRCONN ni MQCHT_RECEIVER.

Este campo siempre está en blanco. En su lugar, la información está contenida en el objeto lateral de comunicaciones.

La longitud de este campo la proporciona MQ_MODE_NAME_LENGTH.

MsgCompList [16] (MQLONG)

Este campo especifica la lista de técnicas de compresión de datos de mensajes que admite el canal.

La lista contiene uno o más de los siguientes valores:

MQCOMPRESS_NONE

No se lleva a cabo ninguna compresión de datos de mensaje.

MQCOMPRESS_RLE

Se lleva a cabo la compresión de datos de mensaje utilizando la codificación de longitud de ejecución.

MQCOMPRESS_ZLIBFAST

Se lleva a cabo la compresión de datos de mensaje utilizando el método de compresión zlib. Se prefiere un tiempo de compresión rápido.

MQCOMPRESS_ZLIBHIGH

Se lleva a cabo la compresión de datos de mensaje utilizando el método de compresión zlib. Se prefiere un nivel elevado de compresión.

 LZ4FAST

La compresión de los datos del mensaje se realiza mediante codificación LZ4, dando prioridad a la velocidad.

 LZ4HIGH

La compresión de los datos del mensaje se realiza mediante la codificación LZ4, dando prioridad a la compresión.

MQCOMPRESS_ANY

Cualquier técnica de compresión compatible con el gestor de colas puede utilizarse para la compresión de mensajes. MQCOMPRESS_ANY solo es válido para canales de conexión de receptor, solicitante y servidor.

MQCOMPRESS_NO_DISPONIBLE

Los valores no utilizados en la lista se establecen en este valor.

Este es un campo de entrada a la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_8.

MsgExit (MQCHARn)

Este campo especifica el nombre de salida de mensaje de canal.

Si este nombre no está en blanco, se llama a la salida en las horas siguientes:

- Inmediatamente después de que se haya recuperado un mensaje de la cola de transmisión (emisor o servidor), o inmediatamente antes de que se coloque un mensaje en una cola de destino (receptor o solicitante).

A la salida se le proporciona toda la cabecera de cola de mensajes y transmisión de la aplicación para su modificación.

- Durante la inicialización y terminación del canal.

Este campo no es relevante para los canales con un *ChannelType* de MQCHT_SVRCONN o MQCHT_CLNTCONN; nunca se invoca una salida de mensaje para dichos canales.

Consulte “MQCD-Definición de canal” en la [página 1502](#) para obtener una descripción del contenido de este campo en varios entornos.

La longitud de este campo la proporciona MQ_EXIT_NAME_LENGTH.

Nota: El valor de esta constante es específico del entorno.

MsgExitPtr (MQPTR)

Este campo especifica la dirección del primer campo *MsgExit*.

Si *MsgExitsDefined* es mayor que cero, esta dirección es la dirección de la lista de nombres de cada salida de mensaje de canal de la cadena.

Cada nombre está en un campo de longitud *ExitNameLength*, rellenado a la derecha con espacios en blanco. Hay *MsgExitsDefined* campos que se unen entre sí-uno para cada salida.

Los cambios realizados en estos nombres por una salida se conservan, aunque la salida de canal de mensajes no realiza ninguna acción explícita-no cambia qué salidas se invocan.

Si *MsgExitsDefined* es cero, este campo es el puntero nulo.

En plataformas en las que el lenguaje de programación no soporta el tipo de datos de puntero, este campo se declara como una serie de bytes de la longitud adecuada.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_4.

MsgExitsDefined (MQLONG)

Este campo especifica el número de salidas de mensajes de canal definidas en la cadena.

Es mayor o igual que cero.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_4.

MsgRetryCount (MQLONG)

Este campo especifica el número de veces que el MCA intenta colocar el mensaje, después de que el primer intento haya fallado.

Este campo indica el número de veces que el MCA intenta la operación de apertura o colocación, si el primer MQOPEN o MQPUT falla con el código de terminación MQCC_FAILED. El efecto de este atributo depende de si *MsgRetryExit* está en blanco o no:

- Si *MsgRetryExit* está en blanco, el atributo **MsgRetryCount** controla si el MCA intenta reintentos. Si el valor del atributo es cero, no se intenta ningún reintento. Si el valor del atributo es mayor que cero, los reintentos se intentan a intervalos proporcionados por el atributo **MsgRetryInterval**.

Los reintentos sólo se intentan para los siguientes códigos de razón:

- MQRC_PAGESET_FULL
- MQRC_PUT_XX_ENCODE_CASE_CAPS_LOCK_ON inhibida
- MQRC_Q_FULL

Para otros códigos de razón, el MCA continúa inmediatamente con su proceso de anomalía normal, sin reintentar el mensaje anómalo.

- Si *MsgRetryExit* no está en blanco, el atributo **MsgRetryCount** no afecta al MCA; en su lugar, es la salida de reintento de mensaje la que determina cuántas veces se intenta el reintento y a qué intervalos; la salida se invoca aunque el atributo **MsgRetryCount** sea cero.

El atributo **MsgRetryCount** está disponible para la salida en la estructura MQCD, pero la salida que no es necesaria para respetarlo-los reintentos continúan indefinidamente hasta que la salida devuelve MQXCC_SUPPRESS_FUNCTION en el campo *ExitResponse* de MQCXP.

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_REQUESTER, MQCHT_RECEIVER o MQCHT_CLUSRCVR.

Este campo no está presente cuando *Version* es menor que MQCD_VERSION_3.

MsgRetryExit (MQCHARn)

Este campo especifica el nombre de salida de reintento de mensaje de canal.

La salida de reintento de mensaje es una salida que invoca el MCA cuando el MCA recibe un código de terminación de MQCC_FAILED de una llamada MQOPEN o MQPUT. La finalidad de la salida es especificar un intervalo de tiempo durante el cual el MCA espera antes de volver a intentar la operación MQOPEN o MQPUT. De forma alternativa, la salida se puede establecer para no volver a intentar la operación.

La salida se invoca para todos los códigos de razón que tienen un código de terminación de MQCC_FAILED-los valores de la salida determinan qué códigos de razón desea que el MCA vuelva a intentarlo, para cuántos intentos y en qué intervalos de tiempo.

Cuando la operación ya no se intenta, el MCA realiza su proceso de anomalía normal; este proceso incluye la generación de un mensaje de informe de excepción (si lo especifica el remitente) y la colocación del mensaje original en la cola de mensajes no entregados o el descarte del mensaje (según si el remitente ha especificado MQRO_DEAD_LETTER_Q o MQRO_DISCARD_MSG). Las anomalías relacionadas con la cola de mensajes no entregados (por ejemplo, la cola de mensajes no entregados llena) no hacen que se invoque la salida de reintento de mensaje.

Si el nombre de salida no está en blanco, se llama a la salida en las horas siguientes:

- Inmediatamente antes de realizar la espera antes de volver a intentar entregar un mensaje
- Durante la inicialización y terminación del canal

Consulte “MQCD-Definición de canal” en la [página 1502](#) para obtener una descripción del contenido de este campo en varios entornos.

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_REQUESTER, MQCHT_RECEIVER o MQCHT_CLUSRCVR.

La longitud de este campo la proporciona MQ_EXIT_NAME_LENGTH.

Nota: El valor de esta constante es específico del entorno.

Este campo no está presente cuando *Version* es menor que MQCD_VERSION_3.

MsgRetryInterval (MQLONG)

Este campo especifica el intervalo mínimo en milisegundos después del cual se reintenta la operación de apertura o colocación.

El efecto de este atributo depende de si *MsgRetryExit* está en blanco o no:

- Si *MsgRetryExit* está en blanco, el atributo **MsgRetryInterval** especifica el periodo mínimo que el MCA espera antes de reintentar un mensaje, si el primer MQOPEN o MQPUT falla con el código de terminación MQCC_FAILED. Un valor de cero significa que el reintento se realizará lo antes posible después del intento anterior. Los reintentos sólo se realizan si *MsgRetryCount* es mayor que cero.

Este atributo también se utiliza como tiempo de espera si la salida de reintento de mensaje devuelve un valor no válido en el campo *MsgRetryInterval* de MQCXP.

- Si *MsgRetryExit* no está en blanco, el atributo **MsgRetryInterval** no afecta al MCA; en su lugar, es la salida de reintento de mensaje la que determina cuánto tiempo espera el MCA. El atributo **MsgRetryInterval** se pone a disposición de la salida en la estructura MQCD, pero la salida no es necesaria para respetarlo.

El valor está en el rango de 0 a 999.999.999.

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_REQUESTER, MQCHT_RECEIVER o MQCHT_CLUSRCVR.

Este campo no está presente cuando *Version* es menor que MQCD_VERSION_3.

Los campos siguientes de esta estructura no están presentes si *Version* es menor que MQCD_VERSION_4.

MsgRetryUserData (MQCHAR32)

Este campo especifica los datos de usuario de salida de reintento de mensaje de canal.

Estos datos se pasan a la salida de reintento de mensaje de canal en el campo *ExitData* del parámetro **ChannelExitParms** (consulte MQ_CHANNEL_EXIT).

Este campo contiene inicialmente los datos que se han establecido en la definición de canal. Sin embargo, durante el tiempo de vida de esta instancia de MCA, el MCA conserva los cambios realizados en el contenido de este campo por una salida de cualquier tipo, y los hace visibles para las invocaciones subsiguientes de salidas (independientemente del tipo) para esta instancia de MCA. Estos cambios no

afectan a la definición de canal utilizada por otras instancias de MCA. Se puede utilizar cualquier carácter (incluidos los datos binarios).

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_REQUESTER, MQCHT_RECEIVER o MQCHT_CLUSRCVR.

La longitud de este campo la proporciona MQ_EXIT_DATA_LENGTH. Este campo no está presente cuando *Version* es menor que MQCD_VERSION_3.

Este campo no es relevante en IBM MQ for IBM i.

MsgUserData (MQCHAR32)

Este campo especifica los datos de usuario de salida de mensajes de canal.

Estos datos se pasan a la salida de mensajes de canal en el campo *ExitData* del parámetro **ChannelExitParms** (consulte MQ_CHANNEL_EXIT).

Este campo contiene inicialmente los datos que se han establecido en la definición de canal. Sin embargo, durante el tiempo de vida de esta instancia de MCA, el MCA conserva los cambios realizados en el contenido de este campo por una salida de cualquier tipo, y los hace visibles para las invocaciones subsiguientes de salidas (independientemente del tipo) para esta instancia de MCA. Estos cambios no afectan a la definición de canal utilizada por otras instancias de MCA. Se puede utilizar cualquier carácter (incluidos los datos binarios).

La longitud de este campo la proporciona MQ_EXIT_DATA_LENGTH.

Este campo no es relevante en IBM MQ for IBM i.

MsgUserDataPtr (MQPTR)

Este campo especifica la dirección del primer campo *MsgUserData*.

Si *MsgExitsDefined* es mayor que cero, esta dirección es la dirección de la lista de elementos de datos de usuario para cada salida de mensaje de canal de la cadena.

Cada elemento de datos de usuario está en un campo de longitud *ExitDataLength*, rellenado a la derecha con espacios en blanco. Hay *MsgExitsDefined* campos que se unen entre sí uno para cada salida. Si el número de elementos de datos de usuario definidos es menor que el número de nombres de salida, los elementos de datos de usuario no definidos se establecen en blancos. Por el contrario, si el número de elementos de datos de usuario definidos es mayor que el número de nombres de salida, los elementos de datos de usuario en exceso se ignoran y no se presentan a la salida.

Los cambios realizados en estos valores por una salida se conservan. Esto permite que una salida pase información a otra salida. No se realiza ninguna validación en ningún cambio, por lo que, por ejemplo, los datos binarios se pueden escribir en estos campos si es necesario.

Si *MsgExitsDefined* es cero, este campo es el puntero nulo.

En plataformas en las que el lenguaje de programación no soporta el tipo de datos de puntero, este campo se declara como una serie de bytes de la longitud adecuada.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_4.

NetworkPriority (MQLONG)

Este campo especifica la prioridad de la conexión de red para el canal.

Cuando hay disponibles varias vías de acceso a un destino determinado, se elige la vía de acceso con la prioridad más alta. El valor está en el rango de 0 a 9; 0 es la prioridad más baja.

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_CLUSSDR o MQCHT_CLUSRCVR.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_5.

Los campos siguientes de esta estructura no están presentes si *Version* es menor que MQCD_VERSION_6.

NonPersistentMsgSpeed (MQLONG)

Este campo especifica la velocidad a la que viajan los mensajes no persistentes a través del canal.

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_SENDER, MQCHT_SERVER, MQCHT_RECEIVER, MQCHT_REQUESTER, MQCHT_CLUSSDR o MQCHT_CLUSRCVR.

El valor puede ser uno de los siguientes:

MQNPMS_NORMAL

Velocidad normal.

Si un canal está definido como MQNPMS_NORMAL, los mensajes no persistentes viajan a través del canal a una velocidad normal. Esto tiene la ventaja de que estos mensajes no se pierden si hay una anomalía de canal. Además, los mensajes persistentes y no persistentes en la misma cola de transmisión mantienen su orden relativo entre sí.

MQNPMS_FAST

Velocidad rápida.

Si un canal está definido como MQNPMS_FAST, los mensajes no persistentes viajan a través del canal a una velocidad rápida. Esto mejora el rendimiento del canal, pero significa que los mensajes no persistentes se pierden si se produce una anomalía de canal. Además, es posible que los mensajes no persistentes salten por delante de los mensajes persistentes que esperan en la misma cola de transmisión, es decir, el orden de los mensajes no persistentes no se mantiene en relación con los mensajes persistentes. Sin embargo, se mantiene el orden de los mensajes no persistentes relativos entre sí. De forma similar, se mantiene el orden de los mensajes persistentes relativos entre sí.

Contraseña (MQCHAR12)

Este campo especifica la contraseña utilizada por el agente de canal de mensajes al intentar iniciar una sesión SNA segura con un agente de canal de mensajes remoto.

Este campo no puede estar en blanco sólo en AIX, Linux, and Windows, y sólo es relevante para canales con un *ChannelType* de MQCHT_SENDER, MQCHT_SERVER, MQCHT_REQUESTER o MQCHT_CLNTCONN. En z/OS, este campo no es relevante.

La longitud de este campo la proporciona MQ_PASSWORD_LENGTH. Sin embargo, sólo se utilizan los primeros 10 caracteres.

Este campo no está presente si *Version* es menor que MQCD_VERSION_2.

PropertyControl (MQLONG)

Este campo especifica qué sucede con las propiedades de los mensajes cuando el mensaje está a punto de enviarse a un gestor de colas V6 o anterior (un gestor de colas que no entiende el concepto de un descriptor de propiedad).

El valor puede ser cualquiera de los valores siguientes:

COMPATIBILIDAD de MQPROP_COMPATIBILITY

Si el mensaje contiene una propiedad con el prefijo **mcd.**, **jms.**, **usr.** o **mqext.**, todas las propiedades del mensaje se entregan a la aplicación en una cabecera MQRFH2. De lo contrario, todas las propiedades del mensaje, excepto las propiedades contenidas en el descriptor de mensaje (o extensión), se descartan y ya no son accesibles para la aplicación.

Este valor es el valor predeterminado; permite que las aplicaciones, que esperan que las propiedades relacionadas con JMS estén en una cabecera MQRFH2 en los datos del mensaje, continúen trabajando sin modificar.

MQPROP_NONE

Todas las propiedades del mensaje, excepto las propiedades del descriptor de mensaje (o extensión), se eliminan del mensaje antes de que el mensaje se envíe al gestor de colas remoto.

MQPROP_ALL

Todas las propiedades del mensaje se incluyen con el mensaje cuando se envía al gestor de colas remoto. Las propiedades, excepto aquellas que se encuentran en el descriptor de mensaje (o extensión), se colocan en una o más cabeceras MQRFH2 en los datos del mensaje.

Este atributo es aplicable a los canales Remitente, Servidor, Remitente de clúster y Receptor de clúster.

[“MQIA_* \(Selektors de atributos enteros\)” en la página 68](#)

[“MQPROP_* \(Valores de control de propiedades de cola y canal y longitud máxima de propiedades\)” en la página 108](#)

PutAuthority (MQLONG)

Este campo especifica si el identificador de usuario en la información de contexto asociada con un mensaje se utiliza para establecer la autoridad para colocar el mensaje en la cola de destino.

Este campo solo es relevante para canales con un *ChannelType* de MQCHT_REQUESTER, MQCHT_RECEIVER o MQCHT_CLUSRCVR. Es uno de los siguientes:

MQPA_DEFAULT

Se utiliza el identificador de usuario predeterminado.

MQPA_CONTEXTO

Se utiliza el identificador de usuario del contexto.

MQPA_ALTERNATIVO_O_MCA

Se utiliza el ID de usuario del campo « UserIdentifier » del descriptor del mensaje. No se utiliza ningún ID de usuario recibido de la red. Este valor solo es compatible con z/OS.

MQPA_SOLO_MCA

Se utiliza el ID de usuario predeterminado. No se utiliza ningún ID de usuario recibido de la red. Este valor solo es compatible con z/OS.

QuantumSafeAlgorithm (MQLONG)

Este campo especifica el algoritmo «Quantum Safe» que se debe utilizar para proteger una conexión TLS.

El valor puede ser uno de los siguientes:

MQQS_SIN PROTECCIÓN

No utilices algoritmos resistentes a la computación cuántica.

MQQS_HYBMLKEM

Utiliza el algoritmo híbrido ML-KEM, seguro desde el punto de vista cuántico.

MQQS_MLKEM

Utiliza el algoritmo ML-KEM, seguro frente a la computación cuántica.

Este es un campo de entrada hacia la salida. Este campo no aparece si *Version* es inferior a MQCD_VERSION_13.

Para más información, consulte «Quantum Safe» en TLS, en IBM MQ.

QuantumSafeRequired (MQLONG)

Este campo especifica si se requieren comunicaciones «Quantum Safe» o si se permiten soluciones alternativas tradicionales de « TLS ».

El valor puede ser uno de los siguientes:

MQQSR_OPCIONAL

Permitir comunicaciones tradicionales (TLS) o comunicaciones a prueba de quantum.

MQQSR_OBLIGATORIO

Evitar que se inicie el canal si no fuera posible establecer una comunicación Quantum Safe.

Este es un campo de entrada hacia la salida. Este campo no aparece si *Version* es inferior a MQCD_VERSION_13.

Para más información, consulte «Quantum Safe» en TLS, en IBM MQ.

QMgrName (MQCHAR48)

Este campo especifica el nombre del gestor de colas al que se puede conectar una salida.

Para canales con un *ChannelType* distinto de MQCHT_CLNTCONN, este campo es el nombre del gestor de colas al que se puede conectar una salida, que en AIX, Linux, and Windows, siempre no está en blanco.

La longitud de este campo la proporciona MQ_Q_MGR_NAME_LENGTH.

ReceiveExit (MQCHARn)

Este campo especifica el nombre de salida de recepción de canal.

Si este nombre no está en blanco, se llama a la salida en las horas siguientes:

- Inmediatamente antes de que se procesen los datos de red recibidos.

A la salida se le proporciona el almacenamiento intermedio de transmisión completo tal como se ha recibido. El contenido del almacenamiento intermedio se puede modificar según sea necesario.

- Durante la inicialización y terminación del canal.

Consulte “MQCD-Definición de canal” en la [página 1502](#) para obtener una descripción del contenido de este campo en varios entornos.

La longitud de este campo la proporciona MQ_EXIT_NAME_LENGTH.

Nota: El valor de esta constante es específico del entorno.

ReceiveExitPtr (MQPTR)

Este campo especifica la dirección del primer campo *ReceiveExit*.

Si *ReceiveExitsDefined* es mayor que cero, esta dirección es la dirección de la lista de nombres de cada salida de recepción de canal de la cadena.

Cada nombre está en un campo de longitud *ExitNameLength*, rellenado a la derecha con espacios en blanco. Hay *ReceiveExitsDefined* campos que se unen entre sí-uno para cada salida.

Los cambios realizados en estos nombres por una salida se conservan, aunque la salida de canal de mensajes no realiza ninguna acción explícita-no cambia qué salidas se invocan.

Si *ReceiveExitsDefined* es cero, este campo es el puntero nulo.

En plataformas en las que el lenguaje de programación no soporta el tipo de datos de puntero, este campo se declara como una serie de bytes de la longitud adecuada.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_4.

ReceiveExitsDefined (MQLONG)

Este campo especifica el número de salidas de recepción de canal definidas en la cadena.

Es mayor o igual que cero.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_4.

ReceiveUserData (MQCHAR32)

Este canal especifica los datos de usuario de salida de recepción de canal.

Estos datos se pasan a la salida de recepción de canal en el campo *ExitData* del parámetro **ChannelExitParms** (consulte MQ_CHANNEL_EXIT).

Este campo contiene inicialmente los datos que se han establecido en la definición de canal. Sin embargo, durante el tiempo de vida de esta instancia de MCA, el MCA conserva los cambios realizados en el contenido de este campo por una salida de cualquier tipo, y los hace visibles para las invocaciones subsiguientes de salidas (independientemente del tipo) para esta instancia de MCA. Esto se aplica a las

salidas en diferentes conversaciones. Estos cambios no afectan a la definición de canal utilizada por otras instancias de MCA. Se puede utilizar cualquier carácter (incluidos los datos binarios).

La longitud de este campo la proporciona MQ_EXIT_DATA_LENGTH.

Este campo no es relevante en IBM MQ for IBM i.

Los campos siguientes de esta estructura no están presentes si *Version* es menor que MQCD_VERSION_2.

ReceiveUserDataPtr (MQPTR)

Este campo especifica la dirección del primer campo *ReceiveUserData*.

Si *ReceiveExitsDefined* es mayor que cero, esta dirección es la dirección de la lista de elementos de datos de usuario para cada salida de recepción de canal de la cadena.

Cada elemento de datos de usuario está en un campo de longitud *ExitDataLength*, rellenado a la derecha con espacios en blanco. Hay *ReceiveExitsDefined* campos que se unen entre sí uno para cada salida. Si el número de elementos de datos de usuario definidos es menor que el número de nombres de salida, los elementos de datos de usuario no definidos se establecen en blancos. Por el contrario, si el número de elementos de datos de usuario definidos es mayor que el número de nombres de salida, los elementos de datos de usuario en exceso se ignoran y no se presentan a la salida.

Los cambios realizados en estos valores por una salida se conservan. Esto permite que una salida pase información a otra salida. No se realiza ninguna validación en ningún cambio, por lo que, por ejemplo, los datos binarios se pueden escribir en estos campos si es necesario.

Si *ReceiveExitsDefined* es cero, este campo es el puntero nulo.

En plataformas en las que el lenguaje de programación no soporta el tipo de datos de puntero, este campo se declara como una serie de bytes de la longitud adecuada.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_4.

Los campos siguientes de esta estructura no están presentes si *Version* es menor que MQCD_VERSION_5.

RemotePassword (MQCHAR12)

Este campo especifica la contraseña de un socio.

Este campo sólo contiene información válida si *ChannelType* es MQCHT_CLNTCONN o MQCHT_SVRCONN.

- Para una salida de seguridad en un canal MQCHT_CLNTCONN, esta contraseña es una contraseña que se ha obtenido del entorno. La salida puede elegir enviarla a la salida de seguridad en el servidor.
- Para una salida de seguridad en un canal MQCHT_SVRCONN, este campo puede contener una contraseña que se ha obtenido del entorno en el cliente, si no hay ninguna salida de seguridad de cliente. La salida puede utilizar esta contraseña para validar el identificador de usuario en *RemoteUserIdentifier*.

Si hay una salida de seguridad en el cliente, esta información se puede obtener en un flujo de seguridad del cliente.

La longitud de este campo la proporciona MQ_PASSWORD_LENGTH. Este campo no está presente si *Version* es menor que MQCD_VERSION_2.

RemoteSecurityId (MQBYTE40)

Este campo especifica el identificador de seguridad para el usuario remoto.

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_CLNTCONN o MQCHT_SVRCONN.

El siguiente valor especial indica que no hay ningún identificador de seguridad:

MQSID_NONE

No se ha especificado ningún identificador de seguridad.

El valor es cero binario para la longitud del campo.

Para el lenguaje de programación C, también se define la constante MQSID_NONE_ARRAY; esta constante tiene el mismo valor que MQSID_NONE, pero es una matriz de caracteres en lugar de una serie.

Es un campo de entrada para la salida. La longitud de este campo la proporciona MQ_SECURITY_ID_LENGTH. Este campo no está presente si *Version* es menor que MQCD_VERSION_6.

Los campos siguientes de esta estructura no están presentes si *Version* es menor que MQCD_VERSION_7.

RemoteUserIdentifier (MQCHAR12)

Este campo especifica los primeros 12 bytes de un identificador de usuario de un socio.

Hay dos campos que contienen el identificador de usuario remoto:

- *RemoteUserIdentifier* contiene los primeros 12 bytes del identificador de usuario remoto y se rellena con espacios en blanco si el identificador es inferior a 12 bytes. *RemoteUserIdentifier* puede estar en blanco.
- *LongRemoteUserIdPtr* apunta al identificador de usuario remoto completo, que puede tener más de 12 bytes. Su longitud la proporciona *LongRemoteUserIdLength*. El identificador completo no contiene blancos de cola y no termina en nulo. Si el identificador está en blanco, *LongRemoteUserIdLength* es cero y el valor de *LongRemoteUserIdPtr* no está definido.

LongRemoteUserIdPtr no está presente si *Version* es menor que MQCD_VERSION_6.

El identificador de usuario remoto sólo es relevante para los canales con un *ChannelType* de MQCHT_CLNTCONN o MQCHT_SVRCONN.

- Para una salida de seguridad en un canal MQCHT_CLNTCONN, este valor es un identificador de usuario que se ha obtenido del entorno. La salida puede elegir enviarla a la salida de seguridad en el servidor.
- Para una salida de seguridad en un canal MQCHT_SVRCONN, este campo puede contener un identificador de usuario que se ha obtenido del entorno en el cliente, si no hay ninguna salida de seguridad de cliente. La salida puede validar este ID de usuario (posiblemente con la contraseña en *RemotePassword*) y actualizar el valor en *MCAUserIdentifier*.

Si hay una salida de seguridad en el cliente, esta información se puede obtener en un flujo de seguridad del cliente.

La longitud de este campo la proporciona MQ_USER_ID_LENGTH. Este campo no está presente si *Version* es menor que MQCD_VERSION_2.

SecurityExit (MQCHARn)

Este campo especifica el nombre de salida de seguridad de canal.

Si este nombre no está en blanco, se llama a la salida en las horas siguientes:

- Inmediatamente después de establecer un canal.

Antes de transferir ningún mensaje, se da la oportunidad a la salida de causar flujos de mensajes de seguridad para validar la autorización de conexión.

- Al recibir una respuesta a un flujo de mensajes de seguridad.

Los flujos de mensajes de seguridad recibidos del procesador remoto en la máquina remota se asignan a la salida.

- Durante la inicialización y terminación del canal.

Consulte “MQCD-Definición de canal” en la [página 1502](#) para obtener una descripción del contenido de este campo en varios entornos.

La longitud de este campo la proporciona MQ_EXIT_NAME_LENGTH.

Nota: El valor de esta constante es específico del entorno.

SecurityUserData (MQCHAR32)

Este canal especifica los datos de usuario de salida de seguridad de canal.

Estos datos se pasan a la salida de seguridad de canal en el campo *ExitData* del parámetro **ChannelExitParms** (consulte MQ_CHANNEL_EXIT).

Este campo contiene inicialmente los datos que se han establecido en la definición de canal. Sin embargo, durante el tiempo de vida de esta instancia de MCA, el MCA conserva los cambios realizados en el contenido de este campo por una salida de cualquier tipo, y los hace visibles para las invocaciones subsiguientes de salidas (independientemente del tipo) para esta instancia de MCA. Esto se aplica a las salidas en diferentes conversaciones. Estos cambios no afectan a la definición de canal utilizada por otras instancias de MCA. Se puede utilizar cualquier carácter (incluidos los datos binarios).

La longitud de este campo la proporciona MQ_EXIT_DATA_LENGTH.

Este campo no es relevante en IBM MQ for IBM i.

SendExit (MQCHARn)

Este campo especifica el nombre de salida de emisión de canal.

Si este nombre no está en blanco, se llama a la salida en las horas siguientes:

- Inmediatamente antes de que se envíen los datos en la red.

A la salida se le proporciona el almacenamiento intermedio de transmisión completo antes de que se transmita. El contenido del almacenamiento intermedio se puede modificar según sea necesario.

- Durante la inicialización y terminación del canal.

Consulte “MQCD-Definición de canal” en la [página 1502](#) para obtener una descripción del contenido de este campo en varios entornos.

La longitud de este campo la proporciona MQ_EXIT_NAME_LENGTH.

Nota: El valor de esta constante es específico del entorno.

SendExitPtr (MQPTR)

Este campo especifica la dirección del primer campo *SendExit*.

Si *SendExitsDefined* es mayor que cero, esta dirección es la dirección de la lista de nombres de cada salida de envío de canal de la cadena.

Cada nombre está en un campo de longitud *ExitNameLength*, rellenado a la derecha con espacios en blanco. Hay *SendExitsDefined* campos que se unen entre sí-uno para cada salida.

Los cambios realizados en estos nombres por una salida se conservan, aunque la salida de envío de mensajes no realiza ninguna acción explícita-no cambia qué salidas se invocan.

Si *SendExitsDefined* es cero, este campo es el puntero nulo.

En plataformas en las que el lenguaje de programación no soporta el tipo de datos de puntero, este campo se declara como una serie de bytes de la longitud adecuada.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_4.

SendExitsDefined (MQLONG)

Este campo especifica el número de salidas de envío de canal definidas en la cadena.

Es mayor o igual que cero.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_4.

SendUserData (MQCHAR32)

Este campo especifica los datos de usuario de salida de emisión de canal.

Estos datos se pasan a la salida de emisión de canal en el campo *ExitData* del parámetro **ChannelExitParms** (consulte MQ_CHANNEL_EXIT).

Este campo contiene inicialmente los datos que se han establecido en la definición de canal. Sin embargo, durante el tiempo de vida de esta instancia de MCA, el MCA conserva los cambios realizados en el contenido de este campo por una salida de cualquier tipo, y los hace visibles para las invocaciones subsiguientes de salidas (independientemente del tipo) para esta instancia de MCA. Esto se aplica a las salidas en diferentes conversaciones. Estos cambios no afectan a la definición de canal utilizada por otras instancias de MCA. Se puede utilizar cualquier carácter (incluidos los datos binarios).

La longitud de este campo la proporciona MQ_EXIT_DATA_LENGTH.

Este campo no es relevante en IBM MQ for IBM i.

SendUserDataPtr (MQPTR)

Este campo especifica la dirección del campo *SendUserData*.

Si *SendExitsDefined* es mayor que cero, esta dirección es la dirección de la lista de elementos de datos de usuario para cada salida de mensaje de canal de la cadena.

Cada elemento de datos de usuario está en un campo de longitud *ExitDataLength*, rellenado a la derecha con espacios en blanco. Hay *MsgExitsDefined* campos que se unen entre sí uno para cada salida. Si el número de elementos de datos de usuario definidos es menor que el número de nombres de salida, los elementos de datos de usuario no definidos se establecen en blancos. Por el contrario, si el número de elementos de datos de usuario definidos es mayor que el número de nombres de salida, los elementos de datos de usuario en exceso se ignoran y no se presentan a la salida.

Los cambios realizados en estos valores por una salida se conservan. Esto permite que una salida pase información a otra salida. No se realiza ninguna validación en ningún cambio, por lo que, por ejemplo, los datos binarios se pueden escribir en estos campos si es necesario.

Si *SendExitsDefined* es cero, este campo es el puntero nulo.

En plataformas en las que el lenguaje de programación no soporta el tipo de datos de puntero, este campo se declara como una serie de bytes de la longitud adecuada.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_4.

SeqNumberWrap (MQLONG)

Este campo especifica el número de secuencia de mensaje más alto permitido.

Cuando se alcanza este valor, los números de secuencia se reinician para empezar de nuevo en 1.

Este valor no es negociable y debe coincidir en las definiciones de canal local y remoto.

Este campo no es relevante para los canales con un *ChannelType* de MQCHT_SVRCONN o MQCHT_CLNTCONN.

SharingConversations (MQLONG)

Este campo especifica el número máximo de conversaciones que pueden compartir una instancia de canal asociada a este canal.

Este campo se utiliza en los canales de conexión de cliente y conexión de servidor.

Un valor de 0 significa que el canal funciona como lo hacía en versiones anteriores a IBM WebSphere MQ 7.0 con respecto a los siguientes atributos:

- Compartimiento de conversaciones
- Lectura hacia adelante
- STOP CHANNEL(*channelname*) MODE(QUIESCE)
- Pulsaciones
- Consumo asíncrono de cliente

El valor 1 es el valor mínimo para IBM MQ el comportamiento. Aunque solo se permite una conversación en la instancia del canal, se puede leer por adelantado, consumir de forma asíncrona y se dispone de CLNTCONN-SVRCONN latidos y parada del canal inactivo.

Este es un campo de entrada a la salida. No está presente si *Version* es menor que MQCD_VERSION_9.

El valor predeterminado de este campo es 10.

Nota: *MaxInstances* y *MaxInstancesPerClient* los límites aplicados a un canal restringen el número de instancias del canal, no el número de conversaciones que podrían compartir esas instancias.

ShortConnectionName (MQCHAR20)

Este campo especifica los primeros 20 bytes de un nombre de conexión.

Si el campo *Version* es MQCD_VERSION_1, *ShortConnectionName* contiene el nombre de conexión completo.

Si el campo *Version* es MQCD_VERSION_2 o superior, *ShortConnectionName* contiene los primeros 20 caracteres del nombre de conexión. El nombre completo de la conexión se proporciona mediante el campo *ConnectionName*; *ShortConnectionName* y los primeros 20 caracteres de *ConnectionName* son idénticos.

Consulte *ConnectionName* para obtener detalles del contenido de este campo.

Nota: El nombre de este campo se ha cambiado para MQCD_VERSION_2 y las versiones posteriores de MQCD; el campo se denominaba anteriormente *ConnectionName*.

La longitud de este campo la proporciona MQ_SHORT_CONN_NAME_LENGTH.

ShortRetryCount (MQLONG)

Este campo especifica el número máximo de intentos que se realizan para conectarse a una máquina remota.

Este campo es el número máximo de intentos que se realizan para conectarse a la máquina remota, a intervalos especificados por *ShortRetryInterval*, antes de que se utilicen *LongRetryCount* y *LongRetryInterval* (normalmente más largos).

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR o MQCHT_CLUSRCVR.

ShortRetryInterval (MQLONG)

Este campo especifica el número máximo de segundos que se debe esperar antes de volver a intentar la conexión con la máquina remota.

El intervalo entre reintentos se puede ampliar si el canal tiene que esperar a estar activo.

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR o MQCHT_CLUSRCVR.

SPLProtection (MQLONG)

Este campo especifica el valor de la protección de AMS la política de seguridad.

El valor puede ser uno de los siguientes:

MQSPL_PASSTHRU

Transmita, sin modificaciones, cualquier mensaje enviado o recibido por el MCA para este canal.

Este valor solo es relevante para canales con un *ChannelType* de MQCHT_SENDER, MQCHT_SERVER, MQCHT_RECEIVER o MQCHT_REQUESTER, y es el valor predeterminado.

MQSPL_REMOVER

Elimine cualquier AMS protección de los mensajes recuperados de la cola de transmisión por el MCA y envíe los mensajes al socio.

Este valor solo es relevante para canales con un *ChannelType* de MQCHT_SENDER o MQCHT_SERVER.

MQSPL_ASPOLICY

Basándose en la política definida para la cola de destino, aplique la protección de AMS a los mensajes de entrada antes de colocarlos en la cola de destino.

Este valor solo es relevante para canales con un *ChannelType* de MQCHT_RECEIVER o MQCHT_REQUESTER.

Este es un campo de entrada a la salida. Este campo no aparece si *Version* es menor que MQCD_VERSION_12.

SSLCipherSpec (MQCHAR32)

Este campo especifica la especificación de cifrado que se utiliza cuando se emplea la función « TLS ».

Si SSLCipherSpec está en blanco, el canal no utiliza TLS. Si no está en blanco, este campo contiene una cadena que especifica el servicio de correo electrónico (CipherSpec) en uso.

Este parámetro es válido para todos los tipos de canales. Es compatible con las siguientes plataformas:

-  AIX
-  IBM i
-  Linux
-  Windows
-  z/OS

Solo es válido para tipos de canal de un tipo de transporte (TRPTYPE) de TCP.

Este es un campo de entrada a la salida. La longitud de este campo viene dada por MQ_SSL_CIPHER_SPEC_LENGTH. El campo no está presente si *Version* es menor que MQCD_VERSION_7.

SSLClientAuth (MQLONG)

Este campo indica si se requiere la autenticación de cliente de « TLS ».

Este campo sólo es relevante para las definiciones de canal SVRCONN.

Es uno de los valores siguientes:

MQSCA_OBLIGATORIO

Se requiere autenticación de cliente.

MQSCA_OPCIONAL

Autenticación de cliente opcional.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_7.

SSLPeerNameLength (MQLONG)

SSLPeerNamePtr Este campo especifica la longitud en bytes del nombre de par TLS al que apunta.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_7.

SSLPeerNamePtr (MQPTR)

Este campo especifica la dirección del nombre del nodo « TLS ».

Cuando se recibe un certificado durante un protocolo de enlace « TLS » correcto, el nombre distintivo del sujeto del certificado se copia en el campo MQCD al que accede « SSLPeerNamePtr » al final del canal que recibe el certificado. Sobrescribe el valor « SSLPeerName » del canal si dicho valor está presente en la definición del canal del usuario local. Si se especifica una salida de seguridad en este extremo del canal, recibe el nombre distinguido del certificado de igual en el MQCD.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_7.

Nota: Es posible que sea necesario actualizar las aplicaciones de salida de seguridad construidas antes del release de IBM WebSphere MQ 7.1 . Para obtener más información, consulte [Programas de salida de seguridad de canal](#).

StrucLength (MQLONG)

Este campo especifica la longitud en bytes de la estructura MQCD.

La longitud no incluye ninguna de las cadenas a las que apuntan los campos de puntero contenidos en la estructura. El valor puede ser uno de los siguientes:

MQCD_LENGTH_4

Longitud de una estructura de definición de canal version-4.

MQCD_LENGTH_5

Longitud de una estructura de definición de canal version-5.

MQCD_LENGTH_6

Longitud de una estructura de definición de canal version-6.

MQCD_LENGTH_7

Longitud de una estructura de definición de canal version-7.

MQCD_LENGTH_8

Longitud de una estructura de definición de canal version-8.

MQCD_LENGTH_9

Longitud de una estructura de definición de canal version-9.

MQCD_LENGTH_10

Longitud de una estructura de definición de canal version-10.

MQCD_LENGTH_11

Longitud de una estructura de definición de canal version-11.

 MQCD_LENGTH_12

Longitud de una estructura de definición de canal version-12.

La siguiente constante especifica la longitud de la versión actual:

MQCD_LONGITUD_ACTUAL

Longitud de la versión actual de la estructura de definición del canal.

Nota: Estas constantes tienen valores específicos del entorno.

El campo no está presente si *Version* es menor que MQCD_VERSION_4.

TpName (MQCHAR64)

Este campo especifica el nombre de programa de transacción de LU 6.2 .

Este campo sólo es relevante si el protocolo de transmisión (*TransportType*) es MQXPT_LU62y *ChannelType* no es MQCHT_SVRCONN ni MQCHT_RECEIVER.

Este campo siempre está en blanco en las plataformas en las que la información está contenida en el objeto lateral de comunicaciones.

La longitud de este campo la proporciona MQ_TP_NAME_LENGTH.

TransportType (MQLONG)

Este campo especifica el protocolo de transmisión que se va a utilizar.

El valor no se comprueba si el canal se ha iniciado desde el otro extremo.

Es uno de los valores siguientes:

MQXPT_LU62

Protocolo de transporte LU 6.2 .

MQXPT_TCP

TCP/IP protocolo de transporte.

MQXPT_NETBIOS

Protocolo de transporte NetBIOS .

Este valor está soportado en los entornos siguientes: Windows.

MQXPT_SPX

Protocolo de transporte SPX.

Este valor está soportado en los entornos siguientes: Windows, más los clientes IBM MQ conectados a estos sistemas.

UseDLQ (MQLONG)

Este campo especifica si se utiliza la cola de mensajes no entregados (o la cola de mensajes no entregados) cuando los canales no pueden entregar los mensajes.

Puede contener uno de los valores siguientes:

MQUSEDLQ_NO

Los mensajes que un canal no puede entregar se tratan como un error. El canal descarta el mensaje o el canal finaliza, de acuerdo con el valor NPMSPEED.

MQUSEDLQ_SÍ

Cuando el atributo de gestor de colas DEADQ proporciona el nombre de una cola de mensajes no entregados, se utiliza, de lo contrario, el comportamiento es como para NO. YES es el valor predeterminado.

UserIdentifier (MQCHAR12)

Este campo especifica el identificador de usuario utilizado por el agente de canal de mensajes al intentar iniciar una sesión SNA segura con un agente de canal de mensajes remoto.

Este campo no puede estar en blanco sólo en AIX, Linux, and Windows, y sólo es relevante para canales con un *ChannelType* de MQCHT_SENDER, MQCHT_SERVER, MQCHT_REQUESTER o MQCHT_CLNTCONN. En z/OS, este campo no es relevante.

La longitud de este campo la proporciona MQ_USER_ID_LENGTH. Sin embargo, sólo se utilizan los primeros 10 caracteres.

Este campo no está presente cuando *Version* es menor que MQCD_VERSION_2.

Versión (MQLONG)

El *Version* campo especifica el número de versión más alto que se puede establecer para la estructura.

El valor depende del entorno:

MQCD _VERSION_1

Estructura de definición de canales de la versión 1.

MQCD _VERSION_2

Estructura de definición de canales de la versión 2.

MQCD _VERSION_3

Estructura de definición de canales de la versión 3.

MQCD _VERSION_4

Estructura de definición de canales de la versión 4.

MQCD _VERSION_5

Estructura de definición de canales de la versión 5.

MQCD _VERSION_6

Estructura de definición de canales de la versión 6.

MQCD _VERSION_7

Estructura de definición de canales de la versión 7.

MQCD _VERSION_8

Estructura de definición de canales de la versión 8.

MQCD _VERSION_9

Estructura de definición de canales de la versión 9.

MQCD _VERSION_10

Estructura de definición de canales de la versión 10.

MQCD _VERSION_11

Estructura de definición de canales de la versión 11.

La versión 11 es la más alta que se puede configurar en el campo en IBM MQ 8.0 todas las plataformas.

MQCD _VERSION_12

Estructura de definición de canales de la versión 12.

La versión 12 es la más alta que se puede establecer en el campo IBM MQ 9.1.3.

Los campos que solo existen en las versiones más recientes de la estructura se identifican como tales en las descripciones de los campos. La siguiente constante especifica el número de versión de la versión actual:

MQCD_CURRENT_VERSION

El valor establecido en MQCD_CURRENT_VERSION es la versión actual de la estructura de definición del canal que se está utilizando.

El valor de MQCD_CURRENT_VERSION depende del entorno. Contiene el valor más alto admitido por la plataforma.

MQCD_CURRENT_VERSION no se utiliza para inicializar las estructuras predeterminadas proporcionadas en los archivos de encabezado, copia e inclusión proporcionados para diferentes lenguajes de programación. La inicialización predeterminada de Version depende de la plataforma y la versión.

Las MQCD declaraciones en los archivos de encabezado, copia e inclusión se inicializan en MQCD_VERSION_6. Para utilizar campos MQCD adicionales, las aplicaciones deben establecer el número de versión en MQCD_CURRENT_VERSION. Si está escribiendo una aplicación que es portátil entre varios entornos, debe elegir una versión que sea compatible con todos los entornos.

Consejo: Cuando se introduce una nueva versión de la MQCD estructura, no se modifica el diseño de la parte existente. La salida debe comprobar el número de versión. Debe ser igual o superior a la versión más baja que contenga los campos que la salida necesita utilizar.

XmitQName (MQCHAR48)

Este campo especifica el nombre de la cola de transmisión de la que se recuperan los mensajes.

Este campo sólo es relevante para los canales con un *ChannelType* de MQCHT_SENDER o MQCHT_SERVER.

La longitud de este campo la proporciona MQ_Q_NAME_LENGTH.

Declaración C

Esta declaración es la declaración C para la estructura MQCD.

 V1000

```
typedef struct tagMQCD MQCD;
typedef MQCD MQPOINTER PMQCD;
typedef PMQCD MQPOINTER PPMQCD;

struct tagMQCD {
    MQCHAR    ChannelName[20];          /* Channel definition name */
    MQLONG    Version;                  /* Structure version number */
};
```

MQLONG	ChannelType;	/* Channel type */
MQLONG	TransportType;	/* Transport type */
MQCHAR	Desc[64];	/* Channel description */
MQCHAR	QMgrName[48];	/* Queue manager name */
MQCHAR	XmitQName[48];	/* Transmission queue name */
MQCHAR	ShortConnectionName[20];	/* First 20 bytes of */
		/* connection name */
MQCHAR	MCAName[20];	/* Reserved */
MQCHAR	ModeName[8];	/* LU 6.2 Mode name */
MQCHAR	TpName[64];	/* LU 6.2 transaction program */
		/* name */
MQLONG	BatchSize;	/* Batch size */
MQLONG	DiscInterval;	/* Disconnect interval */
MQLONG	ShortRetryCount;	/* Short retry count */
MQLONG	ShortRetryInterval;	/* Short retry wait interval */
MQLONG	LongRetryCount;	/* Long retry count */
MQLONG	LongRetryInterval;	/* Long retry wait interval */
MQCHAR	SecurityExit[128];	/* Channel security exit name */
MQCHAR	MsgExit[128];	/* Channel message exit name */
MQCHAR	SendExit[128];	/* Channel send exit name */
MQCHAR	ReceiveExit[128];	/* Channel receive exit name */
MQLONG	SeqNumberWrap;	/* Highest allowable message */
		/* sequence number */
MQLONG	MaxMsgLength;	/* Maximum message length */
MQLONG	PutAuthority;	/* Put authority */
MQLONG	DataConversion;	/* Data conversion */
MQCHAR	SecurityUserData[32];	/* Channel security exit user */
		/* data */
MQCHAR	MsgUserData[32];	/* Channel message exit user */
		/* data */
MQCHAR	SendUserData[32];	/* Channel send exit user */
		/* data */
MQCHAR	ReceiveUserData[32];	/* Channel receive exit user */
		/* data */
/* Ver:1 */		
MQCHAR	UserIdentifier[12];	/* User identifier */
MQCHAR	Password[12];	/* Password */
MQCHAR	MCAUserIdentifier[12];	/* First 12 bytes of MCA user */
		/* identifier */
MQLONG	MCAType;	/* Message channel agent type */
MQCHAR	ConnectionName[264];	/* Connection name */
MQCHAR	RemoteUserIdentifier[12];	/* First 12 bytes of user */
		/* identifier from partner */
MQCHAR	RemotePassword[12];	/* Password from partner */
/* Ver:2 */		
MQCHAR	MsgRetryExit[128];	/* Channel message retry exit */
		/* name */
MQCHAR	MsgRetryUserData[32];	/* Channel message retry exit */
		/* user data */
MQLONG	MsgRetryCount;	/* Number of times MCA will */
		/* try to put the message, */
		/* after first attempt has */
		/* failed */
MQLONG	MsgRetryInterval;	/* Minimum interval in */
		/* milliseconds after which */
		/* the open or put operation */
		/* will be retried */
/* Ver:3 */		
MQLONG	HeartbeatInterval;	/* Time in seconds between */
		/* heartbeat flows */
MQLONG	BatchInterval;	/* Batch duration */
MQLONG	NonPersistentMsgSpeed;	/* Speed at which */
		/* nonpersistent messages are */
		/* sent */
MQLONG	StrucLength;	/* Length of MQCD structure */
MQLONG	ExitNameLength;	/* Length of exit name */
MQLONG	ExitDataLength;	/* Length of exit user data */
MQLONG	MsgExitsDefined;	/* Number of message exits */
		/* defined */
MQLONG	SendExitsDefined;	/* Number of send exits */
		/* defined */
MQLONG	ReceiveExitsDefined;	/* Number of receive exits */
		/* defined */
MQPTR	MsgExitPtr;	/* Address of first MsgExit */
		/* field */
MQPTR	MsgUserDataPtr;	/* Address of first */
		/* MsgUserData field */
MQPTR	SendExitPtr;	/* Address of first SendExit */
		/* field */
MQPTR	SendUserDataPtr;	/* Address of first */
		/* SendUserData field */
MQPTR	ReceiveExitPtr;	/* Address of first */


```

MQPTR      ReceiveUserDataPtr;      /* ReceiveExit field */
/* Ver:4 */
MQPTR      ClusterPtr;              /* Address of first */
/* Address of a list of */
/* cluster names */
MQLONG     ClustersDefined;         /* Number of clusters to */
/* which the channel belongs */
/* Network priority */
MQLONG     NetworkPriority;
/* Ver:5 */
MQLONG     LongMCAUserIdLength;     /* Length of long MCA user */
/* identifier */
MQLONG     LongRemoteUserIdLength; /* Length of long remote user */
/* identifier */
MQPTR      LongMCAUserIdPtr;        /* Address of long MCA user */
/* identifier */
MQPTR      LongRemoteUserIdPtr;     /* Address of long remote */
/* user identifier */
MQBYTE40   MCASecurityId;           /* MCA security identifier */
MQBYTE40   RemoteSecurityId;        /* Remote security identifier */
/* Ver:6 */
MQCHAR     SSLCipherSpec[32];       /* TLS CipherSpec */
MQPTR      SSLPeerNamePtr;          /* Address of TLS peer name */
MQLONG     SSLPeerNameLength;       /* Length of TLS peer name */
MQLONG     SSLClientAuth;           /* Whether TLS client */
/* authentication is required */
/* Keepalive interval */
MQLONG     KeepAliveInterval;
MQCHAR     LocalAddress[48];        /* Local communications */
/* address */
MQLONG     BatchHeartbeat;          /* Batch heartbeat interval */
/* Ver:7 */
MQLONG     HdrCompList[2];          /* Header data compression */
/* list */
MQLONG     MsgCompList[16];         /* Message data compression */
/* list */
MQLONG     CLWLChannelRank;         /* Channel rank */
MQLONG     CLWLChannelPriority;      /* Channel priority */
MQLONG     CLWLChannelWeight;       /* Channel weight */
MQLONG     ChannelMonitoring;       /* Channel monitoring */
MQLONG     ChannelStatistics;       /* Channel statistics */
/* Ver:8 */
MQLONG     SharingConversations;    /* Limit on sharing */
/* conversations */
MQLONG     PropertyControl;          /* Message property control */
MQLONG     MaxInstances;            /* Limit on SVRCONN channel */
/* instances */
MQLONG     MaxInstancesPerClient;    /* Limit on SVRCONN channel */
/* instances per client */
MQLONG     ClientChannelWeight;     /* Client channel weight */
MQLONG     ConnectionAffinity;      /* Connection affinity */
/* Ver:9 */
MQLONG     BatchDataLimit;          /* Batch data limit */
MQLONG     UseDLQ;                  /* Use Dead Letter Queue */
MQLONG     DefReconnect;            /* Default client reconnect */
/* option */
/* Ver:10 */
MQCHAR64   CertificateLabel;        /* Certificate label */
/* Ver:11 */
MQLONG     SPLProtection             /* AMS Security policy protection */
/* Ver:12 */
MQLONG     QuantumSafeAlgorithm;    /* Quantum Cryptography */
/* Algorithm */
MQLONG     QuantumSafeRequired;     /* Quantum Cryptography */
/* Requirement */
/* Ver:13 */
};

```

COBOL declaración

Esta declaración es la declaración « COBOL » para la estructura MQCD.

```

** MQCD structure
   10 MQCD.
      ** Channel definition name
      15 MQCD-CHANNELNAME PIC X(20).
      ** Structure version number
      15 MQCD-VERSION PIC S9(9) BINARY.
      ** Channel type
      15 MQCD-CHANNELTYPE PIC S9(9) BINARY.

```

```

** Transport type
15 MQCD-TRANSPORTTYPE PIC S9(9) BINARY.
** Channel description
15 MQCD-DESC PIC X(64).
** Queue manager name
15 MQCD-QMGRNAME PIC X(48).
** Transmission queue name
15 MQCD-XMITQNAME PIC X(48).
** First 20 bytes of connection name
15 MQCD-SHORTCONNECTIONNAME PIC X(20).
** Reserved
15 MQCD-MCNAME PIC X(20).
** LU 6.2 Mode name
15 MQCD-MODENAME PIC X(8).
** LU 6.2 transaction program name
15 MQCD-TPNAME PIC X(64).
** Batch size
15 MQCD-BATCHSIZE PIC S9(9) BINARY.
** Disconnect interval
15 MQCD-DISCINTERVAL PIC S9(9) BINARY.
** Short retry count
15 MQCD-SHORTRETRYCOUNT PIC S9(9) BINARY.
** Short retry wait interval
15 MQCD-SHORTRETRYINTERVAL PIC S9(9) BINARY.
** Long retry count
15 MQCD-LONGRETRYCOUNT PIC S9(9) BINARY.
** Long retry wait interval
15 MQCD-LONGRETRYINTERVAL PIC S9(9) BINARY.
** Channel security exit name
15 MQCD-SECURITYEXIT PIC X(20).
** Channel message exit name
15 MQCD-MSGEXIT PIC X(20).
** Channel send exit name
15 MQCD-SENDEXIT PIC X(20).
** Channel receive exit name
15 MQCD-RECEIVEEXIT PIC X(20).
** Highest allowable message sequence number
15 MQCD-SEQNUMBERWRAP PIC S9(9) BINARY.
** Maximum message length
15 MQCD-MAXMSGLENGTH PIC S9(9) BINARY.
** Put authority
15 MQCD-PUTAUTHORITY PIC S9(9) BINARY.
** Data conversion
15 MQCD-DATACONVERSION PIC S9(9) BINARY.
** Channel security exit user data
15 MQCD-SECURITYUSERDATA PIC X(32).
** Channel message exit user data
15 MQCD-MSGUSERDATA PIC X(32).
** Channel send exit user data
15 MQCD-SENDUSERDATA PIC X(32).
** Channel receive exit user data
15 MQCD-RECEIVEUSERDATA PIC X(32).
** Ver:1 **
** User identifier
15 MQCD-USERIDENTIFIER PIC X(12).
** Password
15 MQCD-PASSWORD PIC X(12).
** First 12 bytes of MCA user identifier
15 MQCD-MCAUSERIDENTIFIER PIC X(12).
** Message channel agent type
15 MQCD-MCATYPE PIC S9(9) BINARY.
** Connection name
15 MQCD-CONNECTIONNAME PIC X(264).
** First 12 bytes of user identifier from partner
15 MQCD-REMOTEUSERIDENTIFIER PIC X(12).
** Password from partner
15 MQCD-REMOTEPASSWORD PIC X(12).
** Ver:2 **
** Channel message retry exit name
15 MQCD-MSGRETRYEXIT PIC X(20).
** Channel message retry exit user data
15 MQCD-MSGRETRYUSERDATA PIC X(32).
** Number of times MCA will try to put the message, after first
** attempt has failed
15 MQCD-MSGRETRYCOUNT PIC S9(9) BINARY.
** Minimum interval in milliseconds after which the open or put
** operation will be retried
15 MQCD-MSGRETRYINTERVAL PIC S9(9) BINARY.
** Ver:3 **
** Time in seconds between heartbeat flows
15 MQCD-HEARTBEATINTERVAL PIC S9(9) BINARY.
** Batch duration

```

```

15 MQCD-BATCHINTERVAL PIC S9(9) BINARY.
** Speed at which nonpersistent messages are sent
15 MQCD-NONPERSISTENTMSGSPD PIC S9(9) BINARY.
** Length of MQCD structure
15 MQCD-STRUCLNGTH PIC S9(9) BINARY.
** Length of exit name
15 MQCD-EXITNAMELENGTH PIC S9(9) BINARY.
** Length of exit user data
15 MQCD-EXITDATALENGTH PIC S9(9) BINARY.
** Number of message exits defined
15 MQCD-MSGEXITSDEFINED PIC S9(9) BINARY.
** Number of send exits defined
15 MQCD-SENDEXITSDEFINED PIC S9(9) BINARY.
** Number of receive exits defined
15 MQCD-RECEIVEEXITSDEFINED PIC S9(9) BINARY.
** Address of first MsgExit field
15 MQCD-MSGEXITPTR POINTER.
** Address of first MsgUserData field
15 MQCD-MSGUSERDATAPTR POINTER.
** Address of first SendExit field
15 MQCD-SENDEXITPTR POINTER.
** Address of first SendUserData field
15 MQCD-SENDUSERDATAPTR POINTER.
** Address of first ReceiveExit field
15 MQCD-RECEIVEEXITPTR POINTER.
** Address of first ReceiveUserData field
15 MQCD-RECEIVEUSERDATAPTR POINTER.
** Ver:4 **
** Address of a list of cluster names
15 MQCD-CLUSTERPTR POINTER.
** Number of clusters to which the channel belongs
15 MQCD-CLUSTERSDEFINED PIC S9(9) BINARY.
** Network priority
15 MQCD-NETWORKPRIORITY PIC S9(9) BINARY.
** Ver:5 **
** Length of long MCA user identifier
15 MQCD-LONGMCAUSERIDLENGTH PIC S9(9) BINARY.
** Length of long remote user identifier
15 MQCD-LONGREMOTEUSERIDLENGTH PIC S9(9) BINARY.
** Address of long MCA user identifier
15 MQCD-LONGMCAUSERIDPTR POINTER.
** Address of long remote user identifier
15 MQCD-LONGREMOTEUSERIDPTR POINTER.
** MCA security identifier
15 MQCD-MCASECURITYID PIC X(40).
** Remote security identifier
15 MQCD-REMOTESECURITYID PIC X(40).
** Ver:6 **
** TLS CipherSpec
15 MQCD-SSLCIPHERSPEC PIC X(32).
** Address of TLS peer name
15 MQCD-SSLPEERNAMEPTR POINTER.
** Length of TLS peer name
15 MQCD-SSLPEERNAMELENGTH PIC S9(9) BINARY.
** Whether TLS client authentication is required
15 MQCD-SSLCLIENTAUTH PIC S9(9) BINARY.
** Keepalive interval
15 MQCD-KEEPALIVEINTERVAL PIC S9(9) BINARY.
** Local communications address
15 MQCD-LOCALADDRESS PIC X(48).
** Batch heartbeat interval
15 MQCD-BATCHHEARTBEAT PIC S9(9) BINARY.
** Ver:7 **
** Header data compression list
15 MQCD-HDRCOMPLIST PIC S9(9) BINARY.
** Message data compression list
15 MQCD-MSGCOMPLIST PIC S9(9) BINARY.
** Channel rank
15 MQCD-CLWLCHANNELRANK PIC S9(9) BINARY.
** Channel priority
15 MQCD-CLWLCHANNELPRIORITY PIC S9(9) BINARY.
** Channel weight
15 MQCD-CLWLCHANNELWEIGHT PIC S9(9) BINARY.
** Channel monitoring
15 MQCD-CHANNELMONITORING PIC S9(9) BINARY.
** Channel statistics
15 MQCD-CHANNELSTATISTICS PIC S9(9) BINARY.
** Ver:8 **
** Limit on sharing conversations
15 MQCD-SHARINGCONVERSATIONS PIC S9(9) BINARY.
** Message property control
15 MQCD-PROPERTYCONTROL PIC S9(9) BINARY.

```

```

** Limit on SVRCONN channel instances
  15 MQCD-MAXINSTANCES PIC S9(9) BINARY.
** Limit on SVRCONN channel instances per client
  15 MQCD-MAXINSTANCESPERCLIENT PIC S9(9) BINARY.
** Client channel weight
  15 MQCD-CLIENTCHANNELWEIGHT PIC S9(9) BINARY.
** Connection affinity
  15 MQCD-CONNECTIONAFFINITY PIC S9(9) BINARY.
** Ver:9 **
** Batch data limit
  15 MQCD-BATCHDATA LIMIT PIC S9(9) BINARY.
** Use Dead Letter Queue
  15 MQCD-USEDLQ PIC S9(9) BINARY.
** Default client reconnect option
  15 MQCD-DEFRECONNECT PIC S9(9) BINARY.
** Ver:10 **
** Certificate Label
  15 MQCD-CERTLABL PIC X (64)
** Ver:11 **
** AMS Security policy protection
  15 MQCD-SPLPROTECTION PIC S9(9) BINARY
** Ver:12 **

```

Declaración RPG (ILE)

Esta declaración es la declaración RPG para la estructura MQCD.

```

D* MQCD Structure
D*
D* Channel definition name
D CDCHN          1          20
D* Structure version number
D CDVER          21          24I 0
D* Channel type
D CDCHT          25          28I 0
D* Transport type
D CDTRT          29          32I 0
D* Channel description
D CDDES          33          96
D* Queue manager name
D CDQM           97          144
D* Transmission queue name
D CDXQ           145          192
D* First 20 bytes of connection name
D CDSCN          193          212
D* Reserved
D CDMCA          213          232
D* LU 6.2 Mode name
D CDMOD          233          240
D* LU 6.2 transaction program name
D CDTTP          241          304
D* Batch size
D CDBS           305          308I 0
D* Disconnect interval
D CDDI           309          312I 0
D* Short retry count
D CDSRC          313          316I 0
D* Short retry wait interval
D CDSRI          317          320I 0
D* Long retry count
D CDLRC          321          324I 0
D* Long retry wait interval
D CDLRI          325          328I 0
D* Channel security exit name
D CDSCX          329          348
D* Channel message exit name
D CDMSX          349          368
D* Channel send exit name
D CDSNX          369          388
D* Channel receive exit name
D CDRCX          389          408
D* Highest allowable message sequence number
D CDSNW          409          412I 0
D* Maximum message length
D CDMML          413          416I 0
D* Put authority
D CDPA           417          420I 0
D* Data conversion
D CDDC           421          424I 0
D* Channel security exit user data

```

```

D CDSCD 425 456
D* Channel message exit user data
D CDMSD 457 488
D* Channel send exit user data
D CDSND 489 520
D* Channel receive exit user data
D CDRCD 521 552
D* Ver:1 **
D* User identifier
D CDUID 553 564
D* Password
D CDPW 565 576
D* First 12 bytes of MCA user identifier
D CDAUI 577 588
D* Message channel agent type
D CDCAT 589 592I 0
D* Connection name
D CDCON 593 848
D CDCN2 849 856
D* First 12 bytes of user identifier from partner
D CDRUI 857 868
D* Password from partner
D CDRPW 869 880
D* Ver:2 **
D* Channel message retry exit name
D CDMRX 881 900
D* Channel message retry exit user data
D CDMRD 901 932
D* Number of times MCA will try to put the message, after first
D* attempt has failed
D CDMRC 933 936I 0
D* Minimum interval in milliseconds after which the open or put
D* operation will be retried
D CDMRI 937 940I 0
D* Ver:3 **
D* Time in seconds between heartbeat flows
D CDHBI 941 944I 0
D* Batch duration
D CDBI 945 948I 0
D* Speed at which nonpersistent messages are sent
D CDNPM 949 952I 0
D* Length of MQCD structure
D CDLEN 953 956I 0
D* Length of exit name
D CDXNL 957 960I 0
D* Length of exit user data
D CDXDL 961 964I 0
D* Number of message exits defined
D CDMXD 965 968I 0
D* Number of send exits defined
D CDSXD 969 972I 0
D* Number of receive exits defined
D CDRXD 973 976I 0
D* Address of first MsgExit field
D CDMXP 977 992*
D* Address of first MsgUserData field
D CDMUP 993 1008*
D* Address of first SendExit field
D CDSXP 1009 1024*
D* Address of first SendUserData field
D CDSUP 1025 1040*
D* Address of first ReceiveExit field
D CDRXP 1041 1056*
D* Address of first ReceiveUserData field
D CDRUP 1057 1072*
D* Ver:4 **
D* Address of a list of cluster names
D CDCLP 1073 1088*
D* Number of clusters to which the channel belongs
D CDCLD 1089 1092I 0
D* Network priority
D CDNP 1093 1096I 0
D* Ver:5 **
D* Length of long MCA user identifier
D CDLML 1097 1100I 0
D* Length of long remote user identifier
D CDLRL 1101 1104I 0
D* Address of long MCA user identifier
D CDLMP 1105 1120*
D* Address of long remote user identifier
D CDLRP 1121 1136*
D* MCA security identifier

```

```

D CDMSI 1137 1176
D* Remote security identifier
D CDRSI 1177 1216
D* Ver:6 **
D* TLS CipherSpec
D CDSCS 1217 1248
D* Address of TLS peer name
D CDSPN 1249 1264*
D* Length of TLS peer name
D CDSPL 1265 1268I 0
D* Whether TLS client authentication is required
D CDSCA 1269 1272I 0
D* Keepalive interval
D CDKAI 1273 1276I 0
D* Local communications address
D CDLOA 1277 1324
D* Batch heartbeat interval
D CDBHB 1325 1328I 0
D* Ver:7 **
D* Header data compression list
D CDHCL0
D CDHCL1 1329 1332I 0
D CDHCL2 1333 1336I 0
D CDHCL 10I 0 DIM(2) OVERLAY(CDHCL0)
D* Message data compression list
D CDMCL0
D CDMCL1 1337 1340I 0
D CDMCL2 1341 1344I 0
D CDMCL3 1345 1348I 0
D CDMCL4 1349 1352I 0
D CDMCL5 1353 1356I 0
D CDMCL6 1357 1360I 0
D CDMCL7 1361 1364I 0
D CDMCL8 1365 1368I 0
D CDMCL9 1369 1372I 0
D CDMCL10 1373 1376I 0
D CDMCL11 1377 1380I 0
D CDMCL12 1381 1384I 0
D CDMCL13 1385 1388I 0
D CDMCL14 1389 1392I 0
D CDMCL15 1393 1396I 0
D CDMCL16 1397 1400I 0
D CDMCL 10I 0 DIM(16) OVERLAY(CDMCL0)
D* Channel rank
D CDCWCR 1401 1404I 0
D* Channel priority
D CDCWCP 1405 1408I 0
D* Channel weight
D CDCWCW 1409 1412I 0
D* Channel monitoring
D CDCHLMON 1413 1416I 0
D* Channel statistics
D CDCHLST 1417 1420I 0
D* Ver:8 **
D* Limit on sharing conversations
D CDSHC 1421 1424I 0
D* Message property control
D CDPRC 1425 1428I 0
D* Limit on SVRCONN channel instances
D CDMXIN 1429 1432I 0
D* Limit on SVRCONN channel instances per client
D CDMXIC 1433 1436I 0
D* Client channel weight
D CDCLNCHLW 1437 1440I 0
D* Connection affinity
D CDCONNAFF 1441 1444I 0
D* Ver:9 **
D* Batch data limit
D CDBDL 1445 1448I 0
D* Use Dead Letter Queue
D CDUDLQ 1449 1452I 0
D* Default client reconnect option
D CDDRCN 1453 1456I 0
D* Ver:10 **

```

System/390 declaración del ensamblador

Esta declaración es la declaración del ensamblador System/390 para la estructura MQCD.

MQCD	DSECT		
MQCD_CHANNELNAME	DS	CL20	Channel definition name
MQCD_VERSION	DS	F	Structure version number
MQCD_CHANNELTYPE	DS	F	Channel type
MQCD_TRANSPORTTYPE	DS	F	Transport type
MQCD_DESC	DS	CL64	Channel description
MQCD_QMGRNAME	DS	CL48	Queue manager name
MQCD_XMITQNAME	DS	CL48	Transmission queue name
MQCD_SHORTCONNECTIONNAME	DS	CL20	First 20 bytes of connection
*			name
MQCD_MCANAME	DS	CL20	Reserved
MQCD_MODENAME	DS	CL8	LU 6.2 Mode name
MQCD_TPNAME	DS	CL64	LU 6.2 transaction program name
MQCD_BATCHSIZE	DS	F	Batch size
MQCD_DISCINTERVAL	DS	F	Disconnect interval
MQCD_SHORTRETRYCOUNT	DS	F	Short retry count
MQCD_SHORTRETRYINTERVAL	DS	F	Short retry wait interval
MQCD_LONGRETRYCOUNT	DS	F	Long retry count
MQCD_LONGRETRYINTERVAL	DS	F	Long retry wait interval
MQCD_SECURITYEXIT	DS	CLn	Channel security exit name
MQCD_MSGEXIT	DS	CLn	Channel message exit name
MQCD_SENDEXIT	DS	CLn	Channel send exit name
MQCD_RECEIVEEXIT	DS	CLn	Channel receive exit name
MQCD_SEQNUMBERWRAP	DS	F	Highest allowable message
*			sequence number
MQCD_MAXMSGLLENGTH	DS	F	Maximum message length
MQCD_PUTAUTHORITY	DS	F	Put authority
MQCD_DATACONVERSION	DS	F	Data conversion
MQCD_SECURITYUSERDATA	DS	CL32	Channel security exit user data
MQCD_MSGUSERDATA	DS	CL32	Channel message exit user data
MQCD_SENDUSERDATA	DS	CL32	Channel send exit user data
MQCD_RECEIVEUSERDATA	DS	CL32	Channel receive exit user data
MQCD_USERIDENTIFIER	DS	CL12	User identifier
MQCD_PASSWORD	DS	CL12	Password
MQCD_MCAUSERIDENTIFIER	DS	CL12	First 12 bytes of MCA user
*			identifier
MQCD_MCATYPE	DS	F	Message channel agent type
MQCD_CONNECTIONNAME	DS	CL264	Connection name
MQCD_REMOTEUSERIDENTIFIER	DS	CL12	First 12 bytes of user
*			identifier from partner
MQCD_REMOTEPASSWORD	DS	CL12	Password from partner
MQCD_MSGRETRYEXIT	DS	CLn	Channel message retry exit name
MQCD_MSGRETRYUSERDATA	DS	CL32	Channel message retry exit user
*			data
MQCD_MSGRETRYCOUNT	DS	F	Number of times MCA will try to
*			put the message, after the
*			first attempt has failed
MQCD_MSGRETRYINTERVAL	DS	F	Minimum interval in
*			milliseconds after which the
*			open or put operation will be
*			retried
MQCD_HEARTBEATINTERVAL	DS	F	Time in seconds between
*			heartbeat flows
MQCD_BATCHINTERVAL	DS	F	Batch duration
MQCD_NONPERSISTENTMSGSPD	DS	F	Speed at which nonpersistent
*			messages are sent
MQCD_STRUCLLENGTH	DS	F	Length of MQCD structure
MQCD_EXITNAMELENGTH	DS	F	Length of exit name
MQCD_EXITDATALENGTH	DS	F	Length of exit user data
MQCD_MSGEXITSDDEFINED	DS	F	Number of message exits defined
MQCD_SENDEXITSDDEFINED	DS	F	Number of send exits defined
MQCD_RECEIVEEXITSDDEFINED	DS	F	Number of receive exits defined
MQCD_MSGEXITPTR	DS	F	Address of first MSGEXIT field
MQCD_MSGUSERDATAAPTR	DS	F	Address of first MSGUSERDATA
*			field
MQCD_SENDEXITPTR	DS	F	Address of first SENDEXIT field
MQCD_SENDUSERDATAAPTR	DS	F	Address of first SENDUSERDATA
*			field
MQCD_RECEIVEEXITPTR	DS	F	Address of first RECEIVEEXIT
*			field
MQCD_RECEIVEUSERDATAAPTR	DS	F	Address of first
*			RECEIVEUSERDATA field
MQCD_CLUSTERPTR	DS	F	Address of a list of cluster
*			names
MQCD_CLUSTERSDEFINED	DS	F	Number of clusters to which the
*			channel belongs
MQCD_NETWORKPRIORITY	DS	F	Network priority
MQCD_LONGMCAUSERIDLENGTH	DS	F	Length of long MCA user
*			identifier
MQCD_LONGREMOTEUSERIDLENGTH	DS	F	Length of long remote user
*			identifier

MQCD_LONGMCAUSERIDPTR	DS	F	Address of long MCA user identifier
*MQCD_LONGREMOTEUSERIDPTR	DS	F	Address of long remote user identifier
*MQCD_MCASESECURITYID	DS	XL40	MCA security identifier
MQCD_REMOTESECURITYID	DS	XL40	Remote security identifier
MQCD_SSLCIPHERSPEC	DS	CL32	TLS CipherSpec
MQCD_SSLPEERNAMEPTR	DS	F	Address of TLS peer name
MQCD_SSLPEERNAMELENGTH	DS	F	Length of TLS peer name
MQCD_SSLCLIENTAUTH	DS	F	Whether TLS client authentication is required
*MQCD_KEEPAIVEINTERVAL	DS	F	Keepalive interval
MQCD_LOCALADDRESS	DS	CL48	Local communications address
MQCD_BATCHHEARTBEAT	DS	F	Batch heartbeat interval
MQCD_HDRCOMPLIST	DS	CL2	Header data compression list
MQCD_MSGCOMPLIST	DS	CL16	Message data compression list
MQCD_CLWLCHANNELRANK	DS	F	Channel rank
MQCD_CLWLCHANNELPRIORITY	DS	F	Channel priority
MQCD_CLWLCHANNELWEIGHT	DS	F	Channel weight
MQCD_CHANNELMONITORING	DS	F	Channel monitoring
MQCD_CHANNELSTATISTICS	DS	F	Channel statistics
MQCD_SHARINGCONVERSATIONS	DS	F	Limit on sharing conversations
*MQCD_PROPERTYCONTROL	DS	F	Message property control
*MQCD_SHARINGCONVERSATIONS	DS	F	Limit on sharing conversations
MQCD_PROPERTYCONTROL	DS	F	Message property control
MQCD_MAXINSTANCES	DS	F	Limit on SVRCONN chl instances
MQCD_MAXINSTANCESPERCLIENT	DS	F	Limit on SVRCONN chl instances per client
MQCD_CLIENTCHANNELWEIGHT	DS	F	Channel weight
MQCD_CONNECTIONAFFINITY	DS	F	Connection Affinty
MQCD_BATCHDATA LIMIT	DS	F	Batch data limit
MQCD_USED LQ	DS	F	Use dead-letter queue
MQCD_DEFRECONNECT	DS	F	Default client reconnect option
MQCD_CERTLABL	DS	F	Certificate label
MQCD_SPLPROTECTION	DS	F	AMS Security policy protection
MQCD_LENGTH	EQU	*-MQCD	
	ORG	MQCD	
MQCD_AREA	DS	CL(MQCD_LENGTH)	

Declaración de Visual Basic

Esta declaración es la declaración de Visual Basic de la estructura MQCD.

En Visual Basic, la estructura MQCD se puede utilizar con la estructura MQCNO en la llamada MQCONN.

Type MQCD		
ChannelName	As String*20	'Channel definition name'
Version	As Long	'Structure version number'
ChannelType	As Long	'Channel type'
TransportType	As Long	'Transport type'
Desc	As String*64	'Channel description'
QMgrName	As String*48	'Queue manager name'
XmitQName	As String*48	'Transmission queue name'
ShortConnectionName	As String*20	'First 20 bytes of connection name'
MCAName	As String*20	'Reserved'
ModeName	As String*8	'LU 6.2 Mode name'
TpName	As String*64	'LU 6.2 transaction program name'
BatchSize	As Long	'Batch size'
DiscInterval	As Long	'Disconnect interval'
ShortRetryCount	As Long	'Short retry count'
ShortRetryInterval	As Long	'Short retry wait interval'
LongRetryCount	As Long	'Long retry count'
LongRetryInterval	As Long	'Long retry wait interval'
SecurityExit	As String*128	'Channel security exit name'
MsgExit	As String*128	'Channel message exit name'
SendExit	As String*128	'Channel send exit name'
ReceiveExit	As String*128	'Channel receive exit name'
SeqNumberWrap	As Long	'Highest allowable message sequence number'
MaxMsgLength	As Long	'Maximum message length'
PutAuthority	As Long	'Put authority'
DataConversion	As Long	'Data conversion'
SecurityUserData	As String*32	'Channel security exit user data'
MsgUserData	As String*32	'Channel message exit user data'
SendUserData	As String*32	'Channel send exit user data'
ReceiveUserData	As String*32	'Channel receive exit user data'
UserIdentifier	As String*12	'User identifier'

Password	As String*12	'Password'
MCAUserIdentifier	As String*12	'First 12 bytes of MCA user' 'identifier'
MCAType	As Long	'Message channel agent type'
ConnectionName	As String*264	'Connection name'
RemoteUserIdentifier	As String*12	'First 12 bytes of user' 'identifier from partner'
RemotePassword	As String*12	'Password from partner'
MsgRetryExit	As String*128	'Channel message retry exit name'
MsgRetryUserData	As String*32	'Channel message retry exit user' 'data'
MsgRetryCount	As Long	'Number of times MCA will try to' 'put the message, after the' 'first attempt has failed'
MsgRetryInterval	As Long	'Minimum interval in' 'milliseconds after which the' 'open or put operation will be' 'retried'
HeartbeatInterval	As Long	'Time in seconds between' 'heartbeat flows'
BatchInterval	As Long	'Batch duration'
NonPersistentMsgSpeed	As Long	'Speed at which nonpersistent' 'messages are sent'
StrucLength	As Long	'Length of MQCD structure'
ExitNameLength	As Long	'Length of exit name'
ExitDataLength	As Long	'Length of exit user data'
MsgExitsDefined	As Long	'Number of message exits defined'
SendExitsDefined	As Long	'Number of send exits defined'
ReceiveExitsDefined	As Long	'Number of receive exits defined'
MsgExitPtr	As MQPTR	'Address of first MsgExit field'
MsgUserDataPtr	As MQPTR	'Address of first MsgUserData' 'field'
SendExitPtr	As MQPTR	'Address of first SendExit field'
SendUserDataPtr	As MQPTR	'Address of first SendUserData' 'field'
ReceiveExitPtr	As MQPTR	'Address of first ReceiveExit' 'field'
ReceiveUserDataPtr	As MQPTR	'Address of first' 'ReceiveUserData field'
ClusterPtr	As MQPTR	'Address of a list of cluster' 'names'
ClustersDefined	As Long	'Number of clusters to which the' 'channel belongs'
NetworkPriority	As Long	'Network priority'
LongMCAUserIdLength	As Long	'Length of long MCA user' 'identifier'
LongRemoteUserIdLength	As Long	'Length of long remote user' 'identifier'
LongMCAUserIdPtr	As MQPTR	'Address of long MCA user' 'identifier'
LongRemoteUserIdPtr	As MQPTR	'Address of long remote user' 'identifier'
MCASecurityId	As MQBYTE40	'MCA security identifier'
RemoteSecurityId	As MQBYTE40	'Remote security identifier'
SSLCipherSpec	As String*32	'TLS CipherSpec'
SSLPeerNamePtr	As MQPTR	'Address of TLS peer name'
SSLPeerNameLength	As Long	'Length of TLS peer name'
SSLClientAuth	As Long	'Whether TLS client' 'authentication is required'
KeepAliveInterval	As Long	'Keepalive interval'
LocalAddress	As String*48	'Local communications address'
BatchHeartbeat	As Long	'Batch heartbeat interval'
HdrCompList(0 to 1)	As Long2	'Header data compression list'
MsgCompList(0 To 15)	As Long16	'Message data compression list'
CLWLChannelRank	As Long	'Channel Rank'
CLWLChannelPriority	As Long	'Channel priority'
CLWLChannelWeight	As Long	'Channel Weight'
ChannelMonitoring	As Long	'Channel Monitoring control'
ChannelStatistics	As Long	'Channel Statistics'
End Type		

Cambio de campos MQCD en una salida de canal

Una salida de canal pueden cambiar los campos del MQCD. Sin embargo, estos cambios normalmente no se realizan, excepto en las circunstancias listadas.

Si un programa de salida de canal cambia un campo en la estructura de datos MQCD, el nuevo valor normalmente lo ignora el proceso de canal de IBM MQ . Sin embargo, el nuevo valor permanece en

el MQCD y se le pasa a las salidas restantes en una cadena de salida y a cualquier conversación que comparta la instancia de canal.

Si SharingConversations se establece en FALSE en la estructura MQCXP, se pueden realizar cambios en determinados campos, en función del tipo de programa de salida, el tipo de canal y el código de razón de salida. La tabla siguiente muestra los campos que se pueden cambiar y afectan al comportamiento del canal, y en qué circunstancias. Si un programa de salida cambia uno de estos campos en cualquier otra circunstancia, o cualquier campo no listado, el nuevo valor es ignorado por el proceso de canal. El nuevo valor permanece en el MQCD y se pasa a las salidas restantes de una cadena de salida y a cualquier conversación que comparta la instancia de canal.

Cualquier tipo de programa de salida cuando se llama para la inicialización (MQXR_INIT) puede cambiar el campo ChannelName de cualquier tipo de canal, siempre que MQCXP SharingConversations esté establecido en FALSE. Sólo una salida de seguridad puede cambiar el campo MCAUserIdentifier, independientemente del valor de MQCXP SharingConversations.

<i>Tabla 859. Campos que se pueden cambiar y que afectan al comportamiento del canal</i>			
Campo	Código de razón de salida	Tipo de salida	Tipo de canal
ChannelName	MQXR_INIT	Todo	Todo
TransportType	MQXR_INIT	Todo	Todo
XmitQName	MQXR_INIT	Todo	SDR, RCVR
ModeName	MQXR_INIT	Todo	Todo
TpName	MQXR_INIT	Todo	Todo
BatchSize	MQXR_INIT	Todo	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
DiscInterval	MQXR_INIT	Todo	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
ShortRetryCount	MQXR_INIT	Todo	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
ShortRetryInterval	MQXR_INIT	Todo	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
LongRetryCount	MQXR_INIT	Todo	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR

Tabla 859. Campos que se pueden cambiar y que afectan al comportamiento del canal (continuación)

Campo	Código de razón de salida	Tipo de salida	Tipo de canal
LongRetryInterval	MQXR_INIT	Todo	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
SeqNumberWrap	MQXR_INIT	Todo	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
MaxMsgLength	MQXR_INIT	Todo	Todo
PutAuthority	MQXR_INIT	Todo	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
DataConversion	MQXR_INIT	Todo	Todo
MCAUserIdentifier	MQXR_INIT, MQXR_INIT_SEC, MQXR_SEC_MSG, MQXR_SEC_PARMS	Seguridad	RCVR, RQSTR, SVRCONN, CLUSRCVR
ConnectionName	MQXR_INIT	Todo	SDR, SVR, RQSTR, CLNTCONN, CLUSSDR, CLUSRCVR
MsgRetryUserData	MQXR_INIT	Todo	RCVR, RQSTR, CLUSRCVR
MsgRetryCount	MQXR_INIT	Todo	RCVR, RQSTR, CLUSRCVR
MsgRetryInterval	MQXR_INIT	Todo	RCVR, RQSTR, CLUSRCVR
HeartbeatInterval	MQXR_INIT	Todo	Todo
BatchInterval	MQXR_INIT	Todo	SDR, SVR, CLUSSDR, CLUSRCVR
NonPersistentMsgSpeed	MQXR_INIT	Todo	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR

Tabla 859. Campos que se pueden cambiar y que afectan al comportamiento del canal (continuación)

Campo	Código de razón de salida	Tipo de salida	Tipo de canal
MCASecurityId	MQXR_INIT, MQXR_INIT_SEC, MQXR_SEC_MSG, MQXR_SEC_PARMS	Seguridad	SDR, SVR, RCVR, RQSTR, SVRCONN, CLUSSDR, CLUSRCVR
SSLCipherSpec	MQXR_INIT	Todo	Todo
SSLPeerNamePtr	MQXR_INIT	Todo	Todo
SSLPeerNameLength	MQXR_INIT	Todo	Todo
SSLClientAuth	MQXR_INIT	Todo	SVR, RCVR, RQSTR, SVRCONN, CLUSRCVR
KeepAliveInterval	MQXR_INIT	Todo	Todo
LocalAddress	MQXR_INIT	Todo	SDR, SVR, RQSTR, CLNTCONN, CLUSSDR, CLUSRCVR
BatchHeartbeat	MQXR_INIT	Todo	SDR, SVR, CLUSSDR, CLUSRCVR
HdrCompList	MQXR_INIT	Todo	Todo
MsgCompList	MQXR_INIT	Todo	Todo
ChannelMonitoring	MQXR_INIT	Todo	SDR, SVR, RCVR, RQSTR, SVRCONN, CLUSSDR, CLUSRCVR
ChannelStatistics	MQXR_INIT	Todo	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
SharingConversations	MQXR_INIT	Todo	SVRCONN, CLNTCONN
PropertyControl	MQXR_INIT	Todo	SDR, SVR, CLUSSDR, CLUSRCVR

MQCXP-Parámetro de salida de canal

La estructura MQCXP se pasa a cada tipo de salida llamada por un agente de canal de mensajes (MCA), un canal de conexión de cliente o un canal de conexión de servidor.

Consulte MQ_CHANNEL_EXIT.

Los campos descritos como "entrada a la salida" en las descripciones que siguen son ignorados por el canal cuando la salida devuelve el control al canal. Los campos de entrada que la salida cambie en el bloque de parámetros de salida de canal no se conservarán para su siguiente invocación. Los cambios realizados en los campos de entrada/salida (por ejemplo, el campo *ExitUserArea*) se conservan sólo para las invocaciones de esa instancia de la salida. Estos cambios no se pueden utilizar para pasar datos entre salidas diferentes definidas en el mismo canal, o entre la misma salida definida en canales diferentes.

Referencia relacionada

[“Campos” en la página 1545](#)

Este tema lista todos los campos de la estructura MQCXP y describe cada campo.

[“Declaración C” en la página 1556](#)

Esta declaración es la declaración C para la estructura MQCXP.

[“COBOL declaración” en la página 1557](#)

Esta declaración es la declaración COBOL para la estructura MQCXP.

[“Declaración RPG \(ILE\)” en la página 1558](#)

Esta declaración es la declaración RPG para la estructura MQCXP.

[“Declaración de ensamblador System/390” en la página 1559](#)

Esta declaración es la declaración de ensamblador System/390 para la estructura MQCXP.

Campos

Este tema lista todos los campos de la estructura MQCXP y describe cada campo.

StrucId (MQCHAR4)

Este campo especifica el identificador de estructura.

El valor debe ser:

MQCXP_STRUC_ID

Identificador de la estructura de parámetros de salida de canal.

Para el lenguaje de programación C, también se define la constante MQCXP_STRUC_ID_ARRAY; esta constante tiene el mismo valor que MQCXP_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Es un campo de entrada para la salida.

Versión (MQLONG)

Este campo especifica el número de versión de la estructura.

El valor depende del entorno:

MQCXP_VERSION_1

Version-1 Estructura del parámetro de salida del canal.

MQCXP_VERSION_2

Version-2 Estructura del parámetro de salida del canal.

MQCXP_VERSION_3

Version-3 Estructura del parámetro de salida del canal.



El campo tiene este valor en AIX and Linux sistemas no enumerados en otra parte.

MQCXP_VERSION_4

Version-4 Estructura del parámetro de salida del canal.

MQCXP_VERSION_5

Version-5 Estructura del parámetro de salida del canal.

MQCXP_VERSION_6

Version-6 Estructura del parámetro de salida del canal.

MQCXP_VERSION_8

Version-8 Estructura del parámetro de salida del canal.

 El campo tiene este valor en z/OS.

MQCXP_VERSION_9

Version-9 Estructura del parámetro de salida del canal.

El campo tiene este valor en los siguientes entornos:

-  AIX
-  IBM i
-  Linux
-  Windows
-  z/OS

Los campos que solo existen en las versiones más recientes de la estructura se identifican como tales en las descripciones de los campos. La siguiente constante especifica el número de versión de la versión actual:

MQCXP_VERSIÓN_ACTUAL

Versión actual de la estructura del parámetro de salida del canal.

El valor depende del entorno.

Nota: Cuando se introduce una nueva versión de la estructura MQCXP, el diseño de la pieza existente no se modifica. Por lo tanto, la salida debe comprobar que el número de versión sea igual o superior a la versión más baja que contenga los campos que la salida necesita utilizar.

Este es un campo de entrada a la salida.

ExitId (MQLONG)

Este campo especifica el tipo de salida que se llama y se establece en la entrada de la rutina de salida.

Son posibles los siguientes valores:

MQXT_CHANNEL_SEC_EXIT

Salida de seguridad de canal.

MQXT_CHANNEL_MSG_EXIT

Salida de mensajes de canal.

MQXT_CHANNEL_SEND_EXIT

Salida de envío de canal.

MQXT_CHANNEL_RCV_EXIT

Salida de recepción de canal.

MQXT_CHANNEL_MSG_RETRY_EXIT

Salida de reintento de mensaje de canal.

MQXT_CHANNEL_AUTO_DEF_EXIT

Salida de definición automática de canal.

En z/OS, este tipo de salida sólo está soportado para canales de tipo MQCHT_CLUSSDR y MQCHT_CLUSRCVR.

Es un campo de entrada para la salida.

ExitReason (MQLONG)

Este campo especifica la razón por la que se llama a la salida y se establece en la entrada de la rutina de salida.

La salida de definición automática no la utiliza. Son posibles los siguientes valores:

MQXR_INIT

Salga de la inicialización.

Este valor indica que la salida se está invocando por primera vez. Permite que la salida adquiera e inicialice los recursos que necesite (por ejemplo: memoria).

MQXR_TERM

Terminación de salida.

Este valor indica que la salida está a punto de terminar. La salida debe liberar los recursos que haya adquirido desde que se inicializó (por ejemplo: memoria).

MQXR_MSG

Procesar un mensaje.

Este valor indica que se está invocando la salida para procesar un mensaje. Este valor sólo se produce para salidas de mensajes de canal.

MQXR_XMIT

Procesar una transmisión.

Este valor sólo se produce para salidas de envío y recepción de canal.

MQXR_SEC_MSG

Se ha recibido el mensaje de seguridad.

Este valor sólo se produce para salidas de seguridad de canal.

MQXR_INIT_SEC

Inicie el intercambio de seguridad.

Este valor sólo se produce para salidas de seguridad de canal.

La salida de seguridad del receptor siempre se invoca con esta razón inmediatamente después de ser invocada con MQXR_INIT, para darle la oportunidad de iniciar un intercambio de seguridad. Si rechaza la oportunidad (devolviendo MQXCC_OK en lugar de MQXCC_SEND_SEC_MSG o MQXCC_SEND_AND_REQUEST_SEC_MSG), la salida de seguridad del remitente se invoca con MQXR_INIT_SEC.

Si la salida de seguridad del receptor no inicia un intercambio de seguridad (devolviendo MQXCC_SEND_SEC_MSG o MQXCC_SEND_AND_REQUEST_SEC_MSG), la salida de seguridad del remitente nunca se invoca con MQXR_INIT_SEC; en su lugar, se invoca con MQXR_SEC_MSG para procesar el mensaje del receptor. (En cualquier caso, se invoca por primera vez con MQXR_INIT.)

A menos que una de las salidas de seguridad solicite la terminación del canal (estableciendo *ExitResponse* en MQXCC_SUPPRESS_FUNCTION o MQXCC_CLOSE_CHANNEL), el intercambio de seguridad debe completarse en el lado que ha iniciado el intercambio. Por lo tanto, si se invoca una salida de seguridad con MQXR_INIT_SEC y inicia un intercambio, la próxima vez que se invoque la salida será con MQXR_SEC_MSG. Esto sucede si hay un mensaje de seguridad para que la salida se procese o no. Hay un mensaje de seguridad si el socio devuelve MQXCC_SEND_SEC_MSG o MQXCC_SEND_AND_REQUEST_SEC_MSG, pero no si el socio devuelve MQXCC_OK o no hay ninguna salida de seguridad en el socio. Si no hay ningún mensaje de seguridad para procesar, la salida de seguridad en el extremo de inicio se vuelve a invocar con un *DataLength* de cero.

MQXR_REINTENTAR

Vuelva a intentar un mensaje.

Este valor sólo se produce para salidas de reintento de mensaje.

MQXR_AUTO_CLUSSDR

Definición automática de un canal de clúster emisor.

Este valor sólo se produce para salidas de definición automática de canal.

MQXR_AUTO_RECEIVER

Definición automática de un canal receptor.

Este valor sólo se produce para salidas de definición automática de canal.

MQXR_AUTO_SVRCONN

Definición automática de un canal de conexión de servidor.

Este valor sólo se produce para salidas de definición automática de canal.

MQXR_AUTO_CLUSRCVR

Definición automática de un canal de clúster receptor.

Este valor sólo se produce para salidas de definición automática de canal.

MQXR_SEC_PARMS

Parámetros de seguridad

Este valor sólo se aplica a las salidas de seguridad e indica que se está pasando una estructura MQCSP a la salida. Para obtener más información, consulte [“MQCSP - Parámetros de seguridad” en la página 291](#)

Nota:

1. Si tiene más de una salida definida para un canal, cada uno de ellos se invoca con MQXR_INIT cuando se inicializa el MCA. Además, cada uno de ellos se invoca con MQXR_TERM cuando termina el MCA.
2. Para la salida de definición automática de canal, *ExitReason* no se establece si *Version* es menor que MQCXP_VERSION_4. El valor MQXR_AUTO_SVRCONN está implícito en este caso.

Es un campo de entrada para la salida.

ExitResponse (MQLONG)

Este campo especifica la respuesta de la salida.

Este campo lo establece la salida para comunicarse con el MCA. Tiene que ser uno de los valores siguientes:

MQXCC_Correcto

La salida se ha completado correctamente.

- Para la salida de seguridad de canal, este valor indica que la transferencia de mensajes ahora puede continuar con normalidad.
- Para la salida de reintento de mensaje de canal, este valor indica que el MCA debe esperar el intervalo de tiempo devuelto por la salida en el campo *MsgRetryInterval* en MQCXP y, a continuación, vuelva a intentar el mensaje.

El campo *ExitResponse2* puede contener información adicional.

MQXCC_SUPPRESS_FUNCTION

Suprimir función.

- Para la salida de seguridad de canal, este valor indica que el canal debe terminar.
- Para la salida de mensajes de canal, este valor indica que el mensaje no debe continuar hacia su destino. En su lugar, el MCA genera un mensaje de informe de excepción (si lo ha solicitado el remitente del mensaje original) y coloca el mensaje contenido en el almacenamiento intermedio original en la cola de mensajes no entregados (si el remitente ha especificado MQRO_DEAD_LETTER_Q), o lo descarta (si el remitente ha especificado MQRO_DISCARD_MSG).

Para los mensajes persistentes, si el remitente ha especificado MQRO_DEAD_LETTER_Q, pero la colocación en la cola de mensajes no entregados falla, o no hay ninguna cola de mensajes no entregados, el mensaje original se deja en la cola de transmisión y el mensaje de informe no se genera. El mensaje original también se deja en la cola de transmisión si el mensaje de informe no se puede generar correctamente.

El campo *Feedback* de la estructura MQDLH al principio del mensaje en la cola de mensajes no entregados indica por qué el mensaje se ha colocado en la cola de mensajes no entregados; este código de retorno también se utiliza en el descriptor de mensaje del mensaje de informe de excepción (si el remitente ha solicitado uno).

- Para la salida de reintento de mensaje de canal, este valor indica que el MCA no espera y vuelve a intentar el mensaje; en su lugar, el MCA continúa inmediatamente con su proceso de anomalía normal (el mensaje se coloca en la cola de mensajes no entregados o se descarta, tal como especifica el emisor del mensaje).
- Para la salida de definición automática de canal, debe especificarse MQXCC_OK o MQXCC_SUPPRESS_FUNCTION. Si no se especifica ninguno de estos valores, se presupone MQXCC_SUPPRESS_FUNCTION de forma predeterminada y se abandona la definición automática.

Esta respuesta no está soportada para las salidas de envío y recepción de canal.

MQXCC_SEND_SEC_MSG

Enviar mensaje de seguridad.

Este valor sólo lo puede establecer una salida de seguridad de canal. Indica que la salida ha proporcionado un mensaje de seguridad que debe transmitirse al socio.

MQXCC_ENVIAR_Y_SOLICITAR_MENSAJE_DE_SEGURIDAD

Envíe un mensaje de seguridad que requiera una respuesta.

Este valor sólo lo puede establecer una salida de seguridad de canal. Indica

- que la salida ha proporcionado un mensaje de seguridad que puede transmitirse al socio, y
- que la salida requiere una respuesta del socio. Si no se recibe ninguna respuesta, el canal debe terminar, porque la salida todavía no ha decidido si las comunicaciones pueden continuar.

MQXCC_SUPPRESS_EXIT

Suprimir salida.

- Este valor lo pueden establecer todos los tipos de salida de canal que no sean una salida de seguridad o una salida de definición automática. Suprime cualquier invocación adicional de esa salida (como si su nombre hubiera estado en blanco en la definición de canal), hasta la terminación del canal, cuando se vuelve a invocar la salida con un *ExitReason* de MQXR_TERM.
- Si una salida de reintento de mensaje devuelve este valor, los reintentos de mensaje para los mensajes posteriores se controlan mediante los atributos de canal *MsgRetryCount* y *MsgRetryInterval* como normales. Para el mensaje actual, el MCA realiza el número de reintentos pendientes, a intervalos proporcionados por el atributo de canal *MsgRetryInterval*, pero sólo si el código de razón es uno que el MCA reintentaría normalmente (consulte el campo *MsgRetryCount* descrito en “MQCD-Definición de canal” en la página 1502). El número de reintentos pendientes es el valor del atributo **MsgRetryCount**, menos el número de veces que la salida ha devuelto MQXCC_OK para el mensaje actual; si este número es negativo, el MCA no realiza más reintentos para el mensaje actual.

MQXCC_CERRAR_CANAL

Cierre el canal.

Este valor lo puede establecer cualquier tipo de salida de canal excepto una salida de definición automática.

Si la compartición de conversaciones no está habilitada, este valor cierra el canal.

Si la compartición de conversaciones está habilitada, este valor finaliza la conversación. Si esta conversación es la única conversación en el canal, el canal también se cierra.

Este campo es un campo de entrada/salida de la salida.

ExitResponse2 (MQLONG)

Este campo especifica la respuesta secundaria de la salida.

Este campo se establece en cero en la entrada a la rutina de salida. Se puede establecer mediante la salida para proporcionar más información a las funciones de canal de IBM MQ . La salida de definición automática no la utiliza.

La salida puede establecer uno o varios de los valores siguientes. Si se necesita más de uno, se añaden los valores. Las combinaciones que no son válidas se anotan; se permiten otras combinaciones.

MQXR2_PUT_WITH_DEF_ACTION

Poner con acción predeterminada.

Este valor lo establece la salida de mensajes de canal del receptor. Indica que el mensaje debe colocarse con la acción predeterminada del MCA, que es el ID de usuario predeterminado del MCA, o el contexto *UserIdentifier* en el MQMD (descriptor de mensaje) del mensaje.

El valor es cero, que corresponde al valor inicial establecido cuando se invoca la salida. La constante se proporciona a efectos de documentación.

MQXR2_PUT_WITH_DEF_USERID

Colocar con identificador de usuario predeterminado.

Este valor sólo lo puede establecer la salida de mensajes de canal del receptor. Indica que el mensaje debe colocarse con el identificador de usuario predeterminado del MCA.

MQXR2_PUT_WITH_MSG_USERID

Transferir con el identificador de usuario del mensaje.

Este valor sólo lo puede establecer la salida de mensajes de canal del receptor. Indica que el mensaje debe colocarse con el contexto *UserIdentifier* en el MQMD (descriptor de mensaje) del mensaje (esto puede haber sido modificado por la salida).

Solo debe establecerse uno de MQXR2_PUT_WITH_DEF_ACTION, MQXR2_PUT_WITH_DEF_USERID y MQXR2_PUT_WITH_MSG_USERID .

MQXR2_USE_AGENT_BUFFER

Utilice el almacenamiento intermedio del agente.

Este valor indica que los datos que deben pasarse están en *AgentBuffer*, no en *ExitBufferAddr*.

El valor es cero, que corresponde al valor inicial establecido cuando se invoca la salida. La constante se proporciona a efectos de documentación.

MQXR2_USE_EXIT_BUFFER

Utilice el almacenamiento intermedio de salida.

Este valor indica que los datos que deben pasarse están en *ExitBufferAddr*, no en *AgentBuffer*.

Solo se debe establecer uno de MQXR2_USE_AGENT_BUFFER y MQXR2_USE_EXIT_BUFFER .

MQXR2_DEFAULT_CONTINUATION

Continuación predeterminada.

La continuación con la siguiente salida de la cadena depende de la respuesta de la última salida invocada:

- Si se devuelve MQXCC_SUPPRESS_FUNCTION o MQXCC_CLOSE_CHANNEL, no se llama a más salidas de la cadena.
- De lo contrario, se invoca la siguiente salida de la cadena.

MQXR2_CONTINUE_CHAIN

Continúe con la siguiente salida.

MQXR2_SUPPRESS_CHAIN

Omita las salidas restantes de la cadena.

Es un campo de entrada/salida para la salida.

Feedback (MQLONG)

Este campo especifica el código de comentarios.

Este campo se establece en MQFB_NONE en la entrada a la rutina de salida.

Si una salida de mensajes de canal establece el campo *ExitResponse* en MQXCC_SUPPRESS_FUNCTION, el campo *Feedback* especifica el código de retorno que identifica por qué el mensaje se ha colocado en la cola de mensajes no entregados (undelivered-message) y también se utiliza para enviar un informe de excepción si se ha solicitado uno. En este caso, si el campo *Feedback* es MQFB_NONE, se utiliza el siguiente código de retorno:

MQFB_DETENIDO_POR_MSG_EXIT

Mensaje detenido por salida de mensajes de canal.

El MCA no utiliza el valor devuelto en este campo por las salidas de seguridad de canal, envío, recepción y reintento de mensaje.

El valor devuelto en este campo por las salidas de definición automática no se utiliza si *ExitResponse* es MQXCC_OK, pero de lo contrario se utiliza para el parámetro *AuxErrorDataInt1* en el mensaje de suceso.

Es un campo de entrada/salida de la salida.

MaxSegmentLength (MQLONG)

Este campo especifica la longitud máxima en bytes que se puede enviar en una sola transmisión.

La salida de definición automática no la utiliza. Es de interés para una salida de emisión de canal, porque esta salida debe asegurarse de que no aumenta el tamaño de un segmento de transmisión a un valor mayor que *MaxSegmentLength*. La longitud incluye los 8 bytes iniciales que la salida no debe cambiar. El valor se negocia entre las funciones de canal de IBM MQ cuando se inicia el canal. Consulte [Escritura de programas de salida de canal](#) para obtener más información sobre las longitudes de segmento.

El valor de este campo no es significativo si *ExitReason* es MQXR_INIT.

Es un campo de entrada para la salida.

ExitUserArea (MQBYTE16)

Este campo especifica el área de usuario de salida: un campo disponible para que lo utilice la salida.

Se inicializa a cero binario antes de la primera invocación de la salida (que tiene un *ExitReason* establecido en MQXR_INIT) y, a partir de entonces, los cambios realizados en este campo por la salida se conservan entre las invocaciones de la salida.

Se define el valor siguiente:

MQXUA_NONE

No hay información de usuario.

El valor es cero binario para la longitud del campo.

Para el lenguaje de programación C, también se define la constante MQXUA_NONE_ARRAY; esta constante tiene el mismo valor que MQXUA_NONE, pero es una matriz de caracteres en lugar de una serie.

La longitud de este campo la proporciona MQ_EXIT_USER_AREA_LENGTH. Es un campo de entrada/salida para la salida.

ExitData (MQCHAR32)

Este campo especifica los datos de salida.

Este campo se establece en la entrada de la rutina de salida en la información que las funciones de canal de IBM MQ han tomado de la definición de canal. Si no hay dicha información disponible, este campo está en blanco.

La longitud de este campo la proporciona MQ_EXIT_DATA_LENGTH.

Es un campo de entrada para la salida.

Los campos siguientes de esta estructura no están presentes si *Version* es menor que MQCXP_VERSION_2.

MsgRetryCount (MQLONG)

Este campo especifica el número de veces que se ha reintentado el mensaje.

La primera vez que se invoca la salida para un mensaje determinado, este campo tiene el valor cero (todavía no se han intentado reintentos). En cada invocación posterior de la salida para ese mensaje, el valor se incrementa en uno por el MCA.

Es un campo de entrada para la salida. El valor de este campo no es significativo si *ExitReason* es MQXR_INIT. El campo no está presente si *Version* es menor que MQCXP_VERSION_2.

MsgRetryInterval (MQLONG)

Este campo especifica el intervalo mínimo en milisegundos después del cual se reintentará la operación de transferencia.

La primera vez que se invoca la salida para un mensaje determinado, este campo contiene el valor del atributo de canal *MsgRetryInterval*. La salida puede dejar el valor sin modificar o modificarlo para especificar un intervalo de tiempo diferente en milisegundos. Si la salida devuelve MQXCC_OK en *ExitResponse*, el MCA espera al menos este intervalo de tiempo antes de reintentar la operación MQOPEN o MQPUT. El intervalo de tiempo especificado debe ser cero o mayor.

La segunda y siguientes veces que se invoca la salida para ese mensaje, este campo contiene el valor devuelto por la invocación anterior de la salida.

Si el valor devuelto en el campo *MsgRetryInterval* es menor que cero o mayor que 999 999 999 999, y *ExitResponse* es MQXCC_OK, el MCA ignora el campo *MsgRetryInterval* en MQCXP y espera en su lugar el intervalo especificado por el atributo de canal *MsgRetryInterval*.

Es un campo de entrada/salida para la salida. El valor de este campo no es significativo si *ExitReason* es MQXR_INIT. El campo no está presente si *Version* es menor que MQCXP_VERSION_2.

MsgRetryReason (MQLONG)

Este campo especifica el código de razón del intento anterior de colocar el mensaje.

Este campo es el código de razón del intento anterior de colocar el mensaje; es uno de los valores MQRC_*

Es un campo de entrada para la salida. El valor de este campo no es significativo si *ExitReason* es MQXR_INIT. El campo no está presente si *Version* es menor que MQCXP_VERSION_2.

Los campos siguientes de esta estructura no están presentes si *Version* es menor que MQCXP_VERSION_3.

HeaderLength (MQLONG)

Este campo especifica la longitud de la información de cabecera.

Este campo sólo es relevante para una salida de mensajes y una salida de reintento de mensajes. El valor es la longitud de las estructuras de cabecera de direccionamiento al principio de los datos del mensaje; son la estructura MQXQH, la MQMDE (cabecera de extensión de descripción de mensaje) y (para un mensaje de lista de distribución) la estructura MQDH y las matrices de registros MQOR y MQPMR que siguen a la estructura MQXQH.

La salida de mensajes puede examinar esta información de cabecera y modificarla si es necesario, pero los datos que devuelve la salida deben estar en el formato correcto. La salida no debe, por ejemplo, cifrar o comprimir los datos de cabecera en el extremo emisor, incluso si la salida de mensaje en el extremo receptor realiza cambios compensatorios.

Si la salida de mensajes modifica la información de cabecera de forma que cambie su longitud (por ejemplo, añadiendo otro destino a un mensaje de lista de distribución), debe cambiar el valor de *HeaderLength* correspondiente antes de devolver.

Es un campo de entrada/salida para la salida. El valor de este campo no es significativo si *ExitReason* es MQXR_INIT. El campo no está presente si *Version* es menor que MQCXP_VERSION_3.

PartnerName (MQCHAR48)

Este campo especifica el nombre del socio.

El nombre del socio, como se indica a continuación:

- Para canales SVRCONN, es el ID de usuario conectado en el cliente.
- Para todos los demás tipos de canal, es el nombre del gestor de colas del socio.

Cuando se inicializa la salida, este campo está en blanco porque el gestor de colas no conoce el nombre del socio hasta después de que se haya realizado la negociación inicial.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCXP_VERSION_3.

FAPLevel (MQLONG)

Nivel de protocolos y formatos negociados.

Es un campo de entrada para la salida. Los cambios en este campo sólo se deben realizar bajo la dirección del servicio de IBM . El campo no está presente si *Version* es menor que MQCXP_VERSION_3.

CapabilityFlags (MQLONG)

Puede establecer el distintivo de prestación en MQCF_NONE o MQCF_DIST_LISTS.

Puede establecer cualquiera de los siguientes distintivos de prestación:

MQCF_NINGUNO

Sin distintivos.

MQCF_DIST_LISTS

Listas de distribución soportadas.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCXP_VERSION_3.

ExitNumber (MQLONG)

Este campo especifica el número ordinal de la salida.

El número ordinal de la salida, dentro del tipo definido en *ExitId*. Por ejemplo, si la salida que se invoca es la tercera salida de mensajes definida, este campo contiene el valor 3. Si el tipo de salida es uno para el que no se puede definir una lista de salidas (por ejemplo, una salida de seguridad), este campo tiene el valor 1.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCXP_VERSION_3.

Los campos siguientes de esta estructura no están presentes si *Version* es menor que MQCXP_VERSION_5.

ExitSpace (MQLONG)

Este campo especifica el número de bytes en el almacenamiento intermedio de transmisión reservado para que lo utilice la salida.

Este campo sólo es relevante para una salida de envío. Especifica la cantidad de espacio en bytes que las funciones de canal de IBM MQ reservan en el almacenamiento intermedio de transmisión para que la utilice la salida. Este campo permite que la salida añada al almacenamiento intermedio de transmisión una pequeña cantidad de datos (normalmente no más de unos pocos cientos de bytes) para que los utilice una salida de recepción complementaria en el otro extremo. La salida de recepción debe eliminar los datos añadidos por la salida de envío.

El valor siempre es cero en z/OS.

Nota: Este recurso no se debe utilizar para enviar grandes cantidades de datos, ya que puede degradar el rendimiento o incluso inhibir el funcionamiento del canal.

Al establecer *ExitSpace* la salida se garantiza que siempre hay al menos ese número de bytes disponibles en el almacenamiento intermedio de transmisión para que los utilice la salida. Sin embargo, la salida puede utilizar menos de la cantidad reservada, o más de la cantidad reservada si hay espacio disponible en el almacenamiento intermedio de transmisión. El espacio de salida en el almacenamiento intermedio se proporciona siguiendo los datos existentes.

La salida sólo puede establecer *ExitSpace* cuando *ExitReason* tiene el valor MQXR_INIT; en todos los demás casos, el valor devuelto por la salida se ignora. En la entrada a la salida, *ExitSpace* es cero para la llamada MQXR_INIT y es el valor devuelto por la llamada MQXR_INIT en otros casos.

Si el valor devuelto por la llamada MQXR_INIT es negativo, o si hay menos de 1024 bytes disponibles en el almacenamiento intermedio de transmisión para los datos de mensaje después de reservar el espacio de salida solicitado para todas las salidas de envío de la cadena, el MCA genera un mensaje de error y cierra el canal. De forma similar, si durante la transferencia de datos las salidas de la cadena de salida de envío asignan más espacio de usuario del que reservaron, de manera que quedan menos de 1024 bytes en el almacenamiento intermedio de transmisión para los datos de mensaje, el MCA genera un mensaje de error y cierra el canal. El límite de 1024 permite que los flujos de control y administrativos del canal sean procesados por la cadena de salidas de envío, sin necesidad de que los flujos sean segmentados.

Es un campo de entrada/salida para la salida si *ExitReason* es MQXR_INIT, y un campo de entrada en todos los demás casos. El campo no está presente si *Version* es menor que MQCXP_VERSION_5.

SSLCertUserId (MQCHAR12)

Este campo especifica el UserId asociado con el certificado remoto.

Está en blanco en todas las plataformas excepto z/OS

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCXP_VERSION_6.

SSLRemCertIssNameLength (MQLONG)

Este campo especifica la longitud en bytes del Nombre Distinguido completo del emisor del certificado remoto apuntado por SSLCertRemoteIssuerNamePtr.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCXP_VERSION_6. El valor es cero si no se trata de un canal de « TLS ».

SSLRemCertIssNamePtr (PMQVOID)

Este campo especifica la dirección del nombre distinguido completo del emisor del certificado remoto.

Su valor es el puntero nulo si no se trata de un canal de tipo « TLS ».

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCXP_VERSION_6.

Nota: El comportamiento de las salidas de seguridad de canal al determinar el nombre distinguido del sujeto y el nombre distinguido del emisor se cambia de IBM WebSphere MQ 7.1. Para obtener más información, consulte [Programas de salida de seguridad de canal](#).

SecurityParms (PMQCSP)

Este campo especifica la dirección de la estructura MQCSP que se utiliza para especificar las credenciales de autenticación.

El valor inicial de este campo es el puntero nulo.

Este es un campo de entrada/salida hacia la salida. El campo no está presente si *Version* es menor que MQCXP_VERSION_6.

El valor que devuelve la salida en este campo debe ser utilizable por IBM MQ hasta MQXR_TERM.

CurHdrCompression (MQLONG)

Este campo especifica qué técnica se está utilizando actualmente para comprimir los datos de cabecera.

Se establece en uno de los valores siguientes:

MQCOMPRESS_NONE

No se lleva a cabo ninguna compresión de datos de cabecera.

MQCOMPRESS_SISTEMA

Se lleva a cabo la compresión de datos de cabecera.

El valor puede ser alterado por la salida de mensajes de un canal emisor a uno de los valores negociados soportados a los que se accede desde el campo HdrCompList del MQCD. Esto permite que la técnica utilizada para comprimir los datos de cabecera se elija para cada mensaje basándose en el contenido del mensaje. El valor alterado sólo se utiliza para el mensaje actual. El canal finaliza si el atributo se modifica a un valor no soportado. El valor se ignora si se altera fuera de la salida de mensajes de un canal emisor.

Es un campo de entrada/salida para la salida. El campo no está presente si *Version* es menor que MQCXP_VERSION_6.

CurMsgCompression (MQLONG)

Este campo especifica qué técnica se está utilizando actualmente para comprimir los datos del mensaje.

Se establece en uno de los valores siguientes:

MQCOMPRESS_NONE

No se lleva a cabo ninguna compresión de datos de cabecera.

MQCOMPRESS_RLE

Se lleva a cabo la compresión de datos de mensaje utilizando la codificación de longitud de ejecución.

MQCOMPRESS_ZLIBFAST

Se lleva a cabo la compresión de datos de mensaje utilizando el método de compresión zlib. Se prefiere un tiempo de compresión rápido.

MQCOMPRESS_ZLIBHIGH

Se lleva a cabo la compresión de datos de mensaje utilizando el método de compresión zlib. Se prefiere un nivel elevado de compresión.

El valor puede ser alterado por la salida de mensajes de un canal emisor a uno de los valores negociados soportados a los que se accede desde el campo MsgCompList del MQCD. Esto permite que la técnica utilizada para comprimir los datos del mensaje se decida para cada mensaje basándose en el contenido del mensaje. El valor alterado sólo se utiliza para el mensaje actual. El canal finaliza si el atributo se modifica a un valor no soportado. El valor se ignora si se altera fuera de la salida de mensajes de un canal emisor.

Es un campo de entrada/salida para la salida. El campo no está presente si *Version* es menor que MQCXP_VERSION_6.

Hconn (MQHCONN)

Este campo especifica el identificador de conexión que utiliza la salida en caso de que necesite realizar llamadas a `MQI` dentro de la propia salida.

Este campo no es relevante para las salidas que se ejecutan en canales de conexión cliente, donde contiene el valor MQHC_UNUSABLE_HCONN (-1).

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCXP_VERSION_7.

SharingConversations (MQBOOL)

Este campo especifica si la conversación es la única que se puede ejecutar actualmente en esta instancia de canal, o si actualmente se puede ejecutar más de una conversación en esta instancia de canal.

También indica si el programa de salida está sujeto al riesgo de que otro programa de salida que se ejecuta al mismo tiempo altere el MQCD.

Este campo sólo es relevante para los programas de salida que se ejecutan en canales de conexión de cliente o de conexión de servidor.

Se establece en uno de los valores siguientes:

FALSE

La instancia de salida es la única instancia de salida que se puede ejecutar actualmente en esta instancia de canal. Esto permite a la salida actualizar de forma segura los campos MQCD sin contención de otras salidas que se ejecutan en otras instancias de canal. Si el canal actúa sobre los cambios en los campos MQCD se define mediante la tabla de campos MQCD en [“Cambio de campos MQCD en una salida de canal” en la página 1541](#).

TRUE

La instancia de salida no es la única instancia de salida que se puede ejecutar actualmente en esta instancia de canal. El canal no actúa sobre los cambios realizados en MQCD, excepto los cambios listados en la tabla de campos MQCD en [“Cambio de campos MQCD en una salida de canal”](#) en la página 1541 por razones de salida que no sean MQXR_INIT. Si esta salida actualiza los campos MQCD, asegúrese de que no haya ninguna contención de otras salidas en ejecución en otras conversaciones al mismo tiempo proporcionando serialización entre las salidas que se ejecutan en esta instancia de canal.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCXP_VERSION_7.

MCAUserSource (MQLONG)

Este campo especifica el origen del ID de usuario de MCA proporcionado.

Puede contener uno de los valores siguientes:

MQUSRC_MAP

El ID de usuario se especifica en el atributo MCAUSER.

MQUSRC_CHANNEL

El ID de usuario fluye desde el socio de entrada o se especifica en el campo MCAUSER definido en el objeto de canal.

Es un campo de entrada para la salida. El campo no está presente si la versión es menor que MQCXP_VERSION_8.

pEntryPoints (PMQIEP)

Este campo especifica la dirección del punto de entrada de la interfaz para la llamada de interfaz de control de datos (MQI) o DCI.

El campo no está presente si *Versión* es menor que MQCXP_VERSION_8.

RemoteProduct (MQCHAR4)

Este campo especifica el nombre del producto remoto.

Este campo identifica el producto remoto del cliente, por ejemplo, C o Java, tal como se muestra en el campo **RPRODUCT** de [DISPLAY CHSATUS](#).

El campo no está presente si *Versión* es menor que MQCXP_VERSION_9.

RemoteVersion (MQCHAR8)

Este campo especifica el nombre de la versión remota.

Este campo identifica la versión de las bibliotecas de cliente, tal como se muestra en el campo **RVERSION** de [DISPLAY CHSTATUS](#).

El campo no está presente si *Versión* es menor que MQCXP_VERSION_9.

Declaración C

Esta declaración es la declaración C para la estructura MQCXP.

```
typedef struct tagMQCXP MQCXP;
struct tagMQCXP {
    MQCHAR4    StructId;           /* Structure identifier */
    MQLONG     Version;           /* Structure version number */
    MQLONG     ExitId;            /* Type of exit */
    MQLONG     ExitReason;        /* Reason for invoking exit */
    MQLONG     ExitResponse;      /* Response from exit */
    MQLONG     ExitResponse2;     /* Secondary response from exit */
    MQLONG     Feedback;          /* Feedback code */
    MQLONG     MaxSegmentLength;  /* Maximum segment length */
    MQBYTE16   ExitUserArea;      /* Exit user area */
    MQCHAR32   ExitData;          /* Exit data */
    MQLONG     MsgRetryCount;     /* Number of times the message has been
                                retried */
    MQLONG     MsgRetryInterval; /* Minimum interval in milliseconds after
                                which the put operation should be
```



```

    MQLONG      MsgRetryReason;    /* Reason code from previous attempt to
                                   put the message */
    MQLONG      HeaderLength;      /* Length of header information */
    MQCHAR48    PartnerName;       /* Partner Name */
    MQLONG      FAPLevel;          /* Negotiated Formats and Protocols
                                   level */
    MQLONG      CapabilityFlags;    /* Capability flags */
    MQLONG      ExitNumber;        /* Exit number */
    /* Ver:3 */
    /* Ver:4 */
    MQLONG      ExitSpace;         /* Number of bytes in transmission buffer
                                   reserved for exit to use */
    /* Ver:5 */
    MQCHAR12    SSLCertUserid;     /* User identifier associated
                                   with remote TLS certificate */
    MQLONG      SSLRemCertIssNameLength; /* Length of
                                   distinguished name of issuer
                                   of remote TLS certificate */
    MQPTR       SSLRemCertIssNamePtr; /* Address of
                                   distinguished name of issuer
                                   of remote TLS certificate */
    PMQVOID     SecurityParms;     /* Security parameters */
    MQLONG      CurHdrCompression; /* Header data compression
                                   used for current message */
    MQLONG      CurMsgCompression; /* Message data compression
                                   used for current message */
    /* Ver:6 */
    MQHCONN     Hconn;            /* Connection handle */
    MQBOOL      SharingConversations; /* Multiple conversations
                                   possible on channel inst? */
    /* Ver:7 */
    MQLONG      MCAUserSource;     /* Source of the provided MCA user ID */
    PMQIEP      pEntryPoints;      /* Address of the MQIEP structure */
    /* Ver:8 */
    MQCHAR4     RemoteProduct;     /* The identifier for the remote product */
    MQCHAR8     RemoteVersion;     /* The version of the remote product */
    /* Ver:9 */
};

```

COBOL declaración

Esta declaración es la declaración COBOL para la estructura MQCXP.

```

**  MQCXP structure
10  MQCXP.
**    Structure identifier
15  MQCXP-STRUCID          PIC X(4).
**    Structure version number
15  MQCXP-VERSION         PIC S9(9) BINARY.
**    Type of exit
15  MQCXP-EXITID          PIC S9(9) BINARY.
**    Reason for invoking exit
15  MQCXP-EXITREASON      PIC S9(9) BINARY.
**    Response from exit
15  MQCXP-EXITRESPONSE    PIC S9(9) BINARY.
**    Secondary response from exit
15  MQCXP-EXITRESPONSE2   PIC S9(9) BINARY.
**    Feedback code
15  MQCXP-FEEDBACK        PIC S9(9) BINARY.
**    Maximum segment length
15  MQCXP-MAXSEGMENTLENGTH PIC S9(9) BINARY.
**    Exit user area
15  MQCXP-EXITUSERAREA    PIC X(16).
**    Exit data
15  MQCXP-EXITDATA        PIC X(32).
**    Number of times the message has been retried
15  MQCXP-MSGRETRYCOUNT  PIC S9(9) BINARY.
**    Minimum interval in milliseconds after which the put operation
**    should be retried
15  MQCXP-MSGRETRYINTERVAL PIC S9(9) BINARY.
**    Reason code from previous attempt to put the message
15  MQCXP-MSGRETRYREASON  PIC S9(9) BINARY.
**    Length of header information
15  MQCXP-HEADERLENGTH    PIC S9(9) BINARY.
**    Partner Name
15  MQCXP-PARTNERNAME     PIC X(48).
**    Negotiated Formats and Protocols level
15  MQCXP-FAPLEVEL        PIC S9(9) BINARY.
**    Capability flags

```

```

15 MQCXP-CAPABILITYFLAGS PIC S9(9) BINARY.
** Exit number
15 MQCXP-EXITNUMBER PIC S9(9) BINARY.
** Number of bytes in transmission buffer reserved for exit to use
15 MQCXP-EXITSPACE PIC S9(9) BINARY.
** User Id associated with remote certificate
15 MQCXP-SSLCERTUSERID PIC X(12).
** Length of distinguished name of issuer of remote TLS
** certificate
15 MQCXP-SSLREMCERTISSNAMELENGTH PIC S9(9) BINARY.
** Address of distinguished name of issuer of remote TLS
** certificate
15 MQCXP-SSLREMCERTISSNAMEPTR POINTER.
** Security parameters
15 MQCXP-SECURITYPARMS PIC S9(18) BINARY.
** Header data compression used for current message
15 MQCXP-CURHDRCOMPRESSION PIC S9(9) BINARY.
** Message data compression used for current message
15 MQCXP-CURMSGCOMPRESSION PIC S9(9) BINARY.
** Connection handle
15 MQCXP-HCONN PIC S9(9) BINARY.
** Multiple conversations possible on channel instance?
15 MQCXP-SHARINGCONVERSATIONS PIC S9(9) BINARY.
** Source of the provided MCA user ID
15 MQCXP-MCAUSERSOURCE PIC S9(9) BINARY.
** Identifier of the remote product
15 MQCXP-RPRODUCT PIC X(4).
** Identifier of the remote version
15 MQCXP-RVERSION PIC X(8).

```

Declaración RPG (ILE)

Esta declaración es la declaración RPG para la estructura MQCXP.

```

D*.1....:....2....:....3....:....4....:....5....:....6....:....7..
D* MQCXP Structure
D*
D* Structure identifier
D CXSID 1 4
D* Structure version number
D CXVER 5 8I 0
D* Type of exit
D CXXID 9 12I 0
D* Reason for invoking exit
D CXREA 13 16I 0
D* Response from exit
D CXRES 17 20I 0
D* Secondary response from exit
D CXRE2 21 24I 0
D* Feedback code
D CXFB 25 28I 0
D* Maximum segment length
D CXMSL 29 32I 0
D* Exit user area
D CXUA 33 48
D* Exit data
D CXDAT 49 80
D* Number of times the message has been retried
D CXMRC 81 84I 0
D* Minimum interval in milliseconds after which the put operation
D* should be retried
D CXMRI 85 88I 0
D* Reason code from previous attempt to put the message
D CXMRR 89 92I 0
D* Length of header information
D CXHDL 93 96I 0
D* Partner Name
D CXPNM 97 144
D* Negotiated Formats and Protocols level
D CXFAP 145 148I 0
D* Capability flags
D CXCAP 149 152I 0
D* Exit number
D CXEXN 153 156I 0
D* Number of bytes in transmission buffer reserved for exit to use
D CXHDL 157 160I 0
D* User identifier associated with remote TLS certificate
D CXSSLCU 161 172
D* Length of distinguished name of issuer of remote TLS certificate
D CXSRCINL 173 176I 0

```

```

D* Address of distinguished name of issuer of remote TLS certificate
D CXSRCINP      177      192*
D* Security parameters
D CXSECP        193      208*
D* Header data compression used for current message
D CXCHC        209      212I 0
D* Message data compression used for current message
D CXCMC        213      216I 0
D* Connection handle
D CXHCONN      217      220I  0
D* Multiple conversations possible on channel instance?
D CXSHARECONV  221      224I  0
D* Source of the provided MCA user ID
D MCAUSERSOURCE 225      228I  0
D* Identifier of the remote product
D CXRPRO       229      232I  0
D* Identifier of the remote version
D CXRVER       233      240I  0

```

Declaración de ensamblador System/390

Esta declaración es la declaración de ensamblador System/390 para la estructura MQCXP.

```

MQCXP          DSECT
MQCXP_STRUCID  DS    CL4   Structure identifier
MQCXP_VERSION  DS    F     Structure version number
MQCXP_EXITID   DS    F     Type of exit
MQCXP_EXITREASON DS    F   Reason for invoking exit
MQCXP_EXITRESPONSE DS    F  Response from exit
MQCXP_EXITRESPONSE2 DS    F  Secondary response from exit
MQCXP_FEEDBACK DS    F     Feedback code
MQCXP_MAXSEGMENTLENGTH DS    F  Maximum segment length
MQCXP_EXITUSERAREA DS    XL16 Exit user area
MQCXP_EXITDATA DS    CL32  Exit data
MQCXP_MSGGRETRYCOUNT DS    F  Number of times the message has been
*                               retrieved
MQCXP_MSGGRETRYINTERVAL DS    F  Minimum interval in milliseconds
*                               after which the put operation should
*                               be retried
MQCXP_MSGGRETRYREASON DS    F  Reason code from previous attempt to
*                               put the message
MQCXP_HEADERLENGTH DS    F   Length of header information
MQCXP_PARTNERNAME DS    CL48  Partner Name
MQCXP_FAPLEVEL   DS    F     Negotiated Formats and Protocols
*                               level
MQCXP_CAPABILITYFLAGS DS    F  Capability flags
MQCXP_EXITNUMBER DS    F     Exit number
MQCXP_EXITSPEACE DS    F     Number of bytes in transmission
*                               buffer reserved for exit to use
MQCXP_SSLCERTUSERID DS    CL12 User identifier associated with
*                               remote TLS certificate
MQCXP_SSLREMCERTISSNAMELENGTH DS    F  Length of distinguished name
*                               of issuer of remote TLS certificate
MQCXP_SSLREMCERTISSNAMEPTR DS    F  Address of distinguished name
*                               of issuer of remote TLS certificate
MQCXP_SECURITYPARMS DS    F   Address of security parameters
MQCXP_CURHDRCOMPRESSION DS    F  Header data compression used for
*                               current message
MQCXP_CURMSGCOMPRESSION DS    F  Message data compression used for
*                               current message
MQCXP_HCONN     DS    F     Connection handle
MQCXP_SHARINGCONVERSATIONS DS    F Multiple conversations possible on
*                               channel inst?
MQCXP_MCAUSERSOURCE DS    F   Source of the provided MCA user ID
MQCXP_RPRODUCT   DS    CL4   Identifier of the remote product
MQCXP_RVERSION   DS    CL8   Identifier of the remote version

MQCXP_LENGTH     EQU    *-MQCXP
MQCXP_AREA       DS     CL(MQCXP_LENGTH)

```

MQXWD-Descriptor de espera de salida

La estructura MQXWD es un parámetro de entrada/salida en la llamada MQXWAIT.

Esta estructura sólo está soportada en z/OS.

Referencia relacionada

[“Campos” en la página 1560](#)

Este tema lista todos los campos de la estructura MQXWD y describe cada campo.

[“Declaración C” en la página 1560](#)

Esta declaración es la declaración C para la estructura MQXWD.

[“Declaración de ensamblador System/390” en la página 1561](#)

Esta declaración es la declaración de ensamblador System/390 para la estructura MQXWD.

Campos

Este tema lista todos los campos de la estructura MQXWD y describe cada campo.

StrucId (MQCHAR4)

Este campo especifica el identificador de estructura.

El valor debe ser:

MQXWD_ID_STRUCD

Identificador de la estructura del descriptor de espera de salida.

Para el lenguaje de programación C, también se define la constante MQXWD_STRUC_ID_ARRAY; esta constante tiene el mismo valor que MQXWD_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

El valor inicial de este campo es MQXWD_STRUC_ID.

Versión (MQLONG)

Este campo especifica el número de versión de la estructura.

El valor debe ser:

MQXWD_VERSION_1

Número de versión para la estructura del descriptor de espera de salida.

El valor inicial de este campo es MQXWD_VERSION_1.

Reserved1 (MQLONG)

Este campo está reservado. Su valor debe ser cero.

Este es un campo de entrada.

Reserved2 (MQLONG)

Este campo está reservado. Su valor debe ser cero.

Este es un campo de entrada.

Reserved3 (MQLONG)

Este campo está reservado. Su valor debe ser cero.

Este es un campo de entrada.

BCE (MQLONG)

Este campo especifica el bloque de control de sucesos en el que se debe esperar.

Este campo es el bloque de control de sucesos (ECB) en el que se debe esperar. Debe establecerse en cero antes de que se emita la llamada MQXWAIT; cuando se complete correctamente, contendrá el código de publicación.

Este campo es un campo de entrada/salida.

Declaración C

Esta declaración es la declaración C para la estructura MQXWD.

```
typedef struct tagMQXWD MQXWD;  
struct tagMQXWD {
```

```

MQCHAR4  StrucId;      /* Structure identifier */
MQLONG   Version;      /* Structure version number */
MQLONG   Reserved1;    /* Reserved */
MQLONG   Reserved2;    /* Reserved */
MQLONG   Reserved3;    /* Reserved */
MQLONG   ECB;          /* Event control block to wait on */
};

```

Declaración de ensamblador System/390

Esta declaración es la declaración de ensamblador System/390 para la estructura MQXWD.

```

MQXWD      DSECT
MQXWD_STRUCID DS CL4 Structure identifier
MQXWD_VERSION DS F Structure version number
MQXWD_RESERVED1 DS F Reserved
MQXWD_RESERVED2 DS F Reserved
MQXWD_RESERVED3 DS F Reserved
MQXWD_ECB DS F Event control block to wait on
*
MQXWD_LENGTH EQU *-MQXWD
ORG MQXWD
MQXWD_AREA DS CL(MQXWD_LENGTH)

```

Llamada de salida de carga de trabajo del clúster y estructuras de datos

Esta sección proporciona información de referencia sobre la salida de la carga de trabajo del clúster y las estructuras de datos. Esta es información general sobre la interfaz de programación.

Puede escribir salidas de carga de trabajo de clúster en los siguientes lenguajes de programación:

- C
- System/390 ensamblador (IBM MQ for z/OS)

La convocatoria se describe en:

- [“MQ_CLUSTER_WORKLOAD_EXIT - Descripción de la llamada”](#) en la página 1562

Los tipos de datos de estructura utilizados por la salida se describen en:

- [“MQXCLWLN - Navegar por los registros de carga de trabajo del clúster”](#) en la página 1563
- [“MQWXP - Estructura del parámetro de salida de la carga de trabajo del clúster”](#) en la página 1567
- [“MQWDR: estructura de registro de destino de carga de trabajo del clúster”](#) en la página 1575
- [“MQWQR - Estructura de registro de cola de carga de trabajo del clúster”](#) en la página 1580
- [“MQWCR - Estructura de registro de clúster de carga de trabajo de clúster”](#) en la página 1585
-  Comportamiento asíncrono de los comandos CLUSTER en z/OS

A lo largo de esta sección, se muestran los atributos del gestor de colas y los atributos de las colas en su totalidad. A continuación se muestran los nombres equivalentes que se utilizan en los comandos MQSC. Para obtener más información sobre los comandos MQSC, consulte [Comandos MQSC](#).

Tabla 860. Atributos de gestor de colas

Nombre completo	Nombre utilizado en MQSC
<i>ClusterWorkloadData</i>	CLWLDATA
<i>ClusterWorkloadExit</i>	CLWLEXIT
<i>ClusterWorkloadLength</i>	CLWLLEN

Tabla 861. Atributos de colas

Nombre completo	Nombre utilizado en MQSC
<i>DefBind</i>	DEFBIND

Tabla 861. Atributos de colas (continuación)	
Nombre completo	Nombre utilizado en MQSC
<i>DefPersistence</i>	DEFPSIST
<i>DefPriority</i>	DEFPRTY
<i>InhibitPut</i>	PUT
<i>QDesc</i>	DESCR

Tareas relacionadas

[Escritura y compilación de salidas de carga de trabajo de clúster](#)

MQ_CLUSTER_WORKLOAD_EXIT - Descripción de la llamada

El gestor de colas llama a la salida de carga de trabajo del clúster para enviar un mensaje a un gestor de colas disponible.

Nota: El gestor de colas no proporciona ningún MQ_CLUSTER_WORKLOAD_EXIT punto de entrada denominado. En su lugar, el nombre de la salida de la carga de trabajo del clúster se define mediante el atributo del ClusterWorkloadExit gestor de colas.

La MQ_CLUSTER_WORKLOAD_EXIT salida es compatible con todas las plataformas.

Sintaxis

```
MQ_CLUSTER_WORKLOAD_EXIT (ExitParms)
```

Referencia relacionada

[MQXCLWLN - Navegar por los registros de carga de trabajo del clúster](#)

La MQXCLWLN llamada se utiliza para navegar por las cadenas de registros MQWDR, MQWQR y MQWCR almacenados en la caché del clúster.

[MQWXP - Estructura del parámetro de salida de la carga de trabajo del clúster](#)

La siguiente tabla resume los campos de la estructura del parámetro de salida de carga de trabajo del clúster MQWXP.

[MQWDR: estructura de registro de destino de carga de trabajo del clúster](#)

La siguiente tabla resume los campos de la estructura del registro de destino de la carga de trabajo del clúster MQWDR.

[MQWQR - Estructura de registro de cola de carga de trabajo del clúster](#)

La siguiente tabla resume los campos de la estructura del registro MQWQR - Cola de carga de trabajo del clúster.

[MQWCR - Estructura de registro de clúster de carga de trabajo de clúster](#)

La siguiente tabla resume los campos de la estructura del registro de carga de trabajo del clúster MQWCR.

parámetros para MQ_CLUSTER_WORKLOAD_EXIT

Descripción de los parámetros de la MQ_CLUSTER_WORKLOAD_EXIT llamada.

ExitParms (MQWXP) - entrada/salida

Salir del bloque de parámetros.

- La salida establece información en MQWXP para indicar cómo gestionar la carga de trabajo.

Referencia relacionada

[Notas de uso](#)

La función que realiza la salida de carga de trabajo del clúster la define el proveedor de la salida. Sin embargo, la salida debe ajustarse a las reglas definidas en el bloque de control asociado MQWXP.

Invocaciones de lenguaje para MQ_CLUSTER_WORKLOAD_EXIT
El MQ_CLUSTER_WORKLOAD_EXIT admite dos lenguajes, C y High Level Assembler

Notas de uso

La función que realiza la salida de carga de trabajo del clúster la define el proveedor de la salida. Sin embargo, la salida debe ajustarse a las reglas definidas en el bloque de control asociado MQWXP.

El gestor de colas no proporciona ningún MQ_CLUSTER_WORKLOAD_EXIT punto de entrada denominado. Sin embargo, se proporciona un typedef para el nombre MQ_CLUSTER_WORKLOAD_EXIT en el lenguaje de programación C. Utilice el typedef para declarar la salida escrita por el usuario, a fin de garantizar que los parámetros sean correctos.

Referencia relacionada

parámetros para MQ_CLUSTER_WORKLOAD_EXIT
Descripción de los parámetros de la MQ_CLUSTER_WORKLOAD_EXIT llamada.

Invocaciones de lenguaje para MQ_CLUSTER_WORKLOAD_EXIT
El MQ_CLUSTER_WORKLOAD_EXIT admite dos lenguajes, C y High Level Assembler

Invocaciones de lenguaje para MQ_CLUSTER_WORKLOAD_EXIT

El MQ_CLUSTER_WORKLOAD_EXIT admite dos lenguajes, C y High Level Assembler

Invocación en C

```
MQ_CLUSTER_WORKLOAD_EXIT (&ExitParms);
```

Reemplace *MQ_CLUSTER_WORKLOAD_EXIT* por el nombre de la función de salida de la carga de trabajo de su clúster.

Declare los **MQ_CLUSTER_WORKLOAD_EXIT** parámetros de la siguiente manera:

```
MQWXP ExitParms; /* Exit parameter block */
```

Invocación en ensamblador de alto nivel

```
CALL EXITNAME,(EXITPARMS)
```

Declare los parámetros como se indica a continuación:

EXITPARMS	CMQWXP	Exit parameter block

Referencia relacionada

parámetros para MQ_CLUSTER_WORKLOAD_EXIT
Descripción de los parámetros de la MQ_CLUSTER_WORKLOAD_EXIT llamada.

Notas de uso

La función que realiza la salida de carga de trabajo del clúster la define el proveedor de la salida. Sin embargo, la salida debe ajustarse a las reglas definidas en el bloque de control asociado MQWXP.

MQXCLWLN - Navegar por los registros de carga de trabajo del clúster

La MQXCLWLN llamada se utiliza para navegar por las cadenas de registros MQWDR, MQWQR y MQWCR almacenados en la caché del clúster.

La caché del clúster es un área de la memoria principal que se utiliza para almacenar información relacionada con el clúster.

Si la caché del clúster es estática, tiene un tamaño fijo. Si lo configura como dinámico, la caché del clúster se puede ampliar según sea necesario.

Establezca el tipo de caché del clúster en ESTÁTICO o DINÁMICO utilizando un parámetro del sistema o una macro.

-  Utilice el parámetro del sistema `ClusterCacheType` en [Multiplataformas](#).
-  Utilice el parámetro `CLCACHE` en la `CSQ6SYSP` macro en z/OS.

Sintaxis

```
MQXCLWLN (ExitParms, CurrentRecord, NextOffset, NextRecord, Compcode, Reason)
```

Referencia relacionada

[MQ_CLUSTER_WORKLOAD_EXIT](#) - Descripción de la llamada

El gestor de colas llama a la salida de carga de trabajo del clúster para enviar un mensaje a un gestor de colas disponible.

[MQWXP](#) - Estructura del parámetro de salida de la carga de trabajo del clúster

La siguiente tabla resume los campos de la estructura del parámetro de salida de carga de trabajo del clúster `MQWXP`.

[MQWDR](#): estructura de registro de destino de carga de trabajo del clúster

La siguiente tabla resume los campos de la estructura del registro de destino de la carga de trabajo del clúster `MQWDR`.

[MQWQR](#) - Estructura de registro de cola de carga de trabajo del clúster

La siguiente tabla resume los campos de la estructura del registro `MQWQR` - Cola de carga de trabajo del clúster.

[MQWCR](#) - Estructura de registro de clúster de carga de trabajo de clúster

La siguiente tabla resume los campos de la estructura del registro de carga de trabajo del clúster `MQWCR`.

Parámetros para MQXCLWLN - Navegar por los registros de carga de trabajo del clúster

Descripción de los parámetros de la `MQXCLWLN` llamada.

ExitParms (MQWXP) - entrada/salida

Salir del bloque de parámetros.

Esta estructura contiene información relacionada con la invocación de la salida. La salida establece información en esta estructura para indicar cómo gestionar la carga de trabajo.

CurrentRecord (MQPTR) - entrada

Dirección del registro actual.

Esta estructura contiene información relacionada con la dirección del registro que está siendo examinado actualmente por la salida. El registro debe ser de uno de los siguientes tipos:

- Registro de destino de carga de trabajo del clúster (`MQWDR`)
- Registro de cola de carga de trabajo del clúster (`MQWQR`)
- Registro de carga de trabajo del clúster (`MQWCR`)

NextOffset (MQLONG) - entrada

Desplazamiento del siguiente registro.

Esta estructura contiene información relacionada con el desplazamiento del siguiente registro o estructura. *NextOffset* es el valor del campo de desplazamiento adecuado en el registro actual, y debe ser uno de los siguientes campos:

- `ChannelDefOffset` campo en `MQWDR`
- `ClusterRecOffset` campo en `MQWDR`
- `ClusterRecOffset` campo en `MQWQR`
- `ClusterRecOffset` campo en `MQWCR`

NextRecord (MQPTR) - salida

Dirección del siguiente registro o estructura.

Esta estructura contiene información relacionada con la dirección del siguiente registro o estructura. Si *CurrentRecord* es la dirección de un MQWDR y *NextOffset* es el valor del ChannelDefOffset campo, *NextRecord* es la dirección de la estructura de definición del canal (MQCD).

Si no hay ningún registro o estructura siguiente, el gestor de colas establece *NextRecord* en el puntero nulo, y la llamada devuelve el código de finalización MQCC_WARNING y el código de motivo MQRC_NO_RECORD_AVAILABLE.

CompCode (MQLONG) - salida

Código de terminación.

El código de finalización tiene uno de los siguientes valores:

MQCC_OK

Realización satisfactoria.

MQCC_WARNING

Aviso (finalización parcial).

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón (MQLONG) - salida

Código de motivo calificativo CompCode

Si CompCode es MQCC_OK:

MQRC_NONE

(0, X'0000')

No hay motivos para informar.

Si CompCode es MQCC_WARNING:

MQRC_NO_RECORD_AVAILABLE

(2359, X'0937')

No hay registros disponibles. Se emitió una MQXCLWLN llamada desde una salida de carga de trabajo del clúster para obtener la dirección del siguiente registro de la cadena. El registro actual es el último registro de la cadena. Medida correctiva: Ninguna.

Si CompCode es MQCC_FAILED:

MQRC_CURRENT_RECORD_ERROR

(2357, X'0935')

CurrentRecord Parámetro no válido. Se emitió una MQXCLWLN llamada desde una salida de carga de trabajo del clúster para obtener la dirección del siguiente registro de la cadena. La dirección especificada por el **CurrentRecord** parámetro no es la dirección de un registro válido.

CurrentRecord Debe ser la dirección de un registro de destino, MQWDR, registro de cola (MQWQR) o registro de clúster (MQWCR) que reside dentro de la caché del clúster. Acción correctiva: Asegúrese de que la salida de la carga de trabajo del clúster pase la dirección de un registro válido que resida en la caché del clúster.

MQRC_ENVIRONMENT_ERROR

(2012, X'07DC')

Llamada no válida en el entorno. Se emitió una MQXCLWLN llamada, pero no desde una salida de carga de trabajo del clúster.

MQRC_NEXT_OFFSET_ERROR

(2358, X'0936')

NextOffset Parámetro no válido. Se emitió una MQXCLWLN llamada desde una salida de carga de trabajo del clúster para obtener la dirección del siguiente registro de la cadena. El desplazamiento especificado por el **NextOffset** parámetro no es válido. **NextOffset** debe ser el valor de uno de los campos siguientes:

- ChannelDefOffset campo en MQWDR

- ClusterRecOffset campo en MQWDR
- ClusterRecOffset campo en MQWQR
- ClusterRecOffset campo en MQWCR

Acción correctiva: Asegúrese de que el valor especificado para el **NextOffset** parámetro sea el valor de uno de los campos enumerados anteriormente.

MQRC_NEXT_RECORD_ERROR (2361, X'0939')

NextRecord Parámetro no válido.

MQRC_WXP_ERROR (2356, X'0934')

Estructura del parámetro de salida de la carga de trabajo no válida. Se emitió una MQXCLWLN llamada desde una salida de carga de trabajo del clúster para obtener la dirección del siguiente registro de la cadena. La estructura del parámetro de salida de **ExitParms** la carga de trabajo no es válida por uno de los siguientes motivos:

- El puntero del parámetro no es válido. No siempre es posible detectar punteros de parámetros que no son válidos; si no se detectan, se producen resultados impredecibles.
- El StrucId campo no es MQWXP_STRUC_ID.
- El campo Versión no es MQWXP_VERSION_2.
- El campo Contexto no contiene el valor pasado a la salida por el gestor de colas.

Acción correctiva: Asegúrese de que el parámetro especificado para **ExitParms** sea la MQWXP estructura que se pasó a la salida cuando se invocó la salida.

Referencia relacionada

Notas de uso para MQXCLWLN: navegar por los registros de carga de trabajo del clúster
Utilice MQXCLWLN para navegar por los registros del clúster, incluso si la caché es estática.

Invocaciones lingüísticas de MQXCLWLN

MQXCLWLN admite dos lenguajes, C y High Level Assembler.

Notas de uso para MQXCLWLN: navegar por los registros de carga de trabajo del clúster

Utilice MQXCLWLN para navegar por los registros del clúster, incluso si la caché es estática.

Si la caché del clúster es dinámica, se debe utilizar la MQXCLWLN llamada para navegar por los registros. La salida finaliza de forma anómala si se utiliza aritmética simple de puntero y desplazamiento para navegar por los registros.

Si la caché del clúster es estática, no MQXCLWLN es necesario utilizarla para navegar por los registros. Normalmente se utiliza MQXCLWLN incluso cuando la caché es estática. A continuación, puede cambiar la caché del clúster a dinámica sin necesidad de cambiar la salida de la carga de trabajo.

Referencia relacionada

Parámetros para MQXCLWLN - Navegar por los registros de carga de trabajo del clúster
Descripción de los parámetros de la MQXCLWLN llamada.

Invocaciones lingüísticas de MQXCLWLN

MQXCLWLN admite dos lenguajes, C y High Level Assembler.

Invocaciones lingüísticas de MQXCLWLN

MQXCLWLN admite dos lenguajes, C y High Level Assembler.

Invocación en C

```
MQXCLWLN (&ExitParms, CurrentRecord, NextOffset, &NextRecord, &CompCode, &Reason) ;
```

Declare los parámetros como se indica a continuación:

```

Typedef struct tagMQXCLWLN {
MQWXP    ExitParms;      /* Exit parameter block */
MQPTR    CurrentRecord; /* Address of current record*/
MQLONG   NextOffset;    /* Offset of next record */
MQPTR    NextRecord;    /* Address of next record or structure */
MQLONG   CompCode;      /* Completion code */
MQLONG   Reason;        /* Reason code qualifying CompCode */

```

Invocación en ensamblador de alto nivel

```
CALL MQXCLWLN, (CLWLEXITPARMS, CURRENTRECORD, NEXTOFFSET, NEXTRECORD, COMPCODE, REASON)
```

Declare los parámetros como se indica a continuación:

```

CLWLEXITPARMS CMQWXP, Cluster workload exit parameter block
CURRENTRECORD CMQWDRA, Current record
NEXTOFFSET    DS F      Next offset
NEXTRECORD    DS F      Next record
COMPCODE      DS F      Completion code
REASON        DS F      Reason code qualifying COMPCODE

```

Referencia relacionada

Parámetros para MQXCLWLN - [Navegar por los registros de carga de trabajo del clúster](#)
Descripción de los parámetros de la MQXCLWLN llamada.

Notas de uso para MQXCLWLN: [navegar por los registros de carga de trabajo del clúster](#)
Utilice MQXCLWLN para navegar por los registros del clúster, incluso si la caché es estática.

MQWXP - Estructura del parámetro de salida de la carga de trabajo del clúster

La siguiente tabla resume los campos de la estructura del parámetro de salida de carga de trabajo del clúster MQWXP.

Tabla 862. Campos en MQWXP		
Campo	Descripción	Página
<i>StrucId</i>	Identificador de la estructura	StrucId
<i>Version</i>	Número de versión de la estructura	Versión
<i>ExitId</i>	Tipo de salida	ExitId
<i>ExitReason</i>	Motivo para invocar la salida	ExitReason
<i>ExitResponse</i>	Respuesta de la salida	ExitResponse
<i>ExitResponse2</i>	Respuesta secundaria desde la salida	ExitResponse2
<i>Feedback</i>	Código de retroalimentación	FEEDBACK
<i>Flags</i>	Valores de las banderas. Estos indicadores de bits se utilizan para indicar información sobre el mensaje que se está colocando	Indicadores
<i>ExitUserArea</i>	Salir del área de usuario	ExitUserArea
<i>ExitData</i>	Datos de salida	ExitData
<i>MsgDescPtr</i>	Dirección del descriptor de mensaje (MQMD)	MsgDescPtr
<i>MsgBufferPtr</i>	Dirección del búfer que contiene algunos o todos los datos del mensaje	MsgBufferPtr
<i>MsgBufferLength</i>	Longitud del búfer que contiene los datos del mensaje	MsgBufferLength

Tabla 862. Campos en MQWXP (continuación)		
Campo	Descripción	Página
<i>MsgLength</i>	Longitud del mensaje completo	MsgLength
<i>QName</i>	Nombre de la cola	QName
<i>QMgrName</i>	Nombre del administrador de colas local	QMgrName
<i>DestinationCount</i>	Número de destinos posibles	DestinationCount
<i>DestinationChosen</i>	Destino elegido	DestinationChosen
<i>DestinationArrayPtr</i>	Dirección de una matriz de punteros a registros de destino (MQWDR)	DestinationArrayPtr
<i>QArrayPtr</i>	Dirección de una matriz de punteros a registros de cola (MQWQR)	QArrayPtr
Nota: Los campos restantes se ignoran si la versión es inferior a MQWXP_VERSION_2.		
<i>CacheContext</i>	Información de contexto	CacheContext
<i>CacheType</i>	Tipo de caché de clúster	CacheType
Nota: Los campos restantes se ignoran si la versión es inferior a MQWXP_VERSION_3.		
<i>CLWLMRUChannels</i>	Número máximo de canales de clúster salientes activos permitidos	CLWLMRUChannels
Nota: Los campos restantes se ignoran si la versión es inferior a MQWXP_VERSION_4.		
<i>pEntryPoints</i>	Configuración de la estructura MQIEP para permitir la realización de llamadas de tipo « MQI » y « DCI »	pEntryPoints

La estructura del parámetro de salida de la carga de trabajo del clúster describe la información que se pasa a la salida de la carga de trabajo del clúster.

La estructura del parámetro de salida de la carga de trabajo del clúster es compatible con todas las plataformas

Además, las estructuras MQWXP1, MQWXP2 MQWXP3 y están disponibles para garantizar la compatibilidad con versiones anteriores.

Referencia relacionada

[MQ_CLUSTER_WORKLOAD_EXIT](#) - Descripción de la llamada

El gestor de colas llama a la salida de carga de trabajo del clúster para enviar un mensaje a un gestor de colas disponible.

[MQXCLWLN](#) - Navegar por los registros de carga de trabajo del clúster

La MQXCLWLN llamada se utiliza para navegar por las cadenas de registros MQWDR, MQWQR y MQWCR almacenados en la caché del clúster.

[MQWDR](#): estructura de registro de destino de carga de trabajo del clúster

La siguiente tabla resume los campos de la estructura del registro de destino de la carga de trabajo del clúster MQWDR.

[MQWQR](#) - Estructura de registro de cola de carga de trabajo del clúster

La siguiente tabla resume los campos de la estructura del registro MQWQR - Cola de carga de trabajo del clúster.

[MQWCR](#) - Estructura de registro de clúster de carga de trabajo de clúster

La siguiente tabla resume los campos de la estructura del registro de carga de trabajo del clúster MQWCR.

Campos en MQWXP : estructura del parámetro de salida de la carga de trabajo del clúster

Descripción de los campos en MQWXP : estructura del parámetro de salida de carga de trabajo del clúster

StrucId (MQCHAR4) - entrada

El identificador de estructura para la estructura del parámetro de salida de la carga de trabajo del clúster.

- El StrucId valor es MQWXP_STRUC_ID.
- Para el lenguaje de programación C, también se MQWXP_STRUC_ID_ARRAY define la constante. Tiene el mismo valor que MQWXP_STRUC_ID. Es una matriz de caracteres en lugar de una cadena.

Versión (MQLONG) - entrada

Indica el número de versión de la estructura. La versión toma uno de los siguientes valores:

MQWXP_VERSION_1

Version-1 Estructura del parámetro de salida de la carga de trabajo del clúster.

MQWXP_VERSION_1 Es compatible con todos los entornos.

MQWXP_VERSION_2

Version-2 Estructura del parámetro de salida de la carga de trabajo del clúster.

MQWXP_VERSION_2 Es compatible con los siguientes entornos:

-  AIX
-  IBM i
-  Linux
-  Windows

MQWXP_VERSION_3

Version-3 Estructura del parámetro de salida de la carga de trabajo del clúster.

MQWXP_VERSION_3 Es compatible con los siguientes entornos:

-  AIX
-  IBM i
-  Linux
-  Windows

MQWXP_VERSION_4

Version-4 Estructura del parámetro de salida de la carga de trabajo del clúster.

MQWXP_VERSION_4 Es compatible con los siguientes entornos:

-  AIX
-  IBM i
-  Linux
-  Windows

MQWXP_CURRENT_VERSION

Versión actual de la estructura del parámetro de salida de la carga de trabajo del clúster.

ExitId (MQLONG) - entrada

Indica el tipo de salida que se está llamando. La salida de carga de trabajo del clúster es la única salida compatible.

- El ExitId valor debe ser MQXT_CLUSTER_WORKLOAD_EXIT

ExitReason (MQLONG) - entrada

Indica el motivo por el que se invoca la salida de la carga de trabajo del clúster. ExitReason toma uno de los siguientes valores:

MQXR_INIT

Indica que la salida se está invocando por primera vez.

Adquiera e inicialice cualquier recurso que la salida pueda necesitar, como el almacenamiento principal.

MQXR_TERM

Indica que la salida está a punto de finalizar.

Libera todos los recursos que la salida haya podido adquirir desde su inicialización, como la memoria principal.

MQXR_CLWL_OPEN

Llamado por MQOPEN.

MQXR_CLWL_PUT

Llamado por MQPUT o MQPUT1.

MQXR_CLWL_MOVE

Llamado por MCA cuando el estado del canal ha cambiado.

MQXR_CLWL_REPOS

Llamado por MQPUT o MQPUT1 para un mensaje PCF del gestor de repositorios.

MQXR_CLWL_REPOS_MOVE

Llamado por MCA para un mensaje PCF del gestor de repositorios si el estado del canal ha cambiado.

ExitResponse (MQLONG) - salida

Establezca ExitResponse para indicar si continúa el procesamiento del mensaje. Tiene que ser uno de los valores siguientes:

MQXCC_OK

Continúa procesando el mensaje normalmente.

- DestinationChosen identifica el destino al que se enviará el mensaje.

MQXCC_SUPPRESS_FUNCTION

Interrumpir el procesamiento del mensaje.

- Las acciones realizadas por el gestor de colas dependen del motivo por el que se invocó la salida:

Tabla 863. Acciones realizadas por el gestor de colas	
ExitReason	Acción emprendida
– MQXR_CLWL_OPEN – MQXR_CLWL_REPOS – MQXR_CLWL_PUT	MQOPEN, MQPUT, o MQPUT1 llamada fallida con código de finalización MQCC_FAILED y código de motivo MQRC_STOPPED_BY_CLUSTER_EXIT.
– MQXR_CLWL_MOVE – MQXR_CLWL_REPOS_MOVE	El mensaje se coloca en la cola de mensajes no entregados.

MQXCC_SUPPRESS_EXIT

Continúa procesando el mensaje actual con normalidad. No vuelva a invocar la salida hasta que el gestor de colas se apague.

El gestor de colas procesa los mensajes posteriores como si el atributo del ClusterWorkloadExit gestor de colas estuviera en blanco. DestinationChosen identifica el destino al que se envía el mensaje actual.

Cualquier otro valor

Procese el mensaje como si MQXCC_SUPPRESS_FUNCTION estuviera especificado.

ExitResponse2 (MQLONG) - entrada/salida

Configure ExitResponse2 para proporcionar más información al gestor de colas.

- MQXR2_STATIC_CACHE es el valor predeterminado y se establece al entrar en la salida.
- Cuando ExitReason tiene el valor MQXR_INIT, la salida puede establecer uno de los siguientes valores en ExitResponse2 :

MQXR2_STATIC_CACHE

La salida requiere una caché de clúster estática.

- Si la caché del clúster es estática, la salida no necesita utilizar la MQXCLWLN llamada para navegar por las cadenas de registros de la caché del clúster.
- Si la caché del clúster es dinámica, la salida no puede navegar correctamente por los registros de la caché.

Nota: El gestor de colas procesa el retorno de la MQXR_INIT llamada como si la salida hubiera retornado MQXCC_SUPPRESS_EXIT en el ExitResponse campo.

MQXR2_DYNAMIC_CACHE

La salida puede funcionar con una caché estática o dinámica.

- Si la salida devuelve este valor, la salida debe utilizar la MQXCLWLN llamada para navegar por las cadenas de registros en la caché del clúster.

Comentarios (MQLONG) - entrada

Un campo reservado. El valor es cero.

Banderas (MQLONG) - entrada

Indica información sobre el mensaje que se está enviando.

- El valor de Flags es MQWXP_PUT_BY_CLUSTER_CHL. El mensaje se origina en un canal de clúster, en lugar de hacerlo localmente o en un canal que no sea de clúster. En otras palabras, el mensaje proviene de otro gestor de colas de clúster.

Reservado (MQLONG) - entrada

Un campo reservado. El valor es cero.

ExitUserArea (MQBYTE16) - entrada/salida

Configurar ExitUserArea para comunicarse entre llamadas a la salida.

- ExitUserArea se inicializa con un cero binario antes de la primera invocación de la salida. Cualquier cambio realizado en este campo por la salida se conserva en todas las invocaciones de la salida que se producen entre la MQCONN llamada y la llamada MQDISC correspondiente. El campo se restablece a cero binario cuando se produce la MQDISC llamada.
- La primera invocación de la salida se indica mediante el ExitReason campo con el valor MQXR_INIT.
- Se definen las siguientes constantes:

MQXUA_NONE - cadena**MQXUA_NONE_ARRAY - matriz de caracteres**

Sin información sobre el usuario. Ambas constantes son cero binario para la longitud del campo.

MQ_EXIT_USER_AREA_LENGTH

La longitud de ExitUserArea.

ExitData (MQCHAR32) - entrada

El valor del atributo del gestor ClusterWorkloadData de colas. Si no se ha definido ningún valor para ese atributo, este campo contendrá espacios en blanco.

- La longitud de ExitData viene dada por MQ_EXIT_DATA_LENGTH.

MsgDescPtr (PMQMD) - entrada

La dirección de una copia del descriptor de mensaje (MQMD) para el mensaje que se está procesando.

- El gestor de colas ignora cualquier cambio realizado en el descriptor del mensaje por la salida.
- Si `ExitReason` tiene uno de los siguientes valores, `MsgDescPtr` se establece en el puntero nulo y no se pasa ningún descriptor de mensaje a la salida:
 - `MQXR_INIT`
 - `MQXR_TERM`
 - `MQXR_CLWL_OPEN`

MsgBufferPtr (PMQVOID) - entrada

La dirección de un búfer que contiene una copia de los primeros `MsgBufferLength` bytes de los datos del mensaje.

- El gestor de colas ignora cualquier cambio realizado en los datos del mensaje por la salida.
- No se transmiten datos de mensaje a la salida cuando:
 - `MsgDescPtr` es el puntero nulo.
 - El mensaje no contiene datos.
 - El atributo del `ClusterWorkloadLength` gestor de colas es cero.

En estos casos, `MsgBufferPtr` es el puntero nulo.

MsgBufferLength (MQLONG) - entrada

La longitud del búfer que contiene los datos del mensaje pasados a la salida.

- La longitud se controla mediante el atributo del `ClusterWorkloadLength` gestor de colas.
- La longitud puede ser inferior a la longitud del mensaje completo, véase `MsgLength`.

MsgLength (MQLONG) - entrada

La longitud del mensaje completo pasado a la salida.

- `MsgBufferLength` puede ser menor que la longitud del mensaje completo.
- `MsgLength` es cero si `ExitReason` es `MQXR_INIT`, `MQXR_TERM`, o `MQXR_CLWL_OPEN`.

QName (MQCHAR48) - entrada

El nombre de la cola de destino. La cola es una cola de clúster.

- La longitud de `QName` es `MQ_Q_NAME_LENGTH`.

QMgrName (MQCHAR48) - entrada

El nombre del administrador de colas local que ha invocado la salida de la carga de trabajo del clúster.

- La longitud de `QMgrName` es `MQ_Q_MGR_NAME_LENGTH`.

DestinationCount (MQLONG) - entrada

El número de destinos posibles. Los destinos son instancias de la cola de destinos y se describen mediante registros de destino.

- Un registro de destino es una estructura `MQWDR`. Hay una estructura para cada ruta posible hacia cada instancia de la cola.
- Las estructuras `MQWDR` se abordan mediante una matriz de punteros, véase `DestinationArrayPtr`.

DestinationChosen (MQLONG) - entrada/salida

El destino elegido.

- El número de la estructura `MQWDR` que identifica la ruta y la instancia de cola donde se enviará el mensaje.
- El valor está en el rango 1 - `DestinationCount`.
- En la entrada a la salida, `DestinationChosen` indica la ruta y la instancia de cola que ha seleccionado el gestor de colas. La salida puede aceptar esta elección o elegir una ruta y una instancia de cola diferentes.

- El valor establecido por la salida debe estar comprendido entre 1 y DestinationCount. Si se devuelve cualquier otro valor, el gestor de colas utiliza el valor de DestinationChosen en la entrada a la salida.

DestinationArrayPtr (PPMQWDR) - entrada

La dirección de una matriz de punteros a registros de destino (MQWDR).

- Hay DestinationCount registros de destino.

QArrayPtr (PPMQWQR) - entrada

La dirección de una matriz de punteros a registros de cola (MQWQR).

- Si hay registros de cola disponibles, hay DestinationCount de ellos.
- Si no hay registros de cola disponibles, QArrayPtr es el puntero nulo.

Nota: QArrayPtr puede ser el puntero nulo incluso cuando DestinationCount es mayor que cero.

CacheContext (MQPTR): Versión 2 - entrada

El CacheContext campo está reservado para su uso por parte del gestor de colas. La salida no debe alterar el valor de este campo.

CacheType (MQLONG): Versión 2 - entrada

La caché del clúster tiene uno de los siguientes tipos:

MQCLCT_STATIC

La caché es estática.

- El tamaño de la caché es fijo y no puede aumentar mientras el gestor de colas está en funcionamiento.
- No es necesario utilizar la MQXCLWLN llamada para navegar por los registros en este tipo de caché.

MQCLCT_DYNAMIC

La caché es dinámica.

- El tamaño de la caché puede aumentar para dar cabida a la información variable del clúster.
- Debe utilizar la MQXCLWLN llamada para navegar por los registros en este tipo de caché.

CLWLMRUChannels (MQLONG): Versión 3 - Entrada

Indica el número máximo de canales de clúster salientes activos que se tendrán en cuenta para su uso por parte del algoritmo de selección de carga de trabajo del clúster.

- CLWLMRUChannels es un valor comprendido entre 1 y 999 999 999.

pEntryPoints (PMQIEP): Versión 4

La dirección de una estructura MQIEP a través de la cual se pueden realizar llamadas de tipo « MQI » y « DCI ».

Referencia relacionada

[Valores iniciales y declaraciones de idioma para MQWXP](#)

Valores iniciales y declaraciones de lenguaje C y e High Level Assembler e para MQWXP : estructura del parámetro de salida de la carga de trabajo del clúster.

Valores iniciales y declaraciones de idioma para MQWXP

Valores iniciales y declaraciones de lenguaje C y e High Level Assembler e para MQWXP : estructura del parámetro de salida de la carga de trabajo del clúster.

Tabla 864. Campos en MQWXP		
Nombre de campo	Nombre de la constante	Valor de la constante
StrucId	MQWXP_STRUC_ID	'WXP' - 1
Version	MQWXP_VERSION_2	2

Tabla 864. Campos en MQWXP (continuación)

Nombre de campo	Nombre de la constante	Valor de la constante
<i>ExitId</i>	Ninguna	0
<i>ExitReason</i>	MQXCC_OK	0
<i>ExitResponse</i>	Ninguna	0
<i>ExitResponse2</i>	Ninguna	0
<i>Flags</i>	Ninguna	0
<i>ExitUserArea</i>	{MQXUA_NONE_ARRAY}	0
<i>ExitData</i>	Ninguna	" "
<i>MsgDescPtr</i>	Ninguna	NULL
<i>MsgBufferPtr</i>	Ninguna	NULL
<i>MsgBufferLength</i>	Ninguna	0
<i>MsgBufferPtr</i>	Ninguna	0
<i>QName</i>	Ninguna	" "
<i>QMgrName</i>	Ninguna	" "
<i>DestinationCount</i>	Ninguna	0
<i>DestinationChosen</i>	Ninguna	0
<i>DestinationArrayPtr</i>	Ninguna	NULL
<i>QArrayPtr</i>	Ninguna	NULL
<i>CacheContext</i>	Ninguna	NULL
<i>CacheType</i>	MQCLCT_DYNAMIC	1
<i>CLWLMRUChannels</i>	Ninguna	0
<i>pEntryPoints</i>	Ninguna	NULL

Notas:

1. El símbolo ↵ representa un único carácter en blanco.
2. En el lenguaje de programación C, la variable MQWXP_DEFAULT macro contiene los valores predeterminados. Úselo de la siguiente manera para proporcionar valores iniciales para los campos de la estructura:

```
MQWDR MyWXP = {MQWXP_DEFAULT};
```

Declaración C

```
typedef struct tagMQWXP {
    MQCHAR4   StrucId;           /* Structure identifier */
    MQLONG    Version;           /* Structure version number */
    MQLONG    ExitId;            /* Type of exit */
    MQLONG    ExitReason;        /* Reason for invoking exit */
    MQLONG    ExitResponse;      /* Response from exit */
    MQLONG    ExitResponse2;     /* Reserved */
    MQLONG    Feedback;          /* Reserved */
    MQLONG    Flags;             /* Flags */
    MQBYTE16  ExitUserArea;      /* Exit user area */
    MQCHAR32  ExitData;          /* Exit data */
}
```

```

PMQMD      MsgDescPtr;          /* Address of message descriptor */
PMQVOID     MsgBufferPtr;        /* Address of buffer containing some
                                or all of the message data */

MQLONG      MsgBufferLength;     /* Length of buffer containing message
                                data */

MQLONG      MsgLength;           /* Length of complete message */
MQCHAR48    QName;              /* Queue name */
MQCHAR48    QMgrName;           /* Name of local queue manager */
MQLONG      DestinationCount;    /* Number of possible destinations */
MQLONG      DestinationChosen;   /* Destination chosen */
PPMQWDR     DestinationArrayPtr; /* Address of an array of pointers to
                                destination records */

PPMQWQR     QArrayPtr;           /* Address of an array of pointers to
                                queue records */

/* version 1 */
MQPTR       CacheContext;        /* Context information */
MQLONG      CacheType;           /* Type of cluster cache */
/* version 2 */
MQLONG      CLWLMRUChannels;     /* Maximum number of most recently
                                used cluster channels */

/* version 3 */
PMQIEP      pEntryPoints;        /* Address of the MQIEP structure */
/* version 4 */
};

```

High Level Assembler

MQWXP	DSECT		
MQWXP_STRUCID	DS	CL4	Structure identifier
MQWXP_VERSION	DS	F	Structure version number
MQWXP_EXITID	DS	F	Type of exit
MQWXP_EXITREASON	DS	F	Reason for invoking exit
MQWXP_EXITRESPONSE	DS	F	Response from exit
MQWXP_EXITRESPONSE2	DS	F	Reserved
MQWXP_FEEDBACK	DS	F	Reserved
MQWXP_RESERVED	DS	F	Reserved
MQWXP_EXITUSERAREA	DS	XL16	Exit user area
MQWXP_EXITDATA	DS	CL32	Exit data
MQWXP_MSGDESCPTR	DS	F	Address of message descriptor
* MQWXP_MSGBUFFERPTR	DS	F	Address of buffer containing some or all of the message data
* MQWXP_MSGBUFFERLENGTH	DS	F	Length of buffer containing message data
* MQWXP_MSGLENGTH	DS	F	Length of complete message
MQWXP_QNAME	DS	CL48	Queue name
MQWXP_QMGRNAME	DS	CL48	Name of local queue manager
MQWXP_DESTINATIONCOUNT	DS	F	Number of possible destinations
* MQWXP_DESTINATIONCHOSEN	DS	F	Destination chosen
MQWXP_DESTINATIONARRAYPTR	DS	F	Address of an array of pointers to destination records
* MQWXP_QARRAYPTR	DS	F	Address of an array of pointers to queue records
* MQWXP_CACHECONTEXT	DS	F	Context information
MQWXP_CACHETYPE	DS	F	Type of cluster cache
MQWXP_CLWLMRUCHANNELS	DS	F	Number of most recently used channels for workload balancing
* MQWXP_LENGTH	EQU	*-MQWXP	Length of structure
MQWXP_AREA	ORG	MQWXP	
	DS	CL(MQWXP_LENGTH)	

Referencia relacionada

[Campos en MQWXP : estructura del parámetro de salida de la carga de trabajo del clúster](#)

Descripción de los campos en MQWXP : estructura del parámetro de salida de carga de trabajo del clúster

MQWDR: estructura de registro de destino de carga de trabajo del clúster

La siguiente tabla resume los campos de la estructura del registro de destino de la carga de trabajo del clúster MQWDR.

Tabla 865. Campos en MQWDR		
Campo	Descripción	Página
<i>StrucId</i>	Identificador de la estructura	StrucId
<i>Version</i>	Número de versión de la estructura	Versión
<i>StrucLength</i>	Longitud de la estructura MQWDR	StrucLength
<i>QMgrFlags</i>	Indicadores del gestor de colas	QMgrFlags
<i>QMgrIdentifier</i>	Identificador del gestor de colas	QMgrIdentifier
<i>QMgrName</i>	Nombre del gestor de colas	QMgrName
<i>ClusterRecOffset</i>	Desplazamiento lógico del primer registro del clúster (MQWCR)	ClusterRecOffset
<i>ChannelState</i>	Estado del canal	ChannelState
<i>ChannelDefOffset</i>	Desplazamiento lógico de la estructura de definición de canal (MQCD)	ChannelDefOffset
Nota: Los campos restantes se ignoran si la versión es inferior a MQWDR_VERSION_2.		
<i>DestSeqNumber</i>	Número de secuencia de destino del canal	DestSeqNumber
<i>DestSeqFactor</i>	Factor de secuencia de destino del canal para la ponderación	DestSeqFactor

La estructura del registro de destino de la carga de trabajo del clúster contiene información relacionada con uno de los posibles destinos del mensaje. Hay una estructura de registro de destino de carga de trabajo de clúster para cada instancia de la cola de destino.

La estructura del registro de destino de la carga de trabajo del clúster es compatible con todos los entornos.

Además, las MQWDR1 estructuras MQWDR2 y están disponibles para garantizar la compatibilidad con versiones anteriores.

Referencia relacionada

[MQ_CLUSTER_WORKLOAD_EXIT](#) - Descripción de la llamada

El gestor de colas llama a la salida de carga de trabajo del clúster para enviar un mensaje a un gestor de colas disponible.

[MQXCLWLN](#) - Navegar por los registros de carga de trabajo del clúster

La MQXCLWLN llamada se utiliza para navegar por las cadenas de registros MQWDR, MQWQR y MQWCR almacenados en la caché del clúster.

[MQWXP](#) - Estructura del parámetro de salida de la carga de trabajo del clúster

La siguiente tabla resume los campos de la estructura del parámetro de salida de carga de trabajo del clúster MQWXP.

[MQWQR](#) - Estructura de registro de cola de carga de trabajo del clúster

La siguiente tabla resume los campos de la estructura del registro MQWQR - Cola de carga de trabajo del clúster.

[MQWCR](#) - Estructura de registro de clúster de carga de trabajo de clúster

La siguiente tabla resume los campos de la estructura del registro de carga de trabajo del clúster MQWCR.

Campos en MQWDR: estructura del registro de destino de la carga de trabajo del clúster

Descripción de los parámetros en el MQWDR : estructura del registro de destino de la carga de trabajo del clúster.

StrucId (MQCHAR4) - entrada

El identificador de estructura para la estructura del registro de destino de la carga de trabajo del clúster.

- El StrucId valor es MQWDR_STRUC_ID.
- Para el lenguaje de programación C, también se MQWDR_STRUC_ID_ARRAY define la constante. Tiene el mismo valor que MQWDR_STRUC_ID. Es una matriz de caracteres en lugar de una cadena.

Versión (MQLONG) - entrada

El número de versión de la estructura. La versión toma uno de los siguientes valores:

MQWDR_VERSION_1

Version-1 Registro de destino de la carga de trabajo del clúster.

MQWDR_VERSION_2

Version-2 Registro de destino de la carga de trabajo del clúster.

MQWDR_CURRENT_VERSION

Versión actual del registro de destino de la carga de trabajo del clúster.

StrucLength (MQLONG) - entrada

La longitud de la estructura MQWDR. StrucLength toma uno de los siguientes valores:

MQWDR_LENGTH_1

Longitud de un registro de destino de carga de trabajo de clúster de version-1.

MQWDR_LENGTH_2

Longitud de un registro de destino de carga de trabajo de clúster de version-2.

MQWDR_CURRENT_LENGTH

Longitud de la versión actual del registro de destino de la carga de trabajo del clúster.

QMgrFlags (MQLONG) - entrada

Indicadores del gestor de colas que indican las propiedades del gestor de colas que aloja la instancia de la cola de destino descrita por la estructura MQWDR. Se definen los siguientes indicadores:

MQQMF_REPOSITORY_Q_MGR

Destination es un gestor de colas de repositorio completo.

MQQMF_CLUSSDR_USER_DEFINED

El canal del emisor del clúster se definió manualmente.

MQQMF_CLUSSDR_AUTO_DEFINED

El canal del emisor del clúster se definió automáticamente.

MQQMF_AVAILABLE

El gestor de colas de destino está disponible para recibir mensajes.

Otros valores

El gestor de colas puede establecer otras banderas en el campo para fines internos.

QMgrIdentifier (MQCHAR48) - entrada

El identificador del gestor de colas es un identificador único para el gestor de colas que aloja la instancia de la cola de destino descrita por la estructura MQWDR.

- El identificador es generado por el gestor de colas.
- La longitud de QMgrIdentifier es MQ_Q_MGR_IDENTIFIER_LENGTH.

QMgrName (MQCHAR48) - entrada

El nombre del gestor de colas que aloja la instancia de la cola de destino descrita por la estructura MQWDR.

- QMgrName puede ser el nombre del administrador de colas local, así como otro administrador de colas del clúster.
- La longitud de QMgrName es MQ_Q_MGR_NAME_LENGTH.

ClusterRecOffset (MQLONG) - entrada

El desplazamiento lógico de la primera estructura MQWCR que pertenece a la estructura MQWDR.

- Para cachés estáticas, `ClusterRecOffset` es el desplazamiento de la primera estructura `MQWCR` que pertenece a la estructura `MQWDR`.
- El desplazamiento se mide en bytes desde el inicio de la estructura `MQWDR`.
- No utilice el desplazamiento lógico para la aritmética de punteros con cachés dinámicas. Para obtener la dirección del siguiente registro, se debe utilizar la `MQXCLWLN` llamada.

ChannelState (MQLONG) - entrada

El estado del canal que conecta el gestor de colas local con el gestor de colas identificado por la estructura `MQWDR`. Son posibles los siguientes valores:

MQCHS_BINDING

El canal está negociando con el socio.

MQCHS_INACTIVE

El canal no está activo.

MQCHS_INITIALIZING

El canal se está inicializando.

MQCHS_PAUSED

El canal se ha detenido.

MQCHS_REQUESTING

El canal solicitante está solicitando conexión.

MQCHS_RETRYING

El canal está reintentando establecer la conexión.

MQCHS_RUNNING

El canal está transfiriendo o esperando mensajes.

MQCHS_STARTING

El canal está esperando a activarse.

MQCHS_STOPPING

El canal se está deteniendo.

MQCHS_STOPPED

El canal se ha detenido.

ChannelDefOffset (MQLONG) - entrada

El desplazamiento lógico de la definición del canal (`MQCD`) para el canal que conecta el gestor de colas local con el gestor de colas identificado por la estructura `MQWDR`.

- `ChannelDefOffset` es como `ClusterRecOffset`
- El desplazamiento lógico no se puede utilizar en la aritmética de punteros. Para obtener la dirección del siguiente registro, se debe utilizar la llamada `MQXCLWLN`.

DestSeqFactor (MQLONG) - entrada

El factor de secuencia de destino que permite elegir el canal en función del peso.

- `DestSeqFactor` se utiliza antes de que el gestor de colas lo cambie.
- El gestor de carga de trabajo aumenta `DestSeqFactor` de manera que garantiza que los mensajes se distribuyan por los canales según su peso.

DestSeqNumber (MQLONG) - entrada

El valor de destino del canal del clúster antes de que el gestor de colas lo modifique.

- El gestor de carga de trabajo aumenta `DestSeqNumber` cada vez que se envía un mensaje por ese canal.
- Las salidas de carga de trabajo pueden utilizar `DestSeqNumber` para decidir en qué canal colocar un mensaje.

Referencia relacionada

[Valores iniciales y declaraciones lingüísticas para MQWDR](#)

Valores iniciales y declaraciones de lenguaje C y e High Level Assembler para MQWDR : registro de destino de carga de trabajo del clúster.

Valores iniciales y declaraciones lingüísticas para MQWDR

Valores iniciales y declaraciones de lenguaje C y e High Level Assembler para MQWDR : registro de destino de carga de trabajo del clúster.

Tabla 866. Campos en MQWDR		
Nombre de campo	Nombre de la constante	Valor de la constante
<i>StrucId</i>	MQWDR_STRUC_ID	'WDR↵'
<i>Version</i>	MQWDR_VERSION_1	1
<i>StrucLength</i>	MQWDR_CURRENT_LENGTH ³	136
<i>QMgrFlags</i>	MQWDR_NONE	0
<i>QMgrIdentifier</i>	Ninguna	" "
<i>QMgrName</i>	Ninguna	" "
<i>ClusterRecOffset</i>	Ninguna	0
<i>ChannelState</i>	Ninguna	0
<i>ChannelDefOffset</i>	Ninguna	0
<i>DestSeqNumber</i>	Ninguna	0
<i>DestSeqFactor</i>	Ninguna	0

Notas:

1. El símbolo ↵ representa un único carácter en blanco.
2. En el lenguaje de programación C, la variable MQWDR_DEFAULT macro contiene los valores predeterminados. Úselo de la siguiente manera para proporcionar valores iniciales para los campos de la estructura:

```
MQWDR MyWDR = {MQWDR_DEFAULT};
```
3. Los valores iniciales establecen intencionadamente la longitud de la estructura en la longitud de la versión actual, y no en la versión 1 de la estructura.

High Level Assembler

MQWDR	DSECT	
MQWDR_STRUCID	DS CL4	Structure identifier
MQWDR_VERSION	DS F	Structure version number
MQWDR_STRUCLNGTH	DS F	Length of MQWDR structure
MQWDR_QMGRFLAGS	DS F	Queue manager flags
MQWDR_QMGRIDENTIFIER	DS CL48	Queue manager identifier
MQWDR_QMGRNAME	DS CL48	Queue manager name
MQWDR_CLUSTERRECOFFSET	DS F	Offset of first cluster
*		record
MQWDR_CHANNELSTATE	DS F	Channel state
MQWDR_CHANNELDEFOFFSET	DS F	Offset of channel definition
*		structure
MQWDR_LENGTH	EQU *-MQWDR	Length of structure
	ORG MQWDR	
MQWDR_AREA	DS CL(MQWDR_LENGTH)	

Declaración C

```
typedef struct tagMQWDR {
    MQCHAR4 StrucId; /* Structure identifier */
```

```

MQLONG    Version;           /* Structure version number */
MQLONG    StrucLength;       /* Length of MQWDR structure */
MQLONG    QMgrFlags;         /* Queue managerflags */
MQCHAR48   QMgrIdentifier;   /* Queue manageridentifier */
MQCHAR48   QMgrName;         /* Queue manager name */
MQLONG    ClusterRecOffset;  /* Offset of first cluster record */
MQLONG    ChannelState;      /* Channel state */
MQLONG    ChannelDefOffset;  /* Offset of channel definition structure */
/* Ver:1 */
MQLONG    DestSeqNumber;     /* Cluster channel destination sequence number */
MQINT64    DestSeqFactor;    /* Cluster channel factor sequence number */
/* Ver:2 */
};

```

Referencia relacionada

Campos en MQWDR: estructura del registro de destino de la carga de trabajo del clúster

Descripción de los parámetros en el MQWDR : estructura del registro de destino de la carga de trabajo del clúster.

MQWQR - Estructura de registro de cola de carga de trabajo del clúster

La siguiente tabla resume los campos de la estructura del registro MQWQR - Cola de carga de trabajo del clúster.

Tabla 867. Campos en MQWQR		
Campo	Descripción	Página
<i>StrucId</i>	Identificador de la estructura	StrucId
<i>Version</i>	Número de versión de la estructura	Versión
<i>StrucLength</i>	Longitud de la estructura MQWQR	StrucLength
<i>QFlags</i>	Indicadores de cola	QFlags
<i>QName</i>	Nombre de cola	QName
<i>QMgrIdentifier</i>	Identificador del gestor de colas	QMgrIdentifier
<i>ClusterRecOffset</i>	Desplazamiento del primer registro del clúster (MQWCR)	ClusterRecOffset
<i>QType</i>	Tipo de cola	QType
<i>QDesc</i>	Descripción de la cola	QDesc
<i>DefBind</i>	Enlace predeterminado	DefBind
<i>DefPersistence</i>	Persistencia de mensajes predeterminada	DefPersistence
<i>DefPriority</i>	Prioridad de mensajes predeterminada	DefPriority
<i>InhibitPut</i>	Si se permiten las operaciones en la cola	InhibitPut
Nota: Los campos restantes se ignoran si la versión es inferior a MQWQR_VERSION_2.		
<i>CWLQueuePriority</i>	Un valor del 0 al 9 que representa la prioridad de la cola	CLWLQueuePriority
<i>CLWLQueueRank</i>	Un valor del 0 al 9 que representa el rango de la cola	CLWLQueueRank
Nota: Los campos restantes se ignoran si la versión es inferior a MQWQR_VERSION_3.		
<i>DefPutResponse</i>	Resp predet de transferencia	DefPutResponse

La estructura del registro de la cola de carga de trabajo del clúster contiene información relacionada con uno de los posibles destinos del mensaje. Hay una estructura de registro de cola de carga de trabajo de clúster para cada instancia de la cola de destino.

La estructura de registros de la cola de carga de trabajo del clúster es compatible con todos los entornos. Además, las MQWQR1 estructuras MQWQR2 y están disponibles para garantizar la compatibilidad con versiones anteriores.

Referencia relacionada

MQ_CLUSTER_WORKLOAD_EXIT - Descripción de la llamada

El gestor de colas llama a la salida de carga de trabajo del clúster para enviar un mensaje a un gestor de colas disponible.

MQXCLWLN - Navegar por los registros de carga de trabajo del clúster

La MQXCLWLN llamada se utiliza para navegar por las cadenas de registros MQWDR, MQWQR y MQWCR almacenados en la caché del clúster.

MQWXP - Estructura del parámetro de salida de la carga de trabajo del clúster

La siguiente tabla resume los campos de la estructura del parámetro de salida de carga de trabajo del clúster MQWXP.

MQWDR: estructura de registro de destino de carga de trabajo del clúster

La siguiente tabla resume los campos de la estructura del registro de destino de la carga de trabajo del clúster MQWDR.

MQWCR - Estructura de registro de clúster de carga de trabajo de clúster

La siguiente tabla resume los campos de la estructura del registro de carga de trabajo del clúster MQWCR.

Campos en MQWQR : estructura del registro de la cola de carga de trabajo del clúster

Descripción de los campos en el MQWQR : estructura del registro de la cola de carga de trabajo del clúster.

StrucId (MQCHAR4) - entrada

El identificador de estructura para la estructura de registro de la cola de carga de trabajo del clúster.

- El StrucId valor es MQWQR_STRUC_ID.
- Para el lenguaje de programación C, también se MQWQR_STRUC_ID_ARRAY define la constante. Tiene el mismo valor que MQWQR_STRUC_ID. Es una matriz de caracteres en lugar de una cadena.

Versión (MQLONG) - entrada

El número de versión de la estructura. La versión toma uno de los siguientes valores:

MQWQR_VERSION_1

Version-1 Registro de la cola de carga de trabajo del clúster.

MQWQR_VERSION_2

Version-2 Registro de la cola de carga de trabajo del clúster.

MQWQR_VERSION_3

Version-3 Registro de la cola de carga de trabajo del clúster.

MQWQR_CURRENT_VERSION

Versión actual del registro de la cola de carga de trabajo del clúster.

StrucLength (MQLONG) - entrada

La longitud de la estructura MQWQR. StrucLength toma uno de los siguientes valores:

MQWQR_LENGTH_1

Longitud de un registro de la cola de carga de trabajo de un clúster de version-1.

MQWQR_LENGTH_2

Longitud de un registro de la cola de carga de trabajo de un clúster de version-2.

MQWQR_LENGTH_3

Longitud de un registro de la cola de carga de trabajo de un clúster de version-3.

MQWQR_CURRENT_LENGTH

Longitud de la versión actual del registro de la cola de carga de trabajo del clúster.

QFlags (MQLONG) - entrada

Los indicadores de cola indican las propiedades de la cola. Se definen los siguientes indicadores:

MQQF_LOCAL_Q

El destino es una cola local.

MQQF_CLWL_USEQ_ANY

Permitir el uso de colas locales y remotas en los puts.

MQQF_CLWL_USEQ_LOCAL

Permitir solo entradas en la cola local.

Otros valores

El gestor de colas puede establecer otras banderas en el campo para fines internos.

QName (MQCHAR48) - entrada

El nombre de la cola que es uno de los posibles destinos del mensaje.

- La longitud de QName es MQ_Q_NAME_LENGTH.

QMgrIdentifier (MQCHAR48) - entrada

El identificador del gestor de colas es un identificador único para el gestor de colas que aloja la instancia de la cola descrita por la estructura MQWQR.

- El identificador es generado por el gestor de colas.
- La longitud de QMgrIdentifier es MQ_Q_MGR_IDENTIFIER_LENGTH.

ClusterRecOffset (MQLONG) - entrada

El desplazamiento lógico de la primera estructura MQWCR que pertenece a la estructura MQWQR.

- Para cachés estáticas, ClusterRecOffset es el desplazamiento de la primera estructura MQWCR que pertenece a la estructura MQWQR.
- El desplazamiento se mide en bytes desde el inicio de la estructura MQWQR.
- No utilice el desplazamiento lógico para la aritmética de punteros con cachés dinámicas. Para obtener la dirección del siguiente registro, se debe utilizar la MQXCLWLN llamada.

QType (MQLONG) - entrada

El tipo de cola de la cola de destino. Son posibles los siguientes valores:

MQCQT_LOCAL_Q

Cola local.

MQCQT_ALIAS_Q

Cola alias.

MQCQT_REMOTE_Q

Cola remota.

MQCQT_Q_MGR_ALIAS

Alias del gestor de colas.

QDesc (MQCHAR64) - entrada

El atributo de descripción de cola definido en el gestor de colas que aloja la instancia de la cola de destino descrita por la estructura MQWQR.

- La longitud de QDesc es MQ_Q_DESC_LENGTH.

DefBind (MQLONG) - entrada

El atributo de cola de enlace predeterminado definido en el gestor de colas que aloja la instancia de la cola de destino descrita por la estructura MQWQR. Cuando se utilicen grupos con clústeres, es necesario especificar «o MQBND_BIND_ON_OPEN bien» o MQBND_BIND_ON_GROUP «o». Los siguientes valores son posibles:

MQBND_BIND_ON_OPEN

Vinculación fijada por MQOPEN llamada.

MQBND_BIND_NOT_FIXED

Encuadernación no fija.

MQBND_BIND_ON_GROUP

Permite a una aplicación solicitar que un grupo de mensajes se asigne en su totalidad a la misma instancia de destino.

DefPersistence (MQLONG) - entrada

El atributo de persistencia de mensajes predeterminado definido en el gestor de colas que aloja la instancia de la cola de destino descrita por la estructura MQWQR. Son posibles los siguientes valores:

MQPER_PERSISTENT

El mensaje es persistente.

MQPER_NOT_PERSISTENT

El mensaje no es persistente.

DefPriority (MQLONG) - entrada

El atributo de prioridad de mensajes predeterminado definido en el gestor de colas que aloja la instancia de la cola de destino descrita por la estructura MQWQR. El rango de prioridad es 0 - MaxPriority.

- 0 es la prioridad más baja.
- MaxPriority es el atributo del gestor de colas que aloja esta instancia de la cola de destino.

InhibitPut (MQLONG) - entrada

El atributo de cola inhibida definido en el gestor de colas que aloja la instancia de la cola de destino descrita por la estructura MQWQR. Son posibles los siguientes valores:

MQQA_PUT_INHIBITED

Las operaciones de inserción están inhibidas.

MQQA_PUT_ALLOWED

Se permiten operaciones de inserción.

CLWLQueuePriority (MQLONG) - entrada

El atributo de prioridad de la cola de carga de trabajo del clúster definido en el gestor de colas que aloja la instancia de la cola de destino descrita por la estructura MQWQR.

CLWLQueueRank (MQLONG) - entrada

El rango de la cola de carga de trabajo del clúster definido en el gestor de colas que aloja la instancia de la cola de destino descrita por la estructura MQWQR.

DefPutResponse (MQLONG) - entrada

El atributo de cola de respuesta de envío predeterminado definido en el gestor de colas que aloja la instancia de la cola de destino descrita por la estructura MQWQR. Son posibles los siguientes valores:

MQPRT_SYNC_RESPONSE

Respuesta sincrónica a MQPUT o MQPUT1 llamadas.

MQPRT_ASYNC_RESPONSE

Respuesta asíncrona a MQPUT o MQPUT1 llamadas.

Referencia relacionada

Valores iniciales y declaraciones de idioma para MQWQR

Valores iniciales y declaraciones de lenguaje C y e High Level Assembler para MQWQR : registro de cola de carga de trabajo del clúster.

Valores iniciales y declaraciones de idioma para MQWQR

Valores iniciales y declaraciones de lenguaje C y e High Level Assembler para MQWQR : registro de cola de carga de trabajo del clúster.

Tabla 868. Campos en MQWQR		
Nombre de campo	Nombre de la constante	Valor de la constante
StrucId	MQWQR_STRUC_ID_ARRAY	'WQR→'
Version	MQWQR_VERSION_1	1

Tabla 868. Campos en MQWQR (continuación)

Nombre de campo	Nombre de la constante	Valor de la constante
<i>StrucLength</i>	MQWQR_CURRENT_LENGTH ³	212
<i>QFlags</i>	Ninguna	0
<i>QName</i>	Ninguna	" "
<i>QMgrIdentifier</i>	Ninguna	" "
<i>ClusterRecOffset</i>	Ninguna	0
<i>QType</i>	Ninguna	0
<i>QDesc</i>	Ninguna	" "
<i>DefBind</i>	Ninguna	0
<i>DefPersistence</i>	Ninguna	0
<i>DefPriority</i>	Ninguna	0
<i>InhibitPut</i>	Ninguna	0
<i>CLWLQueuePriority</i>	Ninguna	0
<i>CLWLQueueRank</i>	Ninguna	0
<i>DefPutResponse</i>	Ninguna	1

Notas:

1. El símbolo `↵` representa un único carácter en blanco.
2. En el lenguaje de programación C, la variable `MQWQR_DEFAULT` macro contiene los valores predeterminados. Úselo de la siguiente manera para proporcionar valores iniciales para los campos de la estructura:

```
MQWQR MyWQR = {MQWQR_DEFAULT};
```

3. Los valores iniciales establecen intencionadamente la longitud de la estructura en la longitud de la versión actual, y no en la versión 1 de la estructura.

Declaración C

```
typedef struct tagMQWQR {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQLONG     StrucLength;       /* Length of MQWQR structure */
    MQLONG     QFlags;           /* Queue flags */
    MQCHAR48    QName;           /* Queue name */
    MQCHAR48    QMgrIdentifier;   /* Queue manager identifier */
    MQLONG     ClusterRecOffset;  /* Offset of first cluster record */
    MQLONG     QType;            /* Queue type */
    MQCHAR64    QDesc;           /* Queue description */
    MQLONG     DefBind;          /* Default binding */
    MQLONG     DefPersistence;    /* Default message persistence */
    MQLONG     DefPriority;       /* Default message priority */
    MQLONG     InhibitPut;       /* Whether put operations on the queue
                                are allowed */

    /* version 2 */
    MQLONG     CLWLQueuePriority; /* Queue priority */
    MQLONG     CLWLQueueRank;    /* Queue rank */
    /* version 3 */
    MQLONG     DefPutResponse;    /* Default put response */
};
```

High Level Assembler

MQWQR	DSECT		
MQWQR_STRUCID	DS	CL4	Structure identifier
MQWQR_VERSION	DS	F	Structure version number
MQWQR_STRUCLength	DS	F	Length of MQWQR structure
MQWQR_QFLAGS	DS	F	Queue flags
MQWQR_QNAME	DS	CL48	Queue name
MQWQR_QMGRIDENTIFIER	DS	CL48	Queue manager identifier
MQWQR_CLUSTERRECOFFSET	DS	F	Offset of first cluster record
*			
MQWQR_QTYPE	DS	F	Queue type
MQWQR_QDESC	DS	CL64	Queue description
MQWQR_DEFBIND	DS	F	Default binding
MQWQR_DEFPERSISTENCE	DS	F	Default message persistence
MQWQR_DEFPRIORITY	DS	F	Default message priority
MQWQR_INHIBITPUT	DS	F	Whether put operations on the queue are allowed
*			
MQWQR_DEFPUTRESPONSE	DS	F	Default put response
MQWQR_LENGTH	EQU	*-MQWQR	Length of structure
	ORG	MQWQR	
MQWQR_AREA	DS	CL(MQWQR_LENGTH)	

Referencia relacionada

[Campos en MQWQR : estructura del registro de la cola de carga de trabajo del clúster](#)

Descripción de los campos en el MQWQR : estructura del registro de la cola de carga de trabajo del clúster.

MQWCR - Estructura de registro de clúster de carga de trabajo de clúster

La siguiente tabla resume los campos de la estructura del registro de carga de trabajo del clúster MQWCR.

Tabla 869. Campos en MQWCR		
Campo	Descripción	Página
<i>ClusterName</i>	Nombre del clúster	ClusterName
<i>ClusterRecOffset</i>	Desplazamiento del siguiente registro de clúster (MQWCR)	ClusterRecOffset
<i>ClusterFlags</i>	Indicadores de clúster	ClusterFlags

La estructura del registro de carga de trabajo del clúster contiene información sobre un clúster. Por cada clúster al que pertenece la cola de destino, hay una estructura de registro de clúster de carga de trabajo de clúster.

La estructura de registros de carga de trabajo del clúster es compatible con todos los entornos.

Referencia relacionada

[MQ_CLUSTER_WORKLOAD_EXIT - Descripción de la llamada](#)

El gestor de colas llama a la salida de carga de trabajo del clúster para enviar un mensaje a un gestor de colas disponible.

[MQXCLWLN - Navegar por los registros de carga de trabajo del clúster](#)

La MQXCLWLN llamada se utiliza para navegar por las cadenas de registros MQWDR, MQWQR y MQWCR almacenados en la caché del clúster.

[MQWXP - Estructura del parámetro de salida de la carga de trabajo del clúster](#)

La siguiente tabla resume los campos de la estructura del parámetro de salida de carga de trabajo del clúster MQWXP.

[MQWDR: estructura de registro de destino de carga de trabajo del clúster](#)

La siguiente tabla resume los campos de la estructura del registro de destino de la carga de trabajo del clúster MQWDR.

[MQWQR - Estructura de registro de cola de carga de trabajo del clúster](#)

La siguiente tabla resume los campos de la estructura del registro MQWQR - Cola de carga de trabajo del clúster.

Campos en el MQWCR : estructura del registro del clúster de carga de trabajo del clúster.

Descripción de los campos en el MQWCR : estructura del registro del clúster de carga de trabajo del clúster.

ClusterName (MQCHAR48) - entrada

El nombre de un clúster al que pertenece la instancia de la cola de destino que posee la estructura MQWCR. La instancia de cola de destino se describe mediante una estructura MQWDR.

- La longitud de ClusterName es MQ_CLUSTER_NAME_LENGTH.

ClusterRecOffset (MQLONG) - entrada

El desplazamiento lógico de la siguiente estructura MQWCR.

- Si no hay más estructuras MQWCR, ClusterRecOffset es cero.
- El desplazamiento se mide en bytes desde el inicio de la estructura MQWCR.

ClusterFlags (MQLONG) - entrada

Los indicadores de clúster indican las propiedades del gestor de colas identificado por la estructura MQWCR. Se definen los siguientes indicadores:

MQQMF_REPOSITORY_Q_MGR

Destination es un gestor de colas de repositorio completo.

MQQMF_CLUSSDR_USER_DEFINED

El canal del emisor del clúster se definió manualmente.

MQQMF_CLUSSDR_AUTO_DEFINED

El canal del emisor del clúster se definió automáticamente.

MQQMF_AVAILABLE

El gestor de colas de destino está disponible para recibir mensajes.

Otros valores

El gestor de colas puede establecer otras banderas en el campo para fines internos.

Referencia relacionada

Valores iniciales y declaraciones de idioma para MQWCR

Valores iniciales y declaraciones de lenguaje C y High Level Assembler para MQWCR : estructura de registro de clúster de carga de trabajo de clúster.

Valores iniciales y declaraciones de idioma para MQWCR

Valores iniciales y declaraciones de lenguaje C y High Level Assembler para MQWCR : estructura de registro de clúster de carga de trabajo de clúster.

Tabla 870. Campos en MQWCR		
Nombre de campo	Nombre de la constante	Valor de la constante
ClusterName	Ninguna	" "
ClusterRecOffset	Ninguna	0
ClusterFlags	Ninguna	0

Declaración C

```
typedef struct tagMQWCR {
    MQCHAR48 ClusterName;      /* Cluster name */
    MQLONG ClusterRecOffset;   /* Offset of next cluster record */
    MQLONG ClusterFlags;       /* Cluster flags */
};
```

High Level Assembler

MQWCR	DSECT	
MQWCR_CLUSTERNAME	DS	CL48 Cluster name
MQWCR_CLUSTERRECOFFSET	DS	F Offset of next cluster

*				record
MQWCR_CLUSTERFLAGS	DS	F		Cluster flags
MQWCR_LENGTH	EQU	*-MQWCR		Length of structure
	ORG	MQWCR		
MQWCR_AREA	DS	CL(MQWCR_LENGTH)		

Referencia relacionada

Campos en el MQWCR : estructura del registro del clúster de carga de trabajo del clúster.

Descripción de los campos en el MQWCR : estructura del registro del clúster de carga de trabajo del clúster.

Referencia a la salida de la API

Esta sección proporciona información de referencia principalmente de interés para un programador que escribe salidas API.

Notas generales de uso

notas:

1. Todas las funciones de salida pueden emitir la llamada MQXEP; esta llamada está diseñada específicamente para su uso desde funciones de salida API.
2. La función MQ_INIT_EXIT no puede emitir ninguna llamada e MQ, salvo MQXEP.
3. No se puede emitir la llamada MQDISC para la conexión actual.
4. Si una función de salida emite la llamada MQCONN o la llamada MQCONNEX con la opción MQCNO_HANDLE_SHARE_NONE, la llamada se completa con el código de motivo MQRC_ALREADY_CONNECTED y el identificador devuelto es el mismo que se pasó a la salida como parámetro.
5. En general, cuando una función de salida de la API emite una llamada `MQI`, las salidas de la API no se invocan de forma recursiva. Sin embargo, si una función de salida emite la llamada MQCONNEX con las opciones MQCNO_HANDLE_SHARE_BLOCK o MQCNO_HANDLE_SHARE_NO_BLOCK, la llamada devuelve un nuevo identificador compartido. Esto proporciona al conjunto de salida un identificador de conexión propio y, por lo tanto, una unidad de trabajo independiente de la unidad de trabajo de la aplicación. La suite de salida puede utilizar este identificador para enviar y recibir mensajes dentro de su propia unidad de trabajo, y confirmar o revertir dicha unidad de trabajo; todo ello sin afectar en modo alguno a la unidad de trabajo de la aplicación.

Dado que la función de salida utiliza un identificador de conexión diferente al que utiliza la aplicación, las llamadas a MQ emitidas por la función de salida dan lugar a la invocación de las funciones de salida de la API correspondientes. Por lo tanto, las funciones de salida pueden invocarse de forma recursiva. Tenga en cuenta que tanto el *ExitUserArea* campo en MQAXP como el área de la cadena de salida tienen un ámbito de control de conexión. Por consiguiente, una función de salida no puede utilizar esas áreas para indicar a otra instancia de sí misma invocada recursivamente que ya está activa.

6. Las funciones de salida también pueden enviar y recibir mensajes dentro de la unidad de trabajo de la aplicación. Cuando la aplicación confirma o cancela la unidad de trabajo, todos los mensajes dentro de la unidad de trabajo se confirman o cancelan juntos, independientemente de quién los haya colocado en la unidad de trabajo (aplicación o función de salida). Sin embargo, la salida puede provocar que la aplicación supere los límites del sistema antes de lo que lo haría en otras circunstancias (por ejemplo, al superar el número máximo de mensajes no confirmados en una unidad de trabajo).

Cuando una función de salida utiliza la unidad de trabajo de la aplicación de esta manera, la función de salida normalmente debe evitar emitir la llamada MQCMIT, ya que esto confirma la unidad de trabajo de la aplicación y podría afectar al correcto funcionamiento de la aplicación. Sin embargo, es posible que la función de salida tenga que emitir la llamada MQBACK en algunos casos, si la función de salida encuentra un error grave que impide que se confirme la unidad de trabajo (por ejemplo, un error al incluir un mensaje como parte de la unidad de trabajo de la aplicación). Cuando se llame a MQBACK, asegúrese de que no se modifiquen los límites de la unidad de trabajo de la aplicación. En esta situación, la función de salida debe establecer los valores adecuados para

garantizar que se devuelvan a la aplicación el código de finalización MQCC_WARNING y el código de motivo MQRC_BACKED_OUT, de modo que la aplicación pueda detectar que se ha revertido la unidad de trabajo.

Si una función de salida utiliza el identificador de conexión de la aplicación para emitir llamadas de e MQ, dichas llamadas no dan lugar a nuevas invocaciones de funciones de salida de la API.

7. Si una función de salida MQXR_BEFORE finaliza de forma anómala, es posible que el gestor de colas pueda recuperarse del fallo. Si es posible, el gestor de colas continúa el procesamiento como si la función de salida hubiera devuelto MQXCC_FAILED. Si el gestor de colas no puede recuperarse, la aplicación se cierra.
8. Si una función de salida MQXR_AFTER finaliza de forma anómala, es posible que el gestor de colas pueda recuperarse del fallo. Si es posible, el gestor de colas continúa el procesamiento como si la función de salida hubiera devuelto MQXCC_FAILED. Si el gestor de colas no puede recuperarse, la aplicación se cierra. Tenga en cuenta que, en este último caso, los mensajes recuperados fuera de una unidad de trabajo se pierden (esta es la misma situación que cuando la aplicación falla inmediatamente después de eliminar un mensaje de la cola).
9. El proceso MCA realiza una confirmación en dos fases.

Si una salida API intercepta un MQCMIT de un proceso MCA preparado e intenta realizar una acción dentro de la unidad de trabajo, la acción fallará con el código de motivo MQRC_UOW_NOT_AVAILABLE.

10. Cuando haya varias instalaciones IBM MQ de disponibles, utilice las salidas escritas para una versión anterior de IBM MQ, ya que es posible que las nuevas funciones añadidas en la versión posterior no funcionen con las versiones anteriores. Para obtener más información sobre los cambios entre versiones, consulte [Qué ha cambiado en IBM MQ 8.0](#).

Estructura de parámetros de salida de API de IBM MQ (MQAXP)

La estructura MQAXP, un bloque de control externo, se utiliza como parámetro de entrada o salida para la salida de API. Este tema también proporciona información sobre cómo los gestores de colas procesan las funciones de salida.

MQAXP tiene la siguiente declaración C:

```
typedef struct tagMQAXP {
    MQCHAR4   StrucId;           /* Structure identifier */
    MQLONG    Version;          /* Structure version number */
    MQLONG    ExitId;           /* Exit Identifier */
    MQLONG    ExitReason;       /* Exit invocation reason */
    MQLONG    ExitResponse;     /* Response code from exit */
    MQLONG    ExitResponse2;    /* Secondary response code from exit */
    MQLONG    Feedback;        /* Feedback code from exit */
    MQLONG    APICallerType;    /* MQSeries API caller type */
    MQBYTE16  ExitUserArea;     /* User area for use by exit */
    MQCHAR32  ExitData;        /* Exit data area */
    MQCHAR48  ExitInfoName;     /* Exit information name */
    MQBYTE48  ExitPDArea;      /* Problem determination area */
    MQCHAR48  QMgrName;        /* Name of local queue manager */
    PMQACH    ExitChainAreaPtr; /* Inter exit communication area */
    MQHCONFIG Hconfig;         /* Configuration handle */
    MQLONG    Function;        /* Function Identifier */
    /* Ver:1 */
    MQHMSG    ExitMsgHandle     /* Exit message handle */
    /* Ver:2 */
};
```

La siguiente lista de parámetros se pasa cuando se invocan las funciones de una salida de API:

StrucId (MQCHAR4)-entrada

El identificador de estructura de parámetro de salida, con un valor de:

```
MQAXP_STRUC_ID.
```

El manejador de salida establece este campo en la entrada para cada función de salida.

Versión (MQLONG)-entrada

El número de versión de la estructura, con un valor de:

MQAXP_VERSION_1

Estructura de parámetros de salida de API de la versión 1.

MQAXP_VERSION_2

Estructura de parámetros de salida de API de la versión 2.

VERSIÓN_ACTUAL_MQAXP

Número de versión actual para la estructura de parámetros de salida de API.

El manejador de salida establece este campo en la entrada para cada función de salida.

ExitId (MQLONG)-entrada

El identificador de salida, establecido en la entrada de la rutina de salida, que indica el tipo de salida:

MQXT_API_EXIT

Salida de API.

ExitReason (MQLONG)-entrada

La razón por la que se invoca la salida, establecida en la entrada de cada función de salida:

MQXR_CONNECTION

La salida se invoca para inicializarse a sí misma antes de una llamada MQCONN o MQCONNEX, o para finalizar a sí misma después de una llamada MQDISC.

MQXR_ANTES

La salida se invoca antes de ejecutar una llamada de API, o antes de convertir datos en un MQGET.

MQXR_DESPUÉS

La salida se está invocando después de ejecutar una llamada de API.

ExitResponse (MQLONG)-salida

La respuesta de la salida, inicializada al entrar en cada función de salida para:

MQXCC_Correcto

Continúe con normalidad.

La función de salida debe establecer este campo para comunicar al gestor de colas el resultado de la ejecución de la función de salida. El valor debe ser uno de los siguientes:

MQXCC_Correcto

La función de salida se ha completado satisfactoriamente. Continúe con normalidad.

Este valor lo pueden establecer todas las funciones de salida MQXR_*. ExitResponse2 se utiliza para decidir si se deben invocar las funciones de salida más adelante en la cadena.

MQXCC_FALLO

La función de salida ha fallado debido a un error.

Este valor lo pueden establecer todas las funciones de salida MQXR_*. El gestor de colas establece CompCode en MQCC_FAILED y Reason en:

- MQRC_API_EXIT_INIT_ERROR si la función es MQ_INIT_EXIT
- MQRC_API_EXIT_TERM_ERROR si la función es MQ_TERM_EXIT
- MQRC_API_EXIT_ERROR para todas las demás funciones de salida

Los valores establecidos pueden ser alterados por una función de salida más adelante en la cadena.

ExitResponse2 se ignora; el gestor de colas continúa el proceso como si se hubiera devuelto MQXR2_SUPPRESS_CHAIN.

MQXCC_SUPPRESS_FUNCTION

Suprimir la función de API de IBM MQ.

Este valor sólo lo puede establecer una función de salida MQXR_BEFORE. Omite la llamada de API. Si lo devuelve MQ_DATA_CONV_ON_GET_EXIT, se omite la conversión de datos. El gestor de

colas establece CompCode en MQCC_FAILED, y Reason en MQRC_SUPPRESSED_BY_EXIT, pero los valores establecidos se pueden modificar mediante una función de salida más adelante en la cadena. Otros parámetros para la llamada permanecen cuando la salida los deja. ExitResponse2 se utiliza para decidir si se deben invocar las funciones de salida más adelante en la cadena.

Si este valor lo establece una función de salida MQXR_AFTER o MQXR_CONNECTION, el gestor de colas continúa el proceso como si se hubiera devuelto MQXCC_FAILED.

MQXCC_OMITIR_FUNCIÓN

Omita la función de API de IBM MQ .

Este valor sólo lo puede establecer una función de salida MQXR_BEFORE. Omita la llamada de API. Si lo devuelve MQ_DATA_CONV_ON_GET_EXIT, se omite la conversión de datos. La función de salida debe establecer CompCode y Reason en los valores que se van a devolver a la aplicación, pero los valores establecidos pueden ser alterados por una función de salida más adelante en la cadena. Otros parámetros para la llamada permanecen cuando la salida los deja. ExitResponse2 se utiliza para decidir si se deben invocar las funciones de salida más adelante en la cadena.

Si este valor lo establece una función de salida MQXR_AFTER o MQXR_CONNECTION, el gestor de colas continúa el proceso como si se hubiera devuelto MQXCC_FAILED.

MQXCC_SUPPRESS_EXIT

Suprimir todas las funciones de salida que pertenecen al conjunto de salidas.

Este valor sólo lo pueden establecer las funciones de salida MQXR_BEFORE y MQXR_AFTER. Omita *todas* las invocaciones posteriores de las funciones de salida que pertenecen a este conjunto de salidas para esta conexión lógica. Esta omisión continúa hasta que se produce la solicitud de desconexión lógica, cuando se invoca la función MQ_TERM_EXIT con una ExitReason de MQXR_CONNECTION.

La función de salida debe establecer CompCode y Reason en los valores que se van a devolver a la aplicación, pero los valores establecidos pueden ser alterados por una función de salida más adelante en la cadena. Otros parámetros para la llamada permanecen cuando la salida los deja. ExitResponse2 se ignora.

Si este valor lo establece una función de salida MQXR_CONNECTION, el gestor de colas continúa el proceso como si se hubiera devuelto MQXCC_FAILED.

Para obtener información sobre la interacción entre ExitResponse y ExitResponse2, y su efecto en el proceso de salida, consulte [“Cómo procesan los gestores de colas las funciones de salida”](#) en la página 1592.

ExitResponse2 (MQLONG)-salida

Este es un código de respuesta de salida secundario que califica el código de respuesta de salida primario para las funciones de salida MQXR_BEFORE. Se inicializa para:

```
MQXR2_DEFAULT_CONTINUATION
```

en la entrada a una función de salida de llamada de API de IBM MQ . A continuación, se puede establecer en uno de los valores:

MQXR2_DEFAULT_CONTINUATION

Indica si se debe continuar con la siguiente salida de la cadena, en función del valor de ExitResponse.

Si ExitResponse es MQXCC_SUPPRESS_FUNCTION o MQXCC_SKIP_FUNCTION, omita las funciones de salida más adelante en la cadena MQXR_BEFORE y las funciones de salida coincidentes en la cadena MQXR_AFTER. Invoque las funciones de salida de la cadena MQXR_AFTER que coincidan con las funciones de salida anteriores de la cadena MQXR_BEFORE.

De lo contrario, invoque la siguiente salida de la cadena.

MQXR2_SUPPRESS_CHAIN

Suprima la cadena.

Omita las funciones de salida más adelante en la cadena MQXR_BEFORE y las funciones de salida coincidentes en la cadena MQXR_AFTER para esta invocación de llamada de API. Invoque las funciones de salida de la cadena MQXR_AFTER que coincidan con las funciones de salida anteriores de la cadena MQXR_BEFORE.

MQXR2_CONTINUE_CHAIN

Continúe con la siguiente salida de la cadena.

Para obtener información sobre la interacción entre ExitResponse y ExitResponse2, y su efecto en el proceso de salida, consulte [“Cómo procesan los gestores de colas las funciones de salida” en la página 1592.](#)

Comentarios (MQLONG)-entrada/salida

Comunicar códigos de retroalimentación entre invocaciones de función de salida. Se inicializa para:

```
MQFB_NONE (0)
```

antes de invocar la primera función de la primera salida de una cadena.

Las salidas pueden establecer este campo en cualquier valor, incluido cualquier valor MQFB_* o MQRC_* válido. Las salidas también pueden establecer este campo en un valor de comentarios definido por el usuario en el rango MQFB_APPL_FIRST a MQFB_APPL_LAST.

APICallerType (MQLONG)-entrada

El tipo de interlocutor de API, que indica si el interlocutor de API de IBM MQ es externo o interno del gestor de colas: MQXACT_EXTERNAL o MQXACT_INTERNAL.

ExitUserAreaMQBYTE16) - entrada/salida

Un área de usuario, disponible para todas las salidas asociadas a un ExitInfoObject concreto. Se inicializa a MQXUA_NONE (ceros binarios para la longitud del ExitUserArea) antes de invocar la primera función de salida (MQ_INIT_EXIT) para la interconexión. A partir de entonces, los cambios realizados en este campo por una función de salida se conservan entre invocaciones de funciones de la misma salida.

Este campo está alineado con un múltiplo de 4 MQLONG.

Las salidas también pueden anclar cualquier almacenamiento que asignen desde esta área.

Para cada hconn, cada salida en una cadena de salidas tiene un ExitUserArea diferente. El ExitUserArea no puede ser compartido por las salidas de una cadena, y el contenido del ExitUserArea de una salida no está disponible para otra salida de una cadena.

Para programas C, la constante MQXUA_NONE_ARRAY también se define con el mismo valor que MQXUA_NONE, pero como una matriz de caracteres en lugar de una serie.

La longitud de este campo la proporciona MQ_EXIT_USER_AREA_LENGTH.

ExitData (MQCHAR32)-entrada

Datos de salida, establecidos en la entrada de cada función de salida en los 32 caracteres de datos específicos de salida que se proporcionan en la salida. Si no define ningún valor en la salida, este campo estará en blanco.

La longitud de este campo la proporciona MQ_EXIT_DATA_LENGTH.

ExitInfoNameMQCHAR48) - entrada

El nombre de la información de salida, establecido en la entrada de cada función de salida al ApiExit_name especificado en las definiciones de salida en las estrofas.

ExitPDArea (MQBYTE48)-entrada/salida

Un área de determinación de problemas, inicializada en MQXPDA_NONE (ceros binarios para la longitud del campo) para cada invocación de una función de salida.

Para programas C, la constante MQXPDA_NONE_ARRAY también se define con el mismo valor que MQXPDA_NONE, pero como una matriz de caracteres en lugar de una serie.

El manejador de salida siempre escribe esta área en el rastreo de IBM MQ al final de una salida, incluso cuando la función es satisfactoria.

La longitud de este campo la proporciona MQ_EXIT_PD_AREA_LENGTH.

QMgrName (MQCHAR48)-entrada

El nombre del gestor de colas al que está conectada la aplicación, que ha invocado una salida como resultado del proceso de una llamada de API de IBM MQ.

Si el nombre de un gestor de colas proporcionado en una llamada MQCONN o MQCONNEX está en blanco, este campo sigue establecido en el nombre del gestor de colas al que está conectada la aplicación, tanto si la aplicación es servidor como si es cliente.

El manejador de salida establece este campo en la entrada para cada función de salida.

La longitud de este campo la proporciona MQ_Q_MGR_NAME_LENGTH.

ExitChainAreaPtr (PMQACH) - entrada/salida

Se utiliza para comunicar datos entre invocaciones de distintas salidas de una cadena. Se establece en un puntero NULL antes de invocar la primera función (MQ_INIT_EXIT con ExitReason MQXR_CONNECTION) de la primera salida de una cadena de salidas. El valor devuelto por la salida en una invocación se pasa a la siguiente invocación.

Consulte [“El área de la cadena de salida y el encabezado del área de la cadena de salida \(MQACH\)”](#) en la [página 1596](#) para obtener más detalles sobre cómo utilizar el área de cadena de salida.

Hconfig (MQHCONFIG)-entrada

El descriptor de contexto de configuración, que representa el conjunto de funciones que se están inicializando. Este valor lo genera el gestor de colas en la función MQ_INIT_EXIT y se pasa posteriormente a la función de salida de API. Se establece en la entrada para cada función de salida.

Puedes utilizar Hconfig como puntero a la estructura MQIEP para realizar llamadas de tipo « MQI » y « DCI ». Debe comprobar que los primeros 4 bytes de HConfig coincidan con el campo « StrucId » de la estructura MQIEP antes de utilizar el parámetro HConfig como puntero a dicha estructura.

Función (MQLONG)-entrada

El identificador de función, cuyos valores válidos son las constantes MQXF_* descritas en [“Constantes externas”](#) en la [página 1597](#).

El manejador de salida establece este campo en el valor correcto, al entrar en cada función de salida, en función de la llamada de API IBM MQ que ha dado como resultado la invocación de la salida.

ExitMsgHandle (MQHMSG) - entrada/salida

Cuando la función es MQXF_GET y ExitReason es MQXR_AFTER, se devuelve un descriptor de mensaje válido en este campo que permite a la salida de API acceder a los campos del descriptor de mensaje y a cualquier otra propiedad que coincida con la serie ExitProperties especificada en la estructura MQXEPO al registrar la salida de API.

Cualquier propiedad no descriptor de mensaje que se devuelva en el ExitMsgHandle no estará disponible desde el MsgHandle en la estructura MQGMO si se especificó una, o en los datos del mensaje.

Cuando Function es MQXF_GET y ExitReason es MQXR_BEFORE, si el programa de salida establece este campo como MQHM_NONE, suprimirá el relleno de las propiedades ExitMsgHandle.

Este campo no se establece si la versión es menor que MQAXP_VERSION_2.

Cómo procesan los gestores de colas las funciones de salida

El proceso realizado por el gestor de colas al devolver una función de salida depende de ExitResponse y de ExitResponse2.

[Tabla 871](#) en la [página 1593](#) resume las combinaciones posibles y sus efectos para una función de salida MQXR_BEFORE, mostrando:

- Quién establece los parámetros CompCode y Reason de la llamada de API
- Si se invocan las funciones de salida restantes de la cadena MQXR_BEFORE y las funciones de salida coincidentes de la cadena MQXR_AFTER

- Si se invoca la llamada de API

Para una función de salida MQXR_AFTER:

- CompCode y Reason se establecen de la misma forma que MQXR_BEFORE
- ExitResponse2 se ignora (las funciones de salida restantes de la cadena MQXR_AFTER siempre se invocan)
- MQXCC_SUPPRESS_FUNCTION y MQXCC_SKIP_FUNCTION no son válidos

Para una función de salida MQXR_CONNECTION:

- CompCode y Reason se establecen de la misma forma que MQXR_BEFORE
- ExitResponse2 se ignora
- MQXCC_SUPPRESS_FUNCTION, MQXCC_SKIP_FUNCTION, MQXCC_SUPPRESS_EXIT no son válidos

En todos los casos, en los que una salida o el gestor de colas establece CompCode y Reason, los valores establecidos se pueden cambiar mediante una salida invocada más adelante, o mediante la llamada de API (si la llamada de API se invoca más tarde).

Tabla 871. Proceso de salida MQXR_BEFORE			
Valor de ExitResponse	CompCode y Reason establecidos por	Valor de ExitResponse2 (continuación predeterminada) Cadena	Valor de la API ExitResponse2 (continuación predeterminada)
MQXCC_Correcto	salida	Y	Y
MQXCC_SUPPRESS_EXIT	salida	Y	Y
MQXCC_SUPPRESS_FUNCTION	gestor de colas	N	N
MQXCC_SKIP FUNCIÓN	salida	N	N
MQXCC_FALLO	gestor de colas	N	N

Cómo procesan los clientes las funciones de salida

En general, los clientes procesan las funciones de salida de la misma forma que las aplicaciones de servidor, y el atributo *QMgrName* de esta estructura se aplica tanto si la función está en un servidor como si está en un cliente.

Sin embargo, el cliente no tiene noción del archivo *mqs.ini*, por lo que las estrofas *ApiExitCommon* y *APIExitTemplate* no se aplican. Sólo se aplica la estrofa *ApiExitLocal*, y esta estrofa se configura en el archivo *mqclient.ini*.

Estructura de contexto de salida de API de IBM MQ (MQAXC)

La estructura MQAXC, un bloque de control externo, se utiliza como parámetro de entrada para una salida de API.

MQAXC tiene la siguiente declaración C:

```
typedef struct tagMQAXC {
    MQCHAR4   StrucId;           /* Structure identifier */
    MQLONG    Version;          /* Structure version number */
    MQLONG    Environment;      /* Environment */
    MQCHAR12   UserId;           /* UserId associated with appl */
    MQBYTE40   SecurityId;      /* Extension to UserId running appl */
    MQCHAR264  ConnectionName;  /* Connection name */
    MQLONG    LongMCAUserIdLength; /* long MCA user identifier length */
    MQLONG    LongRemoteUserIdLength; /* long remote user identifier length */
    MQPTR     LongMCAUserIdPtr;  /* long MCA user identifier address */
    MQPTR     LongRemoteUserIdPtr; /* long remote user identifier address */
    MQCHAR28   AppName;         /* Application name */
}
```

```

MQLONG    ApplType;           /* Application type */
MQPID     ProcessId;          /* Process identifier */
MQTID     ThreadId;           /* Thread identifier */

/* Ver:1 */
MQCHAR    ChannelName[20]     /* Channel Name */
MQBYTE4    Reserved1;         /* Reserved */
PMQCD     pChannelDefinition; /* Channel Definition pointer */
};

```

Los parámetros para MQAXC son:

StrucId (MQCHAR4)-entrada

El identificador de estructura de contexto de salida, con un valor de MQAXC_STRUC_ID. Para los programas C, también se define la constante MQAXC_STRUC_ID_ARRAY, con el mismo valor que MQAXC_STRUC_ID, pero como una matriz de caracteres en lugar de una serie.

El manejador de salida establece este campo en la entrada para cada función de salida.

Versión (MQLONG)-entrada

El número de versión de la estructura, con un valor de:

MQAXC_VERSION_2

Número de versión para la estructura de contexto de salida.

MQAXC_VERSIÓN_ACTUAL

Número de versión actual para la estructura de contexto de salida.

El manejador de salida establece este campo en la entrada para cada función de salida.

Entorno (MQLONG)-entrada

Entorno desde el que se ha emitido una llamada de API de IBM MQ que ha dado como resultado que se haya controlado una función de salida. Los valores válidos para este campo son:

MQXE_OTHER

Este valor es coherente con las invocaciones que una salida de API ve si se llama a la salida desde una aplicación de servidor. Esto significa que una salida de API se ejecuta sin cambios en un cliente y no ve nada diferente.

Si la salida realmente necesita determinar si se está ejecutando en el cliente, puede hacerlo consultando los campos *ChannelName* y *ChannelDefinition*.

MQXE_MCA

Agente de canal de mensajes

MQXE_MCA_SVRCONN

Un agente de canal de mensajes que actúa en nombre de un cliente

MQXE_COMMAND_SERVER

El servidor de mandatos

MQXE_MQSC

El intérprete de mandatos runmqsc

El manejador de salida establece este campo en la entrada para cada función de salida.

UserId (MQCHAR12)-entrada

El ID de usuario asociado a la aplicación. En particular, en el caso de las conexiones de cliente, este campo contiene el ID de usuario del usuario adoptado en contraposición al ID de usuario bajo el que se ejecuta el código de canal. Si un ID de usuario en blanco fluye desde el cliente, no se realiza ningún cambio en el ID de usuario que ya se está utilizando. Es decir, no se adopta ningún ID de usuario nuevo.

El manejador de salida establece este campo en la entrada para cada función de salida. La longitud de este campo la proporciona MQ_USER_ID_LENGTH.

En el caso de un cliente, es el ID de usuario enviado desde el cliente al servidor. Tenga en cuenta que es posible que no sea el ID de usuario efectivo con el que se ejecuta el cliente en el gestor de colas, ya que podría haber una configuración MCAUser o CHLAUTH que cambie el ID de usuario.

SecurityId (MQBYTE40)-entrada

Una extensión del ID de usuario que ejecuta la aplicación. Su longitud la proporciona MQ_SECURITY_ID_LENGTH.

En el caso de un cliente, es el ID de usuario enviado desde el cliente al servidor. Tenga en cuenta que es posible que no sea el ID de usuario efectivo con el que se ejecuta el cliente en el gestor de colas, ya que podría haber una configuración MCAUser o CHLAUTH que cambie el ID de usuario.

ConnectionName (MQCHAR264)-entrada

El campo de nombre de conexión, establecido en la dirección del cliente. Por ejemplo, en el caso de TCP/IP, sería la dirección IP del cliente.

La longitud de este campo la proporciona MQ_CONN_NAME_LENGTH.

En el caso de un cliente, es la dirección asociada del gestor de colas.

LongMCAUserIdLength (MQLONG) - entrada

Longitud del identificador de usuario de MCA largo.

Cuando MCA se conecta al gestor de colas, este campo se establece en la longitud del identificador de usuario de MCA largo (o cero si no existe dicho identificador).

En el caso de un cliente, es el identificador de usuario largo del cliente.

LongRemoteUserIdLength (MQLONG) - entrada

La longitud del identificador de usuario remoto largo.

Cuando MCA se conecta al gestor de colas, este campo se establece en la longitud del identificador de usuario remoto largo. De lo contrario, este campo se establecerá en cero.

En el caso de un cliente, establezca este campo en cero.

LongMCAUserIdPtr (MQPTR) - entrada

Dirección del identificador de usuario de MCA largo.

Cuando MCA se conecta al gestor de colas, este campo se establece en la dirección del identificador de usuario de MCA largo (o en un puntero nulo si no existe dicho identificador).

En el caso de un cliente, es el identificador de usuario largo del cliente.

LongRemoteUserIdPtr (MQPTR) - entrada

La dirección del identificador de usuario remoto largo.

Cuando MCA se conecta al gestor de colas, este campo se establece en la dirección del identificador de usuario remoto largo (o en un puntero nulo si no existe dicho identificador).

En el caso de un cliente, establezca este campo en cero.

ApplName (MQCHAR28)-entrada

El nombre de la aplicación o componente que ha emitido la llamada de API IBM MQ .

Las reglas para generar el ApplName son las mismas que para generar el nombre predeterminado para un MQPUT.

El valor de este campo se encuentra consultando el nombre del programa en el sistema operativo. Su longitud la proporciona MQ_APPL_NAME_LENGTH.

ApplType (MQLONG)-entrada

El tipo de aplicación o componente que ha emitido la llamada de API IBM MQ .

El valor es MQAT_DEFAULT para la plataforma en la que se compila la aplicación, o equivale a uno de los valores MQAT_ * definidos.

El manejador de salida establece este campo en la entrada para cada función de salida.

ProcessId (MQPID)-entrada

Identificador de proceso del sistema operativo.

Cuando sea aplicable, el manejador de salida establece este campo en la entrada para cada función de salida.

ThreadId (MQTID)-entrada

El identificador de hebra de MQ . Es el mismo identificador que se utiliza en el rastreo de MQ y en los volcados FFST , pero puede ser diferente del identificador de hebra del sistema operativo.

Cuando sea aplicable, el manejador de salida establece este campo en la entrada para cada función de salida.

ChannelName (MQCHAR)-entrada

El nombre del canal, rellenado con espacios en blanco, si es aplicable y conocido.

Si no es aplicable, este campo se establece en caracteres NULL.

Reserved1 (MQBYTE4)-entrada

Este campo está reservado.

ChanneDefinition (PMQCD)-entrada

Puntero a la definición de canal que se utiliza, si es aplicable y conocida.

Si no es aplicable, este campo se establece en caracteres NULL.

Tenga en cuenta que el puntero sólo se completa si la conexión se está procesando en nombre de un canal IBM MQ y esa definición de canal se ha leído.

En concreto, la definición de canal no se proporciona en el servidor cuando se realiza la primera llamada MQCONN para el canal. Además, si el puntero se llena, la estructura (y cualquier subestructura) apuntada por el puntero debe ser tratada como de sólo lectura; cualquier actualización de la estructura conduciría a resultados impredecibles y no está soportada.

En el caso de un cliente, los campos distintos de aquellos con un valor especificado para un cliente, contienen valores que son adecuados para una aplicación cliente.

El área de la cadena de salida y el encabezado del área de la cadena de salida (MQACH)

Si es necesario, una función de salida puede adquirir almacenamiento para un área de cadena de salida y establecer el indicador de cadena de salida (ExitChainAreaPtr) en MQAXP para que apunte a este almacenamiento.

Las salidas (ya sean funciones de salida iguales o diferentes) pueden adquirir múltiples áreas de cadena de salida y vincularlas entre sí. Las áreas de cadena de salida solo se pueden añadir o eliminar de esta lista cuando se llaman desde el controlador de salida. Esto garantiza que no haya problemas de serialización causados por diferentes subprocesos que añaden o eliminan áreas de la lista al mismo tiempo.

Un área de cadena de salida debe comenzar con una estructura de encabezado MQACH, cuya declaración C es:

```
typedef struct tagMQACH {
    MQCHAR4   StrucId;           /* Structure identifier */
    MQLONG    Version;          /* Structure version number */
    MQLONG    StrucLength;       /* Length of the MQACH structure */
    MQLONG    ChainAreaLength;   /* Exit chain area length */
    MQCHAR48  ExitInfoName;      /* Exit information name */
    PMQACH    NextChainAreaPtr;  /* Pointer to next exit chain area */
};
```

Los campos del encabezado del área de la cadena de salida son:

StrucId (MQCHAR4) - entrada

El identificador de estructura del área de la cadena de salida, con un valor inicial, definido por MQACH_DEFAULT, de MQACH_STRUC_ID.

Para los programas C, también se define la constante MQACH_STRUC_ID_ARRAY, que tiene el mismo valor que MQACH_STRUC_ID, pero como una matriz de caracteres en lugar de una cadena.

Versión (MQLONG) - entrada

El número de versión de la estructura, como se indica a continuación:

MQACH_VERSION_1

El número de versión de la estructura del parámetro de salida.

MQACH_VERSION_ACTUAL

El número de versión actual de la estructura del contexto de salida.

El valor inicial de este campo, definido por MQACH_DEFAULT, es MQACH_CURRENT_VERSION.

Nota: Si introduces una nueva versión de esta estructura, el diseño de la parte existente no cambia. Las funciones de salida deben comprobar que el número de versión sea igual o superior a la versión más baja que contenga los campos que la función de salida necesita utilizar.

StrucLength (MQLONG) - entrada

La longitud de la estructura MQACH. Las salidas pueden utilizar este campo para determinar el inicio de los datos de salida, estableciéndolo en la longitud de la estructura creada por la salida.

El valor inicial de este campo, definido por MQACH_DEFAULT, es MQACH_CURRENT_LENGTH.

ChainAreaLength (MQLONG) - entrada

La longitud del área de la cadena de salida, establecida en la longitud total del área de la cadena de salida actual, incluida la cabecera MQACH.

El valor inicial de este campo, definido por MQACH_DEFAULT, es cero.

ExitInfoName (MQCHAR48) - entrada

El nombre de la información de salida.

Cuando una salida crea una estructura MQACH, debe inicializar este campo con su propio identificador de salida (ExitInfoName,) para que posteriormente otra instancia de esta salida o una salida cooperante puedan encontrar esta estructura MQACH.

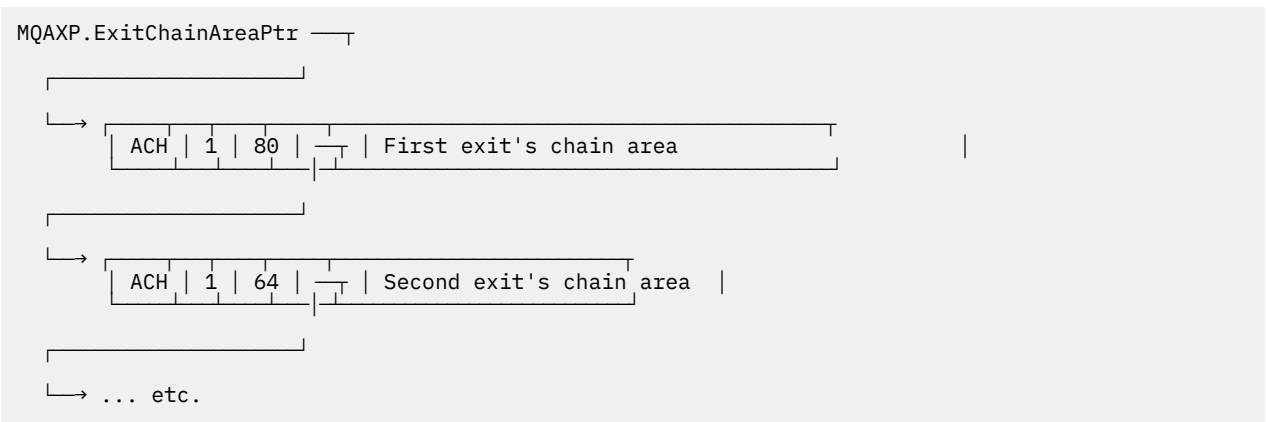
El valor inicial de este campo, definido por MQACH_DEFAULT, es una cadena de longitud cero ({""}).

NextChainAreaPtr (PMQACH) - entrada

Un puntero a la siguiente área de cadena de salida con un valor inicial, definido por MQACH_DEFAULT, de puntero nulo (NULL).

Las funciones de salida deben liberar el almacenamiento de cualquier área de cadena de salida que adquieran y manipular los punteros de cadena para eliminar sus áreas de cadena de salida de la lista.

Un área de cadena de salida se puede construir de la siguiente manera:



Constantes externas

Utilice este tema como información de referencia para las constantes externas disponibles para la API.

Las siguientes constantes externas están disponibles para las salidas de API:

MQXF_ * (identificadores de función de salida)

MQXF_INIT	1	X'00000001'
MQXF_TERM	2	X'00000002'
MQXF_CONN	3	X'00000003'

MQXF_CONNX	4	X'00000004'
MQXF_DISC	5	X'00000005'
MQXF_OPEN	6	X'00000006'
MQXF_CLOSE	7	X'00000007'
MQXF_PUT1	8	X'00000008'
MQXF_PUT	9	X'00000009'
MQXF_GET	10	X'0000000A'
MQXF_DATA_CONV_ON_GET	11	X'0000000B'
MQXF_INQ	12	X'0000000C'
MQXF_SET	13	X'0000000D'
MQXF_BEGIN	14	X'0000000E'
MQXF_CMIT	15	X'0000000F'
MQXF_BACK	16	X'00000010'
MQXF_STAT	18	X'00000012'
MQXF_CB	19	X'00000013'
MQXF_CTL	20	X'00000014'
MQXF_CALLBACK	21	X'00000015'
MQXF_SUB	22	X'00000016'
MQXF_SUBRQ	23	X'00000017'
MQXF_XACLOSE	24	X'00000018'
MQXF_XACOMMIT	25	X'00000019'
MQXF_XACOMplete	26	X'0000001A'
MQXF_XAEND	27	X'0000001B'
MQXF_XAFORGET	28	X'0000001C'
MQXF_XAOPEN	29	X'0000001D'
MQXF_XAPREPARE	30	X'0000001E'
MQXF_XARECOVER	31	X'0000001F'
MQXF_XAROLLBACK	32	X'00000020'
MQXF_XASTART	33	X'00000021'
MQXF_AXREG	34	X'00000022'
MQXF_AXUNREG	35	X'00000023'

MQXR_* (razones de salida)

MQXR_BEFORE	1	X'00000001'
MQXR_AFTER	2	X'00000002'
MQXR_CONNECTION	3	X'00000003'

MQXE_* (entornos)

MQXE_OTHER	0	X'00000000'
MQXE_MCA	1	X'00000001'
MQXE_MCA_SVRCONN	2	X'00000002'
MQXE_COMMAND_SERVER	3	X'00000003'
MQXE_MQSC	4	X'00000004'

MQ*_* (constantes adicionales)

MQAXP_VERSION_1	1	
MQAXP_VERSION_2	2	
MQAXC_VERSION_1	1	
MQACH_VERSION_1	1	
MQAXP_CURRENT_VERSION	1	
MQAXC_CURRENT_VERSION	1	
MQACH_CURRENT_VERSION	1	
MQXACT_EXTERNAL	1	
MQXACT_INTERNAL	2	
MQXT_API_EXIT	2	
MQACH_LENGTH_1	68 (32-bit platforms) 72 (64-bit platforms) 80 (128-bit platforms)	
MQACH_CURRENT_LENGTH	68 (32-bit platforms) 72 (64-bit platforms) 80 (128-bit platforms)	

MQ*_* (constantes nulas)

MQXPDA_NONE	X'00...00' (48 nulls)
MQXPDA_NONE_ARRAY	'\0','\0',...,'\0','\0'

MQXCC_* (códigos de terminación)

MQXCC_FAILED

-8

MQRC_* (códigos de razón)

MQRC_API_EXIT_ERROR 2374 X'00000946'

Una invocación de función de salida ha devuelto un código de respuesta no válido, o ha fallado de algún modo, y el gestor de colas no puede determinar la siguiente acción a realizar.

Examine los campos ExitResponse y ExitResponse2 de MQAXP para determinar el código de respuesta incorrecto y cambie la salida para que devuelva un código de respuesta válido.

MQRC_API_EXIT_INIT_ERROR 2375 X'00000947'

El gestor de colas ha encontrado un error al inicializar el entorno de ejecución para una función de salida de API.

MQRC_API_EXIT_TERM_ERROR 2376 X'00000948'

El gestor de colas ha encontrado un error al cerrar el entorno de ejecución para una función de salida de API.

MQRC_EXIT_REASON_ERROR 2377 X'00000949'

El valor del campo ExitReason proporcionado en una llamada de registro de punto de entrada de salida (MQXEP) es erróneo.

Examine el valor del campo ExitReason para determinar y corregir el valor de razón de salida incorrecta.

MQRC_RESERVED_VALUE_ERROR 2378 X'0000094A'

El valor del campo Reservado es erróneo.

Examine el valor del campo Reservado para determinar y corregir el valor Reservado.

Definiciones de tipo de idioma C

Este tema proporciona información sobre las typedefs asociadas con salidas de API disponibles en lenguaje C.

A continuación se muestran las definiciones de tipo de lenguaje C asociadas con las salidas de API:

```
typedef PMQLONG MQPOINTER PPMQLONG;
typedef PMQBYTE MQPOINTER PPMQBYTE;
typedef PMQHOBX MQPOINTER PPMQHOBX;
typedef PMQOD MQPOINTER PPMQOD;
typedef PMQMD MQPOINTER PPMQMD;
typedef PMQPMO MQPOINTER PPMQPMO;
typedef PMQGM0 MQPOINTER PPMQGM0;
typedef PMQCNO MQPOINTER PPMQCNO;
typedef PMQB0 MQPOINTER PPMQB0;

typedef MQAXP MQPOINTER PMQAXP;
typedef MQACH MQPOINTER PMQACH;
typedef MQAXC MQPOINTER PMQAXC;

typedef MQCHAR MQCHAR16[16];
typedef MQCHAR16 MQPOINTER PMQCHAR16;

typedef MQLONG MQPID;
typedef MQLONG MQTID;
```

Llamada de registro de punto de entrada de salida (MQXEP)

Utilice esta información para obtener información sobre MQXEP, invocación de lenguaje C MQXEP y prototipo de función C MQXEP.

Utilice la llamada MQXEP para:

1. Registrar los puntos de invocación de salida de API antes y después de IBM MQ en los que invocar las funciones de salida

2. Especificar los puntos de entrada de la función de salida
3. Anular el registro de los puntos de entrada de la función de salida

Normalmente codificaría las llamadas MQXEP en la función de salida MQ_INIT_EXIT, pero puede especificarlas en cualquier función de salida posterior.

Si utiliza una llamada MQXEP para registrar una función de salida ya registrada, la segunda llamada MQXEP se completa correctamente, sustituyendo la función de salida registrada.

Si utiliza una llamada MQXEP para registrar una función de salida NULL, la llamada MQXEP se completa correctamente y se anula el registro de la función de salida.

Si se utilizan llamadas MQXEP para registrar, anular el registro y volver a registrar una función de salida determinada durante la vida de una solicitud de conexión, se reactiva la función de salida registrada anteriormente. Cualquier almacenamiento todavía asignado y asociado con esta instancia de función de salida está disponible para que lo utilicen las funciones de la salida. (Este almacenamiento normalmente se libera durante la invocación de la función de salida de terminación.)

La interfaz con MQXEP es:

```
MQXEP (Hconfig, ExitReason, Function, EntryPoint, &ExitOpts, &CompCode, &Reason)
```

donde:

Hconfig (MQHCONFIG)-entrada

El descriptor de contexto de configuración, que representa la salida de API que incluye el conjunto de funciones que se están inicializando. Este valor lo genera el gestor de colas inmediatamente antes de invocar la función MQ_INIT_EXIT y se pasa en MQAXP a cada función de salida de API.

ExitReason (MQLONG)-entrada

Razón por la que se está registrando el punto de entrada, por las razones siguientes:

- Inicialización o terminación de nivel de conexión (MQXR_CONNECTION)
- Antes de una llamada de API de IBM MQ (MQXR_BEFORE)
- Después de una llamada de API de IBM MQ (MQXR_AFTER)

Función (MQLONG)-entrada

El identificador de función, cuyos valores válidos son las constantes MQXF_* (consulte [“Constantes externas”](#) en la página 1597).

EntryPoint (PMQFUNC)-entrada

La dirección del punto de entrada para la función de salida que se va a registrar. El valor NULL indica que no se ha proporcionado la función de salida o que se está anulando el registro de un registro anterior de la función de salida.

ExitOpts(MQXEPO)

Las salidas de API pueden especificar opciones que controlan cómo se registran las salidas de API. Si se especifica un puntero nulo para este campo, se asumen los valores predeterminados de la estructura MQXEPO.

CompCode (MQLONG)-salida

El código de terminación, cuyos valores válidos son:

MQCC_OK

Realización satisfactoria.

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón (MQLONG)-salida

El código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED:

MQRC_HCONFIG_ERROR

(2280, X'8E8') El manejador de configuración suministrado no es válido. Utilice el descriptor de contexto de configuración de MQAXP.

MQRC_MOTIVO_DE_SALIDA_ERROR

(2377, X' 949 ') La razón de invocación de función de salida proporcionada no es válida o no es válida para el identificador de función de salida proporcionado.

Utilice una de las razones de invocación de función de salida válidas (valor MQXR_*) o utilice una combinación válida de identificador de función y razón de salida. (Consulte [Tabla 872 en la página 1601.](#))

MQRC_ERROR_DE_FUNCIÓN

(2281, X'8E9') El identificador de función suministrado no es válido por el motivo de salida de la API. La tabla siguiente muestra combinaciones válidas de identificadores de función y ExitReasons.

Tabla 872. Combinaciones válidas de identificadores de función y ExitReasons	
Función	ExitReason
MQXF_INIT MQXF_TERM	MQXR_CONNECTION
MQXF_CONN MQXF_CONNX MQXF_DISC MQXF_OPEN MQXF_CIERRE MQXF_PUT1 MQXF_PUT MQXF_GET MQXF_INQ MQXF_SET MQXF_INICIO MQXF_CMIT MQXF_BACK MQXF_STAT MQXF_CB MQXF_CTL MQXF_CALLBACK MQXF_SUB MQXF_SUBRQ	MQXR_ANTES MQXR_DESPUÉS
MQXF_DATA_CONV_ON_GET	MQXR_ANTES

MQRC_RESOURCE_PROBLEM

(2102, X'836 ') Un intento de registrar o anular el registro de una función de salida ha fallado debido a un problema de recurso.

MQRC_UNEXPECTED_ERROR

(2195, X'893 ') Un intento de registrar o anular el registro de una función de salida ha fallado inesperadamente.

MQRC_NOMBRE_PROPIEDAD_ERROR

(2442, X'098A') Nombre de ExitProperties no válido.

MQRC_XEPO_ERROR

(2507, X'09CB') La estructura de opciones de salida no es válida.

Invocación de lenguaje C MQXEP

```
MQXEP (Hconfig, ExitReason, Function, EntryPoint, &ExitOpts, &CompCode, &Reason);
```

Declaración para lista de parámetros:

MQHCONFIG	Hconfig;	/* Configuration handle */
MQLONG	ExitReason;	/* Exit reason */
MQLONG	Function;	/* Function identifier */
PMQFUNC	EntryPoint;	/* Function entry point */
MQXEPO	ExitOpts;	/* Options that control the action of MQXEP */
MQLONG	CompCode;	/* Completion code */
MQLONG	Reason;	/* Reason code qualifying completion code */

Prototipo de función C MQXEP

```
void MQXEP (
MQHCONFIG Hconfig,      /* Configuration handle */
MQLONG ExitReason,      /* Exit reason */
MQLONG Function,        /* Function identifier */
PMQFUNC EntryPoint,     /* Function entry point */
PMQXEPO pExitOpts;      /* Options that control the action of MQXEP */
PMQLONG pCompCode;      /* Address of completion code */
PMQLONG pReason);       /* Address of reason code qualifying completion code */
```

Funciones de salida

Esta sección proporciona información general para ayudarle a utilizar las llamadas de función y describe cómo invocar las funciones de salida individuales.

Utilice esta información para comprender las reglas generales para las rutinas de salida de API y para configurar y limpiar el entorno de ejecución de salida.

Reglas generales para rutinas de salida de API

Se aplican las siguientes reglas generales al invocar rutinas de salida de API:

- En todos los casos, las funciones de salida de API se controlan antes de validar los parámetros de llamada de API y antes de cualquier comprobación de seguridad (en el caso de MQCONN, MQCONNX o MQOPEN).
- Los valores de los campos introducidos y la salida de una rutina de salida son:
 - En la entrada a una *antes de la función de salida de API de IBM MQ*, el valor de un campo puede ser establecido por el programa de aplicación o por una invocación de función de salida anterior.
 - En la salida de una función de salida de API *before IBM MQ*, el valor de un campo se puede dejar sin modificar, o se puede establecer en algún otro valor mediante la función de salida.
 - En la entrada a una función de salida de API *after IBM MQ*, el valor de un campo puede ser el valor establecido por el gestor de colas después de procesar la llamada de API IBM MQ, o puede establecerse en un valor mediante una invocación de función de salida anterior en la cadena de funciones de salida.
 - En la salida de una función de salida de llamada de API *after IBM MQ*, el valor de un campo se puede dejar sin modificar o establecer en algún otro valor mediante la función de salida.
- Las funciones de salida deben comunicarse con el gestor de colas utilizando los campos ExitResponse y ExitResponse2.
- Los campos CompCode y Código de razón se comunican de nuevo con la aplicación. El gestor de colas y las funciones de salida pueden establecer los campos CompCode y Código de razón.
- La llamada MQXEP devuelve nuevos códigos de razón a las funciones de salida que llaman a MQXEP. Sin embargo, las funciones de salida pueden convertir estos nuevos códigos de razón a cualquier código de razón existente que las aplicaciones existentes y nuevas puedan comprender.

- Cada prototipo de función de salida tiene parámetros similares a la función de la API con un nivel adicional de indirección, excepto CompCode y Reason.
- Las salidas de la API pueden realizar llamadas « MQI » (excepto MQDISC), pero estas llamadas « MQI » no invocan por sí mismas salidas de la API.

Tenga en cuenta que si la aplicación está en un servidor o un cliente, no puede predecir la secuencia de las llamadas de salida de API. Es posible que una llamada BEFORE de salida de API no vaya seguida inmediatamente de una llamada AFTER .

La llamada BEFORE puede ir seguida de otra llamada BEFORE . Por ejemplo:

```
MQCTL ANTES
ANTES de devolución de llamada
ANTES DE MQPUT
DESPUÉS de MQPUT
DESPUÉS de devolución de llamada
DESPUÉS DE MQCTL
```

o

```
ANTES DE XAOPEN
ANTES DE MQCONN
DESPUÉS de MQCONN
DESPUÉS DE XAOPEN
```

En el cliente, hay una salida que puede modificar el comportamiento de la llamada MQCONN o MQCONN, denominada salida PreConnect . La salida PreConnect puede modificar cualquiera de los parámetros de la llamada MQCONN o MQCONN, incluido el nombre del gestor de colas. El cliente llama primero a esta salida y, a continuación, invoca la llamada MQCONN o MQCONN. Tenga en cuenta que sólo la llamada MQCONN o MQCONN inicial invoca la salida de API; las llamadas de reconexión posteriores no tienen ningún efecto.

Entorno de ejecución

En general, todos los errores de las funciones de salida se comunican de nuevo al manejador de salida utilizando los campos ExitResponse y ExitResponse2 en MQAXP.

Estos errores a su vez se convierten en valores MQCC_* y MQRC_* y se comunican de nuevo a la aplicación en los campos CompCode y Reason. Sin embargo, los errores encontrados en la lógica del manejador de salida se comunican de nuevo a la aplicación como valores MQCC_* y MQRC_* en los campos CompCode y Reason.

Si una función MQ_TERM_EXIT devuelve un error:

- La llamada MQDISC ya ha tenido lugar
- No hay otra oportunidad de controlar la función de salida *after* MQ_TERM_EXIT (y, por lo tanto, realizar la limpieza del entorno de ejecución de salida)
- No se ha realizado la limpieza del entorno de ejecución de salida

La salida no se puede descargar porque puede que todavía esté en uso. Además, otras salidas registradas más abajo en la cadena de salida para las que la salida *anterior* ha sido satisfactoria, se activarán en el orden inverso.

Configuración del entorno de ejecución de salida

Al procesar una llamada MQCONN o MQCONN explícita, la lógica de manejo de salida configura el entorno de ejecución de salida antes de invocar la función de inicialización de salida (MQ_INIT_EXIT). La configuración del entorno de ejecución de salida implica cargar la salida, adquirir almacenamiento para e inicializar las estructuras de parámetros de salida. También se asigna el descriptor de contexto de configuración de salida.

Si se producen errores durante esta fase, la llamada MQCONN o MQCONNX falla con CompCode MQCC_FAILED y uno de los siguientes códigos de razón:

MQRC_API_EXIT_LOAD_ERROR

Ha fallado un intento de cargar un módulo de salida de API.

MQRC_API_EXIT_NOT_FOUND

No se ha podido encontrar una función de salida de API en el módulo de salida de API.

MQRC_STORAGE_NOT_AVAILABLE

Ha fallado un intento de inicializar el entorno de ejecución para una función de salida de API porque no había suficiente almacenamiento disponible.

MQRC_API_EXIT_INIT_ERROR

Se ha encontrado un error al inicializar el entorno de ejecución para una función de salida de API.

Limpieza del entorno de ejecución de salida

Al procesar una llamada MQDISC explícita, o una solicitud de desconexión implícita como resultado de la finalización de una aplicación, es posible que la lógica de manejo de salidas tenga que limpiar el entorno de ejecución de salida después de invocar la función de terminación de salida (MQ_TERM_EXIT), si está registrada.

La limpieza del entorno de ejecución de salida implica la liberación de almacenamiento para estructuras de parámetros de salida, posiblemente la supresión de los módulos cargados previamente en la memoria.

Si se producen errores durante esta fase, una llamada MQDISC explícita falla con CompCode MQCC_FAILED y el siguiente código de razón (los errores no se resaltan en las solicitudes de desconexión implícitas):

MQRC_API_EXIT_TERM_ERROR

Se ha encontrado un error al cerrar el entorno de ejecución para una función de salida de API. La salida no debe devolver ningún error de MQDISC antes o después de las llamadas a la función de salida de la API MQ_TERM*.

Salidas de API en clientes

Un cliente utiliza la salida PreConnect para modificar el comportamiento de las llamadas MQCONN y MQCONNX y no da soporte a las propiedades de salida de API.

Salida PreConnect

En un cliente, se puede utilizar la salida « PreConnect » para consultar la definición del canal en un repositorio central, como un servidor de LDAP.

La salida PreConnect también puede modificar cualquier parámetro, o todos los parámetros, en una llamada MQCONN o MQCONNX, por ejemplo, el nombre del gestor de colas.

En el caso de las aplicaciones cliente, se debe llamar a la salida PreConnect antes de la salida de API porque la salida de API MQCONN o MQCONNX sólo se llama una vez que se conoce el nombre del gestor de colas y la salida PreConnect puede cambiar este nombre.

Tenga en cuenta que sólo la llamada MQCONN o MQCONNX inicial invoca la salida.

Propiedades de salida de API

En un servidor, las salidas de API pueden registrar una estructura MQXEPO en el momento de la inicialización. La estructura MQXEPO contiene el campo ExitProperties que detalla el grupo de propiedades en las que está interesada la salida. Esto tiene el efecto de generar un descriptor de contexto de propiedad de mensaje independiente que la salida puede manipular por separado de cualquier descriptor de contexto de propiedad de mensaje de aplicación.

En un cliente, las propiedades de salida de API no están soportadas. Si se intenta registrar un nombre de grupo de propiedades en un cliente, la función falla con un código de razón de MQRC_EXIT_PROPS_NOT_SUPPORTED.

Restitución-MQ_BACK_EXIT

MQ_BACK_EXIT proporciona una función de salida de restitución para realizar *antes* y *después* del proceso de restitución. Utilice el identificador de función MQXF_BACK con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar *antes* y *después* de las funciones de salida de llamada de restitución.

La interfaz para esta función es:

```
MQ_BACK_EXIT (&ExitParms, &ExitContext, &Hconn, &CompCode, &Reason)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida

Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida

Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada

Descriptor de conexión.

CompCode (MQLONG)-entrada/salida

Código de terminación, cuyos valores válidos son:

MQCC_OK

Realización satisfactoria.

MQCC_WARNING

Finalización parcial.

MQCC_FAILED

Llamada fallida

Razón (MQLONG)-entrada/salida

Código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK, el único valor válido es:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED o MQCC_WARNING, la función de salida puede establecer el campo de código de razón a cualquier valor de MQRC_* válido.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
MQAXP      ExitParms;      /* Exit parameter structure */
MQAXC      ExitContext;    /* Exit context structure */
MQHCONN    Hconn;         /* Connection handle */
MQLONG     CompCode;       /* Completion code */
MQLONG     Reason;         /* Reason code qualifying completion code */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
MQ_BACK_EXIT (&ExitParms, &ExitContext, &Hconn, &CompCode, &Reason);
```

La salida debe coincidir con el siguiente prototipo de función C:

```
void MQENTRY MQ_BACK_EXIT (
PMQAXP     pExitParms,      /* Address of exit parameter structure */
PMQAXC     pExitContext,    /* Address of exit context structure */
PMQHCONN   pHconn,         /* Address of connection handle */
PMQLONG    pCompCode,       /* Address of completion code */
PMQLONG    pReason);        /* Address of reason code qualifying completion
                             code */
```

Inicio-MQ_BEGIN_EXIT

MQ_BEGIN_EXIT proporciona una función de salida de inicio para realizar *antes y después del proceso de llamada* MQBEGIN. Utilice el identificador de función MQXF_BEGIN con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar *antes y después* de las funciones de salida de llamada MQBEGIN.

La interfaz para esta función es:

```
MQ_BEGIN_EXIT (&ExitParms, &ExitContext, &Hconn, &pBeginOptions, &CompCode,
               &Reason)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida

Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida

Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada

Descriptor de conexión.

pBeginOptions (PMQBO)- entrada/salida

Puntero a las opciones de inicio.

CompCode (MQLONG)-entrada/salida

Código de terminación, cuyos valores válidos son:

MQCC_OK

Realización satisfactoria.

MQCC_WARNING

Finalización parcial.

MQCC_FAILED

Llamada fallida

Razón (MQLONG)-entrada/salida

Código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK, el único valor válido es:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED o MQCC_WARNING, la función de salida puede establecer el campo de código de razón a cualquier valor de MQRC_* válido.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
MQAXP    ExitParms;      /* Exit parameter structure */
MQAXC    ExitContext;    /* Exit context structure */
MQHCONN   Hconn;         /* Connection handle */
PMQBO    pBeginOptions;  /* Ptr to begin options */
MQLONG    CompCode;      /* Completion code */
MQLONG    Reason;        /* Reason code qualifying completion code */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
MQ_BEGIN_EXIT (&ExitParms, &ExitContext, &Hconn, &pBeginOptions, &CompCode,
               &Reason);
```

La salida debe coincidir con el siguiente prototipo de función C:

```
void MQENTRY MQ_BEGIN_EXIT (
PMQAXP    pExitParms,      /* Address of exit parameter structure */
PMQAXC    pExitContext,    /* Address of exit context structure */
PMQHCONN   pHconn,         /* Address of connection handle */
```

```

PPMQBO    ppBeginOptions,    /* Address of ptr to begin options */
PMQLONG    pCompCode,        /* Address of completion code */
PMQLONG    pReason);        /* Address of reason code qualifying completion
                             code */

```

Devolución de llamada-MQ_CALLBACK_EXIT

MQ_CALLBACK_EXIT proporciona una función de salida para realizar *antes* y *después* del proceso de devolución de llamada. Utilice el identificador de función MQXF_CALLBACK con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar las funciones de salida de llamada *before* y *after*.

La interfaz para esta función es:

```

MQ_CALLBACK_EXIT (&ExitParms, &ExitContext, &Hconn, &pMsgDesc, &pGetMsgOpts,
                  &pBuffer, &pMQCBCContext)

```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida

Estructura de parámetros de salida

ExitContext (MQAXC)-entrada/salida

Estructura de contexto de salida

Hconn (MQHCONN)-entrada/salida

Descriptor de contexto de conexión

pMsgDesc

Descriptor de mensaje

pGetMsgOpts

Opciones que controlan la acción de MQGET

pBuffer

Área para contener los datos del mensaje

pMQCBCContext

Datos de contexto para la devolución de llamada

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```

MQAXP      ExitParms;        /* Exit parameter structure */
MQAXC      ExitContext;      /* Exit context structure */
MQHCONN     Hconn;           /* Connection handle */
PMQMD       pMsgDesc;        /* Message descriptor */
PMQGMO      pGetMsgOpts;     /* Options that define the operation of the consumer */
PMQVOID     pBuffer;        /* Area to contain the message data */
PMQCBC      pContext;        /* Context data for the callback */

```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```

MQ_SUBRQ_EXIT (&ExitParms, &ExitContext, &Hconn, &pMsgDesc, &pGetMsgOpts, &pBuffer,
               &pContext);

```

La salida debe coincidir con el siguiente prototipo de función C:

```

void MQENTRY MQ_CALLBACK_EXIT (
PMQAXP      pExitParms;      /* Exit parameter structure */
PMQAXC      pExitContext;    /* Exit context structure */
PMQHCONN     pHconn;         /* Connection handle */
PPMQMD       ppMsgDesc;      /* Message descriptor */
PPMQGMO      ppGetMsgOpts;    /* Options that define the operation of the consumer */
PPMQVOID     ppBuffer;       /* Area to contain the message data */
PPMQCBC      ppContext;)     /* Context data for the callback */

```

Notas de uso

1. La salida de devolución de llamada se invoca antes de que se invoque al consumidor y después de que se haya completado la función de consumidor del consumidor. Aunque las estructuras MQMD y MQGMO son modificables, el cambio de los valores de la salida anterior no reconduce la recuperación de un mensaje de la cola, ya que el mensaje ya se ha eliminado de la cola para entregarlo a la función de consumidor.

Gestionar funciones de devolución de llamada-MQ_CB_EXIT

MQ_CB_EXIT proporciona una función de salida para realizar *antes* y *después* de la llamada MQCB. Utilice el identificador de función MQXF_CB con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar *antes* y *después* de las funciones de salida de llamada MQCB.

La interfaz para esta función es:

```
MQ_CB_EXIT (&ExitParms, &ExitContext, &Hconn, &Operation, &pCallbackDesc,  
            &Hobj, &pMsgDesc, &pGetMsgOpts, &CompCode, &Reason)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida

Estructura de parámetros de salida

ExitContext (MQAXC)-entrada/salida

Estructura de contexto de salida

Hconn (MQHCONN)-entrada/salida

Descriptor de contexto de conexión

Operación (MQLONG)-entrada/salida

Valor de operación

pCallbackDesc (PMQCBD) - entrada/salida

Descriptor de devolución de llamada

Hobj (MQHOBJ)-entrada/salida

Descriptor de contexto del objeto

pMsgDesc (PMQMD) - entrada/salida

Descriptor de mensaje

pGetMsgOpts (PMQGMO) - entrada/salida

Opciones que controlan la acción de MQCB

CompCode (MQLONG)-entrada/salida

Código de terminación

Razón (MQLONG)-entrada/salida

Código de razón que califica a CompCode

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
MQAXP      ExitParms;      /* Exit parameter structure */  
MQAXC      ExitContext;    /* Exit context structure */  
MQHCONN    Hconn;          /* Connection handle */  
MQLONG     Operation;      /* Operation value. */  
MQCBD      pMsgDesc;       /* Callback descriptor. */  
MQHOBJ     Hobj;           /* Object handle. */  
PMQMD      pMsgDesc;       /* Message descriptor */  
PMQGMO     pGetMsgOpts;    /* Options that define the operation of the consumer */  
PMQLONG    CompCode;       /* Completion code. */  
PMQLONG    Reason;         /* Reason code qualifying CompCode. */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
MQ_CB_EXIT (&ExitParms, &ExitContext, &Hconn, &Operation, &Hobj, &pMsgDesc,  
            &pGetMsgOpts, &CompCode, &Reason);
```

La salida debe coincidir con el siguiente prototipo de función C:

```
void MQENTRY MQ_CB_EXIT (
    PMQAXP      pExitParms;      /* Exit parameter structure */
    PMQAXC      pExitContext;    /* Exit context structure */
    PMQHCONN    pHconn;          /* Connection handle */
    PMQLONG     pOperation;      /* Callback operation */
    PMQHOBJS    pHobj;          /* Object handle */
    PPMQMD      ppMsgDesc;       /* Message descriptor */
    PPMQGM      ppGetMsgOpts;    /* Options that control the action of MQCB */
    PMQLONG     pCompCode;       /* Completion code */
    PMQLONG     pReason;         /* Reason code qualifying CompCode */
)
```

Cerrar-MQ_CLOSE_EXIT

MQ_CLOSE_EXIT proporciona una función de salida de cierre para realizar *antes y después del proceso de la llamada MQCLOSE* de . Utilice el identificador de función MQXF_CLOSE con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar *antes y después de* las funciones de salida de llamada MQCLOSE.

La interfaz para esta función es:

```
MQ_CLOSE_EXIT (&ExitParms, &ExitContext, &Hconn, &pHobj,
               &Options, &CompCode, &Reason)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida

Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida

Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada

Descriptor de conexión.

pHobj (PMQHOBJS)-entrada

Puntero al descriptor de contexto de objeto.

Opciones (MQLONG)-entrada/salida

Opciones de cierre.

CompCode (MQLONG)-entrada/salida

Código de terminación, cuyos valores válidos son:

MQCC_OK

Realización satisfactoria.

MQCC_FAILED

Llamada fallida

Razón (MQLONG)-entrada/salida

Código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK, el único valor válido es:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED, la función de salida puede establecer el campo de código de razón en cualquier valor MQRC_ * válido.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
MQAXP      ExitParms;      /* Exit parameter structure */
MQAXC      ExitContext;    /* Exit context structure */
MQHCONN    Hconn;          /* Connection handle */
PMQHOBJS    pHobj;         /* Ptr to object handle */
MQLONG     Options;        /* Close options */
```

MQLONG	CompCode;	/* Completion code */
MQLONG	Reason;	/* Reason code */

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
MQ_CLOSE_EXIT (&ExitParms, &ExitContext,&Hconn, &pHobj, &Options,
               &CompCode, &Reason);
```

La salida debe coincidir con el siguiente prototipo de función C:

```
void MQENTRY MQ_CLOSE_EXIT (
PMQAXP      pExitParms,      /* Address of exit parameter structure */
PMQAXC      pExitContext,    /* Address of exit context structure */
PMQHCONN    pHconn,         /* Address of connection handle */
PPMHOBJS    ppHobj,         /* Address of ptr to object handle */
PMQLONG      pOptions,       /* Address of close options */
PMQLONG      pCompCode,      /* Address of completion code */
PMQLONG      pReason);      /* Address of reason code qualifying
                             completion code */
```

Confirmar-MQ_CMIT_EXIT

MQ_CMIT_EXIT proporciona una función de salida de confirmación para realizar *antes* y *después* del proceso de confirmación. Utilice el identificador de función MQXF_CMIT con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar *antes* y *después* de las funciones de salida de llamada de confirmación.

Si una operación de confirmación falla y la transacción se restituye, la llamada MQCMIT falla con MQCC_WARNING y MQRC_BACKED_OUT. Estos códigos de retorno y de razón se pasan a cualquier *después* de las funciones de salida MQCMIT para dar a las funciones de salida una indicación de que la unidad de trabajo se ha restituido.

La interfaz para esta función es:

```
MQ_CMIT_EXIT (&ExitParms, &ExitContext, &Hconn, &CompCode, &Reason)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida

Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida

Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada

Descriptor de conexión.

CompCode (MQLONG)-entrada/salida

Código de terminación, cuyos valores válidos son:

MQCC_OK

Realización satisfactoria.

MQCC_WARNING

Finalización parcial.

MQCC_FAILED

Llamada fallida

Razón (MQLONG)-entrada/salida

Código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK, el único valor válido es:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED o MQCC_WARNING, la función de salida puede establecer el campo de código de razón a cualquier valor de MQRC_* válido.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
MQAXP    ExitParms;      /* Exit parameter structure */
MQAXC    ExitContext;    /* Exit context structure */
MQHCONN  Hconn;          /* Connection handle */
MQLONG   CompCode;       /* Completion code */
MQLONG   Reason;         /* Reason code qualifying completion code */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
MQ_CMIT_EXIT (&ExitParms, &ExitContext,&Hconn, &CompCode, &Reason);
```

La salida debe coincidir con el siguiente prototipo de función C:

```
void MQENTRY MQ_CMIT_EXIT (
PMQAXP    pExitParms,      /* Address of exit parameter structure */
PMQAXC    pExitContext,    /* Address of exit context structure */
PMQHCONN  pHconn,          /* Address of connection handle */
PMQLONG   pCompCode,       /* Address of completion code */
PMQLONG   pReason);        /* Address of reason code qualifying completion
                             code */
```

Notas de uso

1. La interfaz de función MQ_GET_EXIT descrita aquí se utiliza tanto para la función de salida MQXF_GET como para la función de salida [“MQXF_DATA_CONV_ON_GET”](#) en la [página 1617](#) .

Se definen puntos de entrada separados para estas dos funciones de salida, por lo que para interceptar *ambos* la llamada MQXEP debe utilizarse dos veces; para esta llamada utilice el identificador de función MQXF_GET.

Puesto que la interfaz MQ_GET_EXIT es la misma para MQXF_GET y MQXF_DATA_CONV_ON_GET, se puede utilizar una única función de salida para ambos; el campo *Function* de la estructura [MQAXP](#) indica qué función de salida se ha invocado. De forma alternativa, se puede utilizar la llamada MQXEP para registrar distintas funciones de salida para los dos casos.

Extensión de conexión y conexión-MQ_CONNX_EXIT

MQ_CONNX_EXIT proporciona una función de salida de conexión para realizar *antes* y *después* del proceso MQCONN, y una función de salida de extensión de conexión para realizar *antes* y *después* del proceso MQCONNX.

Se invoca la misma interfaz, tal como se describe aquí, para las funciones de salida de llamada MQCONN y MQCONNX.

Cuando el agente de canal de mensajes (MCA) responde a una conexión de cliente de entrada, el MCA puede conectarse y realizar una serie de llamadas de API de IBM MQ antes de que se conozca completamente el estado del cliente. Estas llamadas de API llaman a las funciones de salida de API con MQAXC basadas en el propio programa MCA (por ejemplo, en los campos UserId y ConnectionName de MQAXC).

Cuando el MCA responde a las llamadas de API de cliente de entrada posteriores, la estructura MQAXC se basa en el cliente de entrada, estableciendo los campos UserId y ConnectionName de forma adecuada.

El nombre del gestor de colas establecido por la aplicación en una llamada MQCONN o MQCONNX se pasa a la llamada de conexión subyacente. Cualquier intento de un *anterior* a MQ_CONNX_EXIT de cambiar el nombre del gestor de colas no tiene ningún efecto.

Utilice los identificadores de función MQXF_CONN y MQXF_CONNX con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar *antes* y *después* de las funciones de salida de llamada MQCONN y MQCONNX de .

Una salida MQ_CONNX_EXIT llamada por la razón MQXR_BEFORE *no debe* emitir ninguna llamada de API IBM MQ , ya que el entorno correcto no se ha configurado en este momento.

Un MQ_CONNX_EXIT no puede llamar a MQDISC desde una llamada de salida de API para la conexión para la que se está llamando. Esta restricción es aplicable a las salidas de API de cliente y servidor.

La interfaz con MQCONN y MQCONNX es idéntica:

```
MQ_CONNX_EXIT (&ExitParms, &ExitContext, &pQMgrName, &pConnectOpts,  
&pHconn, &CompCode, &Reason);
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida

Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida

Salir de la estructura de contexto.

pQMgrName (PMQCHAR) - entrada

Puntero al nombre de gestor de colas proporcionado en la llamada MQCONNX. La salida no debe cambiar este nombre en la llamada MQCONN o MQCONNX.

pConnectOpts (PMQCNO) - entrada/salida

Puntero a las opciones que controlan la acción de la llamada MQCONNX.

Consulte “MQCNO - Opciones de conexión” en la [página 269](#) para obtener los detalles.

Para la función de salida MQXF_CONN, pConnectOpts apunta a la estructura de opciones de conexión por defecto (MQCNO_DEFAULT).

pHconn (PMQHCONN)-entrada

Puntero al descriptor de conexión.

CompCode (MQLONG)-entrada/salida

Código de terminación, cuyos valores válidos son:

MQCC_OK

Realización satisfactoria.

MQCC_WARNING

Aviso (terminación parcial)

MQCC_FAILED

Llamada fallida

Razón (MQLONG)-entrada/salida

Código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK, el único valor válido es:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED o MQCC_WARNING, la función de salida puede establecer el campo de código de razón a cualquier valor de MQRC_* válido.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

MQAXP	ExitParms;	/* Exit parameter structure */
MQAXC	ExitContext;	/* Exit context structure */
PMQCHAR	pQMgrName;	/* Ptr to Queue manager name */
PMQCNO	pConnectOpts;	/* Ptr to Connection options */
PMQHCONN	pHconn;	/* Ptr to Connection handle */
MQLONG	CompCode;	/* Completion code */
MQLONG	Reason;	/* Reason code */

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
MQ_CONNX_EXIT (&ExitParms, &ExitContext, &pQMgrName, &pConnectOpts,  
&pHconn, &CompCode, &Reason);
```


La salida debe coincidir con el siguiente prototipo de función C:

```
void MQENTRY MQ_CONNX_EXIT (
    PMQAXP      pExitParms,      /* Address of exit parameter structure */
    PMQAXC      pExitContext,    /* Address of exit context structure */
    PPMQCHAR    ppQMgrName,      /* Address of ptr to queue manager name */
    PPMQCNO     ppConnectOpts,   /* Address of ptr to connection options */
    PPMQHCONN    ppHconn,        /* Address of ptr to connection handle */
    PMQLONG     pCompCode,       /* Address of completion code */
    PMQLONG     pReason);        /* Address of reason code qualifying
                                completion code */
```

Notas de uso

1. La interfaz de función MQ_CONNX_EXIT descrita aquí se utiliza tanto para la llamada MQCONN como para la llamada MQCONNX. Sin embargo, se definen puntos de entrada separados para estas dos llamadas. Para interceptar *ambas* llamadas, la llamada MQXEP debe utilizarse al menos dos veces, una con el identificador de función MQXF_CONN y otra con MQXF_CONNX.

Puesto que la interfaz MQ_CONNX_EXIT es la misma para MQCONN y MQCONNX, se puede utilizar una única función de salida para ambas llamadas; el campo *Function* de la estructura MQAXP indica qué llamada está en curso. De forma alternativa, la llamada MQXEP se puede utilizar para registrar distintas funciones de salida para las dos llamadas.

2. Cuando un agente de canal de mensajes (MCA) responde a una conexión de cliente de entrada, el MCA puede emitir una serie de llamadas MQ antes de que se conozca completamente el estado del cliente. Estas llamadas MQ hacen que las funciones de salida de API se invoquen con la estructura MQAXC que contiene datos relacionados con el MCA, y no con el cliente (por ejemplo, el identificador de usuario y el nombre de conexión). Sin embargo, una vez que el estado del cliente es totalmente conocido, las llamadas MQ posteriores dan como resultado que se invoquen las funciones de salida de API con los datos de cliente adecuados en la estructura MQAXC.
3. Todas las funciones de salida MQXR_BEFORE se invocan antes de que el gestor de colas realice cualquier validación de parámetro. Por lo tanto, es posible que los parámetros no sean válidos (incluidos los punteros no válidos para las direcciones de los parámetros).

La función MQ_CONNX_EXIT se invoca antes de que el gestor de colas realice comprobaciones de autorización.

4. La función de salida no debe cambiar el nombre del gestor de colas especificado en la llamada MQCONN o MQCONNX. Si la función de salida cambia el nombre, los resultados no están definidos.
5. Una función de salida MQXR_BEFORE para MQ_CONNX_EXIT no puede emitir llamadas de MQ distintas de MQXEP.

Devolución de llamada de control-MQ_CTL_EXIT

MQ_CTL_EXIT proporciona una función de salida de solicitud de suscripción para realizar *antes* y *después* del proceso de devolución de llamada de control. Utilice el identificador de función MQXF_CTL con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar *antes* y *después* de las funciones de salida de llamada de retorno de control.

La interfaz para esta función es:

```
MQ_CTL_EXIT (&Hconn, &Operation, &ControlOpts, &CompCode, &Reason)
```

donde los parámetros son:

Hconn (MQHCONN)-entrada/salida

Descriptor de conexión.

Entrada/salida de operación (MQLONG)

La operación que se está procesando en la devolución de llamada definida para el descriptor de objeto especificado

Entrada/salida de ControlOpts (MQCTLO)

Opciones que controlan la acción de MQCTL

CompCode (MQLONG)-entrada/salida

Código de terminación, cuyos valores válidos son:

MQCC_OK

Realización satisfactoria.

MQCC_WARNING

Finalización parcial.

MQCC_FAILED

Llamada fallida

Razón (MQLONG)-entrada/salida

Código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK, el único valor válido es:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED o MQCC_WARNING, la función de salida puede establecer el campo de código de razón a cualquier valor de MQRC_* válido.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
MQHCONN  Hconn;          /* Connection handle */
MQLONG   Operation;      /* Operation being processed */
MQCTLO   ControlOpts;    /* Options that control the action of MQCTL */
MQLONG   CompCode;       /* Completion code */
MQLONG   Reason;         /* Reason code qualifying completion code */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
MQ_CTL_EXIT (&Hconn, &Operation, &ControlOpts, &CompCode, &Reason);
```

La salida debe coincidir con el siguiente prototipo de función C:

```
void MQENTRY MQ_CTL_EXIT (
PMQHCONN  pHconn;        /* Address of connection handle */
PMQLONG   pOperation;     /* Address of operation being processed */
PMQCTLO   pControlOpts;   /* Address of options that control the action of MQCTL */
PMQLONG   pCompCode;      /* Address of completion code */
PMQLONG   pReason;)       /* Address of reason code qualifying completion code */
```

Desconectar-MQ_DISC_EXIT

MQ_DISC_EXIT proporciona una función de salida de desconexión para realizar *antes* y *después* del proceso de salida MQDISC. Utilice el identificador de función MQXF_DISC con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar *antes* y *después* de las funciones de salida de llamada MQDISC.

La interfaz para esta función es

```
MQ_DISC_EXIT (&ExitParms, &ExitContext, &pHconn,
&CompCode, &Reason);
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida

Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida

Salir de la estructura de contexto.

pHconn (PMQHCONN)-entrada

Puntero al descriptor de conexión.

Para la llamada MQDISC anterior, el valor de este campo es uno de los siguientes:

- El descriptor de conexión devuelto en la llamada MQCONN o MQCONNX
- Cero, para entornos en los que un adaptador específico del entorno se ha conectado al gestor de colas
- Un valor establecido por una invocación de función de salida anterior

Para la llamada MQDISC posterior, el valor de este campo es cero o un valor establecido por una invocación de función de salida anterior.

CompCode (MQLONG)-entrada/salida

Código de terminación, cuyos valores válidos son:

MQCC_OK

Realización satisfactoria.

MQCC_WARNING

Terminación parcial

MQCC_FAILED

Llamada fallida

Razón (MQLONG)-entrada/salida

Código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK, el único valor válido es:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED o MQCC_WARNING, la función de salida puede establecer el campo de código de razón a cualquier valor de MQRC_* válido.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

MQAXP	ExitParms;	/* Exit parameter structure */
MQAXC	ExitContext;	/* Exit context structure */
PMQHCONN	pHconn;	/* Ptr to Connection handle */
MQLONG	CompCode;	/* Completion code */
MQLONG	Reason;	/* Reason code */

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
MQ_DISC_EXIT (&ExitParms, &ExitContext, &pHconn,
              &CompCode, &Reason);
```

La salida debe coincidir con el siguiente prototipo de función C:

```
void MQENTRY MQ_DISC_EXIT (
PMQAXP      pExitParms,      /* Address of exit parameter structure */
PMQAXC      pExitContext,    /* Address of exit context structure */
PMQHCONN    ppHconn,         /* Address of ptr to connection handle */
PMQLONG     pCompCode,       /* Address of completion code */
PMQLONG     pReason);        /* Address of reason code qualifying
                               completion code */
```

Obtener-MQ_GET_EXIT

MQ_GET_EXIT proporciona una función de obtención de salida para realizar *antes y después del proceso de la llamada MQGET* de .

Hay dos identificadores de función:

1. Utilice MQXF_GET con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar *antes y después* de las funciones de salida de llamada MQGET.

2. Consulte “MQXF_DATA_CONV_ON_GET” en la página 1617 para obtener información sobre cómo utilizar el identificador de función MQXF_DATA_CONV_ON_GET.

La interfaz para esta función es:

```
MQ_GET_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &pMsgDesc,  
             &pGetMsgOpts, &BufferLength, &pBuffer, &pDataLength,  
             &CompCode, &Reason)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida

Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida

Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada

Descriptor de conexión.

Hobj (MQHOBJ)-entrada/salida

El manejador del objeto.

pMsgDesc (PMQMD) - entrada/salida

Puntero al descriptor de mensaje.

pGetMsgOpts (PMQGMO) - entrada/salida

Puntero para obtener opciones de mensaje.

BufferLength (MQLONG)-entrada/salida

Longitud del almacenamiento intermedio de mensajes.

pBuffer (PMQBYTE)-entrada/salida

Puntero al almacenamiento intermedio de mensajes.

pDataLength (PMQLONG) - entrada/salida

Puntero al campo de longitud de datos.

CompCode (MQLONG)-entrada/salida

Código de terminación, cuyos valores válidos son:

MQCC_OK

Realización satisfactoria.

MQCC_WARNING

Finalización parcial.

MQCC_FAILED

Llamada fallida

Razón (MQLONG)-entrada/salida

Código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK, el único valor válido es:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED o MQCC_WARNING, la función de salida puede establecer el campo de código de razón a cualquier valor de MQRC_* válido.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

MQAXP	ExitParms;	/* Exit parameter structure */
MQAXC	ExitContext;	/* Exit context structure */
MQHCONN	Hconn;	/* Connection handle */
MQHOBJ	Hobj;	/* Object handle */
PMQMD	pMsgDesc;	/* Ptr to message descriptor */
PMQPMO	pGetMsgOpts;	/* Ptr to get message options */
MQLONG	BufferLength;	/* Message buffer length */

PMQBYTE	pBuffer;	/* Ptr to message buffer */
PMQLONG	pDataLength;	/* Ptr to data length field */
MQLONG	CompCode;	/* Completion code */
MQLONG	Reason;	/* Reason code */

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
MQ_GET_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &pMsgDesc,
             &pGetMsgOpts, &BufferLength, &pBuffer, &pDataLength,
             &CompCode, &Reason)
```

La salida debe coincidir con el siguiente prototipo de función C:

```
void MQENTRY MQ_GET_EXIT (
PMQAXP      pExitParms,      /* Address of exit parameter structure */
PMQAXC      pExitContext,    /* Address of exit context structure */
PMQHCONN    pHconn,         /* Address of connection handle */
PMQHOBJS    pHobj,          /* Address of object handle */
PPMQMD      ppMsgDesc,      /* Address of ptr to message descriptor */
PPMQGMO     ppGetMsgOpts,    /* Address of ptr to get message options */
PMQLONG     pBufferLength,   /* Address of message buffer length */
PPMQBYTE    ppBuffer,       /* Address of ptr to message buffer */
PPMQLONG    ppDataLength,    /* Address of ptr to data length field */
PMQLONG     pCompCode,       /* Address of completion code */
PMQLONG     pReason);        /* Address of reason code qualifying
                             completion code */
```

Notas de uso

1. La interfaz de función MQ_GET_EXIT descrita aquí se utiliza tanto para la función de salida MQXF_GET como para la función de salida [“MQXF_DATA_CONV_ON_GET”](#) en la página 1617 .

Se definen puntos de entrada separados para estas dos funciones de salida, por lo que para interceptar *ambos* la llamada MQXEP debe utilizarse dos veces; para esta llamada utilice el identificador de función MQXF_GET.

Puesto que la interfaz MQ_GET_EXIT es la misma para MQXF_GET y MQXF_DATA_CONV_ON_GET, se puede utilizar una única función de salida para ambos; el campo *Function* de la estructura MQAXP indica qué función de salida se ha invocado. De forma alternativa, se puede utilizar la llamada MQXEP para registrar distintas funciones de salida para los dos casos.

MQXF_DATA_CONV_ON_GET

El identificador de función MQXF_DATA_CONV_ON_GET se utiliza con MQ_GET_EXIT.

Consulte [MQ_GET_EXIT](#) para obtener información sobre la interfaz para esta llamada y una declaración de lenguaje C de ejemplo.

Notas de uso

Si está registrado, se llama a este punto de entrada cuando llegan mensajes a la aplicación pero antes de que se haya producido cualquier conversión de datos. Esto puede ser útil si la salida de API necesita realizar el proceso, como el descifrado o la descompresión, antes de que el mensaje se pase a la conversión de datos. La salida puede, si es necesario, hacer que se elude la conversión de datos devolviendo MQXCC_SUPPRESS_FUNCTION; para obtener más información, consulte la estructura MQAXP .

El registro para este punto de entrada en un cliente tiene el efecto de hacer que la conversión de datos se realice localmente en la máquina cliente. Por lo tanto, para que el funcionamiento sea correcto, es posible que sea necesario instalar las salidas de conversión de aplicaciones en el cliente. Tenga en cuenta que MQXF_DATA_CONV_ON_GET también se utiliza para el consumo asíncrono.

Cuando utilice la llamada MQ_GET_EXIT, utilice MQXF_DATA_CONV_ON_GET, con la razón de salida MQXR_BEFORE, para registrar una función de salida de conversión de datos MQGET *anterior a* .

No hay ninguna función de salida MQXR_AFTER para MQXF_DATA_CONV_ON_GET; la función de salida MQXR_AFTER para MQXF_GET proporciona la capacidad necesaria para el proceso de salida después de la conversión de datos.

Se definen puntos de entrada separados para la llamada MQ_GET_EXIT, por lo que para interceptar *ambas* funciones de salida, la llamada MQXEP debe utilizarse dos veces; para esta llamada, utilice el identificador de función MQXF_DATA_CONV_ON_GET.

Puesto que la interfaz MQ_GET_EXIT es la misma para MQXF_GET y MQXF_DATA_CONV_ON_GET, se puede utilizar una única función de salida para ambos; el campo *Function* de la estructura MQAXP indica qué función de salida se ha invocado. De forma alternativa, se puede utilizar la llamada MQXEP para registrar distintas funciones de salida para los dos casos.

Inicialización-MQ_INIT_EXIT

MQ_INIT_EXIT proporciona inicialización de nivel de conexión, indicada estableciendo ExitReason en MQAXP en MQXR_CONNECTION.

Durante la inicialización, tenga en cuenta lo siguiente:

- La función MQ_INIT_EXIT llama a MQXEP para registrar los verbos de la API IBM MQ y los puntos ENTRY y EXIT en los que está interesado.
- No es necesario que las salidas intercepten todos los verbos de la API de IBM MQ . Las funciones de salida sólo se invocan si se ha registrado un interés.
- El almacenamiento que va a utilizar la salida se puede adquirir al inicializarlo.
- Si falla una llamada a esta función, la llamada MQCONN o MQCONNX que la ha invocado también falla con un CompCode y una razón que dependen del valor del campo ExitResponse en MQAXP.
- Una salida MQ_INIT_EXIT no debe emitir llamadas de API de IBM MQ , porque el entorno correcto no se ha configurado en este momento.
- Si un MQ_INIT_EXIT falla con MQXCC_FAILED, el gestor de colas vuelve de la llamada MQCONN o MQCONNX que le ha llamado con MQCC_FAILED y MQRC_API_EXIT_ERROR.
- Si el gestor de colas encuentra un error al inicializar el entorno de ejecución de la función de salida de API antes de invocar la primera MQ_INIT_EXIT, el gestor de colas devuelve la llamada MQCONN o MQCONNX que ha invocado MQ_INIT_EXIT con MQCC_FAILED y MQRC_API_EXIT_INIT_ERROR.

La interfaz con MQ_INIT_EXIT es:

```
MQ_INIT_EXIT (&ExitParms, &ExitContext, &CompCode, &Reason)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida

Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida

Salir de la estructura de contexto.

CompCode (MQLONG)-entrada/salida

Puntero a código de terminación, cuyos valores válidos son:

MQCC_OK

Realización satisfactoria.

MQCC_WARNING

Finalización parcial.

MQCC_FAILED

Llamada fallida

Razón (MQLONG)-entrada/salida

Puntero al código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK, el único valor válido es:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED o MQCC_WARNING, la función de salida puede establecer el campo de código de razón a cualquier valor de MQRC_* válido.

CompCode y Reason devueltos a la aplicación dependen del valor del campo ExitResponse en MQAXP.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

MQAXP	ExitParms;	/* Exit parameter structure */
MQAXC	ExitContext;	/* Exit context structure */
MQLONG	CompCode;	/* Completion code */
MQLONG	Reason;	/* Reason code */

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
MQ_INIT_EXIT (&ExitParms, &ExitContext, &CompCode, &Reason)
```

La salida debe coincidir con el siguiente prototipo de función C:

```
void MQENTRY MQ_INIT_EXIT (
    PMQAXP      pExitParms,      /* Address of exit parameter structure */
    PMQAXC      pExitContext,    /* Address of exit context structure */
    PMQLONG     pCompCode,       /* Address of completion code */
    PMQLONG     pReason);        /* Address of reason code qualifying
                                completion code */
```

Notas de uso

1. La función MQ_INIT_EXIT puede emitir la llamada MQXEP para registrar las direcciones de las funciones de salida para las llamadas MQ concretas que se van a interceptar. No es necesario interceptar todas las llamadas de MQ o interceptar las llamadas MQXR_BEFORE y MQXR_AFTER. Por ejemplo, una suite de salida podría elegir interceptar sólo la llamada MQXR_BEFORE de MQPUT.
2. El almacenamiento que deben utilizar las funciones de salida en la suite de salida puede ser adquirido por la función MQ_INIT_EXIT. De forma alternativa, las funciones de salida pueden adquirir almacenamiento cuando se invocan, como y cuando sea necesario. Sin embargo, todo el almacenamiento debe liberarse antes de que termine la suite de salida; la función MQ_TERM_EXIT puede liberar el almacenamiento o una función de salida invocada anteriormente.
3. Si MQ_INIT_EXIT devuelve MQXCC_FAILED en el campo ExitResponse de MQAXP, o falla de alguna otra forma, la llamada MQCONN o MQCONNX que ha hecho que se invoque MQ_INIT_EXIT también falla, con los parámetros **CompCode** y **Reason** establecidos en los valores adecuados.
4. Una función MQ_INIT_EXIT no puede emitir llamadas MQ que no sean MQXEP.

Consultar-MQ_INQ_EXIT

MQ_INQ_EXIT proporciona una función de salida de consulta para realizar *antes y después del proceso de la llamada MQINQ de* . Utilice el identificador de función MQXF_INQ con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar *antes y después de* las funciones de salida de llamada MQINQ.

La interfaz para esta función es:

```
MQ_INQ_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &SelectorCount,
             &pSelectors, &IntAttrCount, &pIntAttrs, &CharAttrLength,
             &pCharAttrs, &CompCode, &Reason)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida

Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida

Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada

Descriptor de conexión.

Hobj (MQHOBJ)-entrada

El manejador del objeto.

SelectorCount (MQLONG)-entrada

Recuento de selectores

pSelectors (PMQLONG)-entrada/salida

Puntero a matriz de valores de selector.

IntAttrCount (MQLONG) - entrada

Recuento de atributos enteros.

pIntAttrs (PMQLONG) - entrada/salida

Puntero a matriz de valores de atributo de entero.

CharAttrLength (MQLONG) - entrada/salida

Longitud de matriz de atributos de caracteres.

pCharAttrs (PMQCHAR) - entrada/salida

Puntero a matriz de atributos de carácter.

CompCode (MQLONG)-entrada/salida

Código de terminación, cuyos valores válidos son:

MQCC_OK

Realización satisfactoria.

MQCC_WARNING

Finalización parcial.

MQCC_FAILED

Llamada fallida

Razón (MQLONG)-entrada/salida

Código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK, el único valor válido es:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED o MQCC_WARNING, la función de salida puede establecer el campo de código de razón a cualquier valor de MQRC_* válido.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
MQAXP      ExitParms;          /* Exit parameter structure */
MQAXC      ExitContext;        /* Exit context structure */
MQHCONN    Hconn;              /* Connection handle */
MQHOBJ     Hobj;               /* Object handle */
MQLONG     SelectorCount;      /* Count of selectors */
PMQLONG    pSelectors;         /* Ptr to array of attribute selectors */
MQLONG     IntAttrCount;       /* Count of integer attributes */
PMQLONG    pIntAttrs;          /* Ptr to array of integer attributes */
MQLONG     CharAttrLength;     /* Length of char attributes array */
PMQCHAR    pCharAttrs;         /* Ptr to character attributes */
MQLONG     CompCode;           /* Completion code */
MQLONG     Reason;             /* Reason code qualifying completion code */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
MQ_INQ_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &SelectorCount,
             &pSelectors, &IntAttrCount, &pIntAttrs, &CharAttrLength,
             &pCharAttrs, &CompCode, &Reason)
```


La salida debe coincidir con el siguiente prototipo de función C:

```
void MQENTRY MQ_INQ_EXIT (
    PMQAXP    pExitParms,      /* Address of exit parameter structure */
    PMQAXC    pExitContext,    /* Address of exit context structure */
    PMQHCONN  pHConn,         /* Address of connection handle */
    PMQHOBJS  pHobj,          /* Address of object handle */
    PMQLONG   pSelectorCount,  /* Address of selector count */
    PPMQLONG  ppSelectors,     /* Address of ptr to array of selectors */
    PMQLONG   pIntAttrCount;   /* Address of count of integer attributes */
    PMQLONG   ppIntAttrs,      /* Address of ptr to array of integer attributes */
    PMQLONG   pCharAttrLength, /* Address of character attribute length */
    PPMQCHAR  ppCharAttrs,     /* Address of ptr to character attributes array */
    PMQLONG   pCompCode,       /* Address of completion code */
    PMQLONG   pReason;         /* Address of reason code qualifying completion
                                code */
)
```

Abrir-MQ_OPEN_EXIT

MQ_OPEN_EXIT proporciona una función de salida abierta para realizar *antes* y *después* del proceso de llamada MQOPEN. Utilice el identificador de función MQXF_OPEN con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar *antes* y *después* de las funciones de salida de llamada MQOPEN.

La interfaz para esta función es

```
MQ_OPEN_EXIT (&ExitParms, &ExitContext, &Hconn, &pObjDesc, &Options,
              &pHobj, &CompCode, &Reason)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida

Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida

Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada

Descriptor de conexión.

pObjDesc (PMQOD) - entrada/salida

Puntero al descriptor de objeto.

Opciones (MQLONG)-entrada/salida

Opciones de apertura.

pHobj (PMQHOBJS)-entrada

Puntero al descriptor de contexto de objeto.

CompCode (MQLONG)-entrada/salida

Código de terminación, cuyos valores válidos son:

MQCC_OK

Realización satisfactoria.

MQCC_WARNING

Terminación parcial

MQCC_FAILED

Llamada fallida

Razón (MQLONG)-entrada/salida

Código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK, el único valor válido es:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED o MQCC_WARNING, la función de salida puede establecer el campo de código de razón a cualquier valor de MQRC_* válido.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
MQAXP      ExitParms;      /* Exit parameter structure */
MQAXC      ExitContext;    /* Exit context structure */
MQHCONN    Hconn;         /* Connection handle */
PMQOD      pObjDesc;      /* Ptr to object descriptor */
MQLONG     Options;       /* Open options */
MQHOBJS    pHobj;         /* Ptr to object handle */
MQLONG     CompCode;      /* Completion code */
MQLONG     Reason;        /* Reason code */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
MQ_OPEN_EXIT (&ExitParms, &ExitContext, &Hconn, &pObjDesc, &Options,
              &pHobj, &CompCode, &Reason);
```

La salida debe coincidir con el siguiente prototipo de función C:

```
void MQENTRY MQ_OPEN_EXIT (
PMQAXP      pExitParms,    /* Address of exit parameter structure */
PMQAXC      pExitContext,  /* Address of exit context structure */
PMQHCONN    pHconn,       /* Address of connection handle */
PPMQOD      ppObjDesc,    /* Address of ptr to object descriptor */
PMQLONG     pOptions,      /* Address of open options */
PPMQHOBJS   ppHobj,       /* Address of ptr to object handle */
PMQLONG     pCompCode,    /* Address of completion code */
PMQLONG     pReason);     /* Address of reason code qualifying
                           completion code */
```

Put-MQ_PUT_EXIT

MQ_PUT_EXIT proporciona una función de salida de transferencia para realizar *antes* y *después* del proceso de llamada MQPUT. Utilice el identificador de función MQXF_PUT con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar *antes* y *después* de las funciones de salida de llamada MQPUT.

La interfaz para esta función es:

```
MQ_PUT_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &pMsgDesc,
             &pPutMsgOpts, &BufferLength, &pBuffer, &CompCode, &Reason)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida

Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida

Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada

Descriptor de conexión.

Hobj (MQHOBJS)-entrada/salida

El manejador del objeto.

pMsgDesc (PMQMD) - entrada/salida

Puntero al descriptor de mensaje.

pPutMsgOpts (PMQPMO) - entrada/salida

Puntero para colocar opciones de mensaje.

BufferLength (MQLONG)-entrada/salida

Longitud del almacenamiento intermedio de mensajes.

pBuffer (PMQBYTE)-entrada/salida

Puntero al almacenamiento intermedio de mensajes.

CompCode (MQLONG)-entrada/salida

Código de terminación, cuyos valores válidos son:

MQCC_OK

Realización satisfactoria.

MQCC_WARNING

Finalización parcial.

MQCC_FAILED

Llamada fallida

Razón (MQLONG)-entrada/salida

Código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK, el único valor válido es:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED o MQCC_WARNING, la función de salida puede establecer el campo de código de razón a cualquier valor de MQRC_* válido.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
MQAXP      ExitParms;      /* Exit parameter structure */
MQAXC      ExitContext;    /* Exit context structure */
MQHCONN    Hconn;          /* Connection handle */
MQHOBJ     Hobj;           /* Object handle */
MQMD       pMsgDesc;       /* Ptr to message descriptor */
MQPMO      pPutMsgOpts;    /* Ptr to put message options */
MQLONG     BufferLength;    /* Message buffer length */
MQBYTE     pBuffer;        /* Ptr to message data */
MQLONG     CompCode;       /* Completion code */
MQLONG     Reason;         /* Reason code */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
MQ_PUT_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &pMsgDesc,
             &pPutMsgOpts, &BufferLength, &pBuffer, &CompCode, &Reason)
```

La salida debe coincidir con el siguiente prototipo de función C:

```
void MQENTRY MQ_PUT_EXIT (
PMQAXP      pExitParms,    /* Address of exit parameter structure */
PMQAXC      pExitContext,  /* Address of exit context structure */
PMQHCONN    pHconn,       /* Address of connection handle */
PMQHOBJ     pHobj,        /* Address of object handle */
PPMQMD      ppMsgDesc,     /* Address of ptr to message descriptor */
PPMQPMO     ppPutMsgOpts,  /* Address of ptr to put message options */
PMQLONG     pBufferLength, /* Address of message buffer length */
PPMQBYTE    ppBuffer,      /* Address of ptr to message buffer */
PMQLONG     pCompCode,     /* Address of completion code */
PMQLONG     pReason);      /* Address of reason code qualifying
                             completion code */
```

Notas de uso

- Los mensajes de informe generados por el gestor de colas se saltan el proceso de llamada normal. Como resultado, la función MQ_PUT_EXIT o la función MQPUT1 no pueden interceptar dichos mensajes. Sin embargo, los mensajes de informe generados por el agente de canal de mensajes se procesan normalmente y, por lo tanto, pueden ser interceptados por la función MQ_PUT_EXIT o la función MQ_PUT1_EXIT. Para asegurarse de interceptar todos los mensajes de informe generados por el MCA, deben utilizarse MQ_PUT_EXIT y MQ_PUT1_EXIT.

Put1 - MQ_PUT1_EXIT

MQ_PUT1_EXIT proporciona una función de salida *put one message only* para realizar *antes y después del proceso de llamada* MQPUT1. Utilice el identificador de función MQXF_PUT1 con las razones de salida

MQXR_BEFORE y MQXR_AFTER para registrar *antes y después de las funciones de salida de llamada* MQPUT1 .

La interfaz para esta función es:

```
MQ_PUT1_EXIT (&ExitParms, &ExitContext, &Hconn, &pObjDesc, &pMsgDesc,  
&pPutMsgOpts, &BufferLength, &pBuffer, &CompCode, &Reason)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida

Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida

Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada

Descriptor de conexión.

pObjDesc (PMQOD) - entrada/salida

Puntero al descriptor de objeto.

pMsgDesc (PMQMD) - entrada/salida

Puntero al descriptor de mensaje.

pPutMsgOpts (PMQPMO) - entrada/salida

Puntero para colocar opciones de mensaje.

BufferLength (MQLONG)-entrada/salida

Longitud del almacenamiento intermedio de mensajes.

pBuffer (PMQBYTE)-entrada/salida

Puntero al almacenamiento intermedio de mensajes.

CompCode (MQLONG)-entrada/salida

Código de terminación, cuyos valores válidos son:

MQCC_OK

Realización satisfactoria.

MQCC_WARNING

Finalización parcial.

MQCC_FAILED

Llamada fallida

Razón (MQLONG)-entrada/salida

Código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK, el único valor válido es:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED o MQCC_WARNING, la función de salida puede establecer el campo de código de razón a cualquier valor de MQRC_* válido.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

MQAXP	ExitParms;	/* Exit parameter structure */
MQAXC	ExitContext;	/* Exit context structure */
MQHCONN	Hconn;	/* Connection handle */
PMQOD	pObjDesc;	/* Ptr to object descriptor */
PMQMD	pMsgDesc;	/* Ptr to message descriptor */
PMQPMO	pPutMsgOpts;	/* Ptr to put message options */
MQLONG	BufferLength;	/* Message buffer length */
PMQBYTE	pBuffer;	/* Ptr to message data */
MQLONG	CompCode;	/* Completion code */
MQLONG	Reason;	/* Reason code */

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
MQ_PUT1_EXIT (&ExitParms, &ExitContext, &Hconn, &pObjDesc, &pMsgDesc,  
              &pPutMsgOpts, &BufferLength, &pBuffer, &CompCode, &Reason)
```

La salida debe coincidir con el siguiente prototipo de función C:

```
void MQENTRY MQ_PUT1_EXIT (  
    PMQAXP      pExitParms,      /* Address of exit parameter structure */  
    PMQAXC      pExitContext,    /* Address of exit context structure */  
    PMQHCONN    pHconn,          /* Address of connection handle */  
    PPMQOD      ppObjDesc,       /* Address of ptr to object descriptor */  
    PPMQMD      ppMsgDesc,       /* Address of ptr to message descriptor */  
    PPMQPMO     ppPutMsgOpts,    /* Address of ptr to put message options */  
    PMQLONG     pBufferLength,   /* Address of message buffer length */  
    PPMQBYTE    pBuffer,         /* Address of ptr to message buffer */  
    PMQLONG     pCompCode,       /* Address of completion code */  
    PMQLONG     pReason);        /* Address of reason code qualifying  
                                completion code */
```

Establecer-MQ_SET_EXIT

MQ_SET_EXIT proporciona una función de salida de conjunto para realizar *antes y después del proceso de la llamada MQSET* de . Utilice el identificador de función MQXF_SET con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar *antes y después de* las funciones de salida de llamada MQSET.

La interfaz para esta función es:

```
MQ_SET_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &SelectorCount,  
            &pSelectors, &IntAttrCount, &pIntAttrs, &CharAttrLength,  
            &pCharAttr, &CompCode, &Reason)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida

Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida

Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada

Descriptor de conexión.

Hobj (MQHOBJ)-entrada

El manejador del objeto.

SelectorCount (MQLONG)-entrada

Recuento de selectores

pSelectors (PMQLONG)-entrada/salida

Puntero a matriz de valores de selector.

IntAttrCount (MQLONG) - entrada

Recuento de atributos enteros.

pIntAttrs (PMQLONG) - entrada/salida

Puntero a matriz de valores de atributo de entero.

CharAttrLength (MQLONG) - entrada/salida

Longitud de matriz de atributos de caracteres.

pCharAttrs (PMQCHAR) - entrada/salida

Puntero a valores de atributo de carácter.

CompCode (MQLONG)-entrada/salida

Código de terminación, cuyos valores válidos son:

MQCC_OK

Realización satisfactoria.

MQCC_WARNING

Finalización parcial.

MQCC_FAILED

Llamada fallida

Razón (MQLONG)-entrada/salida

Código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK, el único valor válido es:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED o MQCC_WARNING, la función de salida puede establecer el campo de código de razón a cualquier valor de MQRC_* válido.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
MQAXP      ExitParms;          /* Exit parameter structure */
MQAXC      ExitContext;        /* Exit context structure */
MQHCONN     Hconn;             /* Connection handle */
MQHOBJ      Hobj;              /* Object handle */
MQLONG      SelectorCount;     /* Count of selectors */
PMQLONG     pSelectors;        /* Ptr to array of attribute selectors */
MQLONG      IntAttrCount;      /* Count of integer attributes */
PMQLONG     pIntAttrs;         /* Ptr to array of integer attributes */
MQLONG      CharAttrLength;    /* Length of char attributes array */
PMQCHAR     pCharAttrs;        /* Ptr to character attributes */
MQLONG      CompCode;          /* Completion code */
MQLONG      Reason;            /* Reason code qualifying completion code */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
MQ_SET_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &SelectorCount,
             &pSelectors, &IntAttrCount, &pIntAttrs, &CharAttrLength,
             &pCharAttrs, &CompCode, &Reason)
```

La salida debe coincidir con el siguiente prototipo de función C:

```
void MQENTRY MQ_SET_EXIT (
PMQAXP      pExitParms,        /* Address of exit parameter structure */
PMQAXC      pExitContext,      /* Address of exit context structure */
PMQHCONN     pHconn,           /* Address of connection handle */
PMQHOBJ      pHobj,            /* Address of object handle */
PMQLONG      pSelectorCount,    /* Address of selector count */
PPMQLONG     ppSelectors,       /* Address of ptr to array of selectors */
PMQLONG      pIntAttrCount;     /* Address of count of integer attributes */
PPMQLONG     ppIntAttrs,        /* Address of ptr to array of integer attributes */
PMQLONG      pCharAttrLength,   /* Address of character attribute length */
PPMQCHAR     ppCharAttrs,       /* Address of ptr to character attributes array */
PMQLONG      pCompCode,         /* Address of completion code */
PMQLONG      pReason);          /* Address of reason code qualifying completion
                                code */
```

Estado-MQ_STAT_EXIT

MQ_STAT_EXIT proporciona una función de salida de estado para realizar *antes* y *después* del proceso de llamada MQSTAT. Utilice el identificador de función MQXF_STAT con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar *antes* y *después* de las funciones de salida de llamada MQSTAT.

La interfaz para esta función es:

```
MQ_STAT_EXIT (&ExitParms, &ExitContext, &Hconn, &Type, &pStatus
             &CompCode, &Reason)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida

Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida

Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada

Descriptor de conexión.

Tipo (MQLONG)-entrada

Tipo de información de estado a recuperar.

pStatus (PMQSTS)-salida

Puntero al almacenamiento intermedio de estado.

CompCode (MQLONG)-entrada/salida

Código de terminación, cuyos valores válidos son:

MQCC_OK

Realización satisfactoria.

MQCC_WARNING

Finalización parcial.

MQCC_FAILED

Llamada fallida

Razón (MQLONG)-entrada/salida

Código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK, el único valor válido es:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED o MQCC_WARNING, la función de salida puede establecer el campo de código de razón a cualquier valor de MQRC_* válido.

Invocación de lenguaje C

La salida debe coincidir con el siguiente prototipo de función C:

```
void MQENTRY MQ_STAT_EXIT (
PMQAXP    pExitParms,      /* Address of exit parameter structure */
PMQAXC    pExitContext,    /* Address of exit context structure */
PMQHCONN  pHconn,         /* Address of connection handle */
PMQLONG   pType,           /* Address of status type */
PPMQSTS   ppStatus,        /* Address of status buffer */
PMQLONG   pCompCode,       /* Address of completion code */
PMQLONG   pReason);        /* Address of reason code qualifying completion
                             code */
```

Terminación-MQ_TERM_EXIT

MQ_TERM_EXIT proporciona terminación a nivel de conexión, registrada con un identificador de función de MQXF_TERM y ExitReason MQXR_CONNECTION. Si está registrado, se llama a MQ_TERM_EXIT una vez para cada solicitud de desconexión.

Como parte de la terminación, el almacenamiento que ya no necesita la salida se puede liberar y se puede realizar cualquier limpieza necesaria.

Si un MQ_TERM_EXIT falla con MQXCC_FAILED, el gestor de colas devuelve el MQDISC que lo ha llamado con MQCC_FAILED y MQRC_API_EXIT_ERROR.

Si el gestor de colas encuentra un error al terminar el entorno de ejecución de la función de salida de API después de invocar el último MQ_TERM_EXIT, el gestor de colas devuelve la llamada MQDISC que ha invocado MQ_TERM_EXIT con MQCC_FAILED y MQRC_API_EXIT_TERM_ERROR

La interfaz para esta función es:

```
MQ_TERM_EXIT (&ExitParms, &ExitContext, &CompCode, &Reason)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida

Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida

Salir de la estructura de contexto.

CompCode (MQLONG)-entrada/salida

Código de terminación, cuyos valores válidos son:

MQCC_OK

Realización satisfactoria.

MQCC_FAILED

Llamada fallida

Razón (MQLONG)-entrada/salida

Código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK, el único valor válido es:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED, la función de salida puede establecer el campo de código de razón en cualquier valor MQRC_ * válido.

CompCode y Reason devueltos a la aplicación dependen del valor del campo ExitResponse en MQAXP.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
MQAXP      ExitParms;      /* Exit parameter structure */
MQAXC      ExitContext;    /* Exit context structure */
MQLONG     CompCode;       /* Completion code */
MQLONG     Reason;         /* Reason code */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
MQ_TERM_EXIT (&ExitParms, &ExitContext, &CompCode, &Reason)
```

La salida debe coincidir con el siguiente prototipo de función C:

```
void MQENTRY MQ_TERM_EXIT (
PMQAXP      pExitParms,    /* Address of exit parameter structure */
PMQAXC      pExitContext,  /* Address of exit context structure */
PMQLONG     pCompCode,     /* Address of completion code */
PMQLONG     pReason);      /* Address of reason code qualifying
                             completion code */
```

Notas de uso

1. La función MQ_TERM_EXIT es opcional. No es necesario que una suite de salida registre una salida de terminación si no hay que realizar ningún proceso de terminación.

Si las funciones pertenecientes a la suite de salida adquieren recursos durante la conexión, una función MQ_TERM_EXIT es un punto conveniente en el que liberar esos recursos, por ejemplo, liberando almacenamiento obtenido dinámicamente.
2. Si se registra una función MQ_TERM_EXIT cuando se emite la llamada MQDISC, la función de salida se invoca después de que se hayan invocado todas las funciones de salida MQDISC.
3. Si MQ_TERM_EXIT devuelve MQXCC_FAILED en el campo ExitResponse de MQAXP, o falla de alguna otra forma, la llamada MQDISC que ha hecho que se invoque MQ_TERM_EXIT también falla, con los parámetros **CompCode** y **Reason** establecidos en los valores adecuados.

Registrar suscripción-MQ_SUB_EXIT

MQ_SUB_EXIT proporciona una función de salida para realizar *antes* y *después* del proceso de reregistro de suscripción. Utilice el identificador de función MQXF_SUB con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar *antes* y *después* de las funciones de salida de registrationcall de suscripción.

La interfaz para esta función es:

```
MQ_SUB_EXIT (&ExitParms, &ExitContext, &Hconn, &pSubDesc, &pHobj, &pHsub, &CompCode, &Reason)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida

Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida

Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada/salida

Descriptor de conexión.

pSubDesc - entrada/salida

Matriz de selectores de atributos.

pHobj -entrada/salida

Descriptor de contexto del objeto

pHsub (MQHOBJ) entrada/salida

Manejador de suscripciones

CompCode (MQLONG)-entrada/salida

Código de terminación, cuyos valores válidos son:

MQCC_OK

Realización satisfactoria.

MQCC_WARNING

Finalización parcial.

MQCC_FAILED

Llamada fallida

Razón (MQLONG)-entrada/salida

Código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK, el único valor válido es:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED o MQCC_WARNING, la función de salida puede establecer el campo de código de razón a cualquier valor de MQRC_* válido.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
MQAXP      ExitParms;      /* Exit parameter structure */
MQAXC      ExitContext;    /* Exit context structure */
MQHCONN    Hconn;          /* Connection handle */
PMQSD      pSubDesc;       /* Subscription descriptor */
PMQHOBJS   pHobj;          /* Object Handle */
PMQHOBJS   pHsub;          /* Subscription handle */
MQLONG     CompCode;       /* Completion code */
MQLONG     Reason;         /* Reason code qualifying completion code */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
MQ_SUB_EXIT (&ExitParms, &ExitContext, &Hconn, &pSubDesc, &pHobj, &pHsub,
             &CompCode, &Reason);
```

La salida debe coincidir con el siguiente prototipo de función C:

```
PMQAXP    pExitParms;    /* Exit parameter structure */
PMQAXC    pExitContext;  /* Exit context structure */
PMQHCONN  pHconn;        /* Connection handle */
PPMQSD    ppSubDesc;     /* Subscription descriptor */
PPMQHOBj  ppHobj;        /* Object Handle */
PPMQHOBj  ppHsub;        /* Subscription handle */
PMQLONG    pCompCode;    /* Completion code */
PMQLONG    pReason;      /* Reason code qualifying completion code */
```

Solicitud de suscripción-MQ_SUBRQ_EXIT

MQ_SUBRQ_EXIT proporciona una función de salida de solicitud de suscripción para realizar *antes* y *después* del proceso de solicitud de suscripción. Utilice el identificador de función MQXF_SUBRQ con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar las funciones de salida de llamada de solicitud de suscripción *before* y *after*.

La interfaz para esta función es:

```
MQ_SUBRQ_EXIT (&ExitParms, &ExitContext, &Hconn, &pHsub, &Action, &pSubRqOpts,
               &CompCode, &Reason)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida

Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida

Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada/salida

Descriptor de conexión.

pHsub (MQHOBj) entrada/salida

Manejador de suscripciones

Entrada/salida de acción (MQLONG)

Acción

pSubRqOpts (MQSRO) entrada/salida

CompCode (MQLONG)-entrada/salida

Código de terminación, cuyos valores válidos son:

MQCC_OK

Realización satisfactoria.

MQCC_WARNING

Finalización parcial.

MQCC_FAILED

Llamada fallida

Razón (MQLONG)-entrada/salida

Código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK, el único valor válido es:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED o MQCC_WARNING, la función de salida puede establecer el campo de código de razón a cualquier valor de MQRC_* válido.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
MQAXP    ExitParms;    /* Exit parameter structure */
MQAXC    ExitContext;  /* Exit context structure */
```

```

MQHCONN  Hconn;          /* Connection handle */
PMQLONG  pHsub;          /* Subscription handle */
MQLONG   Action;         /* Action */
PMQSRO   pSubRqOpts;     /* Subscription Request Options */
MQLONG   CompCode;       /* Completion code */
MQLONG   Reason;         /* Reason code qualifying completion code */

```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```

MQ_SUBRQ_EXIT (&ExitParms, &ExitContext, &Hconn, &pHsub, &Action, &pSubRqOpts,
               &CompCode, &Reason);

```

La salida debe coincidir con el siguiente prototipo de función C:

```

void MQENTRY MQ_SUBRQ_EXIT (
PMQAXP   pExitParms,      /* Address of exit parameter structure */
PMQAXC   pExitContext,    /* Address of exit context structure */
PMQHCONN pHconn,          /* Address of connection handle */
PPMQHOBJ ppHsub;          /* Address of Subscription handle */
PMQLONG  pAction;          /* Address of Action */
PPMQSRO  ppSubRqOpts;     /* Address of Subscription Request Options */
PMQLONG  pCompCode;        /* Address of completion code */
PMQLONG  pReason;          /* Address of reason code qualifying completion
                           code */

```

xa_close-XA_CLOSE_EXIT

XA_CLOSE_EXIT proporciona una función de salida xa_close para realizar antes y después del proceso xa_close. Utilice el identificador de función MQXF_XACLOSE con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar las funciones de salida de llamada antes y después de xa_close.

La interfaz para esta función es:

```

XA_CLOSE_EXIT (&ExitParms, &ExitContext, &Hconn, &pXa_info, &Rmid, &Flags, &XARetCode)

```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida

Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida

Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada

Descriptor de conexión.

pXa_info (PMQCHAR) - entrada/salida

Información del gestor de recursos específico de la instancia.

Rmid (MQLONG)-entrada/salida

Identificador del gestor de recursos.

Distintivos (MQLONG)-entrada/salida

Opciones del gestor de recursos.

XARetCode (MQLONG)-entrada/salida

Respuesta a una llamada de « XA ».

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```

MQAXP   ExitParms;      /* Exit parameter structure */
MQAXC   ExitContext;    /* Exit context structure */
MQHCONN Hconn;          /* Connection handle */
PMQCHAR pXa_info;       /* Instance-specific RM info */
MQLONG  Rmid;           /* Resource manager identifier */
MQLONG  Flags;          /* Resource manager options*/
MQLONG  XARetCode;      /* Response from XA call */

```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
XA_CLOSE_EXIT (&ExitParms, &ExitContext, &Hconn, &pXa_info, &Rmid, &Flags, &XARetCode);
```

La salida debe coincidir con el siguiente prototipo de función C:

```
typedef void MQENTRY XA_CLOSE_EXIT (
    PMQAXP    pExitParms,    /* Address of exit parameter structure */
    PMQAXC    pExitContext,  /* Address of exit context structure */
    PMQHCONN  pHconn,        /* Address of connection handle */
    PPMQCHAR  ppXa_info,     /* Address of instance-specific RM info */
    PMQLONG   pRmid,         /* Address of resource manager identifier */
    PMQLONG   pFlags,        /* Address of resource manager options */
    PMQLONG   pXARetCode);   /* Address of response from XA call */
```

xa_commit-XA_COMMIT_EXIT

XA_COMMIT_EXIT proporciona una función de salida xa_commit para realizar antes y después del proceso xa_commit. Utilice el identificador de función MQXF_XACOMMIT con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar las funciones de salida de llamada antes y después de xa_commit.

La interfaz para esta función es:

```
XA_COMMIT_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida

Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida

Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada

Descriptor de conexión.

pXID (MQPTR)-entrada/salida

ID de rama de transacción.

Rmid (MQLONG)-entrada/salida

Identificador del gestor de recursos.

Distintivos (MQLONG)-entrada/salida

Opciones del gestor de recursos.

XARetCode (MQLONG)-entrada/salida

Respuesta a una llamada de « XA ».

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
MQAXP    ExitParms;    /* Exit parameter structure */
MQAXC    ExitContext;  /* Exit context structure */
MQHCONN  Hconn;        /* Connection handle */
MQPTR    pXID;         /* Transaction branch ID */
MQLONG   Rmid;         /* Resource manager identifier */
MQLONG   Flags;        /* Resource manager options */
MQLONG   XARetCode;    /* Response from XA call */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
XA_COMMIT_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode);
```

La salida debe coincidir con el siguiente prototipo de función C:

```
typedef void MQENTRY XA_COMMIT_EXIT (
    PMQAXP    pExitParms,    /* Address of exit parameter structure */
    PMQAXC    pExitContext,  /* Address of exit context structure */
    PMQHCONN  pHconn,        /* Address of connection handle */
    PMQPTR    ppXID,         /* Address of transaction branch ID */
    PMQLONG   pRmid,         /* Address of resource manager identifier */
    PMQLONG   pFlags,        /* Address of resource manager options */
    PMQLONG   pXARetCode);   /* Address of response from XA call */
```

```
PMQLONG  pFlags,          /* Address of resource manager options*/
PMQLONG  pXARetCode); /* Address of response from XA call */
```

xa_complete-XA_COMPLETE_EXIT

XA_COMPLETE_EXIT proporciona una función de salida *xa_complete* para realizar antes y después del proceso *xa_complete*. Utilice el identificador de función MQXF_XACOMPLETE con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar las funciones de salida de llamada antes y después de *xa_complete*.

La interfaz para esta función es:

```
XA_COMPLETE_EXIT (&ExitParms, &ExitContext, &Hconn, &pHandle, &pRetVal, &Rmid, &Flags,
&XARetCode)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida

Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida

Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada

Descriptor de conexión.

pHandle (PMQLONG)-entrada/salida

Puntero a operación asíncrona.

pRetVal (PMQLONG) - entrada/salida

Valor de retorno de la operación asíncrona.

Rmid (MQLONG)-entrada/salida

Identificador del gestor de recursos.

Distintivos (MQLONG)-entrada/salida

Opciones del gestor de recursos.

XARetCode (MQLONG)-entrada/salida

Respuesta a una llamada de « XA ».

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
MQAXP  ExitParms; /* Exit parameter structure */
MQAXC  ExitContext; /* Exit context structure */
MQHCONN Hconn; /* Connection handle */
PMQLONG pHandle; /* Ptr to asynchronous op */
PMQLONG pRetVal; /* Return value of async op */
MQLONG Rmid; /* Resource manager identifier */
MQLONG Flags; /* Resource manager options*/
MQLONG XARetCode; /* Response from XA call */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
XA_COMPLETE_EXIT (&ExitParms, &ExitContext, &Hconn, &pHandle, &pRetVal, &Rmid, &Flags,
&XARetCode);
```

La salida debe coincidir con el siguiente prototipo de función C:

```
typedef void MQENTRY XA_COMPLETE_EXIT (
PMQAXP  pExitParms, /* Address of exit parameter structure */
PMQAXC  pExitContext, /* Address of exit context structure */
PMQHCONN pHconn, /* Address of connection handle */
PPMQLONG ppHandle, /* Address of ptr to asynchronous op */
PPMQLONG ppRetVal, /* Address of return value of async op */
PMQLONG pRmid, /* Address of resource manager identifier */
PMQLONG pFlags, /* Address of resource manager options*/
PMQLONG pXARetCode); /* Address of response from XA call */
```

xa_end-XA_END_EXIT

XA_END_EXIT proporciona una función de salida xa_end para realizar antes y después del proceso xa_end. Utilice el identificador de función MQXF_XAEND con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar las funciones de salida de llamada antes y después de xa_end.

La interfaz para esta función es:

```
XA_END_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida

Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida

Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada

Descriptor de conexión.

pXID (MQPTR)-entrada/salida

ID de rama de transacción.

Rmid (MQLONG)-entrada/salida

Identificador del gestor de recursos.

Distintivos (MQLONG)-entrada/salida

Opciones del gestor de recursos.

XARetCode (MQLONG)-entrada/salida

Respuesta a una llamada de « XA ».

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
MQAXP   ExitParms;    /* Exit parameter structure */
MQAXC   ExitContext;  /* Exit context structure */
MQHCONN Hconn;        /* Connection handle */
MQPTR   pXID;         /* Transaction branch ID */
MQLONG  Rmid;         /* Resource manager identifier */
MQLONG  Flags;        /* Resource manager options*/
MQLONG  XARetCode;    /* Response from XA call */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
XA_END_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode);
```

La salida debe coincidir con el siguiente prototipo de función C:

```
typedef void MQENTRY XA_END_EXIT (
    PMQAXP pExitParms, /* Address of exit parameter structure */
    PMQAXC pExitContext, /* Address of exit context structure */
    PMQHCONN pHconn, /* Address of connection handle */
    PMQPTR ppXID, /* Address of transaction branch ID */
    PMQLONG pRmid, /* Address of resource manager identifier */
    PMQLONG pFlags, /* Address of resource manager options*/
    PMQLONG pXARetCode); /* Address of response from XA call */
```

xa_forget-XA_OLVIDO_SALIDA

XA_OLVI_EXIT proporciona una función de salida xa_olvidar para realizar antes y después del proceso xa_olvidar. Utilice el identificador de función MQXF_XAFORGET con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar las funciones de salida de llamada antes y después de xa_forget.

La interfaz para esta función es:

```
XA_FORGET_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida

Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida

Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada

Descriptor de conexión.

pXID (MQPTR)-entrada/salida

ID de rama de transacción.

Rmid (MQLONG)-entrada/salida

Identificador del gestor de recursos.

Distintivos (MQLONG)-entrada/salida

Opciones del gestor de recursos.

XARetCode (MQLONG)-entrada/salida

Respuesta a una llamada de « XA ».

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
MQAXP  ExitParms;    /* Exit parameter structure */
MQAXC  ExitContext;  /* Exit context structure */
MQHCONN Hconn;       /* Connection handle */
MQPTR  pXID;         /* Transaction branch ID */
MQLONG Rmid;         /* Resource manager identifier */
MQLONG Flags;        /* Resource manager options*/
MQLONG XARetCode;    /* Response from XA call */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
XA_FORGET_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode);
```

La salida debe coincidir con el siguiente prototipo de función C:

```
typedef void MQENTRY XA_FORGET_EXIT (
    PMQAXP  pExitParms,    /* Address of exit parameter structure */
    PMQAXC  pExitContext,  /* Address of exit context structure */
    PMQHCONN pHconn,       /* Address of connection handle */
    PMQPTR  ppXID,         /* Address of transaction branch ID */
    PMQLONG pRmid,         /* Address of resource manager identifier */
    PMQLONG pFlags,        /* Address of resource manager options*/
    PMQLONG pXARetCode);   /* Address of response from XA call */
```

xa_open-XA_OPEN_EXIT

XA_OPEN_EXIT proporciona una función de salida xa_open para realizar antes y después del proceso xa_open. Utilice el identificador de función MQXF_XAOPEN con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar las funciones de salida de llamada antes y después de xa_open.

La interfaz para esta función es:

```
XA_OPEN_EXIT (&ExitParms, &ExitContext, &Hconn, &pXa_info, &Rmid, &Flags, &XARetCode)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida

Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida

Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada

Descriptor de conexión.

pXa_info (PMQCHAR) - entrada/salida

Información del gestor de recursos específico de la instancia.

Rmid (MQLONG)-entrada/salida

Identificador del gestor de recursos.

Distintivos (MQLONG)-entrada/salida

Opciones del gestor de recursos.

XARetCode (MQLONG)-entrada/salida

Respuesta a una llamada de « XA ».

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
MQAXP  ExitParms;    /* Exit parameter structure */
MQAXC  ExitContext;  /* Exit context structure */
MQHCONN Hconn;       /* Connection handle */
PMQCHAR pXa_info;    /* Instance-specific RM info */
MQLONG Rmid;         /* Resource manager identifier */
MQLONG Flags;        /* Resource manager options*/
MQLONG XARetCode;    /* Response from XA call */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
XA_OPEN_EXIT (&ExitParms, &ExitContext, &Hconn, &pXa_info, &Rmid, &Flags, &XARetCode);
```

La salida debe coincidir con el siguiente prototipo de función C:

```
typedef void MQENTRY XA_OPEN_EXIT (
    PMQAXP  pExitParms,    /* Address of exit parameter structure */
    PMQAXC  pExitContext,  /* Address of exit context structure */
    PMQHCONN pHconn,       /* Address of connection handle */
    PPMQCHAR ppXa_info,    /* Address of instance-specific RM info */
    PMQLONG pRmid,         /* Address of resource manager identifier */
    PMQLONG pFlags,        /* Address of resource manager options*/
    PMQLONG pXARetCode);   /* Address of response from XA call */
```

xa_prepare-XA_PREPARE_EXIT

XA_PREPARE_EXIT proporciona una función de salida xa_prepare para realizar antes y después del proceso xa_prepare. Utilice el identificador de función MQXF_XAPREPARE con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar las funciones de salida de llamada antes y después de xa_prepare.

La interfaz para esta función es:

```
XA_PREPARE_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida

Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida

Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada

Descriptor de conexión.

pXID (MQPTR)-entrada/salida

ID de rama de transacción.

Rmid (MQLONG)-entrada/salida

Identificador del gestor de recursos.

Distintivos (MQLONG)-entrada/salida

Opciones del gestor de recursos.

XARetCode (MQLONG)-entrada/salida

Respuesta a una llamada de « XA ».

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
MQAXP  ExitParms; /* Exit parameter structure */
MQAXC  ExitContext; /* Exit context structure */
MQHCONN Hconn; /* Connection handle */
MQPTR  pXID; /* Transaction branch ID */
MQLONG Rmid; /* Resource manager identifier */
MQLONG Flags; /* Resource manager options*/
MQLONG XARetCode; /* Response from XA call */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
XA_PREPARE_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode);
```

La salida debe coincidir con el siguiente prototipo de función C:

```
typedef void MQENTRY XA_PREPARE_EXIT (
    PMQAXP pExitParms, /* Address of exit parameter structure */
    PMQAXC pExitContext, /* Address of exit context structure */
    PMQHCONN pHconn, /* Address of connection handle */
    PMQPTR ppXID, /* Address of transaction branch ID */
    PMQLONG pRmid, /* Address of resource manager identifier */
    PMQLONG pFlags, /* Address of resource manager options*/
    PMQLONG pXARetCode); /* Address of response from XA call */
```

xa_recover-XA_RECOVER_EXIT

XA_RECOVER_EXIT proporciona una función de salida `xa_recover` para realizar antes y después del proceso `xa_recover`. Utilice el identificador de función `MQXF_XARECOVER` con las razones de salida `MQXR_BEFORE` y `MQXR_AFTER` para registrar las funciones de salida de llamada antes y después de `xa_recover`.

La interfaz para esta función es:

```
XA_RECOVER_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Count, &Rmid, &Flags, &XARetCode)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida

Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida

Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada

Descriptor de conexión.

pXID (MQPTR)-entrada/salida

ID de rama de transacción.

Recuento (MQLONG)-entrada/salida

Máximo de XID en matriz XID

Rmid (MQLONG)-entrada/salida

Identificador del gestor de recursos.

Distintivos (MQLONG)-entrada/salida

Opciones del gestor de recursos.

XARetCode (MQLONG)-entrada/salida

Respuesta a una llamada de « XA ».

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
MQAXP  ExitParms; /* Exit parameter structure */
MQAXC  ExitContext; /* Exit context structure */
MQHCONN Hconn; /* Connection handle */
```

```

MQPTR   pXID;           /* Transaction branch ID */
MQLONG  Count;          /* Max XIDs in XID array */
MQLONG  Rmid;           /* Resource manager identifier */
MQLONG  Flags;          /* Resource manager options */
MQLONG  XARetCode;      /* Response from XA call */

```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
XA_RECOVER_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Count, &Rmid, &Flags, &XARetCode);
```

La salida debe coincidir con el siguiente prototipo de función C:

```

typedef void MQENTRY XA_RECOVER_EXIT (
    PMQAXP  pExitParms, /* Address of exit parameter structure */
    PMQAXC  pExitContext, /* Address of exit context structure */
    PMQHCONN pHconn, /* Address of connection handle */
    MQPTR   pXID, /* Address of transaction branch ID */
    PMQLONG pCount, /* Address of max XIDs in XID array */
    PMQLONG pRmid, /* Address of resource manager identifier */
    PMQLONG pFlags, /* Address of resource manager options */
    PMQLONG pXARetCode); /* Address of response from XA call */

```

xa_rollback-XA_ROLLBACK_EXIT

XA_ROLLBACK_EXIT proporciona una función de salida *xa_rollback* para realizar antes y después del proceso *xa_rollback*. Utilice el identificador de función *MQXF_XAROLLBACK* con las razones de salida *MQXR_BEFORE* y *MQXR_AFTER* para registrar las funciones de salida de llamada antes y después de *xa_rollback*.

La interfaz para esta función es:

```
XA_ROLLBACK_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida

Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida

Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada

Descriptor de conexión.

pXID (MQPTR)-entrada/salida

ID de rama de transacción.

Rmid (MQLONG)-entrada/salida

Identificador del gestor de recursos.

Distintivos (MQLONG)-entrada/salida

Opciones del gestor de recursos.

XARetCode (MQLONG)-entrada/salida

Respuesta a una llamada de « XA ».

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```

MQAXP  ExitParms; /* Exit parameter structure */
MQAXC  ExitContext; /* Exit context structure */
MQHCONN Hconn; /* Connection handle */
MQPTR   pXID; /* Transaction branch ID */
MQLONG  Rmid; /* Resource manager identifier */
MQLONG  Flags; /* Resource manager options */
MQLONG  XARetCode; /* Response from XA call */

```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
XA_ROLLBACK_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode);
```

La salida debe coincidir con el siguiente prototipo de función C:

```
typedef void MQENTRY XA_ROLLBACK_EXIT (
    PMQAXP    pExitParms,    /* Address of exit parameter structure */
    PMQAXC    pExitContext,  /* Address of exit context structure */
    PMQHCONN  pHconn,        /* Address of connection handle */
    PMQPTR    ppXID,         /* Address of transaction branch ID */
    PMQLONG   pRmid,         /* Address of resource manager identifier */
    PMQLONG   pFlags,        /* Address of resource manager options */
    PMQLONG   pXARetCode);   /* Address of response from XA call */
```

xa_start-XA_START_EXIT

XA_START_EXIT proporciona una función de salida xa_start para realizar antes y después del proceso xa_start. Utilice el identificador de función MQXF_XASTART con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar las funciones de salida de llamada antes y después de xa_start.

La interfaz para esta función es:

```
XA_START_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida

Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida

Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada

Descriptor de conexión.

pXID (MQPTR)-entrada/salida

ID de rama de transacción.

Rmid (MQLONG)-entrada/salida

Identificador del gestor de recursos.

Distintivos (MQLONG)-entrada/salida

Opciones del gestor de recursos.

XARetCode (MQLONG)-entrada/salida

Respuesta a una llamada de « XA ».

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
MQAXP    ExitParms;    /* Exit parameter structure */
MQAXC    ExitContext;  /* Exit context structure */
MQHCONN  Hconn;        /* Connection handle */
MQPTR    pXID;         /* Transaction branch ID */
MQLONG   Rmid;         /* Resource manager identifier */
MQLONG   Flags;        /* Resource manager options */
MQLONG   XARetCode;    /* Response from XA call */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
XA_START_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode);
```

La salida debe coincidir con el siguiente prototipo de función C:

```
typedef void MQENTRY XA_START_EXIT (
    PMQAXP    pExitParms,    /* Address of exit parameter structure */
    PMQAXC    pExitContext,  /* Address of exit context structure */
    PMQHCONN  pHconn,        /* Address of connection handle */
    PMQPTR    ppXID,         /* Address of transaction branch ID */
    PMQLONG   pRmid,         /* Address of resource manager identifier */
    PMQLONG   pFlags,        /* Address of resource manager options */
    PMQLONG   pXARetCode);   /* Address of response from XA call */
```

ax_reg-AX_REG_EXIT

AX_REG_EXIT proporciona una función de salida ax_reg para realizar antes y después del proceso ax_reg. Utilice el identificador de función MQXF_AXREG con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar las funciones de salida de llamada antes y después de ax_reg.

La interfaz para esta función es:

```
AX_REG_EXIT (&ExitParms, &ExitContext, &pXID, &Rmid, &Flags, &XARetCode)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida

Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida

Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada

Descriptor de conexión.

pXID (MQPTR)-entrada/salida

ID de rama de transacción.

Rmid (MQLONG)-entrada/salida

Identificador del gestor de recursos.

Distintivos (MQLONG)-entrada/salida

Opciones del gestor de recursos.

XARetCode (MQLONG)-entrada/salida

Respuesta a una llamada de « XA ».

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
MQAXP  ExitParms;    /* Exit parameter structure */
MQAXC  ExitContext;  /* Exit context structure */
MQPTR  pXID;         /* Transaction branch ID */
MQLONG Rmid;         /* Resource manager identifier */
MQLONG Flags;        /* Resource manager options*/
MQLONG XARetCode;    /* Response from XA call */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
AX_REG_EXIT (&ExitParms, &ExitContext, &pXID, &Rmid, &Flags, &XARetCode);
```

La salida debe coincidir con el siguiente prototipo de función C:

```
typedef void MQENTRY AX_REG_EXIT (
    PMQAXP pExitParms, /* Address of exit parameter structure */
    PMQAXC pExitContext, /* Address of exit context structure */
    PMQPTR ppXID, /* Address of transaction branch ID */
    PMQLONG pRmid, /* Address of resource manager identifier */
    PMQLONG pFlags, /* Address of resource manager options*/
    PMQLONG pXARetCode); /* Address of response from XA call */
```

ax_unreg-AX_UNREG_EXIT

AX_UNREG_EXIT proporciona una función de salida ax_unreg para realizar antes y después del proceso ax_unreg. Utilice el identificador de función MQXF_AXUNREG con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar las funciones de salida de llamada antes y después de ax_unreg.

La interfaz para esta función es:

```
AX_UNREG_EXIT (&ExitParms, &ExitContext, &Rmid, &Flags, &XARetCode);
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida

Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida

Salir de la estructura de contexto.

Rmid (MQLONG)-entrada/salida

Identificador del gestor de recursos.

Distintivos (MQLONG)-entrada/salida

Opciones del gestor de recursos.

XARetCode (MQLONG)-entrada/salida

Respuesta a una llamada de « XA ».

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
MQAXP ExitParms; /* Exit parameter structure */
MQAXC ExitContext; /* Exit context structure */
MQLONG Rmid; /* Resource manager identifier */
MQLONG Flags; /* Resource manager options*/
MQLONG XARetCode; /* Response from XA call */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
AX_UNREG_EXIT (&ExitParms, &ExitContext, &Rmid, &Flags, &XARetCode);
```

La salida debe coincidir con el siguiente prototipo de función C:

```
typedef void MQENTRY AX_UNREG_EXIT (
    PMQAXP pExitParms, /* Address of exit parameter structure */
    PMQAXC pExitContext, /* Address of exit context structure */
    PMQLONG pRmid, /* Address of resource manager identifier */
    PMQLONG pFlags, /* Address of resource manager options*/
    PMQLONG pXARetCode); /* Address of response from XA call */
```

Información general sobre la invocación de funciones de salida

Este tema proporciona algunas directrices generales para ayudarle a planificar las salidas, especialmente relacionadas con el manejo de errores y sucesos inesperados.

Error de salida

Si una función de salida termina de forma anómala después de una llamada MQGET destructiva, fuera de punto de sincronismo, pero antes de que el mensaje se haya pasado a la aplicación, el manejador de salida puede recuperarse de la anomalía y pasar el control a la aplicación.

En este caso, es posible que se pierda el mensaje. Esto es como lo que sucede cuando una aplicación falla inmediatamente después de recibir un mensaje de una cola.

La llamada MQGET puede completarse con MQCC_FAILED y MQRC_API_EXIT_ERROR.

Si una función de salida de llamada de API *before* termina de forma anómala, el manejador de salida puede recuperarse de la anomalía y pasar el control a la aplicación sin procesar la llamada de API. En este caso, la función de salida debe recuperar los recursos que posee.

Si las salidas encadenadas están en uso, las salidas de llamada de API *after* para cualquier salida de llamada de API *before* que se haya controlado correctamente pueden ser controladas por sí mismas. La llamada de API puede fallar con MQCC_FAILED y MQRC_API_EXIT_ERROR.

Manejo de errores de ejemplo para funciones de salida

El diagrama siguiente muestra los puntos (e *N*) en el que se pueden producir errores. Es sólo un ejemplo para mostrar cómo se comportan las salidas y debe leerse junto con la tabla siguiente. En este

ejemplo, se invocan dos funciones de salida antes y después de cada llamada de API para mostrar el comportamiento con salidas encadenadas.

Application	ErrPt	Exit function	API call
-----	-----	-----	-----
Start			
MQCONN	-->		
	e1	MQ_INIT_EXIT	
	e2	before MQ_CONNX_EXIT 1	
	e3	before MQ_CONNX_EXIT 2	
	e4		--> MQCONN
	e5	after MQ_CONNX_EXIT 2	
	e6	after MQ_CONNX_EXIT 1	
	e7		
MQOPEN	<-- -->		
	e8	before MQ_OPEN_EXIT 1	
	e9	before MQ_OPEN_EXIT 2	
	e10		--> MQOPEN
	e11	after MQ_OPEN_EXIT 2	
	e12	after MQ_OPEN_EXIT 1	
	e12		
MQPUT	<-- -->		
	e13	before MQ_PUT_EXIT 1	
	e14	before MQ_PUT_EXIT 2	
	e15		--> MQPUT
	e16	after MQ_PUT_EXIT 2	
	e17	after MQ_PUT_EXIT 1	
	e17		
MQCLOSE	<-- -->		
	e18	before MQ_CLOSE_EXIT 1	
	e19	before MQ_CLOSE_EXIT 2	
	e20		--> MQCLOSE
	e21	after MQ_CLOSE_EXIT 2	
	e22	after MQ_CLOSE_EXIT 1	
	e22		
MQDISC	<-- -->		
	e23	before MQ_DISC_EXIT 1	
	e24	before MQ_DISC_EXIT 2	
	e25		--> MQDISC
	e26	after MQ_DISC_EXIT 2	
	e27	after MQ_DISC_EXIT 1	
	e27		
	<--		
end			

La tabla siguiente lista las acciones que se deben realizar en cada punto de error. Sólo se ha cubierto un subconjunto de los puntos de error, ya que las reglas que se muestran aquí se pueden aplicar a todos los demás. Son las acciones que especifican el comportamiento previsto en cada caso.

Tabla 873. Errores de salida de API y acciones adecuadas a realizar

Err Pt	Descripción	Acciones
e1	Error al configurar el entorno.	1. Deshacer configuración de entorno según sea necesario 2. No hay funciones de salida de unidad 3. Error de MQCONN con MQCC_FAILED, MQRC_API_EXIT_LOAD_ERROR
e2	La función MQ_INIT_EXIT se completa con: • MQXCC_FALLO • MQXCC_*	• Para MQXCC_FAILED: 1. Limpiar entorno 2. Error de MQCONN con MQCC_FAILED, MQRC_API_EXIT_INIT_ERROR • Para MQXCC_* 1. Actúe como para los valores de MQXCC_* y MQXR2_*¹ 2. Limpiar entorno
e3	Antes , la función MQ_CONNX_EXIT 1 finaliza con: • MQXCC_FALLO • MQXCC_*	• Para MQXCC_FAILED: 1. Unidad MQ_TERM_EXIT, función 2. Limpiar entorno 3. Error de llamada MQCONN con MQCC_FAILED, MQRC_API_EXIT_ERROR • Para MQXCC_* 1. Actúe como para los valores de MQXCC_* y MQXR2_*¹ 2. Función MQ_TERM_EXIT de unidad, si es necesario 3. Limpiar entorno si es necesario
e4	Antes , la función MQ_CONNX_EXIT 2 se completa con: • MQXCC_FALLO • MQXCC_*	• Para MQXCC_FAILED: 1. Unidad <i>después de la función</i> MQ_CONNX_EXIT 1 2. Unidad MQ_TERM_EXIT, función 3. Limpiar entorno 4. Error de llamada MQCONN con MQCC_FAILED, MQRC_API_EXIT_ERROR • Para MQXCC_* 1. Actúe como para los valores de MQXCC_* y MQXR2_*¹ 2. Unidad <i>después de la función</i> MQ_CONNX_EXIT 1 si la salida no se suprime 3. Función MQ_TERM_EXIT de unidad, si es necesario 4. Limpiar entorno si es necesario

Tabla 873. Errores de salida de API y acciones adecuadas a realizar (continuación)		
Err Pt	Descripción	Acciones
e5	La llamada MQCONN falla.	1. Pasar MQCONN CompCode y Razón 2. Unidad <i>después</i> de la función MQ_CONNX_EXIT 2 si <i>antes</i> MQ_CONNX_EXIT 2 se ha ejecutado correctamente y la salida no se suprime 3. Unidad <i>después</i> de la función MQ_CONNX_EXIT 1 si <i>antes</i> MQ_CONNX_EXIT 1 se ha ejecutado correctamente y la salida no se suprime 4. Unidad MQ_TERM_EXIT, función 5. Limpiar entorno
e6	<i>Después de la función MQ_CONNX_EXIT 2 se completa con:</i> • MQXCC_FALLO • MQXCC_*	• Para MQXCC_FAILED: 1. Unidad <i>después de la función</i> MQ_CONNX_EXIT 1 2. Complete la llamada MQCONN con MQCC_FAILED, MQRC_API_EXIT_ERROR • Para MQXCC_* 1. Actúe como para los valores de MQXCC_* y MQXR2_*¹ 2. Unidad <i>después de la función</i> MQ_CONNX_EXIT 1 si es necesario
e7	<i>Después de que la función MQ_CONNX_EXIT 1 se complete con:</i> • MQXCC_FALLO • MQXCC_*	• Para MQXCC_FAILED, complete la llamada MQCONN con MQCC_FAILED, MQRC_API_EXIT_ERROR • Para MQXCC_*, actúe como para los valores de MQXCC_* y MQXR2_*¹
e8	<i>Antes la función MQ_OPEN_EXIT 1 se completa con:</i> • MQXCC_FALLO • MQXCC_*	• Para MQXCC_FAILED, complete la llamada MQOPEN con MQCC_FAILED, MQRC_API_EXIT_ERROR • Para MQXCC_*, actúe como para los valores de MQXCC_* y MQXR2_*¹
e9	<i>Antes , la función MQ_OPEN_EXIT 2 finaliza con:</i> • MQXCC_FALLO • MQXCC_*	• Para MQXCC_FAILED: 1. Unidad <i>después de la función</i> MQ_OPEN_EXIT 1 2. Complete la llamada MQOPEN con MQCC_FAILED, MQRC_API_EXIT_ERROR • Para MQXCC_*, actúe como para los valores de MQXCC_* y MQXR2_*¹
e10	La llamada MQOPEN falla	1. Pasar MQOPEN CompCode y Razón 2. Unidad <i>después de la función</i> MQ_OPEN_EXIT 2 si la salida no se suprime 3. Unidad <i>después de la función</i> MQ_OPEN_EXIT 1 si la salida no se ha suprimido y si las salidas encadenadas no se han suprimido

Tabla 873. Errores de salida de API y acciones adecuadas a realizar (continuación)		
Err Pt	Descripción	Acciones
e1 1	Después de que la función MQ_OPEN_EXIT 2 se complete con: • MQXCC_FALLO • MQXCC_*	• Para MQXCC_FAILED: 1. Unidad <i>después de la función</i> MQ_OPEN_EXIT 1 2. Complete la llamada MQOPEN con MQCC_FAILED, MQRC_API_EXIT_ERROR • Para MQXCC_* 1. Actúe como para los valores de MQXCC_* y MQXR2_*¹ 2. Unidad <i>después de la función</i> MQ_OPEN_EXIT 1 si no se suprime la salida
e2 5	Después de la función MQ_DISC_EXIT 2 se completa con: • MQXCC_FALLO • MQXCC_*	• Para MQXCC_FAILED: 1. Unidad <i>después de la función</i> MQ_DISC_EXIT 1 2. Unidad MQ_TERM_EXIT, función 3. Entorno de ejecución de salida de limpieza 4. Completar llamada MQDISC con MQCC_FAILED, MQRC_API_EXIT_ERROR • Para MQXCC_* 1. Actúe como para los valores de MQXCC_* y MQXR2_*¹ 2. Unidad MQ_TERM_EXIT, función 3. Entorno de ejecución de salida de limpieza

Nota:

1. Los valores de MQXCC_* y MQXR2_* y sus acciones correspondientes se definen en [Cómo procesan los gestores de colas las funciones de salida](#).

Los campos ExitResponse se han establecido incorrectamente

Este tema proporciona información sobre lo que sucedería cuando el campo ExitResponse se establece en cualquier valor que no sean los valores soportados.

Si el campo ExitResponse se establece en un valor distinto de uno de los valores soportados, se aplican las acciones siguientes:

- Para una función de salida de API MQCONN o MQDISC *anterior a* :
 - El valor ExitResponse2 se ignora.
 - No se invocan más *antes* de las funciones de salida en la cadena de salida (si las hay); no se emite la propia llamada de API.
 - Para las salidas *anteriores* que se han llamado correctamente, las salidas *posteriores* se llaman en orden inverso.
 - Si está registrado, las funciones de salida de terminación para las funciones de salida *antes* MQCONN o MQDISC de la cadena que se han invocado correctamente se controlan para que se limpien después de estas funciones de salida.
 - La llamada MQCONN o MQDISC falla con MQRC_API_EXIT_ERROR.
- Para una función de salida de API *anterior a* IBM MQ que no sea MQCONN o MQDISC:
 - El valor ExitResponse2 se ignora.
 - No se invocan más *antes* o *después* de las funciones de conversión de datos en la cadena de salida (si las hay).

- Para las salidas *anteriores* que se han llamado correctamente, las salidas *posteriores* se llaman en orden inverso.
- La propia llamada de API IBM MQ no se emite.
- La llamada de API IBM MQ falla con MQRC_API_EXIT_ERROR.
- Para una función de salida de API MQCONN o MQDISC *después de* :
 - El valor ExitResponse2 se ignora.
 - Las funciones de salida restantes que se han llamado correctamente antes de la llamada de API se llaman en orden inverso.
 - Si está registrado, las funciones de salida de terminación para las funciones de salida *antes o después* MQCONN o MQDISC de la cadena que se han invocado correctamente se controlan para que se limpien después de la salida.
 - Se devuelve a la aplicación un CompCode del más grave de MQCC_WARNING y el CompCode devuelto por la salida.
 - Se devuelve una razón de MQRC_API_EXIT_ERROR a la aplicación.
 - La llamada de API IBM MQ se ha emitido correctamente.
- Para una función de salida de llamada de API *after* IBM MQ que no sea MQCONN o MQDISC:
 - El valor ExitResponse2 se ignora.
 - Las funciones de salida restantes que se han llamado correctamente antes de la llamada de API se llaman en orden inverso.
 - Se devuelve a la aplicación un CompCode del más grave de MQCC_WARNING y el CompCode devuelto por la salida.
 - Se devuelve una razón de MQRC_API_EXIT_ERROR a la aplicación.
 - La llamada de API IBM MQ se ha emitido correctamente.
- Para la conversión de datos *before* al obtener la función de salida:
 - El valor ExitResponse2 se ignora.
 - Las funciones de salida restantes que se han llamado correctamente antes de la llamada de API se llaman en orden inverso.
 - El mensaje no se convierte y el mensaje no convertido se devuelve a la aplicación.
 - Se devuelve a la aplicación un CompCode del más grave de MQCC_WARNING y el CompCode devuelto por la salida.
 - Se devuelve una razón de MQRC_API_EXIT_ERROR a la aplicación.
 - La llamada de API IBM MQ se ha emitido correctamente.

Nota: Dado que el error es con la salida, es mejor devolver MQRC_API_EXIT_ERROR que devolver MQRC_NOT_CONVERT.

Si una función de salida establece el campo ExitResponse2 en un valor distinto de uno de los valores soportados, en su lugar se asume un valor de MQXR2_DEFAULT_CONTINUATION .

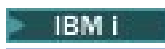
Información de consulta sobre la interfaz de servicios instalables

Esta colección de temas proporciona información de referencia para los servicios instalables.

Las funciones y los tipos de datos se enumeran en orden alfabético dentro del grupo para cada tipo de servicio.

Conceptos relacionados

 [Servicios y componentes instalables para UNIX, Linux y Windows](#)

 [Servicios y componentes instalables para IBM i](#)

 [Información de referencia sobre la interfaz de servicios instalables para IBM i](#)

Tareas relacionadas

[Extensión de recursos del gestor de colas](#)

 [Configuración de servicios instalables](#)

Cómo se muestran las funciones

Cómo se documentan las funciones de servicios instalables.

Para cada función hay una descripción, incluido el identificador de función (para MQZEP).

Los *parámetros* se muestran en el orden en el que deben aparecer. Todos ellos deben estar presentes.

Cada nombre de parámetro va seguido de su tipo de datos. Estos son los tipos de datos elementales descritos en [“Tipos de datos elementales”](#) en la página 177.

También se proporciona la invocación del lenguaje C, después de la descripción de los parámetros.

MQZ_AUTHENTICATE_USER-Autenticar usuario

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_5 y la invoca el gestor de colas para autenticar un usuario o para establecer campos de contexto de identidad. Se invoca cuando se establece el contexto de aplicación de usuario IBM MQ .

El contexto de aplicación se establece durante las llamadas de conexión en el punto en el que se inicializa el contexto de usuario de la aplicación y en cada punto en el que se cambia el contexto de usuario de la aplicación. Cada vez que se realiza una llamada de conexión, la información de contexto de usuario de la aplicación se vuelve a adquirir en el campo *IdentityContext* .

El identificador de función para esta función (para MQZEP) es MQZID_AUTHENTICATE_USER.

Sintaxis

MQZ_AUTHENTICATE_USER (*QMgrName* , *SecurityParms* , *ApplicationContext* , *IdentityContext* , *CorrelationPtr* , *ComponentData* , *Continuation* , *CompCode* , *Motivo*)

Parámetros

QMgrName

Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

SecurityParms

Tipo: MQCSP-entrada

Parámetros de seguridad. Datos relacionados con el ID de usuario, la contraseña y el tipo de autenticación. Si el atributo *AuthenticationType* de la estructura MQCSP se especifica como MQCSP_AUTH_USER_ID_AND_PWD, tanto el ID de usuario como la contraseña se comparan con los campos equivalentes del parámetro *IdentityContext* (MQZIC) para determinar si coinciden. Para obtener más información, consulte [“MQCSP - Parámetros de seguridad”](#) en la página 291.

Durante una llamada de tipo « MQI » a MQCONN, este parámetro contiene valores nulos o predeterminados.

ApplicationContext

Tipo: MQZAC-entrada

Contexto de aplicación. Datos relacionados con la aplicación de llamada. Consulte [MQZAC-Contexto de aplicación](#) para obtener más detalles.

Cada vez que se realiza una llamada a `MQI` en MQCONN o MQCONNX, se vuelve a obtener la información del contexto del usuario contenida en la estructura MQZAC.

IdentityContext

Tipo: MQZIC-entrada/salida

Contexto de identidad. En la entrada de la función de autenticar usuario, identifica el contexto de identidad actual. La función de autenticar usuario puede cambiar esto, momento en el que el gestor de colas adopta el nuevo contexto de identidad. Consulte [MQZIC-Contexto de identidad](#) para obtener más detalles sobre la estructura MQZIC.

CorrelationPtr

Tipo: MQPTR-salida

Puntero de correlación. Especifica la dirección de cualquier dato de correlación. Este puntero se pasa posteriormente a otras llamadas de OAM.

ComponentData

Tipo: MQBYTE x ComponentDataLength - entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de las funciones de este componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro ComponentDataLength de la llamada MQZ_INIT_AUTHORITY.

continuación

Tipo: MQLONG - salida

Distintivo de continuación. Puede especificar los valores siguientes:

MQZCI_DEFAULT

Continuación dependiente de otros componentes.

MQZCI_STOP

No continúe con el componente siguiente.

CompCode

Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_SERVICE_ERROR

(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

Para obtener más información sobre estos códigos de razón, consulte [Mensajes y códigos de razón](#).

Invocación en C

```
MQZ_AUTHENTICATE_USER (QMgrName, SecurityParms, ApplicationContext,  
                        IdentityContext, &CorrelationPtr, ComponentData,  
                        &Continuation, &CompCode, &Reason);
```

Declare los parámetros pasados al servicio como se indica a continuación:

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQCSP     SecurityParms;      /* Security parameters */  
MQZAC     ApplicationContext; /* Application context */  
MQZIC     IdentityContext;    /* Identity context */  
MQPTR     CorrelationPtr;     /* Correlation pointer */  
MQBYTE    ComponentData[n];   /* Component data */  
MQLONG    Continuation;       /* Continuation indicator set by  
                               component */  
MQLONG    CompCode;           /* Completion code */  
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

MQZ_CHECK_AUTHORITY-Comprobar autorización

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_1 y la inicia el gestor de colas para comprobar si una entidad tiene autorización para realizar una acción determinada, o acciones, en un objeto especificado.

El identificador de función para esta función (para MQZEP) es MQZID_CHECK_AUTHORITY.

Sintaxis

MQZ_CHECK_AUTHORITY(QMgrName , EntityName , EntityType , ObjectName ,
ObjectType , Authority , ComponentData , Continuation , CompCode , Reason)

Parámetros

QMgrName

Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

EntityName

Tipo: MQCHAR12 -entrada

Nombre de entidad. El nombre de la entidad cuya autorización para el objeto se va a comprobar. La longitud máxima de la serie es de 12 caracteres; si es más corta que esa, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

No es esencial que esta entidad sea conocida por el servicio de seguridad subyacente. Si no se conoce, las autorizaciones del grupo **nobody** especial (al que se supone que pertenecen todas las entidades) se utilizan para la comprobación. Un nombre totalmente en blanco es válido y se puede utilizar de esta forma.

EntityType

Tipo: MQLONG - entrada

Tipo de entidad. El tipo de entidad especificado por EntityName. Tiene que ser uno de los valores siguientes:

MQZAET_PRINCIPAL

Director.

GRUPO_MQZAC

group.

ObjectName

Tipo: MQCHAR48 -entrada

El nombre del objeto. El nombre del objeto al que se necesita acceso. La longitud máxima de la serie es de 48 caracteres; si es más corta, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

Si *ObjectType* es MQOT_Q_MGR, este nombre es el mismo que *QMgrName*.

ObjectType

Tipo: MQLONG - entrada

Tipo de objeto. El tipo de entidad especificado por *ObjectName*. Tiene que ser uno de los valores siguientes:

MQOT_AUTH_INFO

Información de autenticación.

MQOT_CHANNEL

Canal.

MQOT_CLNTCONN_CHANNEL

Canal de conexión de cliente.

MQOT_ESCUCHA

Escucha.

MQOT_LISTA_DE_NOMBRES

Lista de nombres.

MQOT_PROCESO

.

MQOT_Q

Cola.

MQOT_Q_MGR

Gestor de colas.

SERVICIO MQOT_SERVICE

.

Autorización

Tipo: MQLONG - entrada

Autorización que se debe comprobar. Si se está comprobando una autorización, este campo es igual a la operación de autorización adecuada (constante MQZAO_*). Si se está comprobando más de una autorización, es el OR a nivel de bit de las constantes MQZAO_* correspondientes.

Las siguientes autorizaciones se aplican al uso de las llamadas de MQI :

MQZAO_CONNECT

Posibilidad de utilizar la llamada MQCONN.

MQZAO_BROWSE

Posibilidad de utilizar la llamada MQGET con una opción de examinar.

Esto permite especificar la opción MQGMO_BROWSE_FIRST, MQGMO_BROWSE_MSG_UNDER_CURSOR o MQGMO_BROWSE_NEXT en la llamada MQGET.

MQZAO_INPUT

Director. Posibilidad de utilizar la llamada MQGET con una opción de entrada.

Esto permite especificar la opción MQOO_INPUT_SHARED, MQOO_INPUT_EXCLUSIVE o MQOO_INPUT_AS_Q_DEF en la llamada MQOPEN.

MQZAO_OUTPUT

Posibilidad de utilizar la llamada MQPUT.

Esto permite especificar la opción MQOO_OUTPUT en la llamada MQOPEN.

MQZAO_INQUIRE

Posibilidad de utilizar la llamada MQINQ.

Esto permite especificar la opción MQOO_INQUIRE en la llamada MQOPEN.

MQZAO_SET

Posibilidad de utilizar la llamada MQSET.

Esto permite especificar la opción MQOO_SET en la llamada MQOPEN.

MQZAO_PASS_IDENTITY_CONTEXT

Capacidad de pasar contexto de identidad.

Esto permite especificar la opción MQOO_PASS_IDENTITY_CONTEXT en la llamada MQOPEN y especificar la opción MQPMO_PASS_IDENTITY_CONTEXT en las llamadas MQPUT y MQPUT1 .

MQZAO_PASS_ALL_CONTEXT

Capacidad de pasar todo el contexto.

Esto permite especificar la opción MQOO_PASS_ALL_CONTEXT en la llamada MQOPEN y especificar la opción MQPMO_PASS_ALL_CONTEXT en las llamadas MQPUT y MQPUT1 .

MQZAO_SET_IDENTITY_CONTEXT

Capacidad para establecer el contexto de identidad.

Esto permite especificar la opción MQOO_SET_IDENTITY_CONTEXT en la llamada MQOPEN y especificar la opción MQPMO_SET_IDENTITY_CONTEXT en las llamadas MQPUT y MQPUT1 .

MQZAO_SET_ALL_CONTEXT

Capacidad para establecer todo el contexto.

Esto permite especificar la opción MQOO_SET_ALL_CONTEXT en la llamada MQOPEN y especificar la opción MQPMO_SET_ALL_CONTEXT en las llamadas MQPUT y MQPUT1 .

MQZAO_ALTERNATE_USER_AUTHORITY

Capacidad para utilizar la autorización de usuario alternativo.

Esto permite especificar la opción MQOO_ALTERNATE_USER_AUTHORITY en la llamada MQOPEN y especificar la opción MQPMO_ALTERNATE_USER_AUTHORITY en la llamada MQPUT1 .

MQZAO_ALL_MQI

Todas las autorizaciones de MQI.

Esto habilita todas las autorizaciones.

Las autorizaciones siguientes se aplican a la administración de un gestor de colas:

MQZAO_CREAR

Posibilidad de crear objetos de un tipo especificado.

MQZAO_DELETE

Posibilidad de suprimir un objeto especificado.

MQZAO_DISPLAY

Capacidad para visualizar los atributos de un objeto especificado.

MQZAO_CHANGE

Posibilidad de cambiar los atributos de un objeto especificado.

MQZAO_CLEAR

Posibilidad de suprimir todos los mensajes de una cola especificada.

MQZAO_AUTORIZAR

Posibilidad de autorizar a otros usuarios para un objeto especificado.

MQZAO_CONTROL

Capacidad para iniciar o detener un objeto de canal de escucha, servicio o no cliente, y la capacidad de hacer ping a un objeto de canal no cliente.

MQZAO_CONTROL_EXTENDED

Capacidad para restablecer un número de secuencia o resolver un mensaje dudoso en un objeto de canal no de cliente.

MQZAO_ALL_ADMIN

Capacidad para establecer el contexto de identidad.

Todas las autorizaciones de administración, excepto MQZAO_CREATE.

Las siguientes autorizaciones se aplican tanto al uso del servidor de bases de datos de Oracle (MQI) como a la administración de un gestor de colas:

MQZAO_TODOS

Todas las autorizaciones, excepto MQZAO_CREATE.

MQZAO_NINGUNA

Sin autorizaciones.

ComponentData

Tipo: MQBYTE x ComponentDataLength - entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro **ComponentDataLength** de la llamada MQZ_INIT_AUTHORITY.

continuación

Tipo: MQLONG - salida

Indicador de continuación establecido por componente. Se pueden especificar los siguientes valores:

MQZCI_DEFAULT

Continuación dependiente del gestor de colas.

Para MQZ_CHECK_AUTHORITY, tiene el mismo efecto que MQZCI_STOP.

MQZCI_CONTINUAR

Continúe con el componente siguiente.

MQZCI_STOP

No continúe con el componente siguiente.

Si la llamada a un componente falla (es decir, *CompCode* devuelve MQCC_FAILED), y el parámetro *Continuación* es MQZCI_DEFAULT o MQZCI_CONTINUE, el gestor de colas continúa llamando a otros componentes si los hay.

Si la llamada es satisfactoria (es decir, *CompCode* devuelve MQCC_OK) no se llama a ningún otro componente independientemente del valor de *Continuación*.

Si la llamada falla y el parámetro *Continuación* es MQZCI_STOP, no se llama a ningún otro componente y se devuelve el error al gestor de colas. Los componentes no conocen las llamadas anteriores, por lo que el parámetro *Continuación* siempre se establece en MQZCI_DEFAULT antes de la llamada.

CompCode

Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_NOT_AUTHORIZED

(2035, X'7F3') Acceso no autorizado.

MQRC_SERVICE_ERROR

(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICIO_NO_DISPONIBLE

(2285, X'8ED') Servicio subyacente no disponible.

Para obtener más información sobre estos códigos de razón, consulte [Códigos de terminación y de razón de la API](#).

Invocación en C

```
MQZ_CHECK_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName,  
                    ObjectType, Authority, ComponentData,  
                    &Continuation, &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQCHAR12  EntityName;         /* Entity name */  
MQLONG    EntityType;         /* Entity type */  
MQCHAR48  ObjectName;        /* Object name */  
MQLONG    ObjectType;         /* Object type */  
MQLONG    Authority;          /* Authority to be checked */  
MQBYTE    ComponentData[n];   /* Component data */  
MQLONG    Continuation;       /* Continuation indicator set by  
                               component */  
MQLONG    CompCode;           /* Completion code */  
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

MQZ_CHECK_AUTHORITY_2 -Comprobar autorización (ampliada)

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_2 y la inicia el gestor de colas para comprobar si una entidad tiene autorización para realizar una acción determinada, o acciones, en un objeto especificado.

El identificador de función para esta función (para MQZEP) es MQZID_CHECK_AUTHORITY.

MQZ_CHECK_AUTHORITY_2 es como MQZ_CHECK_AUTHORITY, pero con el parámetro **EntityName** sustituido por el parámetro **EntityData**.

Sintaxis

MQZ_CHECK_AUTHORITY_2(QMgrName , EntityData , EntityType , ObjectName ,
ObjectType , Authority , ComponentData , Continuation , CompCode , Reason)

Parámetros

QMgrName

Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

EntityData

Tipo: MQZED-entrada

Datos de entidad. Datos relacionados con la entidad con autorización sobre el objeto que se va a comprobar. Consulte [“MQZED-Descriptor de entidad”](#) en la página 1705 para obtener los detalles.

No es esencial que esta entidad sea conocida por el servicio de seguridad subyacente. Si no se conoce, las autorizaciones del grupo **nobody** especial (al que se supone que pertenecen todas las entidades) se utilizan para la comprobación. Un nombre totalmente en blanco es válido y se puede utilizar de esta forma.

EntityType

Tipo: MQLONG - entrada

Tipo de entidad. El tipo de entidad especificado por *EntityData*. Tiene que ser uno de los valores siguientes:

MQZAET_PRINCIPAL

Director.

GRUPO_MQZAC

group.

ObjectName

Tipo: MQCHAR48 -entrada

El nombre del objeto. El nombre del objeto al que se necesita acceso. La longitud máxima de la serie es de 48 caracteres; si es más corta, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

Si *ObjectType* es MQOT_Q_MGR, este nombre es el mismo que *QMgrName*.

ObjectType

Tipo: MQLONG - entrada

Tipo de objeto. El tipo de entidad especificado por *ObjectName*. Tiene que ser uno de los valores siguientes:

MQOT_AUTH_INFO

Información de autenticación.

MQOT_CHANNEL

Canal.

MQOT_CLNTCONN_CHANNEL

Canal de conexión de cliente.

MQOT_ESCUCHA

Escucha.

MQOT_LISTA_DE_NOMBRES

Lista de nombres.

MQOT_PROCESO

.

MQOT_Q

Cola.

MQOT_Q_MGR

Gestor de colas.

SERVICIO MQOT_SERVICE

.

MQOT_TEMA

.

Autorización

Tipo: MQLONG - entrada

Autorización que se debe comprobar. Si se está comprobando una autorización, este campo es igual a la operación de autorización adecuada (constante MQZAO_*). Si se está comprobando más de una autorización, es el OR a nivel de bit de las constantes MQZAO_* correspondientes.

Las siguientes autorizaciones se aplican al uso de las llamadas de MQI :

MQZAO_CONNECT

Posibilidad de utilizar la llamada MQCONN.

MQZAO_BROWSE

Posibilidad de utilizar la llamada MQGET con una opción de examinar.

Esto permite especificar la opción MQGMO_BROWSE_FIRST, MQGMO_BROWSE_MSG_UNDER_CURSOR o MQGMO_BROWSE_NEXT en la llamada MQGET.

MQZAO_INPUT

Director. Posibilidad de utilizar la llamada MQGET con una opción de entrada.

Esto permite especificar la opción MQOO_INPUT_SHARED, MQOO_INPUT_EXCLUSIVE o MQOO_INPUT_AS_Q_DEF en la llamada MQOPEN.

MQZAO_OUTPUT

Posibilidad de utilizar la llamada MQPUT.

Esto permite especificar la opción MQOO_OUTPUT en la llamada MQOPEN.

MQZAO_INQUIRE

Posibilidad de utilizar la llamada MQINQ.

Esto permite especificar la opción MQOO_INQUIRE en la llamada MQOPEN.

MQZAO_SET

Posibilidad de utilizar la llamada MQSET.

Esto permite especificar la opción MQOO_SET en la llamada MQOPEN.

MQZAO_PASS_IDENTITY_CONTEXT

Capacidad de pasar contexto de identidad.

Esto permite especificar la opción MQOO_PASS_IDENTITY_CONTEXT en la llamada MQOPEN y especificar la opción MQPMO_PASS_IDENTITY_CONTEXT en las llamadas MQPUT y MQPUT1 .

MQZAO_PASS_ALL_CONTEXT

Capacidad de pasar todo el contexto.

Esto permite especificar la opción MQOO_PASS_ALL_CONTEXT en la llamada MQOPEN y especificar la opción MQPMO_PASS_ALL_CONTEXT en las llamadas MQPUT y MQPUT1 .

MQZAO_SET_IDENTITY_CONTEXT

Capacidad para establecer el contexto de identidad.

Esto permite especificar la opción MQOO_SET_IDENTITY_CONTEXT en la llamada MQOPEN y especificar la opción MQPMO_SET_IDENTITY_CONTEXT en las llamadas MQPUT y MQPUT1 .

MQZAO_SET_ALL_CONTEXT

Capacidad para establecer todo el contexto.

Esto permite especificar la opción MQOO_SET_ALL_CONTEXT en la llamada MQOPEN y especificar la opción MQPMO_SET_ALL_CONTEXT en las llamadas MQPUT y MQPUT1 .

MQZAO_ALTERNATE_USER_AUTHORITY

Capacidad para utilizar la autorización de usuario alternativo.

Esto permite especificar la opción MQOO_ALTERNATE_USER_AUTHORITY en la llamada MQOPEN y especificar la opción MQPMO_ALTERNATE_USER_AUTHORITY en la llamada MQPUT1 .

MQZAO_ALL_MQI

Todas las autorizaciones de MQI.

Esto habilita todas las autorizaciones.

Las autorizaciones siguientes se aplican a la administración de un gestor de colas:

MQZAO_CREAR

Posibilidad de crear objetos de un tipo especificado.

MQZAO_DELETE

Posibilidad de suprimir un objeto especificado.

MQZAO_DISPLAY

Capacidad para visualizar los atributos de un objeto especificado.

MQZAO_CHANGE

Posibilidad de cambiar los atributos de un objeto especificado.

MQZAO_CLEAR

Posibilidad de suprimir todos los mensajes de una cola especificada.

MQZAO_AUTORIZAR

Posibilidad de autorizar a otros usuarios para un objeto especificado.

MQZAO_CONTROL

Capacidad para iniciar o detener un objeto de canal de escucha, servicio o no cliente, y la capacidad de hacer ping a un objeto de canal no cliente.

MQZAO_CONTROL_EXTENDED

Capacidad para restablecer un número de secuencia o resolver un mensaje dudoso en un objeto de canal no de cliente.

MQZAO_ALL_ADMIN

Capacidad para establecer el contexto de identidad.

Todas las autorizaciones de administración, excepto MQZAO_CREATE.

Las siguientes autorizaciones se aplican tanto al uso del servidor de bases de datos de Oracle (MQI) como a la administración de un gestor de colas:

MQZAO_TODOS

Todas las autorizaciones, excepto MQZAO_CREATE.

MQZAO_NINGUNA

Sin autorizaciones.

ComponentData

Tipo: MQBYTE x ComponentDataLength - entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro **ComponentDataLength** de la llamada MQZ_INIT_AUTHORITY.

continuación

Tipo: MQLONG - salida

Indicador de continuación establecido por componente. Se pueden especificar los siguientes valores:

MQZCI_DEFAULT

Continuación dependiente del gestor de colas.

Para MQZ_CHECK_AUTHORITY, tiene el mismo efecto que MQZCI_STOP.

MQZCI_CONTINUAR

Continúe con el componente siguiente.

MQZCI_STOP

No continúe con el componente siguiente.

CompCode

Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_NOT_AUTHORIZED

(2035, X'7F3') Acceso no autorizado.

MQRC_SERVICE_ERROR

(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICIO_NO_DISPONIBLE

(2285, X'8ED') Servicio subyacente no disponible.

Para obtener más información sobre estos códigos de razón, consulte [Códigos de terminación y de razón de la API](#).

Invocación en C

```
MQZ_CHECK_AUTHORITY_2 (QMgrName, &EntityData, EntityType,
                        ObjectName, ObjectType, Authority, ComponentData,
                        &Continuation, &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQCHAR48  QMgrName;           /* Queue manager name */
MQZED     EntityData;         /* Entity data */
MQLONG    EntityType;         /* Entity type */
MQCHAR48  ObjectName;         /* Object name */
MQLONG    ObjectType;         /* Object type */
MQLONG    Authority;          /* Authority to be checked */
MQBYTE    ComponentData[n];   /* Component data */
MQLONG    Continuation;       /* Continuation indicator set by
                               component */
MQLONG    CompCode;           /* Completion code */
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

MQZ_CHECK_PRIVILEGED-Comprobar si el usuario tiene privilegios

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_6 y la invoca el gestor de colas para determinar si un usuario especificado es un usuario privilegiado.

El identificador de función para esta función (para MQZEP) es MQZID_CHECK_PRIVILEGED.

Sintaxis

```
MQZ_CHECK_PRIVILEGED( QMgrName , EntityData , EntityType , ComponentData ,
Continuation , CompCode , Reason )
```

Parámetros

QMGrName

Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

EntityData

Tipo: MQZED-entrada

Datos de entidad. Datos relacionados con la entidad que se va a comprobar. Para obtener más información, consulte [“MQZED-Descriptor de entidad” en la página 1705.](#)

EntityType

Tipo: MQLONG - entrada

Tipo de entidad. El tipo de entidad especificado por EntityData. Tiene que ser uno de los valores siguientes:

MQZAET_PRINCIPAL

Director.

GRUPO_MQZAC

group.

ComponentData

Tipo: MQBYTEComponentDataLength - entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro **ComponentDataLength** de la llamada MQZ_INIT_AUTHORITY.

continuación

Tipo: MQLONG - salida

Indicador de continuación establecido por componente. Se pueden especificar los siguientes valores:

MQZCI_DEFAULT

Continuación dependiente del gestor de colas.

Para MQZ_CHECK_AUTHORITY, tiene el mismo efecto que MQZCI_STOP.

MQZCI_CONTINUAR

Continúe con el componente siguiente.

MQZCI_STOP

No continúe con el componente siguiente.

Si la llamada a un componente falla (es decir, *CompCode* devuelve MQCC_FAILED), y el parámetro *Continuación* es MQZCI_DEFAULT o MQZCI_CONTINUE, el gestor de colas continúa llamando a otros componentes si los hay.

Si la llamada es satisfactoria (es decir, *CompCode* devuelve MQCC_OK) no se llama a ningún otro componente independientemente del valor de *Continuación*.

Si la llamada falla y el parámetro *Continuación* es MQZCI_STOP, no se llama a ningún otro componente y se devuelve el error al gestor de colas. Los componentes no conocen las llamadas anteriores, por lo que el parámetro *Continuación* siempre se establece en MQZCI_DEFAULT antes de la llamada.

CompCode

Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_NO_PRIVILEGIADO

(2584, X'A18') Este usuario no es un ID de usuario privilegiado.

MQRC_ENTIDAD_DESCONOCIDA

(2292, X'8F4') Entidad desconocida para el servicio.

MQRC_SERVICE_ERROR

(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICIO_NO_DISPONIBLE

(2285, X'8ED') Servicio subyacente no disponible.

Para obtener más información sobre estos códigos de razón, consulte [Códigos de terminación y de razón de la API](#).

Invocación en C

```
MQZ_CHECK_PRIVILEGED (QMgrName, &EntityData, EntityType,  
                      ComponentData, &Continuation,  
                      &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQZED     EntityData;         /* Entity name */  
MQLONG    EntityType;         /* Entity type */  
MQBYTE    ComponentData[n];   /* Component data */  
MQLONG    Continuation;       /* Continuation indicator set by  
                               component */  
MQLONG    CompCode;           /* Completion code */  
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

MQZ_COPY_ALL_AUTHORITY-Copiar todas las autorizaciones

Esta función la proporciona un componente de servicio de autorización. Lo inicia el gestor de colas para copiar todas las autorizaciones que están actualmente en vigor para un objeto de referencia en otro objeto.

El identificador de función para esta función (para MQZEP) es MQZID_COPY_ALL_AUTHORITY.

Sintaxis

```
MQZ_COPY_ALL_AUTHORITY( QMgrName , RefObjectName , ObjectName , ObjectType ,  
                        ComponentData , Continuation , CompCode , Reason )
```

Parámetros

QMgrName

Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

RefObjectName

Tipo: MQCHAR48 -entrada

Nombre de objeto de referencia. El nombre del objeto de referencia, cuyas autorizaciones se van a copiar. La longitud máxima de la serie es de 48 caracteres; si es más corta, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

ObjectName

Tipo: MQCHAR48 -entrada

El nombre del objeto. El nombre del objeto para el que se van a establecer los accesos. La longitud máxima de la serie es de 48 caracteres; si es más corta, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

ObjectType

Tipo: MQLONG - entrada

Tipo de objeto. El tipo de entidad especificado por *RefObjectName* y *ObjectName*. Tiene que ser uno de los valores siguientes:

MQOT_AUTH_INFO

Información de autenticación.

MQOT_CHANNEL

Canal.

MQOT_CLNTCONN_CHANNEL

Canal de conexión de cliente.

MQOT_ESCUCHA

Escucha.

MQOT_LISTA_DE_NOMBRES

Lista de nombres.

MQOT_PROCESO

.

MQOT_Q

Cola.

MQOT_Q_MGR

Gestor de colas.

SERVICIO MQOT_SERVICE

.

MQOT_TEMA

.

ComponentData

Tipo: MQBYTEExComponentDataLength - entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro `ComponentDataLength` de la llamada `MQZ_INIT_AUTHORITY`.

continuación

Tipo: `MQLONG` - salida

Indicador de continuación establecido por componente. Se pueden especificar los siguientes valores:

MQZCI_DEFAULT

Continuación dependiente del gestor de colas.

Para `MQZ_CHECK_AUTHORITY`, tiene el mismo efecto que `MQZCI_STOP`.

MQZCI_CONTINUAR

Continúe con el componente siguiente.

MQZCI_STOP

No continúe con el componente siguiente.

CompCode

Tipo: `MQLONG` - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: `MQLONG` - salida

Código de razón que califica *CompCode*.

Si *CompCode* es `MQCC_OK`:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es `MQCC_FAILED`:

MQRC_SERVICE_ERROR

(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICIO_NO_DISPONIBLE

(2285, X'8ED') Servicio subyacente no disponible.

MQRC_OBJETO_DE_REFERENCIA_DESCONOCIDO

(2294, X'8F6') Objeto de referencia desconocido.

Para obtener más información sobre estos códigos de razón, consulte [Códigos de terminación y de razón de la API](#).

Invocación en C

```
MQZ_COPY_ALL_AUTHORITY (QMgrName, RefObjectName, ObjectName, ObjectType,
                        ComponentData, &Continuation, &CompCode,
                        &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQCHAR48  QMgrName;           /* Queue manager name */
MQCHAR48  RefObjectName;       /* Reference object name */
MQCHAR48  ObjectName;         /* Object name */
MQLONG    ObjectType;         /* Object type */
MQBYTE    ComponentData[n];   /* Component data */
MQLONG    Continuation;       /* Continuation indicator set by
                               component */
MQLONG    CompCode;           /* Completion code */
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

MQZ_DELETE_AUTHORITY-Suprimir autorización

Esta función la proporciona un componente de servicio de autorización y la inicia el gestor de colas para suprimir todas las autorizaciones asociadas con el objeto especificado.

El identificador de función para esta función (para MQZEP) es MQZID_DELETE_AUTHORITY.

Sintaxis

MQZ_DELETE_AUTHORITY(QMgrName , ObjectName , ObjectType , ComponentData ,
Continuation , CompCode , Reason)

Parámetros

QMgrName

Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

ObjectName

Tipo: MQCHAR48 -entrada

El nombre del objeto. El nombre del objeto para el que se van a suprimir los accesos. La longitud máxima de la serie es de 48 caracteres; si es más corta, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

Si *ObjectType* es MQOT_Q_MGR, este nombre es el mismo que *QMgrName*.

ObjectType

Tipo: MQLONG - entrada

Tipo de objeto. El tipo de entidad especificado por *ObjectName*. Tiene que ser uno de los valores siguientes:

MQOT_AUTH_INFO

Información de autenticación.

MQOT_CHANNEL

Canal.

MQOT_CLNTCONN_CHANNEL

Canal de conexión de cliente.

MQOT_ESCUCHA

Escucha.

MQOT_LISTA_DE_NOMBRES

Lista de nombres.

MQOT_PROCESO

.

MQOT_Q

Cola.

MQOT_Q_MGR

Gestor de colas.

SERVICIO MQOT_SERVICE

.

MQOT_TEMA

.

ComponentData

Tipo: MQBYTE x ComponentDataLength - entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro ComponentDataLength de la llamada MQZ_INIT_AUTHORITY.

continuación

Tipo: MQLONG - salida

Indicador de continuación establecido por componente. Se pueden especificar los siguientes valores:

MQZCI_DEFAULT

Continuación dependiente del gestor de colas.

Para MQZ_CHECK_AUTHORITY, tiene el mismo efecto que MQZCI_STOP.

MQZCI_CONTINUAR

Continúe con el componente siguiente.

MQZCI_STOP

No continúe con el componente siguiente.

CompCode

Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_SERVICE_ERROR

(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICIO_NO_DISPONIBLE

(2285, X'8ED') Servicio subyacente no disponible.

Para obtener más información sobre estos códigos de razón, consulte [Códigos de terminación y de razón de la API](#).

Invocación en C

```
MQZ_DELETE_AUTHORITY (QMgrName, ObjectName, ObjectType, ComponentData,  
                      &Continuation, &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQCHAR48 QMgrName;          /* Queue manager name */  
MQCHAR48 ObjectName;        /* Object name */  
MQLONG   ObjectType;        /* Object type */  
MQBYTE   ComponentData[n]; /* Component data */
```

```

MQLONG    Continuation;      /* Continuation indicator set by
                               component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */

```

MQZ_ENUMERATE_AUTHORITY_DATA-Enumerar datos de autorización

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_4 y la inicia repetidamente el gestor de colas para recuperar todos los datos de autorización que coinciden con los criterios de selección especificados en la primera invocación.

El identificador de función para esta función (para MQZEP) es MQZID_ENUMERATE_AUTHORITY_DATA.

Sintaxis

```

MQZ_ENUMERATE_AUTHORITY_DATA( QMgrName , StartEnumeration , Filter ,
                               AuthorityBufferLength , AuthorityBuffer , AuthorityDataLength , ComponentData ,
                               Continuation , CompCode , Reason )

```

Parámetros

QMgrName

Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

StartEnumeration

Tipo: MQLONG - entrada

Distintivo que indica si la llamada puede iniciar la enumeración. Esto indica si la llamada puede iniciar la enumeración de datos de autorización o continuar la enumeración de datos de autorización iniciada por una llamada anterior a MQZ_ENUMERATE_AUTHORITY_DATA. El valor es uno de los valores siguientes:

MQZSE_START

Iniciar enumeración. La llamada se inicia con este valor para iniciar la enumeración de datos de autorización. El parámetro **Filter** especifica los criterios de selección que se deben utilizar para seleccionar los datos de autorización devueltos por esta y por llamadas sucesivas.

MQZSE_CONTINUAR

Continuar enumeración. La llamada se inicia con este valor para continuar la enumeración de datos de autorización. El parámetro **Filter** se ignora en este caso y se puede especificar como puntero nulo (los criterios de selección se determinan mediante el parámetro **Filter** especificado por la llamada que tenía *StartEnumeration* establecido en MQZSE_START).

Filtro

Tipo: MQZAD-entrada

Filtro. Si *StartEnumeration* es MQZSE_START, *Filter* especifica los criterios de selección que se van a utilizar para seleccionar los datos de autorización que se van a devolver. Si *Filter* es el puntero nulo, no se utiliza ningún criterio de selección, es decir, se devuelven todos los datos de autorización. Consulte “MQZAD-Datos de autorización” en la [página 1702](#) para obtener detalles de los criterios de selección que se pueden utilizar.

Si *StartEnumeration* es MQZSE_CONTINUE, *Filter* se ignora y se puede especificar como puntero nulo.

AuthorityBufferLength

Tipo: MQLONG - entrada

Longitud de *AuthorityBuffer*. Es la longitud en bytes del parámetro **AuthorityBuffer**. El almacenamiento intermedio de autorización debe ser lo suficientemente grande para dar cabida a los datos que se van a devolver.

AuthorityBuffer

Tipo: MQZAD-salida

Datos de autorización. Es el almacenamiento intermedio en el que se devuelven los datos de autorización. El almacenamiento intermedio debe ser lo suficientemente grande para acomodar una estructura MQZAD, una estructura MQZED, más el nombre de entidad más largo y el nombre de dominio más largo definido.

Nota: Nota: Este parámetro se define como un MQZAD, ya que el MQZAD siempre se produce al principio del almacenamiento intermedio. Sin embargo, si el almacenamiento intermedio se declara como MQZAD, el almacenamiento intermedio será demasiado pequeño; debe ser mayor que un MQZAD para que pueda acomodar los nombres de entidad y dominio MQZAD, MQZED, más.

AuthorityDataLength

Tipo: MQLONG - salida

Longitud de los datos devueltos en *AuthorityBuffer*. Si el almacenamiento intermedio de autorización es demasiado pequeño, *AuthorityDataLength* se establece en la longitud del almacenamiento intermedio necesaria y la llamada devuelve el código de terminación MQCC_FAILED y el código de razón MQRC_BUFFER_LENGTH_ERROR.

ComponentData

Tipo: MQBYTE x ComponentDataLength - entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro ComponentDataLength de la llamada MQZ_INIT_AUTHORITY.

continuación

Tipo: MQLONG - salida

Indicador de continuación establecido por componente. Se pueden especificar los siguientes valores:

MQZCI_DEFAULT

Continuación dependiente del gestor de colas.

Para MQZ_ENUMERATE_AUTHORITY_DATA, tiene el mismo efecto que MQZCI_CONTINUE.

MQZCI_CONTINUAR

Continúe con el componente siguiente.

MQZCI_STOP

No continúe con el componente siguiente.

CompCode

Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_BUFFER_LENGTH_ERROR

(2005, X'7D5') El parámetro de longitud de almacenamiento intermedio no es válido.

MQRC_NO_HAY_DATOS_DISPONIBLES

(2379, X'94B') No hay datos disponibles.

MQRC_SERVICE_ERROR

(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

Para obtener más información sobre estos códigos de razón, consulte [Códigos de terminación y de razón de la API](#).

Invocación en C

```
MQZ_ENUMERATE_AUTHORITY_DATA (QMgrName, StartEnumeration, &Filter,
                               AuthorityBufferLength,
                               &AuthorityBuffer,
                               &AuthorityDataLength, ComponentData,
                               &Continuation, &CompCode,
                               &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQCHAR48  QMgrName;           /* Queue manager name */
MQLONG    StartEnumeration;    /* Flag indicating whether call should
                               start enumeration */

MQZAD     Filter;              /* Filter */
MQLONG    AuthorityBufferLength; /* Length of AuthorityBuffer */
MQZAD     AuthorityBuffer;     /* Authority data */
MQLONG    AuthorityDataLength; /* Length of data returned in
                               AuthorityBuffer */

MQBYTE    ComponentData[n];    /* Component data */
MQLONG    Continuation;        /* Continuation indicator set by
                               component */

MQLONG    CompCode;            /* Completion code */
MQLONG    Reason;              /* Reason code qualifying CompCode */
```

MQZ_FREE_USER-Usuario libre

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_5 y la inicia el gestor de colas para liberar el recurso asignado asociado.

Se inicia cuando una aplicación ha terminado de ejecutarse en todos los contextos de usuario; por ejemplo, durante una llamada a la función `MQDISC` MQI.

El identificador de función para esta función (para MQZEP) es MQZID_FREE_USER.

Sintaxis

```
MQZ_FREE_USER( QMgrName , FreeParms , ComponentData , Continuation , CompCode ,
               Reason )
```

Parámetros**QMgrName**

Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

FreeParms

Tipo: MQZFP-entrada

Parámetros libres. Estructura que contiene datos relacionados con el recurso que se va a liberar. Consulte [“MQZFP-Parámetros libres”](#) en la página 1708 para obtener los detalles.

ComponentData

Tipo: MQBYTE x ComponentDataLength - entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro ComponentDataLength de la llamada MQZ_INIT_AUTHORITY.

continuación

Tipo: MQLONG - salida

Distintivo de continuación. Se pueden especificar los siguientes valores:

MQZCI_DEFAULT

Continuación dependiente de otros componentes.

MQZCI_STOP

No continúe con el componente siguiente.

CompCode

Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_SERVICE_ERROR

(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

Para obtener más información sobre estos códigos de razón, consulte [Códigos de terminación y de razón de la API](#).

Invocación en C

```
MQZ_AUTHENTICATE_USER (QMgrName, SecurityParms, ApplicationContext,  
                      IdentityContext, CorrelationPtr, ComponentData,  
                      &Continuation, &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQZFP     FreeParms;          /* Resource to be freed */  
MQBYTE    ComponentData[n];   /* Component data */  
MQLONG    Continuation;       /* Continuation indicator set by  
                               component */
```

MQLONG	CompCode;	/* Completion code */
MQLONG	Reason;	/* Reason code qualifying CompCode */

MQZ_GET_AUTHORITY-Obtener autorización

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_1 y la inicia el gestor de colas para recuperar la autorización que una entidad tiene para acceder al objeto especificado, incluidas (si la entidad es un principal) las autorizaciones que poseen los grupos de los que el principal es miembro. Las autorizaciones de perfiles genéricos se incluyen en el conjunto de autorizaciones devuelto.

El identificador de función para esta función (para MQZEP) es MQZID_GET_AUTHORITY.

Sintaxis

```
MQZ_GET_AUTHORITY( QMgrName , EntityName , EntityType , ObjectName ,
ObjectType , Authority , ComponentData , Continuation , CompCode , Reason )
```

Parámetros

QMgrName

Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

EntityName

Tipo: MQCHAR12 -entrada

Nombre de entidad. El nombre de la entidad cuyo acceso al objeto se va a recuperar. La longitud máxima de la serie es de 12 caracteres; si es más corta que esa, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

EntityType

Tipo: MQLONG - entrada

Tipo de entidad. El tipo de entidad especificado por *EntityName*. Tiene que ser uno de los valores siguientes:

MQZAET_PRINCIPAL

Director.

GRUPO_MQZAC

group.

ObjectName

Tipo: MQCHAR48 -entrada

El nombre del objeto. El nombre del objeto al que se va a recuperar el acceso. La longitud máxima de la serie es de 48 caracteres; si es más corta, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

Si *ObjectType* es MQOT_Q_MGR, este nombre es el mismo que *QMgrName*.

ObjectType

Tipo: MQLONG - entrada

Tipo de objeto. El tipo de entidad especificado por *ObjectName*. Tiene que ser uno de los valores siguientes:

MQOT_AUTH_INFO

Información de autenticación.

MQOT_CHANNEL

Canal.

MQOT_CLNTCONN_CHANNEL

Canal de conexión de cliente.

MQOT_ESCUCHA

Escucha.

MQOT_LISTA_DE_NOMBRES

Lista de nombres.

MQOT_PROCESO

.

MQOT_Q

Cola.

MQOT_Q_MGR

Gestor de colas.

SERVICIO MQOT_SERVICE

.

MQOT_TEMA

.

Autorización

Tipo: MQLONG - entrada

Autoridad de entidad. Si la entidad tiene una autorización, este campo es igual a la operación de autorización adecuada (constante MQZAO_*). Si tiene más de una autorización, este campo es el OR a nivel de bit de las constantes MQZAO_* correspondientes.

ComponentData

Tipo: MQBYTE×ComponentDataLength - entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro **ComponentDataLength** de la llamada MQZ_INIT_AUTHORITY.

continuación

Tipo: MQLONG - salida

Indicador de continuación establecido por componente. Se pueden especificar los siguientes valores:

MQZCI_DEFAULT

Continuación dependiente del gestor de colas.

Para MQZ_GET_AUTHORITY, tiene el mismo efecto que MQZCI_CONTINUE.

MQZCI_CONTINUAR

Continúe con el componente siguiente.

MQZCI_STOP

No continúe con el componente siguiente.

CompCode

Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_NOT_AUTHORIZED

(2035, X'7F3') Acceso no autorizado.

MQRC_SERVICE_ERROR

(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICIO_NO_DISPONIBLE

(2285, X'8ED') Servicio subyacente no disponible.

MQRC_ENTIDAD_DESCONOCIDA

(2292, X'8F4') Entidad desconocida para el servicio.

Para obtener más información sobre estos códigos de razón, consulte [Códigos de terminación y de razón de la API](#).

Invocación en C

```
MQZ_GET_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName,  
                   ObjectType, &Authority, ComponentData,  
                   &Continuation, &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQCHAR12  EntityName;         /* Entity name */  
MQLONG    EntityType;         /* Entity type */  
MQCHAR48  ObjectName;         /* Object name */  
MQLONG    ObjectType;         /* Object type */  
MQLONG    Authority;          /* Authority of entity */  
MQBYTE    ComponentData[n];   /* Component data */  
MQLONG    Continuation;       /* Continuation indicator set by  
                               component */  
MQLONG    CompCode;           /* Completion code */  
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

MQZ_GET_AUTHORITY_2 -Obtener autorización (ampliada)

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_2 y la inicia el gestor de colas para recuperar la autorización que una entidad tiene para acceder al objeto especificado.

El identificador de función para esta función (para MQZEP) es MQZID_GET_AUTHORITY.

MQZ_GET_AUTHORITY_2 es como MQZ_GET_AUTHORITY, pero con el parámetro **EntityName** sustituido por el parámetro **EntityData**.

Sintaxis

```
MQZ_GET_AUTHORITY_2( QMgrName , EntityData , EntityType , ObjectName ,  
                    ObjectType , Authority , ComponentData , Continuation , CompCode , Reason )
```

Parámetros

QMgrName

Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

EntityData

Tipo: MQZED-entrada

Datos de entidad. Datos relacionados con la entidad para la que se va a recuperar la autorización para el objeto. Consulte [“MQZED-Descriptor de entidad”](#) en la [página 1705](#) para obtener los detalles.

EntityType

Tipo: MQLONG - entrada

Tipo de entidad. El tipo de entidad especificado por *EntityData*. Tiene que ser uno de los valores siguientes:

MQZAET_PRINCIPAL

Director.

GRUPO_MQZAC

group.

ObjectName

Tipo: MQCHAR48 -entrada

El nombre del objeto. El nombre del objeto para el que se va a recuperar la autorización de entidad. La longitud máxima de la serie es de 48 caracteres; si es más corta, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

Si *ObjectType* es MQOT_Q_MGR, este nombre es el mismo que *QMgrName*.

ObjectType

Tipo: MQLONG - entrada

Tipo de objeto. El tipo de entidad especificado por *ObjectName*. Tiene que ser uno de los valores siguientes:

MQOT_AUTH_INFO

Información de autenticación.

MQOT_CHANNEL

Canal.

MQOT_CLNTCONN_CHANNEL

Canal de conexión de cliente.

MQOT_ESCUCHA

Escucha.

MQOT_LISTA_DE_NOMBRES

Lista de nombres.

MQOT_PROCESO

.

MQOT_Q

Cola.

MQOT_Q_MGR

Gestor de colas.

SERVICIO MQOT_SERVICE

.

MQOT_TOPIC

.

Autorización

Tipo: MQLONG - entrada

Autoridad de entidad. Si la entidad tiene una autorización, este campo es igual a la operación de autorización adecuada (constante MQZAO_*). Si tiene más de una autorización, este campo es el OR a nivel de bit de las constantes MQZAO_* correspondientes.

ComponentData

Tipo: MQBYTE×ComponentDataLength - entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro **ComponentDataLength** de la llamada MQZ_INIT_AUTHORITY.

continuación

Tipo: MQLONG - salida

Indicador de continuación establecido por componente. Se pueden especificar los siguientes valores:

MQZCI_DEFAULT

Continuación dependiente del gestor de colas.

Para MQZ_CHECK_AUTHORITY, tiene el mismo efecto que MQZCI_STOP.

MQZCI_CONTINUAR

Continúe con el componente siguiente.

MQZCI_STOP

No continúe con el componente siguiente.

CompCode

Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_NOT_AUTHORIZED

(2035, X'7F3') Acceso no autorizado.

MQRC_SERVICE_ERROR

(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICIO_NO_DISPONIBLE

(2285, X'8ED') Servicio subyacente no disponible.

MQRC_ENTIDAD_DESCONOCIDA

(2292, X'8F4') Entidad desconocida para el servicio.

Para obtener más información sobre estos códigos de razón, consulte [Códigos de terminación y de razón de la API](#).

Invocación en C

```
MQZ_GET_AUTHORITY_2 (QMgrName, &EntityData, EntityType, ObjectName,  
                    ObjectType, &Authority, ComponentData,  
                    &Continuation, &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQZED     EntityData;         /* Entity data */  
MQLONG    EntityType;         /* Entity type */  
MQCHAR48  ObjectName;         /* Object name */  
MQLONG    ObjectType;         /* Object type */  
MQLONG    Authority;          /* Authority of entity */  
MQBYTE    ComponentData[n];   /* Component data */  
MQLONG    Continuation;       /* Continuation indicator set by  
                               component */  
MQLONG    CompCode;           /* Completion code */  
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

MQZ_GET_EXPLICIT_AUTHORITY-Obtener autorización explícita

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_1 y la inicia el gestor de colas para recuperar la autorización que una entidad tiene para acceder al objeto especificado, incluidas (si la entidad es un principal) las autorizaciones que poseen los grupos de los que el principal es miembro. Las autorizaciones de perfiles genéricos se incluyen en el conjunto de autorizaciones devuelto.

En AIX and Linux, para el gestor de autorizaciones sobre objetos (OAM) de IBM MQ incorporado, la autorización devuelta es la que sólo posee el grupo primario del principal.

El identificador de función para esta función (para MQZEP) es MQZID_GET_EXPLICIT_AUTHORITY.

Sintaxis

```
MQZ_GET_EXPLICIT_AUTHORITY( QMgrName , EntityName , EntityType , ObjectName ,  
ObjectType , Authority , ComponentData , Continuation , CompCode , Reason )
```

Parámetros

QMgrName

Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

EntityName

Tipo: MQCHAR12 -entrada

Nombre de entidad. El nombre de la entidad para la que se va a recuperar el acceso al objeto. La longitud máxima de la serie es de 12 caracteres; si es más corta que esa, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

EntityType

Tipo: MQLONG - entrada

Tipo de entidad. El tipo de entidad especificado por *EntityName*. Tiene que ser uno de los valores siguientes:

MQZAET_PRINCIPAL

Director.

GRUPO_MQZAC

group.

ObjectName

Tipo: MQCHAR48 -entrada

El nombre del objeto. El nombre del objeto para el que se va a recuperar la autorización de entidad. La longitud máxima de la serie es de 48 caracteres; si es más corta, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

Si *ObjectType* es MQOT_Q_MGR, este nombre es el mismo que *QMgrName*.

ObjectType

Tipo: MQLONG - entrada

Tipo de objeto. El tipo de entidad especificado por *ObjectName*. Tiene que ser uno de los valores siguientes:

MQOT_AUTH_INFO

Información de autenticación.

MQOT_CHANNEL

Canal.

MQOT_CLNTCONN_CHANNEL

Canal de conexión de cliente.

MQOT_ESCUCHA

Escucha.

MQOT_LISTA_DE_NOMBRES

Lista de nombres.

MQOT_PROCESO

.

MQOT_Q

Cola.

MQOT_Q_MGR

Gestor de colas.

SERVICIO MQOT_SERVICE

.

MQOT_TEMA

.

Autorización

Tipo: MQLONG - entrada

Autoridad de entidad. Si la entidad tiene una autorización, este campo es igual a la operación de autorización adecuada (constante MQZAO_*). Si tiene más de una autorización, este campo es el OR a nivel de bit de las constantes MQZAO_* correspondientes.

ComponentData

Tipo: MQBYTE x ComponentDataLength - entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro **ComponentDataLength** de la llamada MQZ_INIT_AUTHORITY.

continuación

Tipo: MQLONG - salida

Indicador de continuación establecido por componente. Se pueden especificar los siguientes valores:

MQZCI_DEFAULT

Continuación dependiente del gestor de colas.

Para MQZ_GET_AUTHORITY, tiene el mismo efecto que MQZCI_CONTINUE.

MQZCI_CONTINUAR

Continúe con el componente siguiente.

MQZCI_STOP

No continúe con el componente siguiente.

CompCode

Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_NOT_AUTHORIZED

(2035, X'7F3') Acceso no autorizado.

MQRC_SERVICE_ERROR

(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICIO_NO_DISPONIBLE

(2285, X'8ED') Servicio subyacente no disponible.

MQRC_ENTIDAD_DESCONOCIDA

(2292, X'8F4') Entidad desconocida para el servicio.

Para obtener más información sobre estos códigos de razón, consulte [Códigos de terminación y de razón de la API](#).

Invocación en C

```
MQZ_GET_EXPLICIT_AUTHORITY (QMgrName, EntityName, EntityType,
                             ObjectName, ObjectType, &Authority,
                             ComponentData, &Continuation,
                             &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQCHAR48  QMgrName;           /* Queue manager name */
MQCHAR12  EntityName;         /* Entity name */
MQLONG    EntityType;         /* Entity type */
MQCHAR48  ObjectName;        /* Object name */
MQLONG    ObjectType;         /* Object type */
MQLONG    Authority;          /* Authority of entity */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;       /* Continuation indicator set by
                               component */
MQLONG    CompCode;           /* Completion code */
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

MQZ_GET_EXPLICIT_AUTHORITY_2 -Obtener autorización explícita (ampliada)

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_2 y la inicia el gestor de colas para recuperar la autorización que un grupo con nombre tiene para acceder a un objeto especificado (pero sin la autorización adicional del grupo **nobody**), o la autorización que tiene el grupo primario del principal con nombre para acceder a un objeto especificado.

El identificador de función para esta función (para MQZEP) es MQZID_GET_EXPLICIT_AUTHORITY.

MQZ_GET_EXPLICIT_AUTHORITY_2 es como MQZ_GET_EXPLICIT_AUTHORITY, pero con el parámetro **EntityName** sustituido por el parámetro **EntityData**.

Sintaxis

MQZ_GET_EXPLICIT_AUTHORITY_2(*QMgrName* , *EntityData* , *EntityType* , *ObjectName* , *ObjectType* , *Authority* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parámetros

QMgrName

Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

EntityData

Tipo: MQZED-entrada

Datos de entidad. Datos relacionados con la entidad cuya autorización al objeto se va a recuperar. Consulte [“MQZED-Descriptor de entidad”](#) en la [página 1705](#) para obtener los detalles.

EntityType

Tipo: MQLONG - entrada

Tipo de entidad. El tipo de entidad especificado por *EntityData*. Tiene que ser uno de los valores siguientes:

MQZAET_PRINCIPAL

Director.

GRUPO_MQZAC

group.

ObjectName

Tipo: MQCHAR48 -entrada

El nombre del objeto. El nombre del objeto para el que se va a recuperar la autorización de entidad. La longitud máxima de la serie es de 48 caracteres; si es más corta, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

Si *ObjectType* es MQOT_Q_MGR, este nombre es el mismo que *QMgrName*.

ObjectType

Tipo: MQLONG - entrada

Tipo de objeto. El tipo de entidad especificado por *ObjectName*. Tiene que ser uno de los valores siguientes:

MQOT_AUTH_INFO

Información de autenticación.

MQOT_CHANNEL

Canal.

MQOT_CLNTCONN_CHANNEL

Canal de conexión de cliente.

MQOT_ESCUCHA

Escucha.

MQOT_LISTA_DE_NOMBRES

Lista de nombres.

MQOT_PROCESO

.

MQOT_Q

Cola.

MQOT_Q_MGR

Gestor de colas.

SERVICIO MQOT_SERVICE

.

MQOT_TEMA

.

Autorización

Tipo: MQLONG - entrada

Autoridad de entidad. Si la entidad tiene una autorización, este campo es igual a la operación de autorización adecuada (constante MQZAO_*). Si tiene más de una autorización, este campo es el OR a nivel de bit de las constantes MQZAO_* correspondientes.

ComponentData

Tipo: MQBYTE×ComponentDataLength - entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro **ComponentDataLength** de la llamada MQZ_INIT_AUTHORITY.

continuación

Tipo: MQLONG - salida

Indicador de continuación establecido por componente. Se pueden especificar los siguientes valores:

MQZCI_DEFAULT

Continuación dependiente del gestor de colas.

Para MQZ_CHECK_AUTHORITY, tiene el mismo efecto que MQZCI_STOP.

MQZCI_CONTINUAR

Continúe con el componente siguiente.

MQZCI_STOP

No continúe con el componente siguiente.

CompCode

Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_NOT_AUTHORIZED

(2035, X'7F3') Acceso no autorizado.

MQRC_SERVICE_ERROR

(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICIO_NO_DISPONIBLE

(2285, X'8ED') Servicio subyacente no disponible.

MQRC_ENTIDAD_DESCONOCIDA

(2292, X'8F4') Entidad desconocida para el servicio.

Para obtener más información sobre estos códigos de razón, consulte [Códigos de terminación y de razón de la API](#).

Invocación en C

```
MQZ_GET_EXPLICIT_AUTHORITY_2 (QMgrName, &EntityData, EntityType,  
                               ObjectName, ObjectType, &Authority,  
                               ComponentData, &Continuation,  
                               &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQZED     EntityData;         /* Entity data */  
MQLONG    EntityType;         /* Entity type */  
MQCHAR48  ObjectName;         /* Object name */  
MQLONG    ObjectType;         /* Object type */  
MQLONG    Authority;          /* Authority of entity */  
MQBYTE    ComponentData[n];   /* Component data */  
MQLONG    Continuation;       /* Continuation indicator set by  
                               component */  
MQLONG    CompCode;           /* Completion code */  
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

MQZ_INIT_AUTHORITY-Inicializar servicio de autorización

Esta función la proporciona un componente de servicio de autorización y la inicia el gestor de colas durante la configuración del componente. Se espera que llame a MQZEP para proporcionar información al gestor de colas.

El identificador de función para esta función (para MQZEP) es MQZID_INIT_AUTHORITY.

Sintaxis

`MQZ_INIT_AUTHORITY(Hconfig , Options , QMgrName , ComponentDataLength ,
ComponentData , Version , CompCode , Reason)`

Parámetros

Hconfig

Tipo: MQHCONFIG-entrada

Descriptor de contexto de configuración. Este descriptor de contexto representa el componente concreto que se está inicializando. Debe ser utilizado por el componente al llamar al gestor de colas con la función MQZEP.

Opciones

Tipo: MQLONG - entrada

Opciones de inicialización. Tiene que ser uno de los valores siguientes:

MQZIO_PRIMARY

Inicialización primaria.

MQZIO_SECUNDARIO

Inicialización secundaria.

QMgrName

Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

ComponentDataLength

Tipo: MQLONG - entrada

Longitud de los datos de componente. Longitud en bytes del área *ComponentData* . Esta longitud se define en los datos de configuración del componente.

ComponentData

Tipo: MQBYTE x ComponentDataLength - entrada/salida

Datos de componente. Se inicializa con todos los ceros antes de llamar a la función de inicialización primaria del componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones (incluida la función de inicialización) proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro **ComponentDataLength** de la llamada MQZ_INIT_AUTHORITY.

Versión

Tipo: MQLONG-entrada/salida

Número de versión. En la entrada de la función de inicialización, identifica el número de versión más alto al que da soporte el gestor de colas. La función de inicialización debe cambiar esto, si es necesario, a la versión de la interfaz a la que da soporte. Si en la devolución el gestor de colas no soporta la versión devuelta por el componente, llama a la función MQZ_TERM_AUTHORITY del componente y no hace más uso de este componente.

Se admiten los valores siguientes:

MQZAS_VERSION_1

Versión 1.

MQZAS_VERSION_2

Versión 2.

MQZAS_VERSION_3

Versión 3.

MQZAS_VERSION_4

Versión 4.

MQZAS_VERSION_5

Versión 5.

MQZAS_VERSION_6

IBM WebSphere MQ 6.

CompCode

Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_INICIALIZACIÓN_FALLIDA

(2286, X'8EE') La inicialización falló por una razón indefinida.

MQRC_SERVICIO_NO_DISPONIBLE

(2285, X'8ED') Servicio subyacente no disponible.

Para obtener más información sobre estos códigos de razón, consulte [Códigos de terminación y de razón de la API](#).

Invocación en C

```
MQZ_INIT_AUTHORITY (Hconfig, Options, QMgrName, ComponentDataLength,  
                    ComponentData, &Version, &CompCode,  
                    &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQHCONFIG  Hconfig;           /* Configuration handle */  
MQLONG     Options;           /* Initialization options */  
MQCHAR48   QMgrName;         /* Queue manager name */  
MQLONG     ComponentDataLength; /* Length of component data */  
MQBYTE     ComponentData[n]; /* Component data */  
MQLONG     Version;          /* Version number */  
MQLONG     CompCode;         /* Completion code */  
MQLONG     Reason;           /* Reason code qualifying CompCode */
```

MQZ_INQUIRE-Consultar servicio de autorización

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_5 y la inicia el gestor de colas para consultar la funcionalidad soportada.

Cuando se utilizan varios componentes de servicio, se llama a los componentes de servicio en orden inverso al orden en el que se instalaron.

El identificador de función para esta función (para MQZEP) es MQZID_INQUIRE.

Sintaxis

```
MQZ_INQUIRE( QMgrName , SelectorCount , Selectors , IntAttrCount , IntAttrrs ,  
             CharAttrLength , CharAttrrs , SelectorReturned , ComponentData , Continuation ,  
             CompCode , Reason )
```

Parámetros

QMgrName

Tipo: MQCHAR48 - entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

SelectorCount

Tipo: MQLONG - entrada

Número de selectores. Número de selectores proporcionados en el parámetro **Selectors**.

El valor debe estar en el rango de 0 a 256.

Selectores

Tipo: MQLONGxSelectorCount - entrada

Matriz de selectores. Cada selector identifica un atributo necesario y debe ser uno de los siguientes:

- MQIACF_INTERFACE_VERSION (entero)
- MQIACF_USER_ID_SUPPORT (entero)
- MQCACF_SERVICE_COMPONENT (carácter)

Los selectores se pueden especificar en cualquier orden. El número de selectores de la matriz se indica mediante el parámetro **SelectorCount**.

Los atributos enteros identificados por los selectores se devuelven en el parámetro **IntAttrrs** en el mismo orden en que aparecen en *Selectors*.

Los atributos de carácter identificados por los selectores se devuelven en el parámetro **CharAttrrs** en el mismo orden en que aparecen *Selectors*.

IntAttrCount

Tipo: MQLONG - entrada

Número de atributos enteros proporcionados en el parámetro **IntAttrrs**.

El valor debe estar en el rango de 0 a 256.

IntAttrrs

Tipo: MQLONG x IntAttrCount - salida

Atributos enteros. Matriz de atributos enteros. Los atributos de entero se devuelven en el mismo orden que los selectores de entero correspondientes en la matriz *Selectors*.

CharAttrCount

Tipo: MQLONG - entrada

Longitud del almacenamiento intermedio de atributos de caracteres. Longitud en bytes del parámetro **CharAttrrs**.

El valor debe ser como mínimo la suma de las longitudes de los atributos de caracteres solicitados. Si no se solicita ningún atributo de carácter, cero es un valor válido.

CharAttrrs

Tipo: MQLONG x CharAttrCount - salida

Almacenamiento intermedio de atributos de caracteres. Almacenamiento intermedio que contiene atributos de caracteres, concatenados entre sí. Los atributos de carácter se devuelven en el mismo orden que los selectores de caracteres correspondientes en la matriz *Selectors*.

La longitud del búfer viene dada por el parámetro **CharAttrCount**.

SelectorReturned

Tipo: MQLONG x SelectorCount -entrada

Se ha devuelto el selector. Matriz de valores que identifican qué atributos se han devuelto del conjunto solicitado por los selectores en el parámetro **Selectors**. El número de valores de esta matriz se indica mediante el parámetro **SelectorCount**. Cada valor en la matriz se relaciona con el selector desde la posición correspondiente en la matriz de selectores. Cada valor es uno de los siguientes:

MQZSL_DEVUELTO

Se ha devuelto el atributo solicitado por el selector correspondiente en el parámetro **Selectors**.

MQZSL_NO_DEVUELTO

El atributo solicitado por el selector correspondiente en el parámetro **Selectors** no se ha devuelto.

La matriz se inicializa con todos los valores como *MQZSL_NOT_RETURNED*. Cuando un componente de servicio de autorización devuelve un atributo, establece el valor adecuado en la matriz en *MQZSL_NOT_RETURNED*. Esto permite que cualquier otro componente de servicio de autorización, al que se realiza la llamada de consulta, identifique qué atributos ya se han devuelto.

ComponentData

Tipo: MQBYTE x ComponentDataLength - entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro **ComponentDataLength** de la llamada MQZ_INIT_AUTHORITY.

continuación

Tipo: MQLONG - salida

Indicador de continuación establecido por componente. Se pueden especificar los siguientes valores:

MQZCI_DEFAULT

Continuación dependiente del gestor de colas.

Para MQZ_CHECK_AUTHORITY, tiene el mismo efecto que MQZCI_STOP.

MQZCI_STOP

No continúe con el componente siguiente.

CompCode

Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_WARNING

Finalización parcial.

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_WARNING:

MQRC_CHAR_ATTRS_TOO_SHORT

No hay espacio suficiente para los atributos de carácter.

MQRC_INT_COUNT_TOO_SMALL

No hay suficiente espacio para atributos enteros.

Si *CompCode* es MQCC_FAILED:

MQRC_SELECTOR_COUNT_ERROR

El número de selectores no es válido.

MQRC_SELECTOR_ERROR

Selector de atributo no válido.

MQRC_SELECTOR_LIMIT_EXCEEDED

Se han especificado demasiados selectores.

MQRC_INT_ATTR_COUNT_ERROR

El número de atributos enteros no es válido.

MQRC_INT_ATTRS_ARRAY_ERROR

Matriz de atributos enteros no válida.

MQRC_CHAR_ATTR_LENGTH_ERROR

El número de atributos de tipo carácter no es válido.

MQRC_CHAR_ATTRS_ERROR

La serie de atributos de tipo carácter no es válida.

MQRC_SERVICE_ERROR

(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

Para obtener más información sobre estos códigos de razón, consulte [Códigos de terminación y de razón de la API](#).

Invocación en C

```
MQZ_INQUIRE (QMgrName, SelectorCount, Selectors, IntAttrCount,
              &IntAttrs, CharAttrLength, &CharAttrs,
              SelectorReturned, ComponentData, &Continuation,
              &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQCHAR48  QMgrName;           /* Queue manager name */
MQLONG    SelectorCount;      /* Selector count */
MQLONG    Selectors[n];       /* Selectors */
MQLONG    IntAttrCount;       /* IntAttrs count */
MQLONG    IntAttrs[n];        /* Integer attributes */
MQLONG    CharAttrCount;      /* CharAttrs count */
MQLONG    CharAttrs[n];       /* Character attributes */
MQLONG    SelectorReturned[n]; /* Selector returned */
MQBYTE    ComponentData[n];   /* Component data */
MQLONG    Continuation;       /* Continuation indicator set by
                               component */
MQLONG    CompCode;           /* Completion code */
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

MQZ_REFRESH_CACHE-Renovar todas las autorizaciones

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_3 y la invoca el gestor de colas para renovar la lista de autorizaciones mantenidas internamente por el componente.

El identificador de función para esta función (para MQZEP) es MQZID_REFRESH_CACHE (8L).

Sintaxis

```
MQZ_REFRESH_CACHE( QMgrName , ComponentData , Continuation , CompCode ,
Reason )
```

Parámetros

QMgrName

Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

ComponentData

Tipo: MQBYTE×ComponentDataLength - entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de las funciones de este componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro **ComponentDataLength** de la llamada MQZ_INIT_AUTHORITY.

continuación

Tipo: MQLONG - salida

Indicador de continuación establecido por componente. Se pueden especificar los siguientes valores:

MQZCI_DEFAULT

Continuación dependiente del gestor de colas.

Para MQZ_CHECK_AUTHORITY tiene el mismo efecto que MQZCI_STOP.

MQZCI_CONTINUAR

Continúe con el componente siguiente.

MQZCI_STOP

No continúe con el componente siguiente.

CompCode

Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_WARNING:

MQRC_SERVICE_ERROR

(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

Invocación en C

```
MQZ_REFRESH_CACHE (QMgrName, ComponentData,  
                  &Continuation, &CompCode, &Reason);
```


Declare los parámetros como se indica a continuación:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQBYTE    ComponentData[n]; /* Component data */
MQLONG    Continuation;     /* Continuation indicator set by
                             component */
MQLONG    CompCode;         /* Completion code */
MQLONG    Reason;          /* Reason code qualifying CompCode */
```

MQZ_SET_AUTHORITY-Establecer autorización

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_1 y la inicia el gestor de colas para establecer la autorización que una entidad tiene para acceder al objeto especificado.

El identificador de función para esta función (para MQZEP) es MQZID_SET_AUTHORITY.

Nota: Esta función altera temporalmente las autorizaciones existentes. Para conservar las autorizaciones existentes, debe volver a establecerlas con esta función.

Sintaxis

MQZ_SET_AUTHORITY(QMgrName , EntityName , EntityType , ObjectName ,
ObjectType , Authority , ComponentData , Continuation , CompCode , Reason)

Parámetros

QMgrName

Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

EntityName

Tipo: MQCHAR12 -entrada

Nombre de entidad. El nombre de la entidad para la que se va a recuperar el acceso al objeto. La longitud máxima de la serie es de 12 caracteres; si es más corta que esa, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

EntityType

Tipo: MQLONG - entrada

Tipo de entidad. El tipo de entidad especificado por *EntityName*. Tiene que ser uno de los valores siguientes:

MQZAET_PRINCIPAL

Director.

GRUPO_MQZAC

group.

ObjectName

Tipo: MQCHAR48 -entrada

El nombre del objeto. El nombre del objeto al que se necesita acceso. La longitud máxima de la serie es de 48 caracteres; si es más corta, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

Si *ObjectType* es MQOT_Q_MGR, este nombre es el mismo que *QMgrName*.

ObjectType

Tipo: MQLONG - entrada

Tipo de objeto. El tipo de entidad especificado por *ObjectName*. Tiene que ser uno de los valores siguientes:

MQOT_AUTH_INFO

Información de autenticación.

MQOT_CHANNEL

Canal.

MQOT_CLNTCONN_CHANNEL

Canal de conexión de cliente.

MQOT_ESCUCHA

Escucha.

MQOT_LISTA_DE_NOMBRES

Lista de nombres.

MQOT_PROCESO

.

MQOT_Q

Cola.

MQOT_Q_MGR

Gestor de colas.

SERVICIO MQOT_SERVICE

.

MQOT_TEMA

.

Autorización

Tipo: MQLONG - entrada

Autoridad de entidad. Si se establece una autorización, este campo es igual a la operación de autorización adecuada (constante MQZAO_*). Si se establece más de una autorización, este campo es el OR a nivel de bit de las constantes MQZAO_* correspondientes.

ComponentDataname>

Tipo: MQBYTEComponentDataLength - entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro **ComponentDataLength** de la llamada MQZ_INIT_AUTHORITY.

continuación

Tipo: MQLONG - salida

Indicador de continuación establecido por componente. Se pueden especificar los siguientes valores:

MQZCI_DEFAULT

Continuación dependiente del gestor de colas.

Para MQZ_GET_AUTHORITY, tiene el mismo efecto que MQZCI_CONTINUE.

MQZCI_CONTINUAR

Continúe con el componente siguiente.

MQZCI_STOP

No continúe con el componente siguiente.

CompCode

Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_NOT_AUTHORIZED

(2035, X'7F3') Acceso no autorizado.

MQRC_SERVICE_ERROR

(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICIO_NO_DISPONIBLE

(2285, X'8ED') Servicio subyacente no disponible.

MQRC_ENTIDAD_DESCONOCIDA

(2292, X'8F4') Entidad desconocida para el servicio.

Para obtener más información sobre estos códigos de razón, consulte [Códigos de terminación y de razón de la API](#).

Invocación en C

```
MQZ_SET_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName,  
                  ObjectType, Authority, ComponentData,  
                  &Continuation, &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQCHAR12  EntityName;         /* Entity name */  
MQLONG    EntityType;         /* Entity type */  
MQCHAR48  ObjectName;         /* Object name */  
MQLONG    ObjectType;         /* Object type */  
MQLONG    Authority;          /* Authority to be checked */  
MQBYTE    ComponentData[n];   /* Component data */  
MQLONG    Continuation;       /* Continuation indicator set by  
                               component */  
MQLONG    CompCode;           /* Completion code */  
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

MQZ_SET_AUTHORITY_2 -Establecer autorización (ampliado)

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_2 y la inicia el gestor de colas para establecer la autorización que una entidad tiene para acceder al objeto especificado.

El identificador de función para esta función (para MQZEP) es MQZID_SET_AUTHORITY.

Nota: Esta función altera temporalmente las autorizaciones existentes. Para conservar las autorizaciones existentes, debe volver a establecerlas con esta función.

MQZ_SET_AUTHORITY_2 es como MQZ_SET_AUTHORITY, pero con el parámetro **EntityName** sustituido por el parámetro **EntityData**.

Sintaxis

```
MQZ_SET_AUTHORITY_2( QMgrName , EntityData , EntityType , ObjectName ,  
                    ObjectType , Authority , ComponentData , Continuation , CompCode , Reason )
```

Parámetros

QMgrName

Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

EntityData

Tipo: MQZED-entrada

Datos de entidad. Datos relacionados con la entidad cuya autorización para el objeto se va a establecer. Consulte [“MQZED-Descriptor de entidad”](#) en la página 1705 para obtener los detalles.

EntityType

Tipo: MQLONG - entrada

Tipo de entidad. El tipo de entidad especificado por *EntityData*. Tiene que ser uno de los valores siguientes:

MQZAET_PRINCIPAL

Director.

GRUPO_MQZAC

group.

ObjectName

Tipo: MQCHAR48 -entrada

El nombre del objeto. El nombre del objeto sobre el que se va a establecer la autorización de entidad. La longitud máxima de la serie es de 48 caracteres; si es más corta, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

Si *ObjectType* es MQOT_Q_MGR, este nombre es el mismo que *QMgrName*.

ObjectType

Tipo: MQLONG - entrada

Tipo de objeto. El tipo de entidad especificado por *ObjectName*. Tiene que ser uno de los valores siguientes:

MQOT_AUTH_INFO

Información de autenticación.

MQOT_CHANNEL

Canal.

MQOT_CLNTCONN_CHANNEL

Canal de conexión de cliente.

MQOT_ESCUCHA

Escucha.

MQOT_LISTA_DE_NOMBRES

Lista de nombres.

MQOT_PROCESO

.

MQOT_Q

Cola.

MQOT_Q_MGR

Gestor de colas.

SERVICIO MQOT_SERVICE

.

MQOT_TEMA

Autorización

Tipo: MQLONG - entrada

Autoridad de entidad. Si se establece una autorización, este campo es igual a la operación de autorización adecuada (constante MQZAO_*). Si se establece más de una autorización, este campo es el OR a nivel de bit de las constantes MQZAO_* correspondientes.

ComponentData

Tipo: MQBYTE×ComponentDataLength - entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro **ComponentDataLength** de la llamada MQZ_INIT_AUTHORITY.

continuación

Tipo: MQLONG - salida

Indicador de continuación establecido por componente. Se pueden especificar los siguientes valores:

MQZCI_DEFAULT

Continuación dependiente del gestor de colas.

Para MQZ_CHECK_AUTHORITY, tiene el mismo efecto que MQZCI_STOP.

MQZCI_CONTINUAR

Continúe con el componente siguiente.

MQZCI_STOP

No continúe con el componente siguiente.

CompCode

Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_NOT_AUTHORIZED

(2035, X'7F3') Acceso no autorizado.

MQRC_SERVICE_ERROR

(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICIO_NO_DISPONIBLE

(2285, X'8ED') Servicio subyacente no disponible.

MQRC_ENTIDAD_DESCONOCIDA

(2292, X'8F4') Entidad desconocida para el servicio.

Para obtener más información sobre estos códigos de razón, consulte [Códigos de terminación y de razón de la API](#).

Invocación en C

```
MQZ_SET_AUTHORITY_2 (QMgrName, &EntityData, EntityType, ObjectName,  
                    ObjectType, Authority, ComponentData,  
                    &Continuation, &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQZED     EntityData;         /* Entity data */  
MQLONG    EntityType;         /* Entity type */  
MQCHAR48  ObjectName;         /* Object name */  
MQLONG    ObjectType;         /* Object type */  
MQLONG    Authority;          /* Authority to be checked */  
MQBYTE    ComponentData[n];   /* Component data */  
MQLONG    Continuation;       /* Continuation indicator set by  
                               component */  
MQLONG    CompCode;           /* Completion code */  
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

MQZ_TERM_AUTHORITY-Terminar servicio de autorización

Esta función la proporciona un componente de servicio de autorización y la inicia el gestor de colas cuando ya no necesita los servicios de este componente. La función debe realizar cualquier limpieza necesaria para el componente.

El identificador de función para esta función (para MQZEP) es MQZID_TERM_AUTHORITY.

Sintaxis

MQZ_TERM_AUTHORITY([Hconfig](#) , [Options](#) , [QMgrName](#) , [ComponentData](#) , [CompCode](#) , [Reason](#))

Parámetros

Hconfig

Tipo: MQHCONFIG-entrada

Descriptor de contexto de configuración. Este descriptor de contexto representa el componente concreto que se está terminando. Debe ser utilizado por el componente al llamar al gestor de colas con la función MQZEP.

Opciones

Tipo: MQLONG - entrada

Opciones de terminación. Tiene que ser uno de los valores siguientes:

MQZTO_PRIMARY

Terminación primaria.

MQZTO_SECUNDARIO

Terminación secundaria.

QMgrName

Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

ComponentData

Tipo: MQBYTE x ComponentDataLength - entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro ComponentDataLength en la llamada MQZ_INIT_AUTHORITY.

Cuando se ha completado la llamada MQZ_TERM_AUTHORITY, el gestor de colas descarta estos datos.

CompCode

Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_SERVICIO_NO_DISPONIBLE

(2285, X'8ED') Servicio subyacente no disponible.

MQRC_TERMINACIÓN_FALLIDA

(2287, X'8FF') La finalización ha fallado por un motivo indefinido.

Para obtener más información sobre estos códigos de razón, consulte [Códigos de terminación y de razón de la API](#).

Invocación en C

```
MQZ_TERM_AUTHORITY (Hconfig, Options, QMgrName, ComponentData,
                    &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQHCONFIG  Hconfig;           /* Configuration handle */
MQLONG     Options;           /* Termination options */
MQCHAR48   QMgrName;         /* Queue manager name */
MQBYTE     ComponentData[n]; /* Component data */
MQLONG     CompCode;         /* Completion code */
MQLONG     Reason;           /* Reason code qualifying CompCode */
```

MQZ_DELETE_NAME-Suprimir nombre

Esta función la proporciona un componente de servicio de nombres y la inicia el gestor de colas para suprimir una entrada para la cola especificada.

El identificador de función para esta función (para MQZEP) es MQZID_DELETE_NAME.

Sintaxis

MQZ_DELETE_NAME(QMgrName , QName , ComponentData , Continuation , CompCode , Reason)

Parámetros

QMgrName

Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

QName

Tipo: MQCHAR48 -entrada

Nombre de cola. El nombre de la cola para la que se va a suprimir una entrada. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

ComponentData

Tipo: MQBYTE x ComponentDataLength - entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro ComponentDataLength en la llamada MQZ_INIT_NAME.

continuación

Tipo: MQLONG - salida

Indicador de continuación establecido por componente. Tiene que ser uno de los valores siguientes:

MQZCI_DEFAULT

Continuación dependiente del gestor de colas.

MQZCI_STOP

No continúe con el componente siguiente.

Para el mandato **MQZ_DELETE_NAME** , el gestor de colas no intenta iniciar otro componente, independientemente de lo que se devuelva en el parámetro **Continuation** .

CompCode

Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_WARNING

Aviso (finalización parcial).

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_WARNING:

MQRC_NOMBRE_DESCONOCIDO

(2288, X'8F0') Nombre de cola no encontrado.

Nota: Es posible que no sea posible devolver este código si el servicio subyacente responde correctamente en este caso.

Si *CompCode* es MQCC_FAILED:

MQRC_SERVICE_ERROR

(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICIO_NO_DISPONIBLE

(2285, X'8ED') Servicio subyacente no disponible.

Para obtener más información sobre estos códigos de razón, consulte [Códigos de terminación y de razón de la API](#).

Invocación en C

```
MQZ_DELETE_NAME (QMgrName, QName, ComponentData, &Continuation,  
                 &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQCHAR48  QName;              /* Queue name */  
MQBYTE    ComponentData[n];   /* Component data */  
MQLONG    Continuation;       /* Continuation indicator set by  
                               component */  
MQLONG    CompCode;           /* Completion code */  
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

MQZ_INIT_NAME-Inicializar servicio de nombres

Esta función la proporciona un componente de servicio de nombres y la inicia el gestor de colas durante la configuración del componente. Se espera que llame a MQZEP para proporcionar información al gestor de colas.

El identificador de función para esta función (para MQZEP) es MQZID_INIT_NAME.

Sintaxis

```
MQZ_INIT_NAME( Hconfig , Options , QMgrName , ComponentDataLength ,  
               ComponentData , Version , CompCode , Reason )
```

Parámetros

Hconfig

Tipo: MQHCONFIG-entrada

Descriptor de contexto de configuración. Este descriptor de contexto representa el componente concreto que se está inicializando. Debe ser utilizado por el componente al llamar al gestor de colas con la función MQZEP.

Opciones

Tipo: MQLONG - entrada

Opciones de inicialización. Tiene que ser uno de los valores siguientes:

MQZIO_PRIMARY

Inicialización primaria.

MQZIO_SECUNDARIO

Inicialización secundaria.

QMgrName

Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

ComponentDataLength

Tipo: MQLONG - entrada

Longitud de los datos de componente. Longitud en bytes del área *ComponentData* . Esta longitud se define en los datos de configuración del componente.

ComponentData

Tipo: MQBYTE x ComponentDataLength - entrada/salida

Datos de componente. Se inicializa con todos los ceros antes de llamar a la función de inicialización primaria del componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones (incluida la función de inicialización) proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro **ComponentDataLength** de la llamada MQZ_INIT_AUTHORITY.

Versión

Tipo: MQLONG-entrada/salida

Número de versión. En la entrada de la función de inicialización, identifica el número de versión más alto al que da soporte el gestor de colas. La función de inicialización debe cambiar esto, si es necesario, a la versión de la interfaz a la que da soporte. Si en el momento de la devolución el gestor de colas no soporta la versión devuelta por el componente, llama a la función MQZ_TERM_NAME del componente y no hace más uso de este componente.

Se admiten los valores siguientes:

MQZAS_VERSION_1

Versión 1.

CompCode

Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_INICIALIZACIÓN_FALLIDA

(2286, X'8EE') La inicialización falló por una razón indefinida.

MQRC_SERVICIO_NO_DISPONIBLE

(2285, X'8ED') Servicio subyacente no disponible.

Para obtener más información sobre estos códigos de razón, consulte [Códigos de terminación y de razón de la API](#).

Invocación en C

```
MQZ_INIT_NAME (Hconfig, Options, QMgrName, ComponentDataLength,  
               ComponentData, &Version, &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQHCONFIG    Hconfig;           /* Configuration handle */  
MQLONG       Options;           /* Initialization options */  
MQCHAR48     QMgrName;          /* Queue manager name */  
MQLONG       ComponentDataLength; /* Length of component data */  
MQBYTE       ComponentData[n];  /* Component data */  
MQLONG       Version;           /* Version number */  
MQLONG       CompCode;          /* Completion code */  
MQLONG       Reason;            /* Reason code qualifying CompCode */
```

MQZ_INSERT_NAME-Insertar nombre

Esta función la proporciona un componente de servicio de nombres y la inicia el gestor de colas para insertar una entrada para la cola especificada, que contiene el nombre del gestor de colas propietario de la cola. Si la cola ya está definida en el servicio, la llamada falla.

El identificador de función para esta función (para MQZEP) es MQZID_INSERT_NAME.

Sintaxis

```
MQZ_INSERT_NAME( QMgrName , QName , ResolvedQMgrName , ComponentData ,  
                 Continuation , CompCode , Reason )
```

Parámetros

QMgrName

Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

QName

Tipo: MQCHAR48 -entrada

Nombre de cola. El nombre de la cola para la que se va a insertar una entrada. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

ResolvedQMgrName

Tipo: MQCHAR48 -entrada

Nombre de gestor de colas resuelto. El nombre del gestor de colas en el que se resuelve la cola. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

ComponentData

Tipo: MQBYTE×ComponentDataLength - entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones (incluida la función de inicialización) proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro **ComponentDataLength** de la llamada MQZ_INIT_NAME.

continuación

Tipo: MQLONG-entrada/salida

Indicador de continuación establecido por componente. Para MQZ_INSERT_NAME, el gestor de colas no intenta iniciar otro componente, lo que se devuelve en el parámetro **Continuation**.

Se admiten los valores siguientes:

MQZCI_DEFAULT

Continuación dependiente del gestor de colas.

MQZCI_STOP

No continúe con el componente siguiente.

CompCode

Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_Q_YA_EXISTE

(2290, X'8F2') El objeto de cola ya existe.

MQRC_SERVICE_ERROR

(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICIO_NO_DISPONIBLE

(2285, X'8ED') Servicio subyacente no disponible.

Para obtener más información sobre estos códigos de razón, consulte [Códigos de terminación y de razón de la API](#).

Invocación en C

```
MQZ_INSERT_NAME (QMgrName, QName, ResolvedQMgrName, ComponentData,
                 &Continuation, &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQCHAR48  QMgrName;      /* Queue manager name */
MQCHAR48  QName;         /* Queue name */
MQCHAR48  ResolvedQMgrName; /* Resolved queue manager name */
MQBYTE    ComponentData[n]; /* Component data */
MQLONG    Continuation;   /* Continuation indicator set by
                           component */
```

MQLONG	CompCode;	/* Completion code */
MQLONG	Reason;	/* Reason code qualifying CompCode */

MQZ_LOOKUP_NAME-Nombre de búsqueda

Esta función la proporciona un componente de servicio de nombres y la inicia el gestor de colas para recuperar el nombre del gestor de colas propietario, para una cola especificada.

El identificador de función para esta función (para MQZEP) es MQZID_LOOKUP_NAME.

Sintaxis

MQZ_LOOKUP_NAME(*QMgrName* , *QName* , *ResolvedQMgrName* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parámetros

QMgrName

Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

QName

Tipo: MQCHAR48 -entrada

Nombre de cola. El nombre de la cola para la que se va a resolver una entrada. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

ResolvedQMgrName

Tipo: MQCHAR48 -salida

Nombre de gestor de colas resuelto. Si la función se completa correctamente, este es el nombre del gestor de colas propietario de la cola.

El nombre devuelto por el componente de servicio debe rellenarse a la derecha con espacios en blanco hasta la longitud completa del parámetro; el nombre no debe terminar con un carácter nulo, ni contener espacios en blanco iniciales o intercalados.

ComponentData

Tipo: MQBYTEExComponentDataLength - entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones (incluida la función de inicialización) proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro **ComponentDataLength** de la llamada MQZ_INIT_NAME.

continuación

Tipo: MQLONG - salida

Indicador de continuación establecido por componente. Para MQZ_LOOKUP_NAME, el gestor de colas especifica si se debe iniciar otro componente de servicio de nombres, como se indica a continuación:

- Si *CompCode* es MQCC_OK, no se inician más componentes, independientemente del valor que se devuelva en *Continuation*.
- Si *CompCode* no es MQCC_OK, se inicia un componente adicional, a menos que *Continuation* sea MQZCI_STOP.

Se admiten los valores siguientes:

MQZCI_DEFAULT

Continuación dependiente del gestor de colas.

MQZCI_CONTINUAR

Continúe con el componente siguiente.

MQZCI_STOP

No continúe con el componente siguiente.

CompCode

Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_SERVICE_ERROR

(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICIO_NO_DISPONIBLE

(2285, X'8ED') Servicio subyacente no disponible.

MQRC_NOMBRE_Q_DESCONOCIDO

(2288, X'8F0') Nombre de cola no encontrado.

Para obtener más información sobre estos códigos de razón, consulte [Códigos de terminación y de razón de la API](#).

Invocación en C

```
MQZ_LOOKUP_NAME (QMgrName, QName, ResolvedQMgrName, ComponentData,  
                 &Continuation, &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQCHAR48  QName;              /* Queue name */  
MQCHAR48  ResolvedQMgrName;    /* Resolved queue manager name */  
MQBYTE    ComponentData[n];    /* Component data */  
MQLONG     Continuation;        /* Continuation indicator set by  
                                component */  
MQLONG     CompCode;            /* Completion code */  
MQLONG     Reason;             /* Reason code qualifying CompCode */
```

MQZ_TERM_NAME-Terminar servicio de nombres

Esta función la proporciona un componente de servicio de nombres y la inicia el gestor de colas cuando ya no necesita los servicios de este componente. La función debe realizar cualquier limpieza necesaria para el componente.

El identificador de función para esta función (para MQZEP) es MQZID_TERM_NAME.

Sintaxis

MQZ_TERM_NAME(Hconfig , Options , QMgrName , ComponentData , CompCode , Reason)

Parámetros

Hconfig

Tipo: MQHCONFIG-entrada

Descriptor de contexto de configuración. Este descriptor de contexto representa el componente concreto que se está terminando. Lo utiliza el componente al llamar al gestor de colas con la función MQZEP.

Opciones

Tipo: MQLONG - entrada

Opciones de terminación. Tiene que ser uno de los valores siguientes:

MQZTO_PRIMARY

Terminación primaria.

MQZTO_SECUNDARIO

Terminación secundaria.

QMgrName

Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

ComponentData

Tipo: MQBYTE x ComponentDataLength - entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones (incluida la función de inicialización) proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

Los datos de componente están en memoria compartida accesible para todos los procesos.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro **ComponentDataLength** de la llamada MQZ_INIT_NAME.

Cuando se ha completado la llamada MQZ_TERM_NAME, el gestor de colas descarta estos datos.

CompCode

Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_TERMINACIÓN_FALLIDA

(2287, X'8FF') La finalización ha fallado por un motivo indefinido.

MQRC_SERVICIO_NO_DISPONIBLE

(2285, X'8ED') Servicio subyacente no disponible.

Para obtener más información sobre estos códigos de razón, consulte [Códigos de terminación y de razón de la API](#).

Invocación en C

```
MQZ_TERM_NAME (Hconfig, Options, QMgrName, ComponentData, &CompCode,  
               &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQHCONFIG  Hconfig;          /* Configuration handle */  
MQLONG     Options;         /* Termination options */  
MQCHAR48   QMgrName;        /* Queue manager name */  
MQBYTE     ComponentData[n]; /* Component data */  
MQLONG     CompCode;         /* Completion code */  
MQLONG     Reason;          /* Reason code qualifying CompCode */
```

MQZAC-Contexto de aplicación

La estructura MQZAC se utiliza en la llamada MQZ_AUTHENTICATE_USER para el parámetro *ApplicationContext* . Este parámetro especifica los datos relacionados con la aplicación de llamada.

[Tabla 1](#) resume los campos de la estructura.

Tabla 874. Campos en MQZAC	
Campo	Descripción
StrucId	Identificador de la estructura
Versión	Número de versión de la estructura
ProcessId	Identificador proceso
ThreadId	Identificador de hebra
ApplName	Nombre de la aplicación
UserID	Identificador de usuario
EffectiveUserID	Identificador de usuario efectivo
Entorno	Entorno
CallerType	Tipo de interlocutor
AuthenticationType	Tipo de autenticación
BindType	tipo de enlace

Campos

StrucId

Tipo: MQCHAR4 -entrada

Identificador de estructura. El valor es el siguiente:

MQZAC_STRUC_ID

Identificador de la estructura de contexto de aplicación.

Para el lenguaje de programación C, también se define la constante MQZAC_STRUC_ID_ARRAY; tiene el mismo valor que MQZAC_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Versión

Tipo: MQLONG - entrada

Número de versión de la estructura. El valor es el siguiente:

MQZAC_VERSION_1

Estructura de contexto de aplicación Version-1 . La constante MQZAC_CURRENT_VERSION especifica el número de versión de la versión actual.

ProcessId

Tipo: MQPID-entrada

Identificador de proceso de la aplicación.

ThreadId

Tipo: MQTID-entrada

Identificador de hebra de la aplicación.

ApplName

Tipo: MQCHAR28 -entrada

Nombre de aplicación.

UserID

Tipo: MQCHAR12 -entrada

Identificador de usuario. En AIX and Linux , este campo especifica el ID de usuario real de la aplicación. En Windows , este campo especifica el ID de usuario de la aplicación.

EffectiveUserID

Tipo: MQCHAR12 -entrada

Identificador de usuario efectivo. En AIX and Linux , este campo especifica el ID de usuario efectivo de la aplicación. En Windows , este campo está en blanco.

Entorno

Tipo: MQLONG - entrada

Medio ambiente. Este campo especifica el entorno desde el que se ha realizado la llamada. El campo es uno de los valores siguientes:

MQXE_COMMAND_SERVER

Servidor de mandatos

MQXE_MQSC

Intérprete de mandatos **runmqsc**

MQXE_MCA

Agente de canal de mensajes MQXE_OTHER

MQXE_OTHER

Entorno no definido

CallerType

Tipo: MQLONG - entrada

Tipo de interlocutor. Este campo especifica el tipo de programa que ha realizado la llamada. El campo es uno de los valores siguientes:

MQXACT_EXTERNAL

La llamada es externa al gestor de colas.

MQXACT_INTERNAL

La llamada es interna al gestor de colas.

AuthenticationType

Tipo: MQLONG - entrada

Tipo de autenticación. Este campo especifica el tipo de autenticación que se está realizando. El campo es uno de los valores siguientes:

MQZAT_CONTEXTO_INICIAL

La llamada de autenticación se debe a que se está inicializando el contexto de usuario. Este valor se utiliza durante una llamada MQCONN o MQCONNX.

CONTEXTO_CAMBIO_MQZAT_CONTEXTO

La llamada de autenticación se debe a que se ha cambiado el contexto de usuario. Este valor se utiliza cuando el MCA cambia el contexto de usuario. Tema principal: MQZAC-

BindType

Tipo: MQLONG - entrada

Tipo de enlace. Este campo especifica el tipo de enlace en uso. El campo es uno de los valores siguientes:

MQCNO_FASTPATH_BINDING

Enlace de vía de acceso rápida.

MQCNO_SHARED_BINDING

Enlace compartido.

MQCNO_ISOLATED_BINDING

Enlace aislado.

Declaración C

Declare los campos de la estructura como se indica a continuación:

```
typedef struct tagMQZAC MQZAC;
struct tagMQZAC {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQPID      ProcessId;        /* Process identifier */
    MQTID      ThreadId;         /* Thread identifier */
    MQCHAR28    ApplName;        /* Application name */
    MQCHAR12    UserID;           /* User identifier */
    MQCHAR12    EffectiveUserID;  /* Effective user identifier */
    MQLONG      Environment;      /* Environment */
    MQLONG      CallerType;       /* Caller type */
    MQLONG      AuthenticationType; /* Authentication type */
    MQLONG      BindType;        /* Bind type */
};
```

MQZAD-Datos de autorización

La estructura MQZAD se utiliza en la llamada MQZ_ENUMERATE_AUTHORITY_DATA para dos parámetros, uno de entrada y otro de salida.

Consulte “MQZ_ENUMERATE_AUTHORITY_DATA-Enumerar datos de autorización” en la página 1664 para obtener más información sobre los parámetros **Filter** y **AuthorityBuffer** :

- MQZAD se utiliza para el parámetro **Filter** que se especifica en la llamada. Este parámetro especifica los criterios de selección que deben utilizarse para seleccionar los datos de autorización devueltos por la llamada.
- MQZAD también se utiliza para el parámetro **AuthorityBuffer** que es la salida de la llamada. Este parámetro especifica las autorizaciones para una combinación de nombre de perfil, tipo de objeto y entidad.

Tabla 1. resume los campos de la estructura.

Tabla 875. Campos en MQZAD

Campo	Descripción
<u>StrucId</u>	Identificador de la estructura
<u>Versión</u>	Número de versión de la estructura
<u>ProfileName</u>	Nombre de perfil
<u>ObjectType</u>	Tipo de objeto
<u>Autorización</u>	Autorización
<u>EntityDataPtr</u>	Puntero a datos de entidad
<u>EntityType</u>	Tipo de entidad
<u>Opciones</u>	Opciones

Campos

StrucId

Tipo: MQCHAR4 -entrada

Identificador de estructura. El valor es el siguiente:

MQZAD_STRUC_ID

Identificador de la estructura de datos de autorización.

Para el lenguaje de programación C, también se define la constante MQZAD_STRUC_ID_ARRAY; tiene el mismo valor que MQZAD_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Versión

Tipo: MQLONG - entrada

Número de versión de la estructura. El valor es el siguiente:

MQZAD_VERSION_1

Estructura de contexto de aplicación Version-1 . La constante MQZAD_CURRENT_VERSION especifica el número de versión de la versión actual.

La constante siguiente especifica el número de versión de la versión actual:

MQZAD_VERSION_ACTUAL

Versión actual de la estructura de datos de autorización.

ProfileName

Tipo: MQCHAR48 -entrada

Nombre de perfil.

Para el parámetro **Filter** , este campo es el nombre de perfil para el que se necesitan datos de autorización. Si el nombre está completamente en blanco hasta el final del campo o el primer carácter nulo, se devuelven los datos de autorización para todos los nombres de perfil.

Para el parámetro **AuthorityBuffer** , este campo es el nombre de un perfil que coincide con los criterios de selección especificados.

ObjectType

Tipo: MQLONG - entrada

Tipo de objeto.

Para el parámetro **Filter** , este campo es el tipo de objeto para el que se necesitan datos de autorización. Si el valor es MQOT_ALL, se devuelven los datos de autorización para todos los tipos de objeto.

Para el parámetro **AuthorityBuffer** , este campo es el tipo de objeto al que se aplica el perfil identificado por el parámetro **ProfileName** .

El valor es uno de los siguientes; para el parámetro **Filter** , el valor MQOT_ALL también es válido:

MQOT_AUTH_INFO

Información de autenticación

MQOT_CHANNEL

Canal

MQOT_CLNTCONN_CHANNEL

Canal de conexión de cliente

MQOT_ESCUCHA

Escucha

MQOT_LISTA_DE_NOMBRES

Lista de nombres

MQOT_PROCESO

Definición de proceso

MQOT_Q

Cola

MQOT_Q_MGR

Gestor de colas

SERVICIO MQOT_SERVICE

Servicio

Autorización

Tipo: MQLONG - entrada

Autorización.

Para el parámetro **Filter** , este campo se ignora.

Para el parámetro **AuthorityBuffer** , este campo representa las autorizaciones que la entidad tiene para los objetos identificados por **ProfileName** y **ObjectType**. Si la entidad sólo tiene una autorización, el campo es igual al valor de autorización adecuado (constante MQZAO_ *). Si la entidad tiene más de una autorización, el campo es el OR a nivel de bit de las constantes MQZAO_ * correspondientes.

EntityDataPtr

Tipo: PMQZED-entrada

Dirección de la estructura MQZED que identifica una entidad.

Para el parámetro **Filter** , este campo apunta a una estructura MQZED que identifica la entidad para la que se necesitan datos de autorización. Si **EntityDataPtr** es el puntero nulo, se devuelven los datos de autorización para todas las entidades.

Para el parámetro **AuthorityBuffer** , este campo apunta a una estructura MQZED que identifica la entidad para la que se han devuelto datos de autorización.

EntityType

Tipo: MQLONG - entrada

Tipo de entidad.

Para el parámetro **Filter** , este campo especifica el tipo de entidad para el que se necesitan datos de autorización. Si el valor es MQZAET_NONE, se devuelven los datos de autorización para todos los tipos de entidad.

Para el parámetro **AuthorityBuffer** , este campo especifica el tipo de la entidad identificada por la estructura MQZED a la que apunta el parámetro **EntityDataPtr** .

El valor es uno de los siguientes; para el parámetro **Filter** , el valor MQZAET_NONE también es válido:

MQZAET_PRINCIPAL

Principal

GRUPO_MQZAC

Grupo

Opciones

Tipo: MQAUTHOPT-entrada

Opciones. Este campo especifica las opciones que dan control sobre los perfiles que se visualizan. Se debe especificar uno de los valores siguientes:

MQAUTHOPT_NAME_ALL_MATCHING

Muestra todos los perfiles

MQAUTHOPT_NOMBRE_EXPLÍCITO

Muestra perfiles que tienen exactamente el mismo nombre que el especificado en el campo **ProfileName** .

Además, también debe especificarse uno de los siguientes:

MQAUTHOPT_ENTITY_SET

Visualizar todos los perfiles que se utilizan para calcular la autorización acumulativa que la entidad tiene sobre el objeto especificado por el parámetro **ProfileName** . El parámetro **ProfileName** no debe contener ningún carácter comodín.

- Si la entidad especificada es un principal, para cada miembro del conjunto {entity, groups} se muestra el perfil más aplicable que se aplica al objeto.
- Si la entidad especificada es un grupo, se visualiza el perfil más aplicable del grupo que se aplica al objeto.
- Si se especifica este valor, los valores de **ProfileName**, **ObjectType**, **EntityType** y el nombre de entidad especificado en la estructura MQZED de **EntityDataPtr** no deben estar en blanco.

Si ha especificado MQAUTHOPT_NAME_ALL_MATCHING, también puede especificar el valor siguiente:

MQAUTHOPT_ENTIDAD_EXPLÍCITA

Muestra perfiles que tienen exactamente el mismo nombre de entidad que el nombre de entidad especificado en la estructura MQZED de **EntityDataPtr** .

Declaración C

```
typedef struct tagMQZAD MQZAD;
struct tagMQZAD {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQCHAR48   ProfileName;      /* Profile name */
    MQLONG     ObjectType;       /* Object type */
    MQLONG     Authority;        /* Authority */
    PMQZED     EntityDataPtr;    /* Address of MQZED structure identifying an
                                entity */
    MQLONG     EntityType;       /* Entity type */
    MQAUTHOPT  Options;         /* Options */
};
```

MQZED-Descriptor de entidad

La estructura MQZED se utiliza en un número de llamadas de servicio de autorización para especificar la entidad para la que se va a comprobar la autorización.

Tabla 1. resume los campos de la estructura.

Tabla 876. Campos en MQZED

Campo	Descripción
<u>StrucId</u>	Identificador de la estructura
<u>Versión</u>	Versión
<u>EntityNamePtr</u>	Nombre de entidad
<u>EntityDomainPtr</u>	Puntero de dominio de entidad
<u>SecurityId</u>	Identificador de seguridad
<u>CorrelationPtr</u>	Puntero de correlación

Campos

StrucId

Tipo: MQCHAR4 -entrada

Identificador de estructura. El valor es el siguiente:

MQZED_STRUC_ID

Identificador de la estructura del descriptor de entidad.

Para el lenguaje de programación C, también se define la constante MQZED_STRUC_ID_ARRAY; tiene el mismo valor que MQZED_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Versión

Tipo: MQLONG - entrada

Número de versión de la estructura. El valor es el siguiente:

MQZED_VERSION_1

Estructura del descriptor de entidad Version-1 .

La constante siguiente especifica el número de versión de la versión actual:

MQZED_VERSION_ACTUAL

Versión actual de la estructura del descriptor de entidad.

EntityNamePtr

Tipo: PMQCHAR-entrada

Nombre de perfil.

Dirección del nombre de entidad. Se trata de un puntero al nombre de la entidad cuya autorización se va a comprobar.

EntityDomainPtr

Tipo: PMQCHAR-entrada

Dirección del nombre de dominio de entidad. Es un puntero al nombre del dominio que contiene la definición de la entidad cuya autorización se va a comprobar.

SecurityId

Tipo: MQBYTE40 -entrada

Autorización.

Identificador de seguridad. Es el identificador de seguridad cuya autorización se debe comprobar.

CorrelationPtr

Tipo: MQPTR-entrada

Puntero de correlación. Esto facilita el paso de datos correlacionales entre la función de autenticar usuario y otras funciones de OAM adecuadas.

Declaración C

```
typedef struct tagMQZED MQZED;
struct tagMQZED {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    PMQCHAR    EntityNamePtr;    /* Address of entity name */
    PMQCHAR    EntityDomainPtr;  /* Address of entity domain name */
    MQBYTE40   SecurityId;       /* Security identifier */
    MQPTR      CorrelationPtr;   /* Address of correlation data */
}
```

MQZEP-Añadir punto de entrada de componente

Un componente de servicio inicia esta función, durante la inicialización, para añadir un punto de entrada al vector de punto de entrada para dicho componente de servicio.

Sintaxis

MQZEP (*Hconfig* , *Función* , *EntryPoint* , *CompCode* , *Razón*)

Parámetros

Hconfig

Tipo: MQHCONFIG-entrada

Descriptor de contexto de configuración. Este descriptor de contexto representa el componente que se está configurando para este servicio instalable en particular. Debe ser el mismo que el componente que el gestor de colas ha pasado a la función de configuración de componente en la llamada de inicialización de componente.

Función

Tipo: MQLONG - entrada

Identificador de función. Los valores válidos para esto se definen para cada servicio instalable.

Si se llama a MQZEP más de una vez para la misma función, la última llamada realizada proporciona el punto de entrada que se utiliza.

EntryPoint

Tipo: PMQFUNC-entrada

Punto de entrada de función. Es la dirección del punto de entrada proporcionado por el componente para realizar la función.

El valor NULL es válido e indica que este componente no proporciona la función. Se asume NULL para puntos de entrada que no están definidos utilizando MQZEP.

CompCode

Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_FUNCTION_ERROR

(2281, X'8E9') Identificador de función no válido.

MQRC_HCONFIG_ERROR

(2280, X'8E8') Mango de configuración no válido.

Para obtener más información sobre estos códigos de razón, consulte [Códigos de terminación y de razón de la API](#).

Invocación en C

```
MQZEP (Hconfig, Function, EntryPoint, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQHCONFIG  Hconfig;      /* Configuration handle */
MQLONG     Function;     /* Function identifier */
PMQFUNC    EntryPoint;   /* Function entry point */
MQLONG     CompCode;     /* Completion code */
MQLONG     Reason;       /* Reason code qualifying CompCode */
```

MQZFP-Parámetros libres

La estructura MQZFP se utiliza en la llamada MQZ_FREE_USER para el parámetro *FreeParms* . Este parámetro especifica los datos relacionados con el recurso que se va a liberar.

Tabla 1. resume los campos de la estructura.

Tabla 877. Campos en MQZFP	
Campo	Descripción
<u>StrucId</u>	Identificador de la estructura
<u>Versión</u>	Versión
<u>reservado</u>	Reservado, campo
<u>CorrelationPtr</u>	Puntero de correlación

Campos

StrucId

Tipo: MQCHAR4 -entrada

Identificador de estructura. El valor es el siguiente:

MQZIC_STRUC_ID

Identificador de la estructura de contexto de identidad. Para el lenguaje de programación C, también se define la constante MQZIC_STRUC_ID_ARRAY; tiene el mismo valor que MQZIC_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Versión

Tipo: MQLONG - entrada

Número de versión de la estructura. El valor es el siguiente:

MQZFP_VERSION_1

Estructura de parámetros libres Version-1 .

La constante siguiente especifica el número de versión de la versión actual:

MQZFP_VERSION_ACTUAL

Versión actual de la estructura de parámetros libres.

Reserved

Tipo: MQBYTE8 -entrada

Campo reservado. El valor inicial es nulo.

CorrelationPtr

Tipo: MQPTR-entrada

Puntero de correlación. Dirección de los datos de correlación relacionados con el recurso que se va a liberar.

Declaración C

```
typedef struct tagMQZFP MQZFP;  
struct tagMQZFP {  
    MQCHAR4    StrucId;          /* Structure identifier */  
    MQLONG     Version;         /* Structure version number */  
    MQBYTE8    Reserved;       /* Reserved field */  
    MQPTR      CorrelationPtr;  /* Address of correlation data */  
};
```

MQZIC - Contexto de identidad

La estructura MQZIC se utiliza en la llamada MQZ_AUTHENTICATE_USER para el *IdentityContext* parámetro.

La estructura MQZIC contiene información de contexto de identidad, que identifica al usuario de la aplicación que colocó primero el mensaje en una cola:

- El gestor de colas rellena el *UserIdentifier* campo con un nombre que identifica al usuario. La forma en que el gestor de colas puede hacerlo depende del entorno en el que se ejecute la aplicación.
- El gestor de colas rellena el *AccountingToken* campo con un token o número que ha determinado a partir de la aplicación que ha introducido el mensaje.
- Las aplicaciones pueden utilizar el *ApplIdentityData* campo para cualquier información adicional que deseen incluir sobre el usuario (por ejemplo, una contraseña cifrada).

Las aplicaciones debidamente autorizadas pueden establecer el contexto de identidad utilizando la función MQZ_AUTHENTICATE_USER.

Un identificador Windows de seguridad del sistema (SID) se almacena en el *AccountingToken* campo cuando se crea un mensaje en IBM MQ for Windows. El SID se puede utilizar para complementar el *UserIdentifier* campo y establecer las credenciales de un usuario.

Tabla 1. resume los campos de la estructura.

Tabla 878. Campos en MQZIC	
Campo	Descripción
StrucId	Identificador de la estructura
Versión	Versión
UserIdentifier	Identificador de usuario
AccountingToken	Señal de contabilidad
ApplIdentityData	Datos de identidad de la aplicación

Campos

StrucId

Tipo: MQCHAR4 - entrada

Identificador de estructura. El valor es el siguiente:

MQZIC_STRUC_ID

Identificador para la estructura del contexto de identidad. Para el lenguaje de programación C, también se define la constante MQZIC_STRUC_ID_ARRAY, que tiene el mismo valor que MQZIC_STRUC_ID, pero es una matriz de caracteres en lugar de una cadena.

Versión

Tipo: MQLONG - entrada

Número de versión de la estructura. El valor es el siguiente:

MQZIC_VERSION_1

Version-1 estructura del contexto de identidad.

La siguiente constante especifica el número de versión de la versión actual:

MQZIC_VERSION_ACTUAL

Versión actual de la estructura del contexto de identidad.

UserIdentifier

Tipo: MQCHAR12 - entrada

Identificador de usuario. Esto forma parte del contexto de identidad del mensaje. *UserIdentifier*

Especifica el identificador de usuario de la aplicación que originó el mensaje. El gestor de colas trata esta información como datos de caracteres, pero no define su formato. Para obtener más información sobre el *UserIdentifier* campo, consulte [“UserIdentifier \(MQCHAR12\)” en la página 429.](#)

AccountingToken

Tipo: MQBYTE32 - entrada

Ficha contable. Esto forma parte del contexto de identidad del mensaje. *AccountingToken* permite que una aplicación haga que el trabajo realizado como resultado del mensaje se cobre adecuadamente. El gestor de colas trata esta información como una cadena de bits y no comprueba su contenido. Para obtener más información sobre el *AccountingToken* campo, consulte [“AccountingToken \(MQBYTE32\)” en la página 430.](#)

ApplIdentityData

Tipo: MQCHAR32 - entrada

Datos de la aplicación relacionados con la identidad. Esto forma parte del contexto de identidad del mensaje. *ApplIdentityData* Es información definida por el conjunto de aplicaciones que se puede utilizar para proporcionar información adicional sobre el origen del mensaje. Por ejemplo, podrían establecerlo las aplicaciones que se ejecutan con la autoridad de usuario adecuada para indicar si los datos de identidad son fiables. Para obtener más información sobre el campo « *ApplIdentityData* », consulte [“ApplIdentityData \(MQCHAR32\)” en la página 432.](#)

Declaración C

```
typedef struct tagMQZED MQZED;
struct tagMQZED {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQCHAR12   UserIdentifier;    /* User identifier */
    MQBYTE32   AccountingToken;  /* Accounting token */
    MQCHAR32   ApplIdentityData; /* Application data relating to identity */
};
```

Información de consulta sobre la interfaz de servicios instalables en IBM i

Utilice esta información para comprender la información de referencia sobre los servicios instalables para IBM i.

Para cada función hay una descripción, incluyendo el identificador de la función (para MQZEP).

Los *parámetros* se muestran en el orden en que deben aparecer. Todos deben estar presentes.

Cada nombre de parámetro va seguido de su tipo de datos entre paréntesis. Estos son los tipos de datos elementales descritos en [“Tipos de datos elementales en IBM i”](#) en la página 1016.

La invocación en lenguaje C también se proporciona, después de la descripción de los parámetros.

Conceptos relacionados

 [Servicios y componentes instalables para IBM i](#)

 [Servicios y componentes instalables para UNIX, Linux y Windows](#)

Referencia relacionada

[“Información de consulta sobre la interfaz de servicios instalables”](#) en la página 1646
Esta colección de temas proporciona información de referencia para los servicios instalables.

MQZEP (Añadir punto de entrada de componente) en IBM i

Esta función la invoca un componente de servicio, durante la inicialización, para añadir un punto de entrada al vector de punto de entrada para dicho componente de servicio.

Sintaxis

```
MQZEP (Hconfig, Function, EntryPoint, CompCode, Reason)
```

Parámetros

La llamada MQZEP tiene los parámetros siguientes.

Hconfig (MQHCONFIG)-entrada

Descriptor de contexto de configuración.

Este descriptor de contexto representa el componente que se está configurando para este servicio instalable en particular. Debe ser el mismo que el que el gestor de colas ha pasado a la función de configuración de componente en la llamada de inicialización de componente.

Función (MQLONG)-entrada

Identificador de función.

Los valores válidos para esto se definen para cada servicio instalable. Si se llama a MQZEP más de una vez para la misma función, la última llamada realizada proporciona el punto de entrada que se utiliza.

EntryPoint (PMQFUNC)-entrada

Punto de entrada de función.

Es la dirección del punto de entrada proporcionado por el componente para realizar la función. El valor NULL es válido e indica que este componente no proporciona la función. Se asume NULL para puntos de entrada que no están definidos utilizando MQZEP.

CompCode (MQLONG)-salida

Código de terminación.

Es uno de los siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón (MQLONG)-salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_FUNCTION_ERROR

(2281, X'8E9') Identificador de función no válido.

MQRC_HCONFIG_ERROR

(2280, X'8E8') Mango de configuración no válido.

Para obtener más información sobre estos códigos de razón, consulte [Mensajes y códigos de razón](#).

Invocación en C

```
MQZEP (Hconfig, Function, EntryPoint, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQHCONFIG  Hconfig;      /* Configuration handle */
MQLONG     Function;     /* Function identifier */
PMQFUNC    EntryPoint;   /* Function entry point */
MQLONG     CompCode;     /* Completion code */
MQLONG     Reason;       /* Reason code qualifying CompCode */
```

MQHCONFIG (descriptor de contexto de configuración) en IBM i

El tipo de datos MQHCONFIG representa un descriptor de contexto de configuración, es decir, el componente que se está configurando para un servicio instalable determinado. Un descriptor de contexto de configuración debe estar alineado en su límite natural.

Las aplicaciones deben probar las variables de este tipo sólo para la igualdad.

Declaración C

```
typedef void MQPOINTER MQHCONFIG;
```

PMQFUNC (Puntero para función) en IBM i

Puntero a una función.

Declaración C

```
typedef void MQPOINTER PMQFUNC;
```

MQZ_AUTHENTICATE_USER (Autenticar usuario) en IBM i

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_5 . Lo invoca el gestor de colas para autenticar un usuario o para establecer campos de contexto de identidad.

Se invoca cuando se establece un contexto de aplicación de usuario IBM MQ . Esto sucede durante las llamadas de conexión en el punto en el que se inicializa el contexto de usuario de la aplicación y en cada punto en el que se cambia el contexto de usuario de la aplicación. Cada vez que se realiza una llamada de conexión, la información de contexto de usuario de la aplicación se vuelve a adquirir en el campo *IdentityContext* .

El identificador de función para esta función (para MQZEP) es MQZID_AUTHENTICATE_USER.

Sintaxis

MQZ_AUTHENTICATE_USER (QMgrName, SecurityParms, ApplicationContext, IdentityContext, CorrelationPtr, ComponentData, Continuation, CompCode, Reason)

Parámetros

La llamada MQZ_AUTHENTICATE_USER tiene los parámetros siguientes.

QMgrName (MQCHAR48)-entrada

Nombre del gestor de colas.

El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo. El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

SecurityParms (MQCSP)-entrada

Parámetros de seguridad.

Datos relacionados con el ID de usuario, la contraseña y el tipo de autenticación.

Durante una llamada de tipo « MQI » a MQCONN, este parámetro contiene valores nulos o valores predeterminados.

ApplicationContext (MQZAC)-entrada

Contexto de aplicación.

Datos relacionados con la aplicación de llamada. Consulte [“MQZAC \(contexto de aplicación\) en IBM i”](#) en la página 1742 para obtener los detalles. Cada vez que se realiza una llamada a `MQI` en MQCONN o MQCONN, se vuelve a obtener la información de contexto del usuario contenida en la estructura MQZAC.

IdentityContext (MQZIC)-entrada/salida

Contexto de identidad.

En la entrada de la función de autenticar usuario, identifica el contexto de identidad actual. La función de autenticar usuario puede cambiar esto, momento en el que el gestor de colas adopta el nuevo contexto de identidad. Consulte [“MQZIC \(Contexto de identidad\) activado IBM i”](#) en la página 1749 para obtener más detalles sobre la estructura MQZIC.

CorrelationPtr (MQPTR)-salida

Puntero de correlación.

Especifica la dirección de cualquier dato de correlación. A continuación, este puntero se pasa a otras llamadas de OAM.

ComponentData (MQBYTE x ComponentDataLength) - entrada/salida

Datos de componente.

El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios que realice en él cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componentes. La longitud de esta área de datos la pasa el gestor de colas en el parámetro **ComponentDataLength** de la llamada MQZ_INIT_AUTHORITY.

Continuación (MQLONG)-salida

Distintivo de continuación.

Se pueden especificar los siguientes valores:

MQZCI_DEFAULT

Continuación dependiente de otros componentes.

MQZCI_STOP

No continúe con el componente siguiente.

CompCode (MQLONG)-salida

Código de terminación.

Es uno de los siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón (MQLONG)-salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_SERVICE_ERROR

(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

Para obtener más información sobre estos códigos de razón, consulte [Mensajes y códigos de razón](#).

Invocación en C

```
MQZ_AUTHENTICATE_USER (QMgrName, SecurityParms, ApplicationContext,
                        IdentityContext, &CorrelationPtr, ComponentData,
                        &Continuation, &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQCHAR48  QMgrName;           /* Queue manager name */
MQCSP      SecurityParms;      /* Security parameters */
MQZAC      ApplicationContext; /* Application context */
MQZIC      IdentityContext;    /* Identity context */
MQPTR      CorrelationPtr;     /* Correlation pointer */
MQBYTE     ComponentData[n];   /* Component data */
MQLONG     Continuation;       /* Continuation indicator set by
                                component */
MQLONG     CompCode;           /* Completion code */
MQLONG     Reason;             /* Reason code qualifying CompCode */
```

IBM i MQZ_CHECK_AUTHORITY (Comprobar autorización) en IBM i

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_1 y la invoca el gestor de colas para comprobar si una entidad tiene autorización para realizar una acción determinada, o acciones, en un objeto especificado.

El identificador de función para esta función (para MQZEP) es MQZID_CHECK_AUTHORITY.

Sintaxis

MQZ_CHECK_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName, ObjectType, Authority, ComponentData, Continuation, CompCode, Reason)

Parámetros

La llamada MQZ_CHECK_AUTHORITY tiene los parámetros siguientes.

QMgrName (MQCHAR48)-entrada

Nombre del gestor de colas.

El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo. El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

EntityName (MQCHAR12)-entrada

Nombre de entidad.

El nombre de la entidad cuya autorización para el objeto se va a comprobar. La longitud máxima de la serie es de 12 caracteres; si es más corta que esa, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

No es esencial que esta entidad sea conocida por el servicio de seguridad subyacente. Si no se conoce, las autorizaciones del grupo **nobody** especial (al que se supone que pertenecen todas las entidades) se utilizan para la comprobación. Un nombre totalmente en blanco es válido y se puede utilizar de esta forma.

EntityType (MQLONG)-entrada

Tipo de entidad.

El tipo de entidad especificado por *EntityName*. Es uno de los siguientes:

MQZAET_PRINCIPAL

Director.

GRUPO_MQZAC

group.

ObjectName (MQCHAR48)-entrada

El nombre del objeto.

El nombre del objeto al que se necesita acceso. La longitud máxima de la serie es de 48 caracteres; si es más corta, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

Si *ObjectType* es MQOT_Q_MGR, este nombre es el mismo que *QMgrName*.

ObjectType (MQLONG)-entrada

Tipo de objeto.

El tipo de entidad especificado por *ObjectName*. Es uno de los siguientes:

MQOT_AUTH_INFO

Información de autenticación.

MQOT_CHANNEL

Canal.

MQOT_CLNTCONN_CHANNEL

Canal de conexión de cliente.

MQOT_ESCUCHA

Escucha.

MQOT_LISTA_DE_NOMBRES

Lista de nombres.

MQOT_PROCESO

.

MQOT_Q

Cola.

MQOT_Q_MGR

Gestor de colas.

SERVICIO MQOT_SERVICE

.

Autorización (MQLONG)-entrada

Autorización que se debe comprobar.

Si se está comprobando una autorización, este campo es igual a la operación de autorización adecuada (constante MQZAO_*). Si se está comprobando más de una autorización, es el OR a nivel de bit de las constantes MQZAO_* correspondientes.

Las siguientes autorizaciones se aplican al uso de las llamadas de MQI :

MQZAO_CONNECT

Posibilidad de utilizar la llamada MQCONN.

MQZAO_BROWSE

Posibilidad de utilizar la llamada MQGET con una opción de examinar.

Esto permite especificar la opción MQGMO_BROWSE_FIRST, MQGMO_BROWSE_MSG_UNDER_CURSOR o MQGMO_BROWSE_NEXT en la llamada MQGET.

MQZAO_INPUT

Posibilidad de utilizar la llamada MQGET con una opción de entrada.

Esto permite especificar la opción MQOO_INPUT_SHARED, MQOO_INPUT_EXCLUSIVE o MQOO_INPUT_AS_Q_DEF en la llamada MQOPEN.

MQZAO_OUTPUT

Posibilidad de utilizar la llamada MQPUT.

Esto permite especificar la opción MQOO_OUTPUT en la llamada MQOPEN.

MQZAO_INQUIRE

Posibilidad de utilizar la llamada MQINQ.

Esto permite especificar la opción MQOO_INQUIRE en la llamada MQOPEN.

MQZAO_SET

Posibilidad de utilizar la llamada MQSET.

Esto permite especificar la opción MQOO_SET en la llamada MQOPEN.

MQZAO_PASS_IDENTITY_CONTEXT

Capacidad de pasar contexto de identidad.

Esto permite especificar la opción MQOO_PASS_IDENTITY_CONTEXT en la llamada MQOPEN y especificar la opción MQPMO_PASS_IDENTITY_CONTEXT en las llamadas MQPUT y MQPUT1 .

MQZAO_PASS_ALL_CONTEXT

Capacidad de pasar todo el contexto.

Esto permite especificar la opción MQOO_PASS_ALL_CONTEXT en la llamada MQOPEN y especificar la opción MQPMO_PASS_ALL_CONTEXT en las llamadas MQPUT y MQPUT1 .

MQZAO_SET_IDENTITY_CONTEXT

Capacidad para establecer el contexto de identidad.

Esto permite especificar la opción MQOO_SET_IDENTITY_CONTEXT en la llamada MQOPEN y especificar la opción MQPMO_SET_IDENTITY_CONTEXT en las llamadas MQPUT y MQPUT1 .

MQZAO_SET_ALL_CONTEXT

Capacidad para establecer todo el contexto.

Esto permite especificar la opción MQOO_SET_ALL_CONTEXT en la llamada MQOPEN y especificar la opción MQPMO_SET_ALL_CONTEXT en las llamadas MQPUT y MQPUT1 .

MQZAO_ALTERNATE_USER_AUTHORITY

Capacidad para utilizar la autorización de usuario alternativo.

Esto permite especificar la opción MQOO_ALTERNATE_USER_AUTHORITY en la llamada MQOPEN y especificar la opción MQPMO_ALTERNATE_USER_AUTHORITY en la llamada MQPUT1 .

MQZAO_ALL_MQI

Todas las autorizaciones de MQI.

Esto habilita todas las autorizaciones descritas anteriormente.

Las autorizaciones siguientes se aplican a la administración de un gestor de colas:

MQZAO_CREAR

Posibilidad de crear objetos de un tipo especificado.

MQZAO_DELETE

Posibilidad de suprimir un objeto especificado.

MQZAO_DISPLAY

Capacidad para visualizar los atributos de un objeto especificado.

MQZAO_CHANGE

Posibilidad de cambiar los atributos de un objeto especificado.

MQZAO_CLEAR

Posibilidad de suprimir todos los mensajes de una cola especificada.

MQZAO_AUTORIZAR

Posibilidad de autorizar a otros usuarios para un objeto especificado.

MQZAO_CONTROL

Capacidad para iniciar, detener o hacer ping a un objeto de canal no cliente.

MQZAO_CONTROL_EXTENDED

Capacidad para restablecer un número de secuencia o resolver un mensaje dudoso en un objeto de canal no de cliente.

MQZAO_ALL_ADMIN

Todas las autorizaciones de administración, excepto MQZAO_CREATE.

Las siguientes autorizaciones se aplican tanto al uso del servidor de bases de datos de Oracle (MQI) como a la administración de un gestor de colas:

MQZAO_TODOS

Todas las autorizaciones, excepto MQZAO_CREATE.

MQZAO_NINGUNA

Sin autorizaciones.

ComponentData (MQBYTE x ComponentDataLength) - entrada/salida

Datos de componente.

El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de las funciones de este componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro **ComponentDataLength** de la llamada MQZ_INIT_AUTHORITY.

Continuación (MQLONG)-salida

Indicador de continuación establecido por componente.

Se pueden especificar los siguientes valores:

MQZCI_DEFAULT

Continuación dependiente del gestor de colas.

Para MQZ_CHECK_AUTHORITY tiene el mismo efecto que MQZCI_STOP.

MQZCI_CONTINUAR

Continúe con el componente siguiente.

MQZCI_STOP

No continúe con el componente siguiente.

CompCode (MQLONG)-salida

Código de terminación.

Es uno de los siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón (MQLONG)-salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_NOT_AUTHORIZED

(2035, X'7F3') Acceso no autorizado.

MQRC_SERVICE_ERROR

(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICIO_NO_DISPONIBLE

(2285, X'8ED') Servicio subyacente no disponible.

Para obtener más información sobre estos códigos de razón, consulte [Mensajes y códigos de razón](#).

Invocación en C

```
MQZ_CHECK_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName,  
                     ObjectType, Authority, ComponentData,  
                     &Continuation, &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQCHAR12  EntityName;         /* Entity name */  
MQLONG    EntityType;         /* Entity type */  
MQCHAR48  ObjectName;         /* Object name */  
MQLONG    ObjectType;         /* Object type */  
MQLONG    Authority;          /* Authority to be checked */  
MQBYTE    ComponentData[n];   /* Component data */  
MQLONG    Continuation;       /* Continuation indicator set by  
                               component */  
MQLONG    CompCode;           /* Completion code */  
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

MQZ_CHECK_PRIVILEGED-Comprobar si el usuario tiene privilegios

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_6 y la invoca el gestor de colas para determinar si un usuario especificado es un usuario privilegiado.

El identificador de función para esta función (para MQZEP) es MQZID_CHECK_PRIVILEGED.

Sintaxis

`MQZ_CHECK_PRIVILEGED(QMgrName , EntityData , EntityType , ComponentData ,
Continuation , CompCode , Reason)`

Parámetros

QMgrName

Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

EntityData

Tipo: MQZED-entrada

Datos de entidad. Datos relacionados con la entidad que se va a comprobar. Para obtener más información, consulte [“MQZED-Descriptor de entidad” en la página 1705.](#)

EntityType

Tipo: MQLONG - entrada

Tipo de entidad. El tipo de entidad especificado por EntityData. Tiene que ser uno de los valores siguientes:

MQZAET_PRINCIPAL

Director.

GRUPO_MQZAC

group.

ComponentData

Tipo: MQBYTEComponentDataLength - entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro **ComponentDataLength** de la llamada MQZ_INIT_AUTHORITY.

continuación

Tipo: MQLONG - salida

Indicador de continuación establecido por componente. Se pueden especificar los siguientes valores:

MQZCI_DEFAULT

Continuación dependiente del gestor de colas.

Para MQZ_CHECK_AUTHORITY, tiene el mismo efecto que MQZCI_STOP.

MQZCI_CONTINUAR

Continúe con el componente siguiente.

MQZCI_STOP

No continúe con el componente siguiente.

Si la llamada a un componente falla (es decir, *CompCode* devuelve MQCC_FAILED), y el parámetro *Continuación* es MQZCI_DEFAULT o MQZCI_CONTINUE, el gestor de colas continúa llamando a otros componentes si los hay.

Si la llamada es satisfactoria (es decir, *CompCode* devuelve MQCC_OK) no se llama a ningún otro componente independientemente del valor de *Continuación*.

Si la llamada falla y el parámetro *Continuación* es MQZCI_STOP, no se llama a ningún otro componente y se devuelve el error al gestor de colas. Los componentes no conocen las llamadas anteriores, por lo que el parámetro *Continuación* siempre se establece en MQZCI_DEFAULT antes de la llamada.

CompCode

Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_NO_PRIVILEGIADO

(2584, X'A18') Este usuario no es un ID de usuario privilegiado.

MQRC_ENTIDAD_DESCONOCIDA

(2292, X'8F4') Entidad desconocida para el servicio.

MQRC_SERVICE_ERROR

(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICIO_NO_DISPONIBLE

(2285, X'8ED') Servicio subyacente no disponible.

Para obtener más información sobre estos códigos de razón, consulte [Códigos de terminación y de razón de la API](#).

Invocación en C

```
MQZ_CHECK_PRIVILEGED (QMgrName, &EntityData, EntityType,  
                     ComponentData, &Continuation,  
                     &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQZED     EntityData;         /* Entity name */  
MQLONG    EntityType;         /* Entity type */  
MQBYTE    ComponentData[n];   /* Component data */  
MQLONG    Continuation;       /* Continuation indicator set by  
                               component */  
MQLONG    CompCode;           /* Completion code */  
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

MQZ_COPY_ALL_AUTHORITY (Copiar todas las autorizaciones) en IBM i

Esta función la proporciona un componente de servicio de autorización. Lo invoca el gestor de colas para copiar todas las autorizaciones que están actualmente en vigor para un objeto de referencia en otro objeto.

El identificador de función para esta función (para MQZEP) es MQZID_COPY_ALL_AUTHORITY.

Sintaxis

MQZ_COPY_ALL_AUTHORITY (*QMgrName*, *RefObjectName*, *ObjectName*,
ObjectType, *ComponentData*, *Continuation*, *CompCode*, *Reason*)

Parámetros

La llamada MQZ_COPY_ALL_AUTHORITY tiene los parámetros siguientes.

QMgrName (MQCHAR48)-entrada

Nombre del gestor de colas.

El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

RefObjectNameMQCHAR48) - entrada

Nombre de objeto de referencia.

El nombre del objeto de referencia, cuyas autorizaciones se van a copiar. La longitud máxima de la serie es de 48 caracteres; si es más corta, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

ObjectName (MQCHAR48)-entrada

El nombre del objeto.

El nombre del objeto para el que se van a establecer los accesos. La longitud máxima de la serie es de 48 caracteres; si es más corta, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

ObjectType (MQLONG)-entrada

Tipo de objeto.

El tipo de objeto especificado por *RefObjectName* y *ObjectName*. Es uno de los siguientes:

MQOT_AUTH_INFO

Información de autenticación.

MQOT_CHANNEL

Canal.

MQOT_CLNTCONN_CHANNEL

Canal de conexión de cliente.

MQOT_ESCUCHA

Escucha.

MQOT_LISTA_DE_NOMBRES

Lista de nombres.

MQOT_PROCESO

.

MQOT_Q

Cola.

MQOT_Q_MGR

Gestor de colas.

SERVICIO MQOT_SERVICE

.

ComponentData (MQBYTE x ComponentDataLength) - entrada/salida

Datos de componente.

El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de las funciones de este componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro **ComponentDataLength** de la llamada MQZ_INIT_AUTHORITY.

Continuación (MQLONG)-salida

Indicador de continuación establecido por componente.

Se pueden especificar los siguientes valores:

MQZCI_DEFAULT

Continuación dependiente del gestor de colas.

Para MQZ_COPY_ALL_AUTHORITY tiene el mismo efecto que MQZCI_STOP.

MQZCI_CONTINUAR

Continúe con el componente siguiente.

MQZCI_STOP

No continúe con el componente siguiente.

CompCode (MQLONG)-salida

Código de terminación.

Es uno de los siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón (MQLONG)-salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_SERVICE_ERROR

(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICIO_NO_DISPONIBLE

(2285, X'8ED') Servicio subyacente no disponible.

MQRC_OBJETO_DE_REFERENCIA_DESCONOCIDO

(2294, X'8F6') Objeto de referencia desconocido.

Para obtener más información sobre estos códigos de razón, consulte [Mensajes y códigos de razón](#).

Invocación en C

```
MQZ_COPY_ALL_AUTHORITY (QMgrName, RefObjectName, ObjectName, ObjectType,  
                        ComponentData, &Continuation, &CompCode,  
                        &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQCHAR48  RefObjectName;      /* Reference object name */  
MQCHAR48  ObjectName;         /* Object name */  
MQLONG    ObjectType;         /* Object type */  
MQBYTE    ComponentData[n];   /* Component data */  
MQLONG    Continuation;       /* Continuation indicator set by  
                               component */  
MQLONG    CompCode;           /* Completion code */  
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

MQZ_DELETE_AUTHORITY (Suprimir autorización) en IBM i

Esta función la proporciona un componente de servicio de autorización y la invoca el gestor de colas para suprimir todas las autorizaciones asociadas con el objeto especificado.

El identificador de función para esta función (para MQZEP) es MQZID_DELETE_AUTHORITY.

Sintaxis

MQZ_DELETE_AUTHORITY (*QMgrName*, *ObjectName*, *ObjectType*,
ComponentData, *Continuation*, *CompCode*, *Reason*)

Parámetros

La llamada MQZ_DELETE_AUTHORITY tiene los parámetros siguientes.

QMgrName (MQCHAR48)-entrada

Nombre del gestor de colas.

El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

ObjectName (MQCHAR48)-entrada

El nombre del objeto.

El nombre del objeto para el que se van a suprimir los accesos. La longitud máxima de la serie es de 48 caracteres; si es más corta, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

Si *ObjectType* es MQOT_Q_MGR, este nombre es el mismo que *QMgrName*.

ObjectType (MQLONG)-entrada

Tipo de objeto.

El tipo de entidad especificado por *ObjectName*. Es uno de los siguientes:

MQOT_AUTH_INFO

Información de autenticación.

MQOT_CHANNEL

Canal.

MQOT_CLNTCONN_CHANNEL

Canal de conexión de cliente.

MQOT_ESCUCHA

Escucha.

MQOT_LISTA_DE_NOMBRES

Lista de nombres.

MQOT_PROCESO

.

MQOT_Q

Cola.

MQOT_Q_MGR

Gestor de colas.

SERVICIO MQOT_SERVICE

.

ComponentData (MQBYTE x ComponentDataLength) - entrada/salida

Datos de componente.

El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de las funciones de este componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro **ComponentDataLength** de la llamada MQZ_INIT_AUTHORITY.

Continuación (MQLONG)-salida

Indicador de continuación establecido por componente.

Se pueden especificar los siguientes valores:

MQZCI_DEFAULT

Continuación dependiente del gestor de colas.

Para MQZ_DELETE_AUTHORITY tiene el mismo efecto que MQZCI_STOP.

MQZCI_CONTINUAR

Continúe con el componente siguiente.

MQZCI_STOP

No continúe con el componente siguiente.

CompCode (MQLONG)-salida

Código de terminación.

Es uno de los siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón (MQLONG)-salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_SERVICE_ERROR

(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICIO_NO_DISPONIBLE

(2285, X'8ED') Servicio subyacente no disponible.

Para obtener más información sobre estos códigos de razón, consulte [Mensajes y códigos de razón](#).

Invocación en C

```
MQZ_DELETE_AUTHORITY (QMgrName, ObjectName, ObjectType, ComponentData,
&Continuation, &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQCHAR48 QMgrName;           /* Queue manager name */
MQCHAR48 ObjectName;         /* Object name */
MQLONG ObjectType;           /* Object type */
MQBYTE ComponentData[n];     /* Component data */
MQLONG Continuation;         /* Continuation indicator set by
                             component */
MQLONG CompCode;             /* Completion code */
MQLONG Reason;               /* Reason code qualifying CompCode */
```

MQZ_ENUMERATE_AUTHORITY_DATA (Enumerar datos de autorización) en IBM i

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_4 y la invoca repetidamente el gestor de colas para recuperar todos los datos de autorización que coinciden con los criterios de selección especificados en la primera invocación.

El identificador de función para esta función (para MQZEP) es MQZID_ENUMERATE_AUTHORITY_DATA.

Sintaxis

```
MQZ_ENUMERATE_AUTHORITY_DATA (QMgrName, StartEnumeration,  
Filter, AuthorityBufferLength, AuthorityBuffer, AuthorityDataLength,  
ComponentData, Continuation, CompCode, Reason)
```


Parámetros

La llamada MQZ_ENUMERATE_AUTHORITY_DATA tiene los parámetros siguientes.

QMgrName (MQCHAR48)-entrada

Nombre del gestor de colas.

El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

StartEnumeration (MQLONG)-entrada

Distintivo que indica si la llamada debe iniciar la enumeración.

Esto indica si la llamada debe iniciar la enumeración de datos de autorización o continuar la enumeración de datos de autorización iniciada por una llamada anterior a MQZ_ENUMERATE_AUTHORITY_DATA. El valor puede ser uno de los siguientes:

MQZSE_START

Iniciar enumeración.

La llamada se invoca con este valor para iniciar la enumeración de datos de autorización. El parámetro **Filter** especifica los criterios de selección que se deben utilizar para seleccionar los datos de autorización devueltos por esta y por llamadas sucesivas.

MQZSE_CONTINUAR

Continuar enumeración.

La llamada se invoca con este valor para continuar la enumeración de datos de autorización. El parámetro **Filter** se ignora en este caso y se puede especificar como puntero nulo (los criterios de selección se determinan mediante el parámetro **Filter** especificado por la llamada que tenía *StartEnumeration* establecido en MQZSE_START).

Filtro (MQZAD)-entrada

Filtro.

Si *StartEnumeration* es MQZSE_START, *Filter* especifica los criterios de selección que se van a utilizar para seleccionar los datos de autorización que se van a devolver. Si *Filter* es el puntero nulo, no se utiliza ningún criterio de selección, es decir, se devuelven todos los datos de autorización. Consulte “MQZAD (datos de autorización) en IBM i” en la [página 1744](#) para obtener detalles de los criterios de selección que se pueden utilizar.

Si *StartEnumeration* es MQZSE_CONTINUE, *Filter* se ignora y se puede especificar como puntero nulo.

AuthorityBufferLength (MQLONG) - entrada

Longitud de *AuthorityBuffer*.

Es la longitud en bytes del parámetro **AuthorityBuffer**. El almacenamiento intermedio de autorización debe ser lo suficientemente grande para acomodar los datos que se van a devolver.

AuthorityBuffer (MQZAD)-salida

Datos de autorización.

Es el almacenamiento intermedio en el que se devuelven los datos de autorización. El almacenamiento intermedio debe ser lo suficientemente grande para acomodar una estructura MQZAD, una estructura MQZED, más el nombre de entidad más largo y el nombre de dominio más largo definido.

Nota: Este parámetro se define como MQZAD, ya que MQZAD siempre se produce al principio del almacenamiento intermedio. Sin embargo, si el almacenamiento intermedio se declara realmente como MQZAD, el almacenamiento intermedio será demasiado pequeño; debe ser más grande que un MQZAD para que pueda acomodar los nombres de entidad y dominio MQZAD, MQZED.

AuthorityDataLength (MQLONG) - salida

Longitud de los datos devueltos en *AuthorityBuffer*.

Es la longitud de los datos devueltos en *AuthorityBuffer*. Si el almacenamiento intermedio de autorización es demasiado pequeño, *AuthorityDataLength* se establece en la longitud del almacenamiento intermedio necesaria y la llamada devuelve el código de terminación MQCC_FAILED y el código de razón MQRC_BUFFER_LENGTH_ERROR.

ComponentData (MQBYTE x ComponentDataLength) - entrada/salida

Datos de componente.

El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de las funciones de este componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro **ComponentDataLength** de la llamada MQZ_INIT_AUTHORITY.

Continuación (MQLONG)-salida

Indicador de continuación establecido por componente.

Se pueden especificar los siguientes valores:

MQZCI_DEFAULT

Continuación dependiente del gestor de colas.

Para MQZ_ENUMERATE_AUTHORITY_DATA, esto tiene el mismo efecto que MQZCI_CONTINUE.

MQZCI_CONTINUAR

Continúe con el componente siguiente.

MQZCI_STOP

No continúe con el componente siguiente.

CompCode (MQLONG)-salida

Código de terminación.

Es uno de los siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón (MQLONG)-salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_BUFFER_LENGTH_ERROR

(2005, X'7D5') El parámetro de longitud de almacenamiento intermedio no es válido.

MQRC_NO_HAY_DATOS_DISPONIBLES

(2379, X'94B') No hay datos disponibles.

MQRC_SERVICE_ERROR

(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

Para obtener más información sobre estos códigos de razón, consulte [Mensajes y códigos de razón](#).

Invocación en C

```
MQZ_ENUMERATE_AUTHORITY_DATA (QMgrName, StartEnumeration, &Filter,  
                               AuthorityBufferLength,
```

```
&AuthorityBuffer,
&AuthorityDataLength, ComponentData,
&Continuation, &CompCode,
&Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQCHAR48  QMgrName;           /* Queue manager name */
MQLONG    StartEnumeration;    /* Flag indicating whether call should
                                start enumeration */

MQZAD     Filter;              /* Filter */
MQLONG    AuthorityBufferLength; /* Length of AuthorityBuffer */
MQZAD     AuthorityBuffer;     /* Authority data */
MQLONG    AuthorityDataLength; /* Length of data returned in
                                AuthorityBuffer */

MQBYTE    ComponentData[n];    /* Component data */
MQLONG    Continuation;        /* Continuation indicator set by
                                component */

MQLONG    CompCode;            /* Completion code */
MQLONG    Reason;              /* Reason code qualifying CompCode */
```

MQZ_FREE_USER-Usuario libre

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_5 y la invoca el gestor de colas para liberar el recurso asignado asociado. Se invoca cuando una aplicación ha terminado de ejecutarse en todos los contextos de usuario, por ejemplo, durante una llamada a MQDISC MQI.

El identificador de función para esta función (para MQZEP) es MQZID_FREE_USER.

MQZ_GET_AUTHORITY (Obtener autorización) en IBM i

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_1 y la invoca el gestor de colas para recuperar la autorización que una entidad tiene para acceder al objeto especificado.

El identificador de función para esta función (para MQZEP) es MQZID_GET_AUTHORITY.

Sintaxis

MQZ_GET_AUTHORITY (*QMgrName, EntityName, EntityType, ObjectName, ObjectType, Authority, ComponentData, Continuation, CompCode, Reason*)

Parámetros

La llamada MQZ_GET_AUTHORITY tiene los parámetros siguientes.

QMgrName (MQCHAR48)-entrada

Nombre del gestor de colas.

El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

EntityName (MQCHAR12)-entrada

Nombre de entidad.

El nombre de la entidad cuyo acceso al objeto se va a recuperar. La longitud máxima de la serie es de 12 caracteres; si es más corta que esa, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

EntityType (MQLONG)-entrada

Tipo de entidad.

El tipo de entidad especificado por *EntityName*. Se puede especificar el valor siguiente:

MQZAET_PRINCIPAL
Director.

GRUPO_MQZAC

group.

ObjectName (MQCHAR48)-entrada

El nombre del objeto.

El nombre del objeto para el que se va a recuperar la autorización de la entidad. La longitud máxima de la serie es de 48 caracteres; si es más corta, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

Si *ObjectType* es MQOT_Q_MGR, este nombre es el mismo que *QMgrName*.

ObjectType (MQLONG)-entrada

Tipo de objeto.

El tipo de entidad especificado por *ObjectName*. Es uno de los siguientes:

MQOT_AUTH_INFO

Información de autenticación.

MQOT_CHANNEL

Canal.

MQOT_CLNTCONN_CHANNEL

Canal de conexión de cliente.

MQOT_ESCUCHA

Escucha.

MQOT_LISTA_DE_NOMBRES

Lista de nombres.

MQOT_PROCESO

.

MQOT_Q

Cola.

MQOT_Q_MGR

Gestor de colas.

SERVICIO MQOT_SERVICE

.

Autorización (MQLONG)-salida

Autoridad de entidad.

Si la entidad tiene una autorización, este campo es igual a la operación de autorización adecuada (constante MQZAO_*). Si tiene más de una autorización, este campo es el OR a nivel de bit de las constantes MQZAO_* correspondientes.

ComponentData (MQBYTE x ComponentDataLength) - entrada/salida

Datos de componente.

El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de las funciones de este componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro **ComponentDataLength** de la llamada MQZ_INIT_AUTHORITY.

Continuación (MQLONG)-salida

Indicador de continuación establecido por componente.

Se pueden especificar los siguientes valores:

MQZCI_DEFAULT

Continuación dependiente del gestor de colas.

Para MQZ_GET_AUTHORITY tiene el mismo efecto que MQZCI_CONTINUE.

MQZCI_CONTINUAR

Continúe con el componente siguiente.

MQZCI_STOP

No continúe con el componente siguiente.

CompCode (MQLONG)-salida

Código de terminación.

Es uno de los siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón (MQLONG)-salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_NOT_AUTHORIZED

(2035, X'7F3') Acceso no autorizado.

MQRC_SERVICE_ERROR

(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICIO_NO_DISPONIBLE

(2285, X'8ED') Servicio subyacente no disponible.

MQRC_ENTIDAD_DESCONOCIDA

(2292, X'8F4') Entidad desconocida para el servicio.

Para obtener más información sobre estos códigos de razón, consulte [Mensajes y códigos de razón](#).

Invocación en C

```
MQZ_GET_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName,  
                   ObjectType, &Authority, ComponentData,  
                   &Continuation, &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQCHAR12  EntityName;         /* Entity name */  
MQLONG    EntityType;         /* Entity type */  
MQCHAR48  ObjectName;         /* Object name */  
MQLONG    ObjectType;         /* Object type */  
MQLONG    Authority;          /* Authority of entity */  
MQBYTE    ComponentData[n];   /* Component data */  
MQLONG    Continuation;       /* Continuation indicator set by  
                               component */  
MQLONG    CompCode;           /* Completion code */  
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

MQZ_GET_EXPLICIT_AUTHORITY (Obtener autorización explícita) en IBM i

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_1 y la invoca el gestor de colas para recuperar la autorización que un grupo con nombre tiene para acceder a un objeto especificado (pero sin la autorización adicional del grupo **nobody**), o la autorización que el grupo primario del principal con nombre tiene para acceder a un objeto especificado.

El identificador de función para esta función (para MQZEP) es MQZID_GET_EXPLICIT_AUTHORITY.

Sintaxis

MQZ_GET_EXPLICIT_AUTHORITY (*QMgrName, EntityName, EntityType, ObjectName, ObjectType, Authority, ComponentData, Continuation, CompCode, Reason*)

Parámetros

La llamada MQZ_GET_EXPLICIT_AUTHORITY tiene los parámetros siguientes.

QMgrName (MQCHAR48)-entrada

Nombre del gestor de colas.

El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

EntityName (MQCHAR12)-entrada

Nombre de entidad.

El nombre de la entidad desde la que se va a recuperar el acceso al objeto. La longitud máxima de la serie es de 12 caracteres; si es más corta que esa, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

EntityType (MQLONG)-entrada

Tipo de entidad.

El tipo de entidad especificado por *EntityName*. Se puede especificar el valor siguiente:

MQZAET_PRINCIPAL

Director.

GRUPO_MQZAC

group.

ObjectName (MQCHAR48)-entrada

El nombre del objeto.

El nombre del objeto para el que se va a recuperar la autorización de la entidad. La longitud máxima de la serie es de 48 caracteres; si es más corta, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

Si *ObjectType* es MQOT_Q_MGR, este nombre es el mismo que *QMgrName*.

ObjectType (MQLONG)-entrada

Tipo de objeto.

El tipo de entidad especificado por *ObjectName*. Es uno de los siguientes:

MQOT_AUTH_INFO

Información de autenticación.

MQOT_CHANNEL

Canal.

MQOT_CLNTCONN_CHANNEL

Canal de conexión de cliente.

MQOT_ESCUCHA

Escucha.

MQOT_LISTA_DE_NOMBRES

Lista de nombres.

MQOT_PROCESO

MQOT_Q

Cola.

MQOT_Q_MGR

Gestor de colas.

SERVICIO MQOT_SERVICE

Autorización (MQLONG)-salida

Autoridad de entidad.

Si la entidad tiene una autorización, este campo es igual a la operación de autorización adecuada (constante MQZAO_ *). Si tiene más de una autorización, este campo es el OR a nivel de bit de las constantes MQZAO_ * correspondientes.

ComponentData (MQBYTE x ComponentDataLength) - entrada/salida

Datos de componente.

El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de las funciones de este componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro **ComponentDataLength** de la llamada MQZ_INIT_AUTHORITY.

Continuación (MQLONG)-salida

Indicador de continuación establecido por componente.

Se pueden especificar los siguientes valores:

MQZCI_DEFAULT

Continuación dependiente del gestor de colas.

Para MQZ_GET_EXPLICIT_AUTHORITY esto tiene el mismo efecto que MQZCI_CONTINUE.

MQZCI_CONTINUAR

Continúe con el componente siguiente.

MQZCI_STOP

No continúe con el componente siguiente.

CompCode (MQLONG)-salida

Código de terminación.

Es uno de los siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón (MQLONG)-salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_NOT_AUTHORIZED

(2035, X'7F3') Acceso no autorizado.

MQRC_SERVICE_ERROR

(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICIO_NO_DISPONIBLE

(2285, X'8ED') Servicio subyacente no disponible.

MQRC_ENTIDAD_DESCONOCIDA

(2292, X'8F4') Entidad desconocida para el servicio.

Para obtener más información sobre estos códigos de razón, consulte [Mensajes y códigos de razón](#).

Invocación en C

```
MQZ_GET_EXPLICIT_AUTHORITY (QMGrName, EntityName, EntityType,  
                             ObjectName, ObjectType, &Authority,  
                             ComponentData, &Continuation,  
                             &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQCHAR48  QMGrName;           /* Queue manager name */  
MQCHAR12  EntityName;         /* Entity name */  
MQLONG    EntityType;         /* Entity type */  
MQCHAR48  ObjectName;         /* Object name */  
MQLONG    ObjectType;         /* Object type */  
MQLONG    Authority;          /* Authority of entity */  
MQBYTE    ComponentData[n];   /* Component data */  
MQLONG    Continuation;       /* Continuation indicator set by  
                               component */  
MQLONG    CompCode;           /* Completion code */  
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

MQZ_INIT_AUTHORITY (inicializar servicio de autorización) en IBM i

Esta función la proporciona un componente de servicio de autorización y la invoca el gestor de colas durante la configuración del componente. Se espera que llame a MQZEP para proporcionar información al gestor de colas.

El identificador de función para esta función (para MQZEP) es MQZID_INIT_AUTHORITY.

Sintaxis

MQZ_INIT_AUTHORITY (*Hconfig, Options, QMGrName, ComponentDataLength, ComponentData, Version, CompCode, Reason*)

Parámetros

La llamada MQZ_INIT_AUTHORITY tiene los parámetros siguientes.

Hconfig (MQHCONFIG)-entrada

Descriptor de contexto de configuración.

Este descriptor de contexto representa el componente concreto que se está inicializando. Debe ser utilizado por el componente al llamar al gestor de colas con la función MQZEP.

Opciones (MQLONG)-entrada

Opciones de inicialización.

Es uno de los siguientes:

MQZIO_PRIMARY

Inicialización primaria.

MQZIO_SECUNDARIO

Inicialización secundaria.

QMGrName (MQCHAR48)-entrada

Nombre del gestor de colas.

El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

ComponentDataLength (MQLONG) - entrada

Longitud de los datos de componente.

Longitud en bytes del área *ComponentData* . Esta longitud se define en los datos de configuración del componente.

ComponentData (MQBYTE x ComponentDataLength) - entrada/salida

Datos de componente.

Se inicializa con todos los ceros antes de llamar a la función de inicialización primaria del componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones (incluida la función de inicialización) proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de las funciones de este componente.

Versión (MQLONG)-entrada/salida

Número de versión.

En la entrada de la función de inicialización, identifica el número de versión *más alto* al que da soporte el gestor de colas. La función de inicialización debe cambiar esto, si es necesario, a la versión de la interfaz a la que *da soporte* . Si en el retorno el gestor de colas no da soporte a la versión devuelta por el componente, llama a la función MQZ_TERM_AUTHORITY del componente y no hace más uso de este componente.

Se admiten los valores siguientes:

MQZAS_VERSION_1

Versión 1.

MQZAS_VERSION_2

Versión 2.

MQZAS_VERSION_3

Versión 3.

MQZAS_VERSION_4

Versión 4.

MQZAS_VERSION_5

Versión 5.

MQZAS_VERSION_6

IBM WebSphere MQ 6.

CompCode (MQLONG)-salida

Código de terminación.

Es uno de los siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón (MQLONG)-salida

Código de razón que califica *CompCode* .

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_INICIALIZACIÓN_FALLIDA

(2286, X'8EE') La inicialización falló por una razón indefinida.

MQRC_SERVICIO_NO_DISPONIBLE

(2285, X'8ED') Servicio subyacente no disponible.

Para obtener más información sobre estos códigos de razón, consulte [Mensajes y códigos de razón](#).

Invocación en C

```
MQZ_INIT_AUTHORITY (Hconfig, Options, QMgrName, ComponentDataLength,  
                    ComponentData, &Version, &CompCode,  
                    &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQHCONFIG  Hconfig;           /* Configuration handle */  
MQLONG     Options;           /* Initialization options */  
MQCHAR48   QMgrName;          /* Queue manager name */  
MQLONG     ComponentDataLength; /* Length of component data */  
MQBYTE     ComponentData[n];  /* Component data */  
MQLONG     Version;           /* Version number */  
MQLONG     CompCode;          /* Completion code */  
MQLONG     Reason;            /* Reason code qualifying CompCode */
```

MQZ_INQUIRE (Consultar servicio de autorización) en IBM i

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_5 y la invoca el gestor de colas para consultar la funcionalidad soportada. Cuando se utilizan varios componentes de servicio, se llama a los componentes de servicio en orden inverso al orden en el que se instalaron.

El identificador de función para esta función (para MQZEP) es MQZID_INQUIRE.

Sintaxis

MQZ_INQUIRE

(QMGrName, SelectorCount, Selectors, IntAttrCount, IntAttrs, CharAttrLength, CharAttrs, SelectorReturned, ComponentData, Continuation, CompCode, Reason)

Parámetros

La llamada MQZ_INQUIRE tiene los parámetros siguientes.

QMGrName (MQCHAR48)-entrada

Nombre del gestor de colas.

El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

SelectorCount (MQLONG)-entrada

Número de selectores.

Número de selectores proporcionados en el parámetro Selectores.

El valor debe estar entre cero y 256.

Selectores (MQLONG x SelectorCount)-entrada

Selectores.

Matriz de selectores. Cada selector identifica un atributo necesario y debe ser de uno de los tipos siguientes:

- MQIACF_ * (entero)

- MQCACF_ * (carácter)

Los selectores se pueden especificar en cualquier orden. El número de selectores de la matriz se indica mediante el parámetro SelectorCount .

Los atributos enteros identificados por los selectores se devuelven en el parámetro IntAttrs en el mismo orden en que aparecen en los selectores.

Los atributos de carácter identificados por los selectores se devuelven en el parámetro CharAttrs en el mismo orden en que aparecen los selectores.

IntAttrCount (MQLONG) - entrada

Número de atributos enteros.

El número de atributos enteros proporcionados en el parámetro IntAttrs .

El valor debe estar en el rango de 0 a 256.

IntAttrs (MQLONG x IntAttrCount) - salida

Atributos enteros.

Matriz de atributos enteros. Los atributos de entero se devuelven en el mismo orden que los selectores de entero correspondientes en la matriz de selectores.

CharAttrCount (MQLONG) - entrada

Longitud del almacenamiento intermedio de atributos de caracteres.

Longitud en bytes del parámetro CharAttrs .

El valor debe al menos sumar las longitudes de los atributos de caracteres solicitados. Si no se solicita ningún atributo de carácter, cero es un valor válido.

CharAttrs (MQLONG x CharAttrCount) - salida

Almacenamiento intermedio de atributos de caracteres.

Almacenamiento intermedio que contiene atributos de caracteres, concatenados entre sí. Los atributos de carácter se devuelven en el mismo orden que los selectores de caracteres correspondientes en la matriz Selectores.

La longitud del búfer viene dada por el parámetro CharAttrCount.

SelectorReturnedMQLONGxSelectorCount) - entrada

Se ha devuelto el selector.

Matriz de valores que identifican qué atributos se han devuelto del conjunto solicitado por los selectores en el parámetro Selectores. El número de valores de esta matriz se indica mediante el parámetro SelectorCount . Cada valor en la matriz se relaciona con el selector desde la posición correspondiente en la matriz de selectores. Cada valor es uno de los siguientes:

MQZSL_DEVUELTO

Se ha devuelto el atributo solicitado por el selector correspondiente en el parámetro Selectores.

MQZSL_NO_DEVUELTO

El atributo solicitado por el selector correspondiente en el parámetro Selectores no se ha devuelto.

La matriz se inicializa con todos los valores como *MQZSL_NOT_RETURNED*. Cuando un componente de servicio de autorización devuelve un atributo, establece el valor adecuado en la matriz en *MQZSL_RETURNED*. Esto permite que cualquier otro componente de servicio de autorización, al que se realiza la llamada de consulta, identifique qué atributos ya se han devuelto.

ComponentData (MQBYTE x ComponentDataLength) - entrada/salida

Datos de componente.

El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de las funciones de este componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro **ComponentDataLength** de la llamada MQZ_INIT_AUTHORITY.

Continuación (MQLONG)-salida

Distintivo de continuación.

Se pueden especificar los siguientes valores:

MQZCI_DEFAULT

Continuación dependiente de otros componentes.

MQZCI_STOP

No continúe con el componente siguiente.

CompCode (MQLONG)-salida

Código de terminación.

Es uno de los siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_WARNING

Finalización parcial.

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón (MQLONG)-salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_WARNING:

MQRC_CHAR_ATTRS_TOO_SHORT

No hay espacio suficiente para los atributos de carácter.

MQRC_INT_COUNT_TOO_SMALL

No hay suficiente espacio para atributos enteros.

Si *CompCode* es MQCC_FAILED:

MQRC_SELECTOR_COUNT_ERROR

El número de selectores no es válido.

MQRC_SELECTOR_ERROR

Selector de atributo no válido.

MQRC_SELECTOR_LIMIT_EXCEEDED

Se han especificado demasiados selectores.

MQRC_INT_ATTR_COUNT_ERROR

El número de atributos enteros no es válido.

MQRC_INT_ATTRS_ARRAY_ERROR

Matriz de atributos enteros no válida.

MQRC_CHAR_ATTR_LENGTH_ERROR

El número de atributos de tipo carácter no es válido.

MQRC_CHAR_ATTRS_ERROR

La serie de atributos de tipo carácter no es válida.

MQRC_SERVICE_ERROR

(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

Invocación en C

```
MQZ_INQUIRE (QMgrName, SelectorCount, Selectors, IntAttrCount,
              &IntAttrs, CharAttrLength, &CharAttrs,
              SelectorReturned, ComponentData, &Continuation,
              &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQCHAR48  QMgrName;           /* Queue manager name */
MQLONG    SelectorCount;      /* Selector count */
MQLONG    Selectors[n];       /* Selectors */
MQLONG    IntAttrCount;       /* IntAttrs count */
MQLONG    IntAttrs[n];        /* Integer attributes */
MQLONG    CharAttrCount;      /* CharAttrs count */
MQLONG    CharAttrs[n];       /* Character attributes */
MQLONG    SelectorReturned[n]; /* Selector returned */
MQBYTE    ComponentData[n];   /* Component data */
MQLONG    Continuation;       /* Continuation indicator set by
                               component */
MQLONG    CompCode;           /* Completion code */
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

MQZ_REFRESH_CACHE (Renovar todas las autorizaciones) en IBM i

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_3 . Lo invoca el gestor de colas para renovar la lista de autorizaciones mantenidas internamente por el componente.

El identificador de función para esta función (para MQZEP) es MQZID_REFRESH_CACHE (8L).

Sintaxis

MQZ_REFRESH_CACHE

(*QMgrName*, *ComponentData*, *Continuación*, *CompCode*, *Motivo*)

Parámetros

QMgrName (MQCHAR48)-entrada

Nombre del gestor de colas.

El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

ComponentData (MQBYTE x ComponentDataLength) - entrada/salida

Datos de componente.

El gestor de colas conserva estos datos en nombre de este componente en particular. Los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una función del componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro *ComponentDataLength* de la llamada MQZ_INIT_AUTHORITY.

Continuación (MQLONG)-salida

Indicador de continuación establecido por componente.

Se pueden especificar los siguientes valores:

MQZCI_DEFAULT

Continuación dependiente del gestor de colas.

Para MQZ_REFRESH_CACHE, tiene el mismo efecto que MQZCI_CONTINUE.

MQZCI_CONTINUAR

Continúe con el componente siguiente.

MQZCI_STOP

No continúe con el componente siguiente.

CompCode (MQLONG)-salida

Código de terminación.

Es uno de los siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón (MQLONG)-salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_SERVICE_ERROR

(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

Invocación en C

```
MQZ_REFRESH_CACHE (QMgrName, ComponentData,  
                  &Continuation, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQBYTE    ComponentData[n];   /* Component data */  
MQLONG    Continuation;       /* Continuation indicator set by  
                               component */  
MQLONG    CompCode;           /* Completion code */  
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

MQZ_SET_AUTHORITY (Establecer autorización) en IBM i

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_1 y la invoca el gestor de colas para establecer la autorización que una entidad tiene para acceder al objeto especificado.

El identificador de función para esta función (para MQZEP) es MQZID_SET_AUTHORITY.

Nota: Esta función altera temporalmente las autorizaciones existentes. Para conservar las autorizaciones existentes, debe volver a establecerlas con esta función.

Sintaxis

MQZ_SET_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName, ObjectType, Authority, ComponentData, Continuation, CompCode, Reason)

Parámetros

La llamada MQZ_SET_AUTHORITY tiene los parámetros siguientes.

QMgrName (MQCHAR48)-entrada

Nombre del gestor de colas.

El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

EntityName (MQCHAR12)-entrada

Nombre de entidad.

El nombre de la entidad para la que se va a establecer el acceso al objeto. La longitud máxima de la serie es de 12 caracteres; si es más corta que esa, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

EntityType (MQLONG)-entrada

Tipo de entidad.

El tipo de entidad especificado por *EntityName*. Se puede especificar el valor siguiente:

MQZAET_PRINCIPAL

Director.

GRUPO_MQZAC

group.

ObjectName (MQCHAR48)-entrada

El nombre del objeto.

El nombre del objeto al que se necesita acceso. La longitud máxima de la serie es de 48 caracteres; si es más corta, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

Si *ObjectType* es MQOT_Q_MGR, este nombre es el mismo que *QMgrName*.

ObjectType (MQLONG)-entrada

Tipo de objeto.

El tipo de entidad especificado por *ObjectName*. Es uno de los siguientes:

MQOT_AUTH_INFO

Información de autenticación.

MQOT_CHANNEL

Canal.

MQOT_CLNTCONN_CHANNEL

Canal de conexión de cliente.

MQOT_ESCUCHA

Escucha.

MQOT_LISTA_DE_NOMBRES

Lista de nombres.

MQOT_PROCESO

.

MQOT_Q

Cola.

MQOT_Q_MGR

Gestor de colas.

SERVICIO MQOT_SERVICE

.

Autorización (MQLONG)-entrada

Autorización que se debe comprobar.

Si se establece una autorización, este campo es igual a la operación de autorización adecuada (constante MQZAO_*). Si se establece más de una autorización, es el OR a nivel de bit de las constantes MQZAO_* correspondientes.

ComponentData (MQBYTE x ComponentDataLength) - entrada/salida

Datos de componente.

El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de las funciones de este componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro **ComponentDataLength** de la llamada MQZ_INIT_AUTHORITY.

Continuación (MQLONG)-salida

Indicador de continuación establecido por componente.

Se pueden especificar los siguientes valores:

MQZCI_DEFAULT

Continuación dependiente del gestor de colas.

Para MQZ_SET_AUTHORITY tiene el mismo efecto que MQZCI_STOP.

MQZCI_CONTINUAR

Continúe con el componente siguiente.

MQZCI_STOP

No continúe con el componente siguiente.

CompCode (MQLONG)-salida

Código de terminación.

Es uno de los siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón (MQLONG)-salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_NOT_AUTHORIZED

(2035, X'7F3') Acceso no autorizado.

MQRC_SERVICE_ERROR

(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICIO_NO_DISPONIBLE

(2285, X'8ED') Servicio subyacente no disponible.

MQRC_ENTIDAD_DESCONOCIDA

(2292, X'8F4') Entidad desconocida para el servicio.

Invocación en C

```
MQZ_SET_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName,  
                  ObjectType, Authority, ComponentData,  
                  &Continuation, &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQCHAR48 QMgrName;          /* Queue manager name */  
MQCHAR12 EntityName;        /* Entity name */  
MQLONG   EntityType;        /* Entity type */  
MQCHAR48 ObjectName;        /* Object name */  
MQLONG   ObjectType;        /* Object type */  
MQLONG   Authority;         /* Authority to be checked */  
MQBYTE   ComponentData[n]; /* Component data */
```


MQLONG	Continuation;	/* Continuation indicator set by component */
MQLONG	CompCode;	/* Completion code */
MQLONG	Reason;	/* Reason code qualifying CompCode */

MQZ_TERM_AUTHORITY-Terminar servicio de autorización

Esta función la proporciona un componente de servicio de autorización y la invoca el gestor de colas cuando ya no necesita los servicios de este componente. La función debe realizar cualquier limpieza necesaria para el componente.

El identificador de función para esta función (para MQZEP) es MQZID_TERM_AUTHORITY.

Sintaxis

MQZ_TERM_AUTHORITY (*Hconfig, Options, QMgrName, ComponentData, CompCode, Reason*)

Parámetros

La llamada MQZ_TERM_AUTHORITY tiene los parámetros siguientes.

Hconfig (MQHCONFIG)-entrada

Descriptor de contexto de configuración.

Este descriptor de contexto representa el componente concreto que se está terminando.

Opciones (MQLONG)-entrada

Opciones de terminación.

Es uno de los siguientes:

MQZTO_PRIMARY

Terminación primaria.

MQZTO_SECUNDARIO

Terminación secundaria.

QMgrName (MQCHAR48)-entrada

Nombre del gestor de colas.

El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

ComponentData (MQBYTE x ComponentDataLength) - entrada/salida

Datos de componente.

El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de las funciones de este componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro **ComponentDataLength** de la llamada MQZ_INIT_AUTHORITY.

Cuando se ha completado la llamada MQZ_TERM_AUTHORITY, el gestor de colas descarta estos datos.

CompCode (MQLONG)-salida

Código de terminación.

Es uno de los siguientes:

MQCC_OK

Realización satisfactoria.

MQCC_FAILED

La llamada no se ha realizado satisfactoriamente.

Razón (MQLONG)-salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE

(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_SERVICIO_NO_DISPONIBLE

(2285, X'8ED') Servicio subyacente no disponible.

MQRC_TERMINACIÓN_FALLIDA

(2287, X'8FF') La finalización ha fallado por un motivo indefinido.

Para obtener más información sobre estos códigos de razón, consulte [Mensajes y códigos de razón](#).

Invocación en C

```
MQZ_TERM_AUTHORITY (Hconfig, Options, QMgrName, ComponentData,  
                    &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
MQHCONFIG  Hconfig;          /* Configuration handle */  
MQLONG     Options;          /* Termination options */  
MQCHAR48   QMgrName;        /* Queue manager name */  
MQBYTE     ComponentData[n]; /* Component data */  
MQLONG     CompCode;         /* Completion code */  
MQLONG     Reason;          /* Reason code qualifying CompCode */
```

MQZAC (contexto de aplicación) en IBM i

Este parámetro especifica los datos relacionados con la aplicación de llamada.

La estructura MQZAC se utiliza en la llamada MQZ_AUTHENTICATE_USER para el parámetro **ApplicationContext**.

Campos

StrucId (MQCHAR4)

Identificador de estructura.

El valor es:

MQZAC_STRUC_ID

Identificador de la estructura de contexto de aplicación.

Para el lenguaje de programación C, también se define la constante MQZAC_STRUC_ID_ARRAY; tiene el mismo valor que MQZAC_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Es un campo de entrada para el servicio.

Versión (MQLONG)

Número de versión de la estructura.

El valor es:

MQZAC_VERSION_1

Estructura de contexto de aplicación Version-1.

La constante siguiente especifica el número de versión de la versión actual:

MQZAC_VERSION_ACTUAL

Versión actual de la estructura de contexto de aplicación.

Es un campo de entrada para el servicio.

ProcessId (MQPID)

Identificador de proceso.

Identificador de proceso de la aplicación.

ThreadId (MQTID)

Identificador de hebra.

Identificador de hebra de la aplicación.

ApplName (MQCHAR28)

Nombre de aplicación.

Nombre de la aplicación.

UserID (MQCHAR12)

Identificador de usuario.

Para sistemas IBM i , el perfil de usuario bajo el que se ha creado el trabajo de aplicación. (En IBM i, cuando se realiza un intercambio de perfil con la API QWTSETP en el trabajo de aplicación, se devuelve el perfil de usuario actual).

EffectiveUserID (MQCHAR12)

Identificador de usuario efectivo.

Para sistemas IBM i , el perfil de usuario actual del trabajo de aplicación.

Entorno (MQLONG)

Medio ambiente.

Este campo especifica el entorno desde el que se ha realizado la llamada.

Puede tener uno de los valores siguientes:

MQXE_COMMAND_SERVER

Servidor de mandatos.

MQXE_MQSC

Intérprete de mandatos runmqsc .

MQXE_MCA

Agente de canal de mensajes

MQXE_OTHER

Entorno no definido

CallerType (MQLONG)

Tipo de interlocutor.

Este campo especifica el tipo de programa que ha realizado la llamada.

Puede tener uno de los valores siguientes:

MQXACT_EXTERNAL

La llamada es externa al gestor de colas.

MQXACT_INTERNAL

La llamada es interna al gestor de colas.

AuthenticationType (MQLONG)

Tipo de autenticación.

Este campo especifica el tipo de autenticación que se está realizando.

Puede tener uno de los valores siguientes:

MQZAT_CONTEXTO_INICIAL

La llamada de autenticación se debe a que se está inicializando el contexto de usuario. Este valor se utiliza durante una llamada MQCONN o MQCONNX.

CONTEXTO_CAMBIO_MQZAT_CONTEXTO

La llamada de autenticación se debe a que se ha cambiado el contexto de usuario. Este valor se utiliza cuando el MCA cambia el contexto de usuario.

V

BindType (MQLONG)

Tipo de enlace.

Este campo especifica el tipo de enlace en uso.

Puede tener uno de los valores siguientes:

MQCNO_FASTPATH_BINDING

Enlace de vía de acceso rápida.

MQCNO_SHARED_BINDING

Enlace compartido.

MQCNO_ISOLATED_BINDING

Enlace aislado.

Declaración C

```
typedef struct tagMQZAC MQZAC;
struct tagMQZAC {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQPID      ProcessId;        /* Process identifier */
    MQTID      ThreadId;         /* Thread identifier */
    MQCHAR28    ApplName;        /* Application name */
    MQCHAR12    UserID;           /* User identifier */
    MQCHAR12    EffectiveUserID;  /* Effective user identifier */
    MQLONG      Environment;     /* Environment */
    MQLONG      CallerType;       /* Caller type */
    MQLONG      AuthenticationType; /* Authentication type */
    MQLONG      BindType;        /* Bind type */
};
```

IBM i MQZAD (datos de autorización) en IBM i

La estructura MQZAD se utiliza en la llamada MQZ_ENUMERATE_AUTHORITY_DATA para dos parámetros.

Consulte “MQZ_ENUMERATE_AUTHORITY_DATA (Enumerar datos de autorización) en IBM i” en la [página 1724](#) para obtener más información sobre los parámetros **Filter** y **AuthorityBuffer**:

- MQZAD se utiliza para el parámetro **Filter** que se especifica en la llamada. Este parámetro especifica los criterios de selección que deben utilizarse para seleccionar los datos de autorización devueltos por la llamada.
- MQZAD también se utiliza para el parámetro **AuthorityBuffer** que es la salida de la llamada. Este parámetro especifica las autorizaciones para una combinación de nombre de perfil, tipo de objeto y entidad.

Campos

StrucId (MQCHAR4)

Identificador de estructura.

El valor es:

MQZAD_STRUC_ID

Identificador de la estructura de datos de autorización.

Para el lenguaje de programación C, también se define la constante MQZAD_STRUC_ID_ARRAY; tiene el mismo valor que MQZAD_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Es un campo de entrada para el servicio.

Versión (MQLONG)

Número de versión de la estructura.

El valor es:

MQZAD_VERSION_1

Estructura de datos de autorización Version-1 .

La constante siguiente especifica el número de versión de la versión actual:

MQZAD_VERSION_ACTUAL

Versión actual de la estructura de datos de autorización.

Es un campo de entrada para el servicio.

ProfileName (MQCHAR48)

Nombre de perfil.

Para el parámetro **Filter** , este campo es el nombre de perfil desde el que se necesitan los datos de autorización. Si el nombre está completamente en blanco hasta el final del campo o el primer carácter nulo, se devuelven los datos de autorización para todos los nombres de perfil.

Para el parámetro **AuthorityBuffer** , este campo es el nombre de un perfil que coincide con los criterios de selección especificados.

ObjectType (MQLONG)

Tipo de objeto.

Para el parámetro **Filter** , este campo es el tipo de objeto para el que se necesitan datos de autorización. Si el valor es MQOT_ALL, se devuelven los datos de autorización para todos los tipos de objeto.

Para el parámetro **AuthorityBuffer** , este campo es el tipo de objeto al que se aplica el perfil identificado por **ProfileName** .

El valor es uno de los siguientes; para el parámetro **Filter** , el valor MQOT_ALL también es válido:

MQOT_AUTH_INFO

Información de autenticación.

MQOT_CHANNEL

Canal.

MQOT_CLNTCONN_CHANNEL

Canal de conexión de cliente.

MQOT_ESCUCHA

Escucha.

MQOT_LISTA_DE_NOMBRES

Lista de nombres.

MQOT_PROCESO

.

MQOT_Q

Cola.

MQOT_Q_MGR

Gestor de colas.

SERVICIO MQOT_SERVICE

.

Autorización (MQLONG)

Autorización.

Para el parámetro **Filter** , este campo se ignora.

Para el parámetro **AuthorityBuffer** , este campo representa las autorizaciones que la entidad tiene para los objetos identificados por **ProfileName** y **ObjectType**. Si la entidad sólo tiene una autorización, el campo es igual al valor de autorización adecuado (constante MQZAO_ *). Si la entidad tiene más de una autorización, el campo es el OR a nivel de bit de las constantes MQZAO_ * correspondientes.

EntityDataPtr (PMQZED)

Dirección de la estructura MQZED que identifica una entidad.

Para el parámetro **Filter** , este campo apunta a una estructura MQZED que identifica la entidad de la que se necesitan datos de autorización. Si **EntityDataPtr** es el puntero nulo, se devuelven los datos de autorización para todas las entidades.

Para el parámetro **AuthorityBuffer** , este campo apunta a una estructura MQZED que identifica la entidad de la que proceden los datos de autorización devueltos.

EntityType (MQLONG)

Tipo de entidad.

Para el parámetro **Filter** , este campo especifica el tipo de entidad para el que se necesitan datos de autorización. Si el valor es MQZAET_NONE, se devuelven los datos de autorización para todos los tipos de entidad.

Para el parámetro **AuthorityBuffer** , este campo especifica el tipo de la entidad identificada por la estructura MQZED a la que apunta **EntityDataPtr**.

El valor es uno de los siguientes; para el parámetro **Filter** , el valor MQZAET_NONE también es válido:

MQZAET_PRINCIPAL

Director.

GRUPO_MQZAC

group.

Opciones (MQAUTHOPT)

Opciones.

Este campo especifica las opciones que dan control sobre los perfiles que se visualizan.

Se debe especificar uno de los siguientes:

MQAUTHOPT_NAME_ALL_MATCHING

Muestra todos los perfiles

MQAUTHOPT_NOMBRE_EXPLÍCITO

Muestra perfiles que tienen exactamente el mismo nombre que el especificado en el campo **ProfileName** .

Además, también debe especificarse uno de los siguientes:

MQAUTHOPT_ENTITY_SET

Visualizar todos los perfiles utilizados para calcular la autorización acumulativa que la entidad tiene sobre el objeto especificado por **ProfileName**. El campo **ProfileName** no debe contener ningún carácter comodín.

- Si la entidad especificada es un principal, para cada miembro del conjunto {entity, groups} se muestra el perfil más aplicable que se aplica al objeto.
- Si la entidad especificada es un grupo, se visualiza el perfil más aplicable del grupo que se aplica al objeto.

- Si se especifica este valor, los valores de **ProfileName**, **ObjectType**, **EntityType** y el nombre de entidad especificado en la estructura MQZED de **EntityDataPtr** no deben estar en blanco.

Si ha especificado *MQAUTHOPT_NAME_ALL_MATCHING*, también puede especificar lo siguiente:

MQAUTHOPT_ENTIDAD_EXPLÍCITA

Muestra perfiles que tienen exactamente el mismo nombre de entidad que el nombre de entidad especificado en la estructura MQZED de **EntityDataPtr**.

Declaración C

```
typedef struct tagMQZAD MQZAD;
struct tagMQZAD {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQCHAR48   ProfileName;      /* Profile name */
    MQLONG     ObjectType;       /* Object type */
    MQLONG     Authority;        /* Authority */
    PMQZED     EntityDataPtr;    /* Address of MQZED structure identifying an
                                entity */
    MQLONG     EntityType;       /* Entity type */
    MQAUTHOPT  Options;         /* Options */
};
```

MQZED (descriptor de entidad) en IBM i

La estructura MQZED se utiliza en un número de llamadas de servicio de autorización para especificar la entidad para la que se va a comprobar la autorización.

Campos

StrucId (MQCHAR4)

Identificador de estructura.

El valor es:

MQZED_STRUC_ID

Identificador de la estructura del descriptor de entidad.

Para el lenguaje de programación C, también se define la constante *MQZED_STRUC_ID_ARRAY*; tiene el mismo valor que *MQZED_STRUC_ID*, pero es una matriz de caracteres en lugar de una serie.

Es un campo de entrada para el servicio.

Versión (MQLONG)

Número de versión de la estructura.

El valor es:

MQZED_VERSION_1

Estructura del descriptor de entidad Version-1.

La constante siguiente especifica el número de versión de la versión actual:

MQZED_VERSION_ACTUAL

Versión actual de la estructura del descriptor de entidad.

Es un campo de entrada para el servicio.

EntityNamePtr (PMQCHAR)

Dirección del nombre de entidad.

Se trata de un puntero al nombre de la entidad cuya autorización se va a comprobar.

EntityDomainPtr (PMQCHAR)

Dirección del nombre de dominio de entidad.

Es un puntero al nombre del dominio que contiene la definición de la entidad cuya autorización se va a comprobar.

SecurityId (MQBYTE40)

Identificador de seguridad.

Es el identificador de seguridad cuya autorización se debe comprobar.

CorrelationPtr (MQPTR)

Puntero de correlación.

Esto facilita el paso de datos correlacionales entre la función de autenticar usuario y otras funciones de OAM adecuadas.

Declaración C

```
typedef struct tagMQZED MQZED;  
struct tagMQZED {  
    MQCHAR4    StrucId;           /* Structure identifier */  
    MQLONG     Version;          /* Structure version number */  
    PMQCHAR    EntityNamePtr;    /* Address of entity name */  
    PMQCHAR    EntityDomainPtr;  /* Address of entity domain name */  
    MQBYTE40   SecurityId;       /* Security identifier */  
    MQPTR      CorrelationPtr;   /* Address of correlation data */  
};
```

MQZFP (parámetros libres) en IBM i

Este parámetro especifica los datos relacionados con el recurso que se va a liberar.

La estructura MQZFP se utiliza en la llamada MQZ_FREE_USER para el parámetro **FreeParms**.

Campos

StrucId (MQCHAR4)

Identificador de estructura.

El valor es:

MQZFP_STRUC_ID

Identificador de la estructura de parámetros libres.

Para el lenguaje de programación C, la constante MQZFP_STRUC_ID_ARRAY también está definida; tiene el mismo valor que MQZFP_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Es un campo de entrada para el servicio.

Versión (MQLONG)

Número de versión de la estructura.

El valor es:

MQZFP_VERSION_1

Estructura de parámetros libres Version-1.

La constante siguiente especifica el número de versión de la versión actual:

MQZFP_VERSION_ACTUAL

Versión actual de la estructura de parámetros libres.

Es un campo de entrada para el servicio.

Reservado (MQBYTE8)

Campo reservado.

El valor inicial es nulo.

CorrelationPtr (MQPTR)

Puntero de correlación.

Dirección de los datos de correlación relacionados con el recurso que se va a liberar.

Declaración C

```
typedef struct tagMQZFP MQZFP;  
struct tagMQZFP {  
    MQCHAR4    StrucId;           /* Structure identifier */  
    MQLONG     Version;          /* Structure version number */  
    MQBYTE8    Reserved;        /* Reserved field */  
    MQPTR      CorrelationPtr;   /* Address of correlation data */  
};
```

IBM i MQZIC (Contexto de identidad) activado IBM i

La estructura MQZIC se utiliza en la llamada MQZ_AUTHENTICATE_USER para el **IdentityContext** parámetro.

La estructura MQZIC contiene información de contexto de identidad, que identifica al usuario de la aplicación que colocó por primera vez el mensaje en una cola:

- El gestor de colas rellena el campo « UserIdentifier » con un nombre que identifica al usuario. La forma en que el gestor de colas puede hacerlo depende del entorno en el que se ejecute la aplicación.
- El gestor de colas rellena el campo « AccountingToken » con un token o número que ha determinado a partir de la aplicación que ha enviado el mensaje.
- Las aplicaciones pueden utilizar el campo « ApplIdentityData » para cualquier información adicional que deseen incluir sobre el usuario (por ejemplo, una contraseña cifrada).

Las aplicaciones debidamente autorizadas pueden establecer el contexto de identidad utilizando la función MQZ_AUTHENTICATE_USER.

El identificador Windows de seguridad del sistema (SID) se almacena en el campo « AccountingToken » cuando se crea un mensaje en IBM MQ for Windows. El SID se puede utilizar para complementar el campo « UserIdentifier » (Identificador de usuario) y para establecer las credenciales de un usuario.

Campos

StrucId (MQCHAR4)

Identificador de estructura.

El valor es:

MQZIC_STRUC_ID

Identificador para la estructura del contexto de identidad.

Para el lenguaje de programación C, también se define la constante MQZIC_STRUC_ID_ARRAY, que tiene el mismo valor que MQZIC_STRUC_ID, pero es una matriz de caracteres en lugar de una cadena.

Este es un campo de entrada al servicio.

Versión (MQLONG)

Número de versión de la estructura.

El valor es:

MQZIC_VERSION_1

Version-1 estructura del contexto de identidad.

La siguiente constante especifica el número de versión de la versión actual:

MQZIC_VERSIÓN_ACTUAL

Versión actual de la estructura del contexto de identidad.

Este es un campo de entrada al servicio.

UserIdentifier (MQCHAR12)

Identificador de usuario.

Esto forma parte del **contexto** de identidad del mensaje.

UserIdentifier Especifica el identificador de usuario de la aplicación que originó el mensaje. El gestor de colas trata esta información como datos de caracteres, pero no define su formato. Para obtener más información sobre el *UserIdentifier* campo, consulte [“UserIdentifier \(MQCHAR12\)” en la página 429.](#)

AccountingToken (MQBYTE32)

Ficha contable.

Esto forma parte del **contexto** de identidad del mensaje.

AccountingToken permite que una aplicación haga que el trabajo realizado como resultado del mensaje se cobre adecuadamente. El gestor de colas trata esta información como una cadena de bits y no comprueba su contenido. Para obtener más información sobre el *AccountingToken* campo, consulte [“AccountingToken \(MQBYTE32\)” en la página 430.](#)

ApplIdentityData (MQCHAR32)

Datos de la aplicación relacionados con la identidad.

Esto forma parte del **contexto** de identidad del mensaje.

ApplIdentityData Es información definida por el conjunto de aplicaciones que se puede utilizar para proporcionar información adicional sobre el origen del mensaje. Por ejemplo, podrían establecerlo las aplicaciones que se ejecutan con la autoridad de usuario adecuada para indicar si los datos de identidad son fiables. Para obtener más información sobre el *ApplIdentityData* campo, consulte [“ApplIdentityData \(MQCHAR32\)” en la página 432.](#)

Declaración C

```
typedef struct tagMQZED MQZED;
struct tagMQZED {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQCHAR12   UserIdentifier;    /* User identifier */
    MQBYTE32   AccountingToken;  /* Accounting token */
    MQCHAR32   ApplIdentityData; /* Application data relating to identity */
};
```

Las clases e interfaces de IBM MQ .NET

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

Clase MQAsyncStatus.NET

Utilice *MQAsyncStatus* para consultar el estado de actividades anteriores de « MQI »; por ejemplo, para comprobar si se han realizado correctamente operaciones de escritura asincrónicas anteriores. *MQAsyncStatus* encapsula las características de la estructura de datos de MQSTS .

Clase

```
System.Object
├── IBM.WMQ.MQBase
│   └── IBM.WMQ.MQBaseObject
│       └── IBM.WMQ.MQAsyncStatus
```

```
public class IBM.WMQ.MQAsyncStatus extends IBM.WMQ.MQBaseObject;
```

- [“Propiedades” en la página 1751](#)
- [“Constructores” en la página 1751](#)

Propiedades

Prueba de MQException que se genera al obtener las propiedades.

```
public static int CompCode {get;}
```

El código de terminación del primer error o aviso.

```
public static int Reason {get;}
```

El código de razón del primer error o aviso.

```
public static int PutSuccessCount {get;}
```

El número de llamadas de inserción asincrónicas MQI que se han realizado correctamente.

```
public static int PutWarningCount {get;}
```

El número de llamadas asíncronas de «put» a MQI que se han ejecutado correctamente con una advertencia.

```
public static int PutFailureCount {get;}
```

El número de llamadas de inserción asincrónicas fallidas a `MQI`.

```
public static int ObjectType {get;}
```

El tipo de objeto para el primer error. Son posibles los siguientes valores:

- MQC.MQOT_ALIAS_Q
- MQC.MQOT_LOCAL_Q
- MQC.MQOT_MODEL_Q
- MQC.MQOT_Q
- MQC.MQOT_REMOTE_Q
- MQC.MQOT_TOPIC
- 0, lo que significa que no se devuelve ningún objeto

```
public static string ObjectName {get;}
```

El nombre de objeto.

```
public static string ObjectQMgrName {get;}
```

El nombre del gestor de colas de objetos.

```
public static string ResolvedObjectName {get;}
```

El nombre de objeto resuelto.

```
public static string ResolvedObjectQMgrName {get;}
```

El nombre del gestor de colas de objeto resuelto.

Constructores

```
public MQAsyncStatus() throws MQException;
```

Método constructor, construye un objeto con campos inicializados en cero o en blanco según corresponda.

Clase MQAuthenticationInformationRecord.NET

Utilice MQAuthenticationInformationRecord para especificar la información sobre un autenticador que se va a utilizar en una conexión de cliente IBM MQTLS. MQAuthenticationInformationRecord encapsula un registro de información de autenticación, MQAIR.

Clase

```
System.Object
└─ IBM.WMQ.MQAuthenticationInformationRecord
```

```
public class IBM.WMQ.MQAuthenticationInformationRecord extends System.Object;
```

- [“Propiedades” en la página 1752](#)
- [“Constructores” en la página 1752](#)

Propiedades

Prueba de MQException que se genera al obtener las propiedades.

public long Version {get; set;}

Número de versión de la estructura.

public long AuthInfoType {get; set;}

El tipo de información de autenticación. Este atributo debe establecerse en uno de los valores siguientes:

- OCSP -La comprobación de estado de revocación de certificados se realiza utilizando OCSP.
- CRLLDAP : la comprobación del estado de revocación de certificados se realiza mediante las listas de revocación de certificados (CRL) de los servidores de LDAP.

public string AuthInfoConnName {get; set;}

El nombre DNS o la dirección IP del servidor en el que se ejecuta el servidor LDAP, con un número de puerto opcional. Esta palabra clave es necesaria.

public string LDAPPassword {get; set;}

La contraseña asociada al nombre distinguido del usuario que está accediendo al servidor LDAP. Esta propiedad sólo se aplica cuando **AuthInfoType** se establece en CRLLDAP.

public string LDAPUserName {get; set;}

El nombre de usuario de la persona que está accediendo al servidor LDAP. Al configurar esta propiedad, las entradas « LDAPUserNameLength » y « LDAPUserNamePtr » se configuran automáticamente de forma correcta. Esta propiedad solo se aplica cuando la opción « AuthInfoType » está configurada en «CRLLDAP ».

public string OCSPResponderURL {get; set;}

El URL en el que se puede establecer contacto con el programa de respuesta OCSP. Esta propiedad sólo se aplica cuando AuthInfoType está configurada como OCSP

Este campo distingue entre mayúsculas y minúsculas. Debe comenzar con la cadena http:// en minúsculas. El resto de URL puede distinguir entre mayúsculas y minúsculas, dependiendo de la implementación del servidor OCSP.

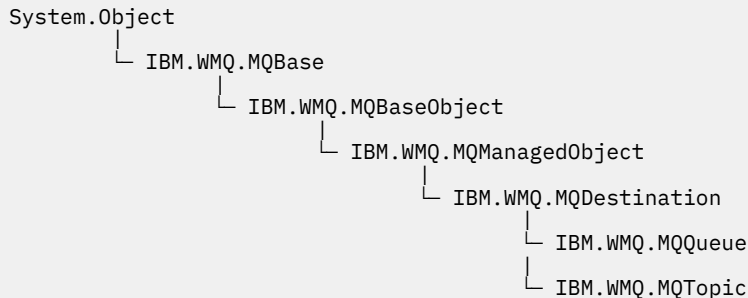
Constructores

MQAuthenticationInformationRecord();

Clase MQDestination.NET

Utilice MQDestination para acceder a métodos comunes a MQQueue y MQTopic. MQDestination es una clase base abstracta y no se puede instanciar.

Clase



```
public class IBM.WMQ.MQDestination extends IBM.WMQ.MQManagedObject;
```

- [“Propiedades” en la página 1753](#)
- [“Métodos” en la página 1753](#)
- [“Constructores” en la página 1755](#)

Propiedades

Prueba de MQException que se genera al obtener las propiedades.

```
public DateTime CreationDateTime {get;}
```

Fecha y hora en que se ha creado la cola o el tema. Originalmente contenida en MQQueue, esta propiedad se ha movido a la clase MQDestination base.

No hay ningún valor predeterminado.

```
public int DestinationType {get;}
```

Valor entero que describe el tipo de destino que se está utilizando. Inicializado desde el constructor de subclases, MQQueue o MQTopic, este valor puede tomar uno de estos valores:

- MQOT_Q
- MQOT_TOPIC

No hay ningún valor predeterminado.

Métodos

```
public void Get(MQMessage message);  
public void Get(MQMessage message, MQGetMessageOptions getMessageOptions);  
public void Get(MQMessage message, MQGetMessageOptions getMessageOptions, int  
MaxMsgSize);
```

Genera MQException.

Obtiene un mensaje de una cola si el destino es un objeto MQQueue o de un tema si el destino es un objeto MQTopic, utilizando una instancia predeterminada de MQGetMessageOptions para realizar la obtención.

Si la obtención falla, el objeto MQMessage no se modifica. Si se ejecuta correctamente, el descriptor de mensaje y las partes de datos de mensaje del MQMessage se sustituyen por el descriptor de mensaje y los datos de mensaje del mensaje de entrada.

Todas las llamadas a IBM MQ desde un `MQQueueManager` determinado son síncronas. Por lo tanto, si realiza una operación `get` con espera, todas las demás hebras que utilizan el mismo `MQQueueManager` no pueden realizar más llamadas IBM MQ hasta que se realice la llamada `Get`. Si necesita varias hebras para acceder a IBM MQ simultáneamente, cada hebra debe crear su propio objeto `MQQueueManager`.

mensaje

Contiene el descriptor de mensaje y los datos de mensaje devueltos. Algunos de los campos del descriptor de mensaje son parámetros de entrada. Es importante asegurarse de que los parámetros de entrada `MessageId` y `CorrelationId` estén establecidos según sea necesario.

Un cliente reconectable devuelve el código de razón `MQRC_BACKED_OUT` después de una reconexión satisfactoria, para los mensajes recibidos en `MQGM_SYNCPOINT`.

getMessageOptions

Opciones que controlan la acción de la obtención.

El uso de la opción `MQC.MQGMO_CONVERT` puede dar como resultado una excepción con el código de razón `MQC.MQRC_CONVERTED_STRING_TOO_BIG` al convertir de códigos de caracteres de un solo byte a códigos de doble byte. En este caso, el mensaje se copia en el almacenamiento intermedio sin conversión.

Si no se especifica `getMessageOptions`, la opción de mensaje utilizada es `MQGMO_NOWAIT`.

Si utiliza la opción `MQGMO_LOGICAL_ORDER` en un cliente reconectable, se devuelve el código de razón `MQRC_RECONNECT_INCOMPATIBLE`.

MaxMsgSize

El mensaje más grande que este objeto de mensaje va a recibir. Si el mensaje en la cola es mayor que este tamaño, se produce una de las dos cosas siguientes:

- Si el distintivo `MQGMO_ACCEPT_TRUNCATED_MSG` se establece en el objeto `MQGetMessageOptions`, el mensaje se rellena con la mayor cantidad posible de datos de mensaje. Se genera una excepción con el código de terminación `MQCC_WARNING` y el código de razón `MQRC_TRUNCATED_MSG_ACCEPTED`.
- Si el distintivo `MQGMO_ACCEPT_TRUNCATED_MSG` no está establecido, el mensaje permanece en la cola. Se genera una excepción con el código de terminación `MQCC_WARNING` y el código de razón `MQRC_TRUNCATED_MSG_FAILED`.

Si no se especifica `MaxMsgSize`, se recupera todo el mensaje.

```
public void Put(MQMessage message);  
public void Put(MQMessage message, MQPutMessageOptions putMessageOptions);
```

Genera `MQException`.

Coloca un mensaje en una cola si el destino es un objeto `MQQueue` o publica un mensaje en un tema si el destino es un objeto `MQTopic`.

Las modificaciones en el objeto `MQMessage` después de que se haya realizado la llamada `Put` no afectan al mensaje real en el tema de publicación o cola IBM MQ.

`Put` actualiza las propiedades `MessageId` y `CorrelationId` del objeto `MQMessage` y no borra los datos del mensaje. Las llamadas `Put` o `Get` adicionales hacen referencia a la información actualizada en el objeto `MQMessage`. Por ejemplo, en el siguiente fragmento de código, el primer mensaje contiene a y el segundo ab.

```
msg.WriteString("a");  
q.Put(msg, pmo);  
msg.WriteString("b");  
q.Put(msg, pmo);
```

mensaje

Un objeto `MQMessage` que contiene los datos del descriptor de mensaje y el mensaje que se va a enviar. El descriptor de mensaje se puede modificar como consecuencia de este método. Los

valores del descriptor de mensaje inmediatamente después de la finalización de este método son los valores que se colocaron en la cola o se publicaron en el tema.

Se devuelven los siguientes códigos de razón a un cliente reconectable:

- MQRC_CALL_INTERRUPTED si la conexión se interrumpe al ejecutar una llamada Put en un mensaje persistente y la reconexión se realiza correctamente.
- MQRC_NONE si la conexión es satisfactoria al ejecutar una llamada Put en un mensaje no persistente (consulte [Recuperación de aplicaciones](#)).

putMessageOptions

Opciones que controlan la acción de la colocación.

Si no se especifica *putMessageOptions*, se utiliza la instancia predeterminada de MQPutMessageOptions.

Si utiliza la opción MQPMO_LOGICAL_ORDER en un cliente reconectable, se devuelve el código de razón MQRC_RECONNECT_INCOMPATIBLE.

Nota: Por motivos de simplicidad y rendimiento, si desea transferir un único mensaje a una cola, utilice el objeto MQQueueManager.Put. Debe tener un objeto MQQueue para esto.

Constructores

MQDestination es una clase base abstracta y no se puede instanciar. Acceda a destinos utilizando constructores MQQueue y MQTopic, o utilizando MQQueueManager.AccessQueue y MQQueueManager.AccessTopic methods.

MQEnvironment.NET clase

Utilice MQEnvironment para controlar cómo se invoca el MQQueueManager constructor y para seleccionar una IBM MQ MQI client conexión. La MQEnvironment clase contiene propiedades que controlan el comportamiento de IBM MQ.

Clase

```
System.Object
└─ IBM.WMQ.MQEnvironment
```

```
public class IBM.WMQ.MQEnvironment extends System.Object;
```

- [“Propiedades - solo para clientes” en la página 1755](#)
- [“Propiedades” en la página 1756](#)
- [“Constructores” en la página 1758](#)

Propiedades - solo para clientes

Prueba de MQException que se genera al obtener las propiedades.

```
public static int CertificateValPolicy {get; set;}
```

Establezca qué política de validación de certificados TLS se utiliza para validar los certificados digitales recibidos de sistemas asociados remotos. Los valores válidos son:

- MQC.CERTIFICATE_VALIDATION_POLICY_ANY
- MQC.CERTIFICATE_VALIDATION_POLICY_RFC5280

```
public static ArrayList EncryptionPolicySuiteB {get; set;}
```

Establezca el nivel de criptografía compatible con Suite B. Los valores válidos son:

- MQC.MQ_SUITE_B_NONE - Este es el valor predeterminado.

- MQC.MQ_SUITE_B_128_BIT
- MQC.MQ_SUITE_B_192_BIT

public static string Channel {get; set;}

El nombre del canal para conectarse al gestor de colas de destino. Debe establecer la propiedad del canal antes de instanciar una MQQueueManager instancia en modo cliente.

public static int FipsRequired {get; set;}

Especificar MQC.MQSSL_FIPS_YES para utilizar únicamente FIPS algoritmos certificados si se lleva a cabo criptografía en IBM MQ. El valor predeterminado es MQC.MQSSL_FIPS_NO.

Si se configura hardware criptográfico, los módulos criptográficos utilizados son los proporcionados por el producto de hardware. Dependiendo del hardware utilizado, es posible que estos no estén FIPS certificados para un nivel concreto.

public static string Hostname {get; set;}

El nombre de host de TCP/IP del ordenador en el que se encuentra el IBM MQ servidor. Si no se ha establecido el nombre del host y no se han establecido propiedades de anulación, se utiliza el modo de enlaces del servidor para conectarse al gestor de colas local.

public static int Port {get; set;}

El puerto al que conectarse. Este es el puerto en el que el IBM MQ servidor está a la espera de solicitudes de conexión entrantes. El valor predeterminado es 1414.

public static string SSLCipherSpec {get; set;}

Establezca SSLCipherSpec el valor de la configuración de control de sesión (CipherSpec) en el canal SVRCONN para habilitar el control de sesión (TLS) para la conexión. El valor predeterminado es Null, y la conexión no tiene habilitada la opción « TLS » (Compartir conexión de red).

public static string sslPeerName {get; set;}

Un patrón de nombres distinguido. Si sslCipherSpec está configurado, esta variable se puede utilizar para garantizar que se utilice el gestor de colas correcto. Si se establece en nulo (valor predeterminado), no se ejecuta el DN del gestor de colas. sslPeerName se ignora si sslCipherSpec es nulo.

public static string SSLKeyRepository {get; set;}

La frase de contraseña en texto plano o cifrada para acceder al repositorio de claves. Las frases de contraseña del repositorio de claves se cifran para su uso por parte de las aplicaciones cliente mediante la **runmqicred** utilidad.

Si SSLKeyRepositoryPassword se establece en nulo (valor predeterminado), se utiliza el valor de la variable de entorno MQKEYRPWD o el **SSLKeyRepositoryPassword** atributo del archivo de configuración del cliente.

public static string InitialKey {get; set;}

La clave inicial que se utilizó para cifrar la frase de contraseña del repositorio de claves especificada en SSLKeyRepositoryPassword.

Se debe especificar la clave inicial si se especificó un archivo de clave inicial cuando se cifró la frase de contraseña del repositorio de claves con la **runmqicred** utilidad.

Propiedades

Prueba de MQException que se genera al obtener las propiedades.

public static ArrayList HdrCompList {get; set;}

Lista de compresión de datos de cabecera

public static int KeyResetCount {get; set;}

Indica el número de bytes sin cifrar enviados y recibidos dentro de una conversación TLS antes de que se renegocie la clave secreta.

public static ArrayList MQAIRArray {get; set;}

Una matriz de MQAuthenticationInformationRecord objetos.

public static ArrayList MsgCompList {get; set;}

Lista de compresión de datos de mensajes

public static string Password {get; set;}

La contraseña que se va a autenticar. La contraseña a la que se hace referencia desde la estructura MQCSP se rellena configurando esta propiedad Contraseña.

public static string ReceiveExit {get; set;}

Una salida de recepción le permite examinar y modificar los datos recibidos de un gestor de colas. Normalmente se utiliza con una salida de envío correspondiente en el gestor de colas. Si ReceiveExit se establece en nulo, no se llama a ninguna salida de recepción.

public static string ReceiveUserData {get; set;}

Los datos de usuario asociados con una salida de recepción. Limitado a 32 caracteres.

public static string SecurityExit {get; set;}

Una salida de seguridad le permite personalizar los flujos de seguridad que se producen cuando se intenta conectarse a un gestor de colas. Si SecurityExit se establece en nulo, no se llama a ninguna salida de seguridad.

public static string SecurityUserData {get; set;}

Los datos de usuario asociados a una salida de seguridad. Limitado a 32 caracteres.

public static string SendExit {get; set;}

Una salida de envío le permite examinar o modificar los datos enviados a un gestor de colas. Normalmente se utiliza con una salida de recepción correspondiente en el gestor de colas. Si SendExit se establece en nulo, no se llama a la salida de envío.

public static string SendUserData {get; set;}

Los datos de usuario asociados con una salida de envío. Limitado a 32 caracteres.

public static string SharingConversations {get; set;}

El SharingConversations campo se utiliza en conexiones desde .NET aplicaciones, cuando estas aplicaciones no utilizan una tabla de definición de canales de cliente (CCDT).

SharingConversations determina el número máximo de conversaciones que se pueden compartir en un socket asociado a esta conexión.

Un valor de 0 significa que el canal funciona como antes IBM WebSphere MQ 7.0, en lo que respecta al intercambio de conversaciones, la lectura anticipada y el latido.

El campo se pasa en la tabla hash de propiedades como un SHARING_CONVERSATIONS_PROPERTY, al instanciar un IBM MQ gestor de colas.

Si no se especifica SharingConversations, se utiliza el valor predeterminado 10.

public static string SSLCryptoHardware {get; set;}

Establece el nombre de la cadena de parámetros necesaria para configurar el hardware criptográfico presente en el sistema. SSLCryptoHardware se ignora si sslCipherSpec es nulo.

public static string SSLKeyRepository {get; set;}

Establezca el nombre completo del archivo del repositorio de claves.

Si no se proporciona la extensión del archivo, se supone que es .kdb

Si SSLKeyRepository se establece en nulo (valor predeterminado), se utiliza la variable de entorno del certificado MQSSLKEYR para localizar el repositorio de claves.

public static string UserId {get; set;}

El ID de usuario que se va a autenticar. El ID de usuario al que se hace referencia desde la estructura MQCSP se rellena estableciendo UserId. Autenticar UserId utilizando una API o una salida de seguridad.

Constructores

public MQEnvironment()

Clase MQException.NET

Utilice MQException para averiguar el código de terminación y de razón de una función IBM MQ anómala. Se genera un MQException siempre que se produce un error IBM MQ .

Clase

```
System.Object
├── System.Exception
│   └── System.ApplicationException
│       └── IBM.WMQ.MQException
```

```
public class IBM.WMQ.MQException extends System.ApplicationException;
```

- [“Propiedades” en la página 1758](#)
- [“Constructores” en la página 1758](#)

Propiedades

public int CompletionCode {get; set;}

El código de terminación de IBM MQ asociado con el error. Los valores posibles son:

- MQException.MQCC_OK
- MQException.MQCC_WARNING
- MQException.MQCC_FAILED

public int ReasonCode {get; set;}

Código de razón IBM MQ que describe el error.

Constructores

public MQException(int *completionCode*, int *reasonCode*)

completionCode

El código de terminación de IBM MQ .

reasonCode

El código de terminación de IBM MQ .

Clase MQGetMessageOptions.NET

Utilice MQGetMessageOptions para especificar cómo se recuperan los mensajes. Modifica el comportamiento de MQDestination.Get.

Clase

```
System.Object
├── IBM.WMQ.MQBase
│   └── IBM.WMQ.MQBaseObject
│       └── IBM.WMQ.MQGetMessageOptions
```

```
public class IBM.WMQ.MQGetMessageOptions extends IBM.WMQ.MQBaseObject;
```

- “Propiedades” en la página 1759
- “Constructores” en la página 1761

Propiedades

Nota: El comportamiento de algunas de las opciones disponibles en esta clase depende del entorno en el que se utilizan. Estos elementos se marcan con un asterisco *.

Prueba de `MQException` que se genera al obtener las propiedades.

public int GroupStatus {get;}*

`GroupStatus` indica si el mensaje recuperado está en un grupo y si es el último del grupo. Los valores posibles son:

MQC.MQGS_LAST_MSG_IN_GROUP

El mensaje es el último o único mensaje del grupo.

MQC.MQGS_MSG_IN_GROUP

El mensaje está en un grupo, pero no es el último del grupo.

MQC.MQGS_NOT_IN_GROUP

El mensaje no está en un grupo.

public int MatchOptions {get; set;}*

`MatchOptions` determina cómo se selecciona un mensaje. Se pueden establecer las siguientes opciones de coincidencia:

MQC.MQMO_MATCH_CORREL_ID

ID de correlación que debe coincidir.

MQC.MQMO_MATCH_GROUP_ID

ID de grupo que debe coincidir.

MQC.MQMO_MATCH_MSG_ID

ID de mensaje que debe coincidir.

MQC.MQMO_MATCH_MSG_SEQ_NUMBER

Coincide con el número de secuencia de mensaje.

MQC.MQMO_NONE

No se requiere ninguna coincidencia.

public int Options {get; set;}*

Las Opciones controlan la acción de `MQQueue.get`. Se puede especificar cualquiera de los valores siguientes. Si se necesita más de una opción, los valores se pueden añadir o combinar utilizando el operador OR a nivel de bit.

MQC.MQGMO_ACCEPT_TRUNCATED_MSG

Permitir truncamiento de datos de mensaje.

MQC.MQGMO_ALL_MSGS_AVAILABLE*

Recuperar mensajes de un grupo sólo cuando todos los mensajes del grupo están disponibles.

MQC.MQGMO_ALL_SEGMENTS_AVAILABLE*

Recuperar los segmentos de un mensaje lógico sólo cuando todos los segmentos del grupo están disponibles.

MQC.MQGMO_BROWSE_FIRST

Examinar desde el inicio de la cola.

MQC.MQGMO_BROWSE_MSG_UNDER_CURSOR*

Examine el mensaje bajo el cursor para examinar.

MQC.MQGMO_BROWSE_NEXT

Examinar desde la posición actual en la cola.

MQC.MQGMO_COMPLETE_MSG*

Recuperar sólo mensajes lógicos completos.

MQC.MQGMO_CONVERT

Solicite que los datos de aplicación se conviertan, para que se ajusten a los atributos CharacterSet y Codificación del MQMessage, antes de que los datos se copien en el almacenamiento intermedio de mensajes. Puesto que la conversión de datos también se aplica cuando los datos se recuperan del almacenamiento intermedio de mensajes, las aplicaciones no establecen esta opción.

El uso de esta opción puede causar problemas al convertir de juegos de caracteres de un solo byte a juegos de caracteres de doble byte. En su lugar, realice la conversión utilizando los métodos readString, readLine y writeString después de que se haya entregado el mensaje.

MQC.MQGMO_FAIL_IF QUIESCING

Fallar si el gestor de colas se está desactivando temporalmente.

MQC.MQGMO_LOCK*

Bloquear el mensaje que se examina.

MQC.MQGMO_LOGICAL_ORDER*

Devuelve mensajes en grupos y segmentos de mensajes lógicos, en orden lógico.

Si utiliza la opción MQGMO_LOGICAL_ORDER en un cliente reconectable, el código de razón MQRC_RECONNECT_INCOMPATIBLE se devuelve a la aplicación.

MQC.MQGMO_MARK_SKIP_BACKOUT*

Permitir que se restituya una unidad de trabajo sin restablecer el mensaje en la cola.

MQC.MQGMO_MSG_UNDER_CURSOR

Obtener mensaje bajo el cursor para examinar.

MQC.MQGMO_NONE

No se ha especificado ninguna otra opción; todas las opciones asumen sus valores predeterminados.

MQC.MQGMO_NO_PROPERTIES

No se recupera ninguna propiedad del mensaje, excepto las propiedades contenidas en el descriptor de mensaje (o extensión).

MQC.MQGMO_NO_SYNCPOINT

Obtener mensaje sin control de punto de sincronización.

MQC.MQGMO_NO_WAIT

Vuelva inmediatamente si no hay ningún mensaje adecuado.

MQC.MQGMO_PROPERTIES_AS_Q_DEF

Recupere las propiedades de mensaje tal como las define el atributo PropertyControl de MQQueue. El acceso a las propiedades del mensaje en el descriptor de mensaje, o extensión, no se ve afectado por el atributo PropertyControl.

MQC.MQGMO_PROPERTIES_COMPATIBILITY

Recupere las propiedades de mensaje con el prefijo mcd, jms, usro mqext, en las cabeceras MQRFH2. Otras propiedades del mensaje, excepto las propiedades contenidas en el descriptor de mensaje, o extensión, se descartan.

MQC.MQGMO_PROPERTIES_FORCE_MQRFH2

Recuperar propiedades de mensaje, excepto las propiedades contenidas en el descriptor de mensaje, o extensión, en las cabeceras MQRFH2. Utilice MQC.MQGMO_PROPERTIES_FORCE_MQRFH2 en aplicaciones que esperan recuperar propiedades pero que no se pueden cambiar para utilizar manejadores de mensajes.

MQC.MQGMO_PROPERTIES_IN_HANDLE

Recuperar propiedades de mensaje utilizando un MsgHandle.

MQC.MQGMO_SYNCPOINT

Obtenga el mensaje bajo control de punto de sincronización. El mensaje se marca como no disponible para otras aplicaciones, pero sólo se suprime de la cola cuando se confirma la unidad de trabajo. El mensaje vuelve a estar disponible si se restituye la unidad de trabajo.

MQC.MQGMO_SYNCPOINT_IF_PERSISTENT*

Obtener mensaje con control de punto de sincronización si el mensaje es persistente.

MQC.MQGMO_UNLOCK*

Desbloquear un mensaje bloqueado anteriormente.

MQC.MQGMO_WAIT

Espere a que llegue un mensaje.

public string ResolvedQueueName {get;}

El gestor de colas establece ResolvedQueueName en el nombre local de la cola desde la que se recuperó el mensaje. ResolvedQueueName es diferente del nombre utilizado para abrir la cola si se abrió una cola de alias o una cola modelo.

public char Segmentation {get;}*

Segmentación indica si puede permitir la segmentación para el mensaje recuperado. Los valores posibles son:

MQC.MQSEG_INHIBITED

No permitir segmentación.

MQC.MQSEG_ALLOWED

Permitir segmentación

public byte SegmentStatus {get;}*

SegmentStatus es un campo de salida que indica si el mensaje recuperado es un segmento de un mensaje lógico. Si el mensaje es un segmento, el distintivo indica si es el último segmento. Los valores posibles son:

MQC.MQSS_LAST_SEGMENT

El mensaje es el último o único segmento del mensaje lógico.

MQC.MQSS_NOT_A_SEGMENT

El mensaje no es un segmento.

MQC.MQSS_SEGMENT

El mensaje es un segmento, pero no es el último segmento del mensaje lógico.

public int WaitInterval {get; set;}

WaitInterval es el tiempo máximo en milisegundos que una llamada MQQueue.get espera a que llegue un mensaje adecuado. Utilice WaitInterval con MQC.MQGMO_WAIT. Establezca un valor de MQC.MQWI_UNLIMITED para esperar un tiempo ilimitado para un mensaje.

Constructores**public MQGetMessageOptions()**

Construye un nuevo objeto 'MQGetMessageOptions' con Options establecido a 'MQC.MQGMO_NO_WAIT', WaitInterval establecido a cero, y ResolvedQueueName establecido a blank.

Clase MQManagedObject.NET

Utilice MQManagedObject para consultar y establecer atributos de MQDestination, MQProcess, MQQueueManagery MQSubscription. MQManagedObject es una superclase de estas clases.

Clases

```

System.Object
├── IBM.WMQ.MQBase
│   └── IBM.WMQ.MQBaseObject
│       └── IBM.WMQ.MQManagedObject
│           ├── IBM.WMQ.MQProcess
│           ├── IBM.WMQ.MQQueueManager
│           ├── IBM.WMQ.MQSubscription
│           └── IBM.WMQ.MQDestination

```

```
public class IBM.WMQ.MQManagedObject extends IBM.WMQ.MQBaseObject;
```

- [“Propiedades” en la página 1762](#)
- [“Métodos” en la página 1763](#)
- [“Constructores” en la página 1764](#)

Propiedades

Prueba de MQException que se genera al obtener las propiedades.

```
public string AlternateUserId {get; set;}
```

El ID de usuario alternativo, si lo hay, establecido cuando se abrió el recurso.
AlternateUserId.set se ignora cuando se emite para un objeto que está abierto.
AlternateUserId no es válido para las suscripciones.

```
public int CloseOptions {get; set;}
```

Establezca este atributo para controlar la forma en que se cierra el recurso. El valor predeterminado es MQC.MQCO_NONE. MQC.MQCO_NONE es el único valor permitido para todos los recursos que no sean colas dinámicas permanentes, colas dinámicas temporales, suscripciones y temas a los que acceden los objetos que las han creado.

Para colas y temas, se permiten los siguientes valores adicionales:

MQC.MQCO_DELETE

Suprime la cola si no hay mensajes.

MQC.MQCO_DELETE_PURGE

Suprime la cola, depurando los mensajes que haya en ella.

MQC.MQCO_QUIESCE

Solicite que se cierre la cola, recibiendo un aviso si queda algún mensaje (permitiendo que se recuperen antes del cierre final).

Para suscripciones, se permiten los siguientes valores adicionales:

MQC.MQCO_KEEP_SUB

La suscripción no se suprime. Esta opción sólo es válida si la suscripción original es duradera.
MQC.MQCO_KEEP_SUB es el valor predeterminado para un tema duradero.

MQC.MQCO_REMOVE_SUB

La suscripción se suprime. MQC.MQCO_REMOVE_SUB es el valor predeterminado para un tema no duradero y no gestionado.

MQC.MQCO_PURGE_SUB

La suscripción se suprime. MQC.MQCO_PURGE_SUB es el valor predeterminado para un tema gestionado no duradero.

```
public MQQueueManager ConnectionReference {get;}
```

El gestor de colas al que pertenece este recurso.

public string MQDescription {get;}

La descripción del recurso tal como la retiene el gestor de colas. MQDescription devuelve una serie vacía para suscripciones y temas.

public boolean IsOpen {get;}

Indica si el recurso está abierto actualmente.

public string Name {get;}

Nombre del recurso. El nombre es el proporcionado en el método de acceso o el asignado por el gestor de colas para una cola dinámica.

public int OpenOptions {get; set;}

Las OpenOptions se establecen cuando se abre un objeto IBM MQ . El método OpenOptions.set se ignora y no provoca un error. Las suscripciones no tienen OpenOptions.

Métodos

public virtual void Close();

Genera MQException.

Cierra el objeto. No se permiten más operaciones en este recurso después de llamar a Close. Para cambiar el comportamiento del método Close , establezca el atributo closeOptions .

public string GetAttributeString(int selector, int length);

Genera MQException.

Obtiene una serie de atributo.

selector

Entero que indica qué atributo se está consultando.

longitud

Entero que indica la longitud de la serie necesaria.

public void Inquire(int[] selectors, int[] intAttrs, byte[] charAttrs);

Genera MQException.

Devuelve una matriz de enteros y un conjunto de series de caracteres que contienen los atributos de una cola, un proceso o un gestor de colas. Los atributos que se van a consultar se especifican en la matriz de selectores.

Nota: Muchos de los atributos más comunes se pueden consultar utilizando los métodos Get definidos en MQManagedObject, MQQueue y MQQueueManager.

selectores

Matriz de enteros que identifica los atributos con los valores que se van a consultar.

intAttrs

La matriz en la que se devuelven los valores de atributo de entero. Los valores de atributos enteros se devuelven en el mismo orden que los selectores de atributos enteros de la matriz de selectores.

charAttrs

El almacenamiento intermedio en el que se devuelven los atributos de carácter, concatenado. Los atributos de caracteres se devuelven en el mismo orden que los selectores de atributos de caracteres de la matriz de selectores. La longitud de cada serie de atributo es fija para cada atributo.

public void Set(int[] selectors, int[] intAttrs, byte[] charAttrs);

Genera MQException.

Establece los atributos definidos en el vector de selectores. Los atributos que se van a establecer se especifican en la matriz de selectores.

selectores

Matriz de enteros que identifica los atributos con valores que se van a establecer.

intAttrs

La matriz de valores de atributo de entero que se van a establecer. Estos valores deben estar en el mismo orden que los selectores de atributos enteros de la matriz de selectores.

charAttrs

El almacenamiento intermedio en el que se concatenan los atributos de carácter que se van a establecer. Estos valores deben estar en el mismo orden que los selectores de atributos de caracteres de la matriz de selectores. La longitud de cada atributo de carácter es fija.

public void SetAttributeString(int selector, string value, int length);

Genera MQException.

Establece una serie de atributo.

selector

Entero que indica qué atributo se está definiendo.

valor

La serie que se va a establecer como valor de atributo.

longitud

Entero que indica la longitud de la serie necesaria.

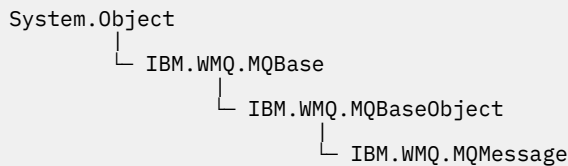
Constructores

protected MQManagedObject()

Método constructor. Este objeto es una clase base abstracta que no se puede instanciar por sí misma.

Clase MQMessage.NET

Utilice MQMessage para acceder al descriptor de mensaje y a los datos de un mensaje IBM MQ . MQMessage encapsula un mensaje IBM MQ .

Clase

```
public class IBM.WMQ.MQMessage extends IBM.WMQ.MQBaseObject;
```

Cree un objeto MQMessage y, a continuación, utilice los métodos Read y Write para transferir datos entre el mensaje y otros objetos de la aplicación. Enviar y recibir objetos MQMessage utilizando los métodos Put y Get de las clases MQDestination, MQQueue y MQTopic .

Obtenga y establezca las propiedades del descriptor de mensaje utilizando las propiedades de MQMessage. Establezca y obtenga propiedades de mensajes ampliadas utilizando los métodos SetProperty y GetProperty .

- [“Propiedades” en la página 1764](#)
- [“Métodos de mensaje Read y Write” en la página 1770](#)
- [“Métodos de almacenamiento intermedio” en la página 1773](#)
- [“Métodos de propiedad” en la página 1773](#)
- [“Constructores” en la página 1775](#)

Propiedades

Prueba de MQException que se genera al obtener las propiedades.

public string AccountingToken {get; set;}

Parte del contexto de identidad del mensaje; ayuda a una aplicación a cobrar por el trabajo realizado como resultado del mensaje. El valor predeterminado es MQC.MQACT_NONE.

public string ApplicationIdData {get; set;}

Parte del contexto de identidad del mensaje. ApplicationIdData es información definida por el conjunto de aplicaciones, y puede utilizarse para proporcionar información adicional sobre el mensaje o su emisor. El valor predeterminado es "".

public string ApplicationOriginData {get; set;}

Información definida por la aplicación que se puede utilizar para proporcionar información adicional sobre el origen del mensaje. El valor predeterminado es "".

public int BackoutCount {get;}

Recuento del número de veces que una llamada MQQueue.Get ha devuelto y restituido previamente el mensaje como parte de una unidad de trabajo. El valor predeterminado es cero.

public int CharacterSet {get; set;}

El identificador de juego de caracteres codificado de datos de caracteres en el mensaje.

Establezca CharacterSet para identificar el juego de caracteres de los datos de caracteres en el mensaje. Obtenga CharacterSet para averiguar en qué juego de caracteres se ha utilizado para codificar datos de caracteres en el mensaje.

Las aplicaciones .NET siempre se ejecutan en Unicode, mientras que en otros entornos las aplicaciones se ejecutan en el mismo juego de caracteres con el que se ejecuta el gestor de colas.

Los métodos ReadString y ReadLine convierten los datos de caracteres del mensaje a Unicode automáticamente.

El método WriteString se convierte de Unicode al juego de caracteres codificado en CharacterSet. Si CharacterSet se establece en su valor predeterminado, MQC.MQCCSI_Q_MGR, que es 0, no tiene lugar ninguna conversión y CharacterSet se establece en 1200. Si establece CharacterSet en algún otro valor, WriteString se convierte de Unicode al valor alternativo.

Nota: Otros métodos de lectura y escritura no utilizan CharacterSet.

- ReadChar y WriteChar leen y escriben un carácter Unicode en y desde el almacenamiento intermedio de mensajes sin conversión.
- ReadUTF y WriteUTF se convierten entre una serie Unicode en la aplicación y una serie UTF-8, con el prefijo de un campo de longitud de 2 bytes, en el almacenamiento intermedio de mensajes.
- Los métodos de bytes transfieren bytes entre la aplicación y el almacenamiento intermedio de mensajes sin modificación.

public byte[] CorrelationId {get; set;}

- Para una llamada MQQueue.Get, el identificador de correlación del mensaje que se va a recuperar. El gestor de colas devuelve el primer mensaje con un identificador de mensaje y un identificador de correlación que coinciden con los campos del descriptor de mensaje. El valor predeterminado, MQC.MQCI_NONE, ayuda a que coincida cualquier identificador de correlación.
- Para una llamada MQQueue.Put, el identificador de correlación a establecer.

public int DataLength {get;}

Número de bytes de datos de mensaje que quedan por leer.

public int DataOffset {get; set;}

La posición actual del cursor dentro de los datos del mensaje. Las lecturas y grabaciones entran en vigor en la posición actual.

public int Encoding {get; set;}

Representación utilizada para valores numéricos en los datos de mensaje de aplicación. Codificación se aplica a datos binarios, decimales empaquetados y de coma flotante. El comportamiento de los métodos de lectura y escritura para estos formatos numéricos se modifica en consecuencia. Construya un valor para el campo de codificación añadiendo un valor de cada una

de estas tres secciones. De forma alternativa, construya el valor que combina los valores de cada una de las tres secciones utilizando el operador OR a nivel de bit.

1. Entero binario

MQC.MQENC_INTEGER_NORMAL

Enteros big-endian.

MQC.MQENC_INTEGER_REVERSED

Enteros little-endian, tal como se utiliza en la arquitectura de Intel .

2. Decimal empaquetado

MQC.MQENC_DECIMAL_NORMAL

Big-endian packed-decimal, tal como lo utiliza z/OS.

MQC.MQENC_DECIMAL_REVERSED

Decimal empaquetado little-endian.

3. Coma flotante

MQC.MQENC_FLOAT_IEEE_NORMAL

Flotadores IEEE de big-endian.

MQC.MQENC_FLOAT_IEEE_REVERSED

Little-endian IEEE floats, tal como se utiliza en la arquitectura de Intel .

MQC.MQENC_FLOAT_S390

z/OS formatear puntos flotantes.

El valor predeterminado es:

```
MQC.MQENC_INTEGER_REVERSED |  
MQC.MQENC_DECIMAL_REVERSED |  
MQC.MQENC_FLOAT_IEEE_REVERSED
```

El valor predeterminado hace que WriteInt escriba un entero little-endian y que ReadInt lea un entero little-endian. Si establece el distintivo MQC.MQENC_INTEGER_NORMAL en su lugar, WriteInt escribe un entero big-endian y ReadInt lee un entero big-endian.

Nota: Se puede producir una pérdida de precisión al convertir de puntos flotantes de formato IEEE a puntos flotantes de formato zSeries .

public int Expiry {get; set;}

Tiempo de caducidad expresado en décimas de segundo, establecido por la aplicación que coloca el mensaje. Una vez transcurrido el tiempo de caducidad de un mensaje, el gestor de colas lo puede descartar. Si el mensaje ha especificado uno de los distintivos MQC.MQRO_EXPIRATION , se genera un informe cuando se descarta el mensaje. El valor predeterminado es MQC.MQEI_UNLIMITED, lo que significa que el mensaje nunca caduca.

public int Feedback {get; set;}

Utilice Comentarios con un mensaje de tipo MQC.MQMT_REPORT para indicar la naturaleza del informe. El sistema define los siguientes códigos de retorno:

- MQC.MQFB_EXPIRATION
- MQC.MQFB_COA
- MQC.MQFB_COD
- MQC.MQFB_QUIT
- MQC.MQFB_PAN
- MQC.MQFB_NAN
- MQC.MQFB_DATA_LENGTH_ZERO
- MQC.MQFB_DATA_LENGTH_NEGATIVE
- MQC.MQFB_DATA_LENGTH_TOO_BIG
- MQC.MQFB_BUFFER_OVERFLOW

- MQC.MQFB_LENGTH_OFF_BY_ONE
- MQC.MQFB_IIH_ERROR

También se pueden utilizar los valores de comentarios definidos por la aplicación en el rango de MQC.MQFB_APPL_FIRST a MQC.MQFB_APPL_LAST . El valor predeterminado de este campo es MQC.MQFB_NONE, lo que indica que no se proporciona ningún comentario.

public string Format {get; set;}

Nombre de formato utilizado por el remitente del mensaje para indicar al destinatario la naturaleza de los datos del mensaje. Puede utilizar sus propios nombres de formato, pero los nombres que empiezan por las letras MQ tienen significados definidos por el gestor de colas. Los formatos incorporados del gestor de colas son:

MQC.MQFMT_ADMIN

Mensaje de solicitud/respuesta del servidor de mandatos.

MQC.MQFMT_COMMAND_1

Mensaje de respuesta de mandato de tipo 1.

MQC.MQFMT_COMMAND_2

Mensaje de respuesta de mandato de tipo 2.

MQC.MQFMT_DEAD_LETTER_HEADER

Cabecera de mensaje no entregado.

MQC.MQFMT_EVENT

Mensaje de suceso.

MQC.MQFMT_NONE

No hay nombre de formato.

MQC.MQFMT_PCF

Mensaje definido por el usuario en formato de mandato programable.

MQC.MQFMT_STRING

Mensaje que consta totalmente de caracteres.

MQC.MQFMT_TRIGGER

Mensaje de desencadenante

MQC.MQFMT_XMIT_Q_HEADER

Cabecera de cola de transmisión.

El valor predeterminado es MQC.MQFMT_NONE.

public byte[] GroupId {get; set;}

Serie de bytes que identifica el grupo de mensajes al que pertenece el mensaje físico. El valor predeterminado es MQC.MQGI_NONE.

public int MessageFlags {get; set;}

Distintivos que controlan la segmentación y el estado de un mensaje.

public byte[] MessageId {get; set;}

Para una llamada MQQueue .Get , este campo especifica el identificador de mensaje del mensaje que se va a recuperar. Normalmente, el gestor de colas devuelve el primer mensaje con un identificador de mensaje y un identificador de correlación que coinciden con los campos del descriptor de mensaje. Permitir que cualquier identificador de mensaje coincida utilizando el valor especial MQC.MQMI_NONE.

Para una llamada MQQueue .Put , este campo especifica el identificador de mensaje que se va a utilizar. Si se especifica MQC.MQMI_NONE , el gestor de colas genera un identificador de mensaje exclusivo cuando se coloca el mensaje. El valor de esta variable de miembro se actualiza después de la colocación, para indicar el identificador de mensaje que se ha utilizado. El valor predeterminado es MQC.MQMI_NONE.

public int MessageLength {get;}

Número de bytes de datos de mensaje en el objeto MQMessage .

public int MessageSequenceNumber {get; set;}

Número de secuencia de un mensaje lógico dentro de un grupo.

public int MessageType {get; set;}

Indica el tipo del mensaje. El sistema define actualmente los valores siguientes:

- MQC.MQMT_DATAGRAM
- MQC.MQMT_REPLY
- MQC.MQMT_REPORT
- MQC.MQMT_REQUEST

También se pueden utilizar valores definidos por la aplicación, en el rango de MQC.MQMT_APPL_FIRST a MQC.MQMT_APPL_LAST. El valor predeterminado de este campo es MQC.MQMT_DATAGRAM.

public int Offset {get; set;}

En un mensaje segmentado, el desplazamiento de datos en un mensaje físico desde el inicio de un mensaje lógico.

public int OriginalLength {get; set;}

La longitud original de un mensaje segmentado.

public int Persistence {get; set;}

Persistencia de mensajes. Los valores siguientes están definidos:

- MQC.MQPER_NOT_PERSISTENT

Si establece esta opción en un cliente reconectable, el código de razón MQRC_NONE se devuelve a la aplicación cuando la conexión es satisfactoria.

- MQC.MQPER_PERSISTENT

Si establece esta opción en un cliente reconectable, el código de razón MQRC_CALL_INTERRUPTED se devuelve a la aplicación después de que la conexión sea satisfactoria.

- MQC.MQPER_PERSISTENCE_AS_Q_DEF

El valor predeterminado es MQC.MQPER_PERSISTENCE_AS_Q_DEF, que toma la persistencia del mensaje del atributo de persistencia predeterminado de la cola de destino.

public int Priority {get; set;}

Prioridad del mensaje. El valor especial MQC.MQPRI_PRIORITY_AS_Q_DEF también se puede establecer en el mensaje de salida. A continuación, la prioridad del mensaje se toma del atributo de prioridad predeterminado de la cola de destino. El valor predeterminado es MQC.MQPRI_PRIORITY_AS_Q_DEF.

public int PropertyValidation {get; set;}

Especifica si la validación de propiedades tiene lugar cuando se establece una propiedad del mensaje. Los valores posibles son:

- MQCMHO_DEFAULT_VALIDATION
- MQCMHO_VALIDATE
- MQCMHO_NO_VALIDATION

El valor predeterminado es MQCMHO_DEFAULT_VALIDATION.

public string PutApplicationName {get; set;}

El nombre de la aplicación que ha transferido el mensaje. El valor predeterminado es "".

public int PutApplicationType {get; set;}

El tipo de aplicación que transfiere el mensaje. PutApplicationType puede ser un valor definido por el sistema o por el usuario. El sistema define los valores siguientes:

- MQC.MQAT_AIX
- MQC.MQAT_CICS
- MQC.MQAT_DOS
- MQC.MQAT_IMS

- MQC.MQAT_MVS
- MQC.MQAT_OS2
- MQC.MQAT_OS400
- MQC.MQAT_QMGR
- MQC.MQAT_UNIX
- MQC.MQAT_WINDOWS
- MQC.MQAT_JAVA

El valor predeterminado es MQC.MQAT_NO_CONTEXT, que indica que no hay información de contexto en el mensaje.

public DateTime PutDateTime {get; set;}

La fecha y hora en que se ha colocado el mensaje.

public string ReplyToQueueManagerName {get; set;}

El nombre del gestor de colas para enviar mensajes de respuesta o de informe. El valor por defecto es "", y el gestor de colas proporciona el ReplyToQueueManagerName.

public string ReplyToQueueName {get; set;}

El nombre de la cola de mensajes a la que la aplicación que ha emitido la solicitud get para el mensaje envía mensajes MQC.MQMT_REPLY y MQC.MQMT_REPORT. El ReplyToQueueName por defecto es "".

public int Report {get; set;}

Utilice Informe para especificar opciones sobre los mensajes de informe y respuesta:

- Indica si los informes son necesarios.
- Indica si los datos de mensaje de aplicación se van a incluir en los informes.
- Cómo establecer los identificadores de mensaje y correlación en el informe o respuesta.

Se puede solicitar cualquier combinación de los cuatro tipos de informe:

- Especifique cualquier combinación de los cuatro tipos de informe. Seleccionar cualquiera de las tres opciones para cada tipo de informe, en función de si los datos del mensaje de aplicación se van a incluir en el mensaje de informe.

1. Confirmar al recibir

- MQC.MQRO_COA
- MQC.MQRO_COA_WITH_DATA
- MQC.MQRO_COA_WITH_FULL_DATA **

2. Confirmar al entregar

- MQC.MQRO_COD
- MQC.MQRO_COD_WITH_DATA
- MQC.MQRO_COD_WITH_FULL_DATA **

3. Excepción

- MQC.MQRO_EXCEPTION
- MQC.MQRO_EXCEPTION_WITH_DATA
- MQC.MQRO_EXCEPTION_WITH_FULL_DATA **

4. Caducidad

- MQC.MQRO_EXPIRATION
- MQC.MQRO_EXPIRATION_WITH_DATA
- MQC.MQRO_EXPIRATION_WITH_FULL_DATA **

Nota: Los valores marcados con ** en la lista no están soportados por los gestores de colas de z/OS . No los utilice si es probable que la aplicación acceda a un gestor de colas de z/OS , independientemente de la plataforma en la que se ejecute la aplicación.

- Especifique una de las opciones siguientes para controlar cómo se genera el ID de mensaje para el informe o el mensaje de respuesta:
 - MQC.MQRO_NEW_MSG_ID
 - MQC.MQRO_PASS_MSG_ID
- Especifique una de las opciones siguientes para controlar cómo se va a establecer el ID de correlación del informe o mensaje de respuesta:
 - MQC.MQRO_COPY_MSG_ID_TO_CORREL_ID
 - MQC.MQRO_PASS_CORREL_ID
- Especifique uno de los siguientes para controlar la disposición del mensaje original cuando no se pueda entregar a la cola de destino:
 - MQC.MQRO_DEAD_LETTER_Q
 - MQC.MQRO_DISCARD_MSG **
- Si no se especifica ninguna opción de informe, el valor predeterminado es:

```
MQC.MQRO_NEW_MSG_ID |  
MQC.MQRO_COPY_MSG_ID_TO_CORREL_ID |  
MQC.MQRO_DEAD_LETTER_Q
```

- Puede especificar uno o ambos de los siguientes para solicitar que la aplicación receptora envíe un mensaje de informe de acción positiva o de acción negativa.
 - MQC.MQRO_PAN
 - MQC.MQRO_NAN

public int TotalMessageLength {get;}

Número total de bytes en el mensaje tal como se ha almacenado en la cola de mensajes desde la que se ha recibido este mensaje.

public string UserId {get; set;}

UserId forma parte del contexto de identidad del mensaje. El gestor de colas generalmente proporciona el valor. Puede alterar temporalmente el valor si tiene autorización para establecer el contexto de identidad.

public int Version {get; set;}

La versión de la estructura MQMD en uso.

Métodos de mensaje Read y Write

Los métodos Read y Write realizan las mismas funciones que los miembros de las clases `BinaryReader` y `BinaryWriter` en el espacio de nombres `.NET System.IO` . Consulte MSDN para ver los ejemplos de uso y sintaxis de lenguaje completo. Los métodos leen o escriben desde la posición actual en el almacenamiento intermedio de mensajes. Mueven la posición actual hacia adelante por el número de bytes leídos o grabados.

Nota: Si los datos del mensaje contienen una cabecera MQRFH o MQRFH2 , debe utilizar el método `ReadBytes` para leer los datos.

- Todos los métodos emiten `IOException`.
- Los métodos `ReadFully` redimensionan automáticamente la matriz `byte` o `sbyte` de destino para que se ajuste exactamente al mensaje. También se cambia el tamaño de una matriz nula.
- `Read` métodos throw `EndOfStreamException`.
- `WriteDecimal` métodos throw `MQException`.
- Los métodos `ReadString`, `ReadLine` y `WriteString` se convierten entre Unicode y el juego de caracteres del mensaje; consulte [CharacterSet](#) .

- Los métodos `Decimal` leen y escriben números decimales empaquetados codificados en formato big-endian, `MQC.MQENC_DECIMAL_NORMAL` o little-endian `MQC.MQENC_DECIMAL_REVERSE`, según el valor de Codificación. Los rangos decimales y los tipos .NET correspondientes son los siguientes:

Decimal2/short

-999 a 999

Decimal4/int

-9999999 a 9999999

Decimal8/long

-9999999999999999 a 9999999999999999

- Los métodos `Double` y `Float` leen y escriben valores flotantes codificados en formatos big-endian y little-endian de IEEE, `MQC.MQENC_FLOAT_IEEE_NORMAL` y `MQC.MQENC_FLOAT_IEEE_REVERSED`, o en formato S/390, `MQC.MQENC_FLOAT_S390`, según el valor de Codificación.
- Los métodos `Int` leen y escriben valores enteros codificados en formato big-endian, `MQC.MQENC_INTEGER_NORMAL` o little-endian, `MQC.MQENC_INTEGER_REVERSED`, según el valor de Codificación. Los enteros están todos con signo, excepto la adición de un tipo entero de 2 bytes sin signo. Los tamaños enteros y los tipos .NET y IBM MQ son los siguientes:

2 bytes

`short`, `Int2`, `ushort`, `UInt2`

4 bytes

`int`, `Int4`

8 bytes

`long`, `Int8`

- `WriteObject` transfiere la clase de un objeto, los valores de sus campos no transitorios y no estáticos, y los campos de sus supertipos, al almacenamiento intermedio de mensajes.
- `ReadObject` crea un objeto a partir de la clase del objeto, la firma de la clase y los valores de sus campos no transitorios y no estáticos, y los campos de sus supertipos.

Tabla 879. Métodos de lectura y escritura de mensajes	
Tipo de destino	Signaturas de método
Boolean	<pre>public bool ReadBoolean(); public void WriteBoolean(bool value);</pre>
Byte	<pre>public byte ReadByte() public byte ReadUnsignedByte() public void Write(int value) public void WriteByte(int value) public void WriteByte(byte value) public void WriteByte(sbyte value)</pre>
Bytes	<pre>public byte[] ReadBytes(int count) public void ReadFully(ref byte[] value) public void ReadFully(ref sbyte[] value) public void ReadFully(ref byte[] value, int offset, int length) public void ReadFully(ref sbyte[] value, int offset, int length) public void Write(byte[] value) public void Write(sbyte[] value) public void Write(byte[] value, int offset, int length) public void Write(sbyte[] value, int offset, int length) public void WriteBytes(string value)</pre>

Tabla 879. Métodos de lectura y escritura de mensajes (continuación)

Tipo de destino	Signaturas de método
Decimal2	<code>public void WriteDecimal2(short value)</code>
Decimal4	<code>public void WriteDecimal4(short value)</code>
Decimal8	<code>public void WriteDecimal8(short value)</code>
Double	<code>public double ReadDouble()</code> <code>public void WriteDouble(double value)</code>
Float	<code>public float ReadFloat()</code> <code>public void WriteFloat(float value)</code>
Int2	<code>public void WriteInt2(int value)</code>
Int4	<code>public int readDecimal4()</code> <code>public int ReadInt()</code> <code>public int ReadInt4()</code> <code>public void WriteInt(int value)</code> <code>public void WriteInt4(int value)</code>
Int8	<code>public void WriteInt8(long value)</code>
Long	<code>public long ReadDecimal8()</code> <code>public long ReadLong()</code> <code>public long ReadInt8()</code> <code>public void WriteLong(long value)</code>
Object	<code>public Object ReadObject()</code> <code>public void WriteObject(Object object)</code>
Short	<code>public short ReadShort()</code> <code>public short ReadDecimal2()</code> <code>public short ReadInt2()</code> <code>public void WriteShort(int value)</code>
string	<code>public string ReadString(int length)</code> <code>public void WriteString(string string)</code>

Tabla 879. Métodos de lectura y escritura de mensajes (continuación)

Tipo de destino	Signaturas de método
Unsigned Short	<pre>public ushort ReadUnsignedShort() public ushort ReadUInt2()</pre>
Unicode	<pre>public string ReadLine() public char ReadChar() public void WriteChar(int value) public void WriteChars(string string)</pre>
UTF	<pre>public string ReadUTF() public void WriteUTF(string string)</pre>

Métodos de almacenamiento intermedio

public void ClearMessage();

Genera `IOException`.

Descarta los datos del almacenamiento intermedio de mensajes y vuelve a establecer el desplazamiento de datos en cero.

public void ResizeBuffer(int size)

Genera `IOException`.

Una sugerencia para el objeto `MQMessage` sobre el tamaño del almacenamiento intermedio que puede ser necesario para las operaciones `get` posteriores. Si el mensaje contiene actualmente datos de mensaje y el nuevo tamaño es menor que el tamaño actual, los datos del mensaje se truncan.

public void Seek(int pos)

Genera `IOException`, `ArgumentOutOfRangeException`, `ArgumentException`.

Mueve el cursor a la posición absoluta en el almacenamiento intermedio de mensajes proporcionado por *pos*. Las lecturas y grabaciones posteriores actúan en esta posición del almacenamiento intermedio.

public int SkipBytes(int i)

Genera `IOException`, `EndOfStreamException`.

Avanza *n* bytes en el almacenamiento intermedio de mensajes y devuelve *n*, el número de bytes omitidos.

El método `SkipBytes` se bloquea hasta que se produce uno de los sucesos siguientes:

- Se omiten todos los bytes
- Se ha detectado el final del almacenamiento intermedio de mensajes
- Se genera una excepción

Métodos de propiedad

public void DeleteProperty(string name);

Genera `MQException`.

Suprime una propiedad con el nombre especificado del mensaje.

nombre

El nombre de la propiedad que se va a suprimir.

public System.Collections.IEnumerator GetPropertyNames(string name)

Genera MQException.

Devuelve un IEnumerator de todos los nombres de propiedad que coinciden con el nombre especificado. El signo de porcentaje '%' se puede utilizar al final del nombre como carácter comodín para filtrar las propiedades del mensaje, coincidiendo con cero o más caracteres, incluido el punto.

nombre

El nombre de la propiedad en la que debe coincidir.

Métodos SetProperty y GetProperty

Todos los métodos SetProperty y GetProperty emiten MQException.

El método SetProperty de la clase MQMessage.NET añade una propiedad nueva si todavía no existe una propiedad. Sin embargo, si la propiedad ya existe, el valor de propiedad proporcionado se añade al final de la lista. Cuando varios valores se establecen en un nombre de propiedad utilizando SetProperty, al llamar a GetProperty para ese nombre se devuelven estos valores secuencialmente en el orden en el que se han establecido esos valores.

El comportamiento es el mismo para todos los métodos con tipo Set*Property y Get*Property como GetLongProperty, SetLongProperty, GetBooleanProperty, SetBooleanProperty, GetStringProperty y SetStringProperty.

Tabla 880. Métodos SetProperty y GetProperty	
Tipo	Signaturas de método
Boolean	<pre>public boolean GetBooleanProperty(string name); public boolean GetBooleanProperty(string name, MQPropertyDescriptor pd); public void SetBooleanProperty(string name, boolean value); public void SetBooleanProperty(string name, MQPropertyDescriptor pd, boolean value);</pre>
Byte	<pre>public sbyte GetByteProperty(string name); public sbyte GetByteProperty(string name, MQPropertyDescriptor pd); public void SetByteProperty(string name, sbyte value); public void SetByteProperty(string name, MQPropertyDescriptor pd, sbyte value);</pre>
Bytes	<pre>public sbyte[] GetBytesProperty(string name); public sbyte[] GetBytesProperty(string name, MQPropertyDescriptor pd); public void SetBytesProperty(string name, sbyte[] value); public void SetBytesProperty(string name, MQPropertyDescriptor pd, sbyte[] value);</pre>
Double	<pre>public double GetDoubleProperty(string name); public double GetDoubleProperty(string name, MQPropertyDescriptor pd); public void SetDoubleProperty(string name, double value); public void SetDoubleProperty(string name, MQPropertyDescriptor pd, double value);</pre>
Float	<pre>public float GetFloatProperty(string name); public float GetFloatProperty(string name, MQPropertyDescriptor pd); public void SetFloatProperty(string name, float value); public void SetFloatProperty(string name, MQPropertyDescriptor pd, float value);</pre>

Tabla 880. Métodos SetProperty y GetProperty (continuación)

Tipo	Signaturas de método
Int2	<pre>public short GetInt2Property(string name); public short GetInt2Property(string name, MQPropertyDescriptor pd); public void SetInt2Property(string name, short value); public void SetInt2Property(string name, MQPropertyDescriptor pd, short value);</pre>
Int4	<pre>public int GetInt4Property(string name); public int GetInt4Property(string name, MQPropertyDescriptor pd); public void SetInt4Property(string name, int value); public void SetInt4Property(string name, MQPropertyDescriptor pd, int value);</pre>
Int8	<pre>public long GetInt8Property(string name); public long GetInt8Property(string name, MQPropertyDescriptor pd); public void SetInt8Property(string name, long value); public void SetInt8Property(string name, MQPropertyDescriptor pd, long value);</pre>
Long	<pre>public long GetLongProperty(string name); public long GetLongProperty(string name, MQPropertyDescriptor pd); public void SetLongProperty(string name, long value); public void SetLongProperty(string name, MQPropertyDescriptor pd, long value);</pre>
Object	<pre>public Object GetObjectProperty(string name); public Object GetObjectProperty(string name, MQPropertyDescriptor pd); public void SetObjectProperty(string name, Object value); public void SetObjectProperty(string name, MQPropertyDescriptor pd, Object value);</pre>
Short	<pre>public short GetShortProperty(string name); public short GetShortProperty(string name, MQPropertyDescriptor pd); public void SetShortProperty(string name, short value); public void SetShortProperty(string name, MQPropertyDescriptor pd, short value);</pre>
string	<pre>public string GetStringProperty(string name); public string GetStringProperty(string name, MQPropertyDescriptor pd); public void SetStringProperty(string name, string value); public void SetStringProperty(string name, MQPropertyDescriptor pd, string value);</pre>

Constructores

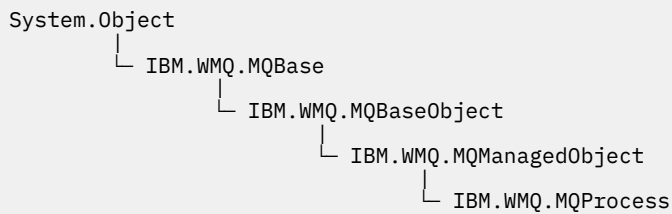
public MQMessage();

Crea un objeto MQMessage con información de descriptor de mensaje predeterminado y un almacenamiento intermedio de mensajes vacío.

Clase MQProcess.NET

Utilice MQProcess para consultar los atributos de un proceso de IBM MQ . Cree un objeto MQProcess utilizando un constructor o un método MQQueueManager AccessProcess .

Clase



```
public class IBM.WMQ.MQProcess extends IBM.WMQ.MQManagedObject;
```

- [“Propiedades” en la página 1776](#)
- [“Constructores” en la página 1777](#)

Propiedades

Prueba de MQException que se genera al obtener las propiedades.

public string ApplicationId {get;}

Obtiene la serie de caracteres que identifica la aplicación que se va a iniciar. ApplicationId lo utiliza una aplicación de supervisor desencadenante. ApplicationId se envía a la cola de inicio como parte del mensaje desencadenante.

El valor por omisión es nulo.

public int ApplicationType {get;}

Identifica el tipo de proceso que debe iniciar una aplicación de supervisor desencadenante. Los tipos estándar están definidos, pero se pueden utilizar otros:

- MQAT_AIX
- MQAT_CICS
- MQAT_IMS
- MQAT_MVS
- MQAT_NATIVE
- MQAT_OS400
- MQAT_UNIX
- MQAT_WINDOWS
- MQAT_JAVA
- MQAT_USER_FIRST
- MQAT_USER_LAST

El valor predeterminado es MQAT_NATIVE.

public string EnvironmentData {get;}

Obtiene información sobre el entorno de la aplicación que se va a iniciar.

El valor por omisión es nulo.

public string UserData {get;}

Obtiene información que el usuario ha proporcionado sobre la aplicación que se va a iniciar.

El valor por omisión es nulo.

Constructores

```
public MQProcess(MQQueueManager queueManager, string processName, int
openOptions);
public MQProcess(MQQueueManager qMgr, string processName, int openOptions,
string queueManagerName, string alternateUserId);
```

Genera MQException.

Acceda a un proceso IBM MQ en el gestor de colas *qMgr* para consultar los atributos de proceso.

qMgr

Gestor de colas al que acceder.

processName

El nombre del proceso que se va a abrir.

openOptions

Opciones que controlan la apertura del proceso. Las opciones válidas que se pueden añadir o combinar utilizando un OR a nivel de bit son:

- MQC.MQ00_FAIL_IF QUIESCING
- MQC.MQ00_INQUIRE
- MQC.MQ00_SET
- MQC.MQ00_ALTERNATE_USER_AUTHORITY

queueManagerName

El nombre del gestor de colas en el que se define el proceso. Puede dejar un nombre de gestor de colas en blanco o nulo si el gestor de colas es el mismo que al que está accediendo el proceso.

alternateUserId

Si se especifica MQC.MQ00_ALTERNATE_USER_AUTHORITY en el parámetro **openOptions**, *alternateUserId* especifica el ID de usuario alternativo utilizado para comprobar la autorización de la acción. Si no se especifica MQ00_ALTERNATE_USER_AUTHORITY, *alternateUserId* puede estar en blanco o ser nulo.

Se utiliza la autorización de usuario predeterminada para la conexión con el gestor de colas si no se especifica MQC.MQ00_ALTERNATE_USER_AUTHORITY.

```
public MQProcess MQQueueManager.AccessProcess(string processName, int
openOptions);
public MQProcess MQQueueManager.AccessProcess(string processName, int
openOptions, string queueManagerName, string alternateUserId);
```

Genera MQException.

Acceda a un proceso de IBM MQ en este gestor de colas para consultar los atributos de proceso.

processName

El nombre del proceso que se va a abrir.

openOptions

Opciones que controlan la apertura del proceso. Las opciones válidas que se pueden añadir o combinar utilizando un OR a nivel de bit son:

- MQC.MQ00_FAIL_IF QUIESCING
- MQC.MQ00_INQUIRE
- MQC.MQ00_SET
- MQC.MQ00_ALTERNATE_USER_AUTHORITY

queueManagerName

El nombre del gestor de colas en el que se define el proceso. Puede dejar un nombre de gestor de colas en blanco o nulo si el gestor de colas es el mismo que al que está accediendo el proceso.

alternateUserId

Si se especifica MQC.MQOO_ALTERNATE_USER_AUTHORITY en el parámetro **openOptions**, *alternateUserId* especifica el ID de usuario alternativo utilizado para comprobar la autorización de la acción. Si no se especifica MQOO_ALTERNATE_USER_AUTHORITY, *alternateUserId* puede estar en blanco o ser nulo.

Se utiliza la autorización de usuario predeterminada para la conexión con el gestor de colas si no se especifica MQC.MQOO_ALTERNATE_USER_AUTHORITY.

Clase MQPropertyDescriptor.NET

Utilice MQPropertyDescriptor como parámetro para los métodos MQMessage.GetProperty y SetProperty. MQPropertyDescriptor describe una propiedad MQMessage.

Clase

```
System.Object
└─ IBM.WMQ.MQPropertyDescriptor
```

```
public class IBM.WMQ.MQPropertyDescriptor extends System.Object;
```

- [“Propiedades” en la página 1778](#)
- [“Constructores” en la página 1779](#)

Propiedades

Prueba de MQException que se genera al obtener las propiedades.

```
public int Context {get; set;}
```

El contexto de mensaje al que pertenece la propiedad. Los valores posibles son:

MQC.MQPD_NO_CONTEXT

La propiedad no está asociada a un contexto de mensaje.

MQC.MQPD_USER_CONTEXT

La propiedad está asociada al contexto de usuario.

Si el usuario está autorizado, se guarda una propiedad asociada con el contexto de usuario cuando se recupera un mensaje. Un método Put posterior que haga referencia al contexto guardado, puede pasar la propiedad al nuevo mensaje.

```
public int CopyOptions {get; set;}
```

CopyOptions describe en qué tipo de mensaje se puede copiar la propiedad.

Cuando un gestor de colas recibe un mensaje que contiene una propiedad definida IBM MQ que el gestor de colas reconoce como incorrecta, el gestor de colas corrige el valor del campo CopyOptions.

Se puede especificar cualquier combinación de las opciones siguientes. Combine las opciones añadiendo los valores o utilizando OR a nivel de bit.

MQC.MQCOPY_ALL

La propiedad se copia en todos los tipos de mensajes posteriores.

MQC.MQCOPY_FORWARD

La propiedad se copia en un mensaje que se está reenviando.

MQC.MQCOPY_PUBLISH

La propiedad se copia en el mensaje recibido por un suscriptor cuando se publica un mensaje.

MQC.MQCOPY_REPLY

La propiedad se copia en un mensaje de respuesta.

MQC.MQCOPY_REPORT

La propiedad se copia en un mensaje de informe.

MQC.MQCOPY_DEFAULT

El valor ha indicado que no se ha especificado ninguna otra opción de copia. No existe ninguna relación entre la propiedad y los mensajes posteriores. MQC.MQCOPY_DEFAULT siempre se devuelve para las propiedades del descriptor de mensaje.

MQC.MQCOPY_NONE

Lo mismo que MQC.MQCOPY_DEFAULT

public int Options { get; set; }

Opciones toma como valor predeterminado CMQC.MQPD_NONE. No puede establecer ningún otro valor.

public int Support { get; set; }

Establezca Soporte para especificar el nivel de soporte necesario para las propiedades de mensaje definidas por IBM MQ. El soporte para todas las demás propiedades es opcional. Se puede especificar cualquiera o ninguno de los valores siguientes

MQC.MQPD_SUPPORT_OPTIONAL

Un gestor de colas acepta la propiedad aunque no esté soportada. La propiedad se puede descartar para que el mensaje fluya a un gestor de colas que no da soporte a las propiedades de mensaje. Este valor también se asigna a propiedades que no están definidas por IBM MQ.

MQC.MQPD_SUPPORT_REQUIRED

Se necesita soporte para la propiedad. Si coloca el mensaje en un gestor de colas que no da soporte a la propiedad definida por IBM MQ, el método falla. Devuelve el código de terminación MQC.MQCC_FAILED y el código de razón MQC.MQRC_UNSUPPORTED_PROPERTY.

MQC.MQPD_SUPPORT_REQUIRED_IF_LOCAL

Se necesita soporte para la propiedad, si el mensaje está destinado a una cola local. Si coloca el mensaje en una cola local en un gestor de colas que no da soporte a la propiedad definida por IBM MQ, el método falla. Devuelve el código de terminación MQC.MQCC_FAILED y el código de razón MQC.MQRC_UNSUPPORTED_PROPERTY.

No se realiza ninguna comprobación si el mensaje se coloca en un gestor de colas remoto.

Constructores

PropertyDescriptor();

Cree un descriptor de propiedad.

Clase MQPutMessageOptions.NET

Utilice MQPutMessageOptions para especificar cómo se envían los mensajes. Modifica el comportamiento de MQDestination.Put.

Clase

```
System.Object
├── IBM.WMQ.MQBase
│   └── IBM.WMQ.MQBaseObject
│       └── IBM.WMQ.MQPutMessageOptions
```

```
public class IBM.WMQ.MQPutMessageOptions extends IBM.WMQ.MQBaseObject;
```

- [“Propiedades” en la página 1780](#) [“Constructores” en la página 1782](#)

Propiedades

Prueba de `MQException` que se genera al obtener las propiedades.

Nota: El comportamiento de algunas de las opciones disponibles en esta clase depende del entorno en el que se utilizan. Estos elementos se marcan con un asterisco, *.

public MQQueue ContextReference {get; set;}

Si el campo `options` incluye `MQC.MQPMO_PASS_IDENTITY_CONTEXT` o `MQC.MQPMO_PASS_ALL_CONTEXT`, establezca este campo para que haga referencia al `MQQueue` del que se tomará la información de contexto.

El valor inicial de este campo es nulo.

public int InvalidDestCount {get;} *

Generalmente, utilizado para las listas de distribución, `InvalidDestCount` indica el número de mensajes que no se han podido enviar a las colas de una lista de distribución. El recuento incluye las colas que no se han podido abrir y también las colas que se han abierto correctamente, pero para las que la operación de colocación ha fallado.

.NET no admite listas de distribución, pero se establece `InvalidDestCount` al abrir una cola única.

public int KnownDestCount {get;} *

Generalmente utilizado para listas de distribución, `KnownDestCount` indica el número de mensajes que la llamada actual ha enviado con éxito a colas que resuelven colas locales.

.NET no admite listas de distribución, pero se establece `InvalidDestCount` al abrir una cola única.

public int Options {get; set;}

Opciones que controlan la acción de `MQDestination.put` y `MQQueueManager.put`. Se puede especificar cualquiera de los valores siguientes o ninguno de ellos. Si se necesita más de una opción, los valores se pueden añadir o combinar utilizando el operador OR a nivel de bit.

MQC.MQPMO_ASYNC_RESPONSE

Esta opción hace que la llamada `MQDestination.put` se realice de forma asíncrona, con algunos datos de respuesta.

MQC.MQPMO_DEFAULT_CONTEXT

Asocie el contexto predeterminado con el mensaje.

MQC.MQPMO_FAIL_IF QUIESCING

Fallar si el gestor de colas se está desactivando temporalmente.

MQC.MQPMO_LOGICAL_ORDER *

Coloque los mensajes lógicos y los segmentos de los grupos de mensajes en su orden lógico.

Si utiliza la opción `MQPMO_LOGICAL_ORDER` en un cliente reconectable, el código de razón `MQRC_RECONNECT_INCOMPATIBLE` se devuelve a la aplicación.

MQC.MQPMO_NEW_CORREL_ID *

Genere un nuevo ID de correlación para cada mensaje enviado.

MQC.MQPMO_NEW_MSG_ID *

Genere un nuevo ID de mensaje para cada mensaje enviado.

MQC.MQPMO_NONE

No se ha especificado ninguna opción. No lo utilice con otras opciones.

MQC.MQPMO_NO_CONTEXT

No se debe asociar ningún contexto con el mensaje.

MQC.MQPMO_NO_SYNCPOINT

Colocar un mensaje sin control de punto de sincronización. Si no se especifica la opción de control de punto de sincronización, se presupone un valor predeterminado de ningún punto de sincronización.

MQC.MQPMO_PASS_ALL_CONTEXT

Pasar todo el contexto desde un descriptor de contexto de cola de entrada.

MQC.MQPMO_PASS_IDENTITY_CONTEXT

Pasar contexto de identidad desde un descriptor de contexto de cola de entrada.

MQC.MQPMO_RESPONSE_AS_Q_DEF

Para una llamada `MQDestination.put`, esta opción toma el tipo de respuesta `put` del atributo `DEFPRESP` de la cola.

Para una llamada `MQQueueManager.put`, esta opción hace que la llamada se realice de forma síncrona.

MQC.MQPMO_RESPONSE_AS_TOPIC_DEF

`MQC.MQPMO_RESPONSE_AS_TOPIC_DEF` es un sinónimo de `MQC.MQPMO_RESPONSE_AS_Q_DEF` para su uso con objetos de tema.

MQC.MQPMO_RETAIN

El gestor de colas debe retener la publicación que se está enviando. Si se utiliza esta opción y la publicación no se puede retener, el mensaje no se publica y la llamada falla con `MQC.MQRC_PUT_NOT_RETAINED`.

Solicite una copia de esta publicación después de la hora en que se publicó, llamando al método `MQSubscription.RequestPublicationUpdate`. La publicación guardada se envía a las aplicaciones que crean una suscripción sin establecer la opción `MQC.MQSO_NEW_PUBLICATIONS_ONLY`. Compruebe la propiedad de mensaje `MQIsRetained` de una publicación, cuando se reciba, para averiguar si era la publicación retenida.

Cuando un suscriptor solicita las publicaciones retenidas, la suscripción utilizada puede contener un comodín en la serie de tema. Si hay varias publicaciones retenidas en el árbol de temas que coinciden con la suscripción, se envían todas.

MQC.MQPMO_SET_ALL_CONTEXT

Establezca todo el contexto de la aplicación.

MQC.MQPMO_SET_IDENTITY_CONTEXT

Establezca el contexto de identidad de la aplicación.

MQC.MQPMO_SYNC_RESPONSE

Esta opción hace que la llamada `MQDestination.put` o `MQQueueManager.put` se realice de forma síncrona, con datos de respuesta completos.

MQC.MQPMO_SUPPRESS_REPLYTO

La información introducida en los campos `ReplyToQueueName` y `ReplyToQueueManagerName` de la publicación no se transmite a los suscriptores. Si esta opción se utiliza en combinación con una opción de informe que requiere un `ReplyToQueueName`, la llamada falla con '`MQC.MQRC_MISSING_REPLY_TO_Q`'.

MQC.MQPMO_SYNCPOINT

Colocar un mensaje con control de punto de sincronización. El mensaje no está visible fuera de la unidad de trabajo hasta que se confirme la unidad de trabajo. Si la unidad de trabajo se restituye, se suprime el mensaje.

public int RecordFields {get; set;} *

Información sobre listas de distribución. Las listas de distribución no son compatibles con .NET.

public string ResolvedQueueManagerName {get;}

Campo de salida establecido por el gestor de colas en el nombre del gestor de colas que es propietario de la cola especificada por el nombre de cola remota. `ResolvedQueueManagerName` puede ser diferente del nombre del gestor de colas desde el que se accedió a la cola si se trata de una cola remota.

Sólo se devuelve un valor no en blanco si el objeto es una sola cola. Si el objeto es una lista de distribución o un tema, el valor devuelto no está definido.

public string ResolvedQueueName {get;}

Campo de salida establecido por el gestor de colas en el nombre de la cola en la que se coloca el mensaje. ResolvedQueueName puede ser diferente del nombre utilizado para abrir la cola si la cola abierta era un alias o una cola modelo.

Sólo se devuelve un valor que no esté en blanco si el objeto es una sola cola. Si el objeto es una lista de distribución o un tema, el valor devuelto no está definido.

public int UnknownDestCount {get;} *

Generalmente utilizado para listas de distribución, UnknownDestCount es un campo de salida establecido por el gestor de colas. Informa del número de mensajes que la llamada actual ha enviado satisfactoriamente a las colas que se resuelven en colas remotas.

.NET no admite listas de distribución, pero se establece InvalidDestCount al abrir una cola única.

Constructores

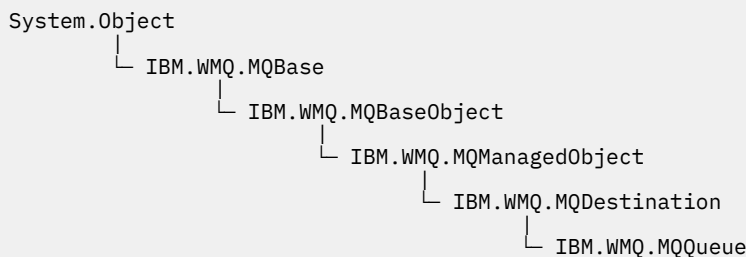
public MQPutMessageOptions();

Construye un nuevo objeto " MQPutMessageOptions " sin opciones y con ResolvedQueueName y ResolvedQueueManagerName en blanco.

MQQueue.NET clase

Utilízelo MQQueue para enviar y recibir mensajes, y consultar los atributos de una IBM MQ cola. Crea un MQQueue objeto utilizando un constructor o un MQQueueManager . AccessProcess método.

Clase



```
public class IBM.WMQ.MQQueue extends IBM.WMQ.MQDestination;
```

- [“Propiedades” en la página 1782](#)
- [“Métodos” en la página 1784](#)
- [“Constructores” en la página 1787](#)

Propiedades

Prueba de MQException que se genera al obtener las propiedades.

public int ClusterWorkLoadPriority {get;}

Especifica la prioridad de la cola. Este parámetro solo es válido para colas locales, remotas y alias.

public int ClusterWorkLoadRank {get;}

Especifica el rango de la cola. Este parámetro solo es válido para colas locales, remotas y alias.

public int ClusterWorkLoadUseQ {get;}

Especifica el comportamiento de una operación MQPUT cuando la cola de destino tiene una instancia local y al menos una instancia de clúster remota. Este parámetro no se aplica si MQPUT se origina en un canal de clúster. Este parámetro solo es válido para colas locales.

public DateTime CreationDateTime {get;}

La fecha y hora en que se creó esta cola.

public int CurrentDepth {get;}

Obtiene el número de mensajes que hay actualmente en la cola. Este valor se incrementa durante una llamada put y durante la retirada de una llamada get. Se decrementa durante una obtención sin exploración y durante la retirada de una llamada de inserción.

public int DefinitionType {get;}

Cómo se definió la cola. Los valores posibles son:

- MQC.MQQDT_PREDEFINED
- MQC.MQQDT_PERMANENT_DYNAMIC
- MQC.MQQDT_TEMPORARY_DYNAMIC

public int InhibitGet {get; set;}

Controla si puedes recibir mensajes en esta cola o para este tema. Los valores posibles son:

- MQC.MQQA_GET_INHIBITED
- MQC.MQQA_GET_ALLOWED

public int InhibitPut {get; set;}

Controla si se pueden enviar mensajes a esta cola o para este tema. Los valores posibles son:

- MQQA_PUT_INHIBITED
- MQQA_PUT_ALLOWED

public int MaximumDepth {get;}

El número máximo de mensajes que pueden existir en la cola en un momento dado. El intento de colocar un mensaje en una cola que ya contiene tantos mensajes falla con el código de motivo MQC.MQRC_Q_FULL.

public int MaximumMessageLength {get;}

La longitud máxima de los datos de la aplicación que pueden existir en cada mensaje de esta cola. Un intento de incluir un mensaje más grande que este valor falla con el código de motivo MQC.MQRC_MSG_TOO_BIG_FOR_Q.

public int NonPersistentMessageClass {get;}

El nivel de fiabilidad para los mensajes no permanentes colocados en esta cola.

public int OpenInputCount {get;}

El número de identificadores que son válidos actualmente para eliminar mensajes de la cola. OpenInputCount es el número total de identificadores de entrada válidos conocidos por el gestor de colas local, no solo los identificadores creados por la aplicación.

public int OpenOutputCount {get;}

El número de identificadores que son válidos actualmente para añadir mensajes a la cola. OpenOutputCount es el número total de identificadores de salida válidos conocidos por el gestor de colas local, no solo los identificadores creados por la aplicación.

public int QueueAccounting {get;}

Especifica si se puede habilitar la recopilación de información contable para la cola.

public int QueueMonitoring {get;}

Especifica si se puede habilitar la supervisión de la cola.

public int QueueStatistics {get;}

Especifica si se puede habilitar la recopilación de estadísticas para la cola.

public int QueueType {get;}

El tipo de esta cola con uno de los siguientes valores:

- MQC.MQQT_ALIAS
- MQC.MQQT_LOCAL
- MQC.MQQT_REMOTE

- MQC.MQQT_CLUSTER

public int Shareability {get;}

Si la cola se puede abrir para introducir datos varias veces. Los valores posibles son:

- MQC.MQQA_SHAREABLE
- MQC.MQQA_NOT_SHAREABLE

public string TPIPE {get;}

El TPIPE nombre utilizado para la comunicación con OTMA mediante el puente IBM MQIMS.

public int TriggerControl {get; set;}

Si los mensajes de activación se escriben en una cola de inicio, para iniciar una aplicación que dé servicio a la cola. Los valores posibles son:

- MQC.MQTC_OFF
- MQC.MQTC_ON

public string TriggerData {get; set;}

Los datos de formato libre que el gestor de colas inserta en el mensaje de activación. Se inserta TriggerData cuando un mensaje que llega a esta cola provoca que se escriba un mensaje de activación en la cola de inicio. La longitud máxima permitida de la cadena viene dada por MQC.MQ_TRIGGER_DATA_LENGTH.

public int TriggerDepth {get; set;}

El número de mensajes que deben estar en la cola antes de que se escriba un mensaje de activación cuando el tipo de activación se establece en MQC.MQTT_DEPTH.

public int TriggerMessagePriority {get; set;}

La prioridad de los mensajes según la cual los mensajes no contribuyen a la generación de mensajes desencadenantes. Es decir, el gestor de colas ignora estos mensajes al decidir si generar un disparador. Un valor de cero hace que todos los mensajes contribuyan a la generación de mensajes de activación.

public int TriggerType {get; set;}

Las condiciones en las que se escriben los mensajes desencadenantes como resultado de los mensajes que llegan a esta cola. Los valores posibles son:

- MQC.MQTT_NONE
- MQC.MQTT_FIRST
- MQC.MQTT EVERY
- MQC.MQTT_DEPTH

Métodos

public void Get(MQMessage message);

public void Get(MQMessage message, MQGetMessageOptions getMessageOptions);

public void Get(MQMessage message, MQGetMessageOptions getMessageOptions, int MaxMsgSize);

Lanza MQException.

Obtiene un mensaje de una cola.

Si la obtención falla, el objeto MQMessage no se modifica. Si se ejecuta correctamente, el descriptor de mensaje y las partes de datos de mensaje del MQMessage se sustituyen por el descriptor de mensaje y los datos de mensaje del mensaje de entrada.

Todas las llamadas a IBM MQ desde un MQQueueManager determinado son síncronas. Por lo tanto, si realiza una operación get con espera, todas las demás hebras que utilizan el mismo MQQueueManager no pueden realizar más llamadas IBM MQ hasta que se realice la llamada Get. Si necesita varias hebras para acceder a IBM MQ simultáneamente, cada hebra debe crear su propio objeto MQQueueManager.

mensaje

Contiene el descriptor de mensaje y los datos de mensaje devueltos. Algunos de los campos del descriptor de mensaje son parámetros de entrada. Es importante asegurarse de que los parámetros de entrada MessageId y CorrelationId estén establecidos según sea necesario.

Un cliente reconectable devuelve el código de razón MQRC_BACKED_OUT después de una reconexión satisfactoria, para los mensajes recibidos en MQGM_SYNCPOINT.

getMessageOptions

Opciones que controlan la acción de la obtención.

El uso de la opción MQC.MQGMO_CONVERT puede dar como resultado una excepción con el código de razón MQC.MQRC_CONVERTED_STRING_TOO_BIG al convertir de códigos de caracteres de un solo byte a códigos de doble byte. En este caso, el mensaje se copia en el almacenamiento intermedio sin conversión.

Si no se especifica *getMessageOptions*, la opción de mensaje utilizada es MQGM_NOWAIT.

Si utiliza la opción MQGMO_LOGICAL_ORDER en un cliente reconectable, se devuelve el código de razón MQRC_RECONNECT_INCOMPATIBLE.

MaxMsgSize

El mensaje más grande que este objeto de mensaje va a recibir. Si el mensaje en la cola es mayor que este tamaño, se produce una de las dos cosas siguientes:

- Si el distintivo MQGMO_ACCEPT_TRUNCATED_MSG se establece en el objeto MQGetMessageOptions, el mensaje se rellena con la mayor cantidad posible de datos de mensaje. Se genera una excepción con el código de terminación MQCC_WARNING y el código de razón MQRC_TRUNCATED_MSG_ACCEPTED.
- Si el distintivo MQGMO_ACCEPT_TRUNCATED_MSG no está establecido, el mensaje permanece en la cola. Se genera una excepción con el código de terminación MQCC_WARNING y el código de razón MQRC_TRUNCATED_MSG_FAILED.

Si no se especifica *MaxMsgSize*, se recupera todo el mensaje.

```
public void Put(MQMessage message);  
public void Put(MQMessage message, MQPutMessageOptions putMessageOptions);
```

Lanza MQException.

Coloca un mensaje en una cola.

Las modificaciones en el objeto MQMessage después de que se haya realizado la llamada Put no afectan al mensaje real en el tema de publicación o cola IBM MQ.

Put actualiza las propiedades MessageId y CorrelationId del objeto MQMessage y no borra los datos del mensaje. Las llamadas Put o Get adicionales hacen referencia a la información actualizada en el objeto MQMessage. Por ejemplo, en el siguiente fragmento de código, el primer mensaje contiene a y el segundo ab.

```
msg.WriteString("a");  
q.Put(msg, pmo);  
msg.WriteString("b");  
q.Put(msg, pmo);
```

mensaje

Un objeto MQMessage que contiene los datos del descriptor de mensaje y el mensaje que se va a enviar. El descriptor de mensaje se puede modificar como consecuencia de este método. Los valores del descriptor de mensaje inmediatamente después de la finalización de este método son los valores que se colocaron en la cola o se publicaron en el tema.

Se devuelven los siguientes códigos de razón a un cliente reconectable:

- MQRC_CALL_INTERRUPTED si la conexión se interrumpe al ejecutar una llamada Put en un mensaje persistente y la reconexión se realiza correctamente.

- MQRC_NONE si la conexión es satisfactoria al ejecutar una llamada Put en un mensaje no persistente (consulte [Recuperación de aplicaciones](#)).

putMessageOptions

Opciones que controlan la acción de la colocación.

Si no se especifica *putMessageOptions* , se utiliza la instancia predeterminada de MQPutMessageOptions .

Si utiliza la opción MQPMO_LOGICAL_ORDER en un cliente reconectable, se devuelve el código de razón MQRC_RECONNECT_INCOMPATIBLE .

Nota: Por motivos de simplicidad y rendimiento, si desea transferir un único mensaje a una cola, utilice el objeto MQQueueManager .Put . Debe tener un objeto MQQueue para esto.

```
public void PutForwardMessage(MQMessage message);  
public void PutForwardMessage(MQMessage message, MQPutMessageOptions  
putMessageOptions);
```

Genera MQException

Coloca un mensaje reenviado en la cola, donde *message* es el mensaje original.

mensaje

Un objeto MQMessage que contiene los datos del descriptor de mensaje y el mensaje que se va a enviar. El descriptor de mensaje se puede modificar como consecuencia de este método. Los valores del descriptor de mensaje inmediatamente después de la finalización de este método son los valores que se colocaron en la cola o se publicaron en el tema.

Se devuelven los siguientes códigos de razón a un cliente reconectable:

- MQRC_CALL_INTERRUPTED si la conexión se interrumpe al ejecutar una llamada Put en un mensaje persistente y la reconexión se realiza correctamente.
- MQRC_NONE si la conexión es satisfactoria al ejecutar una llamada Put en un mensaje no persistente (consulte [Recuperación de aplicaciones](#)).

putMessageOptions

Opciones que controlan la acción de la colocación.

Si no se especifica *putMessageOptions* , se utiliza la instancia predeterminada de MQPutMessageOptions .

Si utiliza la opción MQPMO_LOGICAL_ORDER en un cliente reconectable, se devuelve el código de razón MQRC_RECONNECT_INCOMPATIBLE .

```
public void PutReplyMessage(MQMessage message)  
public void PutReplyMessage(MQMessage message, MQPutMessageOptions  
putMessageOptions)
```

Lanza MQException.

Coloca un mensaje de respuesta en la cola, donde *message* es el mensaje original.

mensaje

Contiene el descriptor de mensaje y los datos de mensaje devueltos. Algunos de los campos del descriptor de mensaje son parámetros de entrada. Es importante asegurarse de que los parámetros de entrada MessageId y CorrelationId estén establecidos según sea necesario.

Un cliente reconectable devuelve el código de razón MQRC_BACKED_OUT después de una reconexión satisfactoria, para los mensajes recibidos en MQGM_SYNCPOINT.

putMessageOptions

Opciones que controlan la acción de la colocación.

Si no se especifica *putMessageOptions* , se utiliza la instancia predeterminada de MQPutMessageOptions .

Si utiliza la opción MQPMO_LOGICAL_ORDER en un cliente reconectable, se devuelve el código de razón MQRC_RECONNECT_INCOMPATIBLE .

```
public void PutReportMessage(MQMessage message)  
public void PutReportMessage(MQMessage message, MQPutMessageOptions  
putMessageOptions)
```

Lanza MQException.

Coloca un mensaje de informe en la cola, donde *message* es el mensaje original.

mensaje

Contiene el descriptor de mensaje y los datos de mensaje devueltos. Algunos de los campos del descriptor de mensaje son parámetros de entrada. Es importante asegurarse de que los parámetros de entrada MessageId y CorrelationId estén establecidos según sea necesario.

Un cliente reconectable devuelve el código de razón MQRC_BACKED_OUT después de una reconexión satisfactoria, para los mensajes recibidos en MQGM_SYNCPOINT.

putMessageOptions

Opciones que controlan la acción de la colocación.

Si no se especifica *putMessageOptions*, se utiliza la instancia predeterminada de MQPutMessageOptions.

Si utiliza la opción MQPMO_LOGICAL_ORDER en un cliente reconectable, se devuelve el código de razón MQRC_RECONNECT_INCOMPATIBLE.

Constructores

```
public MQQueue MQQueueManager.AccessQueue(string queueName, int openOptions);  
public MQQueue MQQueueManager.AccessQueue(string queueName, int openOptions,  
string queueManagerName, string dynamicQueueName, string alternateUserId);
```

Lanza MQException.

Accede a una cola en este gestor de colas.

Puede obtener o examinar mensajes, enviar mensajes, consultar los atributos de la cola o establecer los atributos de la cola. Si la cola nombrada es una cola modelo, se crea una cola local dinámica. Consulte el name atributo del objeto MQQueue resultante para averiguar el nombre de la cola dinámica.

queueName

Nombre de la cola que se va a abrir.

openOptions

Opciones que controlan la apertura de la cola.

MQC.MQOO_ALTERNATE_USER_AUTHORITY

Validar con el identificador de usuario especificado.

MQC.MQOO_BIND_AS_QDEF

Utilizar el enlace predeterminado para la cola.

MQC.MQOO_BIND_NOT_FIXED

No se vincule a un destino específico.

MQC.MQOO_BIND_ON_OPEN

Vincular el identificador al destino cuando se abre la cola.

MQC.MQOO_BROWSE

Abrir para leer el mensaje.

MQC.MQOO_FAIL_IF QUIESCING

Falla si el gestor de colas está inactivo.

MQC.MQOO_INPUT_AS_Q_DEF

Abrir para obtener mensajes utilizando el valor predeterminado definido en la cola.

MQC.MQOO_INPUT_SHARED

Abrir para obtener mensajes con acceso compartido.

MQC.MQOO_INPUT_EXCLUSIVE

Abre para recibir mensajes con acceso exclusivo.

MQC.MQOO_INQUIRE

Abierto para consultas: necesario si desea consultar propiedades.

MQC.MQOO_OUTPUT

Abierto para poner mensajes.

MQC.MQOO_PASS_ALL_CONTEXT

Permitir que se pase todo el contexto.

MQC.MQOO_PASS_IDENTITY_CONTEXT

Permitir que se transmita el contexto de identidad.

MQC.MQOO_SAVE_ALL_CONTEXT

Guardar el contexto al recuperar el mensaje.

MQC.MQOO_SET

Abrir para establecer atributos: necesario si desea establecer propiedades.

MQC.MQOO_SET_ALL_CONTEXT

Permite establecer todo el contexto.

MQC.MQOO_SET_IDENTITY_CONTEXT

Permite establecer el contexto de identidad.

queueManagerName

Nombre del gestor de colas en el que se define la cola. Un nombre completamente en blanco o nulo denota el gestor de colas al que está conectado el MQQueueManager objeto.

dynamicQueueName

dynamicQueueName se ignora a menos que *queueName* especifique el nombre de una cola de modelos. Si es así, *dynamicQueueName* especifica el nombre de la cola dinámica que se va a crear. Un nombre en blanco o nulo no es válido si *queueName* especifica el nombre de una cola de modelo. Si el último carácter no en blanco del nombre es un asterisco, * el gestor de colas sustituye el asterisco por una cadena de caracteres. Los caracteres garantizan que el nombre generado para la cola sea único en este gestor de colas.

alternateUserId

Si MQC.MQOO_ALTERNATE_USER_AUTHORITY se especifica en el *openOptions* parámetro, *alternateUserId* especifica el identificador de usuario alternativo que se utiliza para comprobar la autorización para la apertura. Si no MQC.MQOO_ALTERNATE_USER_AUTHORITY se especifica, *alternateUserId* se puede dejar en blanco o nulo.

```
public MQQueue(MQQueueManager queueManager, string queueName, int openOptions,  
string queueManagerName, string dynamicQueueName, string alternateUserId);
```

Lanza MQException.

Accede a una cola en *queueManager*.

Puede obtener o examinar mensajes, enviar mensajes, consultar los atributos de la cola o establecer los atributos de la cola. Si la cola nombrada es una cola modelo, se crea una cola local dinámica. Consulte el *name* atributo del objeto MQQueue resultante para averiguar el nombre de la cola dinámica.

queueManager

Administrador de colas para acceder a la cola.

queueName

Nombre de la cola que se va a abrir.

openOptions

Opciones que controlan la apertura de la cola.

MQC.MQOO_ALTERNATE_USER_AUTHORITY

Validar con el identificador de usuario especificado.

MQC.MQOO_BIND_AS_QDEF

Utilizar el enlace predeterminado para la cola.

MQC.MQOO_BIND_NOT_FIXED

No se vincule a un destino específico.

MQC.MQOO_BIND_ON_OPEN

Vincular el identificador al destino cuando se abre la cola.

MQC.MQOO_BROWSE

Abrir para leer el mensaje.

MQC.MQOO_FAIL_IF QUIESCING

Falla si el gestor de colas está inactivo.

MQC.MQOO_INPUT_AS_Q_DEF

Abrir para obtener mensajes utilizando el valor predeterminado definido en la cola.

MQC.MQOO_INPUT_SHARED

Abrir para obtener mensajes con acceso compartido.

MQC.MQOO_INPUT_EXCLUSIVE

Abre para recibir mensajes con acceso exclusivo.

MQC.MQOO_INQUIRE

Abierto para consultas: necesario si desea consultar propiedades.

MQC.MQOO_OUTPUT

Abierto para poner mensajes.

MQC.MQOO_PASS_ALL_CONTEXT

Permitir que se pase todo el contexto.

MQC.MQOO_PASS_IDENTITY_CONTEXT

Permitir que se transmita el contexto de identidad.

MQC.MQOO_SAVE_ALL_CONTEXT

Guardar el contexto al recuperar el mensaje.

MQC.MQOO_SET

Abrir para establecer atributos: necesario si desea establecer propiedades.

MQC.MQOO_SET_ALL_CONTEXT

Permite establecer todo el contexto.

MQC.MQOO_SET_IDENTITY_CONTEXT

Permite establecer el contexto de identidad.

queueManagerName

Nombre del gestor de colas en el que se define la cola. Un nombre completamente en blanco o nulo denota el gestor de colas al que está conectado el MQQueueManager objeto.

dynamicQueueName

dynamicQueueName se ignora a menos que *queueName* especifique el nombre de una cola de modelos. Si es así, *dynamicQueueName* especifica el nombre de la cola dinámica que se va a crear. Un nombre en blanco o nulo no es válido si *queueName* especifica el nombre de una cola de modelo. Si el último carácter no en blanco del nombre es un asterisco, * el gestor de colas sustituye el asterisco por una cadena de caracteres. Los caracteres garantizan que el nombre generado para la cola sea único en este gestor de colas.

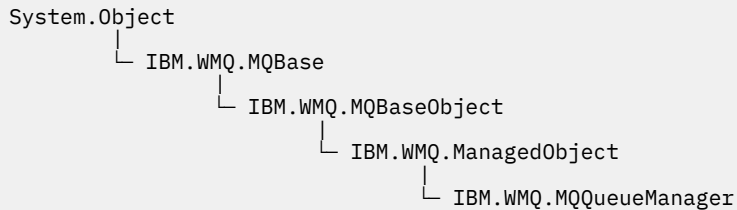
alternateUserId

Si MQC.MQOO_ALTERNATE_USER_AUTHORITY se especifica en el *openOptions* parámetro, *alternateUserId* especifica el identificador de usuario alternativo que se utiliza para comprobar la autorización para la apertura. Si no MQC.MQOO_ALTERNATE_USER_AUTHORITY se especifica, *alternateUserId* se puede dejar en blanco o nulo.

MQQueueManager.NET clase

Utilice MQQueueManager para conectarse a un gestor de colas y acceder a los objetos del gestor de colas. También controla las transacciones. El MQQueueManager constructor crea una conexión de cliente o de servidor.

Clase



```
public class IBM.WMQ.MQQueueManager extends IBM.WMQ.MQManagedObject;
```

- [“Propiedades” en la página 1790](#)
- [“Métodos” en la página 1793](#)
- [“Constructores” en la página 1799](#)

Propiedades

Prueba de MQException que se genera al obtener las propiedades.

public int AccountingConnOverride {get;}

Si las aplicaciones pueden anular la configuración de los valores de contabilidad de « MQI » y de contabilidad de colas.

public int AccountingInterval {get;}

Cuánto tiempo transcurre antes de que se registren los registros contables intermedios (en segundos).

public int ActivityRecording {get;}

Controla la generación de los informes de actividades.

public int AdoptNewMCACheck {get;}

Especifica qué elementos se comprueban para determinar si se adopta el MCA cuando se detecta un nuevo canal entrante. Para ser adoptado, el nombre MCA debe coincidir con el nombre de un MCA activo.

public int AdoptNewMCAInterval {get;}

El tiempo, en segundos, que el nuevo canal espera a que finalice el canal huérfano.

public int AdoptNewMCAType {get;}

Si una instancia MCA huérfana debe adoptarse (reiniciarse) cuando se detecta una nueva solicitud de canal entrante que coincide con el valor AdoptNewMCACheck.

public int BridgeEvent {get;}

Si se generan eventos de puent IMS.

public int ChannelEvent {get;}

Si se generan eventos de canal.

public int ChannelInitiatorControl {get;}

Si el iniciador del canal se inicia automáticamente cuando se inicia el gestor de colas.

public int ChannelInitiatorAdapters {get;}

El número de subtareas del adaptador para procesar IBM MQ llamadas.

public int ChannelInitiatorDispatchers {get;}

El número de despachadores que se utilizarán para el iniciador del canal.

public int ChannelInitiatorTraceAutoStart {get;}
Especifica si el seguimiento del iniciador del canal se inicia automáticamente.

public int ChannelInitiatorTraceTableSize {get;}
El tamaño, en megabytes, del espacio de datos de rastreo de un iniciador de canal.

public int ChannelMonitoring {get;}
Si se utiliza la supervisión de canales.

public int ChannelStatistics {get;}
Controla la recopilación de los datos estadísticos para los canales.

public int CharacterSet {get;}
Devuelve el identificador del conjunto de caracteres codificados (CCSID) del gestor de colas. CharacterSet Es utilizado por el gestor de colas para todos los campos de cadena de caracteres en la interfaz de programación de aplicaciones.

public int ClusterSenderMonitoring {get;}
Controla la recopilación de datos de supervisión en línea para canales emisores de clúster definidos automáticamente.

public int ClusterSenderStatistics {get;}
Controla la recopilación de datos estadísticos para los canales de envío de clústeres definidos automáticamente.

public int ClusterWorkLoadMRU {get;}
El número máximo de canales de clúster salientes.

public int ClusterWorkLoadUseQ {get;}
El valor predeterminado de la MQQueue propiedad ClusterWorkLoadUseQ, si especifica un valor de QMGR.

public int CommandEvent {get;}
Especifica si se generan sucesos de mandato.

public string CommandInputQueueName {get;}
Devuelve el nombre de la cola de entrada de comandos definida en el gestor de colas. Las aplicaciones pueden enviar comandos a esta cola, si están autorizadas para ello.

public int CommandLevel {get;}
Indica el nivel de función del gestor de colas. El conjunto de funciones que corresponden a un nivel de función concreto depende de la plataforma. En una plataforma concreta, puede confiar en que todos los gestores de colas admitirán las funciones del nivel funcional más bajo común a todos los gestores de colas.

public int CommandLevel {get;}
Si el servidor de comandos se inicia automáticamente cuando se inicia el gestor de colas.

public string DNSGroup {get;}
Ya no se utiliza.

public int DNSWLM {get;}
Ya no se utiliza.

public int IPAddressVersion {get;}
¿Qué protocolo IP (IPv4 o IPv6) se debe utilizar para una conexión de canal?

public boolean IsConnected {get;}
Devuelve el valor de `isConnected`

Si es cierto, se ha establecido una conexión con el gestor de colas y no se sabe si se ha interrumpido. Las llamadas a `IsConnected` no intentan activamente comunicarse con el gestor de colas, por lo que es posible que la conectividad física se interrumpa, pero `IsConnected` puede seguir devolviendo el valor `true`. El estado « `IsConnected` » solo se actualiza cuando se realiza una actividad, por ejemplo, enviar un mensaje o recibir un mensaje, en el gestor de colas.

Si es falso, no se ha establecido una conexión con el gestor de colas, o se ha interrumpido o desconectado.

public int KeepAlive {get;}

Especifica si se debe utilizar la función KEEPALIVE de TCP para comprobar que el otro extremo de la conexión sigue estando disponible. Si no está disponible, el canal está cerrado.

public int ListenerTimer {get;}

El intervalo de tiempo, en segundos, entre los intentos de IBM MQ reiniciar el listener tras un fallo de « APPC » o « TCP/IP ».

public int LoggerEvent {get;}

Si se generan eventos de registro.

public string LU62ARMSuffix {get;}

El sufijo del miembro APPCPM de SYS1.PARMLIB. Este sufijo designa el LUADD de este iniciador de canal. Cuando el gestor de reinicio automático (ARM) reinicia el iniciador de canal, se ejecuta el z/OS comando SET APPC =xx.

public string LUGroupName {get; z/os}

El nombre genérico de la unidad lógica (LU) que utilizará el listener de la unidad lógica (6.2) que gestiona las transmisiones entrantes para el grupo de uso compartido de colas.

public string LUName {get;}

El nombre de la LU que se utilizará para las transmisiones LU 6.2 salientes.

public int MaximumActiveChannels {get;}

Número máximo de canales que pueden estar activos en cualquier momento.

public int MaximumCurrentChannels {get;}

El número máximo de canales que pueden estar activos en cualquier momento (incluidos los canales de conexión al servidor con clientes conectados).

public int MaximumLU62Channels {get;}

El número máximo de canales que pueden estar activos, o clientes que pueden estar conectados, que utilizan el protocolo de transmisión LU 6.2.

public int MaximumMessageLength {get;}

Devuelve la longitud máxima de un mensaje (en bytes) que puede gestionar el gestor de colas. No se puede definir ninguna cola con una longitud máxima de mensaje superior a MaximumMessageLength.

public int MaximumPriority {get;}

Devuelve la prioridad máxima de mensajes admitida por el gestor de colas. Las prioridades van desde cero (la más baja) hasta este valor. Lanza MQException si se llama a este método después de desconectarse del gestor de colas.

public int MaximumTCPChannels {get;}

El número máximo de canales activos, o de clientes conectados, que utilizan el protocolo de transmisión « TCP/IP ».

public int MQIAccounting {get;}

Controla la recopilación de la información contable para los datos de la Interfaz de Colas de Mensajes (MQI).

public int MQIStatistics {get;}

Controla la recopilación de la información sobre la supervisión de estadísticas para el gestor de colas.

public int OutboundPortMax {get;}

El valor máximo en el rango de números de puerto que se utilizará al vincular canales salientes.

public int OutboundPortMin {get;}

El valor mínimo en el rango de números de puerto que se utilizará al vincular canales salientes.

public int QueueAccounting {get;}

Si los datos de contabilidad de clase 3 (contabilidad a nivel de subproceso y a nivel de cola) se deben utilizar para todas las colas.

public int QueueMonitoring {get;}

Controla la recopilación de los datos de supervisión para las colas.

public int QueueStatistics {get;}

Controla la recopilación de los datos estadísticos para las colas.

public int ReceiveTimeout {get;}

El tiempo que un canal « TCP/IP » espera a recibir datos, incluidos los pulsos de control, de su interlocutor antes de volver al estado inactivo.

public int ReceiveTimeoutMin {get;}

El tiempo mínimo que un canal « TCP/IP » espera para recibir datos, incluidos los pulsos de control, de su interlocutor antes de volver a un estado inactivo.

public int ReceiveTimeoutType {get;}

El calificador que se aplica al valor en ReceiveTimeout.

public int SharedQueueQueueManagerName {get;}

Especifica cómo entregar mensajes a una cola compartida. Si la orden especifica un gestor de colas diferente del mismo grupo de intercambio de colas que el gestor de colas de destino, el mensaje se entrega de dos maneras:

MQC.MQSQQM_USE

Los mensajes se envían al gestor de colas de objetos antes de colocarse en la cola compartida.

MQCMQSQQM_IGNORE

Los mensajes se colocan directamente en la cola compartida.

public int SSLEvent {get;}

Si se generan eventos « TLS ».

public int SSLFips {get;}

Si solo deben FIPS utilizarse algoritmos certificados cuando se realiza criptografía en IBM MQ, en lugar de hardware criptográfico.

public int SSLKeyResetCount {get;}

Indica el número de bytes sin cifrar enviados y recibidos en una conversación TLS antes de que se renegocie la clave secreta.

public int ClusterSenderStatistics {get;}

Especifica el intervalo, en minutos, entre recopilaciones consecutivas de estadísticas.

public int SyncpointAvailability {get;}

Indica si el gestor de colas admite unidades de trabajo y puntos de sincronización con los MQQueue.get métodos MQQueue.put y.

public string TCPName {get;}

El nombre del único sistema de « TCP/IP » que se utilizará, o del predeterminado, dependiendo del valor de « TCPStackType ».

public int TCPStackType {get;}

Especifica si el iniciador del canal utiliza únicamente el espacio de direcciones TCP/IP especificado en TCPName. Como alternativa, el iniciador del canal puede vincularse a cualquier dirección TCP/IP.

public int TraceRouteRecording {get;}

Controla el registro de la información de seguimiento de la ruta.

Métodos

public MQProcess AccessProcess(string processName, int openOptions);

public MQProcess AccessProcess(string processName, int openOptions, string queueManagerName, string alternateUserId);

Genera MQException.

Acceda a un proceso de IBM MQ en este gestor de colas para consultar los atributos de proceso.

processName

El nombre del proceso que se va a abrir.

openOptions

Opciones que controlan la apertura del proceso. Las opciones válidas que se pueden añadir o combinar utilizando un OR a nivel de bit son:

- MQC.MQ00_FAIL_IF QUIESCING

- MQC.MQOO_INQUIRE
- MQC.MQOO_SET
- MQC.MQOO_ALTERNATE_USER_AUTHORITY

queueManagerName

El nombre del gestor de colas en el que se define el proceso. Puede dejar un nombre de gestor de colas en blanco o nulo si el gestor de colas es el mismo que al que está accediendo el proceso.

alternateUserId

Si se especifica MQC.MQOO_ALTERNATE_USER_AUTHORITY en el parámetro **openOptions**, *alternateUserId* especifica el ID de usuario alternativo utilizado para comprobar la autorización de la acción. Si no se especifica MQOO_ALTERNATE_USER_AUTHORITY, *alternateUserId* puede estar en blanco o ser nulo.

Se utiliza la autorización de usuario predeterminada para la conexión con el gestor de colas si no se especifica MQC.MQOO_ALTERNATE_USER_AUTHORITY.

```
public MQQueue AccessQueue(string queueName, int openOptions);  
public MQQueue AccessQueue(string queueName, int openOptions, string  
queueManagerName, string dynamicQueueName, string alternateUserId);
```

Lanza MQException.

Accede a una cola en este gestor de colas.

Puede obtener o examinar mensajes, enviar mensajes, consultar los atributos de la cola o establecer los atributos de la cola. Si la cola nombrada es una cola modelo, se crea una cola local dinámica. Consulte el name atributo del objeto MQQueue resultante para averiguar el nombre de la cola dinámica.

queueName

Nombre de la cola que se va a abrir.

openOptions

Opciones que controlan la apertura de la cola.

MQC.MQOO_ALTERNATE_USER_AUTHORITY

Validar con el identificador de usuario especificado.

MQC.MQOO_BIND_AS_QDEF

Utilizar el enlace predeterminado para la cola.

MQC.MQOO_BIND_NOT_FIXED

No se vincule a un destino específico.

MQC.MQOO_BIND_ON_OPEN

Vincular el identificador al destino cuando se abre la cola.

MQC.MQOO_BROWSE

Abrir para leer el mensaje.

MQC.MQOO_FAIL_IF QUIESCING

Falla si el gestor de colas está inactivo.

MQC.MQOO_INPUT_AS_Q_DEF

Abrir para obtener mensajes utilizando el valor predeterminado definido en la cola.

MQC.MQOO_INPUT_SHARED

Abrir para obtener mensajes con acceso compartido.

MQC.MQOO_INPUT_EXCLUSIVE

Abre para recibir mensajes con acceso exclusivo.

MQC.MQOO_INQUIRE

Abierto para consultas: necesario si desea consultar propiedades.

MQC.MQOO_OUTPUT

Abierto para poner mensajes.

MQC.MQOO_PASS_ALL_CONTEXT

Permitir que se pase todo el contexto.

MQC.MQOO_PASS_IDENTITY_CONTEXT

Permitir que se transmita el contexto de identidad.

MQC.MQOO_SAVE_ALL_CONTEXT

Guardar el contexto al recuperar el mensaje.

MQC.MQOO_SET

Abrir para establecer atributos: necesario si desea establecer propiedades.

MQC.MQOO_SET_ALL_CONTEXT

Permite establecer todo el contexto.

MQC.MQOO_SET_IDENTITY_CONTEXT

Permite establecer el contexto de identidad.

queueManagerName

Nombre del gestor de colas en el que se define la cola. Un nombre completamente en blanco o nulo denota el gestor de colas al que está conectado el MQQueueManager objeto.

dynamicQueueName

dynamicQueueName se ignora a menos que *queueName* especifique el nombre de una cola de modelos. Si es así, *dynamicQueueName* especifica el nombre de la cola dinámica que se va a crear. Un nombre en blanco o nulo no es válido si *queueName* especifica el nombre de una cola de modelo. Si el último carácter no en blanco del nombre es un asterisco, * el gestor de colas sustituye el asterisco por una cadena de caracteres. Los caracteres garantizan que el nombre generado para la cola sea único en este gestor de colas.

alternateUserId

Si MQC.MQOO_ALTERNATE_USER_AUTHORITY se especifica en el openOptions parámetro, *alternateUserId* especifica el identificador de usuario alternativo que se utiliza para comprobar la autorización para la apertura. Si no MQC.MQOO_ALTERNATE_USER_AUTHORITY se especifica, *alternateUserId* se puede dejar en blanco o nulo.

```
public MQTopic AccessTopic( MQDestination destination, string topicName, string
topicObject, int options);
public MQTopic AccessTopic( MQDestination destination, string topicName, string
topicObject, int options, string alternateUserId);
public MQTopic AccessTopic( MQDestination destination, string topicName, string
topicObject, int options, string alternateUserId, string subscriptionName);
public MQTopic AccessTopic( MQDestination destination, string topicName, string
topicObject, int options, string alternateUserId, string subscriptionName,
System.Collections.Hashtable properties);
public MQTopic AccessTopic(string topicName, string topicObject, int openAs,
int options);
public MQTopic AccessTopic(string topicName, string topicObject, int openAs,
int options, string alternateUserId);
public MQTopic AccessTopic(string topicName, string topicObject, int options,
string alternateUserId, string subscriptionName);
public MQTopic AccessTopic(string topicName, string topicObject, int options,
string alternateUserId, string subscriptionName, System.Collections.Hashtable
properties);
```

Acceder a un tema en este gestor de colas.

Los objetos de MQTopic están estrechamente relacionados con los objetos de tema administrativo, que a veces se denominan objetos de tema. En la entrada, *topicObject* apunta a un objeto de tema administrativo. El constructor MQTopic obtiene una serie de tema del objeto de tema y la combina con *topicName* para crear un nombre de tema. Uno o ambos *topicObject* o *topicName* pueden ser nulos. El nombre de tema se compara con el árbol de temas y el nombre del objeto de tema administrativo más coincidente se devuelve en *topicObject*.

Los temas que están asociados con el objeto MQTopic son el resultado de combinar dos series de tema. La primera serie de tema se define mediante el objeto de tema administrativo identificado por *topicObject*. La segunda serie de tema es *topicString*. La serie de tema resultante asociada con el objeto MQTopic puede identificar varios temas incluyendo comodines.

En función de si el tema está abierto para su publicación o suscripción, puede utilizar los métodos MQTopic .Put para publicar en temas, o los métodos MQTopic .Get para recibir publicaciones en temas. Si desea publicar y suscribirse al mismo tema, debe acceder al tema dos veces, una para publicar y otra para suscribirse.

Si crea un objeto MQTopic para la suscripción, sin proporcionar un objeto MQDestination , se presupone una suscripción gestionada. Si pasa una cola como un objeto MQDestination , se presupone una suscripción no gestionada. Debe asegurarse de que las opciones de suscripción que establezca sean coherentes con la suscripción que se está gestionando o que no está gestionada.

destino

destination es una instancia de MQQueue . Al proporcionar *destination*, MQTopic se abre como una suscripción no gestionada. Las publicaciones sobre el tema se entregan en la cola a la que se accede como *destination*.

topicName

Serie de tema que es la segunda parte del nombre de tema. *topicName* se concatena con la serie de tema definida en el objeto de tema administrativo *topicObject* . Puede establecer *topicName* en nulo, en cuyo caso el nombre de tema se define mediante la serie de tema en *topicObject*.

topicObject

En la entrada, *topicObject* es el nombre del objeto de tema que contiene la serie de tema que forma la primera parte del nombre de tema. La serie de tema en *topicObject* se concatena con *topicName*. Las reglas para construir series de tema se definen en [Combinación de series de tema](#).

En la salida, *topicObject* contiene el nombre del objeto de tema administrativo que es la coincidencia más cercana en el árbol de temas con el tema identificado por la serie de tema.

openAs

Acceda al tema para publicar o suscribirse. El parámetro sólo puede contener una de estas opciones:

- MQC.MQTOPIC_OPEN_AS_SUBSCRIPTION
- MQC.MQTOPIC_OPEN_AS_PUBLICATION

opciones

Combine las opciones que controlan la apertura del tema para publicación o suscripción. Utilice las constantes de MQC .MQSO_* para acceder a un tema para la suscripción y las constantes de MQC .MQOO_* para acceder a un tema para su publicación.

Si se necesita más de una opción, añada los valores juntos o combine los valores de opción utilizando el operador OR a nivel de bit.

alternateUserId

Especifique el ID de usuario alternativo que se utiliza para comprobar la autorización necesaria para finalizar la operación. Debe especificar *alternateUserId*, si MQC .MQOO_ALTERNATE_USER_AUTHORITY o MQC .MQSO_ALTERNATE_USER_AUTHORITY se ha establecido en el parámetro de opciones.

subscriptionName

subscriptionName es necesario si se proporcionan las opciones MQC .MQSO_DURABLE o MQC .MQSO_ALTER . En ambos casos, MQTopic se abre implícitamente para la suscripción. Se genera una excepción si se ha establecido MQC .MQSO_DURABLE y la suscripción existe, o si se ha establecido MQC .MQSO_ALTER y la suscripción no existe.

Propiedades

Establezca cualquiera de las propiedades de suscripción especiales listadas utilizando una tabla hash. Las entradas especificadas en la tabla hash se actualizan con valores de salida. Las entradas no se añaden a la tabla hash para informar de los valores de salida.

- MQC.MQSUB_PROP_ALTERNATE_SECURITY_ID
- MQC.MQSUB_PROP_SUBSCRIPTION_EXPIRY
- MQC.MQSUB_PROP_SUBSCRIPTION_USER_DATA
- MQC.MQSUB_PROP_SUBSCRIPTION_CORRELATION_ID
- MQC.MQSUB_PROP_PUBLICATION_PRIORITY
- MQC.MQSUB_PROP_PUBLICATION_ACCOUNTING_TOKEN
- MQC.MQSUB_PROP_PUBLICATION_APPLICATIONID_DATA

public MQAsyncStatus GetAsyncStatus();

Genera MQException

Devuelve un MQAsyncStatus objeto que representa la actividad asíncrona para la conexión del gestor de colas.

public void Backout();

Lanza MQException.

Deshacer cualquier mensaje que se haya leído o escrito dentro del punto de sincronización desde el último punto de sincronización.

Los mensajes que se escribieron con la MQC.MQPMO_SYNCPOINT bandera establecida se eliminan de las colas. Los mensajes leídos con la MQC.MQGMO_SYNCPOINT bandera se reincorporan a las colas de las que proceden. Si los mensajes son persistentes, los cambios se registran.

Para los clientes reconectables, el código MQRC_NONE de motivo se devuelve al cliente una vez que la reconexión se ha realizado correctamente.

public void Begin();

Lanza MQException.

Begin Solo es compatible con el modo de enlaces de servidor. Inicia una unidad de trabajo global.

public void Commit();

Lanza MQException.

Confirma cualquier mensaje que se haya leído o escrito dentro del punto de sincronización desde el último punto de sincronización.

Los mensajes escritos con el MQC.MQPMO_SYNCPOINT indicador activado están disponibles para otras aplicaciones. Los mensajes recuperados con la MQC.MQGMO_SYNCPOINT bandera establecida se eliminan. Si los mensajes son persistentes, los cambios se registran.

Los siguientes códigos de motivo se devuelven a un cliente reconectable:

- MQRC_CALL_INTERRUPTED si se pierde la conexión mientras se realiza la llamada de confirmación.
- MQRC_BACKED_OUT si la llamada de confirmación se emite después de la reconexión.

Disconnect();

Lanza MQException.

Cierre la conexión con el gestor de colas. Todos los objetos a los que se accede en este gestor de colas ya no son accesibles para esta aplicación. Para volver a acceder a los objetos, cree un MQQueueManager objeto.

Por lo general, cualquier trabajo realizado como parte de una unidad de trabajo se confirma. Sin embargo, si la unidad de trabajo es gestionada por .NET, la unidad de trabajo podría revertirse.

```

public void Put(int type, string destinationName, MQMessage message);
public void Put(int type, string destinationName, MQMessage message
MQPutMessageOptions putMessageOptions);
public void Put(int type, string destinationName, string queueManagerName,
string topicString, MQMessage message);
public void Put(string queueName, MQMessage message);
public void Put(string queueName, MQMessage message, MQPutMessageOptions
putMessageOptions);
public void Put(string queueName, string queueManagerName, MQMessage message);
public void Put(string queueName, string queueManagerName, MQMessage message,
MQPutMessageOptions putMessageOptions);
public void Put(string queueName, string queueManagerName, MQMessage message,
MQPutMessageOptions putMessageOptions, string alternateUserId);

```

Lanza MQException.

Coloca un único mensaje en una cola o tema sin crear primero un objeto MQQueue MQTopic o.

queueName

El nombre de la cola en la que se colocará el mensaje.

destinationName

El nombre de un objeto de destino. Dependiendo del valor de *type*, puede ser una cola o un tema.

Tipo

El tipo de objeto de destino. No se deben combinar las opciones.

MQC.MQOT_Q

Cola

MQC.MQOT_TOPIC

Tema

queueManagerName

El nombre del gestor de colas o alias del gestor de colas en el que se define la cola. Si se especifica MQC.MQOT_TOPIC el tipo, este parámetro se ignora.

Si la cola es una cola modelo y el nombre del gestor de colas resuelto no es este gestor de colas, se lanza MQException un.

topicString

topicString se combina con el nombre del tema en el *destinationName* objeto de tema.

topicString se ignora si *destinationName* es una cola.

mensaje

El mensaje que se va a enviar. El mensaje es un objeto de entrada/salida.

Los siguientes códigos de motivo se devuelven a un cliente reconectable:

- MQRC_CALL_INTERRUPTED si se interrumpe la conexión mientras se realiza una llamada Put en un mensaje persistente.
- MQRC_NONE si la conexión se establece correctamente al realizar una llamada Put en un mensaje no persistente (consulte [Recuperación de aplicaciones](#)).

putMessageOptions

Opciones que controlan las acciones de la venta.

Si omite *putMessageOptions*, se crea una *putMessageOptions* instancia predeterminada de. *putMessageOptions* es un objeto de entrada/salida.

Si utiliza la MQPMO_LOGICAL_ORDER opción en un cliente reconectable, se devuelve el código MQRC_RECONNECT_INCOMPATIBLE de motivo.

alternateUserId

Especifica un identificador de usuario alternativo que se utiliza para comprobar la autorización al colocar el mensaje en una cola.

Puede omitir *alternateUserId* si no establece `MQC.MQOO_ALTERNATE_USER_AUTHORITY` en *putMessageOptions*. Si establece `MQC.MQOO_ALTERNATE_USER_AUTHORITY`, también debe establecer *alternateUserId*. *alternateUserId* no tiene efecto a menos que también se establezca `MQC.MQOO_ALTERNATE_USER_AUTHORITY`.

Constructores

```
public MQQueueManager();  
public MQQueueManager(string queueManagerName);  
public MQQueueManager(string queueManagerName, Int options);  
public MQQueueManager(string queueManagerName, Int options, string channel,  
string connName);  
public MQQueueManager(string queueManagerName, string channel, string  
connName);  
public MQQueueManager(string queueManagerName, System.Collections.Hashtable  
properties);
```

Lanza `MQException`.

Crea una conexión con un gestor de colas. Seleccione entre crear una conexión de cliente o una conexión de servidor.

Debe tener autorización para `inquire` (inq) en el gestor de colas para conectar con el gestor de colas. Sin autorización para `inquire`, el intento de conexión falla.

Se crea una conexión de cliente si se cumple alguna de las siguientes condiciones:

1. *channel* o *connName* se especifican en el constructor.
2. *HostName*, *Port*, o *Channel* se especifican en *properties*.
3. *MQEnvironment.HostName*, *MQEnvironment.Port*, o *MQEnvironment.Channel* se especifican.

Los valores de las propiedades de conexión se establecen por defecto en el orden que se muestra. Los *channel* y *connName* del constructor tienen prioridad sobre los valores de las propiedades del constructor. Los valores de las propiedades del constructor tienen prioridad sobre los `MQEnvironment` de las propiedades.

El nombre del host, el nombre del canal y el puerto se definen en la `MQEnvironment` clase.

queueManagerName

Nombre del administrador de colas o del grupo de administradores de colas al que conectarse.

Omita el parámetro, déjelo en blanco o como nulo para seleccionar el gestor de colas predeterminado. La conexión predeterminada del gestor de colas en un servidor es al gestor de colas predeterminado en el servidor. La conexión predeterminada del gestor de colas en una conexión de cliente es al gestor de colas al que está conectado el listener.

opciones

Especifique las opciones `MQCNO` de conexión. Los valores deben ser aplicables al tipo de conexión que se está realizando. Por ejemplo, si se especifican las siguientes propiedades de conexión de servidor para una conexión de cliente, se lanza `MQException` una excepción.

- `MQC.MQCNO_FASTPATH_BINDING`
- `MQC.MQCNO_STANDARD_BINDING`

Propiedades

El parámetro `properties` toma una serie de pares clave/valor que anulan las propiedades establecidas por `MQEnvironment`; véase el ejemplo, [“Anular MQEnvironment propiedades” en la página 1802](#). Las siguientes propiedades se pueden anular:

- `MQC.CONNECT_OPTIONS_PROPERTY`
- `MQC.CONNECTION_NAME_PROPERTY`
- `MQC.ENCRYPTION_POLICY_SUITE_B`
- `MQC.HOST_NAME_PROPERTY`
- `MQC.PORT_PROPERTY`
- `MQC.CHANNEL_PROPERTY`
- `MQC.SSL_CIPHER_SPEC_PROPERTY`
- `MQC.SSL_PEER_NAME_PROPERTY`
- `MQC.SSL_CERT_STORE_PROPERTY`
- `MQC.SSL_CRYPTOHARDWARE_PROPERTY`
- `MQC.SECURITY_EXIT_PROPERTY`
- `MQC.SECURITY_USERDATA_PROPERTY`
- `MQC.SEND_EXIT_PROPERTY`
- `MQC.SEND_USERDATA_PROPERTY`
- `MQC.RECEIVE_EXIT_PROPERTY`
- `MQC.RECEIVE_USERDATA_PROPERTY`
- `MQC.USER_ID_PROPERTY`
- `MQC.PASSWORD_PROPERTY`
- `MQC.MQAIR_ARRAY`
- `MQC.KEY_RESET_COUNT`
- `MQC.FIPS_REQUIRED`
- `MQC.HDR_CMP_LIST`
- `MQC.MSG_CMP_LIST`
- `MQC.TRANSPORT_PROPERTY`

canal

Nombre de un canal de conexión al servidor

connName

Nombre de la conexión en el formato *HostName (Puerto)*.

Puede proporcionar una lista de *nombres de host y puertos* como argumento al constructor `MQQueueManager(String queueManagerName, Hashtable properties)` utilizando `CONNECTION_NAME_PROPERTY`.

Por ejemplo:

```
ConnectionName = "fred.mq.com(2344),nick.mq.com(3746),tom.mq.com(4288)";
Hashtable Properties=new Hashtable();
properties.Add(MQC.CONNECTION_NAME_PROPERTY,ConnectionName);
MQQueueManager qmgr=new MQQueue Manager("qmgrname",properties);
```

Cuando se intenta establecer una conexión, la lista de nombres de conexión se procesa por orden. Si falla el intento de conexión al primer nombre de host y puerto, se intenta la conexión al segundo par de atributos. El cliente repite este proceso hasta que se establece una conexión satisfactoria o se agota la lista. Si la lista se agota, se devuelve un código de motivo y un código de finalización adecuados a la aplicación cliente.

Cuando no se proporciona un número de puerto para el nombre de la conexión, se utiliza el puerto predeterminado (configurado en `mqclient.ini`).

Configurar la lista de conexiones

Puede configurar la lista de conexiones utilizando los siguientes métodos cuando se establecen las opciones de reconexión automática del cliente:

Configure la lista de conexiones a través de MQSERVER

Puede configurar la lista de conexiones a través del símbolo del sistema.

En la línea de comandos, introduzca el siguiente comando:

```
MQSERVER=SYSTEM.DEF.SVRCONN/TCP/Hostname1(Port1),Hostname2(Port2),Hostname3(Port3)
```

Por ejemplo:

```
MQSERVER=SYSTEM.DEF.SVRCONN/TCP/fred.mq.com(5266),nick.mq.com(6566),jack.mq.com(8413)
```

Si configura la conexión en MQSERVER, no la configure en la aplicación.

Si configura la lista de conexiones en la aplicación, esta sobrescribirá cualquier configuración establecida en la variable de entorno MQSERVER.

Configure la lista de conexiones a través de la aplicación

Puede configurar la lista de conexiones en la aplicación especificando las propiedades del nombre de host y el puerto.

```
String connName = "fred.mq.com(2344), nick.mq.com(3746), chris.mq.com(4288)";  
MQQueueManager qm = new MQQueueManager("QM1", "TestChannel", connName);
```

Configure la lista de conexiones a través de app.config

App.config es un archivo XML en el que se especifican los pares clave-valor.

En la lista de conexiones, especifique

```
<app.Settings>  
<add key="Connection1" value="Hostname1(Port1)"/>  
<add key="Connection2" value="Hostname2(Port2)"/>  
</app.Settings>
```

Por ejemplo:

```
<app.Settings>  
<add key="Connection1" value="fred.mq.com(2966)"/>  
<add key="Connection2" value="alex.mq.com(6533)"/>  
</app.Settings>
```

Puede cambiar directamente la lista de conexiones en el app.config archivo.

Configure la lista de conexiones a través de MQEnvironment

Para configurar la lista de conexiones a través de MQEnvironment, utilice la *ConnectionName* propiedad.

```
MQEnvironment.ConnectionName = "fred.mq.com(4288),alex.mq.com(5211);
```

La *ConnectionName* propiedad sobrescribe las propiedades del nombre de host y del puerto establecidas en el MQEnvironment.

Crear una conexión de cliente

El siguiente ejemplo muestra cómo crear una conexión de cliente con un gestor de colas. Puede crear una conexión de cliente configurando las MQEnvironment variables antes de crear un nuevo MQQueueManager objeto.

```

MQEnvironment.Hostname = "fred.mq.com"; // host to connect to
MQEnvironment.Port      = 1414;         // port to connect to
                                     // If not explicitly set,
                                     // defaults to 1414
                                     // (the default IBM MQ port)
MQEnvironment.Channel   = "channel.name"; // the case sensitive
                                     // name of the
                                     // SVR CONN channel on
                                     // the queue manager
MQQueueManager qMgr     = new MQQueueManager("MYQM");

```

Figura 10. Conexión de cliente

Anular MQEnvironment propiedades

El siguiente ejemplo muestra cómo crear un gestor de colas con su ID de usuario y contraseña definidos en una tabla hash.

```

Hashtable properties = new Hashtable();
properties.Add( MQC.USER_ID_PROPERTY, "ExampleUserId" );
properties.Add( MQC.PASSWORD_PROPERTY, "ExamplePassword" );

try
{
    MQQueueManager qMgr = new MQQueueManager("qmgrname", properties);
}
catch (MQException mqe)
{
    System.Console.WriteLine("Connect failed with " + mqe.Message);
    return((int)mqe.Reason);
}

```

Figura 11. Propiedades MQEnvironment de anulación

Crear una conexión reconectable

El siguiente ejemplo muestra cómo volver a conectar automáticamente un cliente a un gestor de colas.

```

Hashtable properties = new Hashtable(); // The queue manager name and the
                                     // properties how it has to be connected

properties.Add(MQC.CONNECT_OPTIONS_PROPERTY, MQC.MQCNO_RECONNECT); // Options
                                     // through which reconnection happens

properties.Add(MQC.CONNECTION_NAME_PROPERTY, "fred.mq.com(4789),nick.mq.com(4790)"); // The list
                                     // of queue managers through which reconnection happens

MQ QueueManager qmgr = new MQQueueManager("qmgrname", properties);

```

Figura 12. Reconexión automática de un cliente a un gestor de colas

Clase MQSubscription.NET

Utilice MQSubscription para solicitar que las publicaciones retenidas se envíen al suscriptor. MQSubscription es una propiedad de un objeto MQTopic abierto para suscripción.

Clase

```

System.Object
├── IBM.WMQ.MQBase
│   └── IBM.WMQ.MQBaseObject
│       └── IBM.WMQ.MQManagedObject
│           └── IBM.WMQ.MQSubscription

```

```
public class IBM.WMQ.MQSubscription extends IBM.WMQ.MQManagedObject;
```

- [“Propiedades” en la página 1803](#)
- [“Métodos” en la página 1803](#)
- [“Constructores” en la página 1803](#)

Propiedades

Acceda a las propiedades de suscripción utilizando la clase `MQManagedObject` ; consulte [“Propiedades” en la página 1762](#).

Métodos

Acceda a los métodos de suscripción `Inquire`, `Set` y `Get` utilizando la clase `MQManagedObject` ; consulte [“Métodos” en la página 1763](#).

public int RequestPublicationUpdate(int options);

Genera `MQException`.

Solicite una publicación actualizada para el tema actual. Si el gestor de colas tiene publicaciones retenidas para el tema, se envían al suscriptor.

Antes de llamar a `RequestPublicationUpdate`, abra un tema para la suscripción para obtener un objeto `MQSubscription` .

Normalmente, abra la suscripción con la opción `MQC.MQSO_PUBLICATIONS_ON_REQUEST` . Si no hay comodines en la serie de tema, sólo se envía una publicación como resultado de esta llamada. Si la serie de tema contiene comodines, es posible que se envíen muchas publicaciones. El método devuelve el número de publicaciones retenidas que se envían a la cola de suscripción. No hay garantía de que se reciban tantas publicaciones, especialmente si son mensajes no persistentes.

opciones

MQC.MQSRO_FAIL_IF QUIESCING

El método falla si el gestor de colas está en un estado de inmovilización. En z/OS, para una aplicación CICS o IMS , `MQC.MQSRO_FAIL_IF QUIESCING` también fuerza que el método falle si la conexión está en un estado inactivo.

MQC.MQSRO_NONE

No se especifica ninguna opción.

Constructores

No hay ningún constructor público .

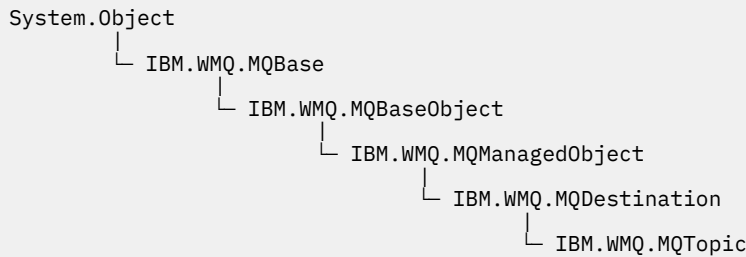
Se devuelve un objeto `MQSubscription` en la propiedad `SubscriptionReference` de un objeto `MQTopic` que está abierto para suscripción.

Llame al método `RequestPublicationUpdate` . `MQSubscription` es una subclase de `MQManagedObject`. Utilice la referencia para acceder a las propiedades y métodos de `MQManagedObject`.

Clase MQTopic.NET

Utilice `MQTopic` para publicar o suscribir mensajes sobre un tema, o para consultar o establecer atributos de un tema. Cree un objeto `MQTopic` para publicar o suscribirse utilizando un constructor o el método `MQQueueManager.AccessTopic` .

Clase



```
public class IBM.WMQ.MQTopic extends IBM.WMQ.MQDestination;
```

- [“Propiedades” en la página 1804](#)
- [“Métodos” en la página 1804](#)
- [“Constructores” en la página 1806](#)

Propiedades

Prueba de `MQException` que se genera al obtener las propiedades.

public Boolean IsDurable {get;}

Propiedad de sólo lectura que devuelve `True` si la suscripción es duradera o `False` de lo contrario. Si el tema se ha abierto para su publicación, la propiedad se ignora y siempre devolverá `False`.

public Boolean IsManaged {get;}

Propiedad de sólo lectura que devuelve `True` si la suscripción está gestionada por el gestor de colas, o `False` de lo contrario. Si el tema se ha abierto para su publicación, la propiedad se ignora y siempre devolverá `False`.

public Boolean IsSubscribed {get;}

Propiedad de sólo lectura que devuelve `True` si el tema se ha abierto para suscripción y `False` si el tema se ha abierto para publicación.

public MQSubscription SubscriptionReference {get;}

Propiedad de sólo lectura que devuelve el objeto `MQSubscription` asociado con un objeto de tema abierto para suscripción. La referencia está disponible si desea modificar las opciones de cierre o iniciar cualquiera de los métodos de objetos.

public MQDestination UnmanagedDestinationReference {get;}

Propiedad de sólo lectura que devuelve el `MQQueue` asociado a una suscripción no gestionada. Es el destino especificado cuando se creó el objeto de tema. La propiedad devuelve un valor nulo para los objetos de tema abiertos para publicación o con una suscripción gestionada.

Métodos

public void Put(MQMessage message);

public void Put(MQMessage message, MQPutMessageOptions putMessageOptions);

Emite `MQException`.

Publica un mensaje en el tema.

Las modificaciones en el objeto `MQMessage` después de que se haya realizado la llamada `Put` no afectan al mensaje real en el tema de publicación o cola IBM MQ.

`Put` actualiza las propiedades `MessageId` y `CorrelationId` del objeto `MQMessage` y no borra los datos del mensaje. Las llamadas `Put` o `Get` adicionales hacen referencia a la información actualizada en el objeto `MQMessage`. Por ejemplo, en el siguiente fragmento de código, el primer mensaje contiene a y el segundo ab.

```
msg.WriteString("a");
q.Put(msg,pmo);
msg.WriteString("b");
q.Put(msg,pmo);
```


mensaje

Un objeto `MQMessage` que contiene los datos del descriptor de mensaje y el mensaje que se va a enviar. El descriptor de mensaje se puede modificar como consecuencia de este método. Los valores del descriptor de mensaje inmediatamente después de la finalización de este método son los valores que se colocaron en la cola o se publicaron en el tema.

Se devuelven los siguientes códigos de razón a un cliente reconectable:

- `MQRC_CALL_INTERRUPTED` si la conexión se interrumpe al ejecutar una llamada `Put` en un mensaje persistente y la reconexión se realiza correctamente.
- `MQRC_NONE` si la conexión es satisfactoria al ejecutar una llamada `Put` en un mensaje no persistente (consulte [Recuperación de aplicaciones](#)).

putMessageOptions

Opciones que controlan la acción de la colocación.

Si no se especifica `putMessageOptions`, se utiliza la instancia predeterminada de `MQPutMessageOptions`.

Si utiliza la opción `MQPMO_LOGICAL_ORDER` en un cliente reconectable, se devuelve el código de razón `MQRC_RECONNECT_INCOMPATIBLE`.

Nota: Por motivos de simplicidad y rendimiento, si desea transferir un único mensaje a una cola, utilice el objeto `MQQueueManager.Put`. Debe tener un objeto `MQQueue` para esto.

```
public void Get(MQMessage message);  
public void Get(MQMessage message, MQGetMessageOptions getMessageOptions);  
public void Get(MQMessage message, MQGetMessageOptions getMessageOptions, int  
MaxMsgSize);
```

Emite `MQException`.

Recupera un mensaje del tema.

Este método utiliza una instancia predeterminada de `MQGetMessageOptions` para realizar la obtención. La opción de mensaje utilizada es `MQGMO_NOWAIT`.

Si la obtención falla, el objeto `MQMessage` no se modifica. Si se ejecuta correctamente, el descriptor de mensaje y las partes de datos de mensaje del `MQMessage` se sustituyen por el descriptor de mensaje y los datos de mensaje del mensaje de entrada.

Todas las llamadas a IBM MQ desde un `MQQueueManager` determinado son síncronas. Por lo tanto, si realiza una operación `get` con espera, todas las demás hebras que utilizan el mismo `MQQueueManager` no pueden realizar más llamadas IBM MQ hasta que se realice la llamada `Get`. Si necesita varias hebras para acceder a IBM MQ simultáneamente, cada hebra debe crear su propio objeto `MQQueueManager`.

mensaje

Contiene el descriptor de mensaje y los datos de mensaje devueltos. Algunos de los campos del descriptor de mensaje son parámetros de entrada. Es importante asegurarse de que los parámetros de entrada `MessageId` y `CorrelationId` estén establecidos según sea necesario.

Un cliente reconectable devuelve el código de razón `MQRC_BACKED_OUT` después de una reconexión satisfactoria, para los mensajes recibidos en `MQGM_SYNCPOINT`.

getMessageOptions

Opciones que controlan la acción de la obtención.

El uso de la opción `MQC.MQGMO_CONVERT` puede dar como resultado una excepción con el código de razón `MQC.MQRC_CONVERTED_STRING_TOO_BIG` al convertir de códigos de caracteres de un solo byte a códigos de doble byte. En este caso, el mensaje se copia en el almacenamiento intermedio sin conversión.

Si no se especifica `getMessageOptions`, la opción de mensaje utilizada es `MQGMO_NOWAIT`.

Si utiliza la opción `MQGMO_LOGICAL_ORDER` en un cliente reconectable, se devuelve el código de razón `MQRC_RECONNECT_INCOMPATIBLE`.

MaxMsgSize

El mensaje más grande que este objeto de mensaje va a recibir. Si el mensaje en la cola es mayor que este tamaño, se produce una de las dos cosas siguientes:

- Si el distintivo MQGMO_ACCEPT_TRUNCATED_MSG se establece en el objeto MQGetMessageOptions, el mensaje se rellena con la mayor cantidad posible de datos de mensaje. Se genera una excepción con el código de terminación MQCC_WARNING y el código de razón MQRC_TRUNCATED_MSG_ACCEPTED.
- Si el distintivo MQGMO_ACCEPT_TRUNCATED_MSG no está establecido, el mensaje permanece en la cola. Se genera una excepción con el código de terminación MQCC_WARNING y el código de razón MQRC_TRUNCATED_MSG_FAILED.

Si no se especifica *MaxMsgSize*, se recupera todo el mensaje.

Constructores

```
public MQTopic(MQQueueManager queueManager, MQDestination destination, string
topicName, string topicObject, int options);
public MQTopic(MQQueueManager queueManager, MQDestination destination, string
topicName, string topicObject, int options, string alternateUserId);
public MQTopic(MQQueueManager queueManager, MQDestination destination, string
topicName, string topicObject, int options, string alternateUserId, string
subscriptionName);
public MQTopic(MQQueueManager queueManager, MQDestination destination, string
topicName, string topicObject, int options, string alternateUserId, string
subscriptionName, System.Collections.Hashtable properties);
public MQTopic(MQQueueManager queueManager, string topicName, string
topicObject, int openAs, int options);
public MQTopic(MQQueueManager queueManager, string topicName, string
topicObject, int openAs, int options, string alternateUserId);
public MQTopic(MQQueueManager queueManager, string topicName, string
topicObject, int options, string alternateUserId, string subscriptionName);
public MQTopic(MQQueueManager queueManager, string topicName, string
topicObject, int options, string alternateUserId, string subscriptionName,
System.Collections.Hashtable properties);
```

Acceda a un tema en *queueManager*.

Los objetos de MQTopic están estrechamente relacionados con los objetos de tema administrativo, que a veces se denominan objetos de tema. En la entrada, *topicObject* apunta a un objeto de tema administrativo. El constructor MQTopic obtiene una serie de tema del objeto de tema y la combina con *topicName* para crear un nombre de tema. Uno o ambos *topicObject* o *topicName* pueden ser nulos. El nombre de tema se compara con el árbol de temas y el nombre del objeto de tema administrativo más coincidente se devuelve en *topicObject*.

Los temas que están asociados con el objeto MQTopic son el resultado de combinar dos series de tema. La primera serie de tema se define mediante el objeto de tema administrativo identificado por *topicObject*. La segunda serie de tema es *topicString*. La serie de tema resultante asociada con el objeto MQTopic puede identificar varios temas incluyendo comodines.

En función de si el tema está abierto para su publicación o suscripción, puede utilizar los métodos MQTopic.Put para publicar en temas, o los métodos MQTopic.Get para recibir publicaciones en temas. Si desea publicar y suscribirse al mismo tema, debe acceder al tema dos veces, una para publicar y otra para suscribirse.

Si crea un objeto MQTopic para la suscripción, sin proporcionar un objeto MQDestination, se presupone una suscripción gestionada. Si pasa una cola como un objeto MQDestination, se presupone una suscripción no gestionada. Debe asegurarse de que las opciones de suscripción que establezca sean coherentes con la suscripción que se está gestionando o que no está gestionada.

queueManager

Gestor de colas sobre el que acceder a un tema.

destino

destination es una instancia de `MQQueue`. Al proporcionar *destination*, `MQTopic` se abre como una suscripción no gestionada. Las publicaciones sobre el tema se entregan en la cola a la que se accede como *destination*.

topicName

Serie de tema que es la segunda parte del nombre de tema. *topicName* se concatena con la serie de tema definida en el objeto de tema administrativo *topicObject*. Puede establecer *topicName* en nulo, en cuyo caso el nombre de tema se define mediante la serie de tema en *topicObject*.

topicObject

En la entrada, *topicObject* es el nombre del objeto de tema que contiene la serie de tema que forma la primera parte del nombre de tema. La serie de tema en *topicObject* se concatena con *topicName*. Las reglas para construir series de tema se definen en [Combinación de series de tema](#).

En la salida, *topicObject* contiene el nombre del objeto de tema administrativo que es la coincidencia más cercana en el árbol de temas con el tema identificado por la serie de tema.

openAs

Acceda al tema para publicar o suscribirse. El parámetro sólo puede contener una de estas opciones:

- `MQC.MQTOPIC_OPEN_AS_SUBSCRIPTION`
- `MQC.MQTOPIC_OPEN_AS_PUBLICATION`

opciones

Combine las opciones que controlan la apertura del tema para publicación o suscripción. Utilice las constantes de `MQC.MQSO_*` para acceder a un tema para la suscripción y las constantes de `MQC.MQOO_*` para acceder a un tema para su publicación.

Si se necesita más de una opción, añada los valores juntos o combine los valores de opción utilizando el operador OR a nivel de bit.

alternateUserId

Especifique el ID de usuario alternativo que se utiliza para comprobar la autorización necesaria para finalizar la operación. Debe especificar *alternateUserId*, si `MQC.MQOO_ALTERNATE_USER_AUTHORITY` o `MQC.MQSO_ALTERNATE_USER_AUTHORITY` se ha establecido en el parámetro de opciones.

subscriptionName

subscriptionName es necesario si se proporcionan las opciones `MQC.MQSO_DURABLE` o `MQC.MQSO_ALTER`. En ambos casos, `MQTopic` se abre implícitamente para la suscripción. Se genera una excepción si se ha establecido `MQC.MQSO_DURABLE` y la suscripción existe, o si se ha establecido `MQC.MQSO_ALTER` y la suscripción no existe.

Propiedades

Establezca cualquiera de las propiedades de suscripción especiales listadas utilizando una tabla hash. Las entradas especificadas en la tabla hash se actualizan con valores de salida. Las entradas no se añaden a la tabla hash para informar de los valores de salida.

- `MQC.MQSUB_PROP_ALTERNATE_SECURITY_ID`
- `MQC.MQSUB_PROP_SUBSCRIPTION_EXPIRY`
- `MQC.MQSUB_PROP_SUBSCRIPTION_USER_DATA`
- `MQC.MQSUB_PROP_SUBSCRIPTION_CORRELATION_ID`
- `MQC.MQSUB_PROP_PUBLICATION_PRIORITY`
- `MQC.MQSUB_PROP_PUBLICATION_ACCOUNTING_TOKEN`
- `MQC.MQSUB_PROP_PUBLICATION_APPLICATIONID_DATA`

```

public MQTopic MQQueueManager.AccessTopic(MQDestination destination, string
topicName, string topicObject, int options);
public MQTopic MQQueueManager.AccessTopic(MQDestination destination, string
topicName, string topicObject, int options, string alternateUserId);
public MQTopic MQQueueManager.AccessTopic(MQDestination destination, string
topicName, string topicObject, int options, string alternateUserId, string
subscriptionName);
public MQTopic MQQueueManager.AccessTopic(MQDestination destination, string
topicName, string topicObject, int options, string alternateUserId, string
subscriptionName, System.Collections.Hashtable properties);
public MQTopic MQQueueManager.AccessTopic(string topicName, string topicObject,
int openAs, int options);
public MQTopic MQQueueManager.AccessTopic(string topicName, string topicObject,
int openAs, int options, string alternateUserId);
public MQTopic MQQueueManager.AccessTopic(string topicName, string topicObject,
int options, string alternateUserId, string subscriptionName);
public MQTopic MQQueueManager.AccessTopic(string topicName, string
topicObject, int options, string alternateUserId, string subscriptionName,
System.Collections.Hashtable properties);

```

Acceda a un tema en este gestor de colas.

Los objetos de MQTopic están estrechamente relacionados con los objetos de tema administrativo, que a veces se denominan objetos de tema. En la entrada, *topicObject* apunta a un objeto de tema administrativo. El constructor MQTopic obtiene una serie de tema del objeto de tema y la combina con *topicName* para crear un nombre de tema. Uno o ambos *topicObject* o *topicName* pueden ser nulos. El nombre de tema se compara con el árbol de temas y el nombre del objeto de tema administrativo más coincidente se devuelve en *topicObject*.

Los temas que están asociados con el objeto MQTopic son el resultado de combinar dos series de tema. La primera serie de tema se define mediante el objeto de tema administrativo identificado por *topicObject*. La segunda serie de tema es *topicString*. La serie de tema resultante asociada con el objeto MQTopic puede identificar varios temas incluyendo comodines.

En función de si el tema está abierto para su publicación o suscripción, puede utilizar los métodos MQTopic .Put para publicar en temas, o los métodos MQTopic .Get para recibir publicaciones en temas. Si desea publicar y suscribirse al mismo tema, debe acceder al tema dos veces, una para publicar y otra para suscribirse.

Si crea un objeto MQTopic para la suscripción, sin proporcionar un objeto MQDestination , se presupone una suscripción gestionada. Si pasa una cola como un objeto MQDestination , se presupone una suscripción no gestionada. Debe asegurarse de que las opciones de suscripción que establezca sean coherentes con la suscripción que se está gestionando o que no está gestionada.

destino

destination es una instancia de MQQueue . Al proporcionar *destination*, MQTopic se abre como una suscripción no gestionada. Las publicaciones sobre el tema se entregan en la cola a la que se accede como *destination*.

topicName

Serie de tema que es la segunda parte del nombre de tema. *topicName* se concatena con la serie de tema definida en el objeto de tema administrativo *topicObject* . Puede establecer *topicName* en nulo, en cuyo caso el nombre de tema se define mediante la serie de tema en *topicObject*.

topicObject

En la entrada, *topicObject* es el nombre del objeto de tema que contiene la serie de tema que forma la primera parte del nombre de tema. La serie de tema en *topicObject* se concatena con *topicName*. Las reglas para construir series de tema se definen en [Combinación de series de tema](#).

En la salida, *topicObject* contiene el nombre del objeto de tema administrativo que es la coincidencia más cercana en el árbol de temas con el tema identificado por la serie de tema.

openAs

Acceda al tema para publicar o suscribirse. El parámetro sólo puede contener una de estas opciones:

- MQC.MQTOPIC_OPEN_AS_SUBSCRIPTION
- MQC.MQTOPIC_OPEN_AS_PUBLICATION

opciones

Combine las opciones que controlan la apertura del tema para publicación o suscripción. Utilice las constantes de MQC.MQSO_* para acceder a un tema para la suscripción y las constantes de MQC.MQOO_* para acceder a un tema para su publicación.

Si se necesita más de una opción, añada los valores juntos o combine los valores de opción utilizando el operador OR a nivel de bit.

alternateUserId

Especifique el ID de usuario alternativo que se utiliza para comprobar la autorización necesaria para finalizar la operación. Debe especificar *alternateUserId*, si MQC.MQOO_ALTERNATE_USER_AUTHORITY o MQC.MQSO_ALTERNATE_USER_AUTHORITY se ha establecido en el parámetro de opciones.

subscriptionName

subscriptionName es necesario si se proporcionan las opciones MQC.MQSO_DURABLE o MQC.MQSO_ALTER. En ambos casos, MQTopic se abre implícitamente para la suscripción. Se genera una excepción si se ha establecido MQC.MQSO_DURABLE y la suscripción existe, o si se ha establecido MQC.MQSO_ALTER y la suscripción no existe.

Propiedades

Establezca cualquiera de las propiedades de suscripción especiales listadas utilizando una tabla hash. Las entradas especificadas en la tabla hash se actualizan con valores de salida. Las entradas no se añaden a la tabla hash para informar de los valores de salida.

- MQC.MQSUB_PROP_ALTERNATE_SECURITY_ID
- MQC.MQSUB_PROP_SUBSCRIPTION_EXPIRY
- MQC.MQSUB_PROP_SUBSCRIPTION_USER_DATA
- MQC.MQSUB_PROP_SUBSCRIPTION_CORRELATION_ID
- MQC.MQSUB_PROP_PUBLICATION_PRIORITY
- MQC.MQSUB_PROP_PUBLICATION_ACCOUNTING_TOKEN
- MQC.MQSUB_PROP_PUBLICATION_APPLICATIONID_DATA

Interfaz de IMQObjectTrigger.NET

Implemente IMQObjectTrigger para procesar los mensajes pasados por el supervisor de **runmqdmn.NET**.

Interfaz

```
public interface IBM.WMQMonitor.IMQObjectTrigger();
```

En función de si se especifica el control de punto de sincronización en el mandato **runmqdmn**, el mensaje se elimina de la cola antes o después de que se devuelva el método Execute.

Métodos

void Execute (MQQueueManager queueManager, MQQueue queue, MQMessage message, string param);

queueManager

Gestor de colas que aloja la cola que se está supervisando.

cola

Cola que se está supervisando.

mensaje

Mensaje leído de la cola.

param

Datos pasados desde UserParameter.

Interfaz de MQC.NET

Consulte una constante MQI añadiendo el prefijo MQC. al nombre de la constante. MQC define todas las constantes utilizadas por MQI.

Interfaz

```
System.Object
└─ IBM.WMQ.MQC
```

```
public interface IBM.WMQ.MQC extends System.Object;
```

Ejemplo

```
MQQueue queue;  
queue.closeOptions = MQC.MQCO_DELETE;
```

Identificadores de juego de caracteres para aplicaciones .NET

Descripciones de los juegos de caracteres que puede seleccionar para codificar mensajes de .NET IBM MQ

Juego de caracteres	Descripción
37	ibm037
437	ibm437 /PC Original
500	ibm500
819	iso-8859-1 / latin1 / ibm819
1200	Unicode
1208	UTF-8
273	ibm273
277	ibm277
278	ibm278
280	ibm280
284	ibm284
285	ibm285

Juego de caracteres	Descripción
297	ibm297
420	ibm420
424	ibm424
737	ibm737 /PC Griego
775	ibm775 /PC Báltico
813	iso-8859-7 /greek/ ibm813
838	ibm838
850	ibm850 /PC Latin 1
852	ibm852 /PC Latin 2
855	ibm855 /PC Cirílico
856	ibm856
857	ibm857 /PC Turco
860	ibm860 /PC Portugués
861	ibm861 /PC islandés
862	ibm862 /PC Hebreo
863	ibm863 /PC francés canadiense
864	ibm864 /PC árabe
865	ibm865 /PC Nordic
868	ibm868
869	ibm869 /PC Modern Griego
870	ibm870
871	ibm871
874	ibm874
875	ibm875
912	iso-8859-2 / latin2 / ibm912
913	iso-8859-3 / latin3 / ibm913
914	iso-8859-4 / latin4 / ibm914
915	iso-8859-5 /cirílico/ ibm915
916	iso-8859-8 /hebreo/ ibm916
918	ibm918
920	iso-8859-9 / latin5 / ibm920
921	ibm921
922	ibm922
930	ibm930
932	PC japonés

Juego de caracteres	Descripción
933	ibm933
935	ibm935
937	ibm937
939	ibm939
942	ibm942
943	ibm943
948	ibm948
949	ibm949
950	ibm950 /Big 5 Chino tradicional
954	eucjis
964	ibm964 /CNS 11643 Chino tradicional
970	ibm970
1006	ibm1006
1025	ibm1025
1026	ibm1026
1089	iso-8859-6 /arabic/ ibm1089
1097	ibm1097
1098	ibm1098
1112	ibm1112
1122	ibm1122
1123	ibm1123
1124	ibm1124
1250	Windows Latín 2
1251	Windows Cirílico
1252	Windows Latín 1
1253	Windows Griego
1254	Windows Turco
1255	Windows Hebreo
1256	Windows Árabe
1257	Windows Bático
1258	Windows Vietnamita
1381	ibm1381
1383	ibm1383
2022	JIS
5601	ksc-5601 Coreano

Juego de caracteres	Descripción
33722	ibm33722

Clases C++ de IBM MQ

Las clases de IBM MQ C++ encapsulan la IBM MQ interfaz de colas de mensajes (MQI). Hay un único archivo de encabezado de C++, « **imqi.hpp** », que engloba todas estas clases.

Para cada clase, se muestra la siguiente información:

Diagrama de jerarquía de clases

Un diagrama de clase que muestra la clase en su relación de herencia con sus clases padre inmediatas, si las hay.

Otras clases pertinentes

El documento enlaza con otras clases relevantes, como las clases padre, y las clases de objetos utilizados en las firmas de método.

Atributos de objetos

Atributos de la clase. Estos atributos se añaden a los definidos para cualquier clase padre. Muchos atributos reflejan miembros de estructura de datos de IBM MQ (consulte [“Referencia cruzada de C++ y MQI”](#) en la página 1814). Para obtener descripciones detalladas, consulte [“objeto, atributos de”](#) en la página 800.

Constructores

Firmas de los métodos especiales utilizados para crear un objeto de la clase.

Métodos de objeto (public)

Firmas de métodos que requieren una instancia de la clase para su operación y que no tienen restricciones de uso.

Cuando sea aplicable, también se mostrará la siguiente información:

Métodos de clase (public)

Firmas de métodos que no requieren una instancia de la clase para su operación y que no tienen restricciones de uso.

Métodos sobrecargados (clase padre)

Firmas de los métodos virtuales definidos en las clases padre, pero que muestran un comportamiento polimórfico diferente para esta clase.

Métodos de objeto (protegidos)

Firmas de métodos que requieren una instancia de la clase para su operación y que están reservadas para que las utilicen las implementaciones de clases derivadas. Esta sección sólo es de interés para los escritores de clase, en contraposición a los usuarios de clase.

Datos de objeto (protegidos)

Detalles de implementación para datos de instancia de objeto disponibles para las implementaciones de clases derivadas. Esta sección sólo es de interés para los escritores de clase, en contraposición a los usuarios de clase.

códigos de razón

Valores MQRC_* (consulte [Códigos de terminación y razón de la API](#)) que se pueden esperar de los métodos que fallan. Para obtener una lista exhaustiva de los códigos de razón que se pueden producir para un objeto de una clase, consulte la documentación de la clase padre. La lista documentada de códigos de razón para una clase no incluye los códigos de razón para las clases padre.

Nota:

1. Los objetos de estas clases no son seguros para hebras. Esto garantiza un rendimiento óptimo, pero tenga cuidado de no acceder a ningún objeto desde más de una hebra.
2. Se recomienda que, para un programa multihilo, se utilice un objeto ImqQueueManager distinto para cada hilo. Cada objeto de gestor debe tener su propia colección independiente de otros objetos, asegurándose de que los objetos de distintas hebras estén aislados entre sí.

Las clases son:

- [“ImqAuthenticationRecord Clase C” en la página 1831](#)
- [“Clase C++ ImqBinary” en la página 1833](#)
- [“Clase C++ ImqCache” en la página 1835](#)
- [“Clase C++ ImqChannel” en la página 1838](#)
- [“ImqCICSBridgeHeader Clase C++” en la página 1844](#)
- [“ImqDeadLetterHeader Clase C” en la página 1850](#)
- [“ImqDistributionList Clase C” en la página 1853](#)
- [“Clase C++ ImqError” en la página 1854](#)
- [“ImqGetMessageOptions Clase C” en la página 1855](#)
- [“Clase C++ ImqHeader” en la página 1859](#)
- [“ImqIMSBridgeHeader Clase C++” en la página 1860](#)
- [“Clase C++ ImqItem” en la página 1863](#)
- [“ImqMessage Clase C++” en la página 1865](#)
- [“ImqMessageTracker Clase C++” en la página 1872](#)
- [“Clase C++ ImqNamelist” en la página 1875](#)
- [“ImqObject Clase C++” en la página 1876](#)
- [“Clase C++ ImqProcess” en la página 1882](#)
- [“ImqPutMessageOptions Clase C” en la página 1884](#)
- [“Clase C++ ImqQueue” en la página 1886](#)
- [“ImqQueueManager Clase C++” en la página 1897](#)
- [“ImqReferenceHeader Clase C” en la página 1913](#)
- [“Clase C++ ImqString” en la página 1916](#)
- [“Clase C++ ImqTrigger” en la página 1921](#)
- [“ImqWorkHeader Clase C” en la página 1924](#)

Referencia cruzada de C++ y MQI

Esta recopilación de temas contiene información sobre la relación entre C++ y el lenguaje de programación de la Agencia de Seguridad de la Información (MQI).

Lea esta información junto con [“Tipos de datos utilizados en el lenguaje de programación « MQI »” en la página 177](#).

Esta tabla establece una correspondencia entre las estructuras de datos de « MQI » y las clases y archivos de inclusión de C++. Los temas siguientes muestran información de referencia cruzada para cada clase C++. Estas referencias cruzadas están relacionadas con el uso de las interfaces de procedimiento de IBM MQ subyacentes. Las clases ImqBinary,, ImqDistributionList, y ImqString no tienen atributos que entren en esta categoría, por lo que quedan excluidas.

Tabla 881. Estructura de datos, clase y referencia cruzada de archivo de inclusión		
Estructura de datos	Clase	Archivo de inclusión
MQAIRE	ImqAuthenticationRecord	imqair.hpp
	ImqBinary	imqbin.hpp
	ImqCache	imqcac.hpp
MQCD	ImqChannel	imqchl.hpp
MQCIH	ImqBridgeHeaderCICS	imqcih.hpp

Tabla 881. Estructura de datos, clase y referencia cruzada de archivo de inclusión (continuación)

Estructura de datos	Clase	Archivo de inclusión
MQDLH	ImqDeadLetterHeader	imqdlh.hpp
MQOR	ImqDistributionList	imqdst.hpp
	ImqError	imqerr.hpp
MQGMO	ImqGetMessageOptions	imqgmo.hpp
	ImqHeader	imqhdr.hpp
MQIIH	ImqBridgeHeaderIMS	imqihi.hpp
	ImqItem	imqitm.hpp
MQMD	ImqMessage	imqmsg.hpp
	ImqMessageTracker	imqmtr.hpp
	ImqNamelist	imqnml.hpp
MQOD, MQRR	ImqObject	imqobj.hpp
MQPMO, MQPMR, MQRR	ImqPutMessageOptions	imqpmo.hpp
	ImqProcess	imqpro.hpp
	ImqQueue	imqque.hpp
MQBO, MQCNO, MQCSP	ImqQueueManager	imqmgr.hpp
MQRMH	ImqReferenceHeader	imqrfh.hpp
	ImqString	imqstr.hpp
MQTM	ImqTrigger	imqtrg.hpp
MQTMC		
MQTMC2	ImqTrigger	imqtrg.hpp
MQXQH		
MQWIH	ImqWorkHeader	imqwih.hpp

ImqAuthenticationRecord referencia cruzada

Referencia cruzada de atributos, estructuras de datos, campos y llamadas para la clase C++ ImqAuthenticationRecord.

Tabla 882. Atributos, estructuras de datos, campos y llamadas

Atributo	Estructura de datos	Campo	Llamada
Nombre de conexión	MQAIRE	AuthInfoConnName	MQCONN
contraseña	MQAIRE	LDAPPassword	MQCONN
Tipo	MQAIRE	AuthInfoType	MQCONN
nombre de usuario	MQAIRE	LDAPUserNamePtr	MQCONN
	MQAIRE	LDAPUserNameOffset	MQCONN
	MQAIRE	LDAPUserNameLength	MQCONN

Referencia cruzada de ImqCache

Referencia cruzada de atributos y llamadas para la clase C++ ImqCache .

Tabla 883. Atributos y llamadas	
Atributo	Llamada
almacenamiento intermedio automático	MQGET
Longitud de almacenamiento intermedio	MQGET
puntero de almacenamiento intermedio	MQGET, MQPUT
Longitud de datos	MQGET
Desplazamiento de datos	MQGET
puntero de datos	MQGET
Longitud del mensaje	MQGET, MQPUT

Referencia cruzada de ImqChannel

Referencia cruzada de atributos, estructuras de datos, campos y llamadas para la clase C++ ImqChannel .

Tabla 884. Atributos, estructuras de datos, campos y llamadas			
Atributo	Estructura de datos	Campo	Llamada
latido del corazón por lotes	MQCD	BatchHeartbeat	MQCONN
nombre de canal	MQCD	ChannelName	MQCONN
Nombre de conexión	MQCD	ConnectionName	MQCONN
	MQCD	ShortConnectionName	MQCONN
Compresión de cabecera	MQCD	HdrCompList	MQCONN
intervalo de latido cardíaco	MQCD	HeartbeatInterval	MQCONN
Intervalo de mantenimiento activado	MQCD	KeepAliveInterval	MQCONN
dirección local	MQCD	LocalAddress	MQCONN
longitud máxima del mensaje	MQCD	MaxMsgLength	MQCONN
Compresión de mensaje	MQCD	MsgCompList	MQCONN
nombre modalidad	MQCD	ModeName	MQCONN
contraseña	MQCD	Contraseña	MQCONN
recuento de salidas de recepción	MQCD		MQCONN
nombres de salida de recepción	MQCD	ReceiveExit	MQCONN
	MQCD	ReceiveExitsDefined	MQCONN
	MQCD	ReceiveExitPtr	MQCONN
recibir datos de usuario	MQCD	ReceiveUserData	MQCONN
	MQCD	ReceiveUserDataPtr	MQCONN
Nombre de salida de seguridad	MQCD	SecurityExit	MQCONN
datos de usuario de seguridad	MQCD	SecurityUserData	MQCONN

Tabla 884. Atributos, estructuras de datos, campos y llamadas (continuación)			
Atributo	Estructura de datos	Campo	Llamada
recuento de salidas de envío	MQCD		MQCONN
nombres de salida de envío	MQCD	SendExit	MQCONN
	MQCD	SendExitsDefined	MQCONN
	MQCD	SendExitPtr	MQCONN
enviar datos de usuario	MQCD	SendUserData	MQCONN
	MQCD	SendUserDataPtr	MQCONN
SSL CipherSpec	MQCD	sslCipherSpecification	MQCONN
SSL tipo de autenticación del cliente	MQCD	sslClientAuthentication	MQCONN
Nombre de igual SSL	MQCD	sslPeerName	MQCONN
nombre del programa de transacciones	MQCD	TpName	MQCONN
tipo de transporte	MQCD	TransportType	MQCONN
ID de usuario	MQCD	UserIdentifier	MQCONN

Referencia cruzada ImqCICSBridgeHeader

Referencia cruzada de atributos, estructuras de datos y campos para la clase C++ ImqCICSBridgeHeader.

Tabla 885. Correlación de atributos, estructuras de datos y campos		
Atributo	Estructura de datos	Campo
código de terminación anómala de puente	MQCIH	AbendCode
Descriptor ADS	MQCIH	AdsDescriptor
identificador de atención	MQCIH	AttentionId
autenticador	MQCIH	Authenticator
código de terminación de puente	MQCIH	BridgeCompletionCode
desplazamiento de error de puente	MQCIH	ErrorOffset
código de razón de puente	MQCIH	BridgeReason
código de cancelación de puente	MQCIH	CancelCode
tarea conversacional	MQCIH	ConversationalTask
Posición del cursor	MQCIH	CursorPosition
señal de recurso	MQCIH	Recurso
tiempo de mantenimiento del recurso	MQCIH	FacilityKeepTime
recurso como	MQCIH	FacilityLike
función	MQCIH	Función
obtener intervalo de espera	MQCIH	GetWaitInterval
Tipo de enlace	MQCIH	LinkType
identificador de transacción siguiente	MQCIH	NextTransactionId

Tabla 885. Correlación de atributos, estructuras de datos y campos (continuación)		
Atributo	Estructura de datos	Campo
longitud de datos de salida	MQCIH	OutputDataLength
formato de respuesta	MQCIH	ReplyToFormat
código de retorno de puente	MQCIH	ReturnCode
código de inicio	MQCIH	StartCode
estado de finalización de tarea	MQCIH	TaskEndStatus
Identificador de transacción	MQCIH	TransactionId
control uow	MQCIH	UowControl
versión	MQCIH	Versión

ImqDeadLetterHeader referencia cruzada

Referencia cruzada de atributos, estructuras de datos y campos para la clase ImqDeadLetterHeader C++.

Tabla 886. Correlación de atributos, estructuras de datos y campos		
Atributo	Estructura de datos	Campo
código de razón de mensaje no enviado	MQDLH	Razón
Nombre del gestor de colas de destino	MQDLH	DestQMgrName
Nombre de la cola de destino	MQDLH	DestQName
Nombre de aplicación de transferencia	MQDLH	PutApplName
Tipo de aplicación de transferencia	MQDLH	PutApplType
Fecha de transferencia	MQDLH	PutDate
Hora de transferencia	MQDLH	PutTime

Referencia cruzada ImqError

Referencia cruzada de atributos y llamadas para la clase C++ ImqError .

Tabla 887. Atributos y llamadas	
Atributo	Llamada
código de terminación	MQBACK, MQBEGIN, MQCLOSE, MQCMIT, MQCONN, MQCONNX, MQDISC, MQGET, MQINQ, MQOPEN, MQPUT, MQSET
código de razón	MQBACK, MQBEGIN, MQCLOSE, MQCMIT, MQCONN, MQCONNX, MQDISC, MQGET, MQINQ, MQOPEN, MQPUT, MQSET

ImqGetMessageOptions referencia cruzada

Referencia cruzada de atributos, estructuras de datos y campos para la clase ImqGetMessageOptions C++.

Tabla 888. Correlación de atributos, estructuras de datos y campos		
Atributo	Estructura de datos	Campo
Estado del grupo	MQGMO	GroupStatus

Tabla 888. Correlación de atributos, estructuras de datos y campos (continuación)

Atributo	Estructura de datos	Campo
opciones de coincidencia	MQGMO	MatchOptions
token de mensaje	MQGMO	MessageToken
opciones	MQGMO	Opciones
Nombre de cola resuelto	MQGMO	ResolvedQName
longitud devuelta	MQGMO	ReturnedLength
segmentación	MQGMO	Segmentación
estado de segmento	MQGMO	SegmentStatus
	MQGMO	Signal1
	MQGMO	Signal2
participación de punto de sincronismo	MQGMO	Opciones
Intervalo de espera	MQGMO	WaitInterval

Referencia cruzada de ImqHeader

Referencia cruzada de atributos, estructuras de datos y campos para la clase C++ ImqHeader .

Tabla 889. Correlación de atributos, estructuras de datos y campos

Atributo	Estructura de datos	Campo
juego de caracteres	MQDLH, MQIIH	CodedCharSetId
codificación	MQDLH, MQIIH	Encoding
formato	MQDLH, MQIIH	Formato
distintivos de cabecera	MQIIH, MQRMH	Flags

Referencia cruzada ImqIMSBridgeHeader

Referencia cruzada de atributos, estructuras de datos y campos para la clase ImqAuthenticationRecord C++.

Tabla 890. Correlación de atributos, estructuras de datos y campos

Atributo	Estructura de datos	Campo
autenticador	MQIIH	Authenticator
Modalidad de confirmación	MQIIH	CommitMode
Alteración temporal del terminal lógico	MQIIH	LTermOverride
Nombre de la correlación de servicios del formato del mensaje	MQIIH	MFSMapName
formato de respuesta	MQIIH	ReplyToFormat
Ámbito de seguridad	MQIIH	SecurityScope
id de instancia de transacción	MQIIH	TranInstanceId
ESTADO DE TRANSACCIÓN	MQIIH	TranState

Referencia cruzada de ImqItem

Referencia cruzada de atributos y llamadas para la clase C++ ImqItem .

Tabla 891. Atributos y llamadas	
Atributo	Llamada
ID de estructura	MQGET

Referencia cruzada de ImqMessage

Referencia cruzada de atributos, estructuras de datos, campos y llamadas para la clase C++ ImqMessage .

Tabla 892. Atributos, estructuras de datos, campos y llamadas			
Atributo	Estructura de datos	Campo	Llamada
Datos de ID de aplicación	MQMD	ApplIdentityData	
Datos de origen de aplicación	MQMD	ApplOriginData	
Recuento de restituciones	MQMD	BackoutCount	
juego de caracteres	MQMD	CodedCharSetId	
codificación	MQMD	Encoding	
caducidad	MQMD	Caducidad	
formato	MQMD	Formato	
Distintivos de mensaje	MQMD	MsgFlags	
tipo de mensaje	MQMD	MsgType	
offset	MQMD	Desplazamiento	
Longitud original	MQMD	OriginalLength	
persistence	MQMD	Persistence	
priority	MQMD	Prioridad	
Nombre de aplicación de transferencia	MQMD	PutApplName	
Tipo de aplicación de transferencia	MQMD	PutApplType	
Fecha de transferencia	MQMD	PutDate	
Hora de transferencia	MQMD	PutTime	
Nombre de gestor de cola de respuestas	MQMD	ReplyToQMgr	
Nombre de cola de respuestas	MQMD	ReplyToQ	
informe	MQMD	Informe	
número de secuencia	MQMD	MsgSeqNumber	
longitud total de mensaje		DataLength	MQGET
ID de usuario	MQMD	UserIdentifier	

ImqMessageTracker referencia cruzada

Referencia cruzada de atributos, estructuras de datos y campos para la clase ImqMessageTracker C++.

Tabla 893. Correlación de atributos, estructuras de datos y campos

Atributo	Estructura de datos	Campo
Señal de contabilidad	MQMD	AccountingToken
ID de correlación	MQMD	CorrelId
información de retorno	MQMD	Comentarios
ID de grupo	MQMD	GroupId
ID de mensaje	MQMD	MsgId

Referencia cruzada de ImqNamelist

Referencia cruzada de atributos, consultas y llamadas para la clase C++ ImqNamelist .

Tabla 894. Atributos, consultas y llamadas

Atributo	Consultar	Llamada
Número de nombres	MQIA_NAME_COUNT	MQINQ
Nombre de lista de nombres	MQCA_LISTA_DE_NOMBRES_NOMBRE	MQINQ

Referencia cruzada de ImqObject

Referencia cruzada de atributos, estructuras de datos, campos, consultas y llamadas para la clase C++ ImqObject .

Tabla 895. Atributos, estructuras de datos, campos, consultas y llamadas

Atributo	Estructura de datos	Campo	Consultar	Llamada
Fecha de modificación			MQCA_ALTERATION_DATE	MQINQ
Hora de modificación			MQCA_ALTERATION_TIME	MQINQ
ID de usuario alternativo	MQOD	AlternateUserId		
ID de seguridad alternativo				
opciones de cierre				MQCLOSE
description			MQCA_Q_DESC, MQCA_Q_MGR_DESC, MQCA_PROCESS_DESC	MQINQ
nombre	MQOD	ObjectName	MQCA_Q_MGR_NAME, MQCQ_Q_NAME, MQCA_PROCESS_NAME	MQINQ
Opciones abiertas				MQOPEN
Estado abierto				MQOPEN, MQCLOSE
identificador de gestor de colas	identificador de gestor de colas		MQCA_Q_MGR_IDENTIFIER	MQINQ

Referencia cruzada de ImqProcess

Referencia cruzada de atributos, consultas y llamadas para la clase ImqAuthenticationRecord C++.

Tabla 896. Atributos, consultas y llamadas		
Atributo	Consultar	Llamada
ID de aplicación	MQCA_APPL_ID	MQINQ
Tipo de aplicación	MQIA_APPL_TYPE	MQINQ
Datos de entorno	MQCA_ENV_DATOS	MQINQ
datos de usuario	MQCA_USER_DATA	MQINQ

ImqPutMessageOptions referencia cruzada

Referencia cruzada de atributos, estructuras de datos y campos para la clase ImqAuthenticationRecord C++.

Tabla 897. Correlación de atributos, estructuras de datos y campos		
Atributo	Estructura de datos	Campo
referencia de contexto	MQPMO	Contexto
	MQPMO	InvalidDestCount
	MQPMO	KnownDestCount
opciones	MQPMO	Opciones
campos de registro	MQPMO	PutMsgRecFields
Nombre del gestor de colas resuelto	MQPMO	ResolvedQMgrName
Nombre de cola resuelto	MQPMO	ResolvedQName
	MQPMO	Tiempo de espera
	MQPMO	UnknownDestCount
participación de punto de sincronismo	MQPMO	Opciones

Referencia cruzada de ImqQueue

Referencia cruzada de atributos, estructuras de datos, campos, consultas y llamadas para la clase C++ ImqQueue .

Tabla 898. Referencia cruzada de ImqQueue				
Atributo	Estructura de datos	Campo	Consultar	Llamada
Nombre de reposición en cola para restitución			MQCA_BACKOUT_REQ_Q_NAME	MQINQ
umbral de restituciones			Umbral MQIA_BACKOUT_THRESHOLD	MQINQ
nombre de cola base			MQCA_BASE_Q_NAME	MQINQ
nombre de clúster			NOMBRE_CLÚSTER MQCA_clúster	MQINQ

Tabla 898. Referencia cruzada de ImqQueue (continuación)

Atributo	Estructura de datos	Campo	Consultar	Llamada
Nombre de la lista de nombres del clúster			MQCA_XX_ENCODE_CASE_ONE nombre_clúster	MQINQ
Rango de carga trabajo del clúster			MQIA_CLWL_Q_RANK	MQINQ
Prioridad de carga de trabajo del clúster			MQIA_CLWL_Q_PRIORIDAD	MQINQ
Cola de uso de carga de trabajo de clúster			MQIA_CLWL_USEQ	MQINQ
fecha de creación			MQCA_CREATION_DATE	MQINQ
Hora de creación			MQCA_CREATION_TIME	MQINQ
Profundidad actual			MQIA_PROFUNDIDAD_Q_ACTUAL	MQINQ
enlace predeterminado			MQIA_DEF_BIND	MQINQ
Opción abierta de entrada predeterminada			MQIA_DEF_INPUT_OPTION_ABIERTA	MQINQ
Persistencia predeterminada			MQIA_PERSISTENCIA	MQINQ
Prioridad predeterminada			MQIA_PRIORIDAD_DEF_PRIORIDAD	MQINQ
Tipo de definición			MQIA_DEFINITION_TYPE	MQINQ
suceso de profundidad alta			MQIA_Q_PROFUNDIDAD_ALTO_EVENTO	MQINQ
límite alto de profundidad			MQIA_Q_PROFUNDIDAD_LIMITE_ALTO	MQINQ
suceso de profundidad baja			MQIA_Q_PROFUNDIDAD_BAJA_EVENTO	MQINQ
límite bajo de profundidad			MQIA_Q_PROFUNDIDAD_LIMITE_BAJO	MQINQ
suceso de profundidad máxima			MQIA_Q_PROFUNDIDAD_LIMITE_MAX	MQINQ
listas de distribución			MQIA_LISTAS_DE_DISTRIBUCIÓN	MQINQ, MQSET
Nombre de la cola dinámica	MQOD	DynamicQName		
Copia en disco de restitución de obtención			MQIA_HARDEN_GET_BACKOUT	MQINQ
Tipo de índice			TIPO_ÍNDICE	MQINQ

Tabla 898. Referencia cruzada de ImqQueue (continuación)

Atributo	Estructura de datos	Campo	Consultar	Llamada
Inhibir obtención			INHIBIDORES DE MQIA_GET	MQINQ, MQSET
Inhibir colocación			INHIBIDORES DE MQIA_PUT	MQINQ, MQSET
Nombre cola iniciación			MQCA_INICIACIÓN_NOMBRE_PREGUNTA	MQINQ
Profundidad máxima			MQIA_PROFUNDIDAD_MAX_Q	MQINQ
longitud máxima del mensaje			MQIA_MAX_LONGITUD_DEL_MENSAJE	MQINQ
Secuencia de entrega de mensajes			MQIA_MSG_SECUENCIA_DE_ENTREGA	MQINQ
siguiente cola distribuida				
Clase de mensajes no permanentes			MQIA_NPM_CLASS	MQINQ
número entradas abiertas			MQIA_RECuento_ENTRADA_ABIERTA	MQINQ
número salidas abiertas			MQIA_RECuento_SALIDA_ABIERTA	MQINQ
cola distribuida anterior				
nombre de proceso			NOMBRE_PROCESO_MQCA	MQINQ
Contabilidad de la cola			MQIA_CUENTA_Q	MQINQ
nombre del gestor de colas	MQOD	ObjectQMgrName		
Supervisión de la cola			MQIA_VIGILANCIA_Q	MQINQ
estadísticas de cola			MQIA_ESTADÍSTICAS_Q	MQINQ
Tipo de cola			MQIA_Q_TYPE	MQINQ
Nombre de gestor de colas remoto			MQCA_REMOTE_Q_MGR_NAME	MQINQ
Nombre de cola remota			MQCA_REMOTE_Q_NAME	MQINQ
Nombre del gestor de colas resuelto	MQOD	ResolvedQMgrName		
Nombre de cola resuelto	MQOD	ResolvedQName		
Intervalo de retención			MQIA_INTERVALO_RETENCIÓN	MQINQ

Tabla 898. Referencia cruzada de ImqQueue (continuación)

Atributo	Estructura de datos	Campo	Consultar	Llamada
ámbito			MQIA_ÁMBITO	MQINQ
intervalo de servicio			MQIA_Q_INTERVALO_DE_SERVICIO	MQINQ
suceso de intervalo de servicio			MQIA_Q_SERVICE_INTERVAL_EVENT	MQINQ
Posibilidad de compartición			MQIA_COMPARTIBILIDAD	MQINQ
clase de almacenamiento			CLASE_ALMACENAMIENTO_MQCA	MQINQ
Nombre de cola de transmisión			MQCA_XMIT_Q_NAME	MQINQ
Control de desencadenante			MQIA_TRIGGER_CONTROL	MQINQ, MQSET
Datos desencadenantes			MQCA_TRIGGER_DATA	MQINQ, MQSET
Profundidad de desencadenante			MQIA_PROFUNDIDAD	MQINQ, MQSET
Prioridad de mensajes desencadenantes			MQIA_TRIGGER_MSG_PRIORITY	MQINQ, MQSET
tipo de activador			MQIA_TIPO_TRIGGER_TYPE	MQINQ, MQSET
uso			MQIA_USO	MQINQ

ImqQueueManager referencia cruzada

Referencia cruzada de atributos, estructuras de datos, campos, consultas y llamadas para la clase ImqQueueManager C++.

Tabla 899. Atributos, estructuras de datos, campos, consultas y llamadas

Atributo	Estructura de datos	Campo	Consultar	Llamada
alteración temporal de conexiones de contabilidad			MQIA_CONEXIÓN_CONTABILIDAD_SOBRE SCRITURA	MQINQ
Intervalo contable			MQIA_INTERVALO_CONTABILIDAD	MQINQ
registro de actividad			MQIA_GRABACIÓN_ACTIVIDAD	MQINQ
Adoptar nueva comprobación MCA			MQIA_ADOPTAR_NUEVA_MCA_VERIFICAR	MQINQ

Tabla 899. Atributos, estructuras de datos, campos, consultas y llamadas (continuación)

Atributo	Estructura de datos	Campo	Consultar	Llamada
Adoptar nuevo tipo MCA			MQIA_ADOPTAR_NUEVO_TIPO_MCA	MQINQ
Tipo de autenticación	MQCSP	AuthenticationType		MQCONN
suceso de autorización			SUCESO_AUTORIZACIÓN_MQIA_EVENT	MQINQ
Opciones de inicio	MQBO	Opciones		MQBEGIN
suceso de puente			MQIA_SUCESO_PUENTE	MQINQ
Definición automática de canal			MQIA_CANAL_AUTO_DEF	MQINQ
suceso de definición automática de canal			MQIA_SUCESO_AUTO_CANAL	MQIA
Salida de definición automática de canal			MQIA_SALIDA_AUTO_CANAL	MQIA
suceso de canal			MQIA_SUCESO_CANAL	MQINQ
Adaptadores del iniciador de canal			MQIA_CHINIT_ADAPTADORES	MQINQ
Control del iniciador de canal			CONTROL MQIA_CHINIT_DE	MQINQ
Asignadores de tareas del iniciador de canal			MQIA_CHINIT_DISPATCHERS	MQINQ
Inicio automático del rastreo del iniciador de canal			MQIA_CHINIT_TRACE_AUTO_START	MQINQ
Tamaño de tabla de rastreo del iniciador de canal			MQIA_CHINIT_TRACE_TABLE_SIZE	MQINQ
Supervisión de canal			CANAL_DE_VIGILANCIA_MQIA	MQINQ
referencia de canal	MQCD	ChannelType		MQCONN
Estadísticas del canal			CANAL_ESTADISTICAS_MQIA	MQINQ
juego de caracteres			MQIA_CODED_CHAR_SET_ID	MQINQ

Tabla 899. Atributos, estructuras de datos, campos, consultas y llamadas (continuación)				
Atributo	Estructura de datos	Campo	Consultar	Llamada
Supervisión de clúster emisor			MQIA_MONITORING_AUTO_CLUSSDR	MQINQ
Estadística de clúster emisor			MQIA_STATISTICS_AUTO_CLUSSDR	MQINQ
Datos de carga de trabajo de clúster			MQCA_CLUSTER_WORKLOAD_DATA	MQINQ
salida de carga de trabajo de clúster			MQCA_CLUSTER_WORKLOAD_EXIT	MQINQ
Longitud de la carga de trabajo de clúster			MQIA_CLUSTER_WORKLOAD_LENGTH	MQINQ
mru de carga de trabajo de clúster			MQIA_CLWL_MRU_CANALES	MQINQ
Cola de uso de carga de trabajo de clúster			MQIA_CLWL_USEQ	MQINQ
suceso de mandato			MQIA_SUCESO_MANDATO	MQINQ
Nombre de cola de entrada de mandatos			MQCA_COMMAND_INPUT_Q_NAME	MQINQ
Nivel de mandato			MQIA_COMMAND_LEVEL	MQINQ
Control del servidor de mandatos			MQIA_CMD_SERVER_CONTROL	MQINQ
Opciones de conexión	MQCNO	Opciones		MQCONN, MQCONNX
ID de conexión	MQCNO	ConnectionId		MQCONNX
estado de conexión				MQCONN, MQCONNX, MQDISC
etiqueta de conexión	MQCD	ConnTag		MQCONNX
hardware de cifrado	MQSCO	CryptoHardware		MQCONNX
nombre de cola de mensajes no entregados			MQCA_CARTA_MUERTA_Q_NOMBRE	MQINQ
nombre de cola de transmisión predeterminada			MQCA_DEF_XMIT_Q_NAME	MQINQ

Tabla 899. Atributos, estructuras de datos, campos, consultas y llamadas (continuación)

Atributo	Estructura de datos	Campo	Consultar	Llamada
listas de distribución			MQIA_LISTAS_DE_DISTRIBUCIÓN	MQINQ
grupo dns			MQCA_DNS_GROUP	MQINQ
DNS WLM			MQIA_DNS_WLM	MQINQ
primer registro de autenticación	MQSCO	AuthInfoRecOffset		MQCONN
	MQSCO	AuthInfoRecPtr		MQCONN
suceso de inhibición			SUCESO INHIBIDOR DE MQIA_ENT	MQINQ
Versión de dirección IP			MQIA_DIRECCIÓN_IP_VERSIÓN	MQINQ
repositorio de claves	MQSCO	KeyRepository		MQCONN
recuento de restablecimiento de clave	MQSCO	KeyResetCount		MQCONN
Temporizador de escucha			MQIA_LISTENER_TIMER	MQINQ
suceso local			MQIA_SUCESO_LOCAL	MQINQ
Suceso de registrador			MQIA_LOGGER_EVENT	MQINQ
Nombre de grupo LU			MQCA_LU_XX_ENCODE_CASE_ONE nombre_grupo	MQINQ
Nombre de LU			MQCA_LU_NAME	MQINQ
Sufijo de brazo lu62			MQCA_LU62_ARM_SUFFIX	MQINQ
Canales lu62			MQIA_LU62_CHANNELS	MQINQ
máximo de canales activos			MQIA_CANALES_ACTIVOS	MQINQ
Canales máximos			MQIA_MÁX_CANALES	MQINQ
Número máximo de descriptores			MQIA_MÁX_DESCRIPTORES de contexto	MQINQ
longitud máxima del mensaje			MQIA_MAX_LONGITUD_DEL_MENSAJE	MQINQ
Prioridad máxima			MQIA_PRIORIDAD_MÁX	MQINQ
Máx. mensajes no confirmados			MQIA_MAX_MENSAJES_SIN_CONFIRMAR	MQINQ
Contabilidad de MQI			MQIA_CUENTA_MQI	MQINQ

Tabla 899. Atributos, estructuras de datos, campos, consultas y llamadas (continuación)

Atributo	Estructura de datos	Campo	Consultar	Llamada
Estadísticas MQI			MQIA_ESTADÍSTICAS_MQI	MQINQ
máximo de puerto de salida			MQIA_PUERTO_SALIDA máx	MQINQ
mínimo de puerto de salida			MQIA_PUERTO_SALIDA min	MQINQ
contraseña	MQCSP	CSPPasswordPtr		MQCONN
	MQCSP	CSPPasswordOffset		MQCONN
	MQCSP	CSPPasswordLength		MQCONN
suceso de rendimiento			MQIA_SUCESO_RENDIMIENTO	MQINQ
platform			MQIA_PLATFORM	MQINQ
Contabilidad de la cola			MQIA_CUENTA_Q	MQINQ
Supervisión de la cola			MQIA_VIGILANCIA_Q	MQINQ
estadísticas de cola			MQIA_ESTADÍSTICAS_Q	MQINQ
Tiempo de espera de recepción			MQIA_TIEMPO_ESPERA	MQINQ
tiempo de espera de recepción mínimo			MQIA_RECEIVE_TIMEOUT_MIN	MQINQ
Tipo de tiempo de espera de recepción			MQIA_RECEIVE_TIMEOUT_TYPE	MQINQ
suceso remoto			MQIA_REMOTE_EVENT	MQINQ
Nombre de depósito			NOMBRE_REPOSITORIO_MQCA	MQINQ
Lista nombres repositorio			MQCA_LISTA_DE_NOMBRES_DEL_REPOSITORIO	MQINQ
nombre de gestor de colas compartido			MQIA_SHARED_Q_Q_MGR_NAME	MQINQ
suceso ssl			MQIA_SSL_EVENT	MQINQ
Fips ssl			MQIA_SSL_FIPS_REQUIRED	MQINQ
Cuenta restablecimiento clave SSL			MQIA_SSL_RESET_COUNT	MQINQ

Tabla 899. Atributos, estructuras de datos, campos, consultas y llamadas (continuación)				
Atributo	Estructura de datos	Campo	Consultar	Llamada
suceso de inicio-detención			MQIA_INICIO_FIN_EVENTO	MQINQ
Intervalo de estadísticas			MQIA_INTERVALO_ESTADÍSTICAS	MQINQ
Disponibilidad de punto de sincronismo			MQIA_SYNCPOINT	MQINQ
canales tcp			MQIA_TCP_CANALES	MQINQ
Keep alive de TCP			MQIA_TCP_KEEP_ALIVE	MQINQ
Nombre TCP			MQCA_TCP_NAME	MQINQ
Tipo de pila TCP			MQIA_TCP_STACK_TYPE	MQINQ
Registro de la ruta de rastreo			MQIA_GRABACIÓN_RUTA_RASTREO	MQINQ
Activar intervalo			MQIA_INTERVALO_DESENCADENANTE	MQINQ
ID de usuario	MQCSP	CSPUserIdPtr		MQCONN
	MQCSP	CSPUserIdOffset		MQCONN
	MQCSP	CSPUserIdLength		MQCONN

ImqReferenceHeader referencia cruzada

Referencia cruzada de atributos, estructuras de datos y campos para la clase ImqAuthenticationRecord C++.

Tabla 900. Correlación de atributos, estructuras de datos y campos		
Atributo	Estructura de datos	Campo
entorno de destino	MQRMH	DestEnvLength, DestEnvOffset
nombre de destino	MQRMH	DestNameLength, DestNameOffset
ID de instancia	MQRMH	ObjectInstanceId
longitud lógica	MQRMH	DataLogicalLength
desplazamiento lógico	MQRMH	DataLogicalOffset
desplazamiento lógico 2	MQRMH	DataLogicalOffset2
Tipo de referencia	MQRMH	ObjectType
Entorno de origen	MQRMH	SrcEnvLength, SrcEnvOffset
nombre de origen	MQRMH	SrcNameLength, SrcNameOffset

Referencia cruzada de ImqTrigger

Referencia cruzada de atributos, estructuras de datos y campos para la clase ImqAuthenticationRecord C++.

Tabla 901. Correlación de atributos, estructuras de datos y campos		
Atributo	Estructura de datos	Campo
ID de aplicación	MQTM	ApplId
Tipo de aplicación	MQTM	ApplType
Datos de entorno	MQTM	EnvData
nombre de proceso	MQTM	ProcessName
nombre de cola	MQTM	QName
Datos desencadenantes	MQTM	TriggerData
datos de usuario	MQTM	UserData

ImqWorkHeader referencia cruzada

Referencia cruzada de atributos, estructuras de datos y campos para la clase ImqAuthenticationRecord C++.

Tabla 902. Correlación de atributos, estructuras de datos y campos		
Atributo	Estructura de datos	Campo
token de mensaje	MQWIH	MessageToken
nombre del servicio	MQWIH	ServiceName
paso de servicio	MQWIH	ServiceStep

ImqAuthenticationRecord Clase C

Esta clase encapsula un registro de información de autenticación (MQAIR) que se utiliza durante la ejecución del método `ImqQueueManager::connect`, para conexiones de clientes personalizadas de TLS.

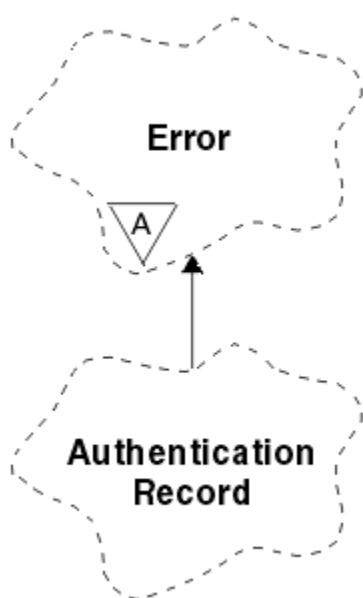


Figura 13. ImqAuthenticationRecord clase

Consulte la descripción del método `ImqQueueManager::connect` para más detalles. Esta clase no está disponible en la plataforma z/OS.

- [“Atributos de objetos” en la página 1832](#)
- [“Constructores” en la página 1832](#)
- [“Métodos de objeto \(public\)” en la página 1832](#)
- [“Métodos de objeto \(protegidos\)” en la página 1833](#)

Atributos de objetos

Nombre de conexión

El nombre de la conexión al servidor CRL de LDAP. Es la dirección IP o el nombre DNS, seguido opcionalmente por el número de puerto, entre paréntesis.

referencia de conexión

Una referencia a un objeto `ImqQueueManager` que proporciona la conexión necesaria a un gestor de colas (local). El valor inicial es cero. No confunda esto con el nombre del gestor de colas que identifica un gestor de colas (posiblemente remoto) para una cola con nombre.

siguiente registro de autenticación

Siguiente objeto de esta clase, en ningún orden concreto, que tenga la misma **referencia de conexión** que este objeto. El valor inicial es cero.

contraseña

Una contraseña proporcionada para la autenticación de la conexión al servidor CRL de LDAP.

registro de autenticación anterior

Objeto anterior de esta clase, en ningún orden concreto, que tenga la misma **referencia de conexión** que este objeto. El valor inicial es cero.

Tipo

El tipo de información de autenticación contenida en el registro.

nombre de usuario

Un identificador de usuario facilitado para la autorización al servidor CRL de LDAP.

Constructores

`ImqAuthenticationRecord()`;

El constructor predeterminado.

Métodos de objeto (public)

`void operador = (const ` ImqAuthenticationRecord ` & air );`

Copia datos de instancia de *air*, sustituyendo los datos de instancia existentes.

`const ImqString & connectionName () Const.`

Devuelve el **nombre de conexión**.

`void setConnectionName (const ImqString & nombre );`

Establece el **nombre de conexión**.

`void setConnectionName ( const char * nombre = 0 );`

Establece el **nombre de conexión**.

`ImqQueueManager * connectionReference () const;`

Devuelve la **referencia de conexión**.

`void setConnectionReference ( ImqQueueManager &manager );`

Establece la **referencia de conexión**.

`void setConnectionReference ( ImqQueueManager *manager = 0);`

Establece la **referencia de conexión**.

`void copyOut (MQAIR * pAir );`

Copia los datos de instancia en *pAir*, sustituyendo los datos de instancia existentes. Esto puede implicar la asignación de almacenamiento dependiente.

void clear (MQAIR * pAir);

Borra la estructura y libera el almacenamiento dependiente al que hace referencia *pAir*.

ImqAuthenticationRecord * nextAuthenticationRecord () const;

Devuelve el **siguiente registro de autenticación**.

const ImqString & password () Const.

Devuelve la **contraseña**.

void setPassword (const ImqString & contraseña);

Establece la **contraseña**.

void setPassword (const char * contraseña = 0);

Establece la **contraseña**.

ImqAuthenticationRecord * previousAuthenticationRecord () const;

Devuelve el **registro de autenticación anterior**.

Tipo MQLONG () Const.

Devuelve el **tipo**.

void setType (const MQLONG tipo);

Establece el **tipo**.

const ImqString & userName () Const.

Devuelve el **nombre de usuario**.

void setUsername (const ImqString & nombre);

Establece el **nombre de usuario**.

void setUsername (const char * nombre = 0);

Establece el **nombre de usuario**.

Métodos de objeto (protegidos)

void setNextAuthenticationRecord (ImqAuthenticationRecord * pAir = 0);

Establece el **siguiente registro de autenticación**.

Atención: Utilice esta función sólo si está seguro de que no romperá la lista de registros de autenticación.

void setPreviousAuthenticationRecord (ImqAuthenticationRecord * pAir = 0);

Establece el **registro de autenticación anterior**.

Atención: Utilice esta función sólo si está seguro de que no romperá la lista de registros de autenticación.

Clase C++ ImqBinary

Esta clase encapsula una matriz de bytes binarios que se puede utilizar para los valores ImqMessage **señal de contabilidad**, **id de correlación** y **id de mensaje** . Permite una fácil asignación, copia y comparación.

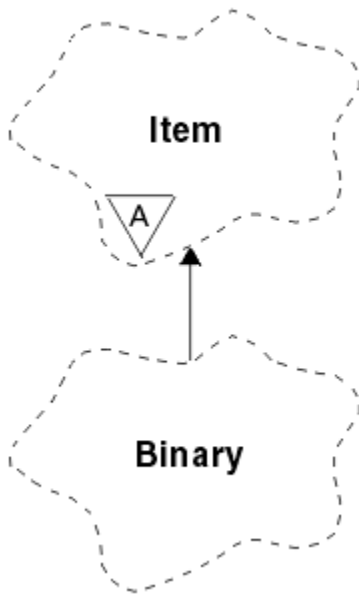


Figura 14. Clase *ImqBinary*

- [“Atributos de objetos” en la página 1834](#)
- [“Constructores” en la página 1834](#)
- [“Métodos ImqItem sobrecargados” en la página 1834](#)
- [“Métodos de objeto \(public\)” en la página 1835](#)
- [“Métodos de objeto \(protegidos\)” en la página 1835](#)
- [“códigos de razón” en la página 1835](#)

Atributos de objetos

datos

Una matriz de bytes de datos binarios. El valor inicial es nulo.

Longitud de datos

Número de bytes. El valor inicial es cero.

puntero de datos

La dirección del primer byte de los **datos**. El valor inicial es cero.

Constructores

ImqBinary();

El constructor predeterminado.

ImqBinary(const ImqBinary, *binary*);

El constructor de copia.

ImqBinary(const void * *data*, const size_t *length*);

Copia *longitud* bytes de *datos*.

Métodos ImqItem sobrecargados

virtual ImqBoolean copyOut (ImqMessage & *msg*);

Copia los **datos** en el almacenamiento intermedio de mensajes, sustituyendo cualquier contenido existente. Establece el formato *msg* en MQFMT_NONE.

Consulte la descripción del método de clase ImqItem para obtener más detalles.

virtual ImqBoolean pasteIn (ImqMessage & msg);

Establece los **datos** transfiriendo los datos restantes del almacenamiento intermedio de mensajes, sustituyendo los **datos** existentes.

Para ser satisfactorio, el **formato** ImqMessage debe ser MQFMT_NONE.

Consulte la descripción del método de clase ImqItem para obtener más detalles.

Métodos de objeto (public)**void operator = (const ImqBinary & binario);**

Copia bytes de *binario*.

ImqBoolean operator == (const ImqBinary & binario);

Compara este objeto con *binario*. Devuelve FALSE si no es igual y TRUE de lo contrario. Los objetos son iguales si tienen la misma **longitud de datos** y los bytes coinciden.

ImqBoolean copyOut (void * buffer, const size_t length, const char pad = 0);

Copia hasta *longitud* bytes del **puntero de datos** al *almacenamiento intermedio*. Si la **longitud de datos** es insuficiente, el espacio restante del *almacenamiento intermedio* se rellena con *relleno* bytes. *buffer* puede ser cero si *length* también es cero. *length* no debe ser negativo. Devuelve TRUE si es satisfactorio.

size_t dataLength () const ;

Devuelve la **longitud de datos**.

ImqBoolean setDataLength (const size_t length);

Establece la **longitud de datos**. Si la **longitud de datos** se cambia como resultado de este método, los datos del objeto no se inicializan. Devuelve TRUE si es satisfactorio.

void * dataPointer () const ;

Devuelve el **puntero de datos**.

ImqBoolean isNull () const ;

Devuelve TRUE si la **longitud de datos** es cero, o si todos los **datos** bytes son cero. De lo contrario, devuelve FALSE.

ImqBoolean set (const void * buffer, const size_t longitud);

Copia *longitud* bytes del *almacenamiento intermedio*. Devuelve TRUE si es satisfactorio.

Métodos de objeto (protegidos)**void clear ();**

Reduce la **longitud de datos** a cero.

códigos de razón

- MQRC_SIN_BUFFER
- MQRC_STORAGE_NOT_AVAILABLE
- MQRC_FORMATO_INCONSISTENTE

Clase C++ ImqCache

Utilice esta clase para mantener o ordenar datos en la memoria.

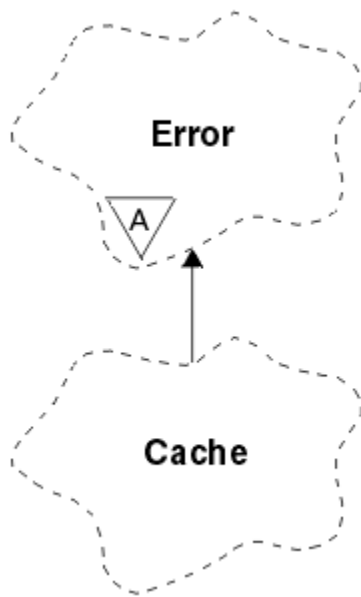


Figura 15. Clase ImqCache

Utilice esta clase para mantener o ordenar datos en la memoria. Puede nombrar un almacenamiento intermedio de memoria de tamaño fijo, o el sistema puede proporcionar una cantidad flexible de memoria automáticamente. Esta clase está relacionada con las llamadas de la clase `MQI` que se enumeran en [“Referencia cruzada de ImqCache” en la página 1816](#).

- [“Atributos de objetos” en la página 1836](#)
- [“Constructores” en la página 1837](#)
- [“Métodos de objeto \(public\)” en la página 1837](#)
- [“códigos de razón” en la página 1838](#)

Atributos de objetos

almacenamiento intermedio automático

Indica si la memoria de almacenamiento intermedio la gestiona automáticamente el sistema (TRUE) o la proporciona el usuario (FALSE). Inicialmente se establece en TRUE.

Este atributo no se ha establecido directamente. Se establece indirectamente mediante los botones **useEmptyBuffer** o el método **useFullBuffer** método.

Si se proporciona almacenamiento de usuario, este atributo es FALSE, la memoria de almacenamiento intermedio no puede crecer y se pueden producir errores de desbordamiento de almacenamiento intermedio. La dirección y la longitud del almacenamiento intermedio permanecen constantes.

Si no se proporciona almacenamiento de usuario, este atributo es TRUE, y la memoria de almacenamiento intermedio puede crecer de forma incremental para acomodar una cantidad arbitraria de datos de mensaje. Sin embargo, cuando el almacenamiento intermedio crece, la dirección del almacenamiento intermedio puede cambiar, por lo que debe tener cuidado al utilizar el **puntero de almacenamiento intermedio** y el **puntero de datos**.

Longitud de almacenamiento intermedio

Número de bytes de memoria en el almacenamiento intermedio. El valor inicial es cero.

puntero de almacenamiento intermedio

La dirección de la memoria de almacenamiento intermedio. El valor inicial es nulo.

Longitud de datos

El número de bytes que suceden al **puntero de datos**. Debe ser igual o menor que la **longitud de mensaje**. El valor inicial es cero.

Desplazamiento de datos

Número de bytes que preceden al **puntero de datos**. Debe ser igual o menor que la **longitud de mensaje**. El valor inicial es cero.

puntero de datos

La dirección de la parte del almacenamiento intermedio que se va a escribir o leer en la siguiente. El valor inicial es nulo.

Longitud del mensaje

Número de bytes de datos significativos en el almacenamiento intermedio. El valor inicial es cero.

Constructores

ImqCache();

El constructor predeterminado.

ImqCache(const ImqCache &cache);

El constructor de copia.

Métodos de objeto (public)

void operador = (const ` ImqCache ` & ` cache `);

Copia hasta **longitud de mensaje** bytes de datos del objeto *cache* en el objeto. Si **almacenamiento intermedio automático** es FALSE, la **longitud de almacenamiento intermedio** ya debe ser suficiente para acomodar los datos copiados.

ImqBoolean automaticBuffer () const ;

Devuelve el valor de **almacenamiento intermedio automático** .

size_t bufferSize () const ;

Devuelve la **longitud de almacenamiento intermedio**.

char * bufferPointer () const ;

Devuelve el **puntero de almacenamiento intermedio**.

void clearMessage ();

Establece la **longitud de mensaje** y el **desplazamiento de datos** en cero.

size_t dataLength () const ;

Devuelve la **longitud de datos**.

size_t dataOffset () const ;

Devuelve el **desplazamiento de datos**.

ImqBooleansetDataOffset (const size_t offset);

Establece el **desplazamiento de datos**. La **longitud de mensaje** se incrementa si es necesario para asegurarse de que no es menor que el **desplazamiento de datos**. Este método devuelve TRUE si es satisfactorio.

char * dataPointer () const ;

Devuelve una copia del **puntero de datos**.

size_t messageLength () const ;

Devuelve la **longitud de mensaje**.

ImqBooleansetMessageLength (const size_t length);

Establece la **longitud de mensaje**. Aumenta la **longitud del almacenamiento intermedio** si es necesario para asegurarse de que la **longitud del mensaje** no es mayor que la **longitud del almacenamiento intermedio**. Reduce el **desplazamiento de datos** si es necesario para asegurarse de que no es mayor que la **longitud de mensaje**. Devuelve TRUE si es satisfactorio.

ImqBoolean moreBytes (const size_t bytes-necesario);

Asegura que *bytes-necesarios* estén disponibles más bytes (para escritura) entre el **puntero de datos** y el final del almacenamiento intermedio. Devuelve TRUE si es satisfactorio.

Si **almacenamiento intermedio automático** es TRUE, se adquiere más memoria según sea necesario; de lo contrario, la **longitud del almacenamiento intermedio** ya debe ser adecuada.

ImqBoolean read (const size_t longitud, char * & almacenamiento intermedio externo);

Copia *length* bytes, desde el almacenamiento intermedio que empieza en la posición de **puntero de datos** , en el *external-buffer*. Una vez copiados los datos, el **desplazamiento de datos** se incrementa en *longitud*. Este método devuelve TRUE si es satisfactorio.

ImqBoolean resizeBuffer (const size_t longitud);

Varía la **longitud del almacenamiento intermedio**, siempre que el **almacenamiento intermedio automático** sea TRUE. Esto se consigue reasignando la memoria intermedia. Hasta **longitud de mensaje** bytes de datos del almacenamiento intermedio existente se copian en el nuevo. El número máximo copiado es de *longitud* bytes. Se cambia el **puntero de almacenamiento intermedio** . La **longitud de mensaje** y el **desplazamiento de datos** se conservan tan cerca como sea posible dentro de los confines del nuevo almacenamiento intermedio. Devuelve TRUE si es satisfactorio, y FALSE si **almacenamiento intermedio automático** es FALSE.

Nota: Este método puede fallar con MQRC_STORAGE_NOT_AVAILABLE si hay algún problema con los recursos del sistema.

ImqBoolean useEmptyBuffer (const char * buffer-externo, const size_t longitud);

Identifica un almacenamiento intermedio de usuario vacío, estableciendo el **puntero de almacenamiento intermedio** para que apunte a *external-buffer*, la **longitud de almacenamiento intermedio** a *length* y la **longitud de mensaje** a cero. Realiza un **clearMessage**. Si el búfer está completamente lleno de datos, utilice el método **useFullBuffer** en su lugar. Si el búfer está parcialmente cebado con datos, utilice el método **setMessageLength** para indicar la cantidad correcta. Este método devuelve TRUE si es satisfactorio.

Este método se puede utilizar para identificar una cantidad fija de memoria, como se ha descrito anteriormente (*almacenamiento intermedio externo* no es nulo y la *longitud* no es cero), en cuyo caso el **almacenamiento intermedio automático** se establece en FALSE, o se puede utilizar para revertir a la memoria flexible gestionada por el sistema (*almacenamiento intermedio externo* es nulo y la *longitud* es cero), en cuyo caso el **almacenamiento intermedio automático** se establece en TRUE.

ImqBoolean useFullBuffer (const char * externalBuffer, const size_t length);

Como para **useEmptyBuffer**, excepto que la **longitud del mensaje** se establece en *length*. Devuelve TRUE si es satisfactorio.

ImqBoolean write (const size_t longitud, const char * búfer externo);

Copia *longitud* bytes, del *almacenamiento intermedio externo*, en el almacenamiento intermedio empezando en la posición **puntero de datos** . Una vez copiados los datos, el **desplazamiento de datos** se incrementa en *longitud*, y la **longitud de mensaje** se incrementa si es necesario para asegurarse de que no es menor que el nuevo valor de **desplazamiento de datos** . Este método devuelve TRUE si es satisfactorio.

Si **almacenamiento intermedio automático** es TRUE, se garantiza una cantidad adecuada de memoria; de lo contrario, el **desplazamiento de datos** final no debe superar la **longitud del almacenamiento intermedio**.

códigos de razón

- MQRC_BUFFER_NOT_AUTOMATIC
- MQRC_DATA_TRUNCADO
- MQRC_INSUFICIENT_BUFFER
- MQRC_INSUFICIENT_DATA
- MQRC_PUNTERO_NULO
- MQRC_STORAGE_NOT_AVAILABLE
- MQRC_LONGITUD_CERO

Clase C++ ImqChannel

Esta clase encapsula una definición de canal (MQCD) para su uso durante la ejecución del método Manager: :connect, para conexiones de cliente personalizadas.

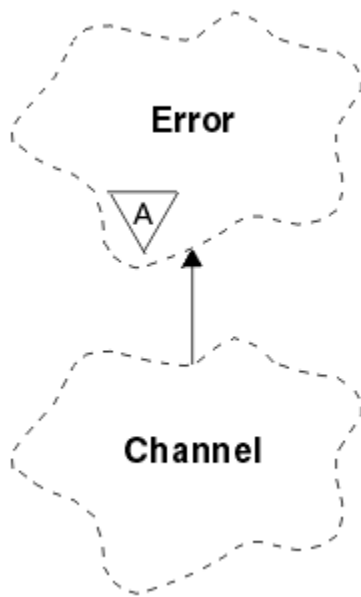


Figura 16. Clase *ImqChannel*

Consulte la descripción del método Manager: `:connect` y [Programa de ejemplo HELLO WORLD \(imqworld.cpp\)](#), para obtener más detalles.

No todos los métodos listados son aplicables a todas las plataformas. Consulte las descripciones de los mandatos [DEFINE CHANNEL](#) y [ALTER CHANNEL](#) para obtener más información.

La clase *ImqChannel* no está soportada en z/OS.

- “Atributos de objetos” en la [página 1839](#)
- “Constructores” en la [página 1840](#)
- “Métodos de objeto (public)” en la [página 1840](#)
- “códigos de razón” en la [página 1844](#)

Atributos de objetos

latido del corazón por lotes

El número de milisegundos entre comprobaciones de que un canal remoto está activo. El valor inicial es 0.

nombre de canal

El nombre del canal. El valor inicial es nulo.

Nombre de conexión

El nombre de la conexión. Por ejemplo, la dirección IP de un sistema host. El valor inicial es nulo.

Compresión de cabecera

La lista de métodos de compresión de datos de cabecera que el canal admite. Todos los valores iniciales se establecen en `MQCOMPRESS_NOT_AVAILABLE`.

intervalo de latido cardíaco

Número de segundos entre comprobaciones de que una conexión sigue funcionando. El valor inicial es 300.

Intervalo de mantenimiento activado

Número de segundos pasados a la pila de comunicaciones especificando la temporización de mantener activo para el canal. El valor inicial es `MQKAI_AUTO`.

dirección local

La dirección de comunicaciones locales para el canal.

longitud máxima del mensaje

Longitud máxima de mensaje soportada por el canal en una sola comunicación. El valor inicial es 4 194 304.

Compresión de mensaje

La lista de métodos de compresión de datos de mensaje que el canal admite. Todos los valores iniciales se establecen en MQCOMPRESS_NOT_AVAILABLE.

nombre modalidad

El nombre de la modalidad. El valor inicial es nulo.

contraseña

Una contraseña proporcionada para la autenticación de conexión. El valor inicial es nulo.

recuento de salidas de recepción

Número de salidas de recepción. El valor inicial es cero. Este atributo es de sólo lectura.

nombres de salida de recepción

Los nombres de las salidas de recepción.

recibir datos de usuario

Datos asociados con salidas de recepción.

Nombre de salida de seguridad

El nombre de una salida de seguridad que se invocará en el lado del servidor de la conexión. El valor inicial es nulo.

datos de usuario de seguridad

Datos que deben pasarse a la salida de seguridad. El valor inicial es nulo.

recuento de salidas de envío

Número de salidas de envío. El valor inicial es cero. Este atributo es de sólo lectura.

nombres de salida de envío

Los nombres de las salidas de envío.

enviar datos de usuario

Datos asociados con salidas de envío.

SSL CipherSpec

CipherSpec para usar con TLS.

SSL tipo de autenticación del cliente

Tipo de autenticación de cliente para su uso con TLS.

Nombre de igual SSL

Nombre de nodo para su uso con TLS.

nombre del programa de transacciones

El nombre del programa de transacción. El valor inicial es nulo.

tipo de transporte

El tipo de transporte de la conexión. El valor inicial es MQXPT_LU62.

ID de usuario

Identificador de usuario proporcionado para la autorización. El valor inicial es nulo.

Constructores**ImqChannel() ;**

El constructor predeterminado.

ImqChannel(const ImqChannel, channel);

El constructor de copia.

Métodos de objeto (public)**void operator = (const ImqChannel & canal);**

Copia datos de instancia del *canal*, sustituyendo los datos de instancia existentes.

MQLONG batchHeartBeat() const;

Devuelve el **latido por lotes**.

ImqBoolean setBatchHeartBeat(const MQLONG latido = 0L);

Establece el **latido por lotes**. Este método devuelve TRUE si es satisfactorio.

ImqString channelName() const;

Devuelve el **nombre de canal**.

ImqBoolean setChannelName(const char * nombre = 0);

Establece el **nombre de canal**. Este método devuelve TRUE si es satisfactorio.

ImqString connectionName() const;

Devuelve el **nombre de conexión**.

ImqBoolean setConnectionName(const char * nombre = 0);

Establece el **nombre de conexión**. Este método devuelve TRUE si es satisfactorio.

size_t headerCompressionCount() const;

Devuelve el recuento de técnicas de compresión de datos de cabecera soportadas.

ImqBoolean headerCompression(const size_t count, MQLONG compress []) const;

Devuelve copias de las técnicas de compresión de datos de cabecera soportadas en **compress**. Este método devuelve TRUE si es satisfactorio.

ImqBoolean setHeaderCompression(const size_t count, const MQLONG compress[]);

Establece las técnicas de compresión de datos de cabecera soportadas en **compress**.

Establece el recuento de las técnicas de compresión de datos de cabecera soportadas en **recuento**.

Este método devuelve TRUE si es satisfactorio.

MQLONG heartBeatInterval() const;

Devuelve el **intervalo de latido**.

ImqBoolean setHeartBeatInterval(const MQLONG interval = 300L);

Establece el **intervalo de latido**. Este método devuelve TRUE si es satisfactorio.

MQLONG keepAliveInterval() const;

Devuelve el **intervalo de mantener activo**.

ImqBoolean setKeepAliveInterval(const MQLONG interval = MQKAI_AUTO);

Establece el **intervalo de mantener activo**. Este método devuelve TRUE si es satisfactorio.

ImqString localAddress() const;

Devuelve la **dirección local**.

ImqBoolean setLocalAddress(const char * dirección = 0);

Establece la **dirección local**. Este método devuelve TRUE si es satisfactorio.

MQLONG maximumMessageLength() const;

Devuelve la **longitud máxima de mensaje**.

ImqBoolean setMaximumMessageLength(const MQLONG longitud = 4194304L);

Establece la **longitud máxima de mensaje**. Este método devuelve TRUE si es satisfactorio.

size_t messageCompressionCount() const;

Devuelve el recuento de técnicas de compresión de datos de mensajes soportadas.

ImqBoolean messageCompression(const size_t count, MQLONG compress []) const;

Devuelve copias de las técnicas de compresión de datos de mensaje soportadas en **comprimir**. Este método devuelve TRUE si es satisfactorio.

ImqBoolean setMessageCompression(const size_t count, const MQLONG compress[]);

Establece las técnicas de compresión de datos de mensaje soportadas para comprimir.

Establece el recuento de las técnicas de compresión de datos de mensajes soportadas.

Este método devuelve TRUE si es satisfactorio.

ImqString modeName() const;

Devuelve el **nombre de modalidad**.

ImqBoolean setModeName(const char * *nombre* = 0);

Establece el **nombre de modalidad**. Este método devuelve TRUE si es satisfactorio.

ImqString contraseña () Const.

Devuelve la **contraseña**.

ImqBoolean setPassword(const char * *contraseña* = 0);

Establece la **contraseña**. Este método devuelve TRUE si es satisfactorio.

size_t receiveExitCount() const;

Devuelve el **recuento de salidas de recepción**.

ImqString receiveExitName();

Devuelve el primero de los **nombres de salida de recepción**, si los hay. Si el **recuento de salidas de recepción** es cero, devuelve una serie vacía.

ImqBoolean receiveExitNames(const size_t *count*, ImqString * *nombres* []);

Devuelve copias de los **nombres de salida de recepción** en *nombres*. Establece los *nombres* que exceden el **recuento de salidas de recepción** en series nulas. Este método devuelve TRUE si es satisfactorio.

ImqBoolean setReceiveExitName(const char * *nombre* = 0);

Establece los **nombres de salida de recepción** en el único *nombre*. *name* puede estar en blanco o ser nulo. Establece el **recuento de salidas de recepción** en 1 o cero. Borra **recibir datos de usuario**. Este método devuelve TRUE si es satisfactorio.

ImqBoolean setReceiveExitNames(const size_t *count*, const char * *names* []);

Establece los **nombres de salida de recepción** en *nombres*. Los valores de *nombres* individuales no deben estar en blanco ni ser nulos. Establece el **recuento de salidas de recepción** en *recuento*. Borra **recibir datos de usuario**. Este método devuelve TRUE si es satisfactorio.

ImqBoolean setReceiveExitNames(const size_t *count*, const ImqString * *names* []);

Establece los **nombres de salida de recepción** en *nombres*. Los valores de *nombres* individuales no deben estar en blanco ni ser nulos. Establece el **recuento de salidas de recepción** en *recuento*. Borra **recibir datos de usuario**. Este método devuelve TRUE si es satisfactorio.

ImqString receiveUserData();

Devuelve el primero de los elementos **recibir datos de usuario**, si los hay. Si el **recuento de salidas de recepción** es cero, devuelve una serie vacía.

ImqBoolean receiveUserData(const size_t *count*, ImqString * *data* []);

Devuelve copias de los elementos **recibir datos de usuario** en *datos*. Establece cualquier *dato* que supere el **recuento de salidas de recepción** en series nulas. Este método devuelve TRUE si es satisfactorio.

ImqBoolean setReceiveUserData(const char * *datos* = 0);

Establece los **datos de usuario de recepción** en el elemento único *datos*. Si *data* no es nulo, el **recuento de salidas de recepción** debe ser como mínimo 1. Este método devuelve TRUE si es satisfactorio.

ImqBoolean setReceiveUserData(const size_t *count*, const char * *data* []);

Establece los **datos de usuario de recepción** en *datos*. *count* no debe ser mayor que el **recuento de salidas de recepción**. Este método devuelve TRUE si es satisfactorio.

ImqBoolean setReceiveUserData(const size_t *count*, const ImqString * *data* []);

Establece los **datos de usuario de recepción** en *datos*. *count* no debe ser mayor que el **recuento de salidas de recepción**. Este método devuelve TRUE si es satisfactorio.

ImqString securityExitName() const;

Devuelve el **nombre de salida de seguridad**.

ImqBoolean setSecurityExitName(const char * *nombre* = 0);

Establece el **nombre de salida de seguridad**. Este método devuelve TRUE si es satisfactorio.

ImqString securityUserData() const;

Devuelve los **datos de usuario de seguridad**.

ImqBoolean setSecurityUserData(const char * *datos* = 0);

Establece los **datos de usuario de seguridad**. Este método devuelve TRUE si es satisfactorio.

size_t sendExitCount() const;

Devuelve el **recuento de salidas de envío**.

ImqString sendExitName();

Devuelve el primero de los **nombres de salida de envío**, si los hay. Devuelve una serie vacía si el **recuento de salidas de envío** es cero.

ImqBoolean sendExitNames(const size_t count, ImqString * nombres []);

Devuelve copias de los **nombres de salida de envío** en *nombres*. Establece los *nombres* que excedan el **recuento de salidas de envío** en series nulas. Este método devuelve TRUE si es satisfactorio.

ImqBoolean setSendExitName(const char * nombre = 0);

Establece los **nombres de salida de envío** en el único *nombre*. *name* puede estar en blanco o ser nulo. Establece el **recuento de salidas de envío** en 1 o cero. Borra los **datos de usuario de envío**. Este método devuelve TRUE si es satisfactorio.

ImqBoolean setSendExitNames(const size_t count, const char * names []);

Establece los **nombres de salida de envío** en *nombres*. Los valores de *nombres* individuales no deben estar en blanco ni ser nulos. Establece el **recuento de salidas de envío** en *recuento*. Borra los **datos de usuario de envío**. Este método devuelve TRUE si es satisfactorio.

ImqBoolean setSendExitNames(const size_t count, const ImqString * names []);

Establece los **nombres de salida de envío** en *nombres*. Los valores de *nombres* individuales no deben estar en blanco ni ser nulos. Establece el **recuento de salidas de envío** en *recuento*. Borra los **datos de usuario de envío**. Este método devuelve TRUE si es satisfactorio.

ImqString sendUserData();

Devuelve el primero de los elementos **enviar datos de usuario**, si los hay. Devuelve una serie vacía si el **recuento de salidas de envío** es cero.

ImqBoolean sendUserData(const size_t count, ImqString * data []);

Devuelve copias de los elementos **enviar datos de usuario** en *datos*. Establece cualquier *dato* que supere el **recuento de salidas de envío** en series nulas. Este método devuelve TRUE si es satisfactorio.

ImqBoolean setSendUserData(const char * datos = 0);

Establece los **datos de usuario de envío** en los *datos* de un solo elemento. Si *data* no es nulo, el **recuento de salidas de envío** debe ser como mínimo 1. Este método devuelve TRUE si es satisfactorio.

ImqBoolean setSendUserData(const size_t count, const char * data []);

Establece **enviar datos de usuario** en *datos*. *count* no debe ser mayor que el **recuento de salidas de envío**. Este método devuelve TRUE si es satisfactorio.

ImqBoolean setSendUserData(const size_t count, const ImqString * data []);

Establece **enviar datos de usuario** en *datos*. *count* no debe ser mayor que el **recuento de salidas de envío**. Este método devuelve TRUE si es satisfactorio.

ImqString sslCipherSpecification() const;

Devuelve la especificación del cifrado « TLS ».

ImqBoolean setSslCipherSpecification(const char * nombre = 0);

Establece la especificación de cifrado « TLS ». Este método devuelve TRUE si es satisfactorio.

MQLONG sslClientAuthentication() const;

Devuelve el tipo de autenticación de cliente « TLS ».

ImqBoolean setSslClientAuthentication(const MQLONG auth = MQSCA_REQUIRED);

Establece el tipo de autenticación de cliente « TLS ». Este método devuelve TRUE si es satisfactorio.

ImqString sslPeerName() const;

Devuelve el nombre del par « TLS ».

ImqBoolean setSslPeerName(const char * nombre = 0);

Establece el nombre del par « TLS ». Este método devuelve TRUE si es satisfactorio.

ImqString transactionProgramName() const;

Devuelve el **nombre de programa de transacción**.

ImqBoolean setTransactionProgramName(const char * *nombre* = 0);

Establece el **nombre de programa de transacción**. Este método devuelve TRUE si es satisfactorio.

MQLONG transportType() const;

Devuelve el **tipo de transporte**.

ImqBoolean setTransportType(const MQLONG *tipo* = MQXPT_LU62);

Establece el **tipo de transporte**. Este método devuelve TRUE si es satisfactorio.

ImqString userId() const;

Devuelve el **ID de usuario**.

ImqBoolean setUserId(const char * *id* = 0);

Establece el **ID de usuario**. Este método devuelve TRUE si es satisfactorio.

códigos de razón

- MQRC_DATA_LENGTH_ERROR
- MQRC_ERROR_DE_RECuento_DE_ARTÍCULOS
- MQRC_PUNTERO_NULO
- MQRC_SOURCE_BUFFER_ERROR

ImqCICSBridgeHeader Clase C++

Esta clase encapsula características específicas de la estructura de datos MQCIH.

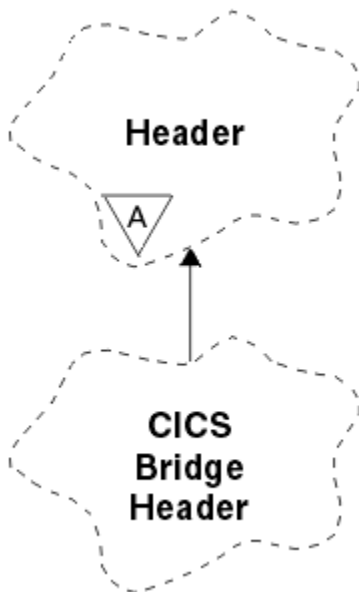


Figura 17. ImqCICSClass BridgeHeader

Los objetos de esta clase los utilizan las aplicaciones que envían mensajes a CICS bridge a través de IBM MQ for z/OS.

- [“Atributos de objetos” en la página 1845](#)
- [“Constructores” en la página 1847](#)
- [“Métodos ImqItem sobrecargados” en la página 1847](#)
- [“Métodos de objeto \(public\)” en la página 1847](#)
- [“Datos de objeto \(protegidos\)” en la página 1850](#)
- [“códigos de razón” en la página 1850](#)
- [“Códigos de retorno” en la página 1850](#)

Atributos de objetos

Descriptor ADS

Enviar/recibir descriptor ADS. Se establece utilizando MQCADSD_NONE. El valor inicial es MQCADSD_NONE. Son posibles los siguientes valores adicionales:

- MQCADSD_NONE
- MQCADSD_ENVIAR
- MQCADSD_RECV
- MQCADSD_MSGFORMAT

identificador de atención

Tecla AID. El campo debe tener la longitud MQ_ATTENTION_ID_LENGTH.

autenticador

Contraseña o passticket de RACF . El valor inicial contiene espacios en blanco, de longitud MQ_AUTHENTICATOR_LENGTH.

código de terminación anómala de puente

Código de terminación anómala de puente, de longitud MQ_ABEND_CODE_LENGTH. El valor inicial es de cuatro caracteres en blanco. El valor devuelto en este campo depende del código de retorno. Consulte [Tabla 903 en la página 1850](#) para obtener más detalles.

código de cancelación de puente

Código de transacción de terminación anómala de puente. El campo está reservado, debe contener espacios en blanco y tener la longitud MQ_CANCEL_CODE_LENGTH.

código de terminación de puente

Código de terminación, que puede contener el código de terminación IBM MQ o el valor EIBRESP de CICS . El campo tiene el valor inicial de MQCC_OK. El valor devuelto en este campo depende del código de retorno. Consulte [Tabla 903 en la página 1850](#) para obtener más detalles.

desplazamiento de error de puente

Desplazamiento de error de puente. El valor inicial es cero. Este atributo es de sólo lectura.

código de razón de puente

Código de razón. Este campo puede contener la razón IBM MQ o el valor CICS EIBRESP2 . El campo tiene el valor inicial de MQRC_NONE. El valor devuelto en este campo depende del código de retorno. Consulte [Tabla 903 en la página 1850](#) para obtener más detalles.

código de retorno de puente

Código de retorno de CICS bridge. El valor inicial es MQCRC_OK.

tarea conversacional

Si la tarea puede ser conversacional. El valor inicial es MQCCT_NO. Son posibles los siguientes valores adicionales:

- MQCCT_SÍ
- MQCCT_NO

Posición del cursor

Posición del cursor. El valor inicial es cero.

tiempo de mantenimiento del recurso

Hora de liberación del recurso CICS bridge .

recurso como

Atributo emulado de terminal. El campo debe tener la longitud MQ_FACILITY_LIKE_LENGTH.

señal de recurso

Valor de señal BVT. El campo debe tener la longitud MQ_FACILITY_LENGTH. El valor inicial es MQCFAC_NONE.

función

Función, que puede contener el nombre de llamada IBM MQ o la función EIBFN CICS . El campo tiene el valor inicial de MQCFUNC_NONE, con la longitud MQ_FUNCTION_LENGTH. El valor devuelto en

este campo depende del código de retorno. Consulte [Tabla 903 en la página 1850](#) para obtener más detalles.

Los siguientes valores adicionales son posibles cuando la **función** contiene un nombre de llamada IBM MQ :

- MQCFUNC_MQCONN
- MQCFUNC_MQGET
- MQCFUNC_MQINQ
- MQCFUNC_NONE
- MQCFUNC_MQOPEN
- MQCFUNC_PUT
- MQCFUNC_MQPUT1

obtener intervalo de espera

Intervalo de espera para una llamada MQGET emitida por la tarea CICS bridge . El valor inicial es MQCGWI_DEFAULT. El campo sólo se aplica cuando **uow control** tiene el valor MQCUOWC_FIRST. Son posibles los siguientes valores adicionales:

- MQCGWI_DEFAULT
- MQWI_ILIMITADO

Tipo de enlace

Tipo de enlace. El valor inicial es MQCLT_PROGRAM. Son posibles los siguientes valores adicionales:

- MQCLT_PROGRAMA
- TRANSACCIÓN_MQC

identificador de transacción siguiente

ID de la siguiente transacción a adjuntar. El campo debe tener la longitud MQ_TRANSACTION_ID_LENGTH.

longitud de datos de salida

Longitud de datos de COMMAREA. El valor inicial es MQCODL_AS_INPUT.

formato de respuesta

Nombre de formato del mensaje de respuesta. El valor inicial es MQFMT_NONE con la longitud MQ_FORMAT_LENGTH.

código de inicio

Código de inicio de transacción. El campo debe tener la longitud MQ_START_CODE_LENGTH. El valor inicial es MQCSC_NONE. Son posibles los siguientes valores adicionales:

- MQCSC_START
- MQCSC_STARTDATA
- MQCSC_TERMINPUT
- MQCSC_NONE

estado de finalización de tarea

Estado de finalización de tarea. El valor inicial es MQCTES_NOSYNC. Son posibles los siguientes valores adicionales:

- MQCTES_COMMITAR
- MQCTES_BACKOUT
- MQCTES_ENDTASK
- MQCTES_NOSYNC

Identificador de transacción

ID de la transacción a adjuntar. El valor inicial debe contener espacios en blanco y debe tener la longitud MQ_TRANSACTION_ID_LENGTH. El campo sólo se aplica cuando **uow control** tiene el valor MQCUOWC_FIRST o MQCUOWC_ONLY.

Control UOW

Control de UOW. El valor inicial es MQCUOWC_ONLY. Son posibles los siguientes valores adicionales:

- MQCUOWC_FIRST
- MQCUOWC_MIDDLE
- MQCUOWC_LAST
- MQCUOWC_ONLY
- MQCUOWC_COMMITAR
- MQCUOWC_BACKOUT
- MQCUOWC_CONTINUAR

versión

El número de versión de MQCIH. El valor inicial es MQCIH_VERSION_2. El único otro valor soportado es MQCIH_VERSION_1.

Constructores

ImqBridgeHeaderCICS();

El constructor predeterminado.

ImqBridgeHeaderCICS(const CICSImqBridgeHeader & header);

El constructor de copia.

Métodos ImqItem sobrecargados

virtual ImqBoolean copyOut(ImqMessage & msg);

Inserta una estructura de datos MQCIH en el almacenamiento intermedio de mensajes al principio, moviendo los datos de mensaje existentes más adelante y establece el formato de mensaje en MQFMT_CICS.

Consulte la descripción del método de clase padre para obtener más detalles.

virtual ImqBoolean pasteIn(ImqMessage & msg);

Lee una estructura de datos MQCIH del almacenamiento intermedio de mensajes. Para que la codificación del objeto *msg* sea satisfactoria, debe ser MQENC_NATIVE. Recuperar mensajes con MQGMO_CONVERT a MQENC_NATIVE. Para ser satisfactorio, el formato ImqMessage debe ser MQFMT_CICS.

Consulte la descripción del método de clase padre para obtener más detalles.

Métodos de objeto (public)

void operador = (const CICSImqBridgeHeader& header);

Copia datos de instancia de la *cabecera*, sustituyendo los datos de instancia existentes.

MQLONG ADSDescriptor () Const.

Devuelve una copia del **descriptor ADS**.

void setADSDescriptor(const MQLONG descriptor = MQCADSD_NONE);

Establece el **Descriptor ADS**.

ImqString attentionIdentifier() const;

Devuelve una copia del **identificador de atención**, rellenado con espacios en blanco finales hasta la longitud MQ_ATTENTION_ID_LENGTH.

void setAttentionIdentifier(const char *data = 0);

Establece el **identificador de atención**, relleno con espacios en blanco finales en la longitud MQ_ATTENTION_ID_LENGTH. Si no se proporcionan *datos*, restablece el **identificador de atención** en el valor inicial.

Autenticador ImqString () Const.

Devuelve una copia del **autenticador**, rellena con espacios en blanco finales hasta la longitud MQ_AUTHENTICATOR_LENGTH.

void setAuthenticator(const char *data = 0);

Establece el **autenticador**, relleno con espacios en blanco finales en la longitud MQ_AUTHENTICATOR_LENGTH. Si no se proporcionan *datos*, restablece el **autenticador** en el valor inicial.

ImqString bridgeAbendCode() const;

Devuelve una copia del **código de terminación anómala de puente**, relleno con espacios en blanco finales hasta la longitud MQ_ABEND_CODE_LENGTH.

ImqString bridgeCancelCode() const;

Devuelve una copia del **código de cancelación de puente**, relleno con espacios en blanco finales hasta la longitud MQ_CANCEL_CODE_LENGTH.

void setBridgeCancelCode(const char *data = 0);

Establece el **código de cancelación de puente**, relleno con espacios en blanco finales en la longitud MQ_CANCEL_CODE_LENGTH. Si no se proporcionan *datos*, restablece el **código de cancelación de puente** al valor inicial.

MQLONG bridgeCompletionCode() const;

Devuelve una copia del **código de terminación de puente**.

MQLONG bridgeErrorOffset() const;

Devuelve una copia del **desplazamiento de error de puente**.

MQLONG bridgeReasonCode() const;

Devuelve una copia del **código de razón de puente**.

MQLONG bridgeReturnCode() const;

Devuelve el **código de retorno de puente**.

MQLONG conversationalTask() const;

Devuelve una copia de la **tarea conversacional**.

void setConversationalTask(const MQLONG tarea = MQCCT_NO);

Establece la **tarea conversacional**.

MQLONG cursorPosition() const;

Devuelve una copia de la **posición del cursor**.

void setCursorPosition(const MQLONG posición = 0);

Establece la **posición del cursor**.

MQLONG facilityKeepTime() const;

Devuelve una copia del **tiempo de mantenimiento del recurso**.

void setFacilityKeepTime(const MQLONG tiempo = 0);

Establece el **tiempo de mantenimiento del recurso**.

ImqString facilityLike() const;

Devuelve una copia del recurso **como**, rellena con espacios en blanco finales hasta la longitud MQ_FACILITY_LIKE_LENGTH.

void setFacilityLike(const char * nombre = 0);

Establece el recurso **como**, relleno con espacios en blanco finales en la longitud MQ_FACILITY_LIKE_LENGTH. Si no se proporciona ningún *name*, restablece el recurso **como** el valor inicial.

ImqBinary facilityToken() const;

Devuelve una copia de la **señal de recurso**.

ImqBoolean setFacilityToken(const ImqBinary, *token*);

Establece la **señal de recurso**. La **longitud de datos** de *token* debe ser cero o MQ_FACILITY_LENGTH.
Devuelve TRUE si es satisfactorio.

void setFacilityToken(const MQBYTE8 *token* = 0);

Establece la **señal de recurso**. *token* puede ser cero, que es lo mismo que especificar MQCFAC_NONE. Si *token* es distinto de cero, debe direccionar MQ_FACILITY_LENGTH bytes de datos binarios. Cuando se utilizan valores predefinidos como, por ejemplo, MQCFAC_NONE, es posible que tenga que realizar una conversión para garantizar una coincidencia de firma. Por ejemplo, (MQBYTE *) MQCFAC_NONE.

Función ImqString () Const.

Devuelve una copia de la **función**, rellena con espacios en blanco finales hasta la longitud MQ_FUNCTION_LENGTH.

MQLONG getWaitInterval() const;

Devuelve una copia del **intervalo de espera de obtención**.

void setGetWaitInterval(const MQLONG *interval* = MQCGWI_DEFA

Establece el **intervalo de espera de obtención**.

MQLONG linkType() const;

Devuelve una copia del **tipo de enlace**.

void setLinkType(const MQLONG *tipo* = MQCLT_PROGRAM);

Establece el **tipo de enlace**.

ImqString nextTransactionIdentifier() const;

Devuelve una copia de los datos del **siguiente identificador de transacción**, rellenos con blancos de cola hasta la longitud MQ_TRANSACTION_ID_LENGTH.

MQLONG outputDataLength() const;

Devuelve una copia de la **longitud de datos de salida**.

void setOutputDataLength(const MQLONG *longitud* = MQCODL_AS_INPUT);

Establece la **longitud de datos de salida**.

ImqString replyToFormat() const;

Devuelve una copia del nombre de **formato de respuesta**, relleno con blancos de cola hasta la longitud MQ_FORMAT_LENGTH.

void setReplyToFormat(const char * *nombre* = 0);

Establece el **formato de respuesta**, relleno con espacios en blanco finales en la longitud MQ_FORMAT_LENGTH. Si no se proporciona ningún *nombre*, restablece el **formato de respuesta** al valor inicial.

ImqString startCode() const;

Devuelve una copia del **código de inicio**, relleno con espacios en blanco finales hasta la longitud MQ_START_CODE_LENGTH.

void setStartCode(const char **data* = 0);

Establece los datos del **código de inicio**, rellenos con espacios en blanco finales en la longitud MQ_START_CODE_LENGTH. Si no se proporcionan *datos*, restablece el **código de inicio** en el valor inicial.

MQLONG taskEndStatus() const;

Devuelve una copia del **estado de finalización de tarea**.

ImqString transactionIdentifier() const;

Devuelve una copia de los datos del **identificador de transacción**, rellenos con espacios en blanco finales hasta la longitud MQ_TRANSACTION_ID_LENGTH.

void setTransactionIdentifier(const char **data* = 0);

Establece el **identificador de transacción**, relleno con espacios en blanco finales en la longitud MQ_TRANSACTION_ID_LENGTH. Si no se proporcionan *datos*, restablece el **identificador de transacción** en el valor inicial.

MQLONG UOWControl () Const.

Devuelve una copia del **control de UOW**.

void setUOWControl(const MQLONG control = MQCUOWC_ONLY);

Establece el **control de UOW**.

MQLONG versión () Const.

Devuelve el número de **versión** .

ImqBoolean setVersion(const MQLONG versión = MQCIH_VERSION_2);

Establece el número de **versión** . Devuelve TRUE si es satisfactorio.

Datos de objeto (protegidos)

MQLONG olVersion

El número máximo de versión de MQCIH que se puede acomodar en el almacenamiento asignado para *opcih*.

PMQCIH opcih

La dirección de una estructura de datos MQCIH. La cantidad de almacenamiento asignado se indica mediante *olVersion*.

códigos de razón

- MQRC_ERROR_DE_LONGITUD_DE_DATOS_BINARIOS
- VERSIÓN_ERROR_MQRC

Códigos de retorno

Tabla 903. ImqCISCódigos de retorno de la clase BridgeHeader				
Código de retorno	Función	CompCode	Razón	Código de terminación anómala
MQCRC_OK				
ERROR DE MQCRC_BRIDGE_			MQFB_CICS	
MQCRC_MQ_API_ERROR	Nombre de llamada IBM MQ	IBM MQ CompCode	IBM MQ Razón	
MQCRC_BRIDGE_TIMEOUT	Nombre de llamada IBM MQ	IBM MQ CompCode	IBM MQ Razón	
MQCRC_CICS_EXEC_ERROR	CICS EIBFN	CICS EIBRESP	CICS EIBRESP2	
MQCRC_ERROR_DE_SEGURIDAD	CICS EIBFN	CICS EIBRESP	CICS EIBRESP2	
MQCRC_PROGRAMA_NO_DISPONIBLE	CICS EIBFN	CICS EIBRESP	CICS EIBRESP2	
MQCRC_TRANSID_NO_DISPONIBLE	CICS EIBFN	CICS EIBRESP	CICS EIBRESP2	
MQCRC_BRIDGE_ABEND				ABCODE de CICS
MQCRC_APLICACIÓN_ABEND				ABCODE de CICS

ImqDeadLetterHeader Clase C

Esta clase encapsula características de la estructura de datos MQDLH.

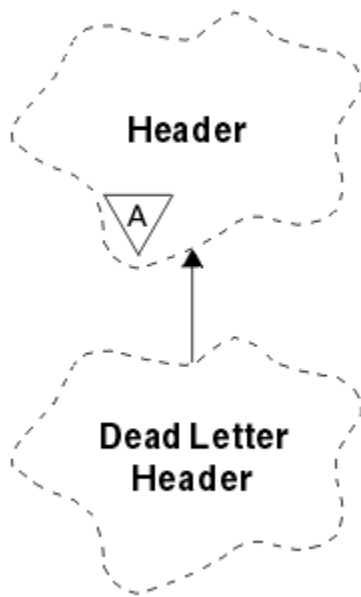


Figura 18. clase *ImqDeadLetterHeader*

Los objetos de esta clase suelen ser utilizados por una aplicación que encuentra un mensaje que no se puede procesar. Un nuevo mensaje que comprende una cabecera de mensaje no entregado y el contenido del mensaje se coloca en la cola de mensajes no entregados y el mensaje se descarta.

- [“Atributos de objetos” en la página 1851](#)
- [“Constructores” en la página 1852](#)
- [“Métodos ImqItem sobrecargados” en la página 1852](#)
- [“Métodos de objeto \(public\)” en la página 1852](#)
- [“Datos de objeto \(protegidos\)” en la página 1853](#)
- [“códigos de razón” en la página 1853](#)

Atributos de objetos

código de razón de mensaje no enviado

La razón por la que el mensaje ha llegado a la cola de mensajes no entregados. El valor inicial es MQRC_NONE.

Nombre del gestor de colas de destino

El nombre del gestor de colas de destino original. El nombre es una serie de longitud MQ_Q_MGR_NAME_LENGTH. Su valor inicial es nulo.

Nombre de la cola de destino

El nombre de la cola de destino original. El nombre es una serie de longitud MQ_Q_NAME_LENGTH. Su valor inicial es nulo.

Nombre de aplicación de transferencia

Nombre de la aplicación que transfiere el mensaje a la cola de mensajes no entregados. El nombre es una serie de longitud MQ_PUT_APPL_NAME_LENGTH. Su valor inicial es nulo.

Tipo de aplicación de transferencia

El tipo de aplicación que coloca el mensaje en la cola de mensajes no entregados. El valor inicial es cero.

Fecha de transferencia

La fecha en la que el mensaje se transfirió a la cola de mensajes no entregados. La fecha es una serie de longitud MQ_PUT_DATE_LENGTH. Su valor inicial es una serie nula.

Hora de transferencia

Hora a la que se transfirió el mensaje a la cola de mensajes no entregados. La hora es una serie de longitud MQ_PUT_TIME_LENGTH. Su valor inicial es una serie nula.

Constructores

ImqDeadLetterHeader();

El constructor predeterminado.

ImqDeadLetterHeader(const ImqDeadLetterHeader &header);

El constructor de copia.

Métodos ImqItem sobrecargados

virtual ImqBoolean copyOut (ImqMessage & msg);

Inserta una estructura de datos MQDLH en el almacenamiento intermedio de mensajes al principio, moviendo los datos de mensaje existentes más adelante. Establece el formato *msg* en MQFMT_DEAD_LETTER_HEADER.

Consulte la descripción del método de clase ImqHeader en la página [“Clase C++ ImqHeader”](#) en la página 1859 para obtener más detalles.

virtual ImqBoolean pasteIn (ImqMessage & msg);

Lee una estructura de datos MQDLH del almacenamiento intermedio de mensajes.

Para ser satisfactorio, el formato de ImqMessage debe ser MQFMT_DEAD_LETTER_HEADER.

Consulte la descripción del método de clase ImqHeader en la página [“Clase C++ ImqHeader”](#) en la página 1859 para obtener más detalles.

Métodos de objeto (public)

void operador = (const ` ImqDeadLetterHeader ` & header);

Copia los datos de instancia desde la *cabecera*, sustituyendo los datos de instancia existentes.

MQLONG deadLetterReasonCode () const;

Devuelve el código de razón de mensaje no enviado.

void setDeadLetterReasonCode (const MQLONG reason);

Establece el código de razón de mensaje no enviado.

ImqString destinationQueueManagerName () const;

Devuelve el nombre del gestor de colas de destino, despojado de los espacios en blanco finales.

void setDestinationQueueManagerName (const char * nombre);

Establece el nombre del gestor de colas de destino. Trunca los datos más largos que MQ_Q_MGR_NAME_LENGTH (48 caracteres).

ImqString destinationQueueName () const;

Devuelve una copia del nombre de cola de destino, despojada de los espacios en blanco finales.

void setDestinationQueueName (const char * nombre);

Establece el nombre de cola de destino. Trunca los datos más largos que MQ_Q_NAME_LENGTH (48 caracteres).

ImqString putApplicationName () const;

Devuelve una copia del nombre de aplicación put, quitada de los espacios en blanco finales.

void setPutApplicationName (const char * nombre = 0);

Establece el nombre de aplicación de colocación. Trunca los datos más largos que MQ_PUT_APPL_NAME_LENGTH (28 caracteres).

MQLONG putApplicationType () const;

Devuelve el tipo de aplicación de colocación.

void setPutApplicationType (const MQLONG tipo = MQAT_NO_CONTEXT);

Establece el tipo de aplicación de colocación.

ImqString putDate () Const.

Devuelve una copia de la fecha de colocación, despojada de los espacios en blanco finales.

void setPutDate (const char * fecha = 0);

Establece la fecha de colocación. Trunca los datos más largos que MQ_PUT_DATE_LENGTH (8 caracteres).

ImqString putTime () Const.

Devuelve una copia del tiempo de colocación, despojada de los espacios en blanco finales.

void setPutTime (const char * tiempo = 0);

Establece la hora de colocación. Trunca los datos más largos que MQ_PUT_TIME_LENGTH (8 caracteres).

Datos de objeto (protegidos)**MQDLH omqdlh**

Estructura de datos MQDLH.

códigos de razón

- MQRC_FORMATO_INCONSISTENTE
- MQRC_STRUC_ID_ERROR
- MQRC_ERROR_DE_CODIFICACIÓN

ImqDistributionList Clase C

Esta clase encapsula una lista de distribución dinámica que hace referencia a una o más colas con el fin de enviar un mensaje o mensajes a varios destinos.

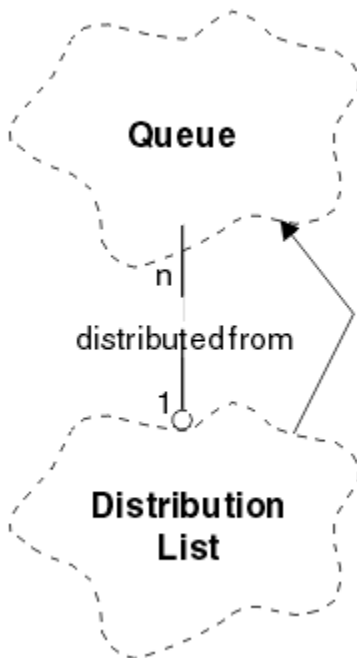


Figura 19. *ImqDistributionList* clase

- [“Atributos de objetos” en la página 1854](#)
- [“Constructores” en la página 1854](#)
- [“Métodos de objeto \(public\)” en la página 1854](#)
- [“Métodos de objeto \(protegidos\)” en la página 1854](#)

Atributos de objetos

primera cola distribuida

El primero de uno o más objetos de clase, en ningún orden concreto, en el que la **referencia de lista de distribución** se dirige a este objeto.

Inicialmente no hay tales objetos. Para abrir correctamente un ImqDistributionList, debe existir al menos un objeto de este tipo.

Nota: Cuando se abre un objeto ImqDistributionList, todos los objetos abiertos que hagan referencia a él se cierran automáticamente.

Constructores

ImqDistributionList();

El constructor predeterminado.

ImqDistributionList(const ImqDistributionList, list);

El constructor de copia.

Métodos de objeto (public)

void operador = (const ` ImqDistributionList ` & lista);

Se anula la referencia de todos los objetos que hacen referencia a **este** objeto antes de copiarlos. Ningún objeto hará referencia a **este** objeto después de la invocación de este método.

* firstDistributedQueue () const;

Devuelve la **primera cola distribuida**.

Métodos de objeto (protegidos)

void setFirstDistributedQueue (* cola = 0);

Establece la **primera cola distribuida**.

Clase C++ ImqError

Esta clase abstracta proporciona información sobre los errores asociados a un objeto.

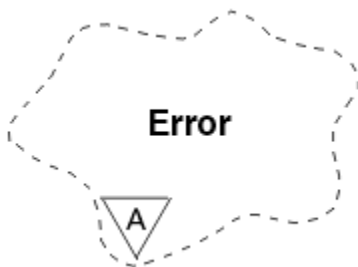


Figura 20. Clase ImqError

- [“Atributos de objetos” en la página 1854](#)
- [“Constructores” en la página 1855](#)
- [“Métodos de objeto \(public\)” en la página 1855](#)
- [“Métodos de objeto \(protegidos\)” en la página 1855](#)
- [“códigos de razón” en la página 1855](#)

Atributos de objetos

código de terminación

El código de terminación más reciente. El valor inicial es cero. Son posibles los siguientes valores adicionales:

- MQCC_OK
- MQCC_WARNING
- MQCC_FAILED

código de razón

El código de razón más reciente. El valor inicial es cero.

Constructores

ImqError();

El constructor predeterminado.

ImqError(const ImqError, *error*);

El constructor de copia.

Métodos de objeto (public)

void operador = (const ` ImqError ` & *error*);

Copia datos de instancia de *error*, sustituyendo los datos de instancia existentes.

void clearErrorCodes ();

Establece el **código de terminación** y el **código de razón** en cero.

MQLONG completionCode () const ;

Devuelve el **código de terminación**.

MQLONG reasonCode () const ;

Devuelve el **código de razón**.

Métodos de objeto (protegidos)

ImqBooleancheckReadPointer (const void * *puntero*, const size_t *longitud*);

Verifica que la combinación de puntero y longitud es válida para el acceso de sólo lectura y devuelve TRUE si es satisfactoria.

ImqBooleancheckWritePointer (const void * *puntero*, const size_t *longitud*);

Verifica que la combinación de puntero y longitud es válida para el acceso de lectura-escritura y devuelve TRUE si es satisfactoria.

void setCompletionCode (const MQLONG *código* = 0);

Establece el **código de terminación**.

void setReasonCode (const MQLONG *código* = 0);

Establece el **código de razón**.

códigos de razón

- MQRC_BUFFER_ERROR

ImqGetMessageOptions Clase C

Esta clase encapsula la estructura de datos MQGMO

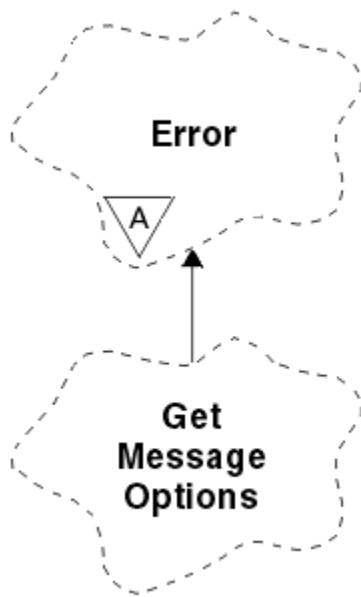


Figura 21. *ImqGetMessageOptions* clase

- [“Atributos de objetos” en la página 1856](#)
- [“Constructores” en la página 1857](#)
- [“Métodos de objeto \(public\)” en la página 1858](#)
- [“Métodos de objeto \(protegidos\)” en la página 1859](#)
- [“Datos de objeto \(protegidos\)” en la página 1859](#)
- [“códigos de razón” en la página 1859](#)

Atributos de objetos

Estado del grupo

Estado de un mensaje para un grupo de mensajes. El valor inicial es MQGS_NOT_IN_GROUP. Son posibles los siguientes valores adicionales:

- MQGS_MSG_IN_GROUP
- MQGS_ÚLTIMO_MENSAJE_EN_GRUPO

opciones de coincidencia

Opciones para seleccionar mensajes entrantes. El valor inicial es MQMO_MATCH_MSG_ID | MQMO_MATCH_CORREL_ID. Son posibles los siguientes valores adicionales:

- MQMO_ID_grupo
- MQMO_MATCH_MSG_SEQ_NUMBER
- MQMO_MATCH_OFFSET
- MQMO_MSG_TOKEN
- MQMO_NONE

token de mensaje

Señal de mensaje. Un valor binario (MQBYTE16) de longitud MQ_MSG_TOKEN_LENGTH. El valor inicial es MQMTOK_NONE.

opciones

Opciones aplicables a un mensaje. El valor inicial es MQGMO_NO_WAIT. Son posibles los siguientes valores adicionales:

- MQGMO_WAIT
- MQGMO_SYNCPOINT

- MQGMO_SYNCPOINT_IF_PERSISTENT
- MQGMO_NO_SYNCPOINT
- MQGMO_MARK_SKIP_BACKOUT
- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_NEXT
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_MSG_UNDER_CURSOR
- MQGMO_LOCK
- MQGMO_UNLOCK
- MQGMO_ACCEPT_TRUNCATED_MSG
- MQGMO_SET_SIGNAL
- MQGMO_FAIL_IF QUIESCING
- MQGMO_CONVERT
- MQGMO_LOGICAL_ORDER
- MQGMO_COMPLETE_MSG
- MQGMO_ALL_MSGS_AVAILABLE
- MQGMO_ALL_SEGMENTS_AVAILABLE
- MQGMO_NONE

Nombre de cola resuelto

Nombre de cola resuelto. Este atributo es de sólo lectura. Los nombres nunca tienen más de 48 caracteres y pueden rellenarse hasta esa longitud con nulos. El valor inicial es una serie nula.

longitud devuelta

Longitud devuelta. El valor inicial es MQRL_UNDEFINED. Este atributo es de sólo lectura.

segmentación

La capacidad de segmentar un mensaje. El valor inicial es MQSEG_INHIBIDO. El valor adicional, MQSEG_ALLOWED, es posible.

estado de segmento

El estado de segmentación de un mensaje. El valor inicial es MQSS_NOT_A_SEGMENT. Son posibles los siguientes valores adicionales:

- SEGMENTO_MQSS
- MQSS_ÚLTIMO_SEGMENTO

participación de punto de sincronismo

TRUE cuando los mensajes se recuperan bajo control de punto de sincronismo.

Intervalo de espera

El tiempo que el método get de clase se detiene mientras espera a que llegue un mensaje adecuado, si todavía no hay uno disponible. El valor inicial es cero, lo que afecta a una espera indefinida. El valor adicional, MQWI_UNLIMITED, es posible. Este atributo se ignora a menos que las opciones incluyan MQGMO_WAIT.

Constructores

ImqGetMessageOptions();

El constructor predeterminado.

ImqGetMessageOptions(const ImqGetMessageOptions, gmo);

El constructor de copia.

Métodos de objeto (public)

void operador = (const ` ImqGetMessageOptions ` & gmo);

Copia datos de instancia de *gmo*, sustituyendo los datos de instancia existentes.

MQCHAR groupStatus () Const.

Devuelve el estado del grupo.

void setGroupStatus (const MQCHAR status);

Establece el estado del grupo.

MQLONG matchOptions () Const.

Devuelve las opciones de coincidencia.

void setMatchOptions (const MQLONG options);

Establece las opciones de coincidencia.

ImqBinary messageToken() const;

Devuelve la señal de mensaje.

ImqBoolean setMessageToken(const ImqBinary, token);

Establece la señal de mensaje. La longitud de datos de *token* debe ser cero o MQ_MSG_TOKEN_LENGTH. Este método devuelve TRUE si es satisfactorio.

void setMessageToken(const MQBYTE16 token = 0);

Establece la señal de mensaje. *token* puede ser cero, que es lo mismo que especificar MQMTOK_NONE. Si *token* es distinto de cero, debe dirigirse a MQ_MSG_TOKEN_LENGTH bytes de datos binarios.

Cuando se utilizan valores predefinidos, como MQMTOK_NONE, es posible que no sea necesario realizar una conversión para garantizar una coincidencia de firma, por ejemplo (MQBYTE *) MQMTOK_NONE.

Opciones MQLONG () Const.

Devuelve las opciones.

void setOptions (const MQLONG opciones);

Establece las opciones, incluido el valor de participación de punto de sincronismo.

ImqString resolvedQueueName () const;

Devuelve una copia del nombre de cola resuelto.

MQLONG returnedLength() const;

Devuelve la longitud devuelta.

Segmentación MQCHAR () Const.

Devuelve la segmentación.

void setSegmentation (const MQCHAR valor);

Establece la segmentación.

MQCHAR segmentStatus () Const.

Devuelve el estado del segmento.

void setSegmentStatus (const MQCHAR status);

Establece el estado del segmento.

ImqBoolean syncPointParticipation () const;

Devuelve el valor de participación de punto de sincronismo, que es TRUE si las opciones incluyen MQGMO_SYNCPOINT o MQGMO_SYNCPOINT_IF_PERSISTENT.

void setSyncPointParticipation (const ImqBoolean sync);

Establece el valor de participación de punto de sincronismo. Si *sync* es TRUE, modifica las opciones para incluir MQGMO_SYNCPOINT y excluir MQGMO_NO_SYNCPOINT y MQGMO_SYNCPOINT_IF_PERSISTENT. Si *sync* es FALSE, modifica las opciones para incluir MQGMO_NO_SYNCPOINT y para excluir MQGMO_SYNCPOINT y MQGMO_SYNCPOINT_IF_PERSISTENT.

MQLONG waitInterval () Const.

Devuelve el intervalo de espera.

void setWaitInterval (const MQLONG interval);

Establece el intervalo de espera.

Métodos de objeto (protegidos)

static void setVersionSupported (const MQLONG);

Establece la versión MQGMO. El valor predeterminado es MQGMO_VERSION_3.

Datos de objeto (protegidos)

MQGMO omqgmo

Una estructura de datos MQGMO Versión 2. Acceda a los campos MQGMO soportados sólo para MQGMO_VERSION_2 .

PMQGMO opgmo

La dirección de una estructura de datos MQGMO. El número de versión de esta dirección se indica en *olVersion*. Inspeccione el número de versión antes de acceder a los campos MQGMO, para asegurarse de que están presentes.

MQLONG olVersion

El número de versión de la estructura de datos MQGMO direccionado por *opgmo*.

códigos de razón

- MQRC_ERROR_DE_LONGITUD_DE_DATOS_BINARIOS

Clase C++ ImqHeader

Esta clase abstracta encapsula características comunes de la estructura de datos MQDLH.

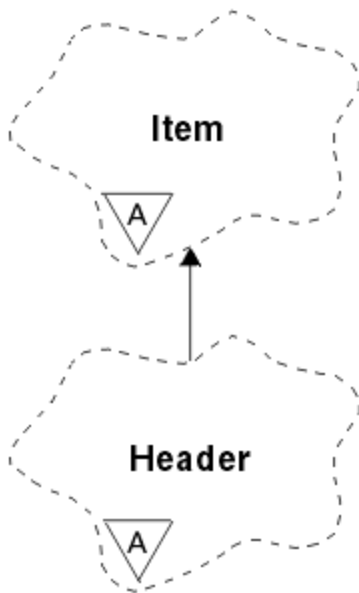


Figura 22. Clase ImqHeader

- [“Atributos de objetos” en la página 1859](#)
- [“Constructores” en la página 1860](#)
- [“Métodos de objeto \(public\)” en la página 1860](#)

Atributos de objetos

juego de caracteres

Identificador del juego de caracteres codificado original. Inicialmente MQCCSI_Q_MGR.

codificación

La codificación original. Inicialmente MQENC_NATIVE.

formato

El formato original. Inicialmente MQFMT_NONE.

distintivos de cabecera

Los valores iniciales son:

- Cero para los objetos de la clase ImqDeadLetterHeader
- MQIIH_NONE para objetos de la clase ImqIMSBridgeHeader
- MQRMHF_LAST para objetos de la clase ImqReferenceHeader
- MQCIH_NONE para objetos de la clase ImqCICSBridgeHeader
- MQWIIH_NONE para objetos de la clase ImqWorkHeader

Constructores**ImqHeader();**

El constructor predeterminado.

ImqHeader(const ImqHeader &header);

El constructor de copia.

Métodos de objeto (public)**void operator = (const ImqHeader & cabecera);**

Copia los datos de instancia de la *cabecera*, sustituyendo los datos de instancia existentes.

virtual MQLONG characterSet () const ;

Devuelve el **juego de caracteres**.

virtual void setCharacterSet (const MQLONG ccsid = MQCCSI_Q_MGR);

Establece el **juego de caracteres**.

Codificación () MQLONG virtual const ;

Devuelve la **codificación**.

virtual void setEncoding (const MQLONG encoding = MQENC_NATIVE);

Establece la **codificación**.

virtual ImqString formato () const ;

Devuelve una copia del **formato**, incluidos los espacios en blanco finales.

virtual void setFormat (const char * name = 0);

Establece el **formato**, rellenado con 8 caracteres con blancos de cola.

virtual MQLONG headerFlags () const ;

Devuelve los **distintivos de cabecera**.

función virtual void setHeaderFlags (const MQLONG flags = 0);

Establece los **distintivos de cabecera**.

ImqIMSBridgeHeader Clase C++

Esta clase encapsula características de la estructura de datos MQIIH.

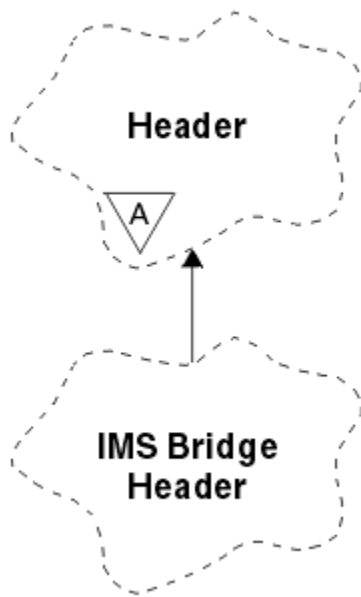


Figura 23. *ImqIMSClase BridgeHeader*

Los objetos de esta clase los utilizan las aplicaciones que envían mensajes al puente IMS a través de IBM MQ for z/OS.

Nota: El juego de caracteres ImqHeader y la codificación deben tener valores predeterminados y no deben establecerse en ningún otro valor.

- [“Atributos de objetos” en la página 1861](#)
- [“Constructores” en la página 1862](#)
- [“Métodos ImqItem sobrecargados” en la página 1862](#)
- [“Métodos de objeto \(public\)” en la página 1862](#)
- [“Datos de objeto \(protegidos\)” en la página 1863](#)
- [“códigos de razón” en la página 1863](#)

Atributos de objetos

autenticador

Contraseña o passticket de RACF , de longitud MQ_AUTHENTICATOR_LENGTH. El valor inicial es MQIAUT_NONE.

Modalidad de confirmación

Modalidad de confirmación. Consulte la publicación *OTMA User's Guide* para obtener más información sobre las modalidades de confirmación de IMS . El valor inicial es MQICM_COMMIT_THEN_SEND. El valor adicional, MQICM_SEND_THEN_COMMIT, es posible.

Alteración temporal del terminal lógico

Alteración temporal de terminal lógico, de longitud MQ_LTERM_OVERRIDE_LENGTH. El valor inicial es una serie nula.

Nombre de la correlación de servicios del formato del mensaje

Nombre de correlación MFS, de longitud MQ_MFS_MAP_NAME_LENGTH. El valor inicial es una serie nula.

formato de respuesta

Formato de cualquier respuesta, de longitud MQ_FORMAT_LENGTH. El valor inicial es MQFMT_NONE.

Ámbito de seguridad

Ámbito del proceso de seguridad de IMS . El valor inicial es MQISS_CHECK. El valor adicional, MQISS_FULL, es posible.

id de instancia de transacción

Identidad de instancia de transacción, un valor binario (MQBYTE16) de longitud MQ_TRAN_INSTANCE_ID_LENGTH. El valor inicial es MQITII_NONE.

ESTADO DE TRANSACCIÓN

Estado de la conversación de IMS . El valor inicial es MQITS_NOT_IN_CONVERSATION. El valor adicional, MQITS_IN_CONVERSATION, es posible.

Constructores

ImqBridgeHeaderIMS();

El constructor predeterminado.

ImqBridgeHeaderIMS(const IMSImqBridgeHeader & header);

El constructor de copia.

Métodos ImqItem sobrecargados

virtual ImqBoolean copyOut (ImqMessage & msg);

Inserta una estructura de datos MQIIH en el almacenamiento intermedio de mensajes al principio, moviendo los datos de mensajes existentes más adelante. Establece el formato *msg* en MQFMT_IMS.

Consulte la descripción del método de clase padre para obtener más detalles.

virtual ImqBoolean pasteIn (ImqMessage & msg);

Lee una estructura de datos MQIIH del almacenamiento intermedio de mensajes.

Para que la codificación del objeto *msg* sea satisfactoria, debe ser MQENC_NATIVE. Recuperar mensajes con MQGMO_CONVERT a MQENC_NATIVE.

Para ser satisfactorio, el formato ImqMessage debe ser MQFMT_IMS.

Consulte la descripción del método de clase padre para obtener más detalles.

Métodos de objeto (public)

void operador = (const IMSImqBridgeHeader& header);

Copia los datos de instancia de la *cabecera*, sustituyendo los datos de instancia existentes.

Autenticador ImqString () Const.

Devuelve una copia del autenticador, rellena con espacios en blanco finales hasta la longitud MQ_AUTHENTICATOR_LENGTH.

void setAuthenticator (const char * nombre);

Establece el autenticador.

MQCHAR commitMode () Const.

Devuelve la modalidad de confirmación.

void setCommitMode (const MQCHAR modo);

Establece la modalidad de confirmación.

ImqString logicalTerminalOverride () const;

Devuelve una copia de la alteración temporal del terminal lógico.

void setLogicalTerminalOverride (const char* override);

Establece la alteración temporal de terminal lógico.

ImqString messageFormatServicesMapName () const;

Devuelve una copia del nombre de correlación de servicios de formato de mensaje.

void setMessageFormatServicesMapName (const char * nombre);

Establece el nombre de correlación de servicios de formato de mensaje.

ImqString replyToFormat () const;

Devuelve una copia del formato de respuesta, rellena con espacios en blanco finales hasta la longitud MQ_FORMAT_LENGTH.

void setReplyToFormat (const char* *format*);

Establece el formato de respuesta, relleno con espacios en blanco finales en la longitud MQ_FORMAT_LENGTH.

MQCHAR securityScope () Const.

Devuelve el ámbito de seguridad.

void setSecurityScope (const MQCHAR *ámbito*);

Establece el ámbito de seguridad.

ImqBinary transactionInstanceId () const;

Devuelve una copia del ID de instancia de transacción.

ImqBoolean setTransactionInstanceId (const ImqBinary & *id*);

Establece el ID de instancia de transacción. La longitud de datos de *token* debe ser cero o MQ_TRAN_INSTANCE_ID_LENGTH. Este método devuelve TRUE si es satisfactorio.

void setTransactionInstanceId (const MQBYTE16 *id* = 0);

Establece el ID de instancia de transacción. *id* puede ser cero, que es lo mismo que especificar MQITII_NONE. Si *id* es distinto de cero, debe direccionar MQ_TRAN_INSTANCE_ID_LENGTH bytes de datos binarios. Cuando se utilizan valores predefinidos como MQITII_NONE, es posible que tenga que realizar una conversión para garantizar una coincidencia de firma, por ejemplo (MQBYTE *) MQITII_NONE.

MQCHAR transactionState () Const.

Devuelve el estado de la transacción.

void setTransactionState (const MQCHAR *estado*);

Establece el estado de la transacción.

Datos de objeto (protegidos)

MQIIH *omqiih*

Estructura de datos MQIIH.

códigos de razón

- MQRC_ERROR_DE_LONGITUD_DE_DATOS_BINARIOS
- MQRC_FORMATO_INCONSISTENTE
- MQRC_ERROR_DE_CODIFICACIÓN
- MQRC_STRUC_ID_ERROR

Clase C++ ImqItem

Esta clase abstracta representa un elemento, quizás uno de varios, dentro de un mensaje.

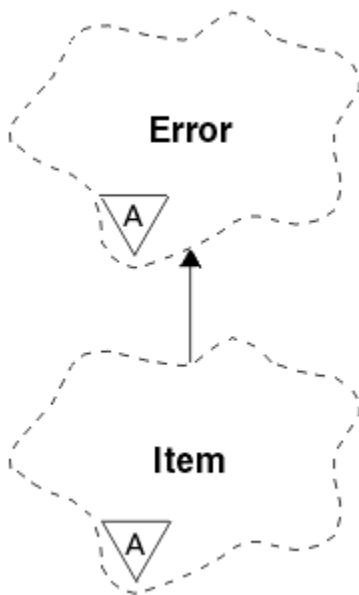


Figura 24. Clase *ImqItem*

Los elementos se concatenan en un almacenamiento intermedio de mensajes. Cada especialización se asocia con una estructura de datos determinada que empieza con un ID de estructura.

Los métodos polimórficos de esta clase abstracta permiten que los elementos se copien a y desde los mensajes. La clase *ImqMessage* **readItem** y los métodos **writeItem** proporcionan otro estilo de invocación de estos métodos polimórficos que es más natural para los programas de aplicación.

- [“Atributos de objetos” en la página 1864](#)
- [“Constructores” en la página 1864](#)
- [“Métodos de clase \(public\)” en la página 1864](#)
- [“Métodos de objeto \(public\)” en la página 1865](#)
- [“códigos de razón” en la página 1865](#)

Atributos de objetos

ID de estructura

Una serie de cuatro caracteres al principio de la estructura de datos. Este atributo es de sólo lectura. Considere este atributo para las clases derivadas. No se incluye automáticamente.

Constructores

ImqItem();

El constructor predeterminado.

ImqItem(const ImqItem, *item*);

El constructor de copia.

Métodos de clase (public)

static ImqBoolean structureIdIs (const char * *estructura-id-a-prueba*, const ImqMessage & *msg*);

Devuelve TRUE si el **ID de estructura** del siguiente *ImqItem* en el *msg* entrante es el mismo que *estructura-id-to-test*. El siguiente elemento se identifica como la parte del almacenamiento intermedio de mensajes a la que se dirige actualmente el **puntero de datos** *ImqCache*. Este método se basa en el **ID de estructura** y, por lo tanto, no se garantiza que funcione para todas las clases derivadas de *ImqItem*.

Métodos de objeto (public)

void operator = (const ImqItem & elemento);

Copia datos de instancia de *elemento*, sustituyendo los datos de instancia existentes.

virtual ImqBoolean copyOut (ImqMessage & msg) = 0;

Escribe este objeto como el siguiente elemento en un almacenamiento intermedio de mensajes de salida, añadiéndolo a los elementos existentes. Si la operación de grabación es satisfactoria, aumenta la **longitud de datos** ImqCache . Este método devuelve TRUE si es satisfactorio.

Altere temporalmente este método para trabajar con una subclase específica.

virtual ImqBoolean pasteIn (ImqMessage & msg) = 0;

Lee este objeto *de forma destructiva* desde el almacenamiento intermedio de mensajes de entrada. La lectura es destructiva en la que se mueve el **puntero de datos** ImqCache . Sin embargo, el contenido del almacenamiento intermedio sigue siendo el mismo, por lo que los datos se pueden volver a leer restableciendo el **puntero de datos** ImqCache .

La clase (sub) de este objeto debe ser coherente con el **ID de estructura** que se encuentra a continuación en el almacenamiento intermedio de mensajes del objeto *msg* .

La **codificación** del objeto *msg* debe ser MQENC_NATIVE. Se recomienda que los mensajes se recuperen con la **codificación** ImqMessage establecida en MQENC_NATIVE, y con las **opciones** ImqGetMessageOptions incluyendo MQGMO_CONVERT.

Si la operación de lectura es satisfactoria, la **longitud de datos** ImqCache se reduce. Este método devuelve TRUE si es satisfactorio.

Altere temporalmente este método para trabajar con una subclase específica.

códigos de razón

- MQRC_ERROR_DE_CODIFICACIÓN
- MQRC_STRUC_ID_ERROR
- MQRC_FORMATO_INCONSISTENTE
- MQRC_INSUFICIENT_BUFFER
- MQRC_INSUFICIENT_DATA

ImqMessage Clase C++

Esta clase encapsula una estructura de datos MQMD y también gestiona la construcción y reconstrucción de datos de mensajes.

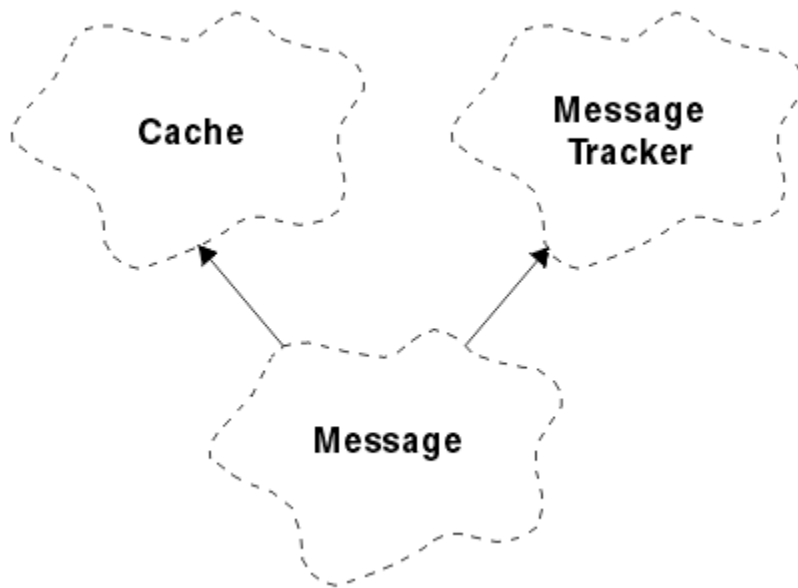


Figura 25. *ImqMessage* clase

- [“Atributos de objetos” en la página 1866](#)
- [“Constructores” en la página 1870](#)
- [“Métodos de objeto \(públicos\)” en la página 1870](#)
- [“Métodos de objeto \(protegidos\)” en la página 1872](#)
- [“Datos del objeto \(protegidos\)” en la página 1872](#)

Atributos de objetos

Datos de ID de aplicación

Información de identidad asociada a un mensaje. El valor inicial es una cadena nula.

Datos de origen de aplicación

Información sobre el origen asociada a un mensaje. El valor inicial es una cadena nula.

Recuento de restituciones

El número de veces que un mensaje se ha recuperado provisionalmente y posteriormente se ha deshecho. El valor inicial es cero. Este atributo es de sólo lectura.

juego de caracteres

Identificador del conjunto de caracteres codificados. El valor inicial es MQCCSI_Q_MGR. Son posibles los siguientes valores adicionales:

- MQCCSI_HEREDAR
- MQCCSI_INTEGRADO

También puede utilizar un identificador de conjunto de caracteres codificados de su elección. Para obtener información al respecto, consulte [“Conversión de páginas de códigos” en la página 947](#).

codificación

La codificación automática de los datos del mensaje. El valor inicial es MQENC_NATIVE.

caducidad

Una cantidad dependiente del tiempo que controla cuánto tiempo IBM MQ conserva un mensaje no recuperado antes de descartarlo. El valor inicial es MQEI_UNLIMITED.

formato

El nombre del formato (plantilla) que describe la disposición de los datos en el búfer. Los nombres que superen los ocho caracteres se truncan a ocho caracteres. Los nombres siempre se rellenan con espacios en blanco hasta alcanzar ocho caracteres. El valor constante inicial es MQFMT_NONE. Son posibles las siguientes constantes adicionales:

- MQFMT_ADMIN
- MQFMT_CICS
- MQFMT_COMMAND_1
- MQFMT_COMMAND_2
- MQFMT_DEAD_LETTER_HEADER
- MQFMT_DIST_HEADER
- MQFMT_EVENT
- MQFMT_IMS
- MQFMT_IMS_VAR_STRING
- MQFMT_MD_EXTENSION
- MQFMT_PCF
- MQFMT_REF_MSG_HEADER
- MQFMT_RF_HEADER
- MQFMT_STRING
- MQFMT_TRIGGER
- MQFMT_WORK_INFO_HEADER
- MQFMT_XMIT_Q_HEADER

También puede utilizar una cadena específica de la aplicación que elija. Para obtener más información al respecto, consulte el [“Format \(MQCHAR8\)”](#) en la [página 417](#) campo del descriptor de mensajes (MQMD).

Distintivos de mensaje

Información sobre el control de segmentación. El valor inicial es MQMF_SEGMENTATION_INHIBITED. Son posibles los siguientes valores adicionales:

- MQMF_SEGMENTATION_ALLOWED
- MQMF_MSG_EN_GRUPO
- Último mensaje en grupo
- MQMF_SEGMENTO
- MQMF_ÚLTIMO_SEGMENTO
- MQMF_NINGUNO

tipo de mensaje

La categorización general de un mensaje. El valor inicial es MQMT_DATAGRAM. Son posibles los siguientes valores adicionales:

- MQMT_SISTEMA_PRIMERO
- MQMT_SISTEMA_ÚLTIMO
- MQMT_DATAGRAM
- MQMT_REQUEST
- MQMT_REPLY
- MQMT_REPORT
- MQMT_APPL_FIRST
- MQMT_APPL_ÚLTIMO

También puede utilizar un valor específico de la aplicación que elija. Para obtener más información al respecto, consulte el [“MsgType \(MQLONG\)”](#) en la [página 407](#) campo del descriptor de mensajes (MQMD).

offset

Información sobre compensación. El valor inicial es cero.

Longitud original

La longitud original de un mensaje segmentado. El valor inicial es MQOL_UNDEFINED.

persistence

Indica que el mensaje es importante y debe respaldarse en todo momento utilizando un almacenamiento persistente. Esta opción implica una penalización en el rendimiento. El valor inicial es MQPER_PERSISTENCE_AS_Q_DEF. Son posibles los siguientes valores adicionales:

- MQPER_PERSISTENT
- MQPER_NOT_PERSISTENT

priority

La prioridad relativa para la transmisión y la entrega. Los mensajes con la misma prioridad suelen entregarse en el mismo orden en que se enviaron (aunque hay varios criterios que deben cumplirse para garantizarlo). El valor inicial es MQPRI_PRIORITY_AS_Q_DEF.

Validación de la propiedad

Especifica si la validación de las propiedades debe realizarse cuando se establece una propiedad del mensaje. El valor inicial es **MQCMHO_DEFAULT_VALIDATION**. Son posibles los siguientes valores adicionales:

- MQCMHO_VALIDAR
- MQCMHO_SIN_VALIDACIÓN

Los siguientes métodos actúan sobre **la validación de propiedades** :

MQLONG propertyValidation() const;

Devuelve la opción **de validación de la propiedad**.

void setPropertyValidation(const MQLONG option);

Establece la opción **de validación de la propiedad**.

Nombre de aplicación de transferencia

El nombre de la aplicación que envió el mensaje. El valor inicial es una cadena nula.

Tipo de aplicación de transferencia

El tipo de aplicación que envía un mensaje. El valor inicial es MQAT_NO_CONTEXT. Son posibles los siguientes valores adicionales:

- MQAT_AIX
- MQAT_CICS
- MQAT_CICS_PUENTE
- MQAT_DOS
- MQAT_IMS
- MQAT_IMS_PUENTE
- MQAT_MVS
- MQAT_NOTAS_AGENTE
- MQAT_OS2
- MQAT_OS390
- MQAT_OS400
- MQAT_QMGR
- MQAT_UNIX
- MQAT_VENTANAS
- MQAT_WINDOWS_NT
- MQAT_XCF
- MQAT_DEFAULT
- MQAT_DESCONOCIDO

- MQAT_USUARIO_PRIMERO
- MQAT_USUARIO_ÚLTIMO

También puede utilizar una cadena específica de la aplicación que elija. Para obtener más información al respecto, consulte el [“PutApplType \(MQLONG\)”](#) en la [página 432](#) campo del descriptor de mensajes (MQMD).

Fecha de transferencia

La fecha en la que se publicó un mensaje. El valor inicial es una cadena nula.

Hora de transferencia

La hora en que se envió un mensaje. El valor inicial es una cadena nula.

Nombre de gestor de cola de respuestas

El nombre del gestor de colas al que se debe enviar cualquier respuesta. El valor inicial es una cadena nula.

Nombre de cola de respuestas

El nombre de la cola a la que se debe enviar cualquier respuesta. El valor inicial es una cadena nula.

informe

Información de retroalimentación asociada a un mensaje. El valor inicial es MQRO_NONE. Son posibles los siguientes valores adicionales:

- MQRO_EXCEPTION
- MQRO_EXCEPTION_WITH_DATA
- MQRO_EXCEPTION_WITH_FULL_DATA *
- MQRO_EXPIRATION
- MQRO_EXPIRATION_WITH_DATA
- MQRO_EXPIRACIÓN_CON_DATOS_COMPLETOS *
- MQRO_COA
- MQRO_COA_WITH_DATA
- MQRO_COA_CON_DATOS_COMPLETOS *
- MQRO_COD
- MQRO_COD_WITH_DATA
- MQRO_COD_CON_DATOS_COMPLETOS *
- MQRO_PAN
- MQRO_NAN
- MQRO_NEW_MSG_ID
- MQRO_NUEVO_ID_CORRELACIÓN
- MQRO_COPY_MSG_ID_TO_CORREL_ID
- MQRO_PASS_CORREL_ID
- MQRO_DEAD_LETTER_Q
- MQRO_DISCARD_MSG

donde * indica valores que no son compatibles con IBM MQ for z/OS.

número de secuencia

Información de secuencia que identifica un mensaje dentro de un grupo. El valor inicial es uno.

longitud total del mensaje

El número de bytes que estaban disponibles durante el último intento de leer un mensaje. Este número será mayor que **la longitud del mensaje** ImqCache si el último mensaje se truncó o si el último mensaje no se leyó porque se habría producido un truncamiento. Este atributo es de sólo lectura. El valor inicial es cero.

Este atributo puede resultar útil en cualquier situación en la que se trate de mensajes truncados.

ID de usuario

Identidad de usuario asociada a un mensaje. El valor inicial es una cadena nula.

Constructores

ImqMessage();

El constructor predeterminado.

ImqMessage(const ImqMessage, *msg*);

El constructor de copia. Consulte el método **del** operador = para obtener más detalles.

Métodos de objeto (públicos)

operador vacío = (const ImqMessage &*msg*);

Copia el MQMD y los datos del mensaje desde *msg*. Si el usuario ha proporcionado un búfer para este objeto, la cantidad de datos copiados se limita al tamaño del búfer disponible. De lo contrario, el sistema garantiza que se disponga de un búfer de tamaño adecuado para los datos copiados.

ImqStringapplicationIdData () const;

Devuelve una copia de los **datos del ID de la aplicación**.

void setApplicationIdData (const char * *datos* = 0);

Establece los **datos del ID de la aplicación**.

ImqStringapplicationOriginData () const;

Devuelve una copia de los **datos de origen de la aplicación**.

void setApplicationOriginData (const char * *datos* = 0);

Establece los **datos de origen de la aplicación**.

MQLONG backoutCount () const;

Devuelve el **recuento de retrocesos**.

MQLONG characterSet () const;

Devuelve el **conjunto de caracteres**.

void setCharacterSet (const MQLONG *ccsid* = MQCCSI_Q_MGR);

Establece el **juego de caracteres**.

Codificación MQLONG () const;

Devuelve la **codificación**.

void setEncoding (const MQLONG *encoding* = MQENC_NATIVE);

Establece la **codificación**.

Caducidad de MQLONG () const;

Devuelve la fecha **de vencimiento**.

void setExpiry (const MQLONG *expiración*);

Establece la **caducidad**.

ImqString formato () const;

Devuelve una copia del **formato**, incluyendo los espacios en blanco finales.

ImqBooleanformatIs (const char * *formato-a-probar*) const;

Devuelve TRUE si el **formato** es igual al *formato que se va a comprobar*.

void setFormat (const char * *nombre* = 0);

Establece el **formato**, rellenado a ocho caracteres con espacios en blanco al final.

MQLONG messageFlags () const;

Devuelve los **indicadores del mensaje**.

void setMessageFlags (const MQLONG *flags*);

Establece los **indicadores del mensaje**.

MQLONG messageType () const;

Devuelve el **tipo de mensaje**.

void setMessageType (const MQLONG *tipo*);
 Establece el **tipo de mensaje**.

Desplazamiento MQLONG () const;
 Devuelve el **desplazamiento**.

void setOffset (const MQLONG *offset*);
 Establece el **desplazamiento**.

MQLONG originalLength () const;
 Devuelve la **longitud original**.

void setOriginalLength (const MQLONG *longitud*);
 Establece la **longitud original**.

Persistencia MQLONG () const;
 Devuelve la **persistencia**.

void setPersistence (const MQLONG *persistence*);
 Establece la **persistencia**.

Prioridad MQLONG () const;
 Devuelve la **prioridad**.

void setPriority (const MQLONG *prioridad*);
 Establece la **prioridad**.

ImqStringputApplicationName () const;
 Devuelve una copia del **nombre** de la aplicación put.

void setPutApplicationName (const char * *nombre* = 0);
 Establece el **nombre de la aplicación put**.

MQLONG putApplicationType () const;
 Devuelve el **tipo de aplicación put**.

void setPutApplicationType (const MQLONG *tipo* = MQAT_NO_CONTEXT);
 Establece el **tipo de aplicación put**.

ImqStringputDate () const;
 Devuelve una copia de la **fecha de ingreso**.

void setPutDate (const char * *fecha* = 0);
 Establece la **fecha de entrega**.

ImqStringputTime () const;
 Devuelve una copia de la **hora de inserción**.

void setPutTime (const char* *time* = 0);
 Establece la **hora de entrega**.

ImqBooleanreadItem (ImqItem &*item*);
 Lee el objeto *del* elemento desde el búfer de mensajes, utilizando el método **pasteInImqItem**.
 Devuelve TRUE si se realiza correctamente.

ImqStringreplyToQueueManagerName () const;
 Devuelve una copia del **nombre** del gestor de colas de respuesta.

void setReplyToQueueManagerName (const char * *nombre* = 0);
 Establece el **nombre del gestor de colas de respuesta**.

ImqStringreplyToQueueName () const;
 Devuelve una copia del **nombre** de la cola de respuesta.

void setReplyToQueueName (const char * *nombre* = 0);
 Establece el **nombre de la cola de respuesta**.

Informe MQLONG () const;
 Devuelve el **informe**.

void setReport (const MQLONG *informe*);
 Establece el **informe**.

MQLONG sequenceNumber () const;

Devuelve el **número de secuencia**.

void setSequenceNumber (const MQLONG *número*);

Establece el **número de secuencia**.

size_t totalMessageLength () const;

Devuelve la **longitud total del mensaje**.

ImqStringuserId () const;

Devuelve una copia del **ID de usuario**.

void setUserId (const char **id* = 0);

Establece el **ID de usuario**.

ImqBooleanwriteItem (ImqItem &*item*);

Escribe desde el objeto *elemento* en el búfer de mensajes, utilizando el método **copyOutImqItem**.

La escritura puede adoptar la forma de inserción, sustitución o adición: esto depende de la clase del objeto *elemento*. Este método devuelve TRUE si se ejecuta correctamente.

Métodos de objeto (protegidos)

static void setVersionSupported (const MQLONG);

Establece la **versión de MQMD**. El valor predeterminado es **MQMD_VERSION_2**.

Datos del objeto (protegidos)

z/OS **MQMD1 omqmd**

La estructura de datos MQMD en z/OS.

Multi **MQMD2 omqmd**

La estructura de datos MQMD en [Multiplataformas](#).

ImqMessageTracker Clase C++

Esta clase encapsula aquellos atributos de un objeto ImqMessage o ImqQueue que pueden asociarse con cualquiera de los dos objetos.

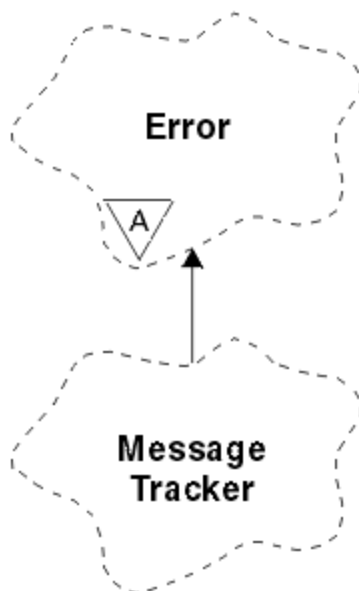


Figura 26. *ImqMessageTracker* clase

Esta clase está relacionada con las llamadas de la clase `MQI` que se enumeran en [“ImqMessageTracker referencia cruzada”](#) en la página 1820.

- [“Atributos de objetos” en la página 1873](#)
- [“Constructores” en la página 1874](#)
- [“Métodos de objeto \(públicos\)” en la página 1874](#)
- [“códigos de razón” en la página 1875](#)

Atributos de objetos

Señal de contabilidad

Un valor binario (MQBYTE32) de longitud MQ_ACCOUNTING_TOKEN_LENGTH. El valor inicial es MQACT_NONE.

ID de correlación

Un valor binario (MQBYTE24) de longitud MQ_CORREL_ID_LENGTH que se asigna para correlacionar mensajes. El valor inicial es MQCI_NONE. El valor adicional, MQCI_NEW_SESSION, es posible.

información de retorno

Información de retroalimentación que se enviará con un mensaje. El valor inicial es MQFB_NONE. Son posibles los siguientes valores adicionales:

- MQFB_SISTEMA_PRIMERO
- MQFB_SISTEMA_ÚLTIMO
- MQFB_APPL_FIRST
- MQFB_APPL_ÚLTIMO
- MQFB_COA
- MQFB_COD
- MQFB_CADUCIDAD
- MQFB_PAN
- MQFB_NAN
- MQFB_SALIR
- MQFB_LONGITUD_DE_DATOS_CERO
- MQFB_LONGITUD_DE_DATOS_NEGATIVA
- MQFB_DATA_LENGTH_TOO_BIG
- MQFB_DESBORDAMIENTO_DE_BUFFER
- MQFB_LONGITUD_DESVIADA_EN_UNO
- MQFB_IIH_ERROR
- MQFB_NO_AUTORIZADO_PARA_IMS
- MQFB_IMS_ERROR
- MQFB_IMS_PRIMERO
- MQFB_IMS_ÚLTIMO
- MQFB_CICS_APPL_ABENDED
- MQFB_CICS_APPL_NO_INICIADO
- Fallo del puente MQFB_CICS
- MQFB_CICS_CCSID_ERROR
- MQFB_CICS_CIH_ERROR
- MQFB_CICS_COMMAREA_ERROR
- MQFB_CICS_CORREL_ID_ERROR
- MQFB_CICS_DLQ_ERROR
- MQFB_CICS_ERROR_DE_CODIFICACIÓN
- MQFB_CICS_ERROR_INTERNO

- MQFB_CICS_NO_AUTORIZADO
- MQFB_CICS_UOW_RETIRADO
- MQFB_CICS_UOW_ERROR

También puede utilizar una cadena específica de la aplicación que elija. Para obtener más información al respecto, consulte el [“Feedback \(MQLONG\)”](#) en la [página 411](#) campo del descriptor de mensajes (MQMD).

ID de grupo

Un valor binario (MQBYTE24) de longitud MQ_GROUP_ID_LENGTH único dentro de una cola. El valor inicial es MQGI_NONE.

ID de mensaje

Un valor binario (MQBYTE24) de longitud MQ_MSG_ID_LENGTH único dentro de una cola. El valor inicial es MQMI_NONE.

Constructores

ImqMessageTracker();

El constructor predeterminado.

ImqMessageTracker(const ImqMessageTracker, *tracker*);

El constructor de copia. Consulte el método **del** operador = para obtener más detalles.

Métodos de objeto (públicos)

operador vacío = (const ImqMessageTracker & *tracker*);

Copia los datos de instancia del *rastreador*, sustituyendo los datos de instancia existentes.

ImqBinaryaccountingToken () const;

Devuelve una copia del **token** de contabilidad.

ImqBooleansetAccountingToken (const ImqBinary & *token*);

Establece el **token de contabilidad**. La **longitud** de *datos del* token debe ser cero o MQ_ACCOUNTING_TOKEN_LENGTH. Este método devuelve TRUE si se ejecuta correctamente.

void setAccountingToken (const MQBYTE32 *token* = 0);

Establece el **token de contabilidad**. El *token* puede ser cero, lo que equivale a especificar MQACT_NONE. Si el *token* es distinto de cero, debe abordar MQ_ACCOUNTING_TOKEN_LENGTH bytes de datos binarios. Al utilizar valores predefinidos como MQACT_NONE, es posible que sea necesario realizar una conversión para garantizar que la firma coincida; por ejemplo, (MQBYTE *)MQACT_NONE.

ImqBinarycorrelationId () const;

Devuelve una copia del **identificador** de correlación.

ImqBooleansetCorrelationId (const ImqBinary & *token*);

Establece el **identificador de correlación**. La **longitud** de *datos del* token debe ser cero o MQ_CORREL_ID_LENGTH. Este método devuelve TRUE si se ejecuta correctamente.

void setCorrelationId (const MQBYTE24 *id* = 0);

Establece el **identificador de correlación**. *id* puede ser cero, lo que equivale a especificar MQCI_NONE. Si *id* es distinto de cero, debe direccionar MQ_CORREL_ID_LENGTH bytes de datos binarios. Al utilizar valores predefinidos como MQCI_NONE, es posible que sea necesario realizar un moldeado para garantizar que la firma coincida; por ejemplo, (MQBYTE *)MQCI_NONE.

MQLONG feedback () const;

Devuelve la **retroalimentación**.

void setFeedback (const MQLONG *feedback*);

Establece la **retroalimentación**.

ImqBinarygroupId () const;

Devuelve una copia del **identificador** del grupo.

ImqBooleansetGroupId (const ImqBinary & token);

Establece el **identificador del grupo**. La **longitud** de *datos del token* debe ser cero o MQ_GROUP_ID_LENGTH. Este método devuelve TRUE si se ejecuta correctamente.

void setGroupId (const MQBYTE24 id = 0);

Establece el **identificador del grupo**. *id* puede ser cero, lo que equivale a especificar MQGI_NONE. Si *id* es distinto de cero, debe abordar MQ_GROUP_ID_LENGTH bytes de datos binarios. Cuando se utilizan valores predefinidos como MQGI_NONE, es posible que sea necesario realizar una conversión para garantizar que la firma coincida, por ejemplo (MQBYTE *)MQGI_NONE.

ImqBinarymessageId () const;

Devuelve una copia del **identificador** del mensaje.

ImqBooleansetMessageId (const ImqBinary & token);

Establece el **identificador del mensaje**. La **longitud** de *datos del token* debe ser cero o MQ_MSG_ID_LENGTH. Este método devuelve TRUE si se ejecuta correctamente.

void setMessageId (const MQBYTE24 id = 0);

Establece el **identificador del mensaje**. *id* puede ser cero, lo que equivale a especificar MQMI_NONE. Si *id* es distinto de cero, debe direccionar MQ_MSG_ID_LENGTH bytes de datos binarios. Cuando se utilizan valores predefinidos como MQMI_NONE, es posible que sea necesario realizar una conversión para garantizar que la firma coincida, por ejemplo (MQBYTE *)MQMI_NONE.

códigos de razón

- ERROR_DE_LONGITUD_DE_DATOS_BINARIOS_MQRC

Clase C++ ImqNamelist

Esta clase encapsula una lista de nombres.

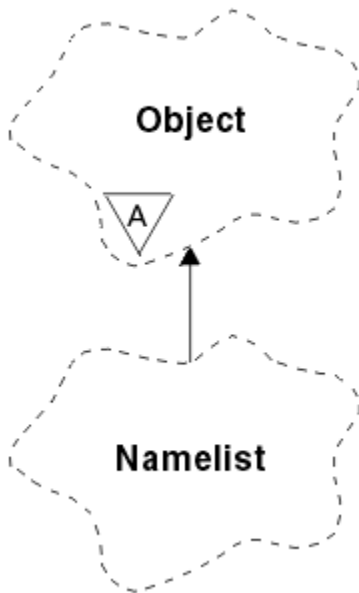


Figura 27. Clase ImqNamelist

Esta clase está relacionada con las llamadas de la clase `MQI` que se enumeran en [“Referencia cruzada de ImqNamelist”](#) en la página 1821.

- [“Atributos de objetos”](#) en la página 1876
- [“Constructores”](#) en la página 1876
- [“Métodos de objeto \(public\)”](#) en la página 1876
- [“códigos de razón”](#) en la página 1876

Atributos de objetos

Número de nombres

El número de nombres de objeto en **nombres de lista de nombres**. Este atributo es de sólo lectura.

Nombres de lista de nombres

Nombres de objeto, cuyo número se indica mediante el **recuento de nombres**. Este atributo es de sólo lectura.

Constructores

ImqNamelist();

El constructor predeterminado.

ImqNamelist(const ImqNamelist, *list*);

El constructor de copia. El **estado de apertura** de ImqObject es false.

ImqNamelist(const char * *nombre*);

Establece el nombre de ImqObject en **name**.

Métodos de objeto (public)

void operator = (const ImqNamelist & *lista*);

Copia los datos de instancia de la *lista*, sustituyendo los datos de instancia existentes. El **estado de apertura** de ImqObject es false.

ImqBoolean nameCount(MQLONG & *count*);

Proporciona una copia del **recuento de nombres**. Devuelve TRUE si es satisfactorio.

MQLONG nameCount ();

Devuelve el **recuento de nombres** sin ninguna indicación de posibles errores.

ImqBoolean namelistName (const MQLONG *índice*, ImqString & *nombre*);

Proporciona una copia de uno de los **nombres de lista de nombres** por índice basado en cero. Devuelve TRUE si es satisfactorio.

ImqString namelistName (const MQLONG *índice*);

Devuelve uno de los **nombres de lista de nombres** por índice basado en cero sin ninguna indicación de posibles errores.

códigos de razón

- MQRC_INDEX_ERROR
- MQRC_ÍNDICE_NO_EXISTE

ImqObject Clase C++

Esta clase es abstracta. Cuando se destruye un objeto de esta clase, se cierra automáticamente y se interrumpe su conexión ImqQueueManager.

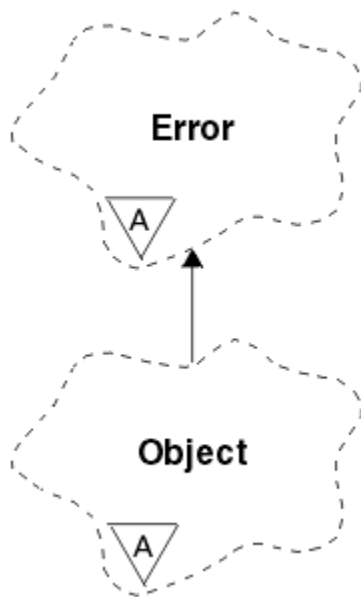


Figura 28. *ImqObject* clase

Esta clase está relacionada con las llamadas de la clase `MQI` que se enumeran en [“Referencia cruzada de ImqObject”](#) en la página 1821.

- [“Atributos de clase”](#) en la página 1877
- [“Atributos de objetos”](#) en la página 1877
- [“Constructores”](#) en la página 1879
- [“Métodos de clase \(públicos\)”](#) en la página 1879
- [“Métodos de objeto \(públicos\)”](#) en la página 1879
- [“Métodos de objeto \(protegidos\)”](#) en la página 1881
- [“Datos del objeto \(protegidos\)”](#) en la página 1882
- [“códigos de razón”](#) en la página 1882
-

Atributos de clase

comportamiento

Controla el comportamiento de la apertura implícita.

IMQ_IMPL_OPEN (8L)

Se permite la apertura implícita. Este es el valor predeterminado.

Atributos de objetos

Fecha de modificación

La fecha de la modificación. Este atributo es de sólo lectura.

Hora de modificación

El tiempo de alteración. Este atributo es de sólo lectura.

ID de usuario alternativo

El ID de usuario alternativo, con un máximo de MQ_USER_ID_LENGTH caracteres. El valor inicial es una cadena nula.

ID de seguridad alternativo

El ID de seguridad alternativo. Un valor binario (MQBYTE40) de longitud MQ_SECURITY_ID_LENGTH. El valor inicial es MQSID_NONE.

opciones de cierre

Opciones que se aplican cuando se cierra un objeto. El valor inicial es MQCO_NONE. Este atributo se ignora durante las operaciones de reapertura implícita, en las que siempre se utiliza el valor MQCO_NONE.

referencia de conexión

Una referencia a un objeto ImqQueueManager que proporciona la conexión necesaria a un gestor de colas (local). Para un objeto ImqQueueManager, es el propio objeto. El valor inicial es cero.

Nota: No confunda esto con el nombre del gestor de colas que identifica a un gestor de colas (posiblemente remoto) para una cola con nombre.

description

El nombre descriptivo (hasta 64 caracteres) del gestor de colas, la cola, la lista de nombres o el proceso. Este atributo es de sólo lectura.

nombre

El nombre (hasta 48 caracteres) del gestor de colas, la cola, la lista de nombres o el proceso. El valor inicial es una cadena nula. El nombre de una cola modelo cambia después de una **open** al nombre de la cola dinámica resultante.

Nota: Un gestor de colas (ImqQueueManager) puede tener un nombre nulo, que representa el gestor de colas predeterminado. El nombre cambia al gestor de colas real tras una apertura satisfactoria. Un objeto « ImqDistributionList » es dinámico y debe tener un nombre nulo.

siguiente objeto gestionado

Este es el siguiente objeto de esta clase, sin ningún orden en particular, que tiene la misma referencia de conexión que este objeto. El valor inicial es cero.

Opciones abiertas

Opciones que se aplican al abrir un objeto. El valor inicial es MQOO_INQUIRE. Hay dos formas de establecer los valores adecuados:

1. No establezca las opciones abiertas y no utilice el método abierto. IBM MQ Ajusta automáticamente las opciones de apertura y abre, vuelve a abrir y cierra objetos automáticamente según sea necesario. Esto puede dar lugar a operaciones de reapertura innecesarias, ya que IBM MQ utiliza el método openFor, y este solo añade opciones de apertura de forma incremental.
2. Configure las opciones de apertura antes de utilizar cualquier método que implique una llamada a `MQI` (véase “Referencia cruzada de C++ y MQI” en la página 1814). Esto garantiza que no se produzcan operaciones de reapertura innecesarias. Establezca las opciones de apertura de forma explícita si es probable que se produzca alguno de los posibles problemas de reapertura (consulte [Reapertura en consideraciones del lenguaje C++](#)).

Si utiliza el método abierto, primero debe asegurarse de que las opciones de apertura sean adecuadas. Sin embargo, el uso del método abierto no es obligatorio; IBM MQ sigue mostrando el mismo comportamiento que en el caso 1, pero en esta circunstancia, el comportamiento es eficiente.

Cero no es un valor válido; establezca el valor adecuado antes de intentar abrir el objeto. Esto se puede hacer utilizando **setOpenOptions** (*IOpenOptions*) seguido de **open** (), o **openFor** (*IRequiredOpenOption*).

Nota:

1. MQOO_OUTPUT se sustituye por MQOO_INQUIRE durante el **open** método para una lista de distribución, ya que MQOO_OUTPUT es el único válido **open option** en este momento. Sin embargo, es una buena práctica establecer siempre MQOO_OUTPUT explícitamente en los programas de aplicación que utilizan el **open** método.
2. Especifique MQOO_RESOLVE_NAMES si desea utilizar los **resolved queue manager name** atributos **resolved queue name** y de la clase.

Estado abierto

Si el objeto está abierto (TRUE) o cerrado (FALSE). El valor inicial es FALSE. Este atributo es de sólo lectura.

objeto gestionado anterior

El objeto anterior de esta clase, sin ningún orden en particular, que tiene la misma referencia de conexión que este objeto. El valor inicial es cero.

identificador-del-gestor-de-colas

El identificador del gestor de colas. Este atributo es de sólo lectura.

Constructores**ImqObject();**

El constructor predeterminado.

ImqObject(const ImqObject, *object*);

El constructor de copia. El estado abierto será FALSE.

Métodos de clase (públicos)**comportamiento estático MQLONG();**

Devuelve el comportamiento.

void setBehavior(const MQLONG *behavior* = 0);

Establece el comportamiento.

Métodos de objeto (públicos)**operador vacío = (const ImqObject & *objeto*);**

Cierra si es necesario y copia los datos de instancia del *objeto*. El estado abierto será FALSE.

ImqBoolean alterationDate(ImqString & *fecha*);

Proporciona una copia de la fecha de modificación. Devuelve TRUE si se realiza correctamente.

ImqString alterationDate();

Devuelve la fecha de modificación sin indicar posibles errores.

ImqBoolean alterationTime(ImqString & *tiempo*);

Proporciona una copia de la hora de la modificación. Devuelve TRUE si se realiza correctamente.

ImqString alterationTime();

Devuelve la hora de modificación sin indicar posibles errores.

ImqString alternateUserId () const;

Devuelve una copia del ID de usuario alternativo.

ImqBoolean setAlternateUserId (const char * *id*);

Establece el ID de usuario alternativo. El ID de usuario alternativo solo se puede configurar cuando el estado abierto es FALSE. Este método devuelve TRUE si se ejecuta correctamente.

ImqBinary alternateSecurityId() const;

Devuelve una copia del ID de seguridad alternativo.

ImqBoolean setAlternateSecurityId(const ImqBinary, *token*);

Establece el identificador de seguridad alternativo. La identificación de seguridad alternativa solo se puede configurar cuando el estado abierto es FALSE. La longitud de datos *del* token debe ser cero o MQ_SECURITY_ID_LENGTH. Devuelve TRUE si se realiza correctamente.

ImqBoolean setAlternateSecurityId(const MQBYTE* *token* = 0);

Establece el identificador de seguridad alternativo. *El token* puede ser cero, lo que equivale a especificar MQSID_NONE. Si *el token* es distinto de cero, debe abordar MQ_SECURITY_ID_LENGTH bytes de datos binarios. Al utilizar valores predefinidos como MQSID_NONE, es posible que sea necesario realizar una conversión para garantizar que la firma coincida; por ejemplo, (MQBYTE*)MQSID_NONE.

La identificación de seguridad alternativa solo se puede configurar cuando el estado abierto es TRUE. Devuelve TRUE si se realiza correctamente.

ImqBoolean setAlternateSecurityId(const unsigned char * *id* = 0);

Establece el identificador de seguridad alternativo.

ImqBoolean cerrar ();

Establece el estado abierto en FALSE. Devuelve TRUE si se realiza correctamente.

MQLONG closeOptions () const;

Devuelve las opciones de cierre.

void setCloseOptions (const MQLONG opciones);

Establece las opciones de cierre.

ImqQueueManager * connectionReference () const;

Devuelve la referencia de conexión.

void setConnectionReference (ImqQueueManager, manager);

Establece la referencia de conexión.

void setConnectionReference (ImqQueueManager *manager = 0);

Establece la referencia de conexión.

descripción del e ImqBoolean o virtual (ImqString &description) = 0;

Proporciona una copia de la descripción. Devuelve TRUE si se realiza correctamente.

ImqString descripción ();

Devuelve una copia de la descripción sin indicar posibles errores.

nombre virtual de ImqBoolean (ImqString &name);

Proporciona una copia del nombre. Devuelve TRUE si se realiza correctamente.

ImqString nombre ();

Devuelve una copia del nombre sin indicar posibles errores.

ImqBoolean setName (const char * nombre = 0);

Establece el nombre. El nombre solo se puede establecer mientras el estado abierto es FALSE y, para un ImqQueueManager,, mientras el estado de conexión es FALSE. Devuelve TRUE si se realiza correctamente.

ImqObject * nextManagedObject () const;

Devuelve el siguiente objeto gestionado.

ImqBoolean abrir ();

Cambia el estado abierto a TRUE abriendo el objeto según sea necesario, utilizando entre otros atributos las opciones de apertura y el nombre. Este método utiliza la información de referencia de la conexión y el método de conexión ImqQueueManager si es necesario para garantizar que el estado de la conexión ImqQueueManager sea TRUE. Devuelve el estado abierto.

ImqBoolean openFor (const MQLONG opciones-requeridas = 0);

Intenta garantizar que el objeto esté abierto con opciones abiertas, o con opciones abiertas que garanticen el comportamiento implícito en el valor del parámetro *required-options*.

Si *required-options* es cero, se requiere una entrada y cualquier opción de entrada es válida. Por lo tanto, si las opciones abiertas ya contienen una de las siguientes:

- MQOO_INPUT_AS_Q_DEF
- MQOO_INPUT_SHARED
- MQOO_INPUT_EXCLUSIVE

Las opciones abiertas ya son satisfactorias y no se modifican; si las opciones abiertas aún no contienen ninguna de estas opciones, se establece MQOO_INPUT_AS_Q_DEF en las opciones abiertas.

Si *required-options* es distinto de cero, las opciones requeridas se añaden a las opciones abiertas; si *required-options* es cualquiera de estas opciones, las demás se restablecen.

Si se modifica alguna de las opciones de apertura y el objeto ya está abierto, este se cerrará temporalmente y se volverá a abrir para ajustar las opciones de apertura.

Devuelve TRUE si se realiza correctamente. El éxito indica que el objeto está abierto con las opciones adecuadas.

MQLONG openOptions () const;

Devuelve las opciones abiertas.

ImqBoolean setOpenOptions (const MQLONG *opciones*);

Establece las opciones de apertura. Las opciones abiertas solo se pueden configurar mientras el estado abierto sea FALSE. Devuelve TRUE si se realiza correctamente.

ImqBoolean openStatus () const;

Devuelve el estado abierto.

ImqObject * previousManagedObject () const;

Devuelve el objeto gestionado anterior.

ImqBoolean queueManagerIdentifier(ImqString & *id*);

Proporciona una copia del identificador del gestor de colas. Devuelve TRUE si se realiza correctamente.

ImqString queueManagerIdentifier();

Devuelve el identificador del gestor de colas sin indicar posibles errores.

Métodos de objeto (protegidos)

virtual ImqBoolean closeTemporarily ();

Cierra un objeto de forma segura antes de volver a abrirlo. Devuelve TRUE si se realiza correctamente. Este método asume que el estado abierto es TRUE.

MQHCONN connectionHandle () const;

Devuelve el MQHCONN asociado con la referencia de conexión. Este valor es cero si no hay ninguna referencia de conexión o si el Administrador no está conectado.

ImqBoolean consultar (const MQLONG *int-attr*, MQLONG & *value*);

Devuelve un valor entero, cuyo índice es un valor MQIA_*. En caso de error, el valor se establece en MQIAV_UNDEFINED.

ImqBoolean consultar (const MQLONG *char-attr*, char * & *buffer*, const size_t *length*);

Devuelve una cadena de caracteres cuyo índice es un valor MQCA_*.

Nota: Ambos métodos devuelven solo un único valor de atributo. Si se requiere una *instantánea* de más de un valor, en la que los valores sean coherentes entre sí durante un instante, IBM MQ C++ no ofrece esta función y debe utilizar la llamada MQINQ con los parámetros adecuados.

virtual void openInformationDisperse ();

Dispersa la información de la sección variable de la estructura de datos MQOD inmediatamente después de una llamada MQOPEN.

virtual ImqBoolean openInformationPrepare ();

Prepara la información para la sección variable de la estructura de datos MQOD inmediatamente antes de una llamada MQOPEN y devuelve TRUE si se realiza correctamente.

ImqBoolean set (const MQLONG *int-attr*, const MQLONG *value*);

Establece un IBM MQ atributo entero.

ImqBoolean set (const MQLONG *atributo_carácter*, const char * *búfer*, const size_t *longitud_requerida*);

Establece un IBM MQ atributo de carácter.

void setNextManagedObject (const ImqObject * *objeto* = 0);

Establece el siguiente objeto gestionado.

Atención: Utilice esta función solo si está seguro de que no romperá la lista de objetos gestionados.

void setPreviousManagedObject (const ImqObject * *objeto* = 0);

Establece el objeto gestionado anterior.

Atención: Utilice esta función solo si está seguro de que no romperá la lista de objetos gestionados.

Datos del objeto (protegidos)

MQHOBj ohobj

El identificador del IBM MQ objeto (válido solo cuando el estado abierto es TRUE).

MQOD omqod

La estructura de datos MQOD integrada. La cantidad de almacenamiento asignada para esta estructura de datos es la requerida para una versión 2 de MQOD. Comprueba el número de versión (*omqod.Version*) y accede a los demás campos de la siguiente manera:

MQOD_VERSION_1

Se puede acceder a todos los demás campos de *omqod*.

MQOD_VERSION_2

Se puede acceder a todos los demás campos de *omqod*.

MQOD_VERSION_3

omqod.pmqod es un puntero a un MQOD más grande, asignado dinámicamente. No se puede acceder a ningún otro campo en *omqod*. Se puede acceder a todos *omqod.pmqod* los campos abordados por.

Nota: *omqod.pmqod.Version* puede ser inferior a *omqod.Version*, lo que indica que el IBM MQ MQI client tiene más funcionalidades que el IBM MQ servidor.

códigos de razón

- MQRC_ATTRIBUTO_BLOQUEADO
- MQRC_ESTADO_OBJETO_INCONSISTENTE
- MQRC_SIN_REFERENCIA_DE_CONEXIÓN
- MQRC_STORAGE_NOT_AVAILABLE
- MQRC_REABRIR_CONTEXTO_GUARDADO_ERROR
- (códigos de motivo de MQCLOSE)
- (códigos de motivo de MQCONN)
- (códigos de motivo de MQINQ)
- (códigos de motivo de MQOPEN)
- (códigos de motivo de MQSET)

Clase C++ ImqProcess

Esta clase encapsula un proceso de aplicación (un objeto IBM MQ de tipo MQOT_PROCESS) que puede desencadenar un supervisor desencadenante.

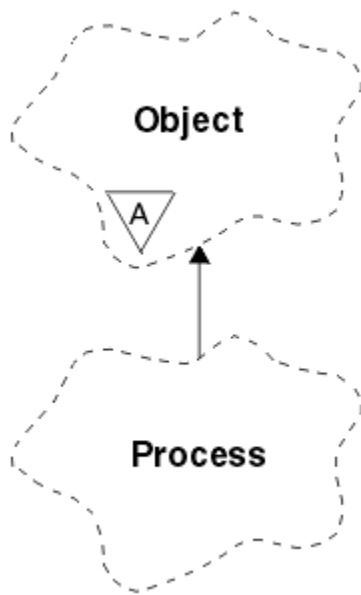


Figura 29. Clase *ImqProcess*

- “Atributos de objetos” en la [página 1883](#)
- “Constructores” en la [página 1883](#)
- “Métodos de objeto (public)” en la [página 1883](#)

Atributos de objetos

ID de aplicación

La identidad del proceso de aplicación. Este atributo es de sólo lectura.

Tipo de aplicación

El tipo del proceso de aplicación. Este atributo es de sólo lectura.

Datos de entorno

La información de entorno para el proceso. Este atributo es de sólo lectura.

datos de usuario

Datos de usuario para el proceso. Este atributo es de sólo lectura.

Constructores

ImqProcess();

El constructor predeterminado.

ImqProcess(const ImqProcess, *process*);

El constructor de copia. El **estado abierto** de ImqObject es FALSE.

ImqProcess(const char * *nombre*);

Establece el **nombre** ImqObject .

Métodos de objeto (public)

void operator = (const ImqProcess & *proceso*);

Realiza un cierre si es necesario y, a continuación, copia los datos de instancia del *proceso*. El **estado abierto** de ImqObject será FALSE.

ImqBooleanapplicationId (ImqString & *id*);

Proporciona una copia del **ID de aplicación**. Devuelve TRUE si es satisfactorio.

ImqString applicationId ();

Devuelve el **ID de aplicación** sin ninguna indicación de posibles errores.

ImqBoolean applicationType (MQLONG & tipo);

Proporciona una copia del **tipo de aplicación**. Devuelve TRUE si es satisfactorio.

MQLONG applicationType ();

Devuelve el **tipo de aplicación** sin ninguna indicación de posibles errores.

ImqBoolean environmentData (ImqString & datos);

Proporciona una copia de los **datos de entorno**. Devuelve TRUE si es satisfactorio.

ImqString environmentData ();

Devuelve los **datos de entorno** sin ninguna indicación de posibles errores.

ImqBoolean userData (ImqString & datos);

Proporciona una copia de los **datos de usuario**. Devuelve TRUE si es satisfactorio.

ImqString userData ();

Devuelve los **datos de usuario** sin ninguna indicación de posibles errores.

ImqPutMessageOptions Clase C

Esta clase encapsula la estructura de datos MQPMO.

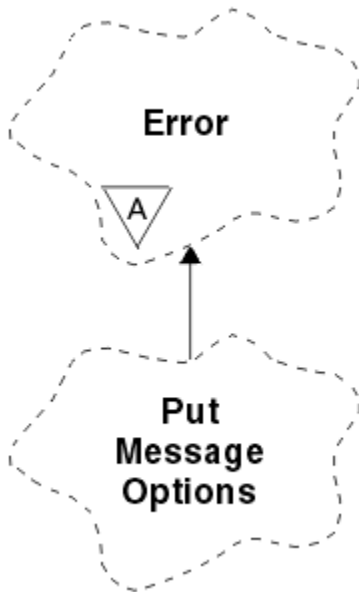


Figura 30. ImqPutMessageOptions clase

- [“Atributos de objetos” en la página 1884](#)
- [“Constructores” en la página 1885](#)
- [“Métodos de objeto \(public\)” en la página 1885](#)
- [“Datos de objeto \(protegidos\)” en la página 1886](#)
- [“códigos de razón” en la página 1886](#)

Atributos de objetos

referencia de contexto

Una ImqQueue que proporciona un contexto para los mensajes. Inicialmente no hay ninguna referencia.

opciones

Las opciones de colocación de mensaje. El valor inicial es MQPMO_NONE. Son posibles los siguientes valores adicionales:

- MQPMO_SYNCPOINT
- MQPMO_NO_SYNCPOINT

- MQPMO_ID_NUEVO_MENSAJE
- MQPMO_NEW_CORREL_ID
- MQPMO_LOGICAL_ORDER
- MQPMO_NO_CONTEXT
- MQPMO_DEFAULT_CONTEXT
- MQPMO_PASS_IDENTITY_CONTEXT
- MQPMO_PASS_ALL_CONTEXT
- MQPMO_SET_IDENTITY_CONTEXT
- MQPMO_SET_ALL_CONTEXT
- MQPMO_ALTERNATE_USER_AUTHORITY
- MQPMO_FAIL_IF QUIESCING

campos de registro

Los distintivos que controlan la inclusión de registros de mensajes de colocación cuando se coloca un mensaje. El valor inicial es MQPMRF_NONE. Son posibles los siguientes valores adicionales:

- MQPMRF_MSG_ID
- ID MQPMRF_CORREL_ID
- MQPMRF_ID_grupo
- MQPMRF_COMENTARIOS
- MQPMRF_TOKEN_CONTABLE

ImqMessageTracker se toman del objeto para cualquier campo que se especifique.

ImqMessageTracker se toman del objeto ImqMessage para cualquier campo que no se especifique.

Nombre del gestor de colas resuelto

Nombre de un gestor de colas de destino determinado durante una colocación. El valor inicial es nulo. Este atributo es de sólo lectura.

Nombre de cola resuelto

Nombre de una cola de destino determinada durante una colocación. El valor inicial es nulo. Este atributo es de sólo lectura.

participación de punto de sincronismo

TRUE cuando los mensajes se ponen bajo control de punto de sincronismo.

Constructores

ImqPutMessageOptions();

El constructor predeterminado.

ImqPutMessageOptions(const ImqPutMessageOptions, *pmo*);

El constructor de copia.

Métodos de objeto (public)

void operador = (const ` ImqPutMessageOptions ` & *pmo*);

Copia datos de instancia de *pmo*, sustituyendo los datos de instancia existentes.

ImqQueue * contextReference () Const.

Devuelve la referencia de contexto.

void setContextReference (const ImqQueue & *queue*);

Establece la referencia de contexto.

void setContextReference (const ImqQueue * *queue* = 0);

Establece la referencia de contexto.

Opciones MQLONG () Const.

Devuelve las opciones.

void setOptions (const MQLONG opciones);

Establece las opciones, incluido el valor de participación de punto de sincronismo.

MQLONG recordFields () Const.

Devuelve los campos de registro.

void setRecordFields (const MQLONG campos);

Establece los campos de registro.

ImqString resolvedQueueManagerName () const;

Devuelve una copia del nombre del gestor de colas resuelto.

ImqString resolvedQueueName () const;

Devuelve una copia del nombre de cola resuelto.

ImqBoolean syncPointParticipation () const;

Devuelve el valor de participación de punto de sincronismo, que es TRUE si las opciones incluyen MQPMO_SYNCPOINT.

void setSyncPointParticipation (const ImqBoolean sync);

Establece el valor de participación de punto de sincronismo. Si *sync* es TRUE, las opciones se modifican para incluir MQPMO_SYNCPOINT y para excluir MQPMO_NO_SYNCPOINT. Si *sync* es FALSE, las opciones se modifican para incluir MQPMO_NO_SYNCPOINT y excluir MQPMO_SYNCPOINT.

Datos de objeto (protegidos)

MQPMO omqpmo

Estructura de datos MQPMO.

códigos de razón

- MQRC_STORAGE_NOT_AVAILABLE

Clase C++ ImqQueue

Esta clase encapsula una cola de mensajes (un objeto IBM MQ de tipo MQOT_Q).

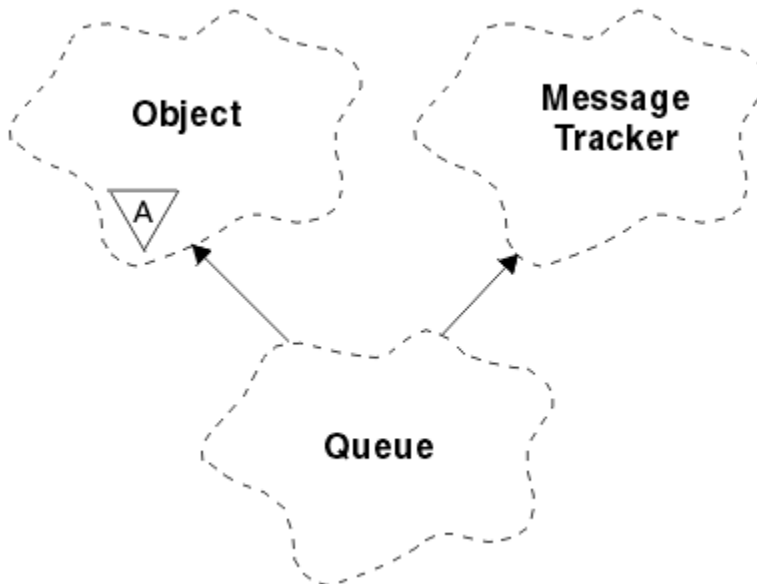


Figura 31. Clase ImqQueue

Esta clase está relacionada con las llamadas de la clase `MQI` que se enumeran en [Tabla 898](#) en la [página 1822](#).

- [“Atributos de objetos” en la página 1887](#)

- [“Constructores” en la página 1890](#)
- [“Métodos de objeto \(public\)” en la página 1890](#)
- [“Métodos de objeto \(protegidos\)” en la página 1896](#)
- [“códigos de razón” en la página 1896](#)

Atributos de objetos

Nombre de reposición en cola para restitución

Nombre de reposición en cola de restitución excesivo. Este atributo es de sólo lectura.

umbral de restituciones

El umbral de restituciones. Este atributo es de sólo lectura.

nombre de cola base

Nombre de la cola en la que se resuelve el alias. Este atributo es de sólo lectura.

nombre de clúster

Nombre de clúster. Este atributo es de sólo lectura.

Nombre de la lista de nombres del clúster

Nombre de lista de nombres de clúster. Este atributo es de sólo lectura.

Rango de carga trabajo del clúster

El rango de la carga de trabajo del clúster. Este atributo es de sólo lectura.

Prioridad de carga de trabajo del clúster

La prioridad de la carga de trabajo del clúster. Este atributo es de sólo lectura.

Cola de uso de carga de trabajo de clúster

La carga de trabajo de clúster utiliza el valor de cola. Este atributo es de sólo lectura.

fecha de creación

Datos de creación de cola. Este atributo es de sólo lectura.

Hora de creación

Hora de creación de cola. Este atributo es de sólo lectura.

Profundidad actual

Número de mensajes en la cola. Este atributo es de sólo lectura.

enlace predeterminado

Enlace predeterminado. Este atributo es de sólo lectura.

Opción abierta de entrada predeterminada

Opción de apertura para entrada predeterminada. Este atributo es de sólo lectura.

Persistencia predeterminada

Persistencia de mensajes predeterminada. Este atributo es de sólo lectura.

Prioridad predeterminada

Prioridad de mensaje predeterminada. Este atributo es de sólo lectura.

Tipo de definición

El tipo de definición de la cola. Este atributo es de sólo lectura.

suceso de profundidad alta

Atributo de control para sucesos de profundidad de cola alta. Este atributo es de sólo lectura.

límite alto de profundidad

Límite alto para la profundidad de cola. Este atributo es de sólo lectura.

suceso de profundidad baja

Atributo de control para sucesos de profundidad de cola baja. Este atributo es de sólo lectura.

límite bajo de profundidad

Límite bajo para la profundidad de cola. Este atributo es de sólo lectura.

suceso de profundidad máxima

Atributo de control para sucesos máximos de profundidad de cola. Este atributo es de sólo lectura.

referencia de lista de distribución

Referencia opcional a un `ImqDistributionList` que puede utilizarse para distribuir mensajes a más de una cola, incluida ésta. El valor inicial es nulo.

Nota: Cuando se abre un objeto `ImqQueue`, cualquier objeto `ImqDistributionList` abierto al que haga referencia se cierra automáticamente.

listas de distribución

Capacidad de una cola de transmisión para dar soporte a listas de distribución. Este atributo es de sólo lectura.

Nombre de la cola dinámica

Nombre de cola dinámica. El valor inicial es `AMQ.*` para todas las plataformas AIX, Linux, and Windows .

Copia en disco de restitución de obtención

Indica si se debe reforzar el recuento de restituciones. Este atributo es de sólo lectura.

Tipo de índice

Tipo de índice. Este atributo es de sólo lectura.

Inhibir obtención

Si se permiten operaciones `get`. El valor inicial depende de la definición de cola. Este atributo sólo es válido para un alias o cola local.

Inhibir colocación

Si se permiten las operaciones de colocación. El valor inicial depende de la definición de cola.

Nombre cola iniciación

Nombre de la cola de inicio. Este atributo es de sólo lectura.

Profundidad máxima

Número máximo de mensajes permitidos en la cola. Este atributo es de sólo lectura.

longitud máxima del mensaje

Longitud máxima para cualquier mensaje de esta cola, que puede ser menor que el máximo para cualquier cola gestionada por el gestor de colas asociado. Este atributo es de sólo lectura.

Secuencia de entrega de mensajes

Indica si la prioridad del mensaje es relevante. Este atributo es de sólo lectura.

siguiente cola distribuida

El siguiente objeto de esta clase, en ningún orden concreto, que tenga la misma **referencia de lista de distribución** que este objeto. El valor inicial es cero.

Si se suprime un objeto de una cadena, el objeto anterior y el siguiente objeto se actualizan para que sus enlaces de cola distribuida ya no apunten al objeto suprimido.

clase de mensaje no persistente

Nivel de fiabilidad para los mensajes no persistentes colocados en esta cola. Este atributo es de sólo lectura.

número entradas abiertas

Número de objetos `ImqQueue` que están abiertos para entrada. Este atributo es de sólo lectura.

número salidas abiertas

Número de objetos `ImqQueue` que están abiertos para salida. Este atributo es de sólo lectura.

cola distribuida anterior

Objeto anterior de esta clase, en ningún orden concreto, que tenga la misma **referencia de lista de distribución** que este objeto. El valor inicial es cero.

Si se suprime un objeto de una cadena, el objeto anterior y el siguiente objeto se actualizan para que sus enlaces de cola distribuida ya no apunten al objeto suprimido.

nombre de proceso

Nombre de la definición de proceso. Este atributo es de sólo lectura.

Contabilidad de la cola

Nivel de información de contabilidad para colas. Este atributo es de sólo lectura.

nombre-gestor-colas

Nombre del gestor de colas (posiblemente remoto) donde reside la cola. No confunda el gestor de colas nombrado aquí con la **referencia de conexión** `ImqObject` , que hace referencia al gestor de colas (local) que proporciona una conexión. El valor inicial es nulo.

Supervisión de la cola

Nivel de recopilación de datos de supervisión para la cola. Este atributo es de sólo lectura.

estadísticas de cola

Nivel de datos de estadísticas para la cola. Este atributo es de sólo lectura.

Tipo de cola

El tipo de cola. Este atributo es de sólo lectura.

Nombre de gestor de colas remoto

Nombre del gestor de colas remoto. Este atributo es de sólo lectura.

Nombre de cola remota

Nombre de la cola remota tal como se conoce en el gestor de colas remoto. Este atributo es de sólo lectura.

Nombre del gestor de colas resuelto

Nombre de gestor de colas resuelto. Este atributo es de sólo lectura.

Nombre de cola resuelto

Nombre de cola resuelto. Este atributo es de sólo lectura.

Intervalo de retención

Intervalo de retención de cola. Este atributo es de sólo lectura.

ámbito

Ámbito de la definición de cola. Este atributo es de sólo lectura.

intervalo de servicio

Intervalo de servicio. Este atributo es de sólo lectura.

suceso de intervalo de servicio

Atributo de control para sucesos de intervalo de servicio. Este atributo es de sólo lectura.

Posibilidad de compartición

Si la cola puede compartirse. Este atributo es de sólo lectura.

clase de almacenamiento

Clase de almacenamiento. Este atributo es de sólo lectura.

Nombre de cola de transmisión

Nombre de la cola de transmisión. Este atributo es de sólo lectura.

Control de desencadenante

Control de desencadenante. El valor inicial depende de la definición de cola. Este atributo sólo es válido para una cola local.

Datos desencadenantes

Datos del desencadenante. El valor inicial depende de la definición de cola. Este atributo sólo es válido para una cola local.

Profundidad de desencadenante

Profundidad del desencadenante. El valor inicial depende de la definición de cola. Este atributo sólo es válido para una cola local.

Prioridad de mensajes desencadenantes

Umbral de prioridad del mensaje para activaciones. El valor inicial depende de la definición de cola. Este atributo sólo es válido para una cola local.

tipo de activador

Tipo de desencadenante. El valor inicial depende de la definición de cola. Este atributo sólo es válido para una cola local.

uso

Uso. Este atributo es de sólo lectura.

Constructores

ImqQueue();

El constructor predeterminado.

ImqQueue(const ImqQueue, *queue*);

El constructor de copia. El **estado abierto** de ImqObject será FALSE.

ImqQueue(const char * *nombre*);

Establece el **nombre** ImqObject .

Métodos de objeto (public)

void operator = (const ImqQueue & *cola*);

Realiza un cierre si es necesario y, a continuación, copia los datos de instancia de la *cola*. El **estado abierto** de ImqObject será FALSE.

ImqBoolean backoutRequeueName (ImqString & *nombre*);

Proporciona una copia del **nombre de reposición en cola de restitución**. Devuelve TRUE si es satisfactorio.

ImqString backoutRequeueName ();

Devuelve el **nombre de reposición en cola de restitución** sin ninguna indicación de posibles errores.

ImqBoolean backoutThreshold (MQLONG & *umbral*);

Proporciona una copia del **umbral de restitución**. Devuelve TRUE si es satisfactorio.

MQLONG backoutThreshold ();

Devuelve el valor de **umbral de restitución** sin ninguna indicación de posibles errores.

ImqBoolean baseQueueName (ImqString & *nombre*);

Proporciona una copia del **nombre de cola base**. Devuelve TRUE si es satisfactorio.

ImqString baseQueueName ();

Devuelve el **nombre de cola base** sin ninguna indicación de posibles errores.

ImqBoolean clusterName(ImqString & *nombre*);

Proporciona una copia del **nombre de clúster**. Devuelve TRUE si es satisfactorio.

ImqString clusterName();

Devuelve el **nombre de clúster** sin ninguna indicación de posibles errores.

ImqBoolean clusterNamelistName(ImqString & *nombre*);

Proporciona una copia del **nombre de lista de nombres de clúster**. Devuelve TRUE si es satisfactorio.

ImqString clusterNamelistName();

Devuelve el **nombre de lista de nombres de clúster** sin ninguna indicación de errores.

ImqBoolean clusterWorkLoadPriority (MQLONG & *prioridad*);

Proporciona una copia del valor de prioridad de carga de trabajo de clúster. Devuelve TRUE si es satisfactorio.

MQLONG clusterWorkLoadPriority ();

Devuelve el valor de prioridad de carga de trabajo de clúster sin ninguna indicación de posibles errores.

ImqBoolean clusterWorkLoadRank (MQLONG & *rango*);

Proporciona una copia del valor de rango de carga de trabajo de clúster. Devuelve TRUE si es satisfactorio.

MQLONG clusterWorkLoadRank ();

Devuelve el valor de rango de carga de trabajo de clúster sin ninguna indicación de posibles errores.

ImqBoolean clusterWorkLoadUseQ (MQLONG & *useq*);

Proporciona una copia del valor de cola de uso de carga de trabajo de clúster. Devuelve TRUE si es satisfactorio.

MQLONG clusterWorkLoadUseQ ();

Devuelve el valor de cola de uso de carga de trabajo de clúster sin ninguna indicación de posibles errores.

ImqBoolean creationDate (ImqString & fecha);

Proporciona una copia de la **fecha de creación**. Devuelve TRUE si es satisfactorio.

ImqString creationDate ();

Devuelve la **fecha de creación** sin ninguna indicación de posibles errores.

ImqBoolean creationTime (ImqString & hora);

Proporciona una copia de la **hora de creación**. Devuelve TRUE si es satisfactorio.

ImqString creationTime ();

Devuelve la **hora de creación** sin ninguna indicación de posibles errores.

ImqBoolean currentDepth (MQLONG & profundidad);

Proporciona una copia de la **profundidad actual**. Devuelve TRUE si es satisfactorio.

MQLONG currentDepth ();

Devuelve la **profundidad actual** sin ninguna indicación de posibles errores.

ImqBooleandefaultInputOpenOption (MQLONG & opción);

Proporciona una copia de la **opción de apertura de entrada predeterminada**. Devuelve TRUE si es satisfactorio.

MQLONG defaultInputOpenOption ();

Devuelve la **opción de apertura de entrada predeterminada** sin ninguna indicación de posibles errores.

ImqBoolean defaultPersistence (MQLONG & persistencia);

Proporciona una copia de la **persistencia predeterminada**. Devuelve TRUE si es satisfactorio.

MQLONG defaultPersistence ();

Devuelve la **persistencia predeterminada** sin ninguna indicación de posibles errores.

ImqBoolean defaultPriority (MQLONG & prioridad);

Proporciona una copia de la **prioridad predeterminada**. Devuelve TRUE si es satisfactorio.

MQLONG defaultPriority ();

Devuelve la **prioridad predeterminada** sin ninguna indicación de posibles errores.

ImqBooleandefaultBind (MQLONG & bind);

Proporciona una copia del **enlace predeterminado**. Devuelve TRUE si es satisfactorio.

MQLONG defaultBind ();

Devuelve el **enlace predeterminado** sin ninguna indicación de posibles errores.

ImqBoolean definitionType (MQLONG & tipo);

Proporciona una copia del **tipo de definición**. Devuelve TRUE si es satisfactorio.

MQLONG definitionType ();

Devuelve el **tipo de definición** sin ninguna indicación de posibles errores.

ImqBooleandepthHighEvent (MQLONG & evento);

Proporciona una copia del estado de habilitación del **suceso de profundidad alta**. Devuelve TRUE si es satisfactorio.

MQLONG depthHighEvent ();

Devuelve el estado de habilitación del **suceso de profundidad alta** sin ninguna indicación de posibles errores.

ImqBooleandepthHighLimit (MQLONG & limit);

Proporciona una copia del **límite alto de profundidad**. Devuelve TRUE si es satisfactorio.

MQLONG depthHighLimit ();

Devuelve el valor de **límite superior de profundidad** sin ninguna indicación de posibles errores.

ImqBooleandepthLowEvent (MQLONG & evento);

Proporciona una copia del estado de habilitación del **suceso de profundidad baja**. Devuelve TRUE si es satisfactorio.

MQLONG depthLowEvent ();

Devuelve el estado de habilitación del **suceso de profundidad baja** sin ninguna indicación de posibles errores.

ImqBooleandepthLowLimit (MQLONG & *limit*);

Proporciona una copia del **límite bajo de profundidad**. Devuelve TRUE si es satisfactorio.

MQLONG depthLowLimit ();

Devuelve el valor de **límite inferior de profundidad** sin ninguna indicación de posibles errores.

ImqBooleandepthMaximumEvent (MQLONG & *evento*);

Proporciona una copia del estado de habilitación del **suceso de profundidad máxima**. Devuelve TRUE si es satisfactorio.

MQLONG depthMaximumEvent ();

Devuelve el estado de habilitación del **suceso máximo de profundidad** sin ninguna indicación de posibles errores.

ImqDistributionList * distributionListReference () const;

Devuelve la **referencia de lista de distribución**.

void setDistributionListReference (ImqDistributionList & *list*);

Establece la **referencia de lista de distribución**.

void setDistributionListReference (ImqDistributionList * *lista* = 0);

Establece la **referencia de lista de distribución**.

ImqBoolean distributionLists (Soporte de MQLONG &);

Proporciona una copia del valor de **listas de distribución** . Devuelve TRUE si es satisfactorio.

MQLONG distributionLists ();

Devuelve el valor de **listas de distribución** sin ninguna indicación de posibles errores.

ImqBooleansetDistributionLists (const MQLONG *soporte*);

Establece el valor de **listas de distribución** . Devuelve TRUE si es satisfactorio.

ImqStringdynamicQueueName () const;

Devuelve una copia del **nombre de cola dinámica**.

ImqBooleansetDynamicQueueName (const char * *nombre*);

Establece el **nombre de cola dinámica**. El **nombre de cola dinámica** solo se puede establecer mientras el **estado abierto** de ImqObject es FALSE. Devuelve TRUE si es satisfactorio.

ImqBoolean get (ImqMessage & *msg*, ImqGetMessageOptions & *options*);

Recupera un mensaje de la cola, utilizando las *opciones* especificadas. Invoca el método ImqObject **openFor** si es necesario para asegurarse de que las **opciones de apertura** ImqObject incluyen uno de los valores MQOO_INPUT_* o el valor MQOO_BROWSE, en función de las *opciones*. Si el objeto *msg* tiene un **almacenamiento intermedio automático** ImqCache , el almacenamiento intermedio crece para acomodar cualquier mensaje recuperado. El método **clearMessage** se invoca en el objeto *msg* antes de la recuperación.

Este método devuelve TRUE si es satisfactorio.

Nota: El resultado de la invocación del método es FALSE si el **código de razón** ImqObject es MQRC_TRUNCATED_MSG_FAILED, aunque este **código de razón** se clasifique como aviso. Si se acepta un mensaje truncado, la **longitud de mensaje** ImqCache refleja la longitud truncada. En cualquier caso, la **longitud total de mensaje** ImqMessage indica el número de bytes que estaban disponibles.

ImqBoolean get (ImqMessage, *msg*);

En cuanto al método anterior, excepto que se utilizan las opciones de obtención de mensaje predeterminadas.

ImqBoolean get (ImqMessage & *msg*, ImqGetMessageOptions & *options*, const size_t *buffer-size*);

En cuanto a los dos métodos anteriores, excepto que se indica una alteración temporal de *tamaño-búfer* . Si el objeto *msg* utiliza un **almacenamiento intermedio automático** ImqCache , el método **resizeBuffer** se invoca en el objeto *msg* antes de la recuperación de mensajes, y el almacenamiento intermedio no crece más para acomodar ningún mensaje más grande.

ImqBoolean get (ImqMessage & *msg*, const size_t *tamaño-búfer*);

En cuanto al método anterior, excepto que se utilizan las opciones de obtención de mensaje predeterminadas.

ImqBooleanhardenGetBackout (MQLONG & *endurecer*);

Proporciona una copia del valor **harden get backout** . Devuelve TRUE si es satisfactorio.

MQLONG hardenGetBackout ();

Devuelve el valor **harden get backout** sin ninguna indicación de posibles errores.

ImqBoolean indexType(MQLONG & *tipo*);

Proporciona una copia del **tipo de índice**. Devuelve TRUE si es satisfactorio.

MQLONG indexType();

Devuelve el **tipo de índice** sin ninguna indicación de posibles errores.

ImqBoolean inhibitGet (MQLONG & *inhibir*);

Proporciona una copia del valor **Impedir obtención** . Devuelve TRUE si es satisfactorio.

MQLONG inhibitGet ();

Devuelve el valor **Impedir obtención** sin ninguna indicación de posibles errores.

ImqBooleansetInhibitGet (const MQLONG *inhibit*);

Establece el valor **Impedir obtención** . Devuelve TRUE si es satisfactorio.

ImqBoolean inhibitPut (MQLONG & *inhibir*);

Proporciona una copia del valor **Impedir colocación** . Devuelve TRUE si es satisfactorio.

MQLONG inhibitPut ();

Devuelve el valor **Impedir colocación** sin ninguna indicación de posibles errores.

ImqBooleansetInhibitPut (const MQLONG *inhibit*);

Establece el valor **Impedir colocación** . Devuelve TRUE si es satisfactorio.

ImqBooleaninitiationQueueName (ImqString & *nombre*);

Proporciona una copia del **nombre de cola de inicio**. Devuelve TRUE si es satisfactorio.

ImqString initiationQueueName ();

Devuelve el **nombre de cola de inicio** sin ninguna indicación de posibles errores.

ImqBoolean maximumDepth (MQLONG & *profundidad*);

Proporciona una copia de la **profundidad máxima**. Devuelve TRUE si es satisfactorio.

MQLONG maximumDepth ();

Devuelve la **profundidad máxima** sin ninguna indicación de posibles errores.

ImqBooleanmaximumMessageLength (MQLONG & *longitud*);

Proporciona una copia de la **longitud máxima de mensaje**. Devuelve TRUE si es satisfactorio.

MQLONG maximumMessageLength ();

Devuelve la **longitud máxima de mensaje** sin ninguna indicación de posibles errores.

ImqBooleanmessageDeliverySequence (MQLONG & *secuencia*);

Proporciona una copia de la **secuencia de entrega de mensajes**. Devuelve TRUE si es satisfactorio.

MQLONG messageDeliverySequence ();

Devuelve el valor de **secuencia de entrega de mensajes** sin ninguna indicación de posibles errores.

ImqQueue * nextDistributedQueue () const;

Devuelve la **siguiente cola distribuida**.

ImqBoolean nonPersistentMessageClass (MQLONG y monq);

Proporciona una copia del valor de clase de mensaje no persistente. Devuelve TRUE si es satisfactorio.

MQLONG nonPersistentMessageClass ();

Devuelve el valor de clase de mensaje no persistente sin ninguna indicación de posibles errores.

ImqBooleanopenInputCount (MQLONG & *count*);

Proporciona una copia del **recuento de entradas abiertas**. Devuelve TRUE si es satisfactorio.

MQLONG openInputCount ();

Devuelve el **recuento de entradas abiertas** sin ninguna indicación de posibles errores.

ImqBooleanopenOutputCount (MQLONG & *count*);

Proporciona una copia del **recuento de salidas abiertas**. Devuelve TRUE si es satisfactorio.

MQLONG openOutputCount ();

Devuelve el **recuento de salidas abiertas** sin ninguna indicación de posibles errores.

ImqQueue * previousDistributedQueue () const;

Devuelve la **cola distribuida anterior**.

ImqBoolean processName (ImqString & nombre);

Proporciona una copia del **nombre de proceso**. Devuelve TRUE si es satisfactorio.

ImqString processName ();

Devuelve el **nombre de proceso** sin ninguna indicación de posibles errores.

ImqBoolean put (ImqMessage, msg);

Coloca un mensaje en la cola, utilizando las opciones de colocación de mensaje predeterminadas.

Utiliza el método ImqObject **openFor** si es necesario para asegurarse de que las **opciones de apertura** de ImqObject incluyen MQOO_OUTPUT.

Este método devuelve TRUE si es satisfactorio.

ImqBoolean put (ImqMessage, msg, ImqPutMessageOptions, pmo);

Coloca un mensaje en la cola, utilizando el *pmo* especificado. Utiliza el método

ImqObject **openFor** según sea necesario para asegurarse de que las **opciones de apertura** de ImqObject incluyen MQOO_OUTPUT y (si las **opciones** de *pmo* incluyen alguno de los valores de MQPMO_PASS_IDENTITY_CONTEXT, MQPMO_PASS_ALL_CONTEXT, MQPMO_SET_IDENTITY_CONTEXT o MQPMO_SET_ALL_CONTEXT correspondientes) MQOO_*_CONTEXT.

Este método devuelve TRUE si es satisfactorio.

Nota: Si *pmo* incluye una **referencia de contexto**, el objeto referenciado se abre, si es necesario, para proporcionar un contexto.

ImqBoolean queueAccounting (MQLONG & acctq);

Proporciona una copia del valor de contabilidad de cola. Devuelve TRUE si es satisfactorio.

MQLONG queueAccounting ();

Devuelve el valor de contabilidad de cola sin ninguna indicación de posibles errores.

ImqStringqueueManagerName () const;

Devuelve el **nombre de gestor de colas**.

ImqBooleansetQueueManagerName (const char * nombre);

Establece el **nombre de gestor de colas**. El **nombre de gestor de colas** sólo se puede establecer mientras el **estado abierto** de ImqObject es FALSE. Este método devuelve TRUE si es satisfactorio.

ImqBoolean queueMonitoring (MQLONG & monq);

Proporciona una copia del valor de supervisión de cola. Devuelve TRUE si es satisfactorio.

MQLONG queueMonitoring ();

Devuelve el valor de supervisión de cola sin ninguna indicación de posibles errores.

ImqBoolean queueStatistics (MQLONG & statq);

Proporciona una copia del valor de estadísticas de cola. Devuelve TRUE si es satisfactorio.

MQLONG queueStatistics ();

Devuelve el valor de estadísticas de cola sin ninguna indicación de posibles errores.

ImqBoolean queueType (MQLONG & tipo);

Proporciona una copia del valor de **tipo de cola** . Devuelve TRUE si es satisfactorio.

MQLONG queueType ();

Devuelve el **tipo de cola** sin ninguna indicación de posibles errores.

ImqBooleanremoteQueueManagerName (ImqString & nombre);

Proporciona una copia del **nombre de gestor de colas remoto**. Devuelve TRUE si es satisfactorio.

ImqString remoteQueueManagerName ();

Devuelve el **nombre de gestor de colas remoto** sin ninguna indicación de posibles errores.

ImqBooleanremoteQueueName (ImqString & nombre);

Proporciona una copia del **nombre de cola remota**. Devuelve TRUE si es satisfactorio.

ImqString remoteQueueName ();

Devuelve el **nombre de cola remota** sin ninguna indicación de posibles errores.

ImqBoolean resolvedQueueManagerName(ImqString & nombre);

Proporciona una copia del **nombre de gestor de colas resuelto**. Devuelve TRUE si es satisfactorio.

Nota: Este método falla a menos que MQOO_RESOLVE_NAMES esté entre las **opciones de apertura** ImqObject .

ImqString resolvedQueueManagerName();

Devuelve el **nombre de gestor de colas resuelto**, sin ninguna indicación de posibles errores.

ImqBoolean resolvedQueueName(ImqString & nombre);

Proporciona una copia del **nombre de cola resuelto**. Devuelve TRUE si es satisfactorio.

Nota: Este método falla a menos que MQOO_RESOLVE_NAMES esté entre las **opciones de apertura** ImqObject .

ImqString resolvedQueueName();

Devuelve el **nombre de cola resuelto**, sin ninguna indicación de posibles errores.

ImqBoolean retentionInterval (MQLONG & intervalo);

Proporciona una copia del **intervalo de retención**. Devuelve TRUE si es satisfactorio.

MQLONG retentionInterval ();

Devuelve el **intervalo de retención** sin ninguna indicación de posibles errores.

ImqBoolean scope (MQLONG & ámbito);

Proporciona una copia del **ámbito**. Devuelve TRUE si es satisfactorio.

MQLONG Ámbito ();

Devuelve el **ámbito** sin ninguna indicación de posibles errores.

ImqBoolean serviceInterval (MQLONG & intervalo);

Proporciona una copia del **intervalo de servicio**. Devuelve TRUE si es satisfactorio.

MQLONG serviceInterval ();

Devuelve el **intervalo de servicio** sin ninguna indicación de posibles errores.

ImqBooleanserviceIntervalEvent (MQLONG & evento);

Proporciona una copia del estado de habilitación del **suceso de intervalo de servicio**. Devuelve TRUE si es satisfactorio.

MQLONG serviceIntervalEvent ();

Devuelve el estado de habilitación del **suceso de intervalo de servicio** sin ninguna indicación de posibles errores.

ImqBoolean compartibilidad (MQLONG & compartibilidad);

Proporciona una copia del valor **shareability** . Devuelve TRUE si es satisfactorio.

MQLONG compartibilidad ();

Devuelve el valor de **shareability** sin ninguna indicación de posibles errores.

ImqBoolean storageClass(ImqString & clase);

Proporciona una copia de la clase de almacenamiento . Devuelve TRUE si es satisfactorio.

ImqString storageClass();

Devuelve la **clase de almacenamiento** sin ninguna indicación de posibles errores.

ImqBooleantransmissionQueueName (ImqString & nombre);

Proporciona una copia del **nombre de cola de transmisión**. Devuelve TRUE si es satisfactorio.

ImqString transmissionQueueName ();

Devuelve el **nombre de cola de transmisión** sin ninguna indicación de posibles errores.

ImqBooleantriggerControl (MQLONG & control);

Proporciona una copia del valor de **control de desencadenante** . Devuelve TRUE si es satisfactorio.

MQLONG triggerControl ();

Devuelve el valor de **control de desencadenante** sin ninguna indicación de posibles errores.

ImqBooleansetTriggerControl (const MQLONG control);

Establece el valor de **control de desencadenante** . Devuelve TRUE si es satisfactorio.

ImqBoolean triggerData (ImqString & datos);

Proporciona una copia de los **datos desencadenantes**. Devuelve TRUE si es satisfactorio.

ImqString triggerData ();

Devuelve una copia de los **datos de desencadenante** sin ninguna indicación de posibles errores.

ImqBooleansetTriggerData (const char * data);

Establece los **datos de desencadenante**. Devuelve TRUE si es satisfactorio.

ImqBoolean triggerDepth (MQLONG & profundidad);

Proporciona una copia de la **profundidad de desencadenante**. Devuelve TRUE si es satisfactorio.

MQLONG triggerDepth ();

Devuelve la **profundidad de desencadenante** sin ninguna indicación de posibles errores.

ImqBooleansetTriggerDepth (const MQLONG profundidad);

Establece la **profundidad de desencadenante**. Devuelve TRUE si es satisfactorio.

ImqBooleantriggerMessagePriority (MQLONG & prioridad);

Proporciona una copia de la **prioridad de mensaje desencadenante**. Devuelve TRUE si es satisfactorio.

MQLONG triggerMessagePriority ();

Devuelve la **prioridad de mensaje desencadenante** sin ninguna indicación de posibles errores.

ImqBooleansetTriggerMessagePriority (const MQLONG prioridad);

Establece la **prioridad de mensaje desencadenante**. Devuelve TRUE si es satisfactorio.

ImqBoolean triggerType (MQLONG & tipo);

Proporciona una copia del **tipo de desencadenante**. Devuelve TRUE si es satisfactorio.

MQLONG triggerType ();

Devuelve el **tipo de desencadenante** sin ninguna indicación de posibles errores.

ImqBooleansetTriggerType (const MQLONG tipo);

Establece el **tipo de desencadenante**. Devuelve TRUE si es satisfactorio.

ImqBoolean uso (MQLONG & uso);

Proporciona una copia del valor de **uso** . Devuelve TRUE si es satisfactorio.

MQLONG uso ();

Devuelve el valor de **uso** sin ninguna indicación de posibles errores.

Métodos de objeto (protegidos)**void setNextDistributedQueue (ImqQueue * cola = 0);**

Establece la **siguiente cola distribuida**.

Atención: Utilice esta función sólo si está seguro de que no romperá la lista de colas distribuidas.

void setPreviousDistributedQueue (ImqQueue * cola = 0);

Establece la **cola distribuida anterior**.

Atención: Utilice esta función sólo si está seguro de que no romperá la lista de colas distribuidas.

códigos de razón

- MQRC_ATTRIBUTE_LOCKED
- MQRC_CONTEXTO_OBJETO_NO_VÁLIDO
- MQRC_CONTEXT_ERROR_AL_ABRIR
- MQRC_CURSOR_NO_VÁLIDO
- MQRC_SIN_BUFFER
- MQRC_REOPEN_EXCL_ERROR_DE_ENTRADA
- MQRC_REOPEN_INQUIRE_ERROR
- MQRC_REOPEN_TEMPORARY_Q_ERROR

- (códigos de razón de MQGET)
- (códigos de razón de MQPUT)

ImqQueueManager Clase C++

Esta clase encapsula un gestor de colas (un IBM MQ objeto de tipo MQOT_Q_MGR).

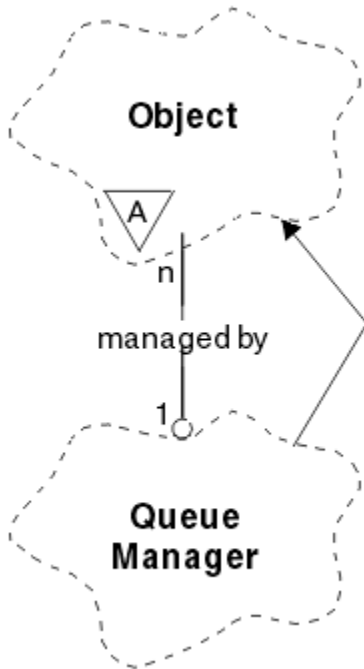


Figura 32. ImqQueueManager clase

Esta clase está relacionada con las llamadas de la clase `MQI` que se enumeran en “ImqQueueManager referencia cruzada” en la página 1825. No todos los métodos enumerados son aplicables a todas las plataformas; consulte [ALTER QMGR](#) para obtener más detalles.

- “Atributos de clase” en la página 1897
- “Atributos de objetos” en la página 1898
- “Constructores” en la página 1903
- “Destruyores” en la página 1903
- “Métodos de clase (públicos)” en la página 1903
- “Métodos de objeto (públicos)” en la página 1904
- “Métodos de objeto (protegidos)” en la página 1913
- “Datos del objeto (protegidos)” en la página 1913
- “códigos de razón” en la página 1913

Atributos de clase

comportamiento

Controla el comportamiento de la conexión y desconexión implícitas.

IMQ_EXPL_DISC_BACKOUT (0L)

Una llamada explícita al método disconnect implica una retirada. Este atributo es mutuamente excluyente con IMQ_EXPL_DISC_COMMIT.

IMQ_EXPL_DISC_COMMIT (1L)

Una llamada explícita al método disconnect implica un commit (por defecto). Este atributo es mutuamente excluyente con IMQ_EXPL_DISC_BACKOUT.

IMQ_IMPL_CONN (2L)

Se permite la conexión implícita (opción predeterminada).

IMQ_IMPL_DISC_BACKOUT (0L)

Una llamada implícita al método disconnect, que puede producirse durante la destrucción de objetos, implica una reversión. Este atributo es mutuamente excluyente con IMQ_IMPL_DISC_COMMIT.

IMQ_IMPL_DISC_COMMIT (4L)

Una llamada implícita al método disconnect, que puede producirse durante la destrucción de objetos, implica un commit (por defecto). Este atributo es mutuamente excluyente con IMQ_IMPL_DISC_BACKOUT.

En IBM MQ V7.0 y versiones posteriores, las aplicaciones C++ que utilizan una conexión implícita deben especificar IMQ_IMPL_CONN junto con cualquier otra opción proporcionada en el `setBehavior()` método en un objeto de clase `ImqQueueManager`. Si su aplicación no utiliza el `setBehavior()` método para establecer explícitamente las opciones de comportamiento, por ejemplo,

```
ImqQueueManager_object.setBehavior(IMQ_IMPL_DISC_COMMIT)
```

Este cambio no le afecta, ya que MQ_IMPL_CONN está habilitado de forma predeterminada.

Si su aplicación establece explícitamente las opciones de comportamiento, por ejemplo,

```
ImqQueueManager_object.setBehavior(IMQ_IMPL_DISC_COMMIT)
```

Debe incluir IMQ_IMPL_CONN en el `setBehavior()` método de la siguiente manera, para permitir que su aplicación complete una conexión implícita:

```
ImqQueueManager_object.setBehavior(IMQ_IMPL_CONN | IMQ_IMPL_DISC_COMMIT)
```

Atributos de objetos

Las conexiones contables tienen prioridad

Permite a las aplicaciones anular la configuración de la contabilidad de MQI y la contabilidad de colas values.This. Este atributo es de solo lectura.

Intervalo contable

Cuánto tiempo transcurre antes de que se registren los registros contables intermedios (en segundos). Este atributo es de sólo lectura.

registro de actividad

Controla la generación de los informes de actividades. Este atributo es de sólo lectura.

Adoptar nueva comprobación MCA

Los elementos que se comprueban para determinar si se debe adoptar un MCA cuando se detecta un nuevo canal entrante que tiene el mismo nombre que un MCA que ya está activo. Este atributo es de sólo lectura.

Adoptar nuevo tipo MCA

Si una instancia huérfana de un MCA de un tipo de canal concreto debe reiniciarse automáticamente cuando se detecta una nueva solicitud de canal entrante que coincide con los parámetros de verificación de adopción de nuevo MCA. Este atributo es de sólo lectura.

Tipo de autenticación

Indica el tipo de autenticación que se está realizando.

evento de autoridad

Controla los eventos de autoridad. Este atributo es de sólo lectura.

Opciones de inicio

Opciones que se aplican al método begin. El valor inicial es MQBO_NONE.

evento puente

Si se generan eventos de puent IMS. Este atributo es de sólo lectura.

Definición automática de canal

Valor de definición automática del canal. Este atributo es de sólo lectura.

evento de definición automática de canal

Valor del evento de definición automática del canal. Este atributo es de sólo lectura.

Salida de definición automática de canal

Nombre de salida de definición automática de canal. Este atributo es de sólo lectura.

suceso de canal

Si se generan eventos de canal. Este atributo es de sólo lectura.

Adaptadores del iniciador de canal

El número de subtareas del adaptador que se utilizarán para procesar IBM MQ las llamadas. Este atributo es de sólo lectura.

Control del iniciador de canal

Si el iniciador de canal debe iniciarse automáticamente cuando se inicia el gestor de colas. Este atributo es de sólo lectura.

Asignadores de tareas del iniciador de canal

El número de despachadores que se utilizarán para el iniciador del canal. Este atributo es de sólo lectura.

iniciador de canal rastrear inicio automático

Si el seguimiento del iniciador del canal debe iniciarse automáticamente o no. Este atributo es de sólo lectura.

Tamaño de tabla de rastreo del iniciador de canal

El tamaño del espacio de datos de rastreo del iniciador del canal (en MB). Este atributo es de sólo lectura.

Supervisión de canal

Controla la recopilación de los datos de supervisión para los canales. Este atributo es de sólo lectura.

referencia de canal

Una referencia a una definición de canal para su uso durante la conexión del cliente. Mientras está conectado, este atributo se puede establecer en nulo, pero no se puede cambiar a ningún otro valor. El valor inicial es nulo.

Estadísticas del canal

Controla la recopilación de los datos estadísticos para los canales. Este atributo es de sólo lectura.

juego de caracteres

Identificador de conjunto de caracteres codificados (CCSID). Este atributo es de sólo lectura.

Supervisión de clúster emisor

Controla la recopilación de datos de supervisión en línea para canales de envío de clústeres definidos automáticamente. Este atributo es de sólo lectura.

Estadística de clúster emisor

Controla la recopilación de datos estadísticos para los canales de envío de clústeres definidos automáticamente. Este atributo es de sólo lectura.

Datos de carga de trabajo de clúster

Datos de salida de la carga de trabajo del clúster. Este atributo es de sólo lectura.

salida de carga de trabajo de clúster

Nombre de salida de la carga de trabajo del clúster. Este atributo es de sólo lectura.

Longitud de la carga de trabajo de clúster

Duración de la carga de trabajo del clúster. Este atributo es de sólo lectura.

carga de trabajo del clúster mru

Valor de los canales más utilizados recientemente por la carga de trabajo del clúster. Este atributo es de sólo lectura.

Cola de uso de carga de trabajo de clúster

Valor de la cola de uso de la carga de trabajo del clúster. Este atributo es de sólo lectura.

suceso de mandato

Si se generan eventos de comando. Este atributo es de sólo lectura.

Nombre de cola de entrada de mandatos

Nombre de la cola de entrada de comandos del sistema. Este atributo es de sólo lectura.

Nivel de mandato

Nivel de comando compatible con el gestor de colas. Este atributo es de sólo lectura.

Control del servidor de mandatos

Si el servidor de comandos debe iniciarse automáticamente al iniciar el gestor de colas. Este atributo es de sólo lectura.

Opciones de conexión

Opciones que se aplican al método connect. El valor inicial es MQCNO_NONE. Los siguientes valores adicionales pueden ser posibles, dependiendo de la plataforma:

- MQCNO_STANDARD_BINDING
- MQCNO_FASTPATH_BINDING
- MQCNO_HANDLE_SHARE_NONE
- MQCNO_MANEJAR_COMPARTIR_BLOQUE
- MQCNO_HANDLE_SHARE_NO_BLOCK
- MQCNO_SERIALIZE_CONN_TAG_Q_MGR
- MQCNO_SERIALIZE_CONN_TAG_QSG
- MQCNO_RESTRICT_CONN_TAG_Q_MGR
- MQCNO_RESTRICT_CONN_TAG_QSG

ID de conexión

Un identificador único que permite a MQ identificar de forma fiable una aplicación.

estado de conexión

VERDADERO cuando está conectado al gestor de colas. Este atributo es de sólo lectura.

etiqueta de conexión

Etiqueta que se asociará a una conexión. Este atributo solo se puede configurar cuando no hay conexión. El valor inicial es nulo.

hardware de cifrado

Detalles de configuración para hardware criptográfico. Para conexiones de clientes de MQ y MQI.

nombre de la cola de mensajes no entregados

Nombre de la cola de mensajes no entregados. Este atributo es de sólo lectura.

Nombre predeterminado de la cola de transmisión

Nombre predeterminado de la cola de transmisión. Este atributo es de sólo lectura.

listas de distribución

Capacidad del gestor de colas para admitir listas de distribución.

grupo dns

El nombre del grupo al que debe unirse el listener de TCP que gestiona las transmisiones entrantes para el grupo de uso compartido de colas cuando se utiliza la compatibilidad con los servicios de nombres de dominio dinámicos de Workload Manager. Este atributo es de sólo lectura.

DNS WLM

Si el listener TCP que gestiona las transmisiones entrantes para el grupo de uso compartido de colas debe registrarse en Workload Manager para los servicios de nombres de dominio dinámicos. Este atributo es de sólo lectura.

primer registro de autenticación

El primero de uno o más objetos de la clase ImqAuthenticationRecord, sin ningún orden en particular, en el que la referencia de conexión ImqAuthenticationRecord se dirige a este objeto. Para conexiones de clientes de MQ y MQI.

primer objeto gestionado

El primero de uno o más objetos de la clase ImqObject, sin ningún orden en particular, en el que la referencia de conexión ImqObject se dirige a este objeto. El valor inicial es cero.

inhibir evento

Los controles inhiben los eventos. Este atributo es de sólo lectura.

Versión de dirección IP

¿Qué protocolo IP (IPv4 o IPv6) se debe utilizar para una conexión de canal? Este atributo es de sólo lectura.

repositorio de claves

Ubicación del archivo de base de datos de claves en el que se almacenan las claves y los certificados. Para IBM MQ MQI client conexiones.

Recuento de restablecimiento de teclas

El número de bytes sin cifrar enviados y recibidos en una conversación de TLS antes de que se renegocie la clave secreta. Este atributo solo se aplica a las conexiones de cliente que utilizan MQCONN. Véase también [recuento de restablecimientos de claves SSL](#).

Temporizador de escucha

El intervalo de tiempo (en segundos) entre los intentos de IBM MQ reiniciar el listener en caso de que se haya producido un error de « APPC » o « TCP/IP ». Este atributo es de sólo lectura.

evento local

Controla los eventos locales. Este atributo es de sólo lectura.

Suceso de registrador

Controla si se generan eventos de registro de recuperación. Este atributo es de sólo lectura.

Nombre de grupo LU

El nombre genérico de LU que debe utilizar el listener de LU 6.2 que gestiona las transmisiones entrantes para el grupo de uso compartido de colas. Este atributo es de sólo lectura.

Nombre de LU

El nombre de la LU que se utilizará para las transmisiones LU 6.2 salientes. Este atributo es de sólo lectura.

lu62 sufijo del brazo

El sufijo del miembro APPCPMxx de SYS1.PARMLIB, que designa el LUADD para este iniciador de canal. Este atributo es de sólo lectura.

lu62 canales

El número máximo de canales que pueden estar activos o de clientes que pueden estar conectados, que utilizan el protocolo de transmisión LU 6.2. Este atributo es de sólo lectura.

canales activos máximos

Número máximo de canales que pueden estar activos en cualquier momento. Este atributo es de sólo lectura.

Canales máximos

Número máximo de canales que pueden ser actuales (incluidos los canales de conexión de servidor con clientes conectados). Este atributo es de sólo lectura.

Número máximo de descriptores

Número máximo de asas. Este atributo es de sólo lectura.

longitud máxima del mensaje

Longitud máxima posible para cualquier mensaje en cualquier cola gestionada por este gestor de colas. Este atributo es de sólo lectura.

Prioridad máxima

Máxima prioridad del mensaje. Este atributo es de sólo lectura.

Máx. mensajes no confirmados

Número máximo de mensajes no confirmados dentro de una unidad o trabajo. Este atributo es de sólo lectura.

Contabilidad de MQI

Controla la recopilación de la información contable para los datos de la Interfaz de Colas de Mensajes (MQI). Este atributo es de sólo lectura.

Estadísticas MQI

Controla la recopilación de la información sobre la supervisión de estadísticas para el gestor de colas. Este atributo es de sólo lectura.

máximo del puerto de salida

El extremo superior del rango de números de puerto que se utilizará al vincular canales salientes. Este atributo es de sólo lectura.

puerto de salida mínimo

El extremo inferior del rango de números de puerto que se utilizará al vincular canales salientes. Este atributo es de sólo lectura.

contraseña

Contraseña asociada al ID de usuario

suceso de rendimiento

Controla los eventos de rendimiento. Este atributo es de sólo lectura.

platform

Plataforma en la que reside el gestor de colas. Este atributo es de sólo lectura.

Contabilidad de la cola

Controla la recopilación de la información contable para las colas. Este atributo es de sólo lectura.

Supervisión de la cola

Controla la recopilación de los datos de supervisión para las colas. Este atributo es de sólo lectura.

estadísticas de cola

Controla la recopilación de los datos estadísticos para las colas. Este atributo es de sólo lectura.

Tiempo de espera de recepción

¿Cuánto tiempo, aproximadamente, esperará un canal de mensajes de « TCP/IP » para recibir datos, incluidos los latidos, de su socio, antes de volver al estado inactivo? Este atributo es de sólo lectura.

tiempo mínimo de espera de recepción

El tiempo mínimo que un canal « TCP/IP » esperará para recibir datos, incluidos los pulsos de control, de su socio, antes de volver al estado inactivo. Este atributo es de sólo lectura.

Tipo de tiempo de espera de recepción

Un calificador aplicado para recibir un tiempo de espera. Este atributo es de sólo lectura.

evento remoto

Controla eventos remotos. Este atributo es de sólo lectura.

Nombre de depósito

Nombre de repositorio. Este atributo es de sólo lectura.

Lista nombres repositorio

Nombre de la lista de nombres del repositorio. Este atributo es de sólo lectura.

nombre del gestor de colas compartidas

Si los MQOPEN de una cola compartida en la que el gestor de colas (ObjectQMGrName) es otro gestor de colas del grupo de intercambio de colas deben resolverse en una apertura de la cola compartida en el gestor de colas local. Este atributo es de sólo lectura.

evento SSL

Si se generan eventos « SSL ». Este atributo es de sólo lectura.

SeFIPS requiere SSL

Si solo deben FIPS utilizarse algoritmos certificados cuando la criptografía se ejecuta en IBM MQ software. Este atributo es de sólo lectura.

Cuenta restablecimiento clave SSL

El número de bytes sin cifrar enviados y recibidos dentro de una conversación SSL antes de que se renegocie la clave secreta. Este atributo es de sólo lectura.

evento de arranque y parada

Controla los eventos de inicio y parada. Este atributo es de sólo lectura.

Intervalo de estadísticas

Con qué frecuencia se escriben los datos de supervisión de estadísticas en la cola de supervisión. Este atributo es de sólo lectura.

Disponibilidad de punto de sincronismo

Disponibilidad de participación en Syncpoint. Este atributo es de sólo lectura.

Nota: Las unidades de trabajo globales coordinadas por el gestor de colas no son compatibles con la IBM i plataforma.  Puede programar una unidad de trabajo, coordinada externamente por IBM i, utilizando las llamadas al sistema nativas `_Rcommit` y `_Rback`. Inicie este tipo de unidad de trabajo ejecutando la IBM MQ aplicación bajo control de compromiso a nivel de trabajo utilizando el comando `STRCMTCTL`. Consulte [Interfaces con el administrador de puntos IBM i de sincronización externos](#) para obtener más detalles. La IBM i plataforma admite las operaciones «backout» y «commit» para unidades de trabajo locales coordinadas por un gestor de colas.

canales TCP

El número máximo de canales activos o de clientes conectados que utilizan el protocolo de transmisión « TCP/IP ». Este atributo es de sólo lectura.

Mantener activo de TCP

Si se va a utilizar la función `KEEPALIVE` de TCP para comprobar que el otro extremo de la conexión sigue estando disponible. Este atributo es de sólo lectura.

Nombre TCP

El nombre del sistema « TCP/IP » único o predeterminado que se va a utilizar, en función del valor del tipo de pila TCP. Este atributo es de sólo lectura.

Tipo de pila TCP

Si el iniciador del canal solo puede utilizar el espacio de direcciones TCP/IP especificado en el nombre TCP o si puede vincularse a cualquier dirección TCP/IP seleccionada. Este atributo es de sólo lectura.

Registro de la ruta de rastreo

Controla el registro de la información de seguimiento de la ruta. Este atributo es de sólo lectura.

Activar intervalo

Intervalo de activación. Este atributo es de sólo lectura.

ID de usuario

En AIX and Linux las plataformas, el ID de usuario real de la aplicación. En Windows las plataformas, el ID de usuario de la aplicación.

Constructores**`ImqQueueManager( );`**

El constructor predeterminado.

`ImqQueueManager( const ImqQueueManager &manager );`

El constructor de copia. El estado de la conexión será `FALSE`.

`ImqQueueManager( const char * nombre );`

Establece el nombre del controlador de entrada de texto (`ImqObject`) como *nombre*.

Destruyores

Cuando se destruye un objeto `ImqQueueManager``, se desconecta automáticamente.

Métodos de clase (públicos)**`comportamiento estático MQLONG( );`**

Devuelve el comportamiento.

`void setBehavior( const MQLONG behavior = 0 );`

Establece el comportamiento.

Métodos de objeto (públicos)

operador vacío = (const ImqQueueManager &mgr);

Se desconecta si es necesario y copia los datos de la instancia desde *mgr*. El estado de la conexión es FALSO.

ImqBoolean accountingConnOverride (MQLONG y statint);

Proporciona una copia del valor de anulación de las conexiones contables. Devuelve TRUE si se realiza correctamente.

MQLONG accountingConnOverride ();

Devuelve el valor de anulación de las conexiones contables sin indicar posibles errores.

ImqBoolean accountingInterval (MQLONG y statint);

Proporciona una copia del valor del intervalo contable. Devuelve TRUE si se realiza correctamente.

MQLONG accountingInterval ();

Devuelve el valor del intervalo contable sin indicar posibles errores.

ImqBoolean activityRecording (MQLONG y rec);

Proporciona una copia del valor del registro de actividad. Devuelve TRUE si se realiza correctamente.

MQLONG activityRecording ();

Devuelve el valor registrado de la actividad sin indicar posibles errores.

ImqBoolean adoptNewMCACheck (MQLONG y comprobar);

Proporciona una copia del nuevo valor de verificación MCA adoptado. Devuelve TRUE si se realiza correctamente.

MQLONG adoptNewMCACheck ();

Devuelve el nuevo valor de comprobación MCA adoptado sin indicar posibles errores.

ImqBoolean adoptNewMCAType (MQLONG y tipo);

Proporciona una copia del nuevo tipo MCA adoptado. Devuelve TRUE si se realiza correctamente.

MQLONG adoptNewMCAType ();

Devuelve el nuevo tipo MCA adoptado sin indicar posibles errores.

QLONG authenticationType () const;

Devuelve el tipo de autenticación.

void setAuthenticationType (const MQLONG tipo = MQCSP_AUTH_NONE);

Establece el tipo de autenticación.

ImqBoolean authorityEvent(MQLONG & evento);

Proporciona una copia del estado de habilitación del evento de autoridad. Devuelve TRUE si se realiza correctamente.

MQLONG authorityEvent();

Devuelve el estado de habilitación del evento de autoridad sin indicar posibles errores.

ImqBoolean retroceder();

Deshace los cambios no confirmados. Devuelve TRUE si se realiza correctamente.

ImqBoolean begin();

Comienza una unidad de trabajo. Las opciones de inicio afectan al comportamiento de este método. Devuelve TRUE si se realiza correctamente, pero también devuelve TRUE aunque la llamada MQBEGIN subyacente devuelva MQRC_NO_EXTERNAL_PARTICIPANTS o MQRC_PARTICIPANT_NOT_AVAILABLE (ambas asociadas a MQCC_WARNING).

MQLONG beginOptions() const;

Devuelve las opciones iniciales.

void setBeginOptions(const MQLONG options = MQBO_NONE);

Establece las opciones de inicio.

ImqBoolean bridgeEvent (MQLONG y evento);

Proporciona una copia del valor del evento bridge. Devuelve TRUE si se realiza correctamente.

MQLONG bridgeEvent ();

Devuelve el valor del evento bridge sin indicar posibles errores.

ImqBoolean channelAutoDefinition(MQLONG & valor);

Proporciona una copia del valor de definición automática del canal. Devuelve TRUE si se realiza correctamente.

MQLONG channelAutoDefinition();

Devuelve el valor de definición automática del canal sin indicar posibles errores.

ImqBoolean channelAutoDefinitionEvent(MQLONG & valor);

Proporciona una copia del valor del evento de definición automática del canal. Devuelve TRUE si se realiza correctamente.

MQLONG channelAutoDefinitionEvent();

Devuelve el valor del evento de definición automática del canal sin indicar posibles errores.

ImqBoolean channelAutoDefinitionExit(ImqString & nombre);

Proporciona una copia del nombre de salida de la definición automática del canal. Devuelve TRUE si se realiza correctamente.

ImqString channelAutoDefinitionExit();

Devuelve el nombre de salida de la definición automática del canal sin indicar posibles errores.

ImqBoolean channelEvent (MQLONG y evento);

Proporciona una copia del valor del evento del canal. Devuelve TRUE si se realiza correctamente.

MQLONG channelEvent();

Devuelve el valor del evento del canal sin indicar posibles errores.

MQLONG channelInitiatorAdapters ();

Devuelve el valor de los adaptadores iniciadores del canal sin indicar posibles errores.

ImqBoolean channelInitiatorAdapters (MQLONG y adaptadores);

Proporciona una copia del valor de los adaptadores iniciadores del canal. Devuelve TRUE si se realiza correctamente.

MQLONG channelInitiatorControl ();

Devuelve el valor de inicio del iniciador del canal sin indicar posibles errores.

ImqBoolean channelInitiatorControl (MQLONG e init);

Proporciona una copia del valor de inicio del control del iniciador del canal. Devuelve TRUE si se realiza correctamente.

MQLONG channelInitiatorDispatchers ();

Devuelve el valor de los distribuidores iniciadores del canal sin indicar posibles errores.

ImqBoolean channelInitiatorDispatchers (MQLONG y despachadores);

Proporciona una copia del valor de los despachadores del iniciador del canal. Devuelve TRUE si se realiza correctamente.

MQLONG channelInitiatorTraceAutoStart ();

Devuelve el valor de inicio automático del rastreo del iniciador del canal sin indicar posibles errores.

ImqBoolean channelInitiatorTraceAutoStart (MQLONG y automático);

Proporciona una copia del valor de inicio automático del rastreo del iniciador del canal. Devuelve TRUE si se realiza correctamente.

MQLONG channelInitiatorTraceTableSize ();

Devuelve el valor del tamaño de la tabla de seguimiento del iniciador del canal sin indicar posibles errores.

ImqBoolean channelInitiatorTraceTableSize (MQLONG y tamaño);

Proporciona una copia del valor del tamaño de la tabla de seguimiento del iniciador del canal. Devuelve TRUE si se realiza correctamente.

ImqBoolean channelMonitoring (MQLONG y monchl);

Proporciona una copia del valor de supervisión del canal. Devuelve TRUE si se realiza correctamente.

MQLONG channelMonitoring ();

Devuelve el valor de supervisión del canal sin indicar posibles errores.

ImqBoolean channelReference(ImqChannel * & canal p);

Proporciona una copia de la referencia del canal. Si la referencia del canal no es válida, establece *pchannel* en nulo. Este método devuelve TRUE si se ejecuta correctamente.

ImqChannel * channelReference();

Devuelve la referencia del canal sin indicar posibles errores.

ImqBoolean setChannelReference(ImqChannel & canal);

Establece la referencia del canal. Este método devuelve TRUE si se ejecuta correctamente.

ImqBoolean setChannelReference(ImqChannel * canal = 0);

Establece o restablece la referencia del canal. Este método devuelve TRUE si se ejecuta correctamente.

ImqBoolean channelStatistics (MQLONG y statchl);

Proporciona una copia del valor de las estadísticas del canal. Devuelve TRUE si se realiza correctamente.

MQLONG channelStatistics ();

Devuelve el valor de las estadísticas del canal sin indicar posibles errores.

ImqBoolean characterSet(MQLONG y ccsid);

Proporciona una copia del conjunto de caracteres. Devuelve TRUE si se realiza correctamente.

MQLONG characterSet();

Devuelve una copia del conjunto de caracteres, sin indicar posibles errores.

MQLONG clientSslKeyResetCount () const;

Devuelve el valor del recuento de restablecimiento de la clave SSL utilizado en las conexiones de cliente.

void setClientSslKeyResetCount(const MQLONG count);

Establece el recuento de restablecimiento de la clave SSL utilizado en las conexiones de cliente.

ImqBoolean clusterSenderMonitoring (MQLONG y monacIs);

Proporciona una copia del valor predeterminado de supervisión del remitente del clúster. Devuelve TRUE si se realiza correctamente.

MQLONG clusterSenderMonitoring ();

Devuelve el valor predeterminado de supervisión del remitente del clúster sin indicar posibles errores.

ImqBoolean clusterSenderStatistics (MQLONG y statacl);

Proporciona una copia del valor de las estadísticas del remitente del clúster. Devuelve TRUE si se realiza correctamente.

MQLONG clusterSenderStatistics ();

Devuelve el valor de las estadísticas del remitente del clúster sin indicar posibles errores.

ImqBoolean clusterWorkloadData(ImqString & datos);

Proporciona una copia de los datos de salida de la carga de trabajo del clúster. Devuelve TRUE si se realiza correctamente.

ImqString clusterWorkloadData();

Devuelve los datos de salida de la carga de trabajo del clúster sin indicar posibles errores.

ImqBoolean clusterWorkloadExit(ImqString & nombre);

Proporciona una copia del nombre de salida de la carga de trabajo del clúster. Devuelve TRUE si se realiza correctamente.

ImqString clusterWorkloadExit();

Devuelve el nombre de salida de la carga de trabajo del clúster sin indicar posibles errores.

ImqBoolean clusterWorkloadLength(MQLONG y longitud);

Proporciona una copia de la duración de la carga de trabajo del clúster. Devuelve TRUE si se realiza correctamente.

MQLONG clusterWorkloadLength();

Devuelve la duración de la carga de trabajo del clúster sin indicar posibles errores.

ImqBoolean clusterWorkLoadMRU (MQLONG y mru);

Proporciona una copia del valor de los canales más utilizados recientemente por la carga de trabajo del clúster. Devuelve TRUE si se realiza correctamente.

MQLONG clusterWorkLoadMRU ();

Devuelve el valor de los canales utilizados más recientemente por la carga de trabajo del clúster sin indicar posibles errores.

ImqBoolean clusterWorkLoadUseQ (MQLONG y useq);

Proporciona una copia del valor de la cola de uso de la carga de trabajo del clúster. Devuelve TRUE si se realiza correctamente.

MQLONG clusterWorkLoadUseQ ();

Devuelve el valor de la cola de uso de la carga de trabajo del clúster sin indicar posibles errores.

ImqBoolean commandEvent (MQLONG y evento);

Proporciona una copia del valor del evento de comando. Devuelve TRUE si se realiza correctamente.

MQLONG commandEvent ();

Devuelve el valor del evento de comando sin indicar posibles errores.

ImqBoolean commandInputQueueName(ImqString & nombre);

Proporciona una copia del nombre de la cola de entrada de comandos. Devuelve TRUE si se realiza correctamente.

ImqString commandInputQueueName();

Devuelve el nombre de la cola de entrada de comandos sin indicar posibles errores.

ImqBoolean commandLevel(MQLONG y nivel);

Proporciona una copia del nivel de comando. Devuelve TRUE si se realiza correctamente.

MQLONG commandLevel();

Devuelve el nivel de comando sin indicar posibles errores.

MQLONG commandServerControl ();

Devuelve el valor de inicio del servidor de comandos sin indicar posibles errores.

ImqBoolean commandServerControl (MQLONG y servidor);

Proporciona una copia del valor de inicio del control del servidor de comandos. Devuelve TRUE si se realiza correctamente.

ImqBoolean commit();

Confirma los cambios no confirmados. Devuelve TRUE si se realiza correctamente.

ImqBoolean conectar();

Se conecta al gestor de colas con el nombre ImqObject especificado, siendo el gestor de colas local el valor predeterminado. Si desea conectarse a un gestor de colas específico, utilice el método ImqObjectsetName antes de la conexión. Si hay una referencia de canal, se utiliza para pasar información sobre la definición del canal a MQCONN en un MQCD. El ChannelType en el MQCD se establece en MQCHT_CLNTCONN. La información de referencia del canal, que solo tiene sentido para las conexiones de cliente, se ignora para las conexiones de servidor. Las opciones de conexión afectan al comportamiento de este método. Este método establece el estado de la conexión en TRUE si se realiza correctamente. Devuelve el nuevo estado de la conexión.

Si existe un primer registro de autenticación, la cadena de registros de autenticación se utiliza para autenticar certificados digitales para canales de cliente seguros.

Puede conectar más de un objeto ImqQueueManager al mismo gestor de colas. Todos utilizan el mismo identificador de conexión MQHCONN y comparten la funcionalidad UOW para la conexión asociada al subproceso. El primer ImqQueueManager que se conecta obtiene el identificador MQHCONN. El último « ImqQueueManager » que se desconecta realiza el MQDISC.

Para un programa multihilo, se recomienda utilizar un objeto ImqQueueManager independiente para cada hilo.

ImqBinary connectionId () const;

Devuelve el ID de conexión.

ImqBinary connectionTag () const;

Devuelve la etiqueta de conexión.

ImqBoolean setConnectionTag (const MQBYTE128 tag = 0);

Establece la etiqueta de conexión. Si *la etiqueta* es cero, borra la etiqueta de conexión. Este método devuelve TRUE si se ejecuta correctamente.

ImqBoolean setConnectionTag (const ImqBinary &tag);

Establece la etiqueta de conexión. La longitud de datos de *la etiqueta* debe ser cero (para borrar la etiqueta de conexión) o MQ_CONN_TAG_LENGTH. Este método devuelve TRUE si se ejecuta correctamente.

MQLONG connectOptions() const;

Devuelve las opciones de conexión.

void setConnectOptions(const MQLONG options = MQCNO_NONE);

Establece las opciones de conexión.

ImqBoolean connectionStatus() const;

Devuelve el estado de la conexión.

ImqString cryptographicHardware ();

Devuelve el hardware criptográfico.

ImqBoolean setCryptographicHardware (const char * hardware = 0);

Configura el hardware criptográfico. Este método devuelve TRUE si se ejecuta correctamente.

ImqBoolean deadLetterQueueName(ImqString & nombre);

Proporciona una copia del nombre de la cola de mensajes no entregados. Devuelve TRUE si se realiza correctamente.

ImqString deadLetterQueueName();

Devuelve una copia del nombre de la cola de mensajes no entregados, sin indicar posibles errores.

ImqBoolean defaultTransmissionQueueName(ImqString & nombre);

Proporciona una copia del nombre predeterminado de la cola de transmisión. Devuelve TRUE si se realiza correctamente.

ImqString defaultTransmissionQueueName();

Devuelve el nombre predeterminado de la cola de transmisión sin indicar posibles errores.

ImqBoolean desconectar();

Se desconecta del gestor de colas y establece el estado de la conexión en FALSE. Cierra todos los objetos ImqProcess y ImqQueue asociados con este objeto y corta su referencia de conexión antes de la desconexión. Si hay más de un objeto ImqQueueManager conectado al mismo gestor de colas, solo el último en desconectarse realiza una desconexión física; los demás realizan una desconexión lógica. Los cambios no confirmados solo se confirman al desconectarse físicamente.

Este método devuelve TRUE si se ejecuta correctamente. Si se invoca cuando no existe ninguna conexión, el código de retorno también es verdadero.

ImqBoolean distributionLists(MQLONG y soporte);

Proporciona una copia del valor de las listas de distribución. Devuelve TRUE si se realiza correctamente.

MQLONG distributionLists();

Devuelve el valor de las listas de distribución sin indicar posibles errores.

ImqBoolean dnsGroup (ImqString y grupo);

Proporciona una copia del nombre del grupo DNS. Devuelve TRUE si se realiza correctamente.

ImqString dnsGroup ();

Devuelve el nombre del grupo DNS sin indicar posibles errores.

ImqBoolean dnsWlm (MQLONG y wlm);

Proporciona una copia del valor DNS WLM. Devuelve TRUE si se realiza correctamente.

MQLONG dnsWlm ();

Devuelve el valor DNS WLM sin indicar posibles errores.

ImqAuthenticationRecord * firstAuthenticationRecord () const;

Devuelve el primer registro de autenticación.

void setFirstAuthenticationRecord (const ImqAuthenticationRecord * air = 0);

Establece el primer registro de autenticación.

ImqObject * firstManagedObject() const;

Devuelve el primer objeto gestionado.

ImqBoolean inhibitEvent(MQLONG & evento);

Proporciona una copia del estado de habilitación del evento de inhibición. Devuelve TRUE si se realiza correctamente.

MQLONG inhibitEvent();

Devuelve el estado de habilitación del evento de inhibición sin indicar posibles errores.

ImqBoolean ipAddressVersion (MQLONG y versión);

Proporciona una copia del valor de la versión de la dirección IP. Devuelve TRUE si se realiza correctamente.

MQLONG ipAddressVersion ();

Devuelve el valor de la versión de la dirección IP sin indicar posibles errores.

ImqBoolean keepAlive (MQLONG y keepalive);

Proporciona una copia del valor de mantenimiento. Devuelve TRUE si se realiza correctamente.

MQLONG keepAlive ();

Devuelve el valor de mantenimiento sin indicar posibles errores.

ImqString keyRepository ();

Devuelve el repositorio de claves.

ImqBoolean setKeyRepository (const char * repositorio = 0);

Establece el repositorio de claves. Devuelve TRUE si se realiza correctamente.

ImqBoolean listenerTimer (MQLONG y temporizador);

Proporciona una copia del valor del temporizador del oyente. Devuelve TRUE si se realiza correctamente.

MQLONG listenerTimer ();

Devuelve el valor del temporizador del oyente sin indicar posibles errores.

ImqBoolean localEvent(MQLONG & evento);

Proporciona una copia del estado de habilitación del evento local. Devuelve TRUE si se realiza correctamente.

MQLONG localEvent();

Devuelve el estado de habilitación del evento local sin indicar posibles errores.

ImqBoolean loggerEvent (MQLONG y recuento);

Proporciona una copia del valor del evento del registrador. Devuelve TRUE si se realiza correctamente.

MQLONG loggerEvent ();

Devuelve el valor del evento del registrador sin indicar posibles errores.

ImqBoolean luGroupName (ImqString &name);.

Proporciona una copia del nombre del grupo LU. Devuelve TRUE si se realiza correctamente

ImqString luGroupName ();

Devuelve el nombre del grupo LU sin indicar posibles errores.

ImqBoolean lu62ARMSuffix (ImqString e y sufijo);

Proporciona una copia del sufijo ARM « LU62 ». Devuelve TRUE si se realiza correctamente.

ImqString lu62ARMSuffix ();

Devuelve el sufijo ARM LU62 sin indicar posibles errores

ImqBoolean luName (ImqString &name);

Proporciona una copia del nombre de la unidad lógica. Devuelve TRUE si se realiza correctamente.

ImqString luName ();

Devuelve el nombre LU sin indicar posibles errores.

ImqBoolean maximumActiveChannels (MQLONG y canales);

Proporciona una copia del valor máximo de canales activos. Devuelve TRUE si se realiza correctamente.

MQLONG maximumActiveChannels ();

Devuelve el valor máximo de canales activos sin indicar posibles errores.

ImqBoolean maximumCurrentChannels (MQLONG y canales);

Proporciona una copia del valor máximo actual de canales. Devuelve TRUE si se realiza correctamente.

MQLONG maximumCurrentChannels ();

Devuelve el valor máximo de canales actuales sin indicar posibles errores.

ImqBoolean maximumHandles(MQLONG & número);

Proporciona una copia de los identificadores máximos. Devuelve TRUE si se realiza correctamente.

MQLONG maximumHandles();

Devuelve el número máximo de identificadores sin indicar posibles errores.

ImqBoolean maximumLu62Channels (MQLONG y canales);

Proporciona una copia del valor máximo de canales LU62. Devuelve TRUE si se realiza correctamente.

MQLONG maximumLu62Channels ();

Devuelve el valor máximo de canales LU62 sin indicar posibles errores

ImqBoolean maximumMessageLength(MQLONG y longitud);

Proporciona una copia de la longitud máxima del mensaje. Devuelve TRUE si se realiza correctamente.

MQLONG maximumMessageLength();

Devuelve la longitud máxima del mensaje sin indicar posibles errores.

ImqBoolean maximumPriority(MQLONG y prioridad);

Proporciona una copia de la prioridad máxima. Devuelve TRUE si se realiza correctamente.

MQLONG maximumPriority();

Devuelve una copia de la prioridad máxima, sin indicar posibles errores.

ImqBoolean maximumTcpChannels (MQLONG y canales);

Proporciona una copia del valor máximo de canales TCP. Devuelve TRUE si se realiza correctamente.

MQLONG maximumTcpChannels ();

Devuelve el valor máximo de canales e TCP es sin indicar posibles errores.

ImqBoolean maximumUncommittedMessages(MQLONG & número);

Proporciona una copia de los mensajes máximos no comprometidos. Devuelve TRUE si se realiza correctamente.

MQLONG maximumUncommittedMessages();

Devuelve el número máximo de mensajes no confirmados sin indicar posibles errores.

ImqBoolean mqiAccounting (MQLONG y statint);

Proporciona una copia del valor contable de « MQI ». Devuelve TRUE si se realiza correctamente.

MQLONG mqiAccounting ();

Devuelve el valor contable de « MQI » sin indicar posibles errores.

ImqBoolean mqiStatistics (MQLONG y statmqi);

Proporciona una copia del valor de la estadística « MQI ». Devuelve TRUE si se realiza correctamente.

MQLONG mqiStatistics ();

Devuelve el valor de la estadística « MQI » sin indicar posibles errores.

ImqBoolean outboundPortMax (MQLONG y max);

Proporciona una copia del valor máximo del puerto de salida. Devuelve TRUE si se realiza correctamente.

MQLONG outboundPortMax ();

Devuelve el valor máximo del puerto de salida sin indicar posibles errores.

ImqBoolean outboundPortMin (MQLONG y min);

Proporciona una copia del valor mínimo del puerto de salida. Devuelve TRUE si se realiza correctamente.

MQLONG outboundPortMin ();

Devuelve el valor mínimo del puerto de salida sin indicar posibles errores.

ImqBinary contraseña () const;

Devuelve la contraseña utilizada en las conexiones del cliente.

ImqBoolean setPassword (const ImqString &password);

Establece la contraseña utilizada en las conexiones de los clientes.

ImqBoolean setPassword (const char * = 0 contraseña);

Establece la contraseña utilizada en las conexiones de los clientes.

ImqBoolean setPassword (const ImqBinary &password);

Establece la contraseña utilizada en las conexiones de los clientes.

ImqBoolean performanceEvent(MQLONG & evento);

Proporciona una copia del estado de habilitación del evento de rendimiento. Devuelve TRUE si se realiza correctamente.

MQLONG performanceEvent();

Devuelve el estado de habilitación del evento de rendimiento sin indicar posibles errores.

ImqBoolean plataforma(MQLONG & plataforma);

Proporciona una copia de la plataforma. Devuelve TRUE si se realiza correctamente.

Plataforma MQLONG();

Devuelve la plataforma sin indicar posibles errores.

ImqBoolean queueAccounting (MQLONG y acctq);

Proporciona una copia del valor contable de la cola. Devuelve TRUE si se realiza correctamente.

MQLONG queueAccounting ();

Devuelve el valor contable de la cola sin indicar posibles errores.

ImqBoolean queueMonitoring (MQLONG y monq);

Proporciona una copia del valor de supervisión de la cola. Devuelve TRUE si se realiza correctamente.

MQLONG queueMonitoring ();

Devuelve el valor de supervisión de la cola sin indicar posibles errores.

ImqBoolean queueStatistics (MQLONG y statq);

Proporciona una copia del valor de las estadísticas de la cola. Devuelve TRUE si se realiza correctamente.

MQLONG queueStatistics ();

Devuelve el valor de las estadísticas de la cola sin indicar posibles errores.

ImqBoolean receiveTimeout (MQLONG y tiempo de espera);

Proporciona una copia del valor del tiempo de espera de recepción. Devuelve TRUE si se realiza correctamente.

MQLONG receiveTimeout ();

Devuelve el valor del tiempo de espera de recepción sin indicar posibles errores.

ImqBoolean receiveTimeoutMin (MQLONG y min);

Proporciona una copia del valor mínimo del tiempo de espera de recepción. Devuelve TRUE si se realiza correctamente.

MQLONG receiveTimeoutMin ();

Devuelve el valor mínimo del tiempo de espera de recepción sin indicar posibles errores.

ImqBoolean receiveTimeoutType (MQLONG y tipo);

Proporciona una copia del tipo de tiempo de espera de recepción. Devuelve TRUE si se realiza correctamente.

MQLONG receiveTimeoutType ();

Devuelve el tipo de tiempo de espera de recepción sin indicar posibles errores.

ImqBoolean remoteEvent(MQLONG & evento);

Proporciona una copia del estado de habilitación del evento remoto. Devuelve TRUE si se realiza correctamente.

MQLONG remoteEvent();

Devuelve el estado de habilitación del evento remoto sin indicar posibles errores.

ImqBoolean repositoryName(ImqString & nombre);

Proporciona una copia del nombre del repositorio. Devuelve TRUE si se realiza correctamente.

ImqString repositoryName();

Devuelve el nombre del repositorio sin indicar posibles errores.

ImqBoolean repositoryNameListName(ImqString & nombre);

Proporciona una copia del nombre de la lista de nombres del repositorio. Devuelve TRUE si se realiza correctamente.

ImqString repositoryNameListName();

Devuelve una copia del nombre de la lista de nombres del repositorio sin indicar posibles errores.

ImqBoolean sharedQueueQueueManagerName (MQLONG y nombre);

Proporciona una copia del valor del nombre del gestor de colas compartidas. Devuelve TRUE si se realiza correctamente.

MQLONG sharedQueueQueueManagerName ();

Devuelve el valor del nombre del gestor de colas compartidas sin indicar posibles errores.

ImqBoolean sslEvent (MQLONG y evento);

Proporciona una copia del valor del evento « SSL ». Devuelve TRUE si se realiza correctamente.

MQLONG sslEvent ();

Devuelve el valor del evento SSL sin indicar posibles errores.

ImqBoolean sslFips (MQLONG y sslfips);

Proporciona una copia del valor **SSLFIPS** . Devuelve TRUE si se realiza correctamente.

MQLONG sslFips ();

Devuelve el valor FIPSSSL sin indicar posibles errores.

ImqBoolean sslKeyResetCount (MQLONG y recuento);

Proporciona una copia del valor del recuento de restablecimiento de la clave « SSL ». Devuelve TRUE si se realiza correctamente.

MQLONG sslKeyResetCount ();

Devuelve el valor del recuento de restablecimiento de la clave SSL sin indicar posibles errores.

ImqBoolean startStopEvent(MQLONG & evento);

Proporciona una copia del estado de habilitación del evento de inicio-parada. Devuelve TRUE si se realiza correctamente.

MQLONG startStopEvent();

Devuelve el estado de habilitación del evento de inicio-parada sin indicar posibles errores.

ImqBoolean statisticsInterval (MQLONG y statint);

Proporciona una copia del valor del intervalo estadístico. Devuelve TRUE si se realiza correctamente.

MQLONG statisticsInterval ();

Devuelve el valor del intervalo estadístico sin indicar posibles errores.

ImqBoolean syncPointAvailability(MQLONG y sincronización);

Proporciona una copia del valor de disponibilidad del punto de sincronización. Devuelve TRUE si se realiza correctamente.

MQLONG syncPointAvailability();

Devuelve una copia del valor de disponibilidad del punto de sincronización, sin indicar posibles errores.

ImqBoolean tcpName (ImqString &name);

Proporciona una copia del nombre del sistema « TCP ». Devuelve TRUE si se realiza correctamente.

ImqString tcpName ();

Devuelve el nombre del sistema TCP sin indicar posibles errores.

ImqBoolean tcpStackType (MQLONG y tipo);

Proporciona una copia del tipo de pila « TCP ». Devuelve TRUE si se realiza correctamente.

MQLONG tcpStackType ();

Devuelve el tipo de pila « TCP » sin indicar posibles errores.

ImqBoolean traceRouteRecording (MQLONG y routerec);

Proporciona una copia del valor de registro de la ruta de seguimiento. Devuelve TRUE si se realiza correctamente.

MQLONG traceRouteRecording ();

Devuelve el valor de registro de la ruta de seguimiento sin indicar posibles errores.

ImqBoolean triggerInterval(MQLONG & intervalo);

Proporciona una copia del intervalo de activación. Devuelve TRUE si se realiza correctamente.

MQLONG triggerInterval();

Devuelve el intervalo de activación sin indicar posibles errores.

ImqBinary userId () const;

Devuelve el ID de usuario utilizado en las conexiones del cliente.

ImqBoolean setUserId (const ImqString & id);

Establece el ID de usuario utilizado en las conexiones de cliente.

ImqBoolean setUserId (const char * = 0 id);

Establece el ID de usuario utilizado en las conexiones de cliente.

ImqBoolean setUserId (const ImqBinary & id);

Establece el ID de usuario utilizado en las conexiones de cliente.

Métodos de objeto (protegidos)**void setFirstManagedObject (const ImqObject * objeto = 0);**

Establece el primer objeto gestionado.

Datos del objeto (protegidos)**MQHCONN ohconn**

El IBM MQ identificador de conexión (solo tiene sentido cuando el estado de la conexión es TRUE).

códigos de razón

- MQRC_ATTRIBUTO_BLOQUEADO
- MQRC_ERROR_DE_ENTORNO
- MQRC_FUNCTION_NOT_SUPPORTED
- ERROR DE REFERENCIA MQRC
- (códigos de motivo para MQBACK)
- (códigos de motivo para MQBEGIN)
- (códigos de motivo para MQCMIT)
- (códigos de motivo para MQCONN)
- (códigos de motivo para MQDISC)
- (códigos de motivo para MQCONN)

ImqReferenceHeader Clase C

Esta clase encapsula características de la estructura de datos MQRMH.

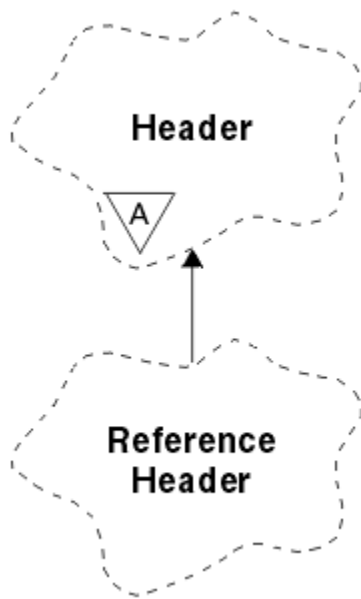


Figura 33. *ImqReferenceHeader* clase

Esta clase está relacionada con las llamadas de la clase `MQI` que se enumeran en [“ImqReferenceHeader referencia cruzada” en la página 1830](#).

- [“Atributos de objetos” en la página 1914](#)
- [“Constructores” en la página 1915](#)
- [“Métodos ImqItem sobrecargados” en la página 1915](#)
- [“Métodos de objeto \(public\)” en la página 1915](#)
- [“Datos de objeto \(protegidos\)” en la página 1916](#)
- [“códigos de razón” en la página 1916](#)

Atributos de objetos

entorno de destino

Entorno para el destino. El valor inicial es una serie nula.

nombre de destino

Nombre del destino de datos. El valor inicial es una serie nula.

ID de instancia

Identificador de instancia. Un valor binario (MQBYTE24) de longitud MQ_OBJECT_INSTANCE_ID_LENGTH. El valor inicial es MQOII_NONE.

longitud lógica

Longitud lógica, o prevista, de los datos de mensaje que siguen a esta cabecera. El valor inicial es cero.

desplazamiento lógico

Desplazamiento lógico para los datos de mensaje siguientes, que se interpretarán en el contexto de los datos en su conjunto, en el destino final. El valor inicial es cero.

desplazamiento lógico 2

Extensión de orden superior al desplazamiento lógico. El valor inicial es cero.

Tipo de referencia

Tipo de referencia. El valor inicial es una serie nula.

Entorno de origen

Entorno para el origen. El valor inicial es una serie nula.

nombre de origen

Nombre del origen de datos. El valor inicial es una serie nula.

Constructores**ImqReferenceHeader();**

El constructor predeterminado.

ImqReferenceHeader(const ImqReferenceHeader &header);

El constructor de copia.

Métodos ImqItem sobrecargados**virtual ImqBoolean copyOut (ImqMessage & msg);**

Inserta una estructura de datos MQRMH en el almacenamiento intermedio de mensajes al principio, moviendo los datos de mensaje existentes más adelante y establece el formato *msg* en MQFMT_REF_MSG_HEADER.

Consulte la descripción del método de clase ImqHeader en [“Clase C++ ImqHeader” en la página 1859](#) para obtener más detalles.

virtual ImqBoolean pasteIn (ImqMessage & msg);

Lee una estructura de datos MQRMH del almacenamiento intermedio de mensajes.

Para ser satisfactorio, el formato de ImqMessage debe ser MQFMT_REF_MSG_HEADER.

Consulte la descripción del método de clase ImqHeader en [“Clase C++ ImqHeader” en la página 1859](#) para obtener más detalles.

Métodos de objeto (public)**void operador = (const ` ImqReferenceHeader ` & header);**

Copia los datos de instancia de la *cabecera*, sustituyendo los datos de instancia existentes.

ImqString destinationEnvironment () Const.

Devuelve una copia del entorno de destino.

void setDestinationEnvironment (const char *environment = 0);

Establece el entorno de destino.

ImqString destinationName () Const.

Devuelve una copia del nombre de destino.

void setDestinationName (const char * nombre = 0);

Establece el nombre de destino.

ImqBinary instanceId () Const.

Devuelve una copia del ID de instancia.

ImqBoolean setInstanceId (const ImqBinary & id);

Establece el ID de instancia. La longitud de datos de *token* debe ser 0 o MQ_OBJECT_INSTANCE_ID_LENGTH. Este método devuelve TRUE si es satisfactorio.

void setInstanceId (const MQBYTE24 id = 0);

Establece el ID de instancia. *id* puede ser cero, que es lo mismo que especificar MQOII_NONE. Si *id* es distinto de cero, debe direccionar MQ_OBJECT_INSTANCE_ID_LENGTH bytes de datos binarios. Cuando se utilizan valores predefinidos como MQOII_NONE, es posible que tenga que realizar una conversión para garantizar una coincidencia de firma, por ejemplo (MQBYTE *) MQOII_NONE.

MQLONG logicalLength () Const.

Devuelve la longitud lógica.

void setLogicalLength (const MQLONG longitud);

Establece la longitud lógica.

MQLONG logicalOffset () Const.

Devuelve el desplazamiento lógico.

void setLogicalOffset (const MQLONG *offset*);

Establece el desplazamiento lógico.

MQLONG logicalOffset2 () Const.

Devuelve el desplazamiento lógico 2.

void setLogicalOffset2 (const MQLONG *desplazamiento*);

Establece el desplazamiento lógico 2.

ImqString referenceType () Const.

Devuelve una copia del tipo de referencia.

void setReferenceType (const char * *nombre* = 0);

Establece el tipo de referencia.

ImqString sourceEnvironment () Const.

Devuelve una copia del entorno de origen.

void setSourceEnvironment (const char **environment* = 0);

Establece el entorno de origen.

ImqString sourceName () Const.

Devuelve una copia del nombre de origen.

void setSourceName (const char * *nombre* = 0);

Establece el nombre de origen.

Datos de objeto (protegidos)

MQRMH *omqrmh*

Estructura de datos MQRMH.

códigos de razón

- MQRC_ERROR_DE_LONGITUD_DE_DATOS_BINARIOS
- MQRC_ERROR_DE_LONGITUD_DE_ESTRUCTURA
- MQRC_STRUC_ID_ERROR
- MQRC_INSUFICIENT_DATA
- MQRC_FORMATO_INCONSISTENTE
- MQRC_ERROR_DE_CODIFICACIÓN

Clase C++ ImqString

Esta clase proporciona almacenamiento de serie de caracteres y manipulación para series terminadas en nulo.

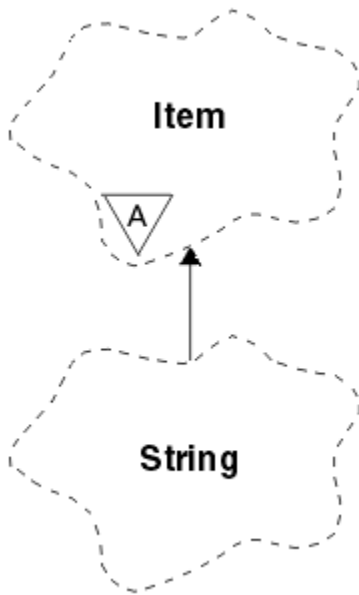


Figura 34. Clase *ImqString*

Utilice un *ImqString* en lugar de un **char *** en la mayoría de las situaciones en las que un parámetro llama a un **char ***.

- [“Atributos de objetos” en la página 1917](#)
- [“Constructores” en la página 1917](#)
- [“Métodos de clase \(public\)” en la página 1918](#)
- [“Métodos *ImqItem* sobrecargados” en la página 1918](#)
- [“Métodos de objeto \(public\)” en la página 1918](#)
- [“Métodos de objeto \(protegidos\)” en la página 1921](#)
- [“códigos de razón” en la página 1921](#)

Atributos de objetos

caracteres

Caracteres del **almacenamiento** que preceden a un nulo final.

longitud

Número de bytes en los **caracteres**. Si no hay ningún **almacenamiento**, la **longitud** es cero. El valor inicial es cero.

almacenamiento

Una matriz volátil de bytes de tamaño arbitrario. Un nulo final siempre debe estar presente en el **almacenamiento** después de los **caracteres**, para que se pueda detectar el final de los **caracteres**. Los métodos aseguran que esta situación se mantiene, pero aseguran, cuando se establecen los bytes en la matriz directamente, que existe un nulo final después de la modificación. Inicialmente, no hay ningún atributo **storage**.

Constructores

ImqString();

El constructor predeterminado.

ImqString(const ImqString, string);

El constructor de copia.

ImqString(const char c);

Los **caracteres** comprenden c.

ImqString(const char * *texto*);

Los **caracteres** se copian de *texto*.

ImqString(const void * *buffer*, const size_t *length*);

Copia *longitud* bytes a partir de *almacenamiento intermedio* y los asigna a los **caracteres**. La sustitución se realiza para cualquier carácter nulo copiado. El carácter de sustitución es un punto (.). No se da ninguna consideración especial a ningún otro carácter no imprimible o no visualizable copiado.

Métodos de clase (public)

static ImqBoolean copy (char * *buffer-destino*, const size_t *length*, const char * *buffer-origen*, const char *pad* = 0);

Copia hasta *longitud* bytes de *source-buffer* a *destination-buffer*. Si el número de caracteres en *almacenamiento intermedio de origen* es insuficiente, rellena el espacio restante en *almacenamiento intermedio de destino* con *relleno* caracteres. *source-buffer* puede ser cero. *destination-buffer* puede ser cero si *length* también es cero. Los códigos de error se pierden. Este método devuelve TRUE si es satisfactorio.

static ImqBoolean copy (char * *buffer-destino*, const size_t *length*, const char * *buffer-origen*, ImqError & *objeto-error*, const char *pad* = 0);

Copia hasta *longitud* bytes de *source-buffer* a *destination-buffer*. Si el número de caracteres en *almacenamiento intermedio de origen* es insuficiente, rellena el espacio restante en *almacenamiento intermedio de destino* con *relleno* caracteres. *source-buffer* puede ser cero. *destination-buffer* puede ser cero si *length* también es cero. Los códigos de error se establecen en *objeto de error*. Este método devuelve TRUE si es satisfactorio.

Métodos ImqItem sobrecargados

virtual ImqBoolean copyOut (ImqMessage & *msg*);

Copia los **caracteres** en el almacenamiento intermedio de mensajes, sustituyendo cualquier contenido existente. Establece el formato *msg* en MQFMT_STRING.

Consulte la descripción del método de clase padre para obtener más detalles.

virtual ImqBoolean pasteIn (ImqMessage & *msg*);

Establece los **caracteres** transfiriendo los datos restantes del almacenamiento intermedio de mensajes, sustituyendo los **caracteres** existentes.

Para que la operación sea satisfactoria, la **codificación** del objeto *msg* debe ser MQENC_NATIVE. Recuperar mensajes con MQGMO_CONVERT a MQENC_NATIVE.

Para ser satisfactorio, el **formato** ImqMessage debe ser MQFMT_STRING.

Consulte la descripción del método de clase padre para obtener más detalles.

Métodos de objeto (public)

char & operator [] (const size_t *desplazamiento*) Const.

Hace referencia al carácter en desplazamiento *offset* en el **almacenamiento**. Asegúrese de que el byte relevante existe y es direccionable.

Operador ImqString () (const size_t *desplazamiento*, const size_t *longitud* = 1) Const.

Devuelve una subserie copiando bytes de los **caracteres** a partir de *desplazamiento*. Si *length* es cero, devuelve el resto de los **caracteres**. Si la combinación de *offset* y *length* no produce una referencia dentro de los **caracteres**, devuelve una ImqString vacía.

void operator = (const ImqString & *serie*);

Copia datos de instancia de *serie*, sustituyendo los datos de instancia existentes.

Operador ImqString + (const char *c*) Const.

Devuelve el resultado de añadir *c* a los **caracteres**.

ImqString operator + (const char * texto) Const.

Devuelve el resultado de añadir *text* a los **caracteres**. Esto también se puede invertir. Por ejemplo:

```
strOne + "string two" ;
"string one" + strTwo ;
```

Nota: Aunque la mayoría de compiladores aceptan **strOne + "string two"**; Microsoft Visual C++ requiere **strOne + (char *) "string two"**;

Operador ImqString + (const ImqString & string1) Const.

Devuelve el resultado de añadir *string1* a los **caracteres**.

ImqString operator + (const double número) Const.

Devuelve el resultado de añadir *número* a los **caracteres** después de la conversión a texto.

Operador ImqString + (const long número) Const.

Devuelve el resultado de añadir *número* a los **caracteres** después de la conversión a texto.

void operator + = (const char c);

Añade *c* a los **caracteres**.

void operator + = (const char * texto);

Añade *text* a los **caracteres**.

void operator + = (const ImqString & serie);

Añade *string* a los **caracteres**.

void operator + = (const double número);

Añade *número* a los **caracteres** después de la conversión a texto.

void operator + = (const long número);

Añade *número* a los **caracteres** después de la conversión a texto.

operador char * () Const.

Devuelve la dirección del primer byte en el **almacenamiento**. Este valor puede ser cero y es volátil. Utilice este método sólo para fines de sólo lectura.

Operador ImqBoolean < (const ImqString & serie) Const.

Compara los **caracteres** con los de *serie* utilizando el método **compare** . El resultado es TRUE si es menor que y FALSE si es mayor que o igual a.

Operador ImqBoolean > (const ImqString & serie) Const.

Compara los **caracteres** con los de *serie* utilizando el método **compare** . El resultado es TRUE si es mayor que y FALSE si es menor o igual a.

Operador ImqBoolean < = (const ImqString & serie) Const.

Compara los **caracteres** con los de *serie* utilizando el método **compare** . El resultado es TRUE si es menor o igual que y FALSE si es mayor que.

Operador ImqBoolean > = (const ImqString & serie) Const.

Compara los **caracteres** con los de *serie* utilizando el método **compare** . El resultado es TRUE si es mayor o igual que y FALSE si es menor que.

Operador ImqBoolean == (const ImqString & serie) Const.

Compara los **caracteres** con los de *serie* utilizando el método **compare** . Devuelve TRUE o FALSE.

Operador ImqBoolean != (const ImqString & serie) Const.

Compara los **caracteres** con los de *serie* utilizando el método **compare** . Devuelve TRUE o FALSE.

comparación corta (const ImqString & serie) Const.

Compara los **caracteres** con los de *serie*. El resultado es cero si los **caracteres** son iguales, negativo si es menor que y positivo si es mayor que. La comparación es sensible a las mayúsculas y minúsculas. Una ImqString nula se considera menor que una ImqString no nula.

ImqBoolean copyOut(char * buffer, const size_t length, const char pad = 0);

Copia hasta *longitud* bytes de los **caracteres** en el *almacenamiento intermedio*. Si el número de **caracteres** es insuficiente, rellena el espacio restante en el *almacenamiento intermedio* con *relleno* caracteres. *buffer* puede ser cero si *length* también es cero. Devuelve TRUE si es satisfactorio.

size_t copyOut(long & number) const;

Establece *número* de los **caracteres** después de la conversión a partir del texto y devuelve el número de caracteres implicados en la conversión. Si es cero, no se ha realizado ninguna conversión y no se ha establecido *número* . Una secuencia de caracteres convertible debe empezar con los valores siguientes:

```
<blank(s)>
<+|->
digit(s)
```

size_t copyOut(ImqString & token, const char c = ' ') const;

Si los **caracteres** contienen uno o más caracteres que son diferentes de *c*, identifica una señal como la primera secuencia contigua de dichos caracteres. En este caso, *token* se establece en esa secuencia y el valor devuelto es la suma del número de caracteres iniciales *c* y el número de bytes de la secuencia. De lo contrario, devuelve cero y no establece *token*.

size_t cutOut(long & number);

Establece *número* como para el método **copy** , pero también elimina de **caracteres** el número de bytes indicado por el valor de retorno. Por ejemplo, la serie que se muestra en el ejemplo siguiente se puede cortar en tres números utilizando **cutOut** (*número*) tres veces:

```
strNumbers = "-1 0 +55 "

while ( strNumbers.cutOut( number ) );
number becomes -1, then 0, then 55
leaving strNumbers == " "
```

size_t cutOut(ImqString & token, const char c = ' ')

Establece *token* como para el método **copyOut** y elimina de **caracteres** los *strToken* caracteres y también los caracteres *c* que preceden a la *token* caracteres. Si *c* no está en blanco, elimina los caracteres *c* que se utilizan directamente para los caracteres *token* . Devuelve el número de caracteres eliminados. Por ejemplo, la serie que se muestra en el ejemplo siguiente se puede cortar en tres señales utilizando **cutOut** (*token*) tres veces:

```
strText = " Program Version 1.1 "

while ( strText.cutOut( token ) );

// token becomes "Program", then "Version",
// then "1.1" leaving strText == " "
```

El ejemplo siguiente muestra cómo analizar un nombre de vía de acceso de DOS:

```
strPath = "C:\OS2\BITMAP\OS2LOGO.BMP"

strPath.cutOut( strDrive, ':' );
strPath.stripLeading( ':' );
while ( strPath.cutOut( strFile, '\\' ) );

// strDrive becomes "C".
// strFile becomes "OS2", then "BITMAP",
// then "OS2LOGO.BMP" leaving strPath empty.
```

ImqBoolean find (const ImqString & serie);

Busca una coincidencia exacta para *serie* en cualquier lugar dentro de los **caracteres**. Si no se encuentra ninguna coincidencia, devuelve FALSE. De lo contrario, devuelve TRUE. Si *string* es nulo, devuelve TRUE.

ImqBoolean find (const ImqString & serie, size_t & desplazamiento);

Busca una coincidencia exacta para *string* en algún lugar dentro de los **caracteres** del desplazamiento *offset* en adelante. Si *string* es nulo, devuelve TRUE sin actualizar *offset*. Si no se encuentra ninguna coincidencia, devuelve FALSE (es posible que se haya aumentado el valor de *offset*). Si se encuentra una coincidencia, devuelve TRUE y actualiza *offset* al desplazamiento de *string* dentro de los **caracteres**.

size_t longitud () Const.

Devuelve la **longitud**.

ImqBoolean pasteIn(const double *número*, const char * *formato* = "%f");

Añade *número* a los **caracteres** después de la conversión a texto. Devuelve TRUE si es satisfactorio.

La especificación *format* se utiliza para formatear la conversión de coma flotante. Si se especifica, debe ser uno adecuado para su uso con **printf** y números de coma flotante, por ejemplo **%3f**.

ImqBoolean pasteIn(const long *número*);

Añade *número* a los **caracteres** después de la conversión a texto. Devuelve TRUE si es satisfactorio.

ImqBoolean pasteIn(const void * *buffer*, const size_t *length*);

Añade *longitud* bytes de *almacenamiento intermedio* a los **caracteres** y añade un valor nulo final.

Sustituye los caracteres nulos copiados. El carácter de sustitución es un punto (.). No se da ninguna consideración especial a ningún otro carácter no imprimible o no visualizable copiado. Este método devuelve TRUE si es satisfactorio.

ImqBoolean set (const char * *buffer*, const size_t *longitud*);

Establece los **caracteres** de un campo de caracteres de longitud fija, que puede contener un valor nulo. Añade un valor nulo a los caracteres del campo de longitud fija si es necesario. Este método devuelve TRUE si es satisfactorio.

ImqBoolean setStorage(const size_t *longitud*);

Asigna (o reasigna) el **almacenamiento**. Conserva los **caracteres** originales, incluidos los nulos finales, si todavía hay espacio para ellos, pero no inicializa ningún almacenamiento adicional.

Este método devuelve TRUE si es satisfactorio.

size_t storage () Const.

Devuelve el número de bytes en el **almacenamiento**.

size_t stripLeading(const char *c* = ' ');

Elimina los caracteres iniciales *c* de los **caracteres** y devuelve el número eliminado.

size_t stripTrailing(const char *c* = ' ');

Elimina los caracteres de cola *c* de los **caracteres** y devuelve el número eliminado.

ImqString upperCase() const;

Devuelve una copia en mayúsculas de los **caracteres**.

Métodos de objeto (protegidos)**ImqBoolean asignar (const ImqString & *serie*);**

Equivalente al método **operator =** equivalente, pero no virtual. Devuelve TRUE si es satisfactorio.

códigos de razón

- MQRC_DATA_TRUNCADO
- MQRC_PUNTERO_NULO
- MQRC_STORAGE_NOT_AVAILABLE
- MQRC_BUFFER_ERROR
- MQRC_FORMATO_INCONSISTENTE

Clase C++ ImqTrigger

Esta clase encapsula la estructura de datos MQTM (mensaje desencadenante).

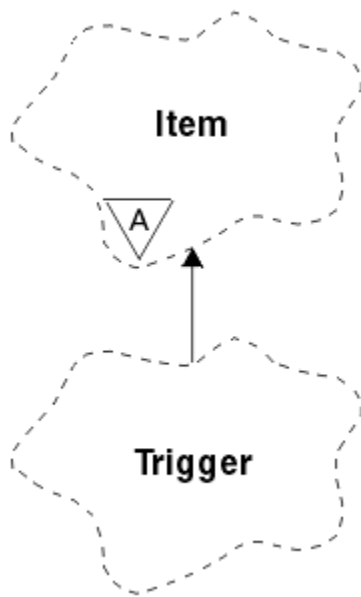


Figura 35. Clase *ImqTrigger*

Los objetos de esta clase suelen ser utilizados por un programa supervisor desencadenante. La tarea de un programa supervisor desencadenante es esperar estos mensajes concretos y actuar sobre ellos para asegurarse de que otras aplicaciones de IBM MQ se inician cuando los mensajes los esperan.

Consulte el programa de ejemplo IMQSTRG para ver un ejemplo de uso.

- [“Atributos de objetos” en la página 1922](#)
- [“Constructores” en la página 1923](#)
- [“Métodos ImqItem sobrecargados” en la página 1923](#)
- [“Métodos de objeto \(public\)” en la página 1923](#)
- [“Datos de objeto \(protegidos\)” en la página 1924](#)
- [“códigos de razón” en la página 1924](#)

Atributos de objetos

ID de aplicación

Identidad de la aplicación que ha enviado el mensaje. El valor inicial es una serie nula.

Tipo de aplicación

Tipo de aplicación que ha enviado el mensaje. El valor inicial es cero. Son posibles los siguientes valores adicionales:

- MQAT_AIX
- MQAT_CICS
- MQAT_DOS
- MQAT_IMS
- MQAT_MVS
- MQAT_NOTAS_AGENTE
- MQAT_OS2
- MQAT_OS390
- MQAT_OS400
- MQAT_UNIX
- MQAT_WINDOWS

- MQAT_WINDOWS_NT
- MQAT_USER_FIRST
- MQAT_USER_LAST

Datos de entorno

Datos de entorno para el proceso. El valor inicial es una serie nula.

nombre de proceso

Nombre del proceso. El valor inicial es una serie nula.

nombre de cola

Nombre de la cola que se va a iniciar. El valor inicial es una serie nula.

Datos desencadenantes

Desencadenar datos para el proceso. El valor inicial es una serie nula.

datos de usuario

Datos de usuario para el proceso. El valor inicial es una serie nula.

Constructores

ImqTrigger();

El constructor predeterminado.

ImqTrigger(const ImqTrigger, *trigger*);

El constructor de copia.

Métodos ImqItem sobrecargados

virtual ImqBoolean copyOut (ImqMessage & *msg*);

Escribe una estructura de datos MQTM en el almacenamiento intermedio de mensajes, sustituyendo cualquier contenido existente. Establece el formato *msg* en MQFMT_TRIGGER.

Consulte la descripción del método de clase ImqItem en [“Clase C++ ImqItem” en la página 1863](#) para obtener más detalles.

virtual ImqBoolean pasteIn (ImqMessage & *msg*);

Lee una estructura de datos MQTM del almacenamiento intermedio de mensajes.

Para ser satisfactorio, el formato ImqMessage debe ser MQFMT_TRIGGER.

Consulte la descripción del método de clase ImqItem en [“Clase C++ ImqItem” en la página 1863](#) para obtener más detalles.

Métodos de objeto (public)

void operator = (const ImqTrigger & *desencadenante*);

Copia datos de instancia del *desencadenante*, sustituyendo los datos de instancia existentes.

ImqString applicationId () Const.

Devuelve una copia del ID de aplicación.

void setApplicationId (const char * *id*);

Establece el ID de aplicación.

MQLONG applicationType () Const.

Devuelve el tipo de aplicación.

void setApplicationType (const MQLONG *tipo*);

Establece el tipo de aplicación.

ImqBoolean copyOut (MQTMC2 * *ptmc2*);

Encapsula la estructura de datos MQTM, que es la que se recibe en las colas de inicio. Rellena una estructura de datos MQTMC2 equivalente proporcionada por el llamante y establece el campo QMgrName (que no está presente en la estructura de datos MQTM) en todos los espacios en blanco.

La estructura de datos MQTMC2 se utiliza tradicionalmente como parámetro para las aplicaciones iniciadas por un supervisor desencadenante. Este método devuelve TRUE si es satisfactorio.

ImqString environmentData () Const.

Devuelve una copia de los datos de entorno.

void setEnvironmentData (const char* data);

Establece los datos de entorno.

ImqString processName () Const.

Devuelve una copia del nombre de proceso.

void setProcessName (const char * nombre);

Establece el nombre de proceso, relleno con espacios en blanco en 48 caracteres.

ImqString queueName () Const.

Devuelve una copia del nombre de cola.

void setQueueName (const char * nombre);

Establece el nombre de cola, relleno con espacios en blanco en 48 caracteres.

ImqString triggerData () Const.

Devuelve una copia de los datos de desencadenante.

void setTriggerData (const char* data);

Establece los datos de desencadenante.

ImqString userData () Const.

Devuelve una copia de los datos de usuario.

void setUserData (const char* data);

Establece los datos de usuario.

Datos de objeto (protegidos)

MQTM omqtm

Estructura de datos MQTM.

códigos de razón

- MQRC_PUNTERO_NULO
- MQRC_FORMATO_INCONSISTENTE
- MQRC_ERROR_DE_CODIFICACIÓN
- MQRC_STRUC_ID_ERROR

ImqWorkHeader Clase C

Esta clase encapsula características específicas de la estructura de datos MQWIH.

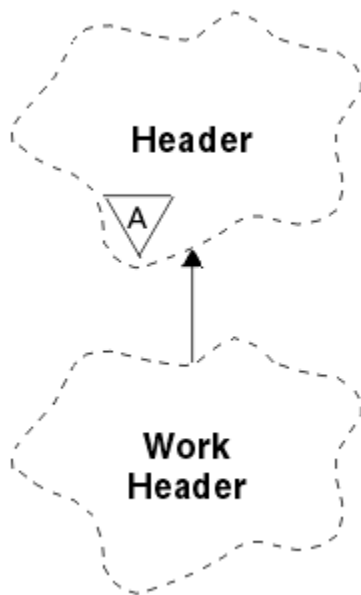


Figura 36. *ImqWorkHeader* clase

Los objetos de esta clase los utilizan las aplicaciones que colocan mensajes en la cola gestionada por el gestor de carga de trabajo de z/OS .

- [“Atributos de objetos” en la página 1925](#)
- [“Constructores” en la página 1925](#)
- [“Métodos ImqItem sobrecargados” en la página 1925](#)
- [“Métodos de objeto \(public\)” en la página 1926](#)
- [“Datos de objeto \(protegidos\)” en la página 1926](#)
- [“códigos de razón” en la página 1926](#)

Atributos de objetos

token de mensaje

Señal de mensaje para el gestor de carga de trabajo de z/OS , de longitud MQ_MSG_TOKEN_LENGTH.
El valor inicial es MQMTOK_NONE.

nombre del servicio

Nombre de 32 caracteres de un proceso. El nombre está inicialmente en blanco.

paso de servicio

El nombre de 8 caracteres de un paso dentro del proceso. El nombre está inicialmente en blanco.

Constructores

ImqWorkHeader();

El constructor predeterminado.

ImqWorkHeader(const ImqWorkHeader &header);

El constructor de copia.

Métodos ImqItem sobrecargados

virtual ImqBoolean copyOut(ImqMessage & msg);

Inserta una estructura de datos MQWIH al principio del almacenamiento intermedio de mensajes, moviendo los datos de mensaje existentes más adelante y establece el formato *msg* en MQFMT_WORK_INFO_HEADER.

Consulte la descripción del método de clase padre para obtener más detalles.

virtual ImqBoolean pasteIn(ImqMessage & msg);

Lee una estructura de datos MQWIH del almacenamiento intermedio de mensajes.

Para que la codificación del objeto *msg* sea satisfactoria, debe ser MQENC_NATIVE. Recuperar mensajes con MQGMO_CONVERT a MQENC_NATIVE.

El formato ImqMessage debe ser MQFMT_WORK_INFO_HEADER.

Consulte la descripción del método de clase padre para obtener más detalles.

Métodos de objeto (public)

void operador = (const ` ImqWorkHeader ` & header);

Copia los datos de instancia de la *cabecera*, sustituyendo los datos de instancia existentes.

ImqBinary messageToken () Const.

Devuelve la **señal de mensaje**.

ImqBoolean setMessageToken(const ImqBinary, token);

Establece la **señal de mensaje**. La longitud de datos de *token* debe ser cero o MQ_MSG_TOKEN_LENGTH. Devuelve TRUE si es satisfactorio.

void setMessageToken(const MQBYTE16 token = 0);

Establece la **señal de mensaje**. *token* puede ser cero, que es lo mismo que especificar MQMTOK_NONE. Si *token* es distinto de cero, debe dirigirse a MQ_MSG_TOKEN_LENGTH bytes de datos binarios.

Cuando se utilizan valores predefinidos como MQMTOK_NONE, es posible que tenga que realizar una conversión para garantizar una coincidencia de firma; por ejemplo, (MQBYTE *) MQMTOK_NONE.

ImqString serviceName () Const.

Devuelve el **nombre de servicio**, incluidos los espacios en blanco finales.

void setServiceName(const char * nombre);

Establece el **nombre de servicio**.

ImqString serviceStep () Const.

Devuelve el **paso de servicio**, incluidos los espacios en blanco finales.

void setServiceStep(const char *step);

Establece el **paso de servicio**.

Datos de objeto (protegidos)

MQWIH omqwih

Estructura de datos MQWIH.

códigos de razón

- MQRC_ERROR_DE_LONGITUD_DE_DATOS_BINARIOS

Propiedades de IBM MQ classes for JMS los objetos

Todos los objetos en IBM MQ classes for JMS tienen propiedades. Las diferentes propiedades se aplican a los diferentes tipos de objetos. Las diferentes propiedades tienen diferentes valores permitidos, y los valores simbólicos de las propiedades difieren entre la herramienta de administración y el código del programa.

IBM MQ classes for JMS proporciona facilidades para establecer y consultar las propiedades de los objetos utilizando la herramienta IBM MQJMS de administración, IBM MQ Explorer, o en una aplicación. Muchas de las propiedades solo son relevantes para un subconjunto específico de los tipos de objetos.

Para obtener información sobre cómo utilizar la herramienta IBM MQJMS de administración, consulte [Configuración de JMS objetos mediante la herramienta de administración](#).

Tabla 904 en la página 1927 ofrece una breve descripción de cada propiedad y muestra para cada propiedad a qué tipos de objetos se aplica. Los tipos de objetos se identifican mediante palabras clave; consulte Configuración JMS de objetos mediante la herramienta de administración para obtener una explicación de estos objetos.

Los números se refieren a las notas al final de la tabla. Consulte también el tema “Dependencias entre propiedades de objetos IBM MQ classes for JMS” en la página 1931.

Una propiedad consiste en un par nombre-valor con el formato:

```
PROPERTY_NAME(property_value)
```

Los temas de esta sección enumeran, para cada propiedad, el nombre de la propiedad y una breve descripción, y muestran los valores válidos de la propiedad utilizados en la herramienta de administración, y el método set que se utiliza para establecer el valor de la propiedad en una aplicación. Los temas también muestran los valores válidos de cada propiedad y la correspondencia entre los valores simbólicos de las propiedades utilizados en la herramienta y sus equivalentes programables.

Los nombres de las propiedades no distinguen entre mayúsculas y minúsculas, y se limitan al conjunto de nombres reconocidos que se muestran en estos temas.

Tabla 904. Nombres de propiedades y tipos de objetos aplicables									
Propiedad	Formato abreviado	Tipo de objeto							
		CF	QCF	TCF	Q	T	XACF	XAQCF	XATCF
“APPLICATIONNAME” en la página 1932	APPNAME	Y	Y	Y			Y	Y	Y
“ASYNCEXCEPTION” en la página 1933	AEX	Y	Y	Y			Y	Y	Y
“BALOPTIONS” en la página 1934	OPCIONES	Y	Y	Y			Y	Y	Y
“BALTYPE” en la página 1934	TIPO	Y	Y	Y			Y	Y	Y
“BALTIMEOUT” en la página 1935	TIMEOUT	Y	Y	Y			Y	Y	Y
“BROKERCCDURSUBQ” en la página 1936 ¹	CCDSUB					Y			
“BROKERCCSUBQ” en la página 1936 ¹	CCSUB	Y		Y			Y		Y
“BROKERCONQ” en la página 1937 ¹	BCON	Y		Y			Y		Y
“BROKERDURSUBQ” en la página 1937 ¹	BDSUB					Y			
“BROKERPUBQ” en la página 1937 ¹	BPUB	Y		Y		Y	Y		Y
“BROKERPUBQMGR” en la página 1938 ¹	BPQM					Y			
“BROKERQMGR” en la página 1938 ¹	BQM	Y		Y			Y		Y
“BROKERSUBQ” en la página 1939 ¹	BSUB	Y		Y			Y		Y
“BROKERVER” en la página 1939 ¹	BVER	Y ²		Y ²		Y	Y		Y
“CCDTURL” en la página 1940 ³	CCDT	Y	Y	Y			Y	Y	Y
“CERTVALPO” en la página 1941	CVAP	Y							
“CCSID” en la página 1940	CCS	Y	Y	Y	Y	Y	Y	Y	Y
“CHANNEL” en la página 1942 ³	CHAN	Y	Y	Y			Y	Y	Y

Tabla 904. Nombres de propiedades y tipos de objetos aplicables (continuación)

Propiedad	Formato abreviado	Tipo de objeto							
		CF	QCF	TCF	Q	T	XACF	XAQCF	XATCF
“CLEANUP” en la página 1942 ¹	CL	Y		Y			Y		Y
“CLEANUPINT” en la página 1943 ¹	CLINT	Y		Y			Y		Y
“ConnectionNameList” en la página 1943	CNLIST	Y	Y	Y					
“CLIENTRECONNECTOPTIONS” en la página 1943	CROPT	Y	Y	Y					
“CLIENTRECONNECTTIMEOUT” en la página 1944	CRT	Y	Y	Y					
“CLIENTID” en la página 1945	CID	Y ²	Y	Y ²			Y	Y	Y
“CLONESUPP” en la página 1945	CLS	Y		Y			Y		Y
“COMPHDR” en la página 1946	HC	Y		Y			Y		Y
“COMPMSG” en la página 1946	MC	Y	Y	Y			Y	Y	Y
“CONNOPT” en la página 1947	CNOPT	Y	Y	Y			Y	Y	Y
“CONNTAG” en la página 1948	CNTAG	Y	Y	Y			Y	Y	Y
“DESCRIPCIÓN” en la página 1948	DESC	Y ²	Y	Y ²	Y	Y	Y	Y	Y
“DIRECTAUTH” en la página 1949	DAUTH	Y ²		Y ²					
“ENCODING” en la página 1949	ENC				Y	Y			
“EXPIRY” en la página 1950	EXP				Y	Y			
“FAILIFQUIESCE” en la página 1950	FIQ	Y	Y	Y	Y	Y	Y	Y	Y
“HOSTNAME” en la página 1951	HOST	Y ²	Y	Y ²			Y	Y	Y
“LOCALADDRESS” en la página 1952	LA	Y ²	Y	Y ²			Y	Y	Y
“ESTILO de nombre de mapa” en la página 1952	MNST	Y	Y	Y			Y	Y	Y
“MAXBUFFSIZE” en la página 1953	MBSZ	Y ²		Y ²					
“MDREAD” en la página 1953	MDR				Y	Y			
“MDWRITE” en la página 1954	MDW				Y	Y			
“MDMSGCTX” en la página 1955	MDCTX				Y	Y			
“MSGBATCHSZ” en la página 1955 ¹	MBS	Y	Y	Y			Y	Y	Y
“MSGBODY” en la página 1956	MBODY				Y	Y			
“MSGRETENTION” en la página 1956	MRET	Y	Y				Y	Y	
“MSGSELECTION” en la página 1957 ¹	MSEL	Y		Y			Y		Y
“MULTICAST” en la página 1957	MCAST	Y ²		Y ²		Y			
“OPTIMISTICPUBLICATION” en la página 1958 ¹	OPTPUB	Y		Y					

Tabla 904. Nombres de propiedades y tipos de objetos aplicables (continuación)

Propiedad	Formato abreviado	Tipo de objeto							
		CF	QCF	TCF	Q	T	XACF	XAQCF	XATCF
“OUTCOMENOTIFICATION” en la página 1959 ¹	NOTIFY	Y		Y					
“PERSISTENCE” en la página 1959	PER				Y	Y			
“POLLINGINT” en la página 1960 ¹	PINT	Y	Y	Y			Y	Y	Y
“Puerto” en la página 1960	Puerto	Y ²	Y	Y ²			Y	Y	Y
“PRIORITY” en la página 1961	PRI				Y	Y			
“PROCESSDURATION” en la página 1961 ¹	PROCDUR	Y		Y					
“PROVIDERVERSION” en la página 1962	PVER	Y	Y	Y			Y	Y	Y
“PROXYHOSTNAME” en la página 1964	PHOST	Y ²		Y ²					
“PROXYPORT” en la página 1964	PPORT	Y ²		Y ²					
“PUBACKINT” en la página 1965 ¹	PAI	Y		Y			Y		Y
“PUTASYNCAALLOWED” en la página 1965	PAALD				Y	Y			
“QMANAGER” en la página 1966	QMGR	Y	Y	Y	Y		Y	Y	Y
“COLA” en la página 1966	QU				Y				
“READAHEADALLOWED” en la página 1967	RAALD				Y	Y			
“READAHEADCLOSEPOLICY” en la página 1968	RACP				Y	Y			
“RECEIVECCSID” en la página 1968	RCCS				Y	Y			
“RECEIVECONVERSION” en la página 1969	RCNV				Y	Y			
“RECEIVEISOLATION” en la página 1969 ¹	RCVISOL	Y		Y					
“RECEXIT” en la página 1970	RCX	Y	Y	Y			Y	Y	Y
“RECEXITINIT” en la página 1970	RCXI	Y	Y	Y			Y	Y	Y
“REPLYTOSTYLE” en la página 1971	RTOST				Y	Y			
“RESCANINT” en la página 1971 ¹	RINT	Y	Y				Y	Y	
“SECEXIT” en la página 1972	SCX	Y	Y	Y			Y	Y	Y
“SECEXITINIT” en la página 1972	SCXI	Y	Y	Y			Y	Y	Y
“SENDCHECKCOUNT” en la página 1973	SCC	Y	Y	Y			Y	Y	Y
“SENDEXIT” en la página 1973	SDX	Y	Y	Y			Y	Y	Y
“SENDEXITINIT” en la página 1974	SDXI	Y	Y	Y			Y	Y	Y

Tabla 904. Nombres de propiedades y tipos de objetos aplicables (continuación)

Propiedad	Formato abreviado	Tipo de objeto							
		CF	QCF	TCF	Q	T	XACF	XAQCF	XATCF
“SHARECONVALLOWED” en la página 1974	QUEMADURA	Y	Y	Y			Y	Y	Y
“SPARSESUBS” en la página 1975 ¹	SSUBS	Y		Y					
“SSLCIPHERSUITE” en la página 1975	SCPHS	Y	Y	Y			Y	Y	Y
“SSLCRL” en la página 1976	SCRL	Y	Y	Y			Y	Y	Y
“SSLFIPSREQUIRED” en la página 1976	SFIPS	Y	Y	Y			Y	Y	Y
“SSLPEERNAME” en la página 1978	SPEER	Y	Y	Y			Y	Y	Y
“SSLRESETCOUNT” en la página 1978	SRC	Y	Y	Y			Y	Y	Y
“STATREFRESHINT” en la página 1979 ¹	SRI	Y		Y			Y		Y
“SUBSTORE” en la página 1979 ¹	SS	Y		Y			Y		Y
“SYNCPOINTALLGETS” en la página 1980	SPAG	Y	Y	Y			Y	Y	Y
“TARGCLIENT” en la página 1980	TC				Y	Y			
“TARGCLIENTMATCHING” en la página 1981	TCM	Y	Y				Y	Y	
“TEMPMODEL” en la página 1981	TM	Y	Y				Y	Y	
“TEMPQPREFIX” en la página 1982	TQP	Y	Y				Y	Y	
“TEMPTOPICPREFIX” en la página 1982	TTP	Y		Y			Y		Y
v 7000 ID_CLIENTE_TOKEN	TKCD	Y	Y	Y			Y	Y	Y
v 7000 SECRETO_DEL_CLIENTE_TOKEN	TKST	Y	Y	Y			Y	Y	Y
v 7000 PUNTO FINAL DEL TOKEN	TKEP	Y	Y	Y			Y	Y	Y
v 7000 POLÍTICA_DE_VALIDACIÓN_DE_CERTIFICADOS_HTTPS_TOKEN	TCVP	Y	Y	Y			Y	Y	Y
“TOPIC” en la página 1985	TOP					Y			
“TRANSPORT” en la página 1985	TRAN	Y ²	Y	Y ²			Y	Y	Y
“WILDCARDFORMAT” en la página 1986	WCFMT	Y		Y			Y		Y
v 7000 Política de validación de certificados HTTPS de WMQ_CCDT	CCVP	Y	Y	Y			Y	Y	Y

Notas:

1. Esta propiedad se puede utilizar con la versión 70 de IBM MQ classes for JMS , pero no tiene ningún efecto para una aplicación conectada a un IBM WebSphere MQ 7.0 gestor de colas a menos que la **PROVIDERVERSION** propiedad de la fábrica de conexiones esté establecida en un número de versión inferior a 7.

2. Solo se admiten las propiedades **BROKERVER**, **CLIENTID**, **DESCRIPTIONTRANSPORT**, **DIRECTAUTH**, **HOSTNAME**, **LOCALADDRESS**, **MAXBUFFSIZE**, **MULTICAST**, **PORT**, **PROXYHOSTNAME**, **PROXYPORT**, y para un objeto `ConnectionFactory` o `TopicConnectionFactory` cuando se utiliza una conexión en tiempo real con un bróker.
3. Las **CCDTURLCHANNEL** propiedades y de un objeto no deben establecerse al mismo tiempo.

Tareas relacionadas

crear objetos de JMS

Dependencias entre propiedades de objetos IBM MQ classes for JMS

La validez de algunas propiedades depende de los valores particulares de otras propiedades.

Esta dependencia puede producirse en los siguientes grupos de propiedades:

- Propiedades del cliente
- Propiedades para una conexión en tiempo real con un intermediario
- Series de inicialización de salida

Propiedades del cliente

Para una conexión con un gestor de colas, las propiedades siguientes sólo son relevantes si `TRANSPORT` tiene el valor `CLIENT`:

- `HOSTNAME`
- Puerto
- `CHANNEL`
- `LOCALADDRESS`
- `CCDTURL`
- `CCSID`
- `COMPHDR`
- `COMPMSG`
- `RECEXIT`
- `RECEXITINIT`
- `SECEXIT`
- `SECEXITINIT`
- `SENDEXIT`
- `SENDEXITINIT`
- `SHARECONVALLOWED`
- `SSLCIPHERSUITE`
- `SSLCRL`
- `SSLFIPSREQUIRED`
- `SSLPEERNAME`
- `SSLRESETCOUNT`
- `APPLICATIONNAME`

No puede establecer valores para estas propiedades utilizando la herramienta de administración si `TRANSPORT` tiene el valor `BIND`.

Si `TRANSPORT` tiene el valor `CLIENT`, el valor predeterminado de la propiedad `BROKERVER` es `V1` y el valor predeterminado de la propiedad `PORT` es `1414`. Si establece el valor de `BROKERVER` o `PORT` explícitamente, un cambio posterior en el valor de `TRANSPORT` no altera temporalmente las opciones.

Propiedades para una conexión en tiempo real con un intermediario

Solo son relevantes las propiedades siguientes si TRANSPORT tiene el valor DIRECT o DIRECTHTTP:

- BROKERVER
- CLIENTID
- DESCRIPCIÓN
- DIRECTAUTH
- HOSTNAME
- LOCALADDRESS
- MAXBUFFSIZE
- MULTICAST (soportado solo para DIRECT)
- Puerto
- PROXYHOSTNAME (soportado sólo para DIRECT)
- PROXYPORT (soportado sólo para DIRECT)

Si TRANSPORT tiene el valor DIRECT o DIRECTHTTP, el valor predeterminado de la propiedad BROKERVER es V2y el valor predeterminado de la propiedad PORT es 1506. Si establece el valor de BROKERVER o PORT explícitamente, un cambio posterior en el valor de TRANSPORT no altera temporalmente las opciones.

Series de inicialización de salida

No establezca ninguna de las series de inicialización de salida sin proporcionar el nombre de salida correspondiente. Las propiedades de inicialización de salida son:

- REEXITINIT
- SEEXITINIT
- SENDEXITINIT

Por ejemplo, si especifica REEXITINIT(myString) sin especificar REEXIT(some.exit.classname) , se produce un error.

Referencia relacionada

[“TRANSPORT” en la página 1985](#)

La naturaleza de una conexión a un gestor de colas o intermediario.

APPLICATIONNAME

Una aplicación puede definir un nombre que identifique su conexión al gestor de colas. El nombre de esta aplicación se muestra mediante el **DISPLAY CONN MQSC/PCF** comando (donde el campo se denomina **APPLTAG**) o en la pantalla **Conexiones** IBM MQ de aplicaciones del Explorador (donde el campo se denomina **App name**).

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS Nombre completo de la herramienta de administración: NOMBRE DE LA APLICACIÓN

JMS Nombre abreviado de la herramienta de administración: APPNAME

Acceso programático

Métodos de lectura y escritura

- MQConnectionFactory.setAppName()
- MQConnectionFactory.getAppName()

Valores

Cualquier cadena válida que no supere los 28 caracteres. Los nombres más largos se ajustan para que quepan eliminando los nombres de los paquetes iniciales, si es necesario. Por ejemplo, si la clase invocadora es `com.example.MainApp`, se utiliza el nombre completo, pero si la clase invocadora es `com.example.dictionaryAndThesaurus.multilingual.mainApp`, se utiliza el nombre `multilingual.mainApp`, porque es la combinación más larga formada por el nombre de clase y el nombre de paquete situado más a la derecha que se ajusta a la longitud disponible.

Si el nombre de clase propiamente dicho tiene más de 28 caracteres de longitud, se trunca para que quepa. Por ejemplo, `com.example.mainApplicationForSecondTestCase` pasa a ser `mainApplicationForSecondTest`.

 En z/OS, el APPNAME en:

- El modo de enlaces se ignora si se establece y, si se establece, solo se puede configurar en blanco.
- Se puede configurar y utilizar el modo cliente.

ASYNCEXCEPTION

Esta propiedad determina si IBM MQ classes for JMS informa a un `ExceptionListener` sólo cuando se interrumpe una conexión o cuando se produce una excepción de forma asíncrona en una llamada de API de JMS. Esto se aplica a todas las conexiones creadas a partir de esta `ConnectionFactory` que tienen un `ExceptionListener` registrado.

Objetos aplicables

`ConnectionFactory`, `QueueConnectionFactory`, `TopicConnectionFactory`, `XAConnectionFactory`, `XAQueueConnectionFactory`, `XATopicConnectionFactory`

Nombre largo de la herramienta de administración de JMS : `ASYNCEXCEPTION`

Nombre abreviado de la herramienta de administración de JMS : `AEX`

Acceso programático

Métodos setter/Getters

- `MQConnectionFactory.setAsyncExceptions()`
- `MQConnectionFactory.getAsyncExceptions()`

Valores

ASÍNCRONOS_EXCEPCIONES_TODO

Cualquier excepción detectada de forma asíncrona, fuera del ámbito de una llamada a API síncrona, y todas las excepciones de interrupción de conexión se envían al `ExceptionListener`.

Tabla 905. Todas las excepciones asíncronas: entornos y nombres de constantes relacionadas	
Entorno	Valor
JMS herramienta de administración	TODOS
Programático	<code>WMQCONSTANTS.ASYNC_EXCEPTIONS_ALL = -1</code>
IBM MQ Explorer	Todo

ASYNC_EXCEPTIONS_CONNECTION_BROKEN

Solo las excepciones que indican una conexión interrumpida se envían al `ExceptionListener`. Cualquier otra excepción que se produce durante el proceso asíncrono no se notifica al `ExceptionListener` y,

por lo tanto, la aplicación no está informada de estas excepciones. Este es el valor predeterminado a partir de IBM MQ 8.0.0 Fix Pack 2. Véase [JMS: Cambios en el receptor de excepciones en IBM MQ 8.0](#).

Tabla 906. Excepciones que indican una conexión interrumpida: entornos y nombres de constantes relacionadas	
Entorno	Valor
JMS herramienta de administración	CONEXIÓN ROTA
Programático	WMQCONSTANTS.ASYNC_EXCEPTIONS_CONNECTIONBROKEN = 1
IBM MQ Explorer	Conexión interrumpida

Se define la siguiente constante adicional:

- Desde IBM MQ 8.0.0 Fix Pack 2: WMQCONSTANTS.ASYNC_EXCEPTIONS_DEFAULT = ASYNC_EXCEPTIONS_CONNECTIONBROKEN
- Antes de IBM MQ 8.0.0 Fix Pack 2: WMQCONSTANTS.ASYNC_EXCEPTIONS_DEFAULT = ASYNC_EXCEPTIONS_ALL

Conceptos relacionados

[Excepciones en IBM MQ classes for JMS](#)

BALOPTIONS

Controla cómo se IBM MQ classes for JMS reequilibran las aplicaciones que utilizan el transporte de cliente en clústeres uniformes.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory.

JMS Nombre completo de la herramienta de administración: **BALOPTIONS**

JMS Nombre abreviado de la herramienta de administración: **OPTIONS**

Acceso programático

Métodos de lectura y escritura

- `MQConnectionFactory.setBalancingOptions()`
- `MQConnectionFactory.getBalancingOptions()`

Valores

IGNNONE

Se aplica el manejo normal de las transacciones, y no se solicita a las aplicaciones que se muevan durante una transacción.

Este valor se asigna a IBM MQ *BalancingOption* MQBNO_OPTIONS_NONE.

IGNTRANS

Es posible que se solicite a las aplicaciones que se trasladen durante una transacción.

Este valor se asigna a IBM MQ *BalancingOption* MQBNO_OPTIONS_IGNORE_TRANS.

BALTYPE

Controla cómo IBM MQ classes for JMS las aplicaciones que utilizan el transporte del cliente pueden reequilibrarse en un clúster uniforme.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory.

JMS Nombre completo de la herramienta de administración: **BALTYPE**

JMS Nombre abreviado de la herramienta de administración: **TYPE**

Acceso programático

Métodos de lectura y escritura

- `MQConnectionFactory.setBalancingApplicationType()`
- `MQConnectionFactory.getBalancingApplicationType()`

Valores

SIMPLE

Se aplica el manejo predeterminado de aplicaciones en un clúster uniforme.

Este valor se asigna a IBM `MQBalancingOption` `MQBNO_BALTYPE_SIMPLE`.

SOLICITUD_RESPUESTA

No se solicitará a la aplicación que se vuelva a conectar si un no **MQPUT** ha sido equilibrado por un **MQGET**, a menos que haya transcurrido el periodo de tiempo de espera.

Este valor se asigna a IBM `MQBalancingOption` `MQBNO_BALTYPE_REQREP`.

BALTIMEOUT

Controla cómo se IBM MQ classes for JMS reequilibran las aplicaciones que utilizan el transporte de cliente en un clúster uniforme.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory.

JMS Nombre completo de la herramienta de administración: **BALTIMEOUT**

JMS Nombre abreviado de la herramienta de administración: **TIMEOUT**

Acceso programático

Métodos de lectura y escritura

- `MQConnectionFactory.setBalancingTimeout()`
- `MQConnectionFactory.getBalancingTimeout()`

Valores

NEVER

La aplicación nunca agota el tiempo de espera con el fin de reequilibrar un clúster uniforme.

Este valor se asigna a IBM `MQBalancingOption` `MQBNO_TIMEOUT_NEVER`.

INMEDIATO

La aplicación agota inmediatamente el tiempo de espera con el fin de reequilibrar un clúster uniforme.

Este valor se asigna a IBM `MQBalancingOption` `MQBNO_TIMEOUT_IMMEDIATE`.

PREDETERMINADO

La aplicación agota el tiempo de espera para reequilibrar un clúster uniforme tras el periodo predeterminado de 10 segundos.

Este valor se asigna a IBM MQ *BalancingOption* MQBNO_TIMEOUT_AS_DEFAULT.

nn

La aplicación agota el tiempo de espera para reequilibrar un clúster uniforme tras un periodo de *nn* segundos.

nn puede estar entre 1 y 999999999.

BROKERCCDURSUBQ

El nombre de la cola de la que se recuperan los mensajes de suscripción duradera para un ConnectionConsumer.

Objetos aplicables

Tema

Nombre largo de la herramienta de administración de JMS : BROKERCCDURSUBQ

Nombre abreviado de la herramienta de administración de JMS : CCDSUB

Acceso programático

Métodos setter/getters

- MQTopic.setBrokerCCDurSubQueue()
- MQTopic.getBrokerCCDurSubQueue()

Valores

SYSTEM.JMS.D.CC.SUBSCRIBER.QUEUE

Éste es el valor predeterminado.

Cualquier serie válida

BROKERCCSUBQ

El nombre de la cola de la que se recuperan los mensajes de suscripción no duradera para un ConnectionConsumer.

Objetos aplicables

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : BROKERCCSUBQ

Nombre abreviado de la herramienta de administración de JMS : CCSUB

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setBrokerCCSubQueue()
- MQConnectionFactory.getBrokerCCSubQueue()

Valores

SYSTEM.JMS.ND.CC.SUBSCRIBER.QUEUE

Éste es el valor predeterminado.

Cualquier serie válida

BROKERCONQ

El nombre de cola de control del intermediario.

Objetos aplicables

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : BROKERCONQ

Nombre abreviado de la herramienta de administración de JMS : BCON

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setBrokerControlQueue()
- MQConnectionFactory.getBrokerControlQueue()

Valores

SYSTEM.BROKER.CONTROL.QUEUE

Éste es el valor predeterminado.

Cualquier serie válida

BROKERDURSUBQ

Cuando se IBM MQ classes for JMS utilizan en el modo de IBM MQ migración del proveedor de mensajería, esta propiedad especifica el nombre de la cola desde la que se recuperan los mensajes de suscripción duraderos.

Objetos aplicables

Tema

JMS Nombre completo de la herramienta de administración: BROKERDURSUBQ

JMS Nombre abreviado de la herramienta de administración: BDSUB

Acceso programático

Métodos de lectura y escritura

- MQTopic.setBrokerDurSubQueue()
- MQTopic.getBrokerDurSubQueue()

Valores

SYSTEM.JMS.D.SUBSCRIBER.QUEUE

Éste es el valor predeterminado.

Cualquier cadena válida

Empezando con SYSTEM.JMS.D

Tareas relacionadas

Configuración de la propiedad JMS [**PROVIDERVERSION**](#)

BROKERPUBQ

El nombre de la cola donde se envían los mensajes publicados (el valor de la corriente de datos).

Objetos aplicables

ConnectionFactory, TopicConnectionFactory, Tema, XAConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : BROKERPUBQ

Nombre abreviado de la herramienta de administración de JMS : BPUB

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setBrokerPubQueue
- MQConnectionFactory.getBrokerPubQueue

Valores

SYSTEM.BROKER.DEFAULT.STREAM

Éste es el valor predeterminado.

Cualquier serie válida

BROKERPUBQMGR

Nombre del gestor de colas al que pertenece la cola a la que se envían los mensajes publicados en el tema.

Objetos aplicables

Tema

Nombre largo de la herramienta de administración de JMS : BROKERPUBQMGR

Nombre abreviado de la herramienta de administración de JMS : BPQM

Acceso programático

Métodos setter/getters

- MQTopic.setBrokerPubQueueManager()
- MQTopic.getBrokerPubQueueManager()

Valores

null

Éste es el valor predeterminado.

Cualquier serie válida

BROKERQMGR

Nombre del gestor de colas en el que se ejecuta el intermediario.

Objetos aplicables

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : BROKERQMGR

Nombre abreviado de la herramienta de administración de JMS : BQM

Acceso programático

Métodos setter/getters

- `MQConnectionFactory.setBrokerQueueManager()`
- `MQConnectionFactory.getBrokerQueueManager()`

Valores

null

Éste es el valor predeterminado.

Cualquier serie válida

BROKERSUBQ

Cuando se IBM MQ classes for JMS utilizan en el modo de IBM MQ migración del proveedor de mensajería, esta propiedad especifica el nombre de la cola desde la que se recuperan los mensajes de suscripción no duraderos.

Objetos aplicables

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

JMS Nombre completo de la herramienta de administración: BROKERSUBQ

JMS Nombre abreviado de la herramienta de administración: BSUB

Acceso programático

Métodos de lectura y escritura

- `MQConnectionFactory.setBrokerSubQueue()`
- `MQConnectionFactory.getBrokerSubQueue()`

Valores

SYSTEM.JMS.ND.SUBSCRIBER.QUEUE

Éste es el valor predeterminado.

Cualquier cadena válida

Empezando con SYSTEM.JMS.ND

Tareas relacionadas

[Configuración de la propiedad JMS PROVIDERVERSION](#)

BROKERVER

La versión del intermediario que se está utilizando.

Objetos aplicables

ConnectionFactory, TopicConnectionFactory, Tema, XAConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : BROKERVER

Nombre abreviado de la herramienta de administración de JMS : BVER

Acceso programático

Métodos setter/getters

- `MQConnectionFactory.setBrokerVersion()`

- MQConnectionFactory.getBrokerVersion()

Valores

V1

Para utilizar un intermediario de publicación/suscripción de IBM MQ , o para utilizar un intermediario de IBM MQ Integrator, WebSphere Event Broker, WebSphere Business Integration Event Broker o WebSphere Business Integration Message Broker en modalidad de compatibilidad. Este es el valor predeterminado si TRANSPORT se establece en BIND o CLIENT.

V2

Para utilizar un intermediario de IBM MQ Integrator, WebSphere Event Broker, WebSphere Business Integration Event Broker o WebSphere Business Integration Message Broker en modalidad nativa. Este es el valor predeterminado si TRANSPORT se establece en DIRECT o DIRECTHTTP.

no especificada

Después de que el intermediario haya migrado de V6 a V7, establezca esta propiedad para que las cabeceras RFH2 ya no se utilicen. Después de la migración, esta propiedad ya no es relevante.

CCDTURL

Un localizador uniforme de recursos (URL) que identifica el nombre y la ubicación del archivo que contiene la tabla de definición de canal de cliente y especifica cómo se puede acceder al archivo.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS Nombre completo de la herramienta de administración: CCDTURL

JMS Nombre abreviado de la herramienta de administración: CCDT

Acceso programático

Métodos de lectura y escritura

- MQConnectionFactory.setCCDTURL()
- MQConnectionFactory.getCCDTURL()

Valores

null

Éste es el valor predeterminado.

Un localizador universal de recursos (URL)

Nota: v 10.0.0 Desde IBM MQ 10.0.0, puede añadir su configuración personalizada de SSLSocketFactory con la **WMQ_CCDT_SSL_SOCKET_FACTORY** propiedad, lo que permite a las aplicaciones gestionar la seguridad de la conexión HTTPS y la validación de certificados a través de su propia implementación. Si una aplicación cliente proporciona la **WMQ_CCDT_SSL_SOCKET_FACTORY** propiedad, entonces la **CCDTURL** propiedad se convierte en obligatoria y, si falta, se devuelve un mensaje de error a la aplicación cliente. Para obtener más información, consulte [Suministro de un SSLSocketFactory personalizado en IBM MQ classes for JMS/Jakarta Messaging](#).

CCSID

Para las fábricas de conexiones, esta propiedad especifica el identificador del juego de caracteres codificado (CCSID) que se utilizará para los flujos de datos internos con el gestor de colas. El CCSID controla cómo se codifican los datos en el flujo de comunicación hacia el gestor de colas para los tipos de datos de caracteres, como el nombre del canal y el nombre del gestor de colas. Dado que estos flujos utilizan estructuras de datos de ancho fijo, el valor CCSID especificado debe ser un conjunto

de caracteres de un solo byte para poder utilizar las longitudes de campo máximas permitidas. Para los destinos, la propiedad define el CCSID que se utilizará para codificar los datos de cadena en MapMessages, StreamMessages, y TextMessages enviados a ese destino.

Nota: Normalmente no es necesario cambiar esta propiedad para las fábricas de conexiones.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, Cola, Tema, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS Nombre completo de la herramienta de administración: CCSID

JMS Nombre abreviado de la herramienta de administración: CCS

Acceso programático

Métodos de lectura y escritura

- MQConnectionFactory.setCCSID()
- MQConnectionFactory.getCCSID()

Valores

819

El valor predeterminado para una fábrica de conexiones.

1208

El valor predeterminado para un destino.

Cualquier entero positivo

Conceptos relacionados

[Conversión de mensajes JMS](#)

CERTVALPO

La política de validación de certificados que se utilizará para una conexión TLS.

Objetos aplicables

ConnectionFactory

JMS Nombre completo de la herramienta administrativa: CERTVALPO

JMS Nombre abreviado de la herramienta administrativa: CVAP

Acceso programático

Métodos de lectura y escritura

- MQConnectionFactory.setCertificateValPolicy()
- MQConnectionFactory.getCertificateValPolicy()

Valores

0/ANY

Este es el valor predeterminado para una fábrica de conexiones.

2 / NINGUNO

Esta política no incluye la validación de certificados. Solo se puede configurar en aplicaciones cliente.

Información relacionada

[Configuración de políticas de validación de certificados en IBM MQ](#)

CHANNEL

El nombre del canal de conexión de cliente que se está utilizando.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : CHANNEL

Nombre abreviado de la herramienta de administración de JMS : CHAN

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setChannel()
- MQConnectionFactory.getChannel()

Valores

SYSTEM.DEF.SVRCONN

Éste es el valor predeterminado.

Cualquier serie válida

CLEANUP

Nivel de limpieza para almacenes de suscripciones BROKER o MIGRATE.

Objetos aplicables

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : CLEANUP

Nombre abreviado de la herramienta de administración de JMS : CL

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setCleanupLevel()
- MQConnectionFactory.getCleanupLevel()

Valores

SAFE

Utilice una limpieza segura. Éste es el valor predeterminado.

ASPROP

Utilice una limpieza segura, fuerte o nula según una propiedad establecida en la línea de mandatos de Java .

Ninguna

No utilizar limpieza.

STRONG

Utilice una limpieza fuerte.

CLEANUPINT

El intervalo, en milisegundos, entre ejecuciones en segundo plano del programa de utilidad de limpieza de publicación/suscripción.

Objetos aplicables

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : CLEANUPINT

Nombre abreviado de la herramienta de administración de JMS : CLINT

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setCleanupInterval()
- MQConnectionFactory.getCleanupInterval()

Valores

3600000

Éste es el valor predeterminado.

Cualquier entero positivo

ConnectionNameList

Lista de nombres de conexión de « TCP/IP ». La lista se intenta en orden, una vez por cada reintento de reconexión.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : CONNECTIONNAMELIST

Nombre abreviado de la herramienta de administración de JMS : CNLIST

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setconnectionNameList()
- MQConnectionFactory.getconnectionNameList()

Valores

Lista separada por comas de HOSTNAME (PORT). HOSTNAME puede ser un nombre DNS o una dirección IP.

PORT toma el valor predeterminado de 1414.

CLIENTRECONNECTOPTIONS

Opciones que rigen la reconexión.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : CLIENTRECONNECTOPTIONS

Nombre abreviado de la herramienta de administración de JMS : CROPT

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setClientReconnectOptions()
- MQConnectionFactory.getClientReconnectOptions()

Valores

QMGR

La aplicación puede volver a conectarse al mismo gestor de colas al que está conectada originalmente.

Se devuelve un error con el código de razón MQRC_RECONNECT_QMID_MISMATCH si el gestor de colas al que la aplicación intenta conectarse, tal como se especifica en la lista de nombres de conexión, tiene un QMID diferente al gestor de colas al que se conectó originalmente.

Utilice este valor si una aplicación se puede reconectar, pero existe una afinidad entre la aplicación IBM MQ classes for JMS y el gestor de colas con el que estableció por primera vez una conexión.

Elija este valor si desea que una aplicación se vuelva a conectar automáticamente a la instancia en espera de un gestor de colas de alta disponibilidad.

Para utilizar este valor mediante programación, utilice la constante WMQConstants.WMQ_CLIENT_RECONNECT_Q_MGR.

CUALQUIERA

La aplicación puede volver a conectarse a cualquiera de los gestores de colas especificados en la lista de nombres de conexión.

Utilice la opción de reconexión sólo si no hay afinidad entre las clases IBM MQ para la aplicación JMS y el gestor de colas con el que estableció inicialmente una conexión.

Para utilizar este valor de un programa, utilice la constante WMQConstants.WMQ_CLIENT_RECONNECT.

DISABLED

La aplicación no se reconectará.

Para utilizar este valor mediante programación, utilice la constante WMQConstants.WMQ_CLIENT_RECONNECT_DISABLED.

ASDEF

Si la aplicación se volverá a conectar automáticamente depende del valor del atributo de canal de IBM MQ DefReconnect.

Éste es el valor predeterminado.

Para utilizar este valor de un programa, utilice la constante WMQConstants.WMQ_CLIENT_RECONNECT_AS_DEF.

CLIENTRECONNECTTIMEOUT

Tiempo antes de que cesen los reintentos de reconexión.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : CLIENTRECONNECTTIMEOUT

Nombre abreviado de la herramienta de administración de JMS : CRT

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setClientReconnectTimeout()
- MQConnectionFactory.setClientReconnectTimeout()

Valores

Intervalo en segundos. Valor predeterminado 1800 (30 minutos).

CLIENTID

El identificador de cliente se utiliza para identificar de forma exclusiva la conexión de aplicación para las suscripciones duraderas.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : CLIENTID

Nombre abreviado de la herramienta de administración de JMS : CID

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setClientId()
- MQConnectionFactory.getClientId()

Valores

null

Éste es el valor predeterminado.

Cualquier serie válida

CLONESUPP

Determina si dos o más instancias del mismo suscriptor de tema duradero se pueden ejecutar de forma simultánea.

Objetos aplicables

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : CLONESUPP

Nombre abreviado de la herramienta de administración de JMS : CLS

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setCloneSupport()
- MQConnectionFactory.getCloneSupport()

Valores

DISABLED

Sólo se puede ejecutar una instancia de un suscriptor de tema duradero a la vez. Éste es el valor predeterminado.

ENABLED

Pueden ejecutarse simultáneamente dos o más instancias del mismo suscriptor de temas duraderos, pero cada instancia debe ejecutarse en una máquina virtual independiente Java (JVM).

COMPHDR

Lista de las técnicas que se pueden utilizar para comprimir datos de cabecera en una conexión.

Objetos aplicables

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : COMPHDR

Nombre abreviado de la herramienta de administración de JMS : HC

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setHdrCompList()
- MQConnectionFactory.getHdrCompList()

Valores

Ninguna

Éste es el valor predeterminado.

SISTEMA

Se realiza la compresión de cabecera de mensaje RLE.

COMPMSG

Lista de las técnicas que se pueden utilizar para comprimir datos de mensaje en una conexión.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS Nombre completo de la herramienta de administración: COMPMSG

JMS Nombre abreviado de la herramienta de administración: MC

Acceso programático

Métodos de lectura y escritura

- MQConnectionFactory.setMsgCompList()
- MQConnectionFactory.getMsgCompList()

Valores

Ninguna

Éste es el valor predeterminado.

Una lista de uno o varios de los valores siguientes separados por caracteres en blanco:

RLE ZLIBFAST ZLIBHIGH LZ4FAST LZ4HIGH

CONNOPT

Controla cómo se conectan al gestor de colas las aplicaciones IBM MQ classes for JMS que utilizan el transporte de enlaces.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory.

Nombre largo de la herramienta de administración de JMS : CONNOPT

Nombre abreviado de la herramienta de administración de JMS : CNOPT

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setMQConnectionFactoryOptions()
- MQConnectionFactory.getMQConnectionFactoryOptions()

Valores

ESTÁNDAR

La naturaleza del enlace entre la aplicación y el gestor de colas depende del valor del atributo *DefaultBindType* del gestor de colas. El valor STANDARD se correlaciona con IBM MQ *ConnectOption* MQCNO_STANDARD_BINDING.

SHARED

La aplicación y el agente del gestor de colas local se ejecutan en unidades de ejecución independientes pero comparten algunos recursos. Este valor se correlaciona con IBM MQ *ConnectOption* MQCNO_SHARED_BINDING.

Aislado

La aplicación y el agente del gestor de colas local se ejecutan en unidades de ejecución independientes y no comparten recursos. El valor AISLADO se correlaciona con IBM MQ *ConnectOption* MQCNO_ISOLATED_BINDING.

FASTPATH

La aplicación y el agente del gestor de colas local se ejecutan en la misma unidad de ejecución. Este valor se correlaciona con IBM MQ *ConnectOption* MQCNO_FASTPATH_BINDING.

SERIALQM

La aplicación solicita el uso exclusivo del código de conexión dentro del ámbito del gestor de colas. Este valor se correlaciona con IBM MQ *ConnectOption* MQCNO_SERIALIZE_CONN_TAG_Q_MGR.

SERIALQSG

La aplicación solicita el uso exclusivo del código de conexión dentro del ámbito del grupo de compartición de colas al que pertenece el gestor de colas. El valor SERIALQSG se correlaciona con IBM MQ *ConnectOption* MQCNO_SERIALIZE_CONN_TAG_QSG.

RESTRICTQM

La aplicación solicita el uso compartido del código de conexión, pero hay restricciones en el uso compartido del código de conexión dentro del ámbito del gestor de colas. Este valor se correlaciona con IBM MQ *ConnectOption* MQCNO_RESTRICT_CONN_TAG_Q_MGR.

RESTRICTQSG

La aplicación solicita el uso compartido del código de conexión, pero existen restricciones sobre el uso compartido del código de conexión dentro del ámbito del grupo de compartición de colas al que pertenece el gestor de colas. Este valor se correlaciona con IBM MQ *ConnectOption* MQCNO_RESTRICT_CONN_TAG_QSG.

Para obtener más información sobre las opciones de conexión de IBM MQ , consulte [Conexión a un gestor de colas utilizando la llamada MQCONN](#).

CONNTAG

Código que el gestor de colas asocia con los recursos actualizados por la aplicación dentro de una unidad de trabajo mientras la aplicación está conectada al gestor de colas.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : CONNTAG

Nombre abreviado de la herramienta de administración de JMS : CNTAG

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setConnTag()
- MQConnectionFactory.getConnTag()

Valores

Una matriz de bytes de 128 elementos, donde cada elemento es 0

Éste es el valor predeterminado.

Cualquier serie

El valor se trunca si tiene más de 128 bytes.

DESCRIPCIÓN

Una descripción del objeto almacenado.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, Cola, Tema, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : DESCRIPTION

Nombre abreviado de la herramienta de administración de JMS : DESC

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setDescription()
- MQConnectionFactory.getDescription()

Valores

null

Éste es el valor predeterminado.

Cualquier serie válida

DIRECTAUTH

Si se utiliza la autenticación de tipo « TLS » en una conexión en tiempo real con un bróker.

Objetos aplicables

ConnectionFactory, TopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : DIRECTAUTH

Nombre abreviado de la herramienta de administración de JMS : DAUTH

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setDirectAuth()
- MQConnectionFactory.getDirectAuth()

Valores

BASIC

Sin autenticación, autenticación de nombre de usuario o autenticación de contraseña. Éste es el valor predeterminado.

CERTIFICATE

Autenticación de certificado de clave pública.

ENCODING

Cómo se representan los datos numéricos en el cuerpo de un mensaje cuando se envía el mensaje a este destino. La propiedad especifica la representación de enteros binarios, enteros decimales empaquetados y números de coma flotante.

Objetos aplicables

Cola, Tema

Nombre largo de la herramienta de administración de JMS : ENCODING

Nombre abreviado de la herramienta de administración de JMS : ENC

Acceso programático

Métodos setter/getters

- MQDestination.setEncoding()
- MQDestination.getEncoding()

Valores

La propiedad de codificación

Los valores válidos que puede tomar la propiedad ENCODING se construyen a partir de las tres subpropiedades:

Codificación de enteros

Normal o invertido

Codificación decimal

Normal o invertido

codificación de coma flotante

IEEE normal, IEEE invertido o z/OS

La propiedad `ENCODING` se expresa como una serie de tres caracteres con la sintaxis siguiente:

```
{N|R}{N|R}{N|R|3}
```

En esta serie:

- N indica normal
- R indica invertido
- 3 denota z/OS
- El primer carácter representa la *codificación de enteros*
- El segundo carácter representa la *codificación decimal*
- El tercer carácter representa la *codificación de coma flotante*

Esto proporciona un conjunto de doce valores posibles para la propiedad `ENCODING`.

Hay un valor adicional, la serie `NATIVE`, que establece los valores de codificación adecuados para la plataforma Java.

Los ejemplos siguientes muestran combinaciones válidas para `ENCODING`:

```
ENCODING (NNR)  
ENCODING (NATIVE)  
ENCODING (RR3)
```

EXPIRY

La hora después de la cual caducan los mensajes en un destino.

Objetos aplicables

Cola, Tema

Nombre largo de la herramienta de administración de JMS : `EXPIRE`

Nombre abreviado de la herramienta de administración de JMS : `EXP`

Acceso programático

Métodos setter/getters

- `MQDestination.setExpiry()`
- `MQDestination.getExpiry()`

Valores

APP

La aplicación JMS puede definir la caducidad. Éste es el valor predeterminado.

UNLIM

No se produce ninguna caducidad.

0

No se produce ninguna caducidad.

Cualquier entero positivo que represente la caducidad en milisegundos.

FAILIFQUIESCE

Esta propiedad determina si las llamadas a determinados métodos fallan si el gestor de colas está en un estado de inmovilización, o si una aplicación se está conectando a un gestor de colas utilizando el transporte `CLIENT` y el canal que la aplicación está utilizando se ha puesto en un estado de

inmovilización, por ejemplo, utilizando el mandato **STOP CHANNEL** o **STOP CHANNEL MODE(QUIESCE)** MQSC.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, Cola, Tema, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : FAILIFQUIESCE

Nombre abreviado de la herramienta de administración de JMS : FIQ

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setFailIfQuiesce()
- MQConnectionFactory.getFailIfQuiesce()

Valores

sí

Las llamadas a determinados métodos fallan si el gestor de colas está en un estado de desactivación temporal o si el canal que se utiliza para conectarse a un gestor de colas está desactivado temporalmente. Si una aplicación detecta cualquiera de estas condiciones, la aplicación puede completar su tarea inmediata y cerrar la conexión, permitiendo que se detenga el gestor de colas o la instancia de canal. Éste es el valor predeterminado.

NO

Ninguna llamada de método falla porque el gestor de colas, o el canal que se está utilizando para conectarse a un gestor de colas, está en un estado de desactivación temporal. Si especifica este valor, una aplicación no puede detectar que el gestor de colas o el canal se está desactivando temporalmente. La aplicación puede continuar realizando operaciones en el gestor de colas y, por lo tanto, impedir que se detenga el gestor de colas.

HOSTNAME

Para una conexión con un gestor de colas, el nombre de host o la dirección IP del sistema en el que se ejecuta el gestor de colas o, para una conexión en tiempo real con un intermediario, el nombre de host o la dirección IP del sistema en el que se ejecuta el intermediario.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : HOSTNAME

Nombre abreviado de la herramienta de administración de JMS : HOST

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setHostName()
- MQConnectionFactory.getHostName()

Valores

localhost

Éste es el valor predeterminado.

Cualquier serie válida

LOCALADDRESS

Para una conexión con un gestor de colas, esta propiedad especifica la interfaz de red local que se va a utilizar, o el puerto local o el rango de puertos locales que se va a utilizar.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : LOCALADDRESS

Nombre abreviado de la herramienta de administración de JMS : LA

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setLocalAddress()
- MQConnectionFactory.getLocalAddress()

Valores

"" (serie vacía)

Éste es el valor predeterminado.

Una serie con el formato [ip-addr] [(low-port [, high port])]

A continuación, se detallan algunos ejemplos:

192.0.2.0

El canal se enlaza a la dirección 192.0.2.0 localmente.

192.0.2.0 (1000)

El canal se enlaza a la dirección 192.0.2.0 localmente y utiliza el puerto 1000.

192.0.2.0 (1000,2000)

El canal se enlaza a la dirección 192.0.2.0 localmente y utiliza un puerto en el rango de 1000 a 2000.
(1000)

El canal se enlaza al puerto 1000 localmente.

(1000,2000)

El canal se enlaza a un puerto en el rango de 1000 a 2000 localmente.

Puede especificar un nombre de host en lugar de una dirección IP. Para una conexión en tiempo real con un intermediario, esta propiedad sólo es relevante cuando se utiliza la multidifusión, y el valor de la propiedad no debe contener un número de puerto o un rango de números de puerto. Los únicos valores válidos de la propiedad en este caso son null, una dirección IP o un nombre de host.

ESTILO de nombre de mapa

Permite utilizar el estilo de compatibilidad para los nombres de elemento MapMessage .

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : MAPNAMESTYLE

Nombre abreviado de la herramienta de administración de JMS : MNST

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setMapNameStyle()
- MQConnectionFactory.getMapNameStyle()

Valores

ESTÁNDAR

Se debe utilizar el formato estándar de nomenclatura de elementos com.ibm.jms.JMSMapMessage. Este es el valor predeterminado y permite que se utilicen identificadores Java no legales como nombre de elemento.

Compatible

Se debe utilizar el formato de denominación de elementos MapMessage de com.ibm.jms.JMS más antiguo. Sólo se pueden utilizar identificadores Java legales como nombre de elemento. Esto sólo es necesario si los mensajes de mapa se envían a una aplicación que utiliza una versión de IBM MQ classes for JMS anterior a 5.3.

MAXBUFFSIZE

Número máximo de mensajes recibidos que se pueden almacenar en un almacenamiento intermedio de mensajes interno mientras se espera a que la aplicación los procese. Esta propiedad sólo se aplica cuando TRANSPORT tiene el valor DIRECT o DIRECTHTTP.

Objetos aplicables

ConnectionFactory, TopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : MAXBUFFSIZE

Nombre abreviado de la herramienta de administración de JMS : MBSZ

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setMaxBufferSize()
- MQConnectionFactory.getMaxBufferSize()

Valores

1000

Éste es el valor predeterminado.

Cualquier entero positivo

MDREAD

Esta propiedad determina si una aplicación JMS puede extraer los valores de los campos MQMD.

Objetos aplicables

Nombre largo de la herramienta de administración de JMS : MDREAD

Nombre abreviado de la herramienta de administración de JMS : MDR

Acceso programático

Métodos setter/getters

- MQDestination.setMQMDReadEnabled()
- MQDestination.getMQMDReadEnabled()

Valores

NO

Al enviar mensajes, las propiedades JMS_IBM_MQMD* en un mensaje enviado no se actualizan para reflejar los valores de campo actualizados en el MQMD. Al recibir mensajes, ninguna de las propiedades JMS_IBM_MQMD* está disponible para un mensaje recibido, incluso si el emisor ha establecido todas o algunas de ellas. Este es el valor predeterminado para las herramientas administrativas.

Para los programas, utilice False.

Sí

Al enviar mensajes, todas las propiedades JMS_IBM_MQMD* de un mensaje enviado se actualizan para reflejar los valores de campo actualizados en MQMD, incluidas las propiedades que el remitente no ha establecido explícitamente. Al recibir mensajes, todas las propiedades JMS_IBM_MQMD* están disponibles en un mensaje recibido, incluidas las propiedades que el remitente no ha establecido explícitamente.

Para los programas, utilice True.

MDWRITE

Esta propiedad determina si una aplicación JMS puede establecer los valores de los campos MQMD.

Objetos aplicables

Cola, Tema

Nombre largo de la herramienta de administración de JMS : MDWRITE

Nombre abreviado de la herramienta de administración de JMS : MDR

Acceso programático

Métodos setter/getters

- MQDestination.setMQMDWriteEnabled()
- MQDestination.getMQMDWriteEnabled()

Valores

NO

Todas las propiedades JMS_IBM_MQMD* se ignoran y sus valores no se copian en la estructura MQMD subyacente. Este es el valor predeterminado para las herramientas administrativas.

Para los programas, utilice False.

sí

Se procesan las propiedades JMS_IBM_MQMD*. Sus valores se copian en la estructura MQMD subyacente.

Para los programas, utilice True.

MDMSGCTX

Determina el nivel de contexto de mensaje que debe ser establecido por la aplicación de JMS. La aplicación debe estar en ejecución con la autoridad de contexto adecuada para que esta propiedad tenga efecto.

Objetos aplicables

Nombre largo de la herramienta de administración de JMS : MDMSGCTX

Nombre abreviado de la herramienta de administración de JMS : MDCTX

Acceso programático

Métodos setter/getters

- MQDestination.setMQMDMessageContext()
- MQDestination.getMQMDMessageContext()

Valores

PREDETERMINADO

La llamada de API MQOPEN y la estructura MQPMO no especifican opciones de contexto de mensaje explícitas. Este es el valor predeterminado para las herramientas administrativas.

Para los programas, utilice WMQ_MDCTX_DEFAULT.

SET_IDENTITY_CONTEXT

La llamada de la API MQOPEN especifica la opción de contexto de mensaje MQOO_SET_IDENTITY_CONTEXT y la estructura MQPMO especifica MQPMO_SET_IDENTITY_CONTEXT.

Para los programas, utilice WMQ_MDCTX_SET_IDENTITY_CONTEXT.

SET_ALL_CONTEXT

La llamada de la API MQOPEN especifica la opción de contexto de mensaje MQOO_SET_ALL_CONTEXT y la estructura MQPMO especifica MQPMO_SET_ALL_CONTEXT.

Para programas, utilice WMQ_MDCTX_SET_ALL_CONTEXT.

MSGBATCHSZ

Número máximo de mensajes que se deben tomar de una cola en un paquete cuando se utiliza la entrega de mensajes asíncrona.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : MAXBUFFSIZE

Nombre abreviado de la herramienta de administración de JMS : MBSZ

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setMsgBatchSize()
- MQConnectionFactory.getMsgBatchSize()

Valores

10

Éste es el valor predeterminado.

Cualquier entero positivo

MSGBODY

Determina si una aplicación JMS accede a la MQRFH2 de un mensaje IBM MQ como parte de la carga útil del mensaje.

Objetos aplicables

Cola, Tema

Nombre largo de la herramienta de administración de JMS : WMQ_MESSAGE_BODY

Nombre abreviado de la herramienta de administración de JMS : MBODY

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setMessageBodyStyle()
- MQConnectionFactory.getMessageBodyStyle()

Valores

SIN ESPECIFICAR

En el envío, determina si IBM MQ classes for JMS genera e incluye una cabecera MQRFH2, dependiendo del valor de WMQ_TARGET_CLIENT. Al recibir, actúa como valor JMS.

JMS

En el envío, IBM MQ classes for JMS genera automáticamente una cabecera MQRFH2 y la incluye en el mensaje de IBM MQ.

En la recepción, IBM MQ classes for JMS establece las propiedades de mensajes de JMS de acuerdo con los valores contenidos en la cabecera MQRFH2 (si existe); no presenta la cabecera MQRFH2 como parte del cuerpo del mensaje JMS.

MQ

En el envío, IBM MQ classes for JMS no genera una cabecera MQRFH2.

En la recepción, IBM MQ classes for JMS presenta MQRFH2 como parte del cuerpo del mensaje de JMS.

MSGRETENTION

Indica si el consumidor de conexión mantiene los mensajes no entregados en la cola de entrada.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory,

Nombre largo de la herramienta de administración de JMS : MSGRETENTION

Nombre abreviado de la herramienta de administración de JMS : MRET

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setMessageRetention()

- `MQConnectionFactory.getMessageRetention()`

Valores

Sí

Los mensajes no entregados permanecen en la cola de entrada. Éste es el valor predeterminado.

No

Los mensajes no entregados se tratan de acuerdo con sus opciones de disposición.

MSGSELECTION

Determina si la selección de mensajes la realiza el IBM MQ classes for JMS o el intermediario. Si `TRANSPORT` tiene el valor `DIRECT`, el intermediario siempre realiza la selección de mensajes y se ignora el valor de `MSGSELECTION`. La selección de mensajes por parte del intermediario no está soportada cuando `BROKERVER` tiene el valor `V1`.

Objetos aplicables

`ConnectionFactory`, `TopicConnectionFactory`, `XAConnectionFactory`, `XATopicConnectionFactory`

Nombre largo de la herramienta de administración de JMS : `MSGSELECTION`

Nombre abreviado de la herramienta de administración de JMS : `MSEL`

Acceso programático

Métodos setter/getters

- `MQConnectionFactory.setMessageSelection()`
- `MQConnectionFactory.getMessageSelection()`

Valores

CLIENTE

La selección de mensajes la realiza IBM MQ classes for JMS. Éste es el valor predeterminado.

BROKER

La selección de mensajes ha sido realizada por el intermediario.

MULTICAST

Para habilitar la multidifusión en una conexión en tiempo real con un intermediario y, si está habilitada, para especificar la forma precisa en la que se utiliza la multidifusión para entregar mensajes del intermediario a un consumidor de mensajes. La propiedad no tiene ningún efecto sobre cómo un productor de mensajes envía mensajes a un intermediario.

Objetos aplicables

`ConnectionFactory`, `TopicConnectionFactory`, Tema

Nombre largo de la herramienta de administración de JMS : `MULTICAST`

Nombre abreviado de la herramienta de administración de JMS : `MCAST`

Acceso programático

Métodos setter/getters

- `MQConnectionFactory.setMulticast()`
- `MQConnectionFactory.getMulticast()`

Valores

DISABLED

Los mensajes no se entregan a un consumidor de mensajes utilizando el transporte de multidifusión. Este es el valor por defecto para los objetos `ConnectionFactory` y `TopicConnectionFactory`.

ASCF

Los mensajes se entregan a un consumidor de mensajes de acuerdo con el valor de multidifusión para la fábrica de conexiones asociada al consumidor de mensajes. El valor de multidifusión para la fábrica de conexiones se anota en el momento en que se crea el consumidor de mensajes. Este valor sólo es válido para los objetos de tema y es el valor predeterminado para los objetos de tema.

ENABLED

Si el tema está configurado para multidifusión en el intermediario, los mensajes se entregan a un consumidor de mensajes utilizando el transporte de multidifusión. Se utiliza una calidad de servicio fiable si el tema está configurado para la multidifusión fiable.

PROVEEDOR

Si el tema está configurado para multidifusión fiable en el intermediario, los mensajes se entregan al consumidor de mensajes utilizando el transporte de multidifusión con una calidad de servicio fiable. Si el tema no está configurado para la multidifusión fiable, no puede crear un consumidor de mensajes para el tema.

NOTR

Si el tema está configurado para multidifusión en el intermediario, los mensajes se entregan al consumidor de mensajes utilizando el transporte de multidifusión. No se utiliza una calidad de servicio fiable, aunque el tema se haya configurado para la multidifusión fiable.

OPTIMISTICPUBLICATION

Esta propiedad determina si IBM MQ classes for JMS devuelve el control inmediatamente a un publicador que ha publicado un mensaje, o si devuelve el control sólo después de que haya completado todo el proceso asociado con la llamada y pueda informar del resultado al publicador.

Objetos aplicables

`ConnectionFactory`, `TopicConnectionFactory`

Nombre largo de la herramienta de administración de JMS : `OPTIMISTICPUBLICATION`

Nombre abreviado de la herramienta de administración de JMS : `OPTPUB`

Acceso programático

Métodos setter/getters

- `MQConnectionFactory.setOptimisticPublication()`
- `MQConnectionFactory.getOptimisticPublication()`

Valores

NO

Cuando un publicador publica un mensaje, IBM MQ classes for JMS no devuelve el control al publicador hasta que haya completado todo el proceso asociado a la llamada y pueda informar del resultado al publicador. Éste es el valor predeterminado.

SÍ

Cuando un publicador publica un mensaje, IBM MQ classes for JMS devuelve el control al publicador inmediatamente, antes de que haya completado todo el proceso asociado a la llamada y pueda informar del resultado al publicador. IBM MQ classes for JMS informa del resultado sólo cuando el publicador confirma el mensaje.

OUTCOMENOTIFICATION

Esta propiedad determina si IBM MQ classes for JMS devuelve el control inmediatamente a un suscriptor que acaba de reconocer o confirmar un mensaje, o si devuelve el control sólo después de que haya completado todo el proceso asociado con la llamada y pueda informar del resultado al suscriptor.

Objetos aplicables

ConnectionFactory, TopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : OUTCOMENOTIFICACIÓN

Nombre abreviado de la herramienta de administración de JMS : NOTIFY

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setOutcomeNotification()
- MQConnectionFactory.getOutcomeNotification()

Valores

sí

Cuando un suscriptor reconoce o confirma un mensaje, IBM MQ classes for JMS no devuelve el control al suscriptor hasta que haya completado todo el proceso asociado con la llamada y pueda informar del resultado al suscriptor. Éste es el valor predeterminado.

NO

Cuando un suscriptor reconoce o confirma un mensaje, IBM MQ classes for JMS devuelve el control al suscriptor inmediatamente, antes de que haya completado todo el proceso asociado con la llamada y pueda informar del resultado al suscriptor.

PERSISTENCE

Persistencia de los mensajes enviados a un destino.

Objetos aplicables

Cola, Tema

Nombre largo de la herramienta de administración de JMS : PERSISTENCE

Nombre abreviado de la herramienta de administración de JMS : PER

Acceso programático

Métodos setter/getters

- MQDestination.setPersistence()
- MQDestination.getPersistence()

Valores

APP

La persistencia la define la aplicación JMS . Éste es el valor predeterminado.

qdef

La persistencia toma el valor predeterminado de la cola.

pers

Los mensajes son persistentes.

no

Los mensajes no son persistentes.

HIGH

Consulte [Mensajes persistentes deJMS](#) para obtener más información sobre el uso de este valor.

POLLINGINT

Si cada escucha de mensajes dentro de una sesión no tiene ningún mensaje adecuado en su cola, este es el intervalo máximo, en milisegundos, que transcurre antes de que cada escucha de mensajes intente de nuevo obtener un mensaje de su cola. Si con frecuencia sucede esto de que no haya disponible ningún mensaje adecuado para ninguno de los escuchas de mensajes de una sesión, considere aumentar el valor de esta propiedad. Esta propiedad sólo tiene relevancia si **TRANSPORT** tiene el valor **BIND** o **CLIENT**.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : **POLLINGINT**

Nombre abreviado de la herramienta de administración de JMS : **PINT**

Acceso programático

Métodos setter/getters

- `MQConnectionFactory.setPollingInterval()`
- `MQConnectionFactory.getPollingInterval()`

Valores

5000

Éste es el valor predeterminado.

Cualquier entero positivo

Puerto

Para una conexión con un gestor de colas, el número del puerto en el que el gestor de colas está a la escucha o, para una conexión en tiempo real con un intermediario, el número del puerto en el que el intermediario está a la escucha de conexiones en tiempo real.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : **PORT**

Nombre abreviado de la herramienta de administración de JMS : **PORT**

Acceso programático

Métodos setter/getters

- `MQConnectionFactory.setPort()`
- `MQConnectionFactory.getPort()`

Valores

1414

Este es el valor predeterminado si `TRANSPORT` se establece en `CLIENT`.

1506

Este es el valor predeterminado si `TRANSPORT` se establece en `DIRECT` o `DIRECTHTTP`.

Cualquier entero positivo

PRIORITY

Prioridad de los mensajes enviados a un destino.

Objetos aplicables

Cola, Tema

Nombre largo de la herramienta de administración de JMS : `PRIORITY`

Nombre abreviado de la herramienta de administración de JMS : `PRI`

Acceso programático

Métodos setter/getters

- `MQDestination.setPriority()`
- `MQDestination.getPriority()`

Valores

APP

La prioridad la define la aplicación JMS . Éste es el valor predeterminado.

qdef

La prioridad toma el valor predeterminado de la cola.

Cualquier entero en el rango 0-9

De menor a mayor.

PROCESSDURATION

Esta propiedad determina si un suscriptor garantiza procesar rápidamente cualquier mensaje que reciba antes de devolver el control a IBM MQ classes for JMS.

Objetos aplicables

`ConnectionFactory`, `TopicConnectionFactory`

Nombre largo de la herramienta de administración de JMS : `PROCESSDURATION`

Nombre abreviado de la herramienta de administración de JMS : `PROC DUR`

Acceso programático

Métodos setter/getters

- `MQConnectionFactory.setProcessDuration()`
- `MQConnectionFactory.getProcessDuration()`

Valores

DESCONOCIDO

Un suscriptor no puede dar ninguna garantía sobre la rapidez con la que puede procesar cualquier mensaje que reciba. Éste es el valor predeterminado.

SHORT

Un suscriptor garantiza procesar rápidamente cualquier mensaje que reciba antes de devolver el control a IBM MQ classes for JMS.

PROVIDERVERSION

Esta propiedad diferencia entre las tres modalidades de operación de mensajería de IBM MQ : modalidad normal de proveedor de mensajería de IBM MQ , modalidad normal de proveedor de mensajería de IBM MQ con restricciones y modalidad de migración de proveedor de mensajería de IBM MQ .

La modalidad normal del proveedor de mensajería de IBM MQ utiliza todas las características de un gestor de colas de IBM MQ para implementar JMS. Esta modalidad está optimizada para utilizar la API y la funcionalidad de JMS 2.0. La modalidad normal con restricciones del proveedor de mensajería de IBM MQ utiliza la API JMS 2.0 , pero no las nuevas características como, por ejemplo, suscripciones compartidas, entrega retardada o envío asíncrono.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : PROVIDERVERSION

Nombre abreviado de la herramienta de administración de JMS : PVER

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setProviderVersion()
- MQConnectionFactory.getProviderVersion()

Valores

Puede establecer la propiedad **PROVIDERVERSION** en cualquiera de los valores 8 (modalidad normal), 7 (modalidad normal con restricciones), 6 (modalidad de migración) o unspecified (el valor predeterminado). El valor que especifique para la propiedad **PROVIDERVERSION** debe ser una serie. Si va a especificar la opción 8, 7 o 6, puede hacerlo en cualquiera de los siguientes formatos:

- V.R.M.F
- V.R.M
- V.R
- V

donde V, R, M y F son valores enteros mayores que o iguales a cero. Los valores R, M y F adicionales son opcionales y están disponibles para su uso en caso de que se necesite un control de grano fino. Por ejemplo, si desea utilizar un nivel **PROVIDERVERSION** de 7, puede establecer **PROVIDERVERSION**=7, 7.0, 7.0.0 o 7.0.0.0.

8 - Modalidad normal

La aplicación JMS utiliza la modalidad normal del proveedor de mensajería IBM MQ. La modalidad normal utiliza todas las características de un gestor de colas IBM MQ para implementar JMS. Esta modalidad se optimiza para utilizar la API y las funciones de JMS 2.0.

Si te conectas a un gestor de colas con un nivel de comando 800 o superior, podrás utilizar todas las JMS 2.0 API y funciones, como el envío asíncrono, la entrega diferida o la suscripción compartida.

Si el gestor de colas especificado en la configuración de la fábrica de conexiones no es una versión IBM MQ 8.0.0 o posterior, el `createConnection` método falla y genera una excepción `JMSMQ0003`.

La modalidad normal del proveedor de mensajería de IBM MQ utiliza la característica de conversaciones compartidas y el número de conversaciones que se pueden compartir lo controla la propiedad **SHARECNV()** en el canal de conexión del servidor. Si esta propiedad se establece en 0, no puede utilizar la modalidad normal del proveedor de mensajería de IBM MQ y el método `createConnection` falla con una excepción `JMSCC5007`.

7 - Modalidad normal con restricciones

La aplicación JMS utiliza la modalidad normal con restricciones del proveedor de mensajería IBM MQ. Esta modalidad utiliza la API de JMS 2.0, pero no las nuevas características tales como suscripciones compartidas, entrega retrasada o envío asíncrono.

Si establece **PROVIDERVERSION** en 7 solo está disponible el proveedor de mensajería IBM MQ normal con la modalidad de restricciones de operación.

Si te conectas en modo normal con restricciones a un gestor de colas con un nivel de comando inferior a 800, puedes utilizar la JMS 2.0 API, pero no las funciones de envío asíncrono, entrega diferida ni suscripción compartida.

La modalidad normal del proveedor de mensajería de IBM MQ con restricciones utiliza la característica de compartición de conversaciones y el número de conversaciones que se pueden compartir lo controla la propiedad **SHARECNV()** en el canal de conexión del servidor. Si esta propiedad se establece a 0, no se podrá usar el modo normal de mensajería con restricciones de IBM MQ, y el método `createConnection` fallará con la excepción `JMSCC5007`.

6 - Modalidad de migración

La aplicación JMS utiliza la modalidad de migración de proveedor de mensajería de IBM MQ.

En este modo, puedes conectarte a un gestor de colas IBM MQ 8.0 de la versión o posterior, pero no se utilizan ninguna de las nuevas funciones de dicho gestor IBM MQ `classes for JMS` de colas, como, por ejemplo, la lectura anticipada o la transmisión.

Si tiene un cliente de IBM MQ 8.0 o posterior que se conecta a un gestor de colas de IBM MQ 8.0 o posterior, la selección de mensajes la realiza el gestor de colas en lugar del sistema cliente.

Si se especifica la modalidad de migración de proveedor de mensajería de IBM MQ e intenta utilizar cualquiera de las API JMS 2.0, la llamada de método de API falla con la excepción `JMSCC5007`.

unspecified (valor predeterminado)

La propiedad **PROVIDERVERSION** se establece en *unspecified* de forma predeterminada.

Una fábrica de conexiones que se ha creado con una versión anterior de IBM MQ `classes for JMS` en JNDI toma este valor cuando se utiliza la fábrica de conexiones con la nueva versión de IBM MQ `classes for JMS`. El siguiente algoritmo se utiliza para determinar qué modalidad de operación se utiliza. Este algoritmo se utiliza cuando se llama al método `createConnection` y utiliza otros aspectos de la fábrica de conexiones para determinar si se necesita la modalidad normal de proveedor de mensajería de IBM MQ, la modalidad normal con restricciones o la modalidad de migración de proveedor de mensajería de IBM MQ.

1. En primer lugar, se realiza un intento de utilizar la modalidad normal de proveedor de mensajería de IBM MQ.
2. Si el gestor de colas conectado no es IBM MQ 8.0 o posterior, se realiza un intento de utilizar la modalidad normal con restricciones del proveedor de mensajería de IBM MQ.
3. Si el gestor de colas conectado no es IBM WebSphere MQ 7.0.1 o posterior, la conexión se cierra y se utiliza en su lugar la modalidad de migración de proveedor de mensajería de IBM MQ.
4. Si la propiedad **SHARECNV** del canal de conexión del servidor se establece en 0, la conexión se cierra y en su lugar se utiliza la modalidad de migración del proveedor de mensajería de IBM MQ.

5. Si **BROKERVER** se establece en « V1 » o en el valor predeterminado *no especificado*, se seguirá utilizando el modo IBM MQ normal del proveedor de mensajería.

Consulte [ALTER QMGR](#) para obtener información sobre el parámetro PSMODE del mandato ALTER QMGR si necesita más información sobre compatibilidad.

6. Si **BROKERVER** se establece en V2, la acción realizada depende del valor de **BROKERQMGR**:

- Si el **BROKERQMGR** está en blanco:

Si la cola especificada mediante la propiedad **BROKERCONQ** se puede abrir para la salida (es decir, MQOPEN para la salida resulta satisfactorio) y **PSMODE** en el gestor de colas se establece en COMPAT o DISABLED, se utiliza la modalidad de migración del proveedor de mensajería de IBM MQ.

- Si la cola especificada por la propiedad **BROKERCONQ** no se puede abrir para la salida, o el atributo **PSMODE** se establece en ENABLED:

Se utiliza la modalidad normal de proveedor de mensajería de IBM MQ.

- Si **BROKERQMGR** no está en blanco:

Se utiliza la modalidad de proveedor de mensajería de IBM MQ.

Si no puede cambiar la fábrica de conexiones que está utilizando, puede utilizar la propiedad `com.ibm.msg.client.wmq.overrideProviderVersion` para alterar temporalmente cualquier valor de la fábrica de conexiones. Esta alteración temporal se aplica a todas las fábricas de conexiones de la JVM, pero los objetos de fábrica de conexiones reales no se modifican.

Tareas relacionadas

[Configuración de la propiedad JMS **PROVIDERVERSION**](#)

PROXYHOSTNAME

Nombre de host o dirección IP del sistema en el que se ejecuta el servidor proxy cuando se utiliza una conexión en tiempo real con un intermediario a través de un servidor proxy.

Objetos aplicables

ConnectionFactory, TopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : PROXYHOSTNAME

Nombre abreviado de la herramienta de administración de JMS : PHOST

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setProxyHostName()
- MQConnectionFactory.getProxyHostName()

Valores

null

Nombre de host del servidor proxy. Éste es el valor predeterminado.

PROXYPORT

Número del puerto en el que el servidor proxy está a la escucha cuando se utiliza una conexión en tiempo real con un intermediario a través de un servidor proxy.

Objetos aplicables

ConnectionFactory, TopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : PROXYPORT

Nombre abreviado de la herramienta de administración de JMS : PPORT

Acceso programático

Métodos setter/getters

MQConnectionFactory.setProxyPort()

MQConnectionFactory.getProxyPort()

Valores

443

Número de puerto del servidor proxy. Éste es el valor predeterminado.

PUBACKINT

Número de mensajes publicados por un publicador antes de que IBM MQ classes for JMS solicite un acuse de recibo al intermediario.

Cuando se reduce el valor de esta propiedad, IBM MQ classes for JMS solicita acuses de recibo con más frecuencia, por lo tanto, el rendimiento del publicador disminuye. Cuando aumente el valor, IBM MQ classes for JMS tardará más tiempo en emitir una excepción si el intermediario falla. Esta propiedad sólo tiene relevancia si TRANSPORT tiene el valor BIND o CLIENT.

Objetos aplicables

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : PROXYPORT

Nombre abreviado de la herramienta de administración de JMS : PPORT

Acceso programático

Métodos setter/getters

MQConnectionFactory.setPubAckInterval()

MQConnectionFactory.getPubAckInterval()

Valores

25

Cualquier entero positivo puede ser el valor predeterminado.

PUTASYNCAALLOWED

Esta propiedad determina si se permite a los productores de mensajes utilizar operaciones de transferencia asíncrona para enviar mensajes a este destino.

Objetos aplicables

Cola, Tema

Nombre largo de la herramienta de administración de JMS : PUTASYNCAALLOWED

Nombre abreviado de la herramienta de administración de JMS : PAALD

Acceso programático

Métodos setter/getters

MQDestination.setPutAsyncAllowed()

MQDestination.getPutAsyncAllowed()

Valores

AS_DEST

Determine si las operaciones de transferencia asíncrona están permitidas consultando la definición de cola o tema. Éste es el valor predeterminado.

AS_Q_DEF

Determine si las operaciones de transferencia asíncrona están permitidas consultando la definición de cola. El nombre de la propiedad de la cola es **DefPResp (MQLONG)**.

AS_TOPIC_DEF

Determine si las operaciones de transferencia asíncrona están permitidas consultando la definición de tema.

NO

Las operaciones de transferencia asíncrona no están permitidas.

SÍ

Las operaciones de transferencia asíncrona están permitidas.

QMANAGER

Nombre del gestor de colas al que se va a conectar.

Sin embargo, si la aplicación utiliza una tabla de definiciones de canal de cliente para conectarse a un gestor de colas, consulte [Utilización de una tabla de definiciones de canal de cliente con IBM MQ classes for JMS](#).

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, Cola, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : QMANAGER

Nombre abreviado de la herramienta de administración de JMS : QMGR

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setQueueManager()
- MQConnectionFactory.getQueueManager()

Valores

"" (serie vacía)

Cualquier serie puede ser el valor predeterminado.

COLA

El nombre del destino de la JMS cola. Esto coincide con el nombre de la cola utilizada por el gestor de colas.

Objetos aplicables

Cola

JMS Nombre completo de la herramienta de administración: QUEUE

JMS Nombre abreviado de la herramienta de administración: QU

Valores

Cualquier serie

Cualquier nombre de cola IBM MQ válido.

Referencia relacionada

[Reglas para nombrar IBM MQ objetos](#)

READAHEADALLOWED

Esta propiedad determina si los consumidores de mensajes y los navegadores de colas pueden utilizar la lectura anticipada para obtener mensajes no persistentes de este destino en un almacenamiento intermedio interno antes de recibirlos.

Objetos aplicables

Cola, Tema

Nombre largo de la herramienta de administración de JMS : READAHEADALLOWED

Nombre abreviado de la herramienta de administración de JMS : RAALD

Acceso programático

Métodos setter/getters

- MQDestination.setReadAheadAllowed()
- MQDestination.getReadAheadAllowed()

Valores

AS_DEST

Determine si se permite la lectura anticipada consultando la definición de cola o tema. Este es el valor predeterminado en las herramientas administrativas.

Utilice WMQConstants.WMQ_READ_AHEAD_ALLOWED_AS_DEST en programas.

AS_Q_DEF

Determine si se permite la lectura anticipada consultando la definición de cola.

Utilice WMQConstants.WMQ_READ_AHEAD_ALLOWED_AS_Q_DEF en programas.

AS_TOPIC_DEF

Determine si se permite la lectura anticipada consultando la definición de tema.

Utilice WMQConstants.WMQ_READ_AHEAD_ALLOWED_AS_TOPIC_DEF en los programas.

NO

No se permite la lectura anticipada.

Utilice WMQConstants.WMQ_READ_AHEAD_ALLOWED_DISABLED en los programas.

SÍ

La lectura anticipada está permitida.

Utilice WMQConstants.WMQ_READ_AHEAD_ALLOWED_ENABLED en los programas.

READAHEADCLOSEPOLICY

Para los mensajes que se entregan a un escucha de mensajes asíncrono, qué sucede con los mensajes del almacenamiento intermedio de lectura anticipada interno cuando se cierra el consumidor de mensajes.

Objetos aplicables

Cola, Tema

Nombre largo de la herramienta de administración de JMS : READAHEADCLOSEPOLICY

Nombre abreviado de la herramienta de administración de JMS : RACP

Acceso programático

Métodos setter/getters

- `MQDestination.setReadAheadClosePolicy()`
- `MQDestination.getReadAheadClosePolicy()`

Valores

ENTREGAR_TODO

Todos los mensajes del almacenamiento intermedio de lectura anticipada interno se entregan al escucha de mensajes de la aplicación antes de devolverlos. Este es el valor predeterminado en las herramientas administrativas.

Utilice `WMQConstants.WMQ_READ_AHEAD_DELIVERALL` en programas.

ACTUAL_ENTREGA

Solo se completa la invocación del escucha de mensajes actual antes de la devolución, posiblemente dejando mensajes en el almacenamiento intermedio de lectura anticipada interno que, después, se descartan.

Utilice `WMQConstants.WMQ_READ_AHEAD_DELIVERCURRENT` en programas.

RECEIVECCSID

La propiedad de destino que define el CCSID de destino para la conversión de mensajes del gestor de colas. El valor se ignora a menos que `RECEPECONVERSION` se establezca en `WMQ_RECEIVE_CONVERSION_QMGR`

Objetos aplicables

Cola, Tema

Nombre largo de la herramienta de administración de JMS : RECEIVECCSID

Nombre abreviado de la herramienta de administración de JMS : RCCS

Acceso programático

Métodos setter/Getters

- `MQDestination.setReceiveCCSID`
- `MQDestination.getReceiveCCSID`

Valores

`WMQConstants.WMQ_RECEIVE_CC_SID_JVM_DEFAULT`

0 - Utilizar `Charset.defaultCharset` de JVM

1208

UTF-8

ccsid

Identificador de juego de caracteres codificado soportado.

RECEIVECONVERSION

Propiedad de destino que determina si el gestor de colas va a realizar la conversión de datos.

Objetos aplicables

Cola, Tema

Nombre largo de la herramienta de administración de JMS : RECEPECONVERSION

Nombre abreviado de la herramienta de administración de JMS : RCNV

Acceso programático

Métodos setter/Getters

- `MQDestination.setReceiveConversion`
- `MQDestination.getReceiveConversion`

Valores

WMQConstants.WMQ_RECEIVE_CONVERSION_CLIENT_MSG

1 -Sólo realiza la conversión de datos en el cliente JMS . El valor predeterminado de hasta V7.0, y de, e incluyendo, 7.0.1.5.

WMQConstants.WMQ_RECEIVE_CONVERSION_QMGR

2 -Realizar la conversión de datos en el gestor de colas antes de enviar un mensaje al cliente. El valor predeterminado (y solo) de V7.0 a V7.0.1.4 inclusive, excepto si se aplica el APAR IC72897 .

RECEIVEISOLATION

Esta propiedad determina si un suscriptor puede recibir mensajes que no se han confirmado en la cola de suscriptores.

Objetos aplicables

ConnectionFactory, TopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : RECEIVEISOLATION

Nombre abreviado de la herramienta de administración de JMS : RCVISOL

Valores

CONFIRMADO

Un suscriptor sólo recibe los mensajes de la cola de suscriptores que se han confirmado. Este es el valor predeterminado en las herramientas administrativas.

Utilice `WMQConstants.WMQ_RCVISOL_COMMITTED` en los programas.

SIN CONFIRMAR

Un suscriptor puede recibir mensajes que no se han confirmado en la cola de suscriptor.

Utilice `WMQConstants.WMQ_RCVISOL_UNCOMMITTED` en programas.

RECEXIT

Identifica una salida de recepción de canal, o una secuencia de salidas de recepción, que se ejecutarán sucesivamente.

Es posible que sea necesaria una configuración adicional para que IBM MQ classes for JMS localice las salidas de recepción. Para obtener más información, consulte [«Configuración de las clases de « IBM MQ » para que JMS utilice salidas de canal »](#).

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : RECEXIT

Nombre abreviado de la herramienta de administración de JMS : RCX

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setReceiveExit()
- MQConnectionFactory.getReceiveExit()

Valores

- `null`. Éste es el valor predeterminado.
- Una serie que consta de uno o más elementos separados por comas, donde cada elemento es:
 - El nombre de una clase que implementa la interfaz `WMQReceiveExit` (para una salida de recepción de canal escrita en Java).
 - Una cadena en el formato *libraryName(entryPointName)* (para una salida de recepción de canal no escrita en Java).

RECEXITINIT

Los datos de usuario que se pasan a las salidas de recepción de canal cuando se llaman.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : RECEXITINIT

Nombre abreviado de la herramienta de administración de JMS : RCXI

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setReceiveExitInit()
- MQConnectionFactory.getReceiveExitInit()

Valores

`null`

Una cadena que comprende uno o más elementos de datos de usuario separados por comas. Éste es el valor predeterminado.

REPLYTOSTYLE

Determina cómo se construye el campo JMSReplyTo en un mensaje recibido.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : REPLYTOSTYLE

Nombre abreviado de la herramienta de administración de JMS : RTOST

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setReplyToStyle()
- MQConnectionFactory.getReplyToStyle()

Valores

PREDETERMINADO

Equivale a MQMD.

RFH2

Utilice el valor proporcionado en la cabecera RFH2 . Si se ha establecido un valor JMSReplyTo en la aplicación emisora, utilice ese valor.

MQMD

Utilice el valor proporcionado por MQMD. Este comportamiento es equivalente al comportamiento predeterminado de IBM WebSphere MQ 6.0.2.4 y 6.0.2.5.

Si el valor JMSReplyTo establecido por la aplicación emisora no contiene un nombre de gestor de colas, el gestor de colas receptor inserta su propio nombre en el MQMD. Si establece este parámetro en MQMD, la cola de respuesta que utiliza se encuentra en el gestor de colas receptor. Si establece este parámetro en RFH2, la cola de respuesta que utiliza se encuentra en el gestor de colas especificado en la RFH2 del mensaje enviado, tal como lo estableció originalmente la aplicación emisora.

Si el valor JMSReplyTo establecido por la aplicación emisora contiene un nombre de gestor de colas, el valor de este parámetro no es importante porque tanto MQMD como RFH2 contienen el mismo valor.

RESCANINT

Cuando un consumidor de mensajes del dominio punto a punto utiliza un selector de mensajes para seleccionar qué mensajes desea recibir, IBM MQ classes for JMS busque en la cola IBM MQ los mensajes adecuados en la secuencia determinada por el atributo `MsgDeliverySequence` de la cola.

Después de que IBM MQ classes for JMS encuentre un mensaje adecuado y lo entregue al consumidor, IBM MQ classes for JMS reanuda la búsqueda del siguiente mensaje adecuado desde su posición actual en la cola. IBM MQ classes for JMS continúa buscando en la cola de esta forma hasta que alcanza el final de la cola, o hasta que el intervalo de tiempo en milisegundos, determinado por el valor de esta propiedad, ha caducado. En cada caso, IBM MQ classes for JMS vuelve al principio de la cola para continuar la búsqueda y comienza un nuevo intervalo de tiempo.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory

Nombre largo de la herramienta de administración de JMS : RESCANINT

Nombre abreviado de la herramienta de administración de JMS : RINT

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setRescanInterval()
- MQConnectionFactory.getRescanInterval()

Valores

5000

Cualquier entero positivo puede ser el valor predeterminado.

SECEXIT

Identifica una salida de seguridad de canal.

Es posible que sea necesaria una configuración adicional para que IBM MQ classes for JMS localice las salidas de seguridad. Para obtener más información, consulte [«Configuración de las clases de « IBM MQ » para que JMS utilice salidas de canal »](#).

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : SECEXIT

Nombre abreviado de la herramienta de administración de JMS : SXC

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setSecurityExit()
- MQConnectionFactory.getSecurityExit()

Valores

- `null`. Éste es el valor predeterminado.
- Una serie que consta de uno o más elementos separados por comas, donde cada elemento es:
 - El nombre de una clase que implementa la interfaz `WMQSecurityExit` (para una salida de seguridad de canal escrita en Java).
 - Una cadena con el formato `libraryName(entryPointName)` (para una salida de seguridad del canal no escrita en Java).

SECEXITINIT

Los datos de usuario que se pasan a una salida de seguridad de canal cuando se invoca.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : SECEXITINIT

Nombre abreviado de la herramienta de administración de JMS : SCXI

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setSecurityExitInit()
- MQConnectionFactory.getSecurityExitInit()

Valores

null

Cualquier serie puede ser el valor predeterminado.

SENDCHECKCOUNT

Número de llamadas de envío que se deben permitir entre las comprobaciones de errores de transferencia asíncrona, dentro de una misma sesión no transaccional de JMS

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : SENDCHECKCOUNT

Nombre abreviado de la herramienta de administración de JMS : SCC

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setSendCheckCount()
- MQConnectionFactory.getSendCheckCount()

Valores

null

Cualquier serie puede ser el valor predeterminado.

SENDEXIT

Identifica una salida de envío de canal o una secuencia de salidas de envío que se van a ejecutar sucesivamente.

Es posible que sea necesaria una configuración adicional para que IBM MQ classes for JMS localice las salidas de envío. Para obtener más información, consulte [«Configuración de las clases de « IBM MQ » para que JMS utilice salidas de canal »](#).

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : SENDEXIT

Nombre abreviado de la herramienta de administración de JMS : SDX

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setSendExit()
- MQConnectionFactory.getSendExit()

Valores

- `null`. Éste es el valor predeterminado.
- Una serie que consta de uno o más elementos separados por comas, donde cada elemento es:
 - El nombre de una clase que implementa la interfaz `WMQSendExit` (para una salida de emisión de canal escrita en Java).
 - Una cadena en el formato `libraryName(entryPointName)` (para una salida de envío de canal no escrita en Java).

SENDEXITINIT

Los datos de usuario que se pasan a las salidas de emisión de canal cuando se invocan.

Objetos aplicables

`ConnectionFactory`, `QueueConnectionFactory`, `TopicConnectionFactory`, `XAConnectionFactory`, `XAQueueConnectionFactory`, `XATopicConnectionFactory`

Nombre largo de la herramienta de administración de JMS : `SENDEXITINIT`

Nombre abreviado de la herramienta de administración de JMS : `SDXI`

Acceso programático

Métodos setter/getters

- `MQConnectionFactory.setSendExitInit()`
- `MQConnectionFactory.getSendExitInit()`

Valores

`null`

Cualquier serie que comprenda uno o más elementos de datos de usuario separados por comas puede ser el valor predeterminado.

SHARECONVALLOWED

Para aplicaciones que utilizan el modo IBM MQ normal del proveedor de mensajería o el modo normal con restricciones, esta propiedad determina si la función de compartir conversaciones se utiliza para JMS conexiones, sesiones y contextos creados desde la fábrica de conexiones.

Objetos aplicables

`ConnectionFactory`, `QueueConnectionFactory`, `TopicConnectionFactory`, `XAConnectionFactory`, `XAQueueConnectionFactory`, `XATopicConnectionFactory`

JMS Nombre completo de la herramienta administrativa: `SHARECONVALLOWED`

JMS Nombre abreviado de la herramienta de administración: `SCALD`

Acceso programático

Métodos de lectura y escritura

- `MQConnectionFactory.setShareConvAllowed()`
- `MQConnectionFactory.getShareConvAllowed()`

Valores

sí

JMS Las conexiones, sesiones y contextos creados desde la fábrica de conexiones dentro de la misma instancia de ` JVM ` pueden compartir una instancia de canal (que se corresponde con una conexión ` TCP/IP `) cuando sea pertinente.

Este es el valor predeterminado para las herramientas administrativas.

Para los programas, utilice `WMQConstants.WMQ_SHARE_CONV_ALLOWED_YES`.

NO

Cada JMS conexión creada a partir de la fábrica de conexiones, y cada JMS sesión creada a partir de esas JMSconexiones, tiene su propia instancia de canal (conexión TCP/IP) con un gestor de colas.

En cuanto a JMS los contextos, el primer contexto que se crea a partir de la fábrica de conexiones genera dos instancias de canal (conexiones TCP/IP). Los demás JMS contextos que se crean a partir del primero tienen su propia instancia de canal (conexión TCP/IP).

Para los programas, utilice `WMQConstants.WMQ_SHARE_CONV_ALLOWED_NO`.

Conceptos relacionados

[Modalidades de funcionamiento del proveedor de mensajería de IBM MQ](#)

[Compartir una conexión « TCP/IP » en clases « IBM MQ » para JMS](#)

SPARSESUBS

Controla la política de recuperación de mensajes de un objeto TopicSubscriber.

Objetos aplicables

ConnectionFactory, TopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : SPARSESUBS

Nombre abreviado de la herramienta de administración de JMS : SSUBS

Acceso programático

Métodos setter/getters

- `MQConnectionFactory.setSparseSubscriptions()`
- `MQConnectionFactory.getSparseSubscriptions()`

Valores

NO

Las suscripciones reciben mensajes coincidentes frecuentes. Este es el valor predeterminado para las herramientas administrativas.

Para los programas, utilice `false`.

sí

Las suscripciones reciben mensajes coincidentes poco frecuentes. Este valor exige que la cola de suscripciones se pueda abrir para examinarla.

Para los programas, utilice `true`.

SSLCIPHERSUITE

CipherSuite que se debe utilizar para una conexión TLS.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS Nombre completo de la herramienta de administración: SSLCIPHERSUITE

JMS Nombre abreviado de la herramienta administrativa: SCPHS

Acceso programático

Métodos de lectura y escritura

- MQConnectionFactory.setSSLCipherSuite()
- MQConnectionFactory.getSSLCipherSuite()

Valores

null

Éste es el valor predeterminado. Para obtener más información, consulte [las propiedades de JMS los objetos de TLS](#).

Referencia relacionada

[Habilitar TLS con la propiedad del objeto SSLCIPHERSUITE](#)

SSLCRL

Servidores CRL para comprobar la revocación de certificados TLS.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS Nombre completo de la herramienta administrativa: SSLCRL

JMS Nombre abreviado de la herramienta de administración: SCRL

Acceso programático

Métodos de lectura y escritura

- MQConnectionFactory.setSSLCertStores()
- MQConnectionFactory.getSSLCertStores()

Valores

null

Lista separada por espacios de direcciones URL LDAP. Éste es el valor predeterminado. Para obtener más información, consulte [las propiedades de JMS los objetos de TLS](#).

SSLFIPSREQUIRED

Esta propiedad determina si una conexión TLS debe utilizar un certificado de firma digital (JSSECipherSuite) que sea compatible IBMJavaJava Secure Socket ExtensionFIPS con el proveedor (IBMJSSEFIPS).

Notas:

-  En AIX, Linux, y Windows, IBM MQ garantiza FIPS 140-3 el cumplimiento de la normativa a través del GSKit 9 módulo IBM Crypto for C (ICC) criptográfico. La certificación

del NIST correspondiente al FIPS 140-3 módulo puede consultarse en <https://csrc.nist.gov/projects/cryptographic-module-validation-program/certificate/4755>.

- V 10.0.0 > V 10.0.0 FIPS Actualmente no hay soporte disponible para Linux s390x. Los clientes de esta plataforma que necesiten FIPS asistencia técnica deben seguir utilizando una versión anterior de IBM MQ. La compatibilidad con FIPS 140-3 se habilitará en una futura versión de FixPack una vez que el IBM Crypto for C (ICC) módulo criptográfico haya obtenido su certificación en esta plataforma.
- V 10.0.0 El módulo FIPS 140-3 criptográfico integrado IBM Semeru Runtime fue aprobado por el NIST en agosto de 2024. IBM MQ 10.0.0 incorpora compatibilidad con la gestión de IBM MQ classes for JMS conexiones de IBM MQ classes for Java cliente mediante TLS para FIPS 140-3 en Java 8 y IBM Semeru Runtime 11+. La certificación del NIST correspondiente al FIPS 140-3 módulo se puede consultar en <https://csrc.nist.gov/projects/cryptographic-module-validation-program/certificate/4755>.
- En IBM MQ el caso de Containers, las imágenes de IBM MQ Operator contenedor de 3.2.0 y del gestor de colas 9.4.0.0 en adelante se basan en UBI 9. FIPS 140-3 El cumplimiento de IBM MQ la normativa para productos envasados está actualmente pendiente.

V 10.0.0 Si FIPS está habilitado, los procesos IBM MQ de control de contenedores utilizan un módulo FIPS 140-3 certificado de OpenSSL. Para obtener más información sobre la certificación del NIST, consulte: <https://access.redhat.com/compliance/fips>. IBM MQ Los gestores de colas que se ejecutan en imágenes de contenedor tienen el mismo nivel FIPS de certificación que la versión de la plataforma de IBM MQ la imagen base.

V 10.0.0 Desde IBM MQ 10.0.0, la propiedad de fábrica de conexiones **SSLFIPSREQUIRED** no es compatible con IBM Semeru Runtime. No causa un error si se incluye en aplicaciones cliente. Se lanza una nueva excepción JMSEException si se ha establecido la propiedad, pero no se han establecido las IBM Semeru Runtime demás FIPS propiedades. De este modo se evita que las aplicaciones cliente existentes se trasladen a Java 11+ y esperen una FIPS conexión.

Nota: Esto no afecta a los clientes actuales que utilizan IBM Java 8, solo afecta a los clientes que utilizan IBM Semeru Runtime Java.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS Nombre completo de la herramienta de administración: SSLFIPSREQUIRED

JMS Nombre abreviado de la herramienta administrativa: SFIPS

Acceso programático

Métodos de lectura y escritura

- MQConnectionFactory.setSSLFipsRequired()
- MQConnectionFactory.getSSLFipsRequired()

Valores

NO

Una conexión TLS puede utilizar cualquier CipherSuite que no sea compatible con el IBMJavaJSSEFIPS proveedor (IBMJSSEFIPS).

Éste es el valor predeterminado. En los programas, utilice false.

SÍ

Una conexión TLS debe utilizar un CipherSuite compatible con IBMJSSEFIPS.

En los programas, utilice true.

Referencia relacionada

Uso de SSLFIPSREQUIRED para especificar si solo se utilizan CipherSpecsFIPS certificados para una conexión

SSLPEERNAME

Para TLS, un esqueleto *de nombre distinguido* que debe coincidir con el proporcionado por el gestor de colas.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS Nombre completo de la herramienta de administración: SSLPEERNAME

JMS Nombre abreviado de la herramienta administrativa: SPEER

Acceso programático

Métodos de lectura y escritura

- MQConnectionFactory.setSSLPeerName()
- MQConnectionFactory.getSSLPeerName()

Valores

null

Éste es el valor predeterminado. Para obtener más información, consulte [las propiedades de JMS los objetos de TLS](#).

Referencia relacionada

[Uso de SSLPEERNAME para especificar un patrón de nombre distinguido](#)

SSLRESETCOUNT

Para TLS, el número total de bytes enviados y recibidos por una conexión antes de que se renegocie la clave secreta que se utiliza para el cifrado.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS Nombre completo de la herramienta de administración: SSLRESETCOUNT

JMS Nombre abreviado de la herramienta de administración: SRC

Acceso programático

Métodos de lectura y escritura

- MQConnectionFactory.setSSLResetCount()
- MQConnectionFactory.getSSLResetCount()

Valores

0

Cero, o cualquier número entero positivo menor o igual a 999 999 999. Éste es el valor predeterminado. Para obtener más información, consulte [las propiedades de JMS los objetos de TLS](#).

Referencia relacionada

[Uso de SSLRESETCOUNT para especificar cuándo se renegocia una clave secreta](#)

STATREFRESHINT

Intervalo, en milisegundos, entre las actualizaciones de la transacción de larga ejecución que detecta cuando un suscriptor pierde su conexión con el gestor de colas.

Esta propiedad solo es relevante si SUBSTORE tiene el valor QUEUE.

Objetos aplicables

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

JMS Nombre completo de la herramienta administrativa: STATREFRESHINT

JMS Nombre abreviado de la herramienta de administración: SRI

Acceso programático

Métodos de lectura y escritura

- MQConnectionFactory.setStatusRefreshInterval()
- MQConnectionFactory.getStatusRefreshInterval()

Valores

60000

Cualquier número entero positivo puede ser el valor predeterminado. Para obtener más información, consulte [las propiedades de JMS los objetos de TLS](#).

SUBSTORE

Donde IBM MQ classes for JMS almacena datos persistentes relacionados con suscripciones activas.

Objetos aplicables

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : SUBSTORE

Nombre abreviado de la herramienta de administración de JMS : SS

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setSubscriptionStore()
- MQConnectionFactory.getSubscriptionStore()

Valores

BROKER

Utilice el almacén de suscripciones basado en intermediario para mantener los detalles de las suscripciones. Este es el valor predeterminado para las herramientas administrativas.

Para los programas, utilice WMQConstants.WMQ_SUBSTORE_BROKER.

MIGRATE

Transfiera la información de suscripción del almacén de suscripciones basado en cola al almacén de suscripciones basado en intermediario.

Para los programas, utilice WMQConstants.WMQ_SUBSTORE_MIGRATE.

COLA

Utilice el almacén de suscripciones basado en cola para contener los detalles de las suscripciones.

Para los programas, utilice WMQConstants.WMQ_SUBSTORE_QUEUE.

SYNCPOINTALLGETS

Esta propiedad determina si todas las obtenciones se deben realizar bajo punto de sincronismo.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : SYNCPOINTALLGETS

Nombre abreviado de la herramienta de administración de JMS : SPAG

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setSyncpointAllGets()
- MQConnectionFactory.getSyncpointAllGets()

Valores

No

Éste es el valor predeterminado.

Sí

TARGCLIENT

Esta propiedad determina si el formato IBM MQ RFH2 se utiliza para intercambiar información con aplicaciones de destino.

Objetos aplicables

Cola, Tema

Nombre largo de la herramienta de administración de JMS : TARGCLIENT

Nombre abreviado de la herramienta de administración de JMS : TC

Acceso programático

Métodos setter/getters

- MQDestination.setTargetClient()
- MQDestination.getTargetClient()

Valores

JMS

El destino del mensaje es una aplicación JMS . Este es el valor predeterminado para las herramientas administrativas.

Para programas, utilice WMQConstants.WMQ_CLIENT_JMS_COMPLIANT.

MQ

El destino del mensaje es una aplicación que no es de JMS IBM MQ .

Para programas, utilice WMQConstants.WMQ_CLIENT_NONJMS_MQ.

TARGCLIENTMATCHING

Esta propiedad determina si un mensaje de respuesta, enviado a la cola identificada por el campo de cabecera JMSReplyTo de un mensaje de entrada, tiene una cabecera MQRFH2 sólo si el mensaje de entrada tiene una cabecera MQRFH2 .

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory

Nombre largo de la herramienta de administración de JMS : TARGCLIENTMATCHING

Nombre abreviado de la herramienta de administración de JMS : TCM

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setTargetClientMatching()
- MQConnectionFactory.getTargetClientMatching()

Valores

sí

Si un mensaje de entrada no tiene una cabecera MQRFH2 , la propiedad TARGCLIENT del objeto Queue derivado del campo de cabecera JMSReplyTo del mensaje se envía a MQ. Si el mensaje tiene una cabecera MQRFH2 , la propiedad TARGCLIENT se establece en JMS en su lugar. Este es el valor predeterminado para las herramientas administrativas.

Para los programas, utilice true.

NO

La propiedad TARGCLIENT del objeto Queue derivada del campo de cabecera JMSReplyTo de un mensaje de entrada siempre se establece en JMS.

Para los programas, utilice false.

TEMPMODEL

Nombre de la cola modelo a partir de la cual se crean colas temporales de JMS.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory

Nombre largo de la herramienta de administración de JMS : TEMPMODEL

Nombre abreviado de la herramienta de administración de JMS : TM

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setTemporaryModel()
- MQConnectionFactory.getTemporaryModel()

Valores

SYSTEM.DEFAULT.MODEL.QUEUE

Cualquier serie puede ser el valor predeterminado.

TEMPQPREFIX

Prefijo que se utiliza para formar el nombre de una cola dinámica de IBM MQ.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory

Nombre largo de la herramienta de administración de JMS : TEMPQPREFIX

Nombre abreviado de la herramienta de administración de JMS : TQP

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setTempQPrefix()
- MQConnectionFactory.getTempQPrefix()

Valores

" " (serie vacía)

El prefijo utilizado es CSQ.* en z/OS y AMQ.* en todas las demás plataformas. Estos son los valores predeterminados.

Prefijo de cola

El prefijo de cola es cualquier serie que se ajuste a las reglas para formar contenido del campo *DynamicQName* en un descriptor de objeto IBM MQ (estructura MQOD), pero el último carácter no en blanco debe ser un asterisco.

TEMPTOPICPREFIX

Al crear temas temporales, JMS genera una serie de tema con el formato "TEMP/*TEMPTOPICPREFIX/unique_id*", o si esta propiedad se deja con el valor predeterminado, sólo "TEMP/*unique_id*".

Especificar un TEMPTOPICPREFIX no vacío permite definir colas de modelo específicas para crear las colas gestionadas para suscriptores a temas temporales creados bajo esta conexión.

Objetos aplicables

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración de JMS : TEMPTOPICPREFIX

Nombre abreviado de la herramienta de administración de JMS : TTP

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setTempTopicPrefix()
- MQConnectionFactory.getTempTopicPrefix()

Valores

Cualquier serie no nula que conste sólo de caracteres válidos para una serie de tema IBM MQ. El valor predeterminado es " " (serie vacía).

V.10.0.0 ID_CLIENTE_TOKEN

El ID de cliente tal y como se define en un proveedor de tokens web JSON (JWT).

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS Nombre completo de la herramienta de administración: TOKEN_CLIENT_ID

JMS Nombre abreviado de la herramienta de administración: TKCD

Acceso programático

Métodos de lectura y escritura

- MQConnectionFactory.setTokenClientId()
- MQConnectionFactory.getTokenClientId()

Valores

null

Éste es el valor predeterminado.

Cualquier cadena válida

Nota: La propiedad **TOKEN_CLIENT_ID** connection factory es obligatoria si se especifica la propiedad **TOKEN_ENDPOINT** connection factory.

V 1000 SECRETO_DEL_CLIENTE_TOKEN

El secreto del cliente, es decir, un secreto que solo conocen la aplicación cliente y el proveedor de tokens.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS Nombre completo de la herramienta de administración: TOKEN_CLIENT_SECRET

JMS Nombre abreviado de la herramienta de administración: TKST

Acceso programático

Métodos de lectura y escritura

- MQConnectionFactory.setTokenClientSecret()
- MQConnectionFactory.getTokenClientSecret()

Valores

null

Éste es el valor predeterminado.

Cualquier cadena válida

Nota: La propiedad **TOKEN_CLIENT_SECRET** connection factory es obligatoria si se especifica la propiedad **TOKEN_ENDPOINT** connection factory.

V 1000 PUNTO FINAL DEL TOKEN

El URI del punto final de un proveedor de tokens.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS Nombre completo de la herramienta de administración: TOKEN_ENDPOINT

JMS Nombre abreviado de la herramienta de administración: TKEP

Acceso programático

Métodos de lectura y escritura

- MQConnectionFactory.setTokenEndpoint()
- MQConnectionFactory.getTokenEndpoint()

Valores

null

Éste es el valor predeterminado.

Cualquier cadena válida

Debe ser un proveedor de tokens válido URL.

Nota: Si se especifica la propiedad de **TOKEN_ENDPOINT** fábrica de conexiones:

- Las **TOKEN_CLIENT_ID** y **TOKEN_CLIENT_SECRET** propiedades y son obligatorias.
- La propiedad de **USER_AUTHENTICATION_MQCSP** la fábrica de conexiones debe establecerse en verdadero.
- Tampoco debe especificar **UserId** en las propiedades de la fábrica de conexiones.

Nota: V 10.00 Puedes añadir una configuración personalizada de « SSLSocketFactory » mediante la **TOKEN_SSL_SOCKET_FACTORY** propiedad para controlar cómo tu aplicación recupera de forma segura los tokens JWT de los puntos finales de HTTPS. Si una aplicación cliente proporciona el **TOKEN_SSL_SOCKET_FACTORY**, la **TOKEN_ENDPOINT** propiedad se convierte en obligatoria y, en caso de que falte, se devuelve un mensaje de error a la aplicación cliente. Para obtener más información, consulta [«Cómo proporcionar un archivo `SSLSocketFactory` personalizado en IBM MQ classes for JMS/Jakarta Messaging»](#).

V 10.00 POLÍTICA_DE_VALIDACIÓN_DE_CERTIFICADOS_HTTPS_TOKEN

El nivel de validación que ejecuta la biblioteca IBM MQ cliente cuando recupera un JWT de una conexión HTTPS.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS Nombre completo de la herramienta administrativa: TOKENCERTVALPOL

JMS Nombre abreviado de la herramienta de administración: TCVP

Acceso programático

Métodos de lectura y escritura

- setTokenHttpsCertValidationPolicy(int valor)
- getTokenHttpsCertValidationPolicy()

Valores

CUALQUIERA

El valor es 1.

Ninguna

El valor es 2.

HOSTNAMECN

El valor es 3. Éste es el valor predeterminado.

TOPIC

El nombre del destino JMS del tema. El gestor de colas utiliza este valor como cadena de tema de una publicación o suscripción.

Objetos aplicables

Tema

JMS Nombre completo de la herramienta de administración: TOPIC

JMS Nombre abreviado de la herramienta de administración: TOP

Valores

Cualquier serie

Una cadena que forma una cadena de tema IBM MQ válida. Cuando utilice IBM MQ como proveedor de mensajería con WebSphere Application Server, especifique un valor que coincida con el nombre por el que se conoce el tema con fines administrativos dentro WebSphere Application Server.

Referencia relacionada

[Series de tema](#)

TRANSPORT

La naturaleza de una conexión a un gestor de colas o intermediario.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

JMS Nombre completo de la herramienta de administración: TRANSPORT

JMS Nombre abreviado de la herramienta administrativa: TRAN

Acceso programático

Métodos de lectura y escritura

- MQConnectionFactory.setTransportType()
- MQConnectionFactory.getTransportType()

Valores

BIND

Para una conexión a un gestor de colas en modo de enlaces. Este es el valor predeterminado para las herramientas administrativas.

En este modo, se establece una conexión local con el gestor de colas cuyo nombre se indica en la propiedad QMANAGER. Sin embargo, si también se especifica la propiedad de modo cliente CCDTURL, la propiedad CCDTURL tiene prioridad (para garantizar la compatibilidad con versiones anteriores de IBM MQ). En su lugar, se establece una conexión en modo cliente con el gestor de colas especificado en el archivo CCDT.

Si se establecen otras propiedades específicas del modo cliente, como CHANNEL, dichas propiedades se ignorarán.

Para los programas, utilice `WMQConstants.WMQ_CM_BINDINGS`.

CLIENTE

Para una conexión a un gestor de colas en modo cliente.

La conexión se controla mediante una de las siguientes propiedades o combinaciones de propiedades: `CONNECTIONNAMELIST`, `HOSTNAME` y `PORT`, o `CCDTURL`.

Para los programas, utilice `WMQConstants.WMQ_CM_CLIENT`.

DIRECT

Para una conexión en tiempo real con un bróker sin utilizar un túnel de HTTP.

Para los programas, utilice `WMQConstants.WMQ_CM_DIRECT_TCPIP`.

DIRECTHTTP

Para una conexión en tiempo real con un bróker mediante un túnel de HTTP. Solo se admite HTTP 1.0.

Para los programas, utilice `WMQConstants.WMQ_CM_DIRECT_HTTP`.

Conceptos relacionados

[“Dependencias entre propiedades de objetos IBM MQ classes for JMS” en la página 1931](#)
La validez de algunas propiedades depende de los valores particulares de otras propiedades.

WILDCARDFORMAT

Esta propiedad determina qué versión de sintaxis de comodín se va a utilizar.

Objetos aplicables

`ConnectionFactory`, `TopicConnectionFactory`, `XAConnectionFactory`, `XATopicConnectionFactory`

Nombre largo de la herramienta de administración de JMS : `WILDCARDFORMAT`

Nombre abreviado de la herramienta de administración de JMS : `WCFMT`

Acceso programático

Métodos setter/getters

- `MQConnectionFactory.setWildCardFormat()`
- `MQConnectionFactory.getWildCardFormat()`

Valores

SOLO EL TEMA

Sólo reconoce comodines de nivel de tema, tal como se utiliza en la versión 2 del intermediario. Este es el valor predeterminado para las herramientas administrativas.

Para los programas, utilice `WMQConstants.WMQ_WILDCARD_TOPIC_ONLY`.

CAR_SOLO

Sólo reconoce caracteres comodín, tal como se utiliza en la versión 1 del intermediario.

Para los programas, utilice `WMQConstants.WMQ_WILDCARD_CHAR_ONLY`.

Política de validación de certificados HTTPS de WMQ_CCDT

El nivel de validación que ejecuta la biblioteca IBM MQ cliente cuando recupera un CCDT de una conexión HTTPS.

Objetos aplicables

`ConnectionFactory`, `QueueConnectionFactory`, `TopicConnectionFactory`, `XAConnectionFactory`, `XAQueueConnectionFactory`, `XATopicConnectionFactory`

JMS Nombre completo de la herramienta administrativa: `CCDTCERTVALPOL`

JMS Nombre abreviado de la herramienta de administración: CCVP

Acceso programático

Métodos de lectura y escritura

- setCCDTHttpsCertValidationPolicy(int valor)
- getCCDTHttpsCertValidationPolicy()

Valores

CUALQUIERA

El valor es 1.

Ninguna

El valor es 2.

HOSTNAMECN

El valor es 3. Éste es el valor predeterminado.

La propiedad ENCODING

La propiedad ENCODING comprende tres subpropiedades, en doce combinaciones posibles.

Los valores válidos que puede tomar la propiedad ENCODING se construyen a partir de las tres subpropiedades:

Codificación de enteros

Normal o invertido

Codificación decimal

Normal o invertido

codificación de coma flotante

IEEE normal, IEEE invertido o z/OS

La propiedad ENCODING se expresa como una serie de tres caracteres con la sintaxis siguiente:

```
{N|R}{N|R}{N|R|3}
```

En esta serie:

- N indica normal
- R indica invertido
- 3 denota z/OS
- El primer carácter representa *codificación de enteros*
- El segundo carácter representa la *codificación decimal*
- El tercer carácter representa la *codificación de coma flotante*

Esto proporciona un conjunto de doce valores posibles para la propiedad ENCODING .

Hay un valor adicional, la serie NATIVE, que establece los valores de codificación adecuados para la plataforma Java .

Los ejemplos siguientes muestran combinaciones válidas para ENCODING:

```
ENCODING(NNR)  
ENCODING(NATIVE)  
ENCODING(RR3)
```

Referencia de IBM MQ Message Service Client (XMS) for .NET

Esta sección de referencia proporciona información sobre las interfaces de clase IBM MQ Message Service Client (XMS) for .NET (XMS .NET) y sobre las propiedades de objeto definidas por XMS.

.NET interfaces

En esta sección se describen las interfaces de clase .NET y sus propiedades y métodos.

En la tabla siguiente se resumen las interfaces, que están definidas en el espacio de nombres IBM.XMS.

Tabla 907. Resumen de las interfaces de clase .NET	
Interfaz	Descripción
“IBytesMessage” en la página 1990	Un mensaje de bytes es un mensaje cuyo cuerpo está formado por una corriente de bytes.
“IConnection” en la página 2000	Un objeto Connection representa la conexión activa de la aplicación a un servidor de mensajería.
“IConnectionFactory” en la página 2002	Una aplicación utiliza una fábrica de conexiones para crear una conexión.
“IConnectionMetaData” en la página 2004	Un objeto ConnectionMetaData proporciona información sobre una conexión.
“IDestination” en la página 2004	Un destino es el lugar al que una aplicación envía mensajes, o es un origen del cual una aplicación recibe mensajes, o ambas cosas.
“ExceptionListener” en la página 2005	Una aplicación utiliza un escucha de excepción al que se le notificará de forma asíncrona un problema con una conexión.
“IllegalStateException” en la página 2006	XMS emite esta excepción si una aplicación llama a un método en un momento incorrecto o inapropiado, o si XMS no está en un estado adecuado para la operación solicitada.
“InitialContext” en la página 2006	Una aplicación utiliza un objeto InitialContext para crear objetos a partir de definiciones de objeto que se han recuperado de un repositorio de objetos administrados.
“InvalidClientIDException” en la página 2008	XMS emite esta excepción si una aplicación intenta establecer un identificador de cliente para una conexión, pero el identificador de cliente no es válido o ya está en uso.
“InvalidDestinationException” en la página 2008	XMS emite esta excepción si una aplicación especifica un destino que no es válido.
“InvalidSelectorException” en la página 2009	XMS emite esta excepción si una aplicación proporciona una expresión de selector de mensajes cuya sintaxis no es válida.
“IMapMessage” en la página 2009	Un mensaje de correlación es un mensaje cuyo cuerpo está formado por un conjunto de pares de nombre-valor, donde cada valor tiene un tipo de datos asociado.

Tabla 907. Resumen de las interfaces de clase .NET (continuación)

Interfaz	Descripción
“IMessage” en la página 2017	Un objeto Message representa un mensaje que una aplicación envía o recibe. IMessage es una superclase para las clases de mensaje como, por ejemplo, IMapMessage.
“IMessageConsumer” en la página 2023	Una aplicación utiliza un consumidor de mensaje para recibir mensajes enviados a un destino.
“MessageEOFException” en la página 2025	XMS emite esta excepción si XMS encuentra el final de una corriente de mensajes de bytes cuando una aplicación está leyendo el cuerpo de un mensaje de bytes.
“MessageFormatException” en la página 2026	XMS emite esta excepción si XMS encuentra un mensaje con un formato que no es válido.
“IMessageListener (delegado)” en la página 2026	Una aplicación utiliza un escucha de mensajes para recibir mensajes de forma asíncrona.
“MessageNotReadableException” en la página 2026	XMS emite esta excepción si una aplicación intenta leer el cuerpo de un mensaje que es sólo de escritura.
“MessageNotWritableException” en la página 2027	XMS emite esta excepción si una aplicación intenta grabar en el cuerpo de un mensaje que es de sólo lectura.
“IMessageProducer” en la página 2027	Una aplicación utiliza un productor de mensajes para enviar mensajes a un destino.
“IObjectMessage” en la página 2032	Un mensaje de objeto es un mensaje cuyo cuerpo está formado por un objeto Java o .NET serializado.
“IPropertyContext” en la página 2033	IPropertyContext es una superclase abstracta que contiene métodos que obtienen y establecen propiedades. Estos métodos son heredados por otras clases.
“IQueueBrowser” en la página 2042	Una aplicación utiliza un examinador de colas para examinar mensajes en una cola sin eliminarlas.
“Solicitante” en la página 2043	Una aplicación utiliza un solicitante para enviar un mensaje de solicitud y, después, esperar una respuesta y recibirla.
“ResourceAllocationException” en la página 2045	XMS emite esta excepción si XMS no puede asignar los recursos necesarios para un método.
“SecurityException” en la página 2045	XMS emite esta excepción si se rechazan el identificador de usuario y la contraseña proporcionados para autenticar una aplicación. XMS también lanza esta excepción si una comprobación de autoridad falla e impide que se complete un método.
“ISession” en la página 2045	Una sesión es un contexto de hebra única para enviar y recibir mensajes.

Tabla 907. Resumen de las interfaces de clase .NET (continuación)

Interfaz	Descripción
“IStreamMessage” en la página 2055	Un mensaje de corriente de datos es un mensaje cuyo cuerpo está formado por una corriente de valores, donde cada valor tiene un tipo de datos asociado.
“ITextMessage” en la página 2064	Un mensaje de texto es un mensaje cuyo cuerpo está formado por una serie.
“TransactionInProgressException” en la página 2065	XMS emite esta excepción si una aplicación solicita una operación que no es válida porque hay una transacción en curso.
“TransactionRolledBackException” en la página 2065	XMS emite esta excepción si una aplicación llama a Session.commit() para confirmar la transacción actual, pero la transacción se retrotrae.
XMSC	Para .NET, los valores y nombres de propiedad XMS se definen en esta clase como constantes públicas. Si desea más detalles, consulte “Propiedades de objetos XMS” en la página 2068 .
“XMSEException” en la página 2065	Si XMS detecta un error al procesar una llamada a un método .NET, XMS emite una excepción. Una excepción es un objeto que encapsula información sobre el error. Hay distintos tipos de excepción XMS y un objeto XMSEException es sólo un tipo de excepción. Sin embargo, la clase XMSEException es una superclase de las otras clases de excepción XMS. XMS lanza un objeto XMSEException en situaciones donde ninguno de los otros tipos de excepción es apropiado.
“XMSFactoryFactory” en la página 2066	Si una aplicación no está utilizando objetos administrados, utilice esta clase para crear fábricas de conexiones, colas y temas.

La definición de cada método lista los códigos de excepción que podría devolver XMS si detecta un error al procesar una llamada al método. Cada código de excepción se representa mediante su constante con nombre, que tiene una excepción correspondiente.

IBytesMessage

Un mensaje de bytes es un mensaje cuyo cuerpo está formado por una corriente de bytes.

Jerarquía de herencia:

```

IBM.XMS.IPropertyContext
|
+----IBM.XMS.IMessage
|
+----IBM.XMS.IBytesMessage

```

Propiedades de .NET

BodyLength - Obtener longitud de cuerpo

Interfaz:

```
Int64 BodyLength
{
    get;
}
```

Obtener la longitud del cuerpo del mensaje en bytes cuando el cuerpo del mensaje es de solo lectura.

El valor devuelto es la longitud de todo el cuerpo independientemente de la posición actual del cursor para leer el mensaje.

Excepciones:

- XMSEException
- MessageNotReadableException

Métodos

ReadBoolean - Leer valor booleano

Interfaz:

```
Boolean ReadBoolean();
```

Leer un valor booleano de la corriente de datos del mensaje de bytes.

Parámetros:

Ninguna

Devuelve:

El valor booleano que se ha leído.

Excepciones:

- XMSEException
- MessageNotReadableException
- MessageEOFException

ReadSignedByte - Leer byte

Interfaz:

```
Int16 ReadSignedByte();
```

Leer el siguiente byte de la corriente de datos del mensaje de bytes como un entero de 8-bits firmado.

Parámetros:

Ninguna

Devuelve:

El byte que se ha leído.

Excepciones:

- XMSEException
- MessageNotReadableException
- MessageEOFException

ReadBytes - Leer bytes

Interfaz:

```
Int32 ReadBytes(Byte[] array);  
Int32 ReadBytes(Byte[] array, Int32 length);
```

Leer una matriz de bytes de la corriente de datos del mensaje de bytes empezando por la posición actual del cursor.

Parámetros:

array (salida)

El almacenamiento intermedio que va a contener la matriz de bytes que se ha leído. Si el número de bytes que faltan por leer de la corriente de datos antes de la llamada es mayor o igual que la longitud del almacenamiento intermedio, el almacenamiento intermedio se rellena. De lo contrario, el almacenamiento intermedio se rellena de forma parcial con todos los bytes restantes.

Si especifica un puntero nulo en la entrada, el método se salta los bytes sin leerlos. Si el número de bytes que faltan por leer de la corriente de datos antes de la llamada es mayor o igual que la longitud del almacenamiento intermedio, el número de bytes omitidos es igual a la longitud del almacenamiento intermedio. De lo contrario, se omiten todos los bytes restantes. El cursor permanece en la siguiente posición para leer en la corriente de datos del mensaje de byte.

length (entrada)

La longitud del almacenamiento intermedio en bytes

Devuelve:

El número de bytes que se van a leer en el almacenamiento intermedio. Si el almacenamiento intermedio se rellena de forma parcial, el valor es menor que la longitud del almacenamiento intermedio, que indica que no queda ningún byte más para leer. Si no queda ningún bytes para leer en la corriente de datos antes de la llamada, el valor es `XMSC_END_OF_STREAM`.

Si especifica un puntero nulo en la entrada, el método no devuelve ningún valor.

Excepciones:

- `XMSEException`
- `MessageNotReadableException`

ReadChar - Leer carácter

Interfaz:

```
Char ReadChar();
```

Leer los 2 bytes siguientes de la corriente de datos del mensaje de bytes como un carácter.

Parámetros:

Ninguna

Devuelve:

El carácter que se ha leído.

Excepciones:

- `XMSEException`
- `MessageNotReadableException`
- `MessageEOFException`

ReadDouble - Leer número de coma flotante de precisión doble

Interfaz:

```
Double ReadDouble();
```

Leer los 8 bytes siguiente de la corriente de datos del mensaje de bytes como un número de coma flotante de precisión doble.

Parámetros:

Ninguna

Devuelve:

El número de coma flotante de precisión doble que se ha leído.

Excepciones:

- XMSEException
- MessageNotReadableException
- MessageEOFException

ReadFloat - Leer número de coma flotante

Interfaz:

```
Single ReadFloat();
```

Leer los 4 bytes siguientes de la corriente de datos del mensaje de bytes como un número de coma flotante.

Parámetros:

Ninguna

Devuelve:

El número de coma flotante que se ha leído.

Excepciones:

- XMSEException
- MessageNotReadableException
- MessageEOFException

ReadInt - Leer entero

Interfaz:

```
Int32 ReadInt();
```

Leer los 4 bytes siguiente de la corriente de datos del mensaje de bytes como un entero de 32-bits firmado.

Parámetros:

Ninguna

Devuelve:

El entero que se ha leído.

Excepciones:

- XMSEException
- MessageNotReadableException
- MessageEOFException

ReadLong - Leer entero largo

Interfaz:

```
Int64 ReadLong();
```

Leer los 8 bytes siguiente de la corriente de datos del mensaje de bytes como un entero de 64-bits firmado.

Parámetros:

Ninguna

Devuelve:

El entero largo que se ha leído.

Excepciones:

- XMSEException
- MessageNotReadableException
- MessageEOFException

ReadShort - Leer entero corto

Interfaz:

```
Int16 ReadShort();
```

Leer los 2 bytes siguientes de la corriente de datos del mensaje de bytes como un entero de 16-bits firmado.

Parámetros:

Ninguna

Devuelve:

El entero corto que se ha leído.

Excepciones:

- XMSEException
- MessageNotReadableException
- MessageEOFException

ReadByte - Leer byte sin firmar

Interfaz:

```
Byte ReadByte();
```

Leer el siguiente byte de la corriente de datos del mensaje de bytes como un entero de 8-bits sin firmar.

Parámetros:

Ninguna

Devuelve:

El byte que se ha leído.

Excepciones:

- XMSEException
- MessageNotReadableException
- MessageEOFException

ReadUnsignedShort - Leer entero corto sin firmar

Interfaz:

```
Int32 ReadUnsignedShort();
```

Leer los 2 bytes siguiente de la corriente de datos del mensaje de bytes como un entero de 16-bits sin firmar.

Parámetros:

Ninguna

Devuelve:

El entero corto sin firma que se ha leído.

Excepciones:

- XMSEException
- MessageNotReadableException
- MessageEOFException

ReadUTF - Leer serie UTF

Interfaz:

```
String ReadUTF();
```

Leer una serie, codificada en UTF-8, de la corriente de datos del mensaje de bytes.

Nota: Antes de llamar a ReadUTF(), asegúrese de que el cursor del almacenamiento intermedio está apuntado al inicio de la corriente de datos del mensaje de bytes.

Parámetros:

Ninguna

Devuelve:

Un objeto String que encapsula la serie que se ha leído.

Excepciones:

- XMSEException
- MessageNotReadableException
- MessageEOFException

Reset - Restablecer

Interfaz:

```
void Reset();
```

Colocar el cuerpo del mensaje en la modalidad de solo lectura y volver a colocar el cursor al inicio de la corriente de datos del mensaje de bytes.

Parámetros:

Ninguna

Devuelve:

Void

Excepciones:

- XMSEException
- MessageNotReadableException

WriteBoolean - Escribir valor booleano

Interfaz:

```
void WriteBoolean(Boolean value);
```

Escribir un valor booleano en la corriente de datos de mensaje de bytes.

Parámetros:**value (entrada)**

El valor booleano que se va a escribir.

Devuelve:

Void

Excepciones:

- XMSEException
- MessageNotWritableException

WriteByte - Escribir byte

Interfaz:

```
void WriteByte(Byte value);  
void WriteSignedByte(Int16 value);
```

Escribir un byte en la corriente de datos del mensaje de bytes.

Parámetros:**value (entrada)**

El byte que se va a escribir.

Devuelve:

Void

Excepciones:

- XMSEException
- MessageNotWritableException

WriteBytes - Escribir bytes

Interfaz:

```
void WriteBytes(Byte[] value);
```

Escribir una matriz de bytes en la corriente de datos del mensaje de bytes.

Parámetros:**value (entrada)**

La matriz de bytes que se va a escribir.

Devuelve:

Void

Excepciones:

- XMSEException
- MessageNotWritableException

WriteBytes - Escribir matriz de bytes parcial

Interfaz:

```
void WriteBytes(Byte[] value, int offset, int length);
```

Escribir una matriz de bytes parcial en la corriente de datos del mensaje de bytes, tal como lo ha definido la longitud especificada.

Parámetros:**value (entrada)**

La matriz de bytes que se va a escribir.

offset (entrada)

El punto de inicio para la matriz de bytes que se va a escribir.

length (entrada)

El número de bytes que se va a escribir.

Devuelve:

Void

Excepciones:

- XMSEException
- MessageNotWritableException

WriteChar - Escribir carácter

Interfaz:

```
void WriteChar(Char value);
```

Escribir un carácter en la corriente de datos del mensaje de bytes como 2 bytes, primero el byte de orden superior.

Parámetros:**value (entrada)**

El carácter que se va a escribir.

Devuelve:

Void

Excepciones:

- XMSEException
- MessageNotWritableException

WriteDouble - Escribir número de coma flotante de precisión doble

Interfaz:

```
void WriteDouble(Double value);
```

Convertir un número de coma flotante de precisión doble a un entero largo y escribir el entero largo en la corriente de datos del mensaje de bytes como 8 bytes, primero el byte de orden superior.

Parámetros:**value (entrada)**

El número de coma flotante de precisión doble que se va a escribir.

Devuelve:

Void

Excepciones:

- XMSEException
- MessageNotWritableException

WriteFloat - Escribir número de coma flotante

Interfaz:

```
void WriteFloat(Single value);
```

Convertir un número de coma flotante a un entero y escribir el entero en la corriente de datos del mensaje de bytes como 4 bytes, primero el byte de orden superior.

Parámetros:**value (entrada)**

El número de coma flotante que se va a escribir.

Devuelve:

Void

Excepciones:

- XMSEException
- MessageNotWritableException

WriteInt - Escribir entero

Interfaz:

```
void WriteInt(Int32 value);
```

Escribir un entero en la corriente de datos del mensaje de bytes como 4 bytes, primero el byte de orden superior.

Parámetros:**value (entrada)**

El entero que se va a escribir.

Devuelve:

Void

Excepciones:

- XMSEException
- MessageNotWritableException

WriteLong - Escribir entero largo

Interfaz:

```
void WriteLong(Int64 value);
```

Escribir un entero largo en la corriente de datos del mensaje de bytes como 8 bytes, primero el byte de orden superior.

Parámetros:**value (entrada)**

El entero largo que se va a escribir.

Devuelve:

Void

Excepciones:

- XMSEException
- MessageNotWritableException

WriteObject - Escribir objeto

Interfaz:

```
void WriteObject(Object value);
```

Escribir el objeto especificado en la corriente de datos del mensaje de bytes.

Parámetros:**value (entrada)**

El objeto que se va a escribir, que debe hacer referencia a un tipo primitivo.

Devuelve:

Void

Excepciones:

- XMSEException
- MessageNotWritableException

WriteShort - Escribir entero corto

Interfaz:

```
void WriteShort(Int16 value);
```

Escribir un entero corto en la corriente de datos del mensaje de bytes como 2 bytes, primero el byte de orden superior.

Parámetros:**value (entrada)**

El entero corto que se va a escribir.

Devuelve:

Void

Excepciones:

- XMSEException
- MessageNotWritableException

WriteUTF - Escribir serie UTF

Interfaz:

```
void WriteUTF(String value);
```

Escribir una serie, codificada en UTF-8, en la corriente de datos del mensaje de bytes.

Parámetros:**value (entrada)**

Un objeto String que encapsula la serie que se va a escribir.

Devuelve:

Void

Excepciones:

- XMSEException
- MessageNotWritableException

Propiedades y métodos heredados

Las propiedades siguientes se heredan de la interfaz [IMessage](#):

[JMSCorrelationID](#), [JMSDeliveryMode](#), [JMSDestination](#), [JMSExpiration](#), [JMSMessageID](#), [JMSPriority](#), [JMSRedelivered](#), [JMSReplyTo](#), [JMSTimestamp](#), [JMSType](#), [Properties](#)

Los métodos siguientes se heredan de la interfaz [IMessage](#):

[`clearBody`](#), [`clearProperties`](#), [`PropertyExists`](#)

Los métodos siguientes se heredan de la interfaz [IPROPERTYContext](#):

[GetBooleanProperty](#), [GetByteProperty](#), [GetBytesProperty](#), [GetCharProperty](#), [GetDoubleProperty](#), [GetFloatProperty](#), [GetIntProperty](#), [GetLongProperty](#), [GetObjectProperty](#), [GetShortProperty](#), [GetStringProperty](#), [SetBooleanProperty](#), [SetByteProperty](#), [SetBytesProperty](#), [SetCharProperty](#), [SetDoubleProperty](#), [SetFloatProperty](#), [SetIntProperty](#), [SetLongProperty](#), [SetObjectProperty](#), [SetShortProperty](#), [SetStringProperty](#)

ICconnection

Un objeto Connection representa la conexión activa de la aplicación a un servidor de mensajería.

Jerarquía de herencia:

```
IBM.XMS.IPropertyContext
|
+---- IBM.XMS.ICconnection
```

Si desea una lista de las propiedades definidas de XMS de un objeto Connection, consulte [“Propiedades de conexión”](#) en la página 2068.

Propiedades de .NET

ClientID - Obtener y establecer ID de cliente

Interfaz:

```
String ClientID
{
    get;
    set;
}
```

Obtener y establecer el identificador de cliente para la conexión.

El administrador puede preconfigurar el identificador de cliente en objeto ConnectionFactory, o asignarlo estableciendo ClientID.

Un identificador de cliente se utiliza solo para dar soporte a suscripciones duraderas en el dominio de publicación/suscripción y se ignora en el dominio punto a punto.

Si una aplicación establece un identificador de cliente para una conexión, la aplicación debe hacerlo inmediatamente después de crear la conexión y antes de realizar cualquier otra operación en la conexión. Si la aplicación intenta establecer un identificador de cliente después de este punto, la llamada lanza la excepción `IllegalStateException`.

Esta propiedad no es válida para una conexión en tiempo con un intermediario.

Excepciones:

- `XMSEException`
- `IllegalStateException`
- `InvalidClientIDException`

ExceptionListener - Obtener y establecer escucha de excepción

Interfaz:

```
ExceptionListener ExceptionListener
{
    get;
    set;
}
```

Obtener el escucha de excepción que está registro con la conexión y registrar un escucha de excepción con la conexión.

Si no hay registrado ningún escucha de excepción con la conexión, el método devuelve un valor nulo. Si ya hay un escucha de excepción registrado con la conexión, puede cancelar el registro especificando un valor nulo, en lugar del escucha de excepción.

Para obtener más información sobre cómo utilizar escuchas de excepciones, consulte [Utilización de escuchas de mensajes y excepciones en .NET](#).

Excepciones:

- XMSEException

Metadata - Obtener metadatos

Interfaz:

```
ICollectionMetadata Metadata
{
    get;
}
```

Obtener los metadatos para la conexión.

Excepciones:

- XMSEException

Métodos

Close - Cerrar conexión

Interfaz:

```
void Close();
```

Cerrar la conexión.

Si una aplicación intenta cerrar una conexión que ya está cerrando, la llamada se ignora.

Parámetros:

Ninguna

Devuelve:

Void

Excepciones:

- XMSEException

CreateSession - Crear sesión

Interfaz:

```
ISession CreateSession(Boolean transacted,
                        AcknowledgeMode acknowledgeMode);
```

Crear una sesión.

Parámetros:**transacted (entrada)**

El valor True significa que la sesión es una sesión con transacción. El valor False significa que la sesión no es una sesión con transacción.

Para una conexión en tiempo real con un intermediario, el valor debe ser False.

acknowledgeMode (entrada)

Indica cómo se acusa el recibo de los mensajes recibidos por una aplicación. El valor debe adoptar uno de los valores siguientes del enumerador AcknowledgeMode:

AcknowledgeMode.AutoAcknowledge
AcknowledgeMode.ClientAcknowledge
AcknowledgeMode.DupsOkAcknowledge

Para una conexión en tiempo real con un intermediario, el valor debe ser AcknowledgeMode.AutoAcknowledge o AcknowledgeMode.DupsOkAcknowledge

Este parámetro se ignora si la sesión es una sesión con transacción. Para obtener más información sobre las modalidades de acuse de recibo, consulte [Acuse de recibo de mensaje](#).

Devuelve:

El objeto Session

Excepciones:

- XMSEException

Start - Iniciar conexión

Interfaz:

```
void Start();
```

Iniciar o reiniciar la entrega de mensajes entrantes para la conexión. La llamada se ignora si la conexión ya se ha iniciado.

Parámetros:

Ninguna

Devuelve:

Void

Excepciones:

- XMSEException

Stop - Detener conexión

Interfaz:

```
void Stop();
```

Detener la entrega de mensajes entrantes para la conexión. La llamada se ignora, si la conexión ya se ha detenido.

Parámetros:

Ninguna

Devuelve:

Void

Excepciones:

- XMSEException

Propiedades y métodos heredados

Los métodos siguientes se heredan de la interfaz [IPropertyContext](#):

[GetBooleanProperty](#), [GetByteProperty](#), [GetBytesProperty](#), [GetCharProperty](#), [GetDoubleProperty](#), [GetFloatProperty](#), [GetIntProperty](#), [GetLongProperty](#), [GetObjectProperty](#), [GetShortProperty](#), [GetStringProperty](#), [SetBooleanProperty](#), [SetByteProperty](#), [SetBytesProperty](#), [SetCharProperty](#), [SetDoubleProperty](#), [SetFloatProperty](#), [SetIntProperty](#), [SetLongProperty](#), [SetObjectProperty](#), [SetShortProperty](#), [SetStringProperty](#)

IConnectionFactory

Una aplicación utiliza una fábrica de conexiones para crear una conexión.

Jerarquía de herencia:

```
IBM.XMS.IPropertyContext
|
+----IBM.XMS.IConnectionFactory
```

Si desea una lista de las propiedades definidas de XMS de un objeto `ConnectionFactory`, consulte [“Propiedades de ConnectionFactory”](#) en la página 2069.

Métodos

CreateConnection - Crear fábrica de conexiones (utilizando la identidad de usuario predeterminada)

Interfaz:

```
IConnection CreateConnection();
```

Crear una fábrica de conexiones con las propiedades predeterminadas.

Si se está conectando a IBM MQ y XMSC_USERID no está establecida, el gestor de colas utiliza el ID de usuario del usuario conectado, de forma predeterminada. Si necesita una autenticación de nivel de conexión adicional de usuarios individuales, puede escribir una salida de autenticación de cliente, que se configura en IBM MQ.

Parámetros:

Ninguna

Excepciones:

- `XMSEException`

CreateConnection - Crear conexión (utilizando una identidad de usuario especificada)

Interfaz:

```
IConnection CreateConnection(String userId, String password);
```

Crear una conexión utilizando una identidad de usuario especificada.

Si se está conectando a IBM MQ y XMSC_USERID no está establecida, el gestor de colas utiliza el ID de usuario del usuario conectado, de forma predeterminada. Si necesita una autenticación de nivel de conexión adicional de usuarios individuales, puede escribir una salida de autenticación de cliente, que se configura en IBM MQ.

La conexión se crea en la modalidad detenida. No se entrega ningún mensaje hasta que la aplicación llama a **Connection.start()**.

Parámetros:

userId (entrada)

Un objeto `String` que encapsula el identificador de usuario que se va a utilizar para autenticar la aplicación. Si proporciona un valor nulo, se realiza un intento de crear la conexión sin autenticación.

password (entrada)

Un objeto `String` que encapsula la contraseña que se va a utilizar para autenticar la aplicación. Si proporciona un valor nulo, se realiza un intento de crear la conexión sin autenticación.

Devuelve:

El objeto `Connection`.

Excepciones:

- `XMSEException`
- `XMS_X_SECURITY_EXCEPTION`

Propiedades y métodos heredados

Los métodos siguientes se heredan de la interfaz [IPropertyContext](#):

[GetBooleanProperty](#), [GetByteProperty](#), [GetBytesProperty](#), [GetCharProperty](#), [GetDoubleProperty](#), [GetFloatProperty](#), [GetIntProperty](#), [GetLongProperty](#), [GetObjectProperty](#), [GetShortProperty](#),

[GetStringProperty](#), [SetBooleanProperty](#), [SetByteProperty](#), [SetBytesProperty](#), [SetCharProperty](#), [SetDoubleProperty](#), [SetFloatProperty](#), [SetIntProperty](#), [SetLongProperty](#), [SetObjectProperty](#), [SetShortProperty](#), [SetStringProperty](#)

ICollectionMetaData

Un objeto ConnectionMetaData proporciona información sobre una conexión.

Jerarquía de herencia:

```
IBM.XMS.IPropertyContext
|
+---- IBM.XMS.ICollectionMetaData
```

Si desea una lista de las propiedades definidas de XMS de un objeto ConnectionMetaData, consulte [“Propiedades de ConnectionMetaData”](#) en la página 2074.

Propiedades de .NET

JMSXPropertyNames - Obtener propiedades de mensaje definidas de JMS

Interfaz:

```
System.Collections.IEnumerator JMSXPropertyNames
{
    get;
}
```

Devolver una enumeración de los nombres de las propiedades de mensaje definidas por JMS soportadas por la conexión.

Las propiedades de mensaje definidas de JMS no están soportadas por una conexión en tiempo real con un intermediario.

Excepciones:

- XMSEException

Propiedades y métodos heredados

Los métodos siguientes se heredan de la interfaz IPropertyContext:

[GetBooleanProperty](#), [GetByteProperty](#), [GetBytesProperty](#), [GetCharProperty](#), [GetDoubleProperty](#), [GetFloatProperty](#), [GetIntProperty](#), [GetLongProperty](#), [GetObjectProperty](#), [GetShortProperty](#), [GetStringProperty](#), [SetBooleanProperty](#), [SetByteProperty](#), [SetBytesProperty](#), [SetCharProperty](#), [SetDoubleProperty](#), [SetFloatProperty](#), [SetIntProperty](#), [SetLongProperty](#), [SetObjectProperty](#), [SetShortProperty](#), [SetStringProperty](#)

IDestination

Un destino es el lugar al que una aplicación envía mensajes, o es un origen del cual una aplicación recibe mensajes, o ambas cosas.

Jerarquía de herencia:

```
IBM.XMS.IPropertyContext
|
+---- IBM.XMS.IDestination
```

Si desea una lista de las propiedades definidas de XMS de un objeto Destination, consulte [“Propiedades de Destination”](#) en la página 2074.

Propiedades de .NET

Name - Obtener nombre de destino

Interfaz:

```
String Name
{
    get;
}
```

Obtener el nombre del destino. El nombre es una serie que encapsula el nombre de una cola, o bien el nombre de un tema.

Excepciones:

- XMSEException

TypeId - Obtener tipo de destino

Interfaz:

```
DestinationType TypeId
{
    get;
}
```

Obtener el tipo del destino. El tipo del destino es uno de los valores siguientes:

DestinationType.Queue
DestinationType.Topic

Excepciones:

- XMSEException

Propiedades y métodos heredados

Los métodos siguientes se heredan de la interfaz `IPropertyContext`:

[GetBooleanProperty](#), [GetByteProperty](#), [GetBytesProperty](#), [GetCharProperty](#), [GetDoubleProperty](#), [GetFloatProperty](#), [GetIntProperty](#), [GetLongProperty](#), [GetObjectProperty](#), [GetShortProperty](#), [GetStringProperty](#), [SetBooleanProperty](#), [SetByteProperty](#), [SetBytesProperty](#), [SetCharProperty](#), [SetDoubleProperty](#), [SetFloatProperty](#), [SetIntProperty](#), [SetLongProperty](#), [SetObjectProperty](#), [SetShortProperty](#), [SetStringProperty](#)

ExceptionListener

Una aplicación utiliza un escucha de excepción al que se le notificará de forma asíncrona un problema con una conexión.

Jerarquía de herencia:

Ninguna

Si una aplicación utiliza una conexión solo para consumir mensajes de forma asíncrona, y sin ninguna otra finalidad, la única forma a través de la cual la aplicación puede obtener información sobre un problema con la conexión es utilizar un escucha de excepción. En otras situaciones, un escucha de excepción puede proporcionar una forma más inmediata de obtener información sobre un problema con una conexión que esperar hasta la siguiente llamada síncrona a XMS.

Delegado

ExceptionListener - Escucha de excepción

Interfaz:

```
public delegate void ExceptionListener(Exception ex)
```

Notifique a la aplicación un problema con una conexión.

Los métodos que implementan este delegado se pueden registrar con la conexión.

Para obtener más información sobre cómo utilizar escuchas de excepciones, consulte [Utilización de escuchas de mensajes y excepciones en .NET](#).

Parámetros:

exception (entrada)

Un puntero para una excepción creada por XMS.

Devuelve:

Void

IllegalStateException

XMS emite esta excepción si una aplicación llama a un método en un momento incorrecto o inapropiado, o si XMS no está en un estado adecuado para la operación solicitada.

Jerarquía de herencia:

```
IBM.XMS.XMSException
|
+----IBM.XMS.Exception
|
+----IBM.XMS.IllegalStateException
```

Propiedades y métodos heredados

Los métodos siguientes se heredan de la interfaz [XMSEException](#):

[GetErrorCode](#), [GetLinkedException](#)

InitialContext

Una aplicación utiliza un objeto InitialContext para crear objetos a partir de definiciones de objeto que se han recuperado de un repositorio de objetos administrados.

Jerarquía de herencia:

Ninguna

Propiedades de .NET

Environment - Obtener el entorno

Interfaz:

```
Hashtable Environment
{
    get;
}
```

Obtener el entorno.

Excepciones:

- Las excepciones son específicas al servicio de directorio que se está utilizando.

Constructores

InitialContext - Crear contexto inicial

Interfaz:

```
InitialContext(Hashtable env);
```

Crear un objeto InitialContext.

Parámetros:

La información necesaria para establecer una conexión al repositorio de objetos administrados se proporciona al constructor en una tabla hash de entorno.

Excepciones:

- XMSEException

Métodos

AddToEnvironment - Añadir una nueva propiedad al entorno

Interfaz:

```
Object AddToEnvironment(String propName, Object propVal);
```

Añadir una nueva propiedad al entorno.

Parámetros:

propName (entrada)

Un objeto String que encapsula el nombre de la propiedad que se va a añadir.

propVal (entrada)

El valor de la propiedad que se va añadir.

Devuelve:

El valor antiguo de la propiedad.

Excepciones:

- Las excepciones son específicas al servicio de directorio que se está utilizando.

Close - Cerrar este contexto

Interfaz:

```
void Close()
```

Cerrar este contexto.

Parámetros:

Ninguna

Devuelve:

Ninguna

Excepciones:

- Las excepciones son específicas al servicio de directorio que se está utilizando.

Lookup - Buscar objeto en contexto inicial

Interfaz:

```
Object Lookup(String name);
```

Crear un objeto a partir de una definición de objeto que se ha recuperado del repositorio de objetos administrados.

Parámetros:

name (entrada)

Un objeto String que encapsula el nombre del objeto administrado que se va a recuperar. El nombre puede ser un nombre sencillo o un nombre complejo. Para obtener más detalles, consulte [Recuperación de objetos administrados](#).

Devuelve:

IConnectionFactory o IDestination, en función del tipo de objeto que se está recuperando. Si la función puede acceder al directorio, pero no puede encontrar el objeto necesario, se devuelve un valor nulo.

Excepciones:

- Las excepciones son específicas al servicio de directorio que se está utilizando.

RemoveFromEnvironment - Eliminar una propiedad del entorno

Interfaz:

```
Object RemoveFromEnvironment(String propName);
```

Eliminar una propiedad del entorno.

Parámetros:**propName (entrada)**

Un objeto String que encapsula el nombre de la propiedad que se va a eliminar.

Devuelve:

El objeto que se ha eliminado.

Excepciones:

- Las excepciones son específicas al servicio de directorio que se está utilizando.

InvalidClientIDException

XMS emite esta excepción si una aplicación intenta establecer un identificador de cliente para una conexión, pero el identificador de cliente no es válido o ya está en uso.

Jerarquía de herencia:

```
IBM.XMS.XMSEException
|
+----IBM.XMS.XMSEException
|
+----IBM.XMS.InvalidClientIDException
```

Propiedades y métodos heredados

Los métodos siguientes se heredan de la interfaz [XMSEException](#):

[GetErrorCode](#), [GetLinkedException](#)

InvalidDestinationException

XMS emite esta excepción si una aplicación especifica un destino que no es válido.

Jerarquía de herencia:

```
IBM.XMS.XMSEException
|
+----IBM.XMS.XMSEException
|
+----IBM.XMS.InvalidDestinationException
```

Propiedades y métodos heredados

Los métodos siguientes se heredan de la interfaz [XMSEException](#):

[GetErrorCode](#), [GetLinkedException](#)

InvalidSelectorException

XMS emite esta excepción si una aplicación proporciona una expresión de selector de mensajes cuya sintaxis no es válida.

Jerarquía de herencia:

```
IBM.XMS.XMSException
|
+----IBM.XMS.XMSException
|
+----IBM.XMS.InvalidSelectorException
```

Propiedades y métodos heredados

Los métodos siguientes se heredan de la interfaz [XMSException](#):

[GetErrorCode](#), [GetLinkedException](#)

IMapMessage

Un mensaje de correlación es un mensaje cuyo cuerpo está formado por un conjunto de pares de nombre-valor, donde cada valor tiene un tipo de datos asociado.

Jerarquía de herencia:

```
IBM.XMS.IPropertyContext
|
+----IBM.XMS.IMessage
|
+----IBM.XMS.IMapMessage
```

Cuando una aplicación obtiene el valor del par nombre-valor, el valor se puede convertir mediante XMS a otro tipo de datos. Para obtener más información sobre esta forma de conversión implícita, consulte la información sobre los mensajes de correlación en [El cuerpo de un mensaje XMS](#).

Propiedades de .NET

MapNames - Obtener nombres de correlación

Interfaz:

```
System.Collections.IEnumerator MapNames
{
    get;
}
```

Obtener una enumeración de los nombres en el cuerpo del mensaje de correlación.

Excepciones:

- [XMSException](#)

Métodos

GetBoolean - Obtener valor booleano

Interfaz:

```
Boolean GetBoolean(String name);
```

Obtener el valor booleano identificado por el nombre del cuerpo del mensaje de correlación.

Parámetros:

name (entrada)

Un objeto String que encapsula el nombre que identifica el valor booleano.

Devuelve:

El valor booleano recuperado del cuerpo del mensaje de correlación.

Excepciones:

- XMSEException

GetByte - Obtener byte

Interfaz:

```
Byte    GetByte(String name);  
Int16   GetSignedByte(String name);
```

Obtener el byte identificado por el nombre del cuerpo del mensaje de correlación.

Parámetros:**name (entrada)**

Un objeto String que encapsula el nombre que identifica el byte.

Devuelve:

El byte recuperado del cuerpo del mensaje de correlación. No se realiza ninguna conversión de datos en el byte.

Excepciones:

- XMSEException

GetBytes - Obtener bytes

Interfaz:

```
Byte[]  GetBytes(String name);
```

Obtener la matriz de bytes identificada por el nombre del cuerpo del mensaje de correlación.

Parámetros:**name (entrada)**

Un objeto String que encapsula el nombre que identifica la matriz de bytes.

Devuelve:

El número de bytes de la matriz.

Excepciones:

- XMSEException

GetChar - Obtener carácter

Interfaz:

```
Char    GetChar(String name);
```

Obtener el carácter identificado por el nombre del cuerpo del mensaje de correlación.

Parámetros:**name (entrada)**

Un objeto String que encapsula el nombre que identifica el carácter.

Devuelve:

El carácter recuperado del cuerpo del mensaje de correlación.

Excepciones:

- XMSEException

GetDouble - Obtener número de coma flotante de precisión doble

Interfaz:

```
Double GetDouble(String name);
```

Obtener el número de coma flotante de precisión doble identificado por el nombre del cuerpo del mensaje de correlación.

Parámetros:

name (entrada)

Un objeto String que encapsula el nombre que identifica el número de coma flotante de precisión doble.

Devuelve:

El número de coma flotante de precisión doble recuperado del cuerpo del mensaje de correlación.

Excepciones:

- XMSEException

GetFloat - Obtener número de coma flotante

Interfaz:

```
Single GetFloat(String name);
```

Obtener el número de coma flotante identificado por el nombre del cuerpo del mensaje de correlación.

Parámetros:

name (entrada)

Un objeto String que encapsula el nombre que identifica el número de coma flotante.

Devuelve:

El número de coma flotante recuperado del cuerpo del mensaje de correlación.

Excepciones:

- XMSEException

GetInt - Obtener entero

Interfaz:

```
Int32 GetInt(String name);
```

Obtener el entero identificado por el nombre del cuerpo del mensaje de correlación.

Parámetros:

name (entrada)

Un objeto String que encapsula el nombre que identifica el entero.

Devuelve:

El entero recuperado del cuerpo del mensaje de correlación.

Excepciones:

- XMSEException

GetLong - Obtener entero largo

Interfaz:

```
Int64 GetLong(String name);
```

Obtener el entero largo identificado por el nombre del cuerpo del mensaje de correlación.

Parámetros:**name (entrada)**

Un objeto String que encapsula el nombre que identifica el entero largo.

Devuelve:

El entero largo recuperado del cuerpo del mensaje de correlación.

Excepciones:

- XMSEException

GetObject - Obtener objeto

Interfaz:

```
Object GetObject(String name);
```

Obtener una referencia al valor de un par de nombre-valor del cuerpo del mensaje de correlación. El par nombre-valor se identifica por el nombre.

Parámetros:**name (entrada)**

Un objeto String que encapsula el nombre del par nombre-valor.

Devuelve:

El valor, que es uno de los tipos de objeto siguientes:

Boolean
Byte
Byte[]
Char
Double
Single
Int32
Int64
Int16
String

Excepciones:

XMSEException

GetShort - Obtener entero corto

Interfaz:

```
Int16 GetShort(String name);
```

Obtener el entero corto identificado por el nombre del cuerpo del mensaje de correlación.

Parámetros:**name (entrada)**

Un objeto String que encapsula el nombre que identifica el entero corto.

Devuelve:

El entero corto recuperado del cuerpo del mensaje de correlación.

Excepciones:

- XMSEException

GetString - Obtener serie

Interfaz:

```
String GetString(String name);
```

Obtener la serie identificada por el nombre del cuerpo del mensaje de correlación.

Parámetros:

name (entrada)

Un objeto String que encapsula el nombre que identifica la serie en el cuerpo del mensaje de correlación.

Devuelve:

Un objeto String que encapsula la serie recuperada del cuerpo del mensaje de correlación. Si es necesaria una conversión de datos, este valor es la serie después de la conversión.

Excepciones:

- XMSEException

ItemExists - Comprobar si existe el par nombre-valor

Interfaz:

```
Boolean ItemExists(String name);
```

Comprobar si el cuerpo del mensaje de correlación contiene un par nombre-valor con el nombre especificado.

Parámetros:

name (entrada)

Un objeto String que encapsula el nombre del par nombre-valor.

Devuelve:

- True, si el cuerpo del mensaje de correlación contiene un par nombre-valor con el nombre especificado.
- False, si el cuerpo del mensaje de correlación no contiene un par nombre-valor con el nombre especificado.

Excepciones:

- XMSEException

SetBoolean - Establecer valor booleano

Interfaz:

```
void SetBoolean(String name, Boolean value);
```

Establecer un valor booleano en el cuerpo del mensaje de correlación.

Parámetros:

name (entrada)

Un objeto String que encapsula el nombre para identificar el valor booleano en el cuerpo del mensaje de correlación.

value (entrada)

El valor booleano que se va a establecer.

Devuelve:

Void

Excepciones:

- XMSEException

SetByte - Establecer byte

Interfaz:

```
void SetByte(String name, Byte value);  
void SetSignedByte(String name, Int16 value);
```

Establecer un byte en el cuerpo del mensaje de correlación.

Parámetros:

name (entrada)

Un objeto String que encapsula el nombre para identificar el byte en el cuerpo del mensaje de correlación.

value (entrada)

El byte que se va a establecer.

Devuelve:

Void

Excepciones:

- XMSEException

SetBytes - Establecer bytes

Interfaz:

```
void SetBytes(String name, Byte[] value);
```

Establecer una matriz de bytes en el cuerpo del mensaje de correlación.

Parámetros:

name (entrada)

Un objeto String que encapsula el nombre para identificar la matriz de bytes en el cuerpo del mensaje de correlación.

value (entrada)

La matriz de bytes que se va a establecer.

Devuelve:

Void

Excepciones:

- XMSEException

SetChar - Establecer carácter

Interfaz:

```
void SetChar(String name, Char value);
```

Establecer un carácter de 2-bytes en el cuerpo del mensaje de correlación.

Parámetros:

name (entrada)

Un objeto String que encapsula el nombre para identificar el carácter en el cuerpo del mensaje de correlación.

value (entrada)

El carácter que se va a establecer.

Devuelve:

Void

Excepciones:

- XMSEException

SetDouble - Establecer número de coma flotante de precisión doble

Interfaz:

```
void SetDouble(String name, Double value);
```

Establecer un número de coma flotante de precisión doble en el cuerpo del mensaje de correlación.

Parámetros:**name (entrada)**

Un objeto String que encapsula el nombre para identificar el número de coma flotante de precisión doble en el cuerpo del mensaje de correlación.

value (entrada)

El número de coma flotante de precisión doble que se va a establecer.

Devuelve:

Void

Excepciones:

- XMSEException

SetFloat - Establecer número de coma flotante

Interfaz:

```
void SetFloat(String name, Single value);
```

Establecer un número de coma flotante en el cuerpo del mensaje de correlación.

Parámetros:**name (entrada)**

Un objeto String que encapsula el nombre para identificar el número de coma flotante en el cuerpo del mensaje de correlación.

value (entrada)

El número de coma flotante que se va a establecer.

Devuelve:

Void

Excepciones:

- XMSEException

SetInt - Establecer entero

Interfaz:

```
void SetInt(String name, Int32 value);
```

Establecer un entero en el cuerpo del mensaje de correlación.

Parámetros:**name (entrada)**

Un objeto String que encapsula el nombre para identificar el entero en el cuerpo del mensaje de correlación.

value (entrada)

El entero que se va a establecer.

Devuelve:

Void

Excepciones:

- XMSEException

SetLong - Establecer entero largo

Interfaz:

```
void SetLong(String name, Int64 value);
```

Establecer un entero largo en el cuerpo del mensaje de correlación.

Parámetros:**name (entrada)**

Un objeto String que encapsula el nombre para identificar el entero largo en el cuerpo del mensaje de correlación.

value (entrada)

El entero largo que se va a establecer.

Devuelve:

Void

Excepciones:

- XMSEException

SetObject - Establecer objeto

Interfaz:

```
void SetObject(String name, Object value);
```

Establecer un valor, que debe ser un tipo primitivo XMS, en el cuerpo del mensaje de correlación.

Parámetros:**name (entrada)**

Un objeto String que encapsula el nombre para identificar el valor en el cuerpo del mensaje de correlación.

value (entrada)

Una matriz de bytes que contiene el valor que se va a establecer.

Devuelve:

Void

Excepciones:

- XMSEException

SetShort - Establecer entero corto

Interfaz:

```
void SetShort(String name, Int16 value);
```

Establecer un entero corto en el cuerpo del mensaje de correlación.

Parámetros:**name (entrada)**

Un objeto String que encapsula el nombre para identificar el entero corto en el cuerpo del mensaje de correlación.

value (entrada)

El entero corto que se va a establecer.

Devuelve:

Void

Excepciones:

- XMSEException

SetString - Establecer serie

Interfaz:

```
void SetString(String name, String value);
```

Establecer una serie en el cuerpo del mensaje de correlación.

Parámetros:

name (entrada)

Un objeto String que encapsula el nombre para identificar la serie en el cuerpo del mensaje de correlación.

value (entrada)

Un objeto String que encapsula la serie que se va a establecer.

Devuelve:

Void

Excepciones:

- XMSEException

Propiedades y métodos heredados

Las propiedades siguientes se heredan de la interfaz IMessage:

JMSCorrelationID, JMSDeliveryMode, JMSDestination, JMSExpiration, JMSMessageID, JMSPriority, JMSRedelivered, JMSReplyTo, JMSTimestamp, JMSType, Properties

Los métodos siguientes se heredan de la interfaz IMessage:

`clearBody`, `clearProperties`, `PropertyExists`

Los métodos siguientes se heredan de la interfaz IPropertyContext:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

IMessage

Un objeto Message representa un mensaje que una aplicación envía o recibe. IMessage es una superclase para las clases de mensaje como, por ejemplo, IMapMessage.

Jerarquía de herencia:

```
IBM.XMS.IPropertyContext
|
+---- IBM.XMS.IMessage
```

Para obtener una lista de los campos de cabecera de mensaje JMS en un objeto Message, consulte Campos de cabecera de un mensaje XMS. Para obtener una lista de las propiedades definidas por JMS de un objeto Message, consulte Propiedades definidas por JMS de un mensaje. Para obtener una lista de las propiedades definidas de IBM de un objeto Message, consulte Propiedades definidas por IBM de un mensaje. Para obtener una lista de las propiedades JMS_IBM_MQMD* para el objeto Message, consulte “Propiedades JMS_IBM_MQMD*” en la página 2078

Los mensajes son suprimidos por el recopilador de basura. Cuando se suprime un mensaje, esta acción libera los recursos que estaba utilizando.

Propiedades de .NET

GetJMSCorrelationID - Obtener y establecer JMSCorrelationID

Interfaz:

```
String JMSCorrelationID
{
    get;
    set;
}
```

Obtener y establecer el identificador de correlación del mensaje como un objeto String.

Excepciones:

- XMSEException

JMSDeliveryMode - Obtener y establecer JMSDeliveryMode

Interfaz:

```
DeliveryMode JMSDeliveryMode
{
    get;
    set;
}
```

Obtener y establecer la modalidad de entrega del mensaje.

La modalidad de entrega del mensaje adopta uno de los valores siguientes:

DeliveryMode.Persistent
DeliveryMode.NonPersistent

Para un mensaje recién creado que no se ha enviado, la modalidad de entrega es DeliveryMode.Persistent, excepto para una conexión en tiempo real a un intermediario para el cual la modalidad de entrega es DeliveryMode.NonPersistent. Para un mensaje que se recibe, el método devuelve la modalidad de entrega que estableció la llamada IMessageProducer.send() cuando se envió el mensaje, a menos que la aplicación receptora cambie la modalidad de entrega estableciendo JMSDeliveryMode.

Excepciones:

- XMSEException

JMSDestination - Obtener y establecer JMSDestination

Interfaz:

```
IDestination JMSDestination
{
    get;
    set;
}
```

Obtener y establecer el destino del mensaje.

El destino se establece mediante la llamada IMessageProducer.send() cuando se envía el mensaje. Se ignora el valor de JMSDestination. Sin embargo, puede utilizar JMSDestination para cambiar el destino de un mensaje que se ha recibido.

Para un mensaje recién creado que no se ha enviado, el método devuelve un objeto Destination nulo, a menos que la aplicación emisora establezca un destino estableciendo JMSDestination. Para un mensaje que se ha recibido, el método devuelve un objeto Destination para el destino que estableció la llamada IMessageProducer.send() cuando se envió el mensaje, a menos que la aplicación receptora cambie el destino estableciendo JMSDestination.

Excepciones:

- XMSEException

JMSEExpiration - Obtener y establecer JMSEExpiration

Interfaz:

```
Int64 JMSEExpiration
{
    get;
    set;
}
```

Obtener y establecer la hora de caducidad del mensaje.

La hora de caducidad se establece mediante la llamada `IMessageProducer.send()` cuando se envía el mensaje. Su valor se calcula añadiendo el tiempo de vida, tal como lo especifica la aplicación emisora, a la hora cuando se envía el mensaje. La hora de caducidad se expresa en milisegundos desde las 00:00:00 GMT el 1 de enero de 1970.

Para un mensaje recién creado que no se ha enviado, la hora de caducidad es 0, a menos que la aplicación emisora defina una hora de caducidad diferente estableciendo `JMSEExpiration`. Para un mensaje que se ha recibido, el método devuelve la hora caducidad que estableció la llamada `IMessageProducer.send()` cuando se envió el mensaje, a menos que la aplicación receptora cambie la hora de caducidad estableciendo `JMSEExpiration`.

Si la hora de caducidad es 0, la llamada `IMessageProducer.send()` establece la hora de caducidad en 0 para indicar que el mensaje no caduca.

XMS descarta mensajes caducados y no los entrega a aplicaciones.

Excepciones:

- XMSEException

JMSMessageID - Obtener y establecer JMSMessageID

Interfaz:

```
String JMSMessageID
{
    get;
    set;
}
```

Obtener y establecer el identificador de mensaje del mensaje como un objeto de serie que encapsula el identificador de mensaje.

El identificador de mensaje se establece mediante la llamada `IMessageProducer.send()` cuando se envía el mensaje. Para un mensaje que se ha recibido, el método devuelve el identificador de mensaje que estableció la llamada `IMessageProducer.send()` cuando se envió el mensaje, a menos que la aplicación receptora cambie el identificador de mensaje estableciendo `JMSMessageID`.

Si el mensaje no tiene ningún identificador de mensaje, el método devuelve un valor nulo.

Excepciones:

- XMSEException

JMSPriority - Obtener y establecer JMSPriority

Interfaz:

```
Int32 JMSPriority
{
    get;
    set;
}
```

Obtener y establecer la prioridad del mensaje.

La prioridad se establece mediante la llamada `IMessageProducer.send()` cuando se envía el mensaje. El valor es un entero dentro del rango de 0, la prioridad inferior, a 9, la prioridad superior.

Para un mensaje recién creado que no se ha enviado, la prioridad es 4, a menos que la aplicación emisora establezca una prioridad diferente estableciendo `JMSPriority`. Para un mensaje que se ha recibido, el método devuelve la prioridad que estableció la llamada `IMessageProducer.send()` cuando se envió el mensaje, a menos que la aplicación receptora cambie la prioridad estableciendo `JMSPriority`.

Excepciones:

- `XMSEException`

JMSRedelivered - Obtener y establecer JMSRedelivered

Interfaz:

```
Boolean JMSRedelivered
{
    get;
    set;
}
```

Obtener una indicación de si el mensaje se está volviendo a entregar e indicar si el mensaje se está volviendo a entregar. La indicación se establece mediante la llamada `IMessageConsumer.receive()` cuando se recibe el mensaje.

Esta propiedad tiene los valores siguientes:

- `True`, si el mensaje se está volviendo a entregar.
- `False`, si el mensaje no se está volviendo a entregar.

Para una conexión en tiempo real a un intermediario, el valor siempre es `False`.

Una indicación de reentrega establecida por `JMSRedelivered` antes de que se envíe el mensaje es ignorada por la llamada `IMessageProducer.send()` cuando se envía el mensaje, y la llamada `IMessageConsumer.receive()` la ignora y la sustituye cuando se recibe el mensaje. Sin embargo, puede utilizar `JMSRedelivered` para cambiar la indicación para un mensaje que se ha recibido.

Excepciones:

- `XMSEException`

JMSReplyTo - Obtener y establecer JMSReplyTo

Interfaz:

```
IDestination JMSReplyTo
{
    get;
    set;
}
```

Obtener y establecer el destino adonde se va a enviar una respuesta al mensaje.

El valor de esta propiedad es un objeto `Destination` para el destino adonde se va a enviar una respuesta al mensaje. Un objeto `Destination` nulo significa que no se espera ninguna respuesta.

Excepciones:

- `XMSEException`

JMSTimestamp - Obtener y establecer JMSTimestamp

Interfaz:

```
Int64 JMSTimestamp
{
    get;
}
```



```
    set;
}
```

Obtener y establecer la hora cuando se envió el mensaje.

La indicación de fecha y hora se establece mediante la llamada `IMessageProducer.send()` cuando se envía el mensaje y se expresa en milisegundos desde las 00:00:00 GMT del 1 de enero de 1970.

Para un mensaje recién creado que no se ha enviado, la indicación de fecha y hora es 0, a menos que la aplicación emisora establezca una indicación de fecha y hora diferente estableciendo `JMSTimestamp`. Para un mensaje que se ha recibido, el método devuelve la indicación de fecha y hora que estableció la llamada `IMessageProducer.send()` cuando se envió el mensaje, a menos que la aplicación receptora cambie la indicación de fecha y hora estableciendo `JMSTimestamp`.

Excepciones:

- `XMSEException`

Notas:

1. Si la indicación de fecha y hora no está definida, el método devuelve 0 pero no lanza ninguna excepción.

JMSType - Obtener y establecer JMSType

Interfaz:

```
String JMSType
{
    get;
    set;
}
```

Obtener y establecer el tipo de mensaje.

El valor de `JMSType` es una serie que encapsula el tipo del mensaje. Si es necesaria una conversión de datos, este valor es el tipo después de la conversión.

Excepciones:

- `XMSEException`

PropertyNames - Obtener propiedades

Interfaz:

```
System.Collections.IEnumerator PropertyNames
{
    get;
}
```

Obtener una enumeración de las propiedades de nombres del mensaje.

Excepciones:

- `XMSEException`

Métodos

Acknowledge - Acusar recibo

Interfaz:

```
void Acknowledge();
```

Acusar recibo de este mensaje y todos los mensajes recibidos previamente con acuse de recibo recibidos por la sesión.

Una aplicación puede llamar a este método si la modalidad de acuse de recibo de la sesión es AcknowledgeMode.ClientAcknowledge. Se ignoran las llamadas al método si la sesión tiene cualquier otra modalidad de acuse de recibo o es una sesión con transacción.

Los mensajes que se han recibido pero que no tienen acuse de recibo se podrían volver a entregar.

Si desea más información sobre cómo acusar recibo de mensajes, consulte ../develop/xms_cmescack.dita#xms_cmescack.

Parámetros:

Ninguna

Devuelve:

Void

Excepciones:

- XMSEException
- IllegalStateException

ClearBody - Borrar cuerpo

Interfaz:

```
void ClearBody();
```

Borrar el cuerpo del mensaje. Los campos de cabecera y las propiedades de mensaje no se borran.

Si una aplicación borra un cuerpo del mensaje, el cuerpo permanece en el mismo estado que un cuerpo vacío en un mensaje recién creado. El estado de un cuerpo vacío en un mensaje recién creado depende del tipo de cuerpo del mensaje. Para obtener más información, consulte [El cuerpo de un mensaje XMS](#).

Una aplicación puede borrar un cuerpo del mensaje en cualquier momento, independientemente del estado en el que se encuentra el cuerpo. Si un cuerpo del mensaje es de solo lectura, la única forma en la que una aplicación puede escribir en el cuerpo es que la aplicación borre primero el cuerpo.

Parámetros:

Ninguna

Devuelve:

Void

Excepciones:

- XMSEException

ClearProperties - Borrar propiedades

Interfaz:

```
void ClearProperties();
```

Borrar las propiedades del mensaje. Los campos de cabecera y el cuerpo del mensaje no se borran.

Si una aplicación borra las propiedades de un mensaje, las propiedades pasan a poderse leer y escribir.

Una aplicación puede borrar las propiedades de un mensaje en cualquier momento, independientemente del estado en que están las propiedades. Si las propiedades de un mensaje son de solo lectura, la única forma en la que las propiedades se pueden grabar es que la aplicación borre primero las propiedades.

Parámetros:

Ninguna

Devuelve:

Void

Excepciones:

- XMSEException

PropertyExists - Comprobar si existe la propiedad

Interfaz:

```
Boolean PropertyExists(String propertyName);
```

Comprobar si el mensaje tiene una prioridad con el nombre especificado.

Parámetros:

propertyName (entrada)

Un objeto String que encapsula el nombre de la propiedad.

Devuelve:

- True, si el mensaje tiene una propiedad con el nombre especificado.
- False, si el mensaje no tiene una propiedad con el nombre especificado.

Excepciones:

- XMSEException

Propiedades y métodos heredados

Los métodos siguientes se heredan de la interfaz [IPropertyContext](#):

[GetBooleanProperty](#), [GetByteProperty](#), [GetBytesProperty](#), [GetCharProperty](#), [GetDoubleProperty](#), [GetFloatProperty](#), [GetIntProperty](#), [GetLongProperty](#), [GetObjectProperty](#), [GetShortProperty](#), [GetStringProperty](#), [SetBooleanProperty](#), [SetByteProperty](#), [SetBytesProperty](#), [SetCharProperty](#), [SetDoubleProperty](#), [SetFloatProperty](#), [SetIntProperty](#), [SetLongProperty](#), [SetObjectProperty](#), [SetShortProperty](#), [SetStringProperty](#)

IMessageConsumer

Una aplicación utiliza un consumidor de mensaje para recibir mensajes enviados a un destino.

Jerarquía de herencia:

```
IBM.XMS.IPropertyContext
|
+----IBM.XMS.IMessageConsumer
```

Para obtener una lista de las propiedades definidas de XMS de un objeto MessageConsumer, consulte [“Propiedades de MessageConsumer”](#) en la página 2082.

Propiedades de .NET

MessageListener - Obtener y establecer escucha de mensajes

Interfaz:

```
MessageListener MessageListener
{
    get;
    set;
}
```

Obtener el escucha de mensajes que está registrado con el consumidor de mensajes y registrar un escucha de mensajes con el consumidor de mensajes.

Si no hay ningún escucha de mensajes registrado con el consumidor de mensajes, MessageListener es nulo. Si un escucha de mensajes ya está registrado con el consumidor de mensajes, puede cancelar el registro especificando en su lugar un valor nulo.

Para obtener más información sobre el uso de los detectores de mensajes, consulta [la sección «Uso de detectores de mensajes y excepciones»](#) en .NET.

Excepciones:

- XMSEException

MessageSelector - Obtener selector de mensajes

Interfaz:

```
String MessageSelector  
{  
    get;  
}
```

Obtener el selector de mensajes para el consumidor de mensajes. el valor de retorno es un objeto String que encapsula la expresión de selector de mensajes. Si es necesaria la conversión de datos, este valor es la expresión de selector de mensajes después de la conversión. Si el consumidor de mensajes no tiene un selector de mensajes, el valor de MessageSelector es un objeto String nulo.

Excepciones:

- XMSEException

Métodos

Close - Cerrar consumidor de mensajes

Interfaz:

```
void Close();
```

Cerrar el consumidor de mensajes.

Si una aplicación intenta cerrar un consumidor de mensajes que ya está cerrado, la llamada se ignora.

Parámetros:

Ninguna

Devuelve:

Void

Excepciones:

- XMSEException

Recibir - Recibir

Interfaz:

```
IMessage Receive();
```

Reciba el siguiente mensaje para el consumidor de mensajes. La llamada espera un mensaje de forma indefinida, o hasta que se cierra el consumidor de mensajes.

Parámetros:

Ninguna

Devuelve:

Un puntero del objeto Message. Si el consumidor de mensajes se cierra cuando la llamada está esperando un mensaje, el método devuelve un puntero de un objeto Message nulo.

Excepciones:

- XMSEException

Recibir - Recibir (con un intervalo de espera)

Interfaz:

```
IMessage Receive(Int64 delay);
```

Reciba el siguiente mensaje para el consumidor de mensajes. La llamada solo espera un mensaje durante un periodo de tiempo especificado, o hasta que se cierra el consumidor de mensajes.

Parámetros:

delay (entrada)

El tiempo, en milisegundos, que espera un mensaje la llamada. Si especifica un intervalo de espera de 0, la llamada espera un mensaje de forma indefinida.

Devuelve:

Un puntero del objeto Message. Si no llega ningún mensaje durante el intervalo de espera, o si el consumidor de mensajes se cierra mientras la llamada está esperando un mensaje, el método devuelve un puntero de un objeto Message nulo, pero no lanza ninguna excepción.

Excepciones:

- XMSEException

ReceiveNoWait - Recibir sin espera

Interfaz:

```
IMessage ReceiveNoWait();
```

Recibir el siguiente mensaje para el consumidor de mensajes si uno está disponible de forma inmediata.

Parámetros:

Ninguna

Devuelve:

Un puntero de un objeto Message. Si no hay ningún mensaje disponible de forma inmediata, el método devuelve un puntero de un objeto Message nulo.

Excepciones:

- XMSEException

Propiedades y métodos heredados

Los métodos siguientes se heredan de la interfaz `IPropertyContext`:

[GetBooleanProperty](#), [GetByteProperty](#), [GetBytesProperty](#), [GetCharProperty](#), [GetDoubleProperty](#), [GetFloatProperty](#), [GetIntProperty](#), [GetLongProperty](#), [GetObjectProperty](#), [GetShortProperty](#), [GetStringProperty](#), [SetBooleanProperty](#), [SetByteProperty](#), [SetBytesProperty](#), [SetCharProperty](#), [SetDoubleProperty](#), [SetFloatProperty](#), [SetIntProperty](#), [SetLongProperty](#), [SetObjectProperty](#), [SetShortProperty](#), [SetStringProperty](#)

MessageEOFException

XMS emite esta excepción si XMS encuentra el final de una corriente de mensajes de bytes cuando una aplicación está leyendo el cuerpo de un mensaje de bytes.

Jerarquía de herencia:

```
IBM.XMS.XMSEException
|
+----IBM.XMS.XMSEException
|
+----IBM.XMS.MessageEOFException
```

Propiedades y métodos heredados

Los métodos siguientes se heredan de la interfaz [XMSEException](#):

[GetErrorCode](#), [GetLinkedException](#)

MessageFormatException

XMS emite esta excepción si XMS encuentra un mensaje con un formato que no es válido.

Jerarquía de herencia:

```
IBM.XMS.XMSEException
|
+----IBM.XMS.XMSEException
|
+----IBM.XMS.MessageFormatException
```

Propiedades y métodos heredados

Los métodos siguientes se heredan de la interfaz [XMSEException](#):

[GetErrorCode](#), [GetLinkedException](#)

IMessageListener (delegado)

Una aplicación utiliza un escucha de mensajes para recibir mensajes de forma asíncrona.

Jerarquía de herencia:

Ninguna

Delegado

MessageListener - Escucha de mensajes

Interfaz:

```
public delegate void MessageListener(IMessage msg);
```

Entregue un mensaje de forma asíncrona al consumidor de mensajes.

Los métodos que implementan este delegado se pueden registrar con la conexión.

Para obtener más información sobre cómo utilizar escuchas de mensajes, consulte [Utilización de escuchas de mensajes y excepciones en .NET](#).

Parámetros:

mesg (entrada)

El objeto Message.

Devuelve:

Void

MessageNotReadableException

XMS emite esta excepción si una aplicación intenta leer el cuerpo de un mensaje que es sólo de escritura.

Jerarquía de herencia:

```
IBM.XMS.XMSEException
|
+----IBM.XMS.XMSEException
|
+----IBM.XMS.MessageNotReadableException
```

Propiedades y métodos heredados

Los métodos siguientes se heredan de la interfaz [XMSEException](#):

[GetErrorCode](#), [GetLinkedException](#)

MessageNotWritableException

XMS emite esta excepción si una aplicación intenta grabar en el cuerpo de un mensaje que es de sólo lectura.

Jerarquía de herencia:

```
IBM.XMS.XMSEException
|
+----IBM.XMS.XMSEException
|
+----IBM.XMS.MessageNotWritableException
```

Propiedades y métodos heredados

Los métodos siguientes se heredan de la interfaz [XMSEException](#):

[GetErrorCode](#), [GetLinkedException](#)

IMessageProducer

Una aplicación utiliza un productor de mensajes para enviar mensajes a un destino.

Jerarquía de herencia:

```
IBM.XMS.IPropertyContext
|
+----IBM.XMS.IMessageProducer
```

Si desea una lista de las propiedades definidas de XMS de un objeto MessageProducer, consulte [“Propiedades de MessageProducer”](#) en la página 2082.

Propiedades de .NET

DeliveryMode - Obtener y establecer modalidad de entrega predeterminada

Interfaz:

```
DeliveryMode DeliveryMode
{
    get;
    set;
}
```

Obtener y establecer la modalidad de entrega predeterminada para mensajes enviados por el productor de mensajes.

La modalidad de entrega predeterminada tiene uno de los valores siguientes:

`DeliveryMode.Persistent`
`DeliveryMode.NonPersistent`

Para una conexión en tiempo real con un intermediario, el valor debe ser `DeliveryMode.NonPersistent`.

El valor predeterminado es `DeliveryMode.Persistent`, excepto para una conexión en tiempo real con un intermediario para el cual el valor predeterminado es `DeliveryMode.NonPersistent`.

Excepciones:

- `XMSEException`

Destination - Obtener destino

Interfaz:

```
IDestination Destination
{
    get;
}
```

Obtener el destino para el productor de mensajes.

Parámetros:

Ninguna

Devuelve:

El objeto Destination. Si el productor de mensajes no tiene un destino, el método devuelve un objeto Destination nulo.

Excepciones:

- XMSEException

DisableMsgID - Obtener y establecer Inhabilitar distintivo de ID de mensaje

Interfaz:

```
Boolean DisableMessageID
{
    get;
    set;
}
```

Obtener una indicación sobre si una aplicación receptora requiere que se incluyan identificadores de mensaje en los mensajes enviados por el productor de mensajes, e indicar si una aplicación receptora requiere que se incluyan identificadores de mensaje en los mensajes enviados por el productor de mensajes.

En una conexión con un gestor de colas, o en una conexión en tiempo real con un intermediario, este distintivo se ignora. En una conexión con un bus de integración de servicios, se respeta el distintivo.

DisabledMsgID tiene los valores siguientes:

- True, si una aplicación receptora no requiere que se incluyan identificadores de mensaje en los mensajes enviados por el productor de mensajes.
- False, si una aplicación requiere que se incluyan identificadores de mensajes en los mensajes enviados por el productor de mensajes.

Excepciones:

- XMSEException

DisableMsgTS - Obtener y establecer Inhabilitar distintivo de indicación de fecha y hora

Interfaz:

```
Boolean DisableMessageTimestamp
{
    get;
    set;
}
```

Obtener una indicación sobre si una aplicación receptora requiere que se incluyan indicaciones de fecha y hora en los mensajes enviados por el productor de mensajes, e indicar si una aplicación receptora requiere que se incluyan indicaciones de fecha y hora en los mensajes enviados por el productor de mensajes.

En una conexión de tiempo real con un intermediario, se ignora este distintivo. En una conexión con un gestor de colas, o en una conexión con un bus de integración de servicios, se respeta el distintivo.

DisableMsgTS tiene los valores siguientes:

- **True**, si una aplicación receptora no requiere que se incluyan indicaciones de fecha y hora en los mensajes enviados por el productor de mensajes.
- **False**, si una aplicación receptora requiere que se incluyan indicaciones de fecha y hora en los mensajes enviados por el productor de mensajes.

Devuelve:

Excepciones:

- XMSEException

Priority - Obtener y establecer prioridad predeterminada

Interfaz:

```
Int32 Priority
{
    get;
    set;
}
```

Obtener y establecer la prioridad predeterminada para los mensajes enviados por el productor de mensajes.

El valor de la prioridad de mensaje predeterminada es un entero dentro del rango de 0, la prioridad más baja, a 9, la prioridad más alta.

En una conexión en tiempo real con un intermediario, se ignora la prioridad de un mensaje.

Excepciones:

- XMSEException

TimeToLive - Obtener y establecer tiempo de vida

Interfaz:

```
Int64 TimeToLive
{
    get;
    set;
}
```

Obtener y establecer el periodo de tiempo predeterminado que existe un mensaje antes de caducar.

El tiempo se mide a partir del momento cuando el productor de mensajes envía el mensaje y el tiempo de vida predeterminado en milisegundos. Un valor de 0 indica que un mensaje no caduca nunca.

Para una conexión en tiempo real con un intermediario, este valor siempre es 0.

Excepciones:

- XMSEException

Métodos

Close - Cerrar productor de mensajes

Interfaz:

```
void Close();
```

Cerrar el productor de mensajes.

Si una aplicación intenta cerrar un productor de mensajes que ya está cerrado, se ignora la llamada.

Parámetros:

Ninguna

Devuelve:

Void

Excepciones:

- XMSEException

Send - Enviar

Interfaz:

```
void Send(IMessage msg) ;
```

Enviar un mensaje al destino que se ha especificado al crear el productor de mensajes. Envíe el mensaje utilizando la modalidad de entrega, la prioridad y el tiempo de vida predeterminados del productor de mensajes.

Parámetros:**msg (entrada)**

El objeto Message.

Devuelve:

Void

Excepciones:

- XMSEException
- MessageFormatException
- InvalidDestinationException

Send - Enviar (especificando una modalidad de entrega, prioridad y tiempo de vida)

Interfaz:

```
void Send(IMessage msg,  
          DeliveryMode deliveryMode,  
          Int32 priority,  
          Int64 timeToLive);
```

Enviar un mensaje al destino que se ha especificado al crear el productor de mensajes. Envíe el mensaje utilizando la modalidad de entrega, prioridad y tiempo de vida especificados.

Parámetros:**msg (entrada)**

El objeto Message.

deliveryMode (entrada)

La modalidad de entrega para el mensaje, que debe adoptar uno de los valores siguientes:

DeliveryMode.Persistent

DeliveryMode.NonPersistent

Para una conexión en tiempo real con un intermediario, el valor debe ser DeliveryMode.NonPersistent.

priority (entrada)

La prioridad del mensaje. El valor puede ser un entero dentro del rango de 0, para la prioridad más baja, a 9, para la prioridad más alta. En una conexión en tiempo real con un intermediario, se ignora el valor.

timeToLive (entrada)

El tiempo de vida para el mensaje en milisegundos. Un valor de 0 significa que el mensaje no caduca nunca. Para una conexión en tiempo real con un intermediario, el valor debe ser 0.

Devuelve:

Void

Excepciones:

- XMSEException
- MessageFormatException
- InvalidDestinationException
- IllegalStateException

Send - Enviar (a un destino especificado)

Interfaz:

```
void Send(IDestination dest, IMessage msg) ;
```

Enviar un mensaje a un destino especificado si está utilizando un productor de mensajes para el que no se ha especificado ningún destino al crear el productor de mensajes. Envíe el mensaje utilizando la modalidad de entrega, la prioridad y el tiempo de vida predeterminados del productor de mensajes.

Normalmente, puede especificar un destino cuando crea un productor de mensajes, pero si no lo hace, debe especificar un destino cada vez que envíe un mensaje.

Parámetros:**dest (entrada)**

El objeto Destination.

msg (entrada)

El objeto Message.

Devuelve:

Void

Excepciones:

- XMSEException
- MessageFormatException
- InvalidDestinationException

Send - Enviar (a un destino especificado, especificando una modalidad de entrega, prioridad y tiempo de vida)

Interfaz:

```
void Send(IDestination dest,
          IMessage msg,
          DeliveryMode deliveryMode,
          Int32 priority,
          Int64 timeToLive) ;
```

Enviar un mensaje a un destino especificado si está utilizando un productor de mensajes para el que no se ha especificado ningún destino al crear el productor de mensajes. Envíe el mensaje utilizando la modalidad de entrega, prioridad y tiempo de vida especificados.

Normalmente, puede especificar un destino cuando crea un productor de mensajes, pero si no lo hace, debe especificar un destino cada vez que envíe un mensaje.

Parámetros:**dest (entrada)**

El objeto Destination.

msg (entrada)

El objeto Message.

deliveryMode (entrada)

La modalidad de entrega para el mensaje, que debe adoptar uno de los valores siguientes:

```
DeliveryMode.Persistent
DeliveryMode.NonPersistent
```

Para una conexión en tiempo real con un intermediario, el valor debe ser `DeliveryMode.NonPersistent`.

priority (entrada)

La prioridad del mensaje. El valor puede ser un entero dentro del rango de 0, para la prioridad más baja, a 9, para la prioridad más alta. En una conexión en tiempo real con un intermediario, se ignora el valor.

timeToLive (entrada)

El tiempo de vida para el mensaje en milisegundos. Un valor de 0 significa que el mensaje no caduca nunca. Para una conexión en tiempo real con un intermediario, el valor debe ser 0.

Devuelve:

Void

Excepciones:

- `XMSEException`
- `MessageFormatException`
- `InvalidDestinationException`
- `IllegalStateException`

Propiedades y métodos heredados

Los métodos siguientes se heredan de la interfaz `IPROPERTYContext`:

`GetBooleanProperty`, `GetByteProperty`, `GetBytesProperty`, `GetCharProperty`, `GetDoubleProperty`, `GetFloatProperty`, `GetIntProperty`, `GetLongProperty`, `GetObjectProperty`, `GetShortProperty`, `GetStringProperty`, `SetBooleanProperty`, `SetByteProperty`, `SetBytesProperty`, `SetCharProperty`, `SetDoubleProperty`, `SetFloatProperty`, `SetIntProperty`, `SetLongProperty`, `SetObjectProperty`, `SetShortProperty`, `SetStringProperty`

IOBJECTMessage

Un mensaje de objeto es un mensaje cuyo cuerpo está formado por un objeto Java o .NET serializado.

Jerarquía de herencia:

```
IBM.XMS.IPROPERTYContext
|
+---- IBM.XMS.IMessage
|
+---- IBM.XMS.IOBJECTMessage
```

Propiedades de .NET

Object - Obtener y establecer objeto como bytes

Interfaz:

```
System.Object Object
{
    get;
    set;
}

Byte[] GetObject();
```

Obtener y establecer el objeto que forma el cuerpo del mensaje de objeto.

Excepciones:

- XMSEException
- MessageNotReadableException
- MessageEOFException
- MessageNotWritableException

Propiedades y métodos heredados

Las propiedades siguientes se heredan de la interfaz IMessage:

JMSCorrelationID, JMSDeliveryMode, JMSDestination, JMSExpiration, JMSMessageID, JMSPriority, JMSRedelivered, JMSReplyTo, JMSTimestamp, JMSType, Properties

Los métodos siguientes se heredan de la interfaz IMessage:

clearBody `, ` clearProperties `, ` PropertyExists `

Los métodos siguientes se heredan de la interfaz IPropertyContext:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

IPropertyContext

IPropertyContext es una superclase abstracta que contiene métodos que obtienen y establecen propiedades. Estos métodos son heredados por otras clases.

Jerarquía de herencia:

Ninguna

Métodos

GetBooleanProperty - Obtener propiedad booleana

Interfaz:

```
Boolean GetBooleanProperty(String property_name);
```

Obtener el valor de la propiedad booleana con el nombre especificado.

Parámetros:**property_name (entrada)**

Un objeto String que encapsula el nombre de la propiedad.

Devuelve:

El valor de la propiedad.

Contexto de hebras

Se determina mediante la subclase.

Excepciones:

- XMSEException

GetByteProperty - Obtener propiedad de byte

Interfaz:

```
Byte    GetByteProperty(String property_name) ;  
Int16   GetSignedByteProperty(String property_name) ;
```

Obtener el valor de la propiedad de byte identificada por el nombre.

Parámetros:

property_name (entrada)

Un objeto String que encapsula el nombre de la propiedad.

Devuelve:

El valor de la propiedad.

Contexto de hebras

Se determina mediante la subclase.

Excepciones:

- XMSEException

GetBytesProperty - Obtener propiedad de matriz de bytes

Interfaz:

```
Byte[] GetBytesProperty(String property_name) ;
```

Obtener el valor de la propiedad de matriz de bytes identificada por el nombre.

Parámetros:

property_name (entrada)

Un objeto String que encapsula el nombre de la propiedad.

Devuelve:

El número de bytes de la matriz.

Contexto de hebras

Se determina mediante la subclase.

Excepciones:

- XMSEException

GetCharProperty - Obtener propiedad de carácter

Interfaz:

```
Char GetCharProperty(String property_name) ;
```

Obtener el valor de la propiedad de carácter de 2-bytes identificada por el nombre.

Parámetros:

property_name (entrada)

Un objeto String que encapsula el nombre de la propiedad.

Devuelve:

El valor de la propiedad.

Contexto de hebras

Se determina mediante la subclase.

Excepciones:

- XMSEException

GetDoubleProperty - Obtener propiedad de coma flotante de precisión doble

Interfaz:

```
Double GetDoubleProperty(String property_name) ;
```

Obtener el valor de la propiedad de coma flotante de precisión doble identificada por el nombre.

Parámetros:**property_name (entrada)**

Un objeto String que encapsula el nombre de la propiedad.

Devuelve:

El valor de la propiedad.

Contexto de hebras

Se determina mediante la subclase.

Excepciones:

- XMSEException

GetFloatProperty - Obtener propiedad de coma flotante

Interfaz:

```
Single GetFloatProperty(String property_name) ;
```

Obtener el valor de la propiedad de coma flotante identificada por el nombre.

Parámetros:**property_name (entrada)**

Un objeto String que encapsula el nombre de la propiedad.

Devuelve:

El valor de la propiedad.

Contexto de hebras

Se determina mediante la subclase.

Excepciones:

- XMSEException

GetIntProperty - GetIntProperty

Interfaz:

```
Int32 GetIntProperty(String property_name) ;
```

Obtener el valor de la propiedad de entero identificada por el nombre.

Parámetros:**property_name (entrada)**

Un objeto String que encapsula el nombre de la propiedad.

Devuelve:

El valor de la propiedad.

Contexto de hebras

Se determina mediante la subclase.

Excepciones:

- XMSEException

GetLongProperty - Obtener propiedad de entero largo

Interfaz:

```
Int64 GetLongProperty(String property_name) ;
```

Obtener el valor de la propiedad de entero largo identificada por el nombre.

Parámetros:**property_name (entrada)**

Un objeto String que encapsula el nombre de la propiedad.

Devuelve:

El valor de la propiedad.

Contexto de hebras

Se determina mediante la subclase.

Excepciones:

- XMSEException

GetObjectProperty - Obtener propiedad de objeto

Interfaz:

```
Object GetObjectProperty( String property_name) ;
```

Obtener el valor y el tipo de datos de la propiedad identificada por el nombre.

Parámetros:**property_name (entrada)**

Un objeto String que encapsula el nombre de la propiedad.

Devuelve:

El valor de la propiedad, que adopta uno de los tipos de objeto siguientes:

Boolean
Byte
Byte[]
Char
Double
Single
Int32
Int64
Int16
String

Contexto de hebras

Se determina mediante la subclase.

Excepciones:

- XMSEException

GetShortProperty - Obtener propiedad de entero corto

Interfaz:

```
Int16 GetShortProperty(String property_name) ;
```

Obtener el valor de la propiedad de entero corto identificada por el nombre.

Parámetros:**property_name (entrada)**

Un objeto String que encapsula el nombre de la propiedad.

Devuelve:

El valor de la propiedad.

Contexto de hebras

Se determina mediante la subclase.

Excepciones:

- XMSEException

GetStringProperty - GetStringProperty

Interfaz:

```
String GetStringProperty(String property_name) ;
```

Obtener el valor de la propiedad de serie identificada por el nombre.

Parámetros:**property_name (entrada)**

Un objeto String que encapsula el nombre de la propiedad.

Devuelve:

Un objeto String que encapsula la serie que es el valor de la propiedad. Si es necesaria una conversión de datos, este valor es la serie después de la conversión.

Contexto de hebras

Se determina mediante la subclase.

Excepciones:

- XMSEException

SetBooleanProperty - Establecer propiedad booleana

Interfaz:

```
void SetBooleanProperty( String property_name, Boolean value) ;
```

Establecer el valor de la propiedad booleana identificada por el nombre.

Parámetros:**property_name (entrada)**

Un objeto String que encapsula el nombre de la propiedad.

value (entrada)

El valor de la propiedad.

Devuelve:

Void

Contexto de hebras

Se determina mediante la subclase.

Excepciones:

- XMSEException
- MessageNotWritableException

SetByteProperty - Establecer propiedad de byte

Interfaz:

```
void SetByteProperty( String property_name, Byte value) ;  
void SetSignedByteProperty( String property_name, Int16 value) ;
```

Establecer el valor de la propiedad de byte identificada por el nombre.

Parámetros:**property_name (entrada)**

Un objeto String que encapsula el nombre de la propiedad.

value (entrada)

El valor de la propiedad.

Devuelve:

Void

Contexto de hebras

Se determina mediante la subclase.

Excepciones:

- XMSEException
- MessageNotWritableException

SetBytesProperty - Establecer propiedad de matriz de bytes

Interfaz:

```
void SetBytesProperty( String property_name, Byte[] value ) ;
```

Establecer el valor de la propiedad de matriz de bytes identificada por el nombre.

Parámetros:**property_name (entrada)**

Un objeto String que encapsula el nombre de la propiedad.

value (entrada)

El valor de la propiedad, que es una matriz de bytes.

Devuelve:

Void

Contexto de hebras

Se determina mediante la subclase.

Excepciones:

- XMSEException
- MessageNotWritableException

SetCharProperty - Establecer propiedad de carácter

Interfaz:

```
void SetCharProperty( String property_name, Char value) ;
```

Establecer el valor de la propiedad de carácter de 2-bytes identificada por el nombre.

Parámetros:**property_name (entrada)**

Un objeto String que encapsula el nombre de la propiedad.

value (entrada)

El valor de la propiedad.

Devuelve:

Void

Contexto de hebras

Se determina mediante la subclase.

Excepciones:

- XMSEException
- MessageNotWritableException

SetDoubleProperty - Establecer propiedad de coma flotante de precisión doble

Interfaz:

```
void SetDoubleProperty( String property_name, Double value) ;
```

Establecer el valor de la propiedad de coma flotante de precisión doble identificada por el nombre.

Parámetros:

property_name (entrada)

Un objeto String que encapsula el nombre de la propiedad.

value (entrada)

El valor de la propiedad.

Devuelve:

Void

Contexto de hebras

Se determina mediante la subclase.

Excepciones:

- XMSEException
- MessageNotWritableException

SetFloatProperty - Establecer propiedad de coma flotante

Interfaz:

```
void SetFloatProperty( String property_name, Single value) ;
```

Establecer el valor de la propiedad de coma flotante identificada por el nombre.

Parámetros:

property_name (entrada)

Un objeto String que encapsula el nombre de la propiedad.

value (entrada)

El valor de la propiedad.

Devuelve:

Void

Contexto de hebras

Se determina mediante la subclase.

Excepciones:

- XMSEException
- MessageNotWritableException

SetIntProperty - Establecer propiedad de entero

Interfaz:

```
void SetIntProperty( String property_name, Int32 value) ;
```

Establecer el valor de la propiedad de entero identificada por el nombre.

Parámetros:

property_name (entrada)

Un objeto String que encapsula el nombre de la propiedad.

value (entrada)

El valor de la propiedad.

Devuelve:

Void

Contexto de hebras

Se determina mediante la subclase.

Excepciones:

- XMSEException
- MessageNotWritableException

SetLongProperty - Establecer propiedad de entero largo

Interfaz:

```
void SetLongProperty( String property_name, Int64 value) ;
```

Establecer el valor de la propiedad de entero largo identificada por el nombre.

Parámetros:**property_name (entrada)**

Un objeto String que encapsula el nombre de la propiedad.

value (entrada)

El valor de la propiedad.

Devuelve:

Void

Contexto de hebras

Se determina mediante la subclase.

Excepciones:

- XMSEException
- MessageNotWritableException

SetObjectProperty - Establecer propiedad de objeto

Interfaz:

```
void SetObjectProperty( String property_name, Object value) ;
```

Establecer el valor y el tipo de datos de una propiedad identificada por el nombre.

Parámetros:**property_name (entrada)**

Un objeto String que encapsula el nombre de la propiedad.

objectType (entrada)

El valor de la propiedad, que debe adoptar uno de los tipos de objeto siguientes:

Boolean
Byte
Byte[]
Char
Double
Single
Int32
Int64
Int16
String

value (entrada)

El valor de la propiedad como una matriz de bytes.

length (entrada)

El número de bytes de la matriz.

Devuelve:

Void

Contexto de hebras

Se determina mediante la subclase.

Excepciones:

- XMSEException
- MessageNotWritableException

SetShortProperty - Establecer propiedad de entero corto

Interfaz:

```
void SetShortProperty( String property_name, Int16 value) ;
```

Establecer el valor de la propiedad de entero corto identificada por el nombre.

Parámetros:**property_name (entrada)**

Un objeto String que encapsula el nombre de la propiedad.

value (entrada)

El valor de la propiedad.

Devuelve:

Void

Contexto de hebras

Se determina mediante la subclase.

Excepciones:

- XMSEException
- MessageNotWritableException

SetStringProperty - Establecer propiedad de serie

Interfaz:

```
void SetStringProperty( String property_name, String value);
```

Establecer el valor de la propiedad de serie identificada por el nombre.

Parámetros:**property_name (entrada)**

Un objeto String que encapsula el nombre de la propiedad.

value (entrada)

Un objeto String que encapsula la serie que es el valor de la propiedad.

Devuelve:

Void

Contexto de hebras

Se determina mediante la subclase.

Excepciones:

- XMSEException
- MessageNotWritableException

IQueueBrowser

Una aplicación utiliza un examinador de colas para examinar mensajes en una cola sin eliminarlas.

Jerarquía de herencia:

```
IBM.XMS.IPropertyContext
System.Collections.IEnumerable
|
+---- IBM.XMS.IQueueBrowser
```

Propiedades de .NET

MessageSelector - Obtener selector de mensajes

Interfaz:

```
String MessageSelector
{
    get;
}
```

Obtener el selector de mensajes para el examinador de colas.

El selector de mensajes es un objeto String que encapsula la expresión de selector de mensajes. Si es necesaria la conversión de datos, este valor es la expresión de selector de mensajes después de la conversión. Si el examinador de colas no tiene un selector de mensajes, el método devuelve un objeto String nulo.

Excepciones:

- XMSEException

Queue - Obtener cola

Interfaz:

```
IDestination Queue
{
    get;
}
```

Obtener la cola asociada al examinador de colas como un objeto de destino que representa la cola.

Excepciones:

- XMSEException

Métodos

Close - Cerrar examinador de colas

Interfaz:

```
void Close();
```

Cerrar el examinador de colas.

Si una aplicación intenta cerrar un examinador de colas que está cerrado, la llamada se ignora.

Parámetros:

Ninguna

Devuelve:

Void

Excepciones:

- XMSEException

GetEnumerator - Obtener mensajes

Interfaz:

```
IEnumerator GetEnumerator();
```

Obtener una lista de los mensajes de la cola.

El método devuelve un enumerador que encapsula una lista de objetos Message. El orden de los objetos Message es el mismo que el orden en el cual se recuperarán los mensajes de la cola. La aplicación puede utilizar el enumerador para examinar cada mensaje a su vez.

El enumerador se actualiza de forma dinámica a medida que los mensajes se colocan en la cola y se eliminan de la cola. Cada vez que la aplicación llama a `IEnumerator.MoveNext()` para examinar el siguiente mensaje en la cola, el mensaje refleja el contenido actual de la cola.

Si una aplicación llama a este método más de una vez para un examinador de colas, cada llamada devuelve un nuevo enumerador. Por lo tanto, la aplicación puede utilizar más de un enumerador para examinar los mensajes de una cola y mantener varias posiciones dentro de la cola.

Parámetros:

Ninguna

Devuelve:

El objeto Iterator.

Excepciones:

- XMSEException

Propiedades y métodos heredados

Los métodos siguientes se heredan de la interfaz [IPropertyContext](#):

[GetBooleanProperty](#), [GetByteProperty](#), [GetBytesProperty](#), [GetCharProperty](#), [GetDoubleProperty](#), [GetFloatProperty](#), [GetIntProperty](#), [GetLongProperty](#), [GetObjectProperty](#), [GetShortProperty](#), [GetStringProperty](#), [SetBooleanProperty](#), [SetByteProperty](#), [SetBytesProperty](#), [SetCharProperty](#), [SetDoubleProperty](#), [SetFloatProperty](#), [SetIntProperty](#), [SetLongProperty](#), [SetObjectProperty](#), [SetShortProperty](#), [SetStringProperty](#)

Solicitante

Una aplicación utiliza un solicitante para enviar un mensaje de solicitud y, después, esperar una respuesta y recibirla.

Jerarquía de herencia:

Ninguna

Constructores

Requestor - Crear solicitante

Interfaz:

```
Requestor(ISession sess, IDestination dest);
```

Crear un solicitante.

Parámetros:**sess (entrada)**

Un objeto Session. La sesión no debe ser una sesión con transacción y debe tener una de las modalidades de acuse de recibo siguientes:

AcknowledgeMode.AutoAcknowledge
AcknowledgeMode.DupsOkAcknowledge

dest (entrada)

Un objeto Destination que representa el destino adonde la aplicación puede enviar mensajes de solicitud.

Contexto de hebras

La sesión asociada al solicitante

Excepciones:

- XMSEException

Métodos

Close - Cerrar solicitante

Interfaz:

```
void Close();
```

Cerrar el solicitante.

Si una aplicación intenta cerrar un solicitante que ya está cerrado, se ignora la llamada.

Nota: Cuando una aplicación cierra un solicitante, la sesión asociada no se cierra. En este sentido, XMS se comporta de forma diferente en comparación con JMS.

Parámetros:

Ninguna

Devuelve:

Void

Contexto de hebras

Cualquiera

Excepciones:

- XMSEException

Solicitar - Solicitar respuesta

Interfaz:

```
IMessage Request(IMessage requestMessage);
```

Enviar un mensaje de solicitud y, después, esperar una respuesta y recibirla de la aplicación que recibe el mensaje de solicitud.

Se bloquea una llamada a este método hasta que se recibe una respuesta o hasta que finaliza la sesión, lo que se produzca antes.

Parámetros:**requestMessage (entrada)**

El objeto Message que encapsula el mensaje de solicitud.

Devuelve:

Un puntero del objeto Message que encapsula el mensaje de respuesta.

Contexto de hebras

La sesión asociada al solicitante

Excepciones:

- `XMSEException`

ResourceAllocationException

XMS emite esta excepción si XMS no puede asignar los recursos necesarios para un método.

Jerarquía de herencia:

```
IBM.XMS.XMSEException
|
+----IBM.XMS.XMSEException
|
+----IBM.XMS.ResourceAllocationException
```

Propiedades y métodos heredados

Los métodos siguientes se heredan de la interfaz [XMSEException](#):

[GetErrorCode](#), [GetLinkedException](#)

SecurityException

XMS emite esta excepción si se rechazan el identificador de usuario y la contraseña proporcionados para autenticar una aplicación. XMS también lanza esta excepción si una comprobación de autoridad falla e impide que se complete un método.

Jerarquía de herencia:

```
IBM.XMS.XMSEException
|
+----IBM.XMS.XMSEException
|
+----IBM.XMS.SecurityException
```

Propiedades y métodos heredados

Los métodos siguientes se heredan de la interfaz [XMSEException](#):

[GetErrorCode](#), [GetLinkedException](#)

ISession

Una sesión es un contexto de hebra única para enviar y recibir mensajes.

Jerarquía de herencia:

```
IBM.XMS.IPropertyContext
|
+----IBM.XMS.ISession
```

Si desea una lista de propiedades definidas de XMS de un objeto Session, consulte [“Propiedades de sesión”](#) en la página 2082.

Propiedades de .NET

AcknowledgeMode - Obtener modalidad de acuse de recibo

Interfaz:

```
AcknowledgeMode AcknowledgeMode
{
```

```
    get;
}
```

Obtener la modalidad de acuse de recibo para la sesión.

La modalidad de acuse de recibo se especifica cuando se crea la sesión.

Si la sesión no es una sesión con transacción, la modalidad de acuse de recibo es uno de los valores siguientes:

```
AcknowledgeMode.AutoAcknowledge
AcknowledgeMode.ClientAcknowledge
AcknowledgeMode.DupsOkAcknowledge
```

Para obtener más información sobre las modalidades de acuse de recibo, consulte [Acuse de recibo de mensaje](#).

Una sesión que es una sesión con transacción no tiene ninguna modalidad de acuse de recibo. Si la sesión es una sesión con transacción, en su lugar el método devuelve `AcknowledgeMode.SessionTransacted`.

Excepciones:

- `XMSEException`

Transacted - Determinar si es transaccional

Interfaz:

```
Boolean Transacted
{
    get;
}
```

Determinar si la sesión es una sesión con transacción.

El estado con transacción es:

- Verdadero, si la sesión es transaccional.
- Falso, si la sesión no es transaccional.

Para una conexión en tiempo real con un intermediario, el método siempre devuelve `False`.

Excepciones:

- `XMSEException`

Métodos

Close - Cerrar sesión

Interfaz:

```
void Close();
```

Cerrar la sesión. Si la sesión es una sesión con transacción, se retrotrae cualquier transacción en curso.

Si una aplicación intenta cerrar una sesión que ya está cerrada, la llamada se ignora.

Parámetros:

Ninguna

Devuelve:

`Void`

Contexto de hebras

Cualquiera

Excepciones:

- XMSEException

Commit - Confirmar

Interfaz:

```
void Commit();
```

Confirmar todos los mensajes procesados en la transacción actual.

La sesión debe ser una sesión con transacción.

Parámetros:

Ninguna

Devuelve:

Void

Excepciones:

- XMSEException
- IllegalStateException
- TransactionRolledBackException

CreateBrowser - Crear examinador de colas

Interfaz:

```
IQueueBrowser CreateBrowser(IDestination queue) ;
```

Crear un examinador de colas para la cola especificada.

Parámetros:**queue (entrada)**

Un objeto Destination que representa la cola.

Devuelve:

El objeto QueueBrowser.

Excepciones:

- XMSEException
- InvalidDestinationException

CreateBrowser - Crear examinador de colas (con selector de mensajes)

Interfaz:

```
IQueueBrowser CreateBrowser(IDestination queue, String selector) ;
```

Crear un examinador de colas para la cola especificada utilizando un selector de mensajes.

Parámetros:**queue (entrada)**

Un objeto Destination que representa la cola.

selector (entrada)

Un objeto String que encapsula una expresión de selector de mensajes. Solo se entregan al examinador de colas los mensajes con propiedades que coinciden con la expresión de selector de mensajes.

Un objeto String nulo significa que no hay ningún selector de mensajes para el examinador de colas.

Devuelve:

El objeto QueueBrowser.

Excepciones:

- XMSEException
- InvalidDestinationException
- InvalidSelectorException

CreateBytesMessage - Crear mensaje de bytes

Interfaz:

```
IBytesMessage CreateBytesMessage();
```

Crear un mensaje de bytes.

Parámetros:

Ninguna

Devuelve:

El objeto BytesMessage.

Excepciones:

- XMSEException
- IllegalStateException (La sesión está cerrada)

CreateConsumer - Crear consumidor

Interfaz:

```
IMessageConsumer CreateConsumer(IDestination dest) ;
```

Crear un consumidor de mensajes para el destino especificado.

Parámetros:**dest (entrada)**

El objeto Destination.

Devuelve:

El objeto MessageConsumer.

Excepciones:

- XMSEException
- InvalidDestinationException

CreateConsumer - Crear consumidor (con selector de mensajes)

Interfaz:

```
IMessageConsumer CreateConsumer(IDestination dest,  
                                String selector) ;
```

Crear un consumidor de mensajes para el destino especificado utilizando un selector de mensajes.

Parámetros:**dest (entrada)**

El objeto Destination.

selector (entrada)

Un objeto String que encapsula una expresión de selector de mensajes. Solo se entregan al consumidor de mensajes los mensajes con propiedades que coinciden con la expresión de selector de mensajes.

Un objeto String nulo significa que no hay ningún selector de mensajes para el consumidor de mensajes.

Devuelve:

El objeto MessageConsumer.

Excepciones:

- XMSEException
- InvalidDestinationException
- InvalidSelectorException

CreateConsumer - Crear consumidor (con el selector de mensajes y el distintivo de mensaje local)

Interfaz:

```
IMessageConsumer CreateConsumer(IDestination dest,  
                                String selector,  
                                Boolean noLocal) ;
```

Crear un consumidor de mensajes para el destino especificado utilizando un selector de mensajes y, si el destino es un tema, especificando si el consumidor de mensajes recibe los mensajes publicados por su propia conexión.

Parámetros:

dest (entrada)

El objeto Destination.

selector (entrada)

Un objeto String que encapsula una expresión de selector de mensajes. Solo se entregan al consumidor de mensajes los mensajes con propiedades que coinciden con la expresión de selector de mensajes.

Un objeto String nulo significa que no hay ningún selector de mensajes para el consumidor de mensajes.

noLocal (entrada)

El valor True significa que el consumidor de mensajes no recibe los mensajes publicados por su propia conexión. El valor False significa que el consumidor de mensajes no recibe los mensajes publicados por su propia conexión. El valor predeterminado es False.

Devuelve:

El objeto MessageConsumer.

Excepciones:

- XMSEException
- InvalidDestinationException
- InvalidSelectorException

CreateDurableSubscriber - Crear suscriptor duradero

Interfaz:

```
IMessageConsumer CreateDurableSubscriber(IDestination dest,  
                                         String subscription) ;
```

Crear un suscriptor duradero para el tema especificado.

Este método no es válido para una conexión en tiempo real con un intermediario.

Para obtener más información sobre los suscriptores duraderos, consulte [Suscriptores duraderos](#).

Parámetros:**dest (entrada)**

Un objeto Destination que representa el tema. El tema no debe ser un tema temporal.

subscription (entrada)

Un objeto String encapsula un nombre que identifica la suscripción duradera. El nombre debe ser exclusivo dentro del identificador de cliente para la conexión.

Devuelve:

El objeto MessageConsumer que representa el suscriptor duradero.

Excepciones:

- XMSEException
- InvalidDestinationException

CreateDurableSubscriber - Crear suscriptores duraderos (con selector de mensajes y distintivo de mensaje local)

Interfaz:

```
IMessageConsumer CreateDurableSubscriber(IDestination dest,  
                                         String subscription,  
                                         String selector,  
                                         Boolean noLocal) ;
```

Crear un suscriptor duradero para el tema especificado utilizando un selector de mensajes y especificando si el suscriptor duradero recibe los mensajes publicados por su propia conexión.

Este método no es válido para una conexión en tiempo real con un intermediario.

Para obtener más información sobre los suscriptores duraderos, consulte [Suscriptores duraderos](#).

Parámetros:**dest (entrada)**

Un objeto Destination que representa el tema. El tema no debe ser un tema temporal.

subscription (entrada)

Un objeto String encapsula un nombre que identifica la suscripción duradera. El nombre debe ser exclusivo dentro del identificador de cliente para la conexión.

selector (entrada)

Un objeto String que encapsula una expresión de selector de mensajes. Solo se devuelven al suscriptor duradero los mensajes con propiedades que coinciden con la expresión de selector de mensajes.

Un objeto String nulo significa que no hay ningún selector de mensajes para el suscriptor duradero.

noLocal (entrada)

El valor True significa que el suscriptor duradero no recibe los mensajes publicados por su propia conexión. El valor False significa que el suscriptor duradero no recibe los mensajes publicados por su propia conexión. El valor predeterminado es False.

Devuelve:

El objeto MessageConsumer que representa el suscriptor duradero.

Excepciones:

- XMSEException
- InvalidDestinationException
- InvalidSelectorException

CreateMapMessage - Crear mensaje de correlación

Interfaz:

```
IMapMessage CreateMapMessage();
```

Crear un mensaje de correlación.

Parámetros:

Ninguna

Devuelve:

El objeto MapMessage.

Excepciones:

- XMSEException
- IllegalStateException (La sesión está cerrada)

CreateMessage - Crear mensaje

Interfaz:

```
IMessage CreateMessage();
```

Crear un mensaje que no tiene un cuerpo.

Parámetros:

Ninguna

Devuelve:

El objeto Message.

Excepciones:

- XMSEException
- IllegalStateException (La sesión está cerrada)

CreateObjectMessage - Crear mensaje de objeto

Interfaz:

```
IObjectMessage CreateObjectMessage();
```

Crear un mensaje de objeto.

Parámetros:

Ninguna

Devuelve:

El objeto ObjectMessage.

Excepciones:

- XMSEException
- IllegalStateException (La sesión está cerrada)

CreateProducer - Crear productor

Interfaz:

```
IMessageProducer CreateProducer(IDestination dest) ;
```

Crear un productor de mensajes para enviar mensajes al destino especificado.

Parámetros:

dest (entrada)

El objeto Destination.

Si especifica un objeto Destination nulo, el productor de mensajes se crea sin un destino. En este caso, la aplicación debe especificar un destino cada vez que utiliza el productor de mensajes envía un mensaje.

Devuelve:

El objeto MessageProducer.

Excepciones:

- XMSEException
- InvalidDestinationException

CreateQueue - Crear cola

Interfaz:

```
IDestination CreateQueue(String queue) ;
```

Crear un objeto Destination para representar una cola en el servidor de mensajería.

Este método no crea la cola en el servidor de mensajería. Debe crear la cola antes de que una aplicación pueda llamar a este método.

Parámetros:

queue (entrada)

Un objeto String que encapsula el nombre de la cola, o encapsula un identificador uniforme de recursos (URI) que identifica la cola.

Devuelve:

El objeto Destination que representa la cola.

Excepciones:

- XMSEException

CreateStreamMessage - Crear mensaje de corriente de datos

Interfaz:

```
IStreamMessage CreateStreamMessage();
```

Crear un mensaje de corriente de datos.

Parámetros:

Ninguna

Devuelve:

El objeto StreamMessage.

Excepciones:

- XMSEException
- XMS_ILLEGAL_STATE_EXCEPTION

CreateTemporaryQueue - Crear cola temporal

Interfaz:

```
IDestination CreateTemporaryQueue() ;
```

Crear una cola temporal.

El ámbito de la cola temporal es la conexión. Solo las sesiones creadas por la conexión pueden utilizar la cola temporal.

La cola temporal permanece hasta que se suprime de forma explícita, o finaliza la conexión, lo que suceda antes.

Para obtener más información sobre las colas temporales, consulte [Destinos temporales](#).

Parámetros:

Ninguna

Devuelve:

El objeto Destination que representa la cola temporal.

Excepciones:

- XMSEException

CreateTemporaryTopic - Crear tema temporal

Interfaz:

```
IDestination CreateTemporaryTopic() ;
```

Crear un tema temporal.

El ámbito del tema temporal es la conexión. Solo las sesiones creadas por la conexión pueden utilizar el tema temporal.

El tema temporal permanece hasta que se suprime de forma explícita, o finaliza la conexión, lo que suceda antes.

Para obtener más información sobre temas temporales, consulte [Destinos temporales](#).

Parámetros:

Ninguna

Devuelve:

El objeto Destination que representa el tema temporal.

Excepciones:

- XMSEException

CreateTextMessage - Crear mensaje de texto

Interfaz:

```
ITextMessage CreateTextMessage();
```

Crear un mensaje de texto con un cuerpo vacío.

Parámetros:

Ninguna

Devuelve:

El objeto TextMessage.

Excepciones:

- XMSEException

CreateTextMessage - Crear mensaje de texto (inicializado)

Interfaz:

```
ITextMessage CreateTextMessage(String initialValue);
```

Crear un mensaje de texto cuyo cuerpo se ha inicializado con el texto especificado.

Parámetros:

initialValue (entrada)

Un objeto String que encapsula el texto para inicializar el cuerpo del mensaje de texto.

Ninguna

Devuelve:

El objeto TextMessage.

Excepciones:

- XMSEException

CreateTopic - Crear tema

Interfaz:

```
IDestination CreateTopic(String topic) ;
```

Crear un objeto Destination para representar un tema.

Parámetros:**topic (entrada)**

Un objeto String que encapsula el nombre del tema, o encapsula un identificador uniforme de recursos (URI) que identifica el tema.

Devuelve:

El objeto Destination que representa el tema.

Excepciones:

- XMSEException

Recover - Recuperar

Interfaz:

```
void Recover();
```

Recuperar la sesión. Se detiene la entrega de mensaje y, después, se reinicia con el mensaje sin acuse de recibo más antiguo.

La sesión no debe ser una sesión con transacción.

Para obtener más información sobre la recuperación de una sesión, consulte [Acuse de recibo de mensaje](#).

Parámetros:

Ninguna

Devuelve:

Void

Excepciones:

- XMSEException
- IllegalStateException

Rollback - Retrotraer

Interfaz:

```
void Rollback();
```

Retrotraer todos los mensajes procesados en la transacción actual.

La sesión debe ser una sesión con transacción.

Parámetros:

Ninguna

Devuelve:

Void

Excepciones:

- XMSEException

- `IllegalStateException`

Unsubscribe - Anular suscripción

Interfaz:

```
void Unsubscribe(String subscription);
```

Suprimir una suscripción duradera. El servidor de mensajería suprime el registro de la suscripción duradera que está manteniendo y no envía ningún mensaje más al suscriptor duradero.

Una aplicación no puede suprimir una suscripción duradera bajo ninguna de las circunstancias siguientes:

- Mientras haya un consumidor de mensajes activo para la suscripción duradera
- Mientras un mensaje consumido forme parte de una transacción pendiente
- Mientras un mensaje consumido no tenga acuse de recibo

Este método no es válido para una conexión en tiempo real con un intermediario.

Parámetros:

subscription (entrada)

Un objeto `String` que encapsula el nombre que identifica la suscripción duradera.

Devuelve:

`Void`

Excepciones:

- `XMSEException`
- `InvalidDestinationException`
- `IllegalStateException`

Propiedades y métodos heredados

Los métodos siguientes se heredan de la interfaz `IPropertyContext`:

[`GetBooleanProperty`](#), [`GetByteProperty`](#), [`GetBytesProperty`](#), [`GetCharProperty`](#), [`GetDoubleProperty`](#), [`GetFloatProperty`](#), [`GetIntProperty`](#), [`GetLongProperty`](#), [`GetObjectProperty`](#), [`GetShortProperty`](#), [`GetStringProperty`](#), [`SetBooleanProperty`](#), [`SetByteProperty`](#), [`SetBytesProperty`](#), [`SetCharProperty`](#), [`SetDoubleProperty`](#), [`SetFloatProperty`](#), [`SetIntProperty`](#), [`SetLongProperty`](#), [`SetObjectProperty`](#), [`SetShortProperty`](#), [`SetStringProperty`](#)

InputStreamMessage

Un mensaje de corriente de datos es un mensaje cuyo cuerpo está formado por una corriente de valores, donde cada valor tiene un tipo de datos asociado. El contenido del cuerpo se escribe y se lee de forma secuencial.

Jerarquía de herencia:

```
IBM.XMS.IPropertyContext
|
+----IBM.XMS.IMessage
|
+----IBM.XMS.InputStreamMessage
```

Cuando una aplicación lee un valor de la corriente de datos de mensaje, el valor se puede convertir mediante XMS a otro tipo de datos. Para obtener más información sobre esta forma de conversión implícita, consulte [El cuerpo de un mensaje de XMS](#).

Métodos

ReadBoolean - Leer valor booleano

Interfaz:

```
Boolean ReadBoolean();
```

Leer un valor booleano en la corriente de datos de mensaje.

Parámetros:

Ninguna

Devuelve:

El valor booleano que se ha leído.

Excepciones:

- XMSEException
- MessageNotReadableException
- MessageEOFException

ReadByte - Leer byte

Interfaz:

```
Int16 ReadSignedByte();  
Byte ReadByte();
```

Leer un entero de 8-bits firmado en la corriente de datos de mensaje.

Parámetros:

Ninguna

Devuelve:

El byte que se ha leído.

Excepciones:

- XMSEException
- MessageNotReadableException
- MessageEOFException

ReadBytes - Leer bytes

Interfaz:

```
Int32 ReadBytes(Byte[] array);
```

Leer una matriz de bytes en la corriente de datos de mensaje.

Parámetros:

array (entrada)

El almacenamiento intermedio que contiene la matriz de bytes que se ha leído y la longitud del almacenamiento intermedio en bytes.

Si el número de bytes de la matriz es menor o igual que la longitud del almacenamiento intermedio, se lee toda la matriz en el almacenamiento intermedio. Si el número de bytes de la matriz es mayor que la longitud del almacenamiento intermedio, el almacenamiento intermedio se rellena con parte de la matriz, y un cursor interno marca la posición del siguiente byte que se va a leer. Una llamada posterior a readBytes() lee bytes de la matriz empezando por la posición actual del cursor.

Si especifica un puntero nulo en la entrada, la llamada se salta la matriz de bytes sin leerla.

Devuelve:

El número de bytes que se van a leer en el almacenamiento intermedio. Si el almacenamiento intermedio se rellena parcialmente, el valor es menor que la longitud del almacenamiento intermedio,

lo que indica que no quedan más bytes en la matriz para leer. Si no queda ningún byte para leer en la matriz antes de la llamada, el valor es XMSC_END_OF_BYTEARRAY.

Si especifica un puntero nulo en la entrada, el método no devuelve ningún valor.

Excepciones:

- XMSEException
- MessageNotReadableException
- MessageEOFException

ReadChar - Leer carácter

Interfaz:

```
Char ReadChar();
```

Leer un carácter de 2-bytes en la corriente de datos de mensaje.

Parámetros:

Ninguna

Devuelve:

El carácter que se ha leído.

Excepciones:

- XMSEException
- MessageNotReadableException
- MessageEOFException

ReadDouble - Leer número de coma flotante de precisión doble

Interfaz:

```
Double ReadDouble();
```

Leer un número de coma flotante de precisión doble de 8-bytes en la corriente de datos de mensaje.

Parámetros:

Ninguna

Devuelve:

El número de coma flotante de precisión doble que se ha leído.

Excepciones:

- XMSEException
- MessageNotReadableException
- MessageEOFException

ReadFloat - Leer número de coma flotante

Interfaz:

```
Single ReadFloat();
```

Leer un número de coma flotante de 4-bytes en la corriente de datos de mensaje.

Parámetros:

Ninguna

Devuelve:

El número de coma flotante que se ha leído.

Excepciones:

- XMSEException
- MessageNotReadableException
- MessageEOFException

ReadInt - Leer entero

Interfaz:

```
Int32 ReadInt();
```

Leer un entero de 32-bits firmado en la corriente de datos de mensaje.

Parámetros:

Ninguna

Devuelve:

El entero que se ha leído.

Excepciones:

- XMSEException
- MessageNotReadableException
- MessageEOFException

ReadLong - Leer entero largo

Interfaz:

```
Int64 ReadLong();
```

Leer un entero de 64-bits firmado en la corriente de datos de mensaje.

Parámetros:

Ninguna

Devuelve:

El entero largo que se ha leído.

Excepciones:

- XMSEException
- MessageNotReadableException
- MessageEOFException

ReadObject - Leer objeto

Interfaz:

```
Object ReadObject();
```

Leer un valor en la corriente de datos de mensaje y devolver su tipo de datos.

Parámetros:

Ninguna

Devuelve:

El valor, que es uno de los tipos de objeto siguientes:

Boolean
Byte
Byte[]
Char
Double

Single
Int32
Int64
Int16
String

Excepciones:

XMSEException

ReadShort - Leer entero corto

Interfaz:

```
Int16 ReadShort();
```

Leer un entero de 16-bits firmado en la corriente de datos de mensaje.

Parámetros:

Ninguna

Devuelve:

El entero corto que se ha leído.

Excepciones:

- XMSEException
- MessageNotReadableException
- MessageEOFException

ReadString - Leer serie

Interfaz:

```
String ReadString();
```

Leer una serie en la corriente de datos de mensaje. Si es necesario, XMS convierte los caracteres de la serie en la página de códigos local.

Parámetros:

Ninguna

Devuelve:

Un objeto String que encapsula la serie que se ha leído. Si es necesaria la conversión de datos, esta es la serie después de la conversión.

Excepciones:

- XMSEException
- MessageNotReadableException
- MessageEOFException

Reset - Restablecer

Interfaz:

```
void Reset();
```

Colocar el cuerpo del mensaje en la modalidad de solo lectura y volver a colocar el cursor al principio de la corriente de datos de mensaje.

Parámetros:

Ninguna

Devuelve:

Void

Excepciones:

- XMSEException
- MessageNotReadableException
- MessageEOFException

WriteBoolean - Escribir valor booleano

Interfaz:

```
void WriteBoolean(Boolean value);
```

Escribir un valor booleano en la corriente de datos de mensaje.

Parámetros:**value (entrada)**

El valor booleano que se va a escribir.

Devuelve:

Void

Excepciones:

- XMSEException
- MessageNotWritableException

WriteByte - Escribir byte

Interfaz:

```
void WriteByte(Byte value);  
void WriteSignedByte(Int16 value);
```

Escribir un byte en la corriente de datos de mensaje.

Parámetros:**value (entrada)**

El byte que se va a escribir.

Devuelve:

Void

Excepciones:

- XMSEException
- MessageNotWritableException

WriteBytes - Escribir bytes

Interfaz:

```
void WriteBytes(Byte[] value);
```

Escribir una matriz de bytes en la corriente de datos de mensaje.

Parámetros:**value (entrada)**

La matriz de bytes que se va a escribir.

length (entrada)

El número de bytes de la matriz.

Devuelve:

Void

Excepciones:

- XMSEException
- MessageNotWritableException

WriteChar - Escribir carácter

Interfaz:

```
void WriteChar(Char value);
```

Escribir un carácter en la corriente de datos de mensaje como 2 bytes, primero el byte de orden superior.

Parámetros:**value (entrada)**

El carácter que se va a escribir.

Devuelve:

Void

Excepciones:

- XMSEException
- MessageNotWritableException

WriteDouble - Escribir número de coma flotante de precisión doble

Interfaz:

```
void WriteDouble(Double value);
```

Convertir un número de coma flotante de precisión doble a un entero largo y escribir el entero largo en la corriente de datos de mensaje como 8 bytes, primero el byte de orden superior.

Parámetros:**value (entrada)**

El número de coma flotante de precisión doble que se va a escribir.

Devuelve:

Void

Excepciones:

- XMSEException
- MessageNotWritableException

WriteFloat - Escribir número de coma flotante

Interfaz:

```
void WriteFloat(Single value);
```

Convertir una número de coma flotante a un entero y escribir el entero en la corriente de datos de mensaje como 4 bytes, primero el byte de orden superior.

Parámetros:**value (entrada)**

El número de coma flotante que se va a escribir.

Devuelve:

Void

Excepciones:

- XMSEException

- MessageNotWritableException

WriteInt - Escribir entero

Interfaz:

```
void WriteInt(Int32 value);
```

Escribir un entero en la corriente de datos de mensaje como 4 bytes, primero el byte de orden superior.

Parámetros:

value (entrada)

El entero que se va a escribir.

Devuelve:

Void

Excepciones:

- XMSEException
- MessageNotWritableException

WriteLong - Escribir entero largo

Interfaz:

```
void WriteLong(Int64 value);
```

Escribir un entero largo en la corriente de datos de mensaje como 8 bytes, primero el byte de orden superior.

Parámetros:

value (entrada)

El entero largo que se va a escribir.

Devuelve:

Void

Excepciones:

- XMSEException
- MessageNotWritableException

WriteObject - Escribir objeto

Interfaz:

```
void WriteObject(Object value);
```

Escribir un valor, con un tipo de datos especificado, en la corriente de datos de mensaje.

Parámetros:

objectType (entrada)

El valor, que debe adoptar uno de los tipos de objeto siguientes:

Boolean
Byte
Byte[]
Char
Double
Single
Int32
Int64

Int16

String

value (entrada)

Una matriz de bytes que contiene el valor que se va a escribir.

length (entrada)

El número de bytes de la matriz.

Devuelve:

Void

Excepciones:

- XMSEException

WriteShort - Escribir entero corto

Interfaz:

```
void WriteShort(Int16 value);
```

Escribir un entero corto en la corriente de datos de mensaje como 2 bytes, primero el byte de orden superior.

Parámetros:

value (entrada)

El entero corto que se va a escribir.

Devuelve:

Void

Excepciones:

- XMSEException
- MessageNotWritableException

WriteString - Escribir serie

Interfaz:

```
void WriteString(String value);
```

Escribir serie en la corriente de datos de mensaje.

Parámetros:

value (entrada)

Un objeto String que encapsula la serie que se va a escribir.

Devuelve:

Void

Excepciones:

- XMSEException
- MessageNotWritableException

Propiedades y métodos heredados

Las propiedades siguientes se heredan de la interfaz [IMessage](#):

[JMSCorrelationID](#), [JMSDeliveryMode](#), [JMSDestination](#), [JMSExpiration](#), [JMSMessageID](#), [JMSPriority](#), [JMSRedelivered](#), [JMSReplyTo](#), [JMSTimestamp](#), [JMSType](#), [Properties](#)

Los métodos siguientes se heredan de la interfaz [IMessage](#):

[`clearBody`](#), [`clearProperties`](#), [`PropertyExists`](#)

Los métodos siguientes se heredan de la interfaz `IPropertyContext`:

[GetBooleanProperty](#), [GetByteProperty](#), [GetBytesProperty](#), [GetCharProperty](#), [GetDoubleProperty](#), [GetFloatProperty](#), [GetIntProperty](#), [GetLongProperty](#), [GetObjectProperty](#), [GetShortProperty](#), [GetStringProperty](#), [SetBooleanProperty](#), [SetByteProperty](#), [SetBytesProperty](#), [SetCharProperty](#), [SetDoubleProperty](#), [SetFloatProperty](#), [SetIntProperty](#), [SetLongProperty](#), [SetObjectProperty](#), [SetShortProperty](#), [SetStringProperty](#)

ITextMessage

Un mensaje de texto es un mensaje cuyo cuerpo está formado por una serie.

Jerarquía de herencia:

```
IBM.XMS.IPropertyContext
|
+----IBM.XMS.IMessage
|
+----IBM.XMS.ITextMessage
```

Propiedades de .NET

Text - Obtener y establecer texto

Interfaz:

```
String Text
{
    get;
    set;
}
```

Obtener y establecer la serie que forma el cuerpo del mensaje de texto.

Si es necesario, XMS convierte los caracteres de la serie en la página de códigos local.

Excepciones:

- `XMSException`
- `MessageNotReadableException`
- `MessageNotWritableException`
- `MessageEOFException`

Propiedades y métodos heredados

Las propiedades siguientes se heredan de la interfaz `IMessage`:

[JMSCorrelationID](#), [JMSDeliveryMode](#), [JMSDestination](#), [JMSExpiration](#), [JMSMessageID](#), [JMSPriority](#), [JMSRedelivered](#), [JMSReplyTo](#), [JMSTimestamp](#), [JMSType](#), [Properties](#)

Los métodos siguientes se heredan de la interfaz `IMessage`:

[clearBody](#), [clearProperties](#), [PropertyExists](#)

Los métodos siguientes se heredan de la interfaz `IPropertyContext`:

[GetBooleanProperty](#), [GetByteProperty](#), [GetBytesProperty](#), [GetCharProperty](#), [GetDoubleProperty](#), [GetFloatProperty](#), [GetIntProperty](#), [GetLongProperty](#), [GetObjectProperty](#), [GetShortProperty](#), [GetStringProperty](#), [SetBooleanProperty](#), [SetByteProperty](#), [SetBytesProperty](#), [SetCharProperty](#), [SetDoubleProperty](#), [SetFloatProperty](#), [SetIntProperty](#), [SetLongProperty](#), [SetObjectProperty](#), [SetShortProperty](#), [SetStringProperty](#)

TransactionInProgressException

XMS emite esta excepción si una aplicación solicita una operación que no es válida porque hay una transacción en curso.

Jerarquía de herencia:

```
IBM.XMS.XMSEException
|
+----IBM.XMS.XMSEException
|
+----IBM.XMS.TransactionInProgressException
```

Propiedades y métodos heredados

Los métodos siguientes se heredan de la interfaz [XMSEException](#):

[GetErrorCode](#), [GetLinkedException](#)

TransactionRolledBackException

XMS emite esta excepción si una aplicación llama a `Session.commit()` para confirmar la transacción actual, pero la transacción se retrotrae.

Jerarquía de herencia:

```
IBM.XMS.XMSEException
|
+----IBM.XMS.XMSEException
|
+----IBM.XMS.TransactionRolledBackException
```

Propiedades y métodos heredados

Los métodos siguientes se heredan de la interfaz [XMSEException](#):

[GetErrorCode](#), [GetLinkedException](#)

XMSEException

Si XMS detecta un error al procesar una llamada a un método .NET, XMS emite una excepción. Una excepción es un objeto que encapsula información sobre el error.

Jerarquía de herencia:

```
System.Exception
|
+----IBM.XMS.XMSEException
```

Hay distintos tipos de excepción XMS y un objeto `XMSEException` es sólo un tipo de excepción. Sin embargo, la clase `XMSEException` es una superclase de las otras clases de excepción XMS. XMS lanza un objeto `XMSEException` en situaciones donde ninguno de los otros tipos de excepción es apropiado.

Propiedades de .NET

ErrorCode - Obtener código de error

Interfaz:

```
public String ErrorCode
{
    get {return errorCode_;}
}
```

Obtener el código de error.

Excepciones:

- XMSEException

LinkedException - Obtener excepción enlazada

Interfaz:

```
public Exception LinkedException
{
    get { return linkedException_;}
    set { linkedException_ = value;}
}
```

Obtener la siguiente excepción de la cadena de excepciones.

El método devuelve un nulo, si no hay más excepciones en la cadena.

Excepciones:

- XMSEException

XMSFactoryFactory

Si una aplicación no está utilizando objetos administrados, utilice esta clase para crear fábricas de conexiones, colas y temas.

Jerarquía de herencia:

Ninguna

Propiedades de .NET

Metadata - Recuperar metadatos

Interfaz:

```
IConnectionMetaData Metadata
```

Obtener los metadatos que son apropiados para el tipo de conexión del objeto XMSFactoryFactory.

Excepciones:

Ninguna

Métodos

CreateConnectionFactory - Crear fábrica de conexiones

Interfaz:

```
IConnectionFactory CreateConnectionFactory();
```

Crear un objeto ConnectionFactory del tipo declarado.

Parámetros:

Ninguna

Devuelve:

El objeto ConnectionFactory.

Excepciones:

- XMSEException

CreateQueue - Crear cola

Interfaz:

```
IDestination CreateQueue(String name);
```

Crear un objeto Destination para representar una cola en el servidor de mensajería.

Este método no crea la cola en el servidor de mensajería. Debe crear la cola antes de que una aplicación pueda llamar a este método.

Parámetros:

name (entrada)

Un objeto String que encapsula el nombre de la cola, o encapsula un identificador uniforme de recursos (URI) que identifica la cola.

Devuelve:

El objeto Destination que representa la cola.

Excepciones:

- XMSEException

CreateTopic - Crear tema

Interfaz:

```
IDestination CreateTopic(String name);
```

Crear un objeto Destination para representar un tema.

Parámetros:

name (entrada)

Un objeto String que encapsula el nombre del tema, o encapsula un identificador uniforme de recursos (URI) que identifica el tema.

Devuelve:

El objeto Destination que representa el tema.

Excepciones:

- XMSEException

GetInstance - Obtener una instancia de XMSFactoryFactory

Interfaz:

```
static XMSFactoryFactory GetInstance(int connectionType);
```

Cree una instancia de XMSFactoryFactory. Una aplicación XMS utiliza un objeto XMSFactoryFactory para obtener una referencia a un objeto ConnectionFactory que es apropiado para el tipo de protocolo necesario. Este objeto ConnectionFactory puede generar conexiones solo para ese tipo de protocolo.

Parámetros:

connectionType (entrada)

El tipo de conexión para el cual el objeto ConnectionFactory genera conexiones.

- XMSC.CT_WPM
- XMSC.CT_RTT
- XMSC.CT_WMQ

Devuelve:

El objeto XMSFactoryFactory dedicado al tipo de conexión declarado.

Excepciones:

- NotSupportedException

Propiedades de objetos XMS

Esta sección documenta las propiedades de objeto definidas por XMS.

Esta sección contiene información sobre los siguientes tipos de objeto:

- “Propiedades de conexión” en la [página 2068](#)
- “Propiedades de ConnectionFactory” en la [página 2069](#)
- “Propiedades de ConnectionMetaData” en la [página 2074](#)
- “Propiedades de Destination” en la [página 2074](#)
- “Propiedades de InitialContext” en la [página 2076](#)
- “Propiedades de Message” en la [página 2077](#)
- “Propiedades de MessageConsumer” en la [página 2082](#)
- “Propiedades de MessageProducer” en la [página 2082](#)
- “Propiedades de sesión” en la [página 2082](#)

La descripción de cada tipo de objeto lista las propiedades de un objeto del tipo especificado y proporciona una breve descripción de cada propiedad.

Esta sección también proporciona una definición de cada propiedad (consulte “Definiciones de propiedad” en la [página 2082](#)).

Si una aplicación define sus propias propiedades de los objetos descritos en esta sección, no provoca un error, pero puede provocar resultados imprevisibles.

Nota: Los nombres y valores de propiedad de esta sección se muestran en el formato XMSC.NAME, que es el formato utilizado para C y C++. Sin embargo, en .NET, el formato del nombre de propiedad puede ser XMSC.NAME o XMSC_NAME, en función de cómo lo esté utilizando:

- Si está especificando una propiedad, el nombre de propiedad debe tener el formato XMSC.NAME tal como se muestra en el ejemplo siguiente:

```
cf.SetStringProperty(XMSC.WMQ_CHANNEL, "DOTNET.SVRCONN");
```

- Si está especificando una serie, el nombre de propiedad debe tener el formato XMSC_NAME tal como se muestra en el ejemplo siguiente:

```
cf.SetStringProperty("XMSC_WMQ_CHANNEL", "DOTNET.SVRCONN");
```

En .NET, los nombres y valores de propiedad se proporcionan como constantes en la clase XMSC. Estas constantes identifican series y las utilizaría cualquier aplicación XMS.NET. Si está utilizando estas constantes predefinidas, los nombres y valores de propiedad tienen el formato XMSC.NAME, de modo que, por ejemplo, utilizaría XMSC.USERID, en lugar de XMSC_USERID.

Los tipos de datos también están en el formato utilizado para C/C++. Puede encontrar los valores correspondientes para .NET en [Tipos de datos para .NET](#).

Propiedades de conexión

Una descripción general de las propiedades del objeto Connection, con enlaces a información de referencia más detallada.

Tabla 908. Propiedades de conexión	
Nombre de la propiedad	Descripción
“XMSC_WMQ_RESOLVED_QUEUE_MANAGER” en la página 2119	Esta propiedad se utiliza para obtener el nombre del gestor de colas que está conectado.

Tabla 908. Propiedades de conexión (continuación)	
Nombre de la propiedad	Descripción
“XMSC_WMQ_RESOLVED_QUEUE_MANAGER_ID” en la página 2120	Esta propiedad se llena con el ID del gestor de colas después de la conexión.
XMSC_WPM_CONNECTION_PROTOCOL	El protocolo de comunicaciones utilizado para la conexión al motor de mensajería. Esta propiedad es de solo lectura.
XMSC_WPM_HOST_NAME	El nombre de host o la dirección IP del sistema que contiene el motor de mensajería al que se conecta la aplicación. Esta propiedad es de solo lectura.
XMSC_WPM_ME_NAME	El nombre del motor de mensajería al que se conecta la aplicación. Esta propiedad es de solo lectura.
XMSC_WPM_PORT	El número del puerto en el que estaba a la escucha el motor de mensajería al que está conectada la aplicación. Esta propiedad es de solo lectura.

Un objeto Connection también tiene prioridad de solo lectura que se han derivado de las propiedades de la fábrica de conexiones que se utilizó para crear la conexión. Estas propiedades no solo se han derivado de las propiedades de fábrica de conexiones que se establecieron en el momento cuando se creó la conexión, sino que también de los valores predeterminados de las propiedades que no se establecieron. Las propiedades solo incluyen las que son relevantes para el tipo de servidor de mensajería al que está conectada la aplicación. Los nombres de las propiedades son los mismos que los nombres de las propiedades de fábrica de conexiones.

Propiedades de ConnectionFactory

Una descripción general de las propiedades del objeto ConnectionFactory, con enlaces a información de referencia más detallada.

Tabla 909. Propiedades de ConnectionFactory	
Nombre de la propiedad	Descripción
“XMSC_ASYNC_EXCEPTIONS” en la página 2094	Esta propiedad determina si XMS informa de un ExceptionListener solo cuando se interrumpe una conexión, o cuando se produce cualquier excepción de forma asíncrona en una llamada de la API XMS. Esta propiedad se aplica a todas las conexiones creadas a partir de esta ConnectionFactory que tienen un ExceptionListener registrado.
“XMSC_WMQ_TIPO_DE_APLICACIÓN_DE_EQUILIBRIO” en la página 2103	Tipo de opción de equilibrio
“XMSC_WMQ_OPCIONES_DE_EQUILIBRIO” en la página 2103	Opciones de equilibrio establecidas por la aplicación emisora
“XMSC_WMQ_TIEMPO_DE_ESPERA_DE_EQUILIBRIO” en la página 2104	Tiempo de espera tras el cual el reequilibrio podría interrumpir la actividad de la aplicación.
XMSC_CLIENT_ID	El identificador de cliente para una conexión.
XMSC_CONNECTION_TYPE	El tipo del servidor de mensajería al que se conecta una aplicación.
XMSC_PASSWORD	Una contraseña que se puede utilizar para autenticar la aplicación cuando intenta conectarse a un servidor de mensajería.

Tabla 909. Propiedades de ConnectionFactory (continuación)	
Nombre de la propiedad	Descripción
“XMSC_RTT_BROKER_PING_INTERVAL” en la página 2100	El intervalo de tiempo, en milisegundos, transcurrido el cual XMS .NET comprueba la conexión con un servidor de mensajería en tiempo real para detectar cualquier actividad.
XMSC_RTT_CONNECTION_PROTOCOL	El protocolo de comunicación utilizado para una conexión en tiempo real con un intermediario.
XMSC_RTT_HOST_NAME	El nombre de host o la dirección IP del sistema en el que se ejecuta un intermediario.
XMSC_RTT_LOCAL_ADDRESS	El nombre de host o la dirección IP de la interfaz de red local que se va a utilizar para una conexión en tiempo real con un intermediario.
XMSC_RTT_MULTICAST	El valor de multidifusión para una fábrica de conexiones o destino.
XMSC_RTT_PORT	El número del puerto en el cual un intermediario escucha solicitudes entrantes.
XMSC_USERID	Un identificador de usuario que se puede utilizar para autenticar la aplicación cuando intenta conectarse a un servidor de mensajería.
XMSC_WMQ_BROKER_CONTROLQ	El nombre de la cola de control que utiliza un broker.
XMSC_WMQ_BROKER_PUBQ	El nombre de la cola supervisada por un intermediario donde las aplicaciones envían mensajes que publican.
XMSC_WMQ_BROKER_QMGR	El nombre del gestor de colas al que está conectado un intermediario.
XMSC_WMQ_BROKER_SUBQ	El nombre de la cola de suscriptor para un consumidor de mensajes no duraderos.
XMSC_WMQ_BROKER_VERSION	El tipo de intermediario utilizado por la aplicación para una conexión o para el destino.
“XMSC_WMQ_CCDTURL” en la página 2106	Un localizador uniforme de recursos (URL) que identifica el nombre y la ubicación del archivo que contiene la tabla de la definición de canal de cliente y, también, especifica cómo se puede acceder al archivo.
XMSC_WMQ_CHANNEL	El nombre del canal que se va a utilizar para una conexión.
“XMSC_WMQ_CLIENT_RECONNECT_OPTIONS” en la página 2107	Esta propiedad especifica las opciones de reconexión de cliente para nuevas conexiones creadas por esta fábrica
“XMSC_WMQ_CLIENT_RECONNECT_TIMEOUT” en la página 2107	Esta propiedad especifica la duración de tiempo, en segundos, que una conexión de cliente intenta reconectarse.
XMSC_WMQ_CONNECTION_MODE	La modalidad a través de la cual una aplicación se conecta a un gestor de colas.
“XMSC_WMQ_CONNECTION_NAME_LIST” en la página 2108	Esta propiedad especifica los hosts a los que el cliente intenta reconectarse después de que se hayan interrumpido sus conexiones.

Tabla 909. Propiedades de ConnectionFactory (continuación)	
Nombre de la propiedad	Descripción
XMSC_WMQ_FAIL_IF QUIESCE	Indica si las llamadas a determinados métodos fallan si el gestor de colas al que está conectada la aplicación está en un estado de inmovilización.
XMSC_WMQ_HOST_NAME	El nombre de host o la dirección IP del sistema en el cual se ejecuta un gestor de colas.
XMSC_WMQ_LOCAL_ADDRESS	Para una conexión a un gestor de colas, esta propiedad especifica la interfaz de red local que se va a utilizar, o el puerto local o el rango de puertos locales que se van a utilizar, o ambos.
XMSC_WMQ_MESSAGE_SELECTION	Determina si la selección de mensajes la realiza el XMS cliente o el broker.
XMSC_WMQ_MSG_BATCH_SIZE	El número máximo de mensajes que se va a recuperar de una cola en un lote cuando se utiliza la entrega de mensajes asíncrona.
XMSC_WMQ_POLLING_INTERVAL	Si cada escucha de mensajes dentro de una sesión no tiene ningún mensaje adecuado en su cola, este valor es el intervalo máximo, en milisegundos, que transcurre antes de que cada escucha de mensajes intente de nuevo obtener un mensaje de su cola.
“XMSC_WMQ_PROVIDER_VERSION” en la página 2117	La versión, release, nivel de modificación y fixpack del gestor de colas al que tiene intención conectarse la aplicación.
XMSC_WMQ_PORT	El número del puerto en el cual un gestor de colas escucha solicitudes entrantes.
XMSC_WMQ_PUB_ACK_INTERVAL	El número de mensajes publicados por un editor antes de que el XMS cliente solicite un acuse de recibo al bróker.
“XMSC_WMQ_PUT_ASYNC_ALLOWED” en la página 2112	Esta propiedad determina si los productores de mensajes están permitidos para utilizar operaciones de transferencia asíncrona para enviar mensajes a este destino.
XMSC_WMQ_QMGR_CC SID	El identificador (CCSID) del conjunto de caracteres codificados, o página de códigos, en el que se intercambian los campos de datos de caracteres definidos en la interfaz de cola de mensajes (MQI) entre el XMS cliente y el IBM MQ cliente.
XMSC_WMQ_QUEUE_MANAGER	El nombre del gestor de colas al que conectarse.
XMSC_WMQ_RECEIVE_EXIT	Identifica una salida de recepción de canal que se va a ejecutar.
XMSC_WMQ_RECEIVE_EXIT_INIT	Los datos de usuario que se pasan a una salida de recepción de canal cuando se llama.
XMSC_WMQ_SECURITY_EXIT	Identifica una salida de seguridad de canal.
XMSC_WMQ_SECURITY_EXIT_INIT	Los datos de usuario que se pasan a una salida de seguridad de canal cuando se llama.

Tabla 909. Propiedades de ConnectionFactory (continuación)	
Nombre de la propiedad	Descripción
“XMSC_WMQ_SEND_CHECK_COUNT” en la página 2121	El número de llamadas de envío que se van a permitir entre las comprobaciones de errores de colocación asíncrona dentro de una sesión XMS única sin transacción.
XMSC_WMQ_SEND_EXIT	Identifica una salida de emisión de canal.
XMSC_WMQ_SEND_EXIT_INIT	Los datos de usuario que se pasan a las salidas de emisión de canal cuando se llaman.
“XMSC_WMQ_SHARE_CONV_ALLOWED” en la página 2121	Si una conexión de cliente puede compartir su socket con otras conexiones de primer nivel XMS del mismo proceso al mismo gestor de colas, siempre que las definiciones de los canales coincidan. Esta propiedad se proporciona para permitir un aislamiento completo de conexiones en sockets separados si es necesario para el desarrollo de aplicaciones, el mantenimiento o por motivos operativos.
XMSC_WMQ_SSL_CERT_STORES	Las ubicaciones de los servidores que contienen las listas de revocaciones de certificados (CRL) que se van a utilizar en una conexión SSL a un gestor de colas.
XMSC_WMQ_SSL_CIPHER_SPEC	El nombre de la CipherSpec que se va a utilizar en una conexión segura a un gestor de colas.
XMSC_WMQ_SSL_CIPHER_SUITE	El nombre de la CipherSuite que se va a utilizar en una conexión TLS a un gestor de colas. El protocolo utilizado en la negociación de la conexión segura depende de la CipherSuite especificada.
XMSC_WMQ_SSL_CRYPT_HW	Detalles de configuración para el hardware de cifrado conectado al sistema cliente.
XMSC_WMQ_SSL_FIPS_REQUIRED	El valor de esta propiedad determina si una aplicación puede o no utilizar conjuntos de cifrado no FIPS conformes. Si esta propiedad se establece en «true», solo FIPS se utilizan algoritmos para la conexión cliente-servidor.
 XMSC_WMQ_SSL_REPOSITORIO_DE_CLAVES_1	La ubicación de un archivo de base de datos de claves en el cual se almacenan claves y certificados.
XMSC_WMQ_SSL_KEY_RESETCOUNT	El KeyResetCount representa el número total de bytes sin cifrado enviados y recibidos en una conversación SSL antes de renegociar la clave secreta.
XMSC_WMQ_SSL_PEER_NAME	El nombre de igual que se va a utilizar en una conexión SSL a un gestor de colas.
XMSC_WMQ_SYNCPOINT_ALL_GETS	Indica si se deben recuperar todos los mensajes de colas dentro del control de punto de sincronización.
“XMSC_WMQ_TARGET_CLIENT” en la página 2128	
XMSC_WMQ_TEMP_Q_PREFIX	El prefijo utilizado para formar el nombre de la IBM MQ cola dinámica que se crea cuando la aplicación genera una XMS cola temporal.

Tabla 909. Propiedades de ConnectionFactory (continuación)

Nombre de la propiedad	Descripción
<u>XMSC_WMQ_TEMP_TOPIC_PREFIX</u>	TEMP/TEMPTOPICPREFIX/unique_idAl crear temas temporales, XMS se genera una cadena de tema con el formato «», o, si esta propiedad contiene el valor predeterminado, se genera esta cadena: «TEMP/unique_id». Si se especifica un valor no vacío se permite que se definan colas de modelo específicas para crear las colas gestionadas para suscriptores de temas temporales creados en esta conexión.
<u>XMSC_WMQ_TEMPORARY_MODEL</u>	El nombre de la IBM MQ cola modelo a partir de la cual se crea una cola dinámica cuando la aplicación crea una XMS cola temporal.
<u>XMSC_WPM_BUS_NAME</u>	Para una fábrica de conexiones, el nombre del bus de integración de servicios al que se conecta la aplicación o, para un destino, el nombre del bus de integración de servicios en el cual existe el destino.
<u>XMSC_WPM_CONNECTION_PROXIMITY</u>	El valor de proximidad de conexión para la conexión.
<u>XMSC_WPM_DUR_SUB_HOME</u>	El nombre del motor de mensajería donde se gestionan todas las suscripciones duraderas para una conexión o destino.
<u>XMSC_WPM_LOCAL_ADDRESS</u>	Para una conexión a un bus de integración de servicios, esta propiedad especifica la interfaz de red local que se va a utilizar, o el puerto local o el rango de puertos locales que se va a utilizar, o ambos.
<u>XMSC_WPM_NON_PERSISTENT_MAP</u>	El nivel de fiabilidad de los mensajes no persistente que se envían utilizando la conexión.
<u>XMSC_WPM_PERSISTENT_MAP</u>	El nivel de fiabilidad de los mensajes persistentes que se envían mediante la conexión.
<u>XMSC_WPM_PROVIDER_ENDPOINTS</u>	Una secuencia de una o más direcciones de punto final de servidores de programa de arranque.
<u>XMSC_WPM_TARGET_GROUP</u>	El nombre de un grupo de destino de motores de mensajería.
<u>XMSC_WPM_TARGET_SIGNIFICANCE</u>	La importancia del grupo de destinos de motores de mensajería.
<u>XMSC_WPM_TARGET_TRANSPORT_CHAIN</u>	El nombre de la cadena de transporte de entrada que debe utilizar la aplicación para conectarse a un motor de mensajería.
<u>XMSC_WPM_TARGET_TYPE</u>	El tipo del grupo de destinos de motores de mensajería.
<u>XMSC_WPM_TEMP_Q_PREFIX</u>	El prefijo utilizado para formar el nombre de la cola temporal que se crea en el bus de integración de servicios cuando la aplicación crea una XMS cola temporal.
<u>XMSC_WPM_TEMP_TOPIC_PREFIX</u>	El prefijo utilizado para formar el nombre de un tema temporal que crea la aplicación.
 <u>MSC_WMQ_SSL_KEY_REPOSITORY_PASSWORD1</u>	Propiedad que permite a los clientes establecer las contraseñas necesarias para abrir el certificado TLS antes del establecimiento de la conexión.

Tabla 909. Propiedades de ConnectionFactory (continuación)	
Nombre de la propiedad	Descripción
X MSC_WMQ_CLAVE_DE_CIFRADO_1	Propiedad que permite a los clientes especificar la ubicación donde se guarda el archivo de la clave de cifrado.

Nota: [V 9.0.0](#) Desde IBM MQ 9.4.4, XMS.NET las aplicaciones pueden utilizar rutas de certificados independientes para las conexiones TLS proporcionando una ruta personalizada. También puede proporcionar contraseñas cifradas utilizando la utilidad Dotnet Password Encrypt o contraseñas de texto sin formato. Para obtener más información, consulte [Ruta personalizada para certificados en XMS .NET para la conectividad TLS mediante contraseñas de texto sin formato o cifradas](#).

Propiedades de ConnectionMetaData

Una descripción general de las propiedades del objeto ConnectionMetaData, con enlaces a información de referencia más detallada.

Tabla 910. Propiedades de ConnectionMetaData	
Nombre de la propiedad	Descripción
XMSC_JMS_MAJOR_VERSION	El número de versión principal de la JMS especificación en la que XMS se basa. Esta propiedad es de solo lectura.
XMSC_JMS_MINOR_VERSION	El número de versión menor de la JMS especificación en la que XMS se basa. Esta propiedad es de solo lectura.
XMSC_JMS_VERSION	El identificador de versión de la JMS especificación en la que XMS se basa. Esta propiedad es de solo lectura.
XMSC_MAJOR_VERSION	El número de versión del XMS cliente. Esta propiedad es de solo lectura.
XMSC_MINOR_VERSION	El número de versión del XMS cliente. Esta propiedad es de solo lectura.
XMSC_PROVIDER_NAME	XMS El proveedor del cliente. Esta propiedad es de solo lectura.
XMSC_VERSION	El identificador de versión del XMScliente. Esta propiedad es de solo lectura.

Propiedades de Destination

Una descripción general de las propiedades del objeto Destination, con enlaces a información de referencia a más detallada.

Tabla 911. Propiedades de Destination	
Nombre de la propiedad	Descripción
XMSC_DELIVERY_MODE	La modalidad de entrega de mensajes enviados al destino.
XMSC_PRIORITY	La prioridad de los mensajes enviados al destino.
XMSC_RTT_MULTICAST	El valor de multidifusión para una fábrica de conexiones o destino.
XMSC_TIME_TO_LIVE	El tiempo de vida para los mensajes enviados al destino.
XMSC_WMQ_BROKER_VERSION	El tipo de intermediario utilizado por la aplicación para una conexión o para el destino.

Tabla 911. Propiedades de Destination (continuación)	
Nombre de la propiedad	Descripción
XMSC_WMQ_CCSID	El identificador (CCSID) del conjunto de caracteres codificados, o página de códigos, en el que se encuentran las cadenas de datos de caracteres del cuerpo de un mensaje cuando el XMS cliente reenvía el mensaje al destinatario.
XMSC_WMQ_DUR_SUBQ	El nombre de la cola de suscriptor para un suscriptor duradero que está recibiendo mensajes del destino. Nota: Esta propiedad se puede utilizar con la versión 2.0 de IBM Message Service Client para .NET pero no tiene ningún efecto para una aplicación conectada a un gestor de colas IBM WebSphere MQ 7.0 a menos que la propiedad XMSC_WMQ_PROVIDER_VERSION de la fábrica de conexiones se establezca en un número de versión menor que 7.
XMSC_WMQ_ENCODING	Cómo se representan los datos numéricos del cuerpo de un mensaje cuando el XMS cliente reenvía el mensaje al destinatario.
XMSC_WMQ_FAIL_IF QUIESCE	Indica si las llamadas a determinados métodos fallan si el gestor de colas al que está conectada la aplicación está en un estado de inmovilización.
“XMSC_WMQ_MESSAGE_BODY” en la página 2110	Esta propiedad determina si una XMS aplicación procesa el campo « MQRFH2 » de un IBM MQ mensaje como parte de la carga útil del mensaje (es decir, como parte del cuerpo del mensaje).
“XMSC_WMQ_MQMD_MESSAGE_CONTEXT” en la página 2111	Determina qué nivel de contexto del mensaje debe establecer la XMS aplicación. La aplicación debe estar en ejecución con la autoridad de contexto adecuada para que esta propiedad tenga efecto.
“XMSC_WMQ_MQMD_READ_ENABLED” en la página 2112	Esta propiedad determina si una XMS aplicación puede extraer los valores de los campos MQMD o no.
“XMSC_WMQ_MQMD_WRITE_ENABLED” en la página 2112	Esta propiedad determina si una XMS aplicación puede establecer los valores de los campos MQMD o no.
“XMSC_WMQ_READ_AHEAD_ALLOWED” en la página 2113	Esta propiedad determina si los consumidores de mensajes y los navegadores de colas están autorizados para utilizar la lectura anticipada para obtener mensajes no persistentes y no transaccionales de este destino en un almacenamiento intermedio interno antes de recibirlos.
“XMSC_WMQ_READ_AHEAD_CLOSE_POLICY” en la página 2114	Esta propiedad determina, para los mensajes que se están entregando a un escucha de mensajes asíncronos, qué sucede con los mensajes en el almacenamiento de lectura anticipada interno, cuando se cierra el consumidor de mensajes.
“XMSC_WMQ_RECEIVE_CCSID” en la página 2118	La propiedad de destino que define el CCSID de destino para la conversión de mensajes del gestor de colas. El valor se ignora, a menos que XMSC_WMQ_RECEIVE_CONVERSION se establezca en WMQ_RECEIVE_CONVERSION_QMGR.

Tabla 911. Propiedades de Destination (continuación)	
Nombre de la propiedad	Descripción
“XMSC_WMQ_RECEIVE_CONVERSION” en la página 2119	La propiedad de destino que determina si el gestor de colas va a realizar la conversión de datos.
XMSC_WMQ_TARGET_CLIENT	Indica si los mensajes enviados al destino contienen una cabecera MQRFH2.
XMSC_WMQ_TEMP_TOPIC_PREFIX	TEMP/TEMPTOPICPREFIX/unique_idAl crear temas temporales, XMS se genera una cadena de tema con el formato «», o, si esta propiedad contiene el valor predeterminado, se genera esta cadena: «TEMP/unique_id». Si se especifica un valor no vacío se permite que se definan colas de modelo específicas para crear las colas gestionadas para suscriptores de temas temporales creados en esta conexión.
XMSC_WPM_BUS_NAME	Para una fábrica de conexiones, el nombre del bus de integración de servicios al que se conecta la aplicación o, para un destino, el nombre del bus de integración de servicios en el cual existe el destino.
XMSC_WPM_TOPIC_SPACE	El nombre del espacio de tema que contiene el tema.

Propiedades de InitialContext

Una descripción general de las propiedades del objeto InitialContext, con enlaces a información de referencia más detallada.

Tabla 912. Propiedades de InitialContext	
Nombre de la propiedad	Descripción
XMSC_IC_PROVIDER_URL	Se utiliza para localizar el directorio de denominación JNDI de forma que no es necesario que el servicio de denominación COS esté en el mismo servidor que el servicio web.
XMSC_IC_SECURITY_AUTHENTICATION	Basado en la Java interfaz de contexto SECURITY_AUTHENTICATION. Esta propiedad solo es aplicable al contexto de denominación COS.
XMSC_IC_SECURITY_CREDENTIALS	Basado en la Java interfaz de contexto SECURITY_CREDENTIALS. Esta propiedad solo es aplicable al contexto de denominación COS.
XMSC_IC_SECURITY_PRINCIPAL	Basado en la Java interfaz de contexto SECURITY_PRINCIPAL. Esta propiedad solo es aplicable al contexto de denominación COS.
XMSC_IC_SECURITY_PROTOCOL	Basado en la Java interfaz Context SECURITY_PROTOCOL Esta propiedad solo es aplicable al contexto de denominación COS.
XMSC_IC_URL	Para contextos de LDAP y FileSystem, la dirección del repositorio que contiene objetos administrados. Para contextos de denominación COS, la dirección del servicio web que busca los objetos en el directorio.

Propiedades de Message

Una descripción general de las propiedades del objeto Message, con enlaces a información de referencia más detallada.

Tabla 913. Propiedades de Message	
Nombre de la propiedad	Descripción
JMS_IBM_CHARACTER_SET	El identificador (CCSID) del conjunto de caracteres codificados, o página de códigos, en el que se encuentran las cadenas de datos de caracteres del cuerpo del mensaje cuando el XMS cliente reenvía el mensaje a su destino previsto. En XMS, esta propiedad tiene un valor numérico y se correlaciona con el identificador de juego de caracteres codificados (CCSID). Si embargo, esta propiedad se basa en una propiedad JMS de modo que tiene un valor de tipo serie y se correlaciona con el juego de caracteres Java que representa este CCSID numérico.
JMS_IBM_ENCODING	Cómo se representan los datos numéricos del cuerpo del mensaje cuando el XMS cliente reenvía el mensaje a su destinatario.
JMS_IBM_EXCEPTIONMESSAGE	Texto que describe la razón por la cual el mensaje se ha enviado al destino de excepciones. Esta propiedad es de solo lectura.
JMS_IBM_EXCEPTIONPROBLEMDESTINATION	El nombre del destino en el que estaba el mensaje antes de que se enviara al destino de excepciones.
JMS_IBM_EXCEPTIONREASON	Un código de razón que indica la razón por la cual el mensaje se envió al destino de excepciones.
JMS_IBM_EXCEPTIONTIMESTAMP	La hora cuando se envió el mensaje al destino de excepciones.
JMS_IBM_FEEDBACK	Un código que indica la naturaleza de un mensaje de informe.
JMS_IBM_FORMAT	La naturaleza de los datos de aplicación en el mensaje.
JMS_IBM_LAST_MSG_IN_GROUP	Indica si el mensaje es el último mensaje de un grupo de mensajes.
JMS_IBM_MSGTYPE	El tipo del mensaje.
JMS_IBM_PUTAPPLTYPE	El tipo de la aplicación que envió el mensaje.
JMS_IBM_PUTDATE	La fecha cuando se envió el mensaje.
JMS_IBM_PUTTIME	La hora cuando se envió el mensaje.
JMS_IBM_REPORT_COA	Solicitar los mensajes de informe 'confirmar en llegada', especificando cuántos datos de aplicación del mensaje original se deben incluir en un mensaje de informe.
JMS_IBM_REPORT_COD	Solicitar mensaje de informe 'confirmar en llegada', que especifica cuántos datos de aplicación del mensaje original se deben incluir en un mensaje de informe.
JMS_IBM_REPORT_DISCARD_MSG	Solicitar que el mensaje se descarte si no se puede entregar a su destino previsto.

Tabla 913. Propiedades de Message (continuación)	
Nombre de la propiedad	Descripción
<u>JMS_IBM_REPORT_EXCEPTION</u>	Solicitar mensajes de informe de excepción, que especifican cuántos datos de aplicación del mensaje original se deben incluir en un mensaje de informe.
<u>JMS_IBM_REPORT_EXPIRATION</u>	Solicitar mensajes de informe de caducidad, que especifican cuántos datos de aplicación del mensaje original se deben incluir en un mensaje de informe.
<u>JMS_IBM_REPORT_NAN</u>	Solicitar mensajes de informe de notificación de acción negativa.
<u>JMS_IBM_REPORT_PAN</u>	Solicitar mensajes de informe de notificación de acción positiva.
<u>JMS_IBM_REPORT_PASS_CORREL_ID</u>	Solicite que el identificador de correlación de cualquier mensaje de informe o de respuesta sea el mismo que el identificador de correlación del mensaje original.
<u>JMS_IBM_REPORT_PASS_MSG_ID</u>	Solicite que el identificador de mensaje de cualquier mensaje de informe o respuesta sea el mismo que el identificador de mensaje del mensaje original.
<u>JMS_IBM_RETAIN</u>	Establecer esta propiedad indica al gestor de colas que trate un mensaje como una Publicación retenida.
<u>JMS_IBM_SYSTEM_MESSAGEID</u>	Un identificador que identifica el mensaje de forma exclusiva dentro del bus de integración de servicios. Esta propiedad es de solo lectura.
<u>JMSX_APPID</u>	El nombre de la aplicación que envió el mensaje.
<u>JMSX_DELIVERY_COUNT</u>	El número de intentos para entregar el mensaje.
<u>JMSX_GROUPID</u>	El identificador del grupo de mensajes al que pertenece el mensaje.
<u>JMSX_GROUPSEQ</u>	El número de secuencia del mensaje de un grupo de mensajes.
<u>JMSX_USERID</u>	El identificador de usuario asociado con la aplicación que envió el mensaje.

Propiedades JMS_IBM_MQMD*

IBM Message Service Client for .NET permite a las aplicaciones cliente leer/escribir campos MQMD utilizando distintas API. También permite el acceso a datos de mensaje MQ. De forma predeterminada, el acceso a MQMD está inhabilitado y la aplicación la debe habilitar de forma explícita utilizando las propiedades Destination XMSC_WMQ_MQMD_WRITE_ENABLED y XMSC_WMQ_MQMD_READ_ENABLED. Estas dos propiedades son independientes entre sí.

Todos los campos MQMD excepto StrucId y Version se exponen como propiedades de objeto Message adicionales y se añaden como prefijo a JMS_IBM_MQMD.

Las propiedades JMS_IBM_MQMD* tienen una mayor prioridad sobre otras propiedades como JMS_IBM* descritas en la tabla anterior.

Envío de mensajes

Están representados todos los campos MQMD excepto StrucId y Version. Estas propiedades solo hacen referencia a los campos MQMD; donde una propiedad se produce tanto en MQMD como en la

cabeceraMQRFH2, la versión en MQRFH2 no se establece ni se extrae. Se puede establecer cualquiera de estas propiedades, excepto JMS_IBM_MQMD_BackoutCount. Se ignora cualquier valor establecido para JMS_IBM_MQMD_BackoutCount.

Si una propiedad tiene una longitud máxima y proporciona un valor que es demasiado largo, el valor se trunca.

Para determinadas propiedades, también debe establecer la propiedad XMSC_WMQ_MQMD_MESSAGE_CONTEXT en el objeto Destination. La aplicación debe estar en ejecución con la autoridad de contexto adecuada para que esta propiedad tenga efecto. Si no establece XMSC_WMQ_MQMD_MESSAGE_CONTEXT en un valor apropiado, se ignora el valor de propiedad. Si establece XMSC_WMQ_MQMD_MESSAGE_CONTEXT en un valor apropiado, pero no tiene suficiente autoridad de contexto para el gestor de colas, se emite una excepción. Las propiedades que requieren valores específicos de XMSC_WMQ_MQMD_MESSAGE_CONTEXT son las siguientes.

Las propiedades siguientes requieren que XMSC_WMQ_MQMD_MESSAGE_CONTEXT esté establecido en XMSC_WMQ_MDCTX_SET_IDENTITY_CONTEXT o XMSC_WMQ_MDCTX_SET_ALL_CONTEXT:

- JMS_IBM_MQMD_UserIdentifier
- JMS_IBM_MQMD_AccountingToken
- JMS_IBM_MQMD_ApplIdentityData

Las propiedades siguientes requieren que XMSC_WMQ_MQMD_MESSAGE_CONTEXT esté establecido en XMSC_WMQ_MDCTX_SET_ALL_CONTEXT:

- JMS_IBM_MQMD_PutApplType
- JMS_IBM_MQMD_PutApplName
- JMS_IBM_MQMD_PutDate
- JMS_IBM_MQMD_PutTime
- JMS_IBM_MQMD_ApplOriginData

Recepción de mensajes

Todas estas propiedades están disponibles en un mensaje recibido si la propiedad XMSC_WMQ_MQMD_READ_ENABLED está establecida en true, independientemente de las propiedades reales que ha establecido la aplicación productora. Una aplicación no puede modificar las propiedades de un mensaje recibido, a menos que primero se borren todas las propiedades, de acuerdo con la especificación JMS. El mensaje recibido se puede enviar sin modificar las propiedades.

Nota: Si la aplicación recibe un mensaje de un destino con la propiedad XMSC_WMQ_MQMD_READ_ENABLED establecida en true, y lo envía a un destino con XMSC_WMQ_MQMD_WRITE_ENABLED establecido en true, esto genera que todos los valores de campo MQMD del mensaje recibido se copien en el mensaje enviado. Tabla de propiedades

Tabla 914. Propiedades del objeto Message que representa los campos MQMD		
Propiedad	Descripción	Tipo
JMS_IBM_MQMD_REPORT	Opciones para mensajes de informe	System.Int32
JMS_IBM_MQMD_MSGTYPE	Tipo de mensaje	System.Int32
JMS_IBM_MQMD_EXPIRY	Duración del mensaje	System.Int32
JMS_IBM_MQMD_FEEDBACK	Código de comentario o razón	System.Int32
JMS_IBM_MQMD_ENCODING	Codificación numérica de datos de mensaje	System.Int32
JMS_IBM_MQMD_CODEDCHARSETID	Identificador de juego de caracteres de datos de mensaje	System.Int32

Tabla 914. Propiedades del objeto Message que representa los campos MQMD (continuación)		
Propiedad	Descripción	Tipo
JMS_IBM_MQMD_FORMAT	Nombre de formato de datos de mensaje	System.String
JMS_IBM_MQMD_PRIORITY Nota: Si asigna un valor a JMS_IBM_MQMD_PRIORITY que no está dentro del rango de 0 a 9, este valor infringe la especificación JMS.	Prioridad de mensaje	System.Int32
JMS_IBM_MQMD_PERSISTENCE	Persistencia de los mensajes	System.Int32
JMS_IBM_MQMD_MSGID Nota: La especificación JMS indica que el ID de mensaje debe ser establecido por el proveedor JMS y que se debe ser exclusivo o nulo. Si asigna un valor a JMS_IBM_MQMD_MSGID, este valor se copia en el JMSMessageID. De esta forma, el proveedor JMS no lo establece y podría no ser exclusivo: este valor infringe la especificación JMS.	Identificador de mensaje	Matriz de bytes Nota: El uso de las propiedades de matriz de bytes en un mensaje infringe la especificación JMS.
JMS_IBM_MQMD_CORRELID Nota: Si asigna un valor a JMS_IBM_MQMD_CORRELID que empieza con la serie 'ID:', este valor infringe la especificación JMS.	Identificador de correlación	Matriz de bytes Nota: El uso de las propiedades de matriz de bytes en un mensaje infringe la especificación JMS.
JMS_IBM_MQMD_BACKOUTCOUNT	Contador de restitución	System.Int32
JMS_IBM_MQMD_REPLYTOQ	Nombre de la cola de respuestas	System.String
JMS_IBM_MQMD_REPLYTOQMGR	Nombre del gestor de colas de respuestas	System.String
JMS_IBM_MQMD_USERIDENTIFIER	Identificador de usuario	System.String
JMS_IBM_MQMD_ACCOUNTINGTOKEN	Señal de contabilidad	Matriz de bytes Nota: El uso de las propiedades de matriz de bytes en un mensaje infringe la especificación JMS.
JMS_IBM_MQMD_APPLIDENTITYDATA	Datos de aplicación relacionados con la identidad	System.String
JMS_IBM_MQMD_PUTAPPLTYPE	Tipo de aplicación que coloca el mensaje	System.Int32
JMS_IBM_MQMD_PUTAPPLNAME	Nombre de la aplicación que coloca el mensaje	System.String
JMS_IBM_MQMD_PUTDATE	Fecha cuando se colocó el mensaje	System.String
JMS_IBM_MQMD_PUTTIME	Hora cuando se colocó el mensaje	System.String

Tabla 914. Propiedades del objeto Message que representa los campos MQMD (continuación)		
Propiedad	Descripción	Tipo
JMS_IBM_MQMD_APPLORIGINDATA	Datos de aplicación relacionados con el origen	System.String
JMS_IBM_MQMD_GROUPID	Identificador de grupo	Matriz de bytes Nota: El uso de las propiedades de matriz de bytes en un mensaje infringe la especificación JMS.
JMS_IBM_MQMD_MSGSEQNUMBER	Número de secuencia del mensaje local dentro del grupo	System.Int32
JMS_IBM_MQMD_OFFSET	Desplazamiento de datos en el mensaje físico desde el inicio del mensaje lógico	System.Int32
JMS_IBM_MQMD_MSGFLAGS	Distintivos de mensajes	System.Int32
JMS_IBM_MQMD_ORIGINALLENGTH	Longitud del mensaje original	System.Int32

Consulte [MQMD](#) si desea más detalles.

Ejemplos

Este ejemplo provoca que un mensaje se coloque en una cola o un tema con MQMD.UserIdentifier establecido en "JoeBloggs".

```
// Create a ConnectionFactory, connection, session, producer, message
// ...

// Create a destination
// ...

// Enable MQMD write
dest.setBooleanProperty(XMSC_WMQ_MQMD_WRITE_ENABLED,
    XMSC_WMQ_MQMD_WRITE_ENABLED_YES);

// Optionally, set a message context if applicable for this MD field
dest.setIntProperty(XMSC_WMQ_MQMD_MESSAGE_CONTEXT,
    XMSC_WMQ_MDCTX_SET_IDENTITY_CONTEXT);

// On the message, set property to provide custom UserId
msg.setStringProperty(JMS_IBM_MQMD_USERIDENTIFIER, "JoeBloggs");

// Send the message
// ...
```

Es necesario establecer XMSC_WMQ_MQMD_MESSAGE_CONTEXT antes de establecer JMS_IBM_MQMD_USERIDENTIFIER. Si desea más información sobre el uso de XMSC_WMQ_MQMD_MESSAGE_CONTEXT, consulte las propiedades del objeto Message.

De forma similar, puede extraer el contenido de los campos MQMD estableciendo XMSC_WMQ_MQMD_READ_ENABLED en true antes de recibir un mensaje y, después, utilizar los métodos get del mensaje como, por ejemplo, getStringProperty. Las propiedades recibidas son de solo lectura.

Este ejemplo tiene como resultado que el campo de valor que contiene el valor del campo MQMD.ApplIdentityData de un mensaje se obtenga de una cola o de un tema.

```
// Create a ConnectionFactory, connection, session, consumer
// ...

// Create a destination
```

```
// ...

// Enable MQMD read
dest.setBooleanProperty(XMSC_WMQ_MQMD_READ_ENABLED, XMSC_WMQ_MQMD_READ_ENABLED_YES);

// Receive a message
// ...

// Get required MQMD field value using a property
System.String value = rcvMsg.getStringProperty(JMS_IBM_MQMD_APPLIDENTITYDATA);
```

Propiedades de MessageConsumer

Una descripción general de las propiedades del objeto MessageConsumer, con enlaces a información de referencia más detallada.

Tabla 915. Propiedades de MessageConsumer	
Nombre de la propiedad	Descripción
XMSC_IS_SUBSCRIPTION_MULTICAST	Indica si se están enviando mensajes al consumidor de mensajes mediante WebSphere MQ Multicast Transport. Esta propiedad es de solo lectura.
XMSC_IS_SUBSCRIPTION_RELIABLE_MULTICAST	Indica si los mensajes se están entregando al destinatario con WebSphere MQ Multicast Transport una calidad de servicio fiable. Esta propiedad es de solo lectura.

Consulte las [Propiedades .NET de IMessageConsumer](#) si desea más detalles.

Propiedades de MessageProducer

Una descripción general de las propiedades del objeto MessageProducer, con enlaces a información de referencia más detallada.

Consulte [las propiedades .NET de IMessageProducer](#) para obtener más detalles.

Propiedades de sesión

Una descripción general de las propiedades del objeto Session, con enlaces a información de referencia más detallada.

Consulte [las propiedades de ISession de .NET](#) para obtener más detalles.

Definiciones de propiedad

Esta sección proporciona una definición de cada propiedad del objeto.

Cada definición de propiedad incluye la información siguiente:

- El tipo de datos de la propiedad
- Los tipos de objeto que tienen la propiedad
- Para una propiedad de Destination, el nombre que se puede utilizar en un identificador uniforme de recursos (URI)
- Una descripción más detallada de la propiedad
- Los valores válidos de la propiedad
- El valor predeterminado de la propiedad

Las propiedades cuyos nombres empiezan con uno de los prefijos siguientes solo son relevantes para el tipo de conexión especificado:

XMSC_RTT

Las propiedades solo son relevantes para una conexión en tiempo real con un intermediario. Los nombres de las propiedades están definidos como constantes con nombre en el archivo de cabecera `xmsc_rtt.h`.

XMSC_WMQ

Las propiedades solo son relevantes cuando una aplicación se conecta a un IBM MQ gestor de colas. Los nombres de las propiedades están definidos como constantes con nombre en el archivo de cabecera `xmsc_wmq.h`.

XMSC_WPM

Las propiedades solo son relevantes cuando una aplicación se conecta a un bus de integración de servicios WebSphere. Los nombres de las propiedades están definidos como constantes con nombre en el archivo de cabecera `xmsc_wpm.h`.

A menos que se indique lo contrario en sus definiciones, las propiedades restantes son relevantes para todos los tipos de conexión. Los nombres de las propiedades están definidos como constantes con nombre en el archivo de cabecera `xmsc.h`. Las propiedades cuyos nombres empiezan con el prefijo JMSX son propiedades definidas de JMS de un mensaje, y las propiedades cuyos nombres empiezan con el prefijo JMS_IBM son las propiedades definidas de IBM de un mensaje. Para obtener más información sobre las propiedades de los mensajes, consulte [Propiedades de un XMS mensaje](#).

A menos que se indique lo contrario en su definición, cada propiedad es relevante tanto en los dominios punto a punto como en los dominios de publicación y suscripción.

Una aplicación puede obtener y establecer el valor de cualquier propiedad, a menos que la propiedad se haya designado como de solo lectura.

Están definidas las propiedades siguientes:

- [“JMS_IBM_CHARACTER_SET” en la página 2085](#)
- [“JMS_IBM_ENCODING” en la página 2085](#)
- [“JMS_IBM_EXCEPTIONMESSAGE” en la página 2086](#)
- [“JMS_IBM_EXCEPTIONPROBLEMDESTINATION” en la página 2086](#)
- [“JMS_IBM_EXCEPTIONREASON” en la página 2087](#)
- [“JMS_IBM_EXCEPTIONTIMESTAMP” en la página 2087](#)
- [“JMS_IBM_FEEDBACK” en la página 2087](#)
- [“JMS_IBM_FORMAT” en la página 2087](#)
- [“JMS_IBM_LAST_MSG_IN_GROUP” en la página 2087](#)
- [“JMS_IBM_MSGTYPE” en la página 2088](#)
- [“JMS_IBM_PUTAPPLTYPE” en la página 2088](#)
- [“JMS_IBM_PUTDATE” en la página 2088](#)
- [“JMS_IBM_PUTTIME” en la página 2089](#)
- [“JMS_IBM_REPORT_COA” en la página 2089](#)
- [“JMS_IBM_REPORT_COD” en la página 2089](#)
- [“JMS_IBM_REPORT_DISCARD_MSG” en la página 2090](#)
- [“JMS_IBM_REPORT_EXCEPTION” en la página 2090](#)
- [“JMS_IBM_REPORT_EXPIRATION” en la página 2091](#)
- [“JMS_IBM_REPORT_NAN” en la página 2091](#)
- [“JMS_IBM_REPORT_PAN” en la página 2091](#)
- [“JMS_IBM_REPORT_PASS_CORREL_ID” en la página 2092](#)
- [“JMS_IBM_REPORT_PASS_MSG_ID” en la página 2092](#)
- [“JMS_IBM_SYSTEM_MESSAGEID” en la página 2093](#)
- [“JMSX_APPID” en la página 2093](#)
- [“JMSX_DELIVERY_COUNT” en la página 2093](#)
- [“JMSX_GROUPID” en la página 2094](#)
- [“JMSX_GROUPSEQ” en la página 2094](#)
- [“JMSX_USERID” en la página 2094](#)

[“XMSC_CLIENT_ID” en la página 2095](#)
[“XMSC_CONNECTION_TYPE” en la página 2095](#)
[“XMSC_DELIVERY_MODE” en la página 2095](#)
[“XMSC_IC_PROVIDER_URL” en la página 2096](#)
[“XMSC_IC_SECURITY_AUTHENTICATION” en la página 2096](#)
[“XMSC_IC_SECURITY_CREDENTIALS” en la página 2097](#)
[“XMSC_IC_SECURITY_PRINCIPAL” en la página 2097](#)
[“XMSC_IC_SECURITY_PROTOCOL” en la página 2097](#)
[“XMSC_IC_URL” en la página 2097](#)
[“XMSC_IS_SUBSCRIPTION_MULTICAST” en la página 2097](#)
[“XMSC_IS_SUBSCRIPTION_RELIABLE_MULTICAST” en la página 2098](#)
[“XMSC_JMS_MAJOR_VERSION” en la página 2098](#)
[“XMSC_JMS_MINOR_VERSION” en la página 2098](#)
[“XMSC_JMS_VERSION” en la página 2098](#)
[“XMSC_MAJOR_VERSION” en la página 2098](#)
[“XMSC_MINOR_VERSION” en la página 2098](#)
[“XMSC_PASSWORD” en la página 2099](#)
[“XMSC_PRIORITY” en la página 2099](#)
[“XMSC_PROVIDER_NAME” en la página 2100](#)
[“XMSC_RTT_BROKER_PING_INTERVAL” en la página 2100](#)
[“XMSC_RTT_CONNECTION_PROTOCOL” en la página 2100](#)
[“XMSC_RTT_HOST_NAME” en la página 2100](#)
[“XMSC_RTT_LOCAL_ADDRESS” en la página 2100](#)
[“XMSC_RTT_MULTICAST” en la página 2101](#)
[“XMSC_RTT_PORT” en la página 2102](#)
[“XMSC_TIME_TO_LIVE” en la página 2102](#)
[“XMSC_USERID” en la página 2102](#)
[“XMSC_VERSION” en la página 2103](#)
[“XMSC_WMQ_BROKER_CONTROLQ” en la página 2104](#)
[“XMSC_WMQ_BROKER_PUBQ” en la página 2105](#)
[“XMSC_WMQ_BROKER_QMGR” en la página 2105](#)
[“XMSC_WMQ_BROKER_SUBQ” en la página 2105](#)
[“XMSC_WMQ_BROKER_VERSION” en la página 2105](#)
[“XMSC_WMQ_CCDTURL” en la página 2106](#)
[“XMSC_WMQ_CCSID” en la página 2106](#)
[“XMSC_WMQ_CHANNEL” en la página 2107](#)
[“XMSC_WMQ_CONNECTION_MODE” en la página 2108](#)
[“XMSC_WMQ_DUR_SUBQ” en la página 2108](#)
[“XMSC_WMQ_ENCODING” en la página 2109](#)
[“XMSC_WMQ_FAIL_IF QUIESCE” en la página 2110](#)
[“XMSC_WMQ_HOST_NAME” en la página 2114](#)
[“XMSC_WMQ_LOCAL_ADDRESS” en la página 2115](#)
[“XMSC_WMQ_MESSAGE_SELECTION” en la página 2116](#)
[“XMSC_WMQ_MSG_BATCH_SIZE” en la página 2116](#)
[“XMSC_WMQ_POLLING_INTERVAL” en la página 2116](#)
[“XMSC_WMQ_PORT” en la página 2117](#)
[“XMSC_WMQ_PUB_ACK_INTERVAL” en la página 2117](#)
[“XMSC_WMQ_QMGR_CCSID” en la página 2118](#)
[“XMSC_WMQ_QUEUE_MANAGER” en la página 2118](#)
[“XMSC_WMQ_RECEIVE_EXIT” en la página 2119](#)
[“XMSC_WMQ_RECEIVE_EXIT_INIT” en la página 2119](#)

[“XMSC_WMQ_SECURITY_EXIT” en la página 2120](#)
[“XMSC_WMQ_SECURITY_EXIT_INIT” en la página 2120](#)
[“XMSC_WMQ_SEND_EXIT” en la página 2120](#)
[“XMSC_WMQ_SEND_EXIT_INIT” en la página 2121](#)
[“XMSC_WMQ_SYNCPOINT_ALL_GETS” en la página 2128](#)
[“XMSC_WMQ_TARGET_CLIENT” en la página 2128](#)
[“XMSC_WMQ_TEMP_Q_PREFIX” en la página 2129](#)
[“XMSC_WMQ_TEMPORARY_MODEL” en la página 2129](#)
[“XMSC_WPM_BUS_NAME” en la página 2130](#)
[“XMSC_WPM_CONNECTION_PROTOCOL” en la página 2131](#)
[“XMSC_WPM_CONNECTION_PROXIMITY” en la página 2131](#)
[“XMSC_WPM_DUR_SUB_HOME” en la página 2131](#)
[“XMSC_WPM_HOST_NAME” en la página 2132](#)
[“XMSC_WPM_LOCAL_ADDRESS” en la página 2132](#)
[“XMSC_WPM_ME_NAME” en la página 2132](#)
[“XMSC_WPM_NON_PERSISTENT_MAP” en la página 2133](#)
[“XMSC_WPM_PERSISTENT_MAP” en la página 2133](#)
[“XMSC_WPM_PORT” en la página 2134](#)
[“XMSC_WPM_PROVIDER_ENDPOINTS” en la página 2134](#)
[“XMSC_WPM_TARGET_GROUP” en la página 2137](#)
[“XMSC_WPM_TARGET_SIGNIFICANCE” en la página 2137](#)
[“XMSC_WPM_TARGET_TRANSPORT_CHAIN” en la página 2138](#)
[“XMSC_WPM_TARGET_TYPE” en la página 2138](#)
[“XMSC_WPM_TEMP_Q_PREFIX” en la página 2138](#)
[“XMSC_WPM_TEMP_TOPIC_PREFIX” en la página 2139](#)
[“XMSC_WPM_TOPIC_SPACE” en la página 2139](#)

JMS_IBM_CHARACTER_SET

Tipo de datos:

System.Int32

Propiedad de:

Mensaje

El identificador (CCSID) del conjunto de caracteres codificados, o página de códigos, en el que se encuentran las cadenas de datos de caracteres del cuerpo del mensaje cuando el XMS cliente reenvía el mensaje a su destino previsto. En XMS, esta propiedad tiene un valor numérico y se correlaciona con el identificador de juego de caracteres codificados (CCSID). Si embargo, esta propiedad se basa en una propiedad JMS de modo que tiene un valor de tipo serie y se correlaciona con el juego de caracteres Java que representa este CCSID numérico. Esta propiedad altera temporalmente cualquier CCSID especificado para el destino mediante la propiedad [XMSC_WMQ_CCSID](#).

De forma predeterminada, la propiedad no está establecida.

Esta propiedad no es relevante cuando la aplicación se conecta a un bus de integración de servicios.

JMS_IBM_ENCODING

Tipo de datos:

System.Int32

Propiedad de:

Mensaje

Cómo se representan los datos numéricos del cuerpo del mensaje cuando el XMS cliente reenvía el mensaje a su destinatario. Esta propiedad anula cualquier codificación especificada para el destino por la propiedad [XMSC_WMQ_ENCODING](#). La propiedad especifica la representación de enteros binarios, enteros decimales empaquetados y números de coma flotante.

Los valores válidos de la propiedad son los mismos que los valores que se pueden especificar en el campo **Encoding** de un descriptor de mensaje.

Una aplicación puede utilizar las constantes con nombre siguientes para establecer la propiedad:

Constante con nombre	Significado
MQENC_INTEGER_NORMAL	Codificación de entero normal
MQENC_INTEGER_REVERSED	Codificación de entero inverso
MQENC_DECIMAL_NORMAL	Codificación de decimal empaquetado normal
MQENC_DECIMAL_REVERSED	Codificación de decimal empaquetado inverso
MQENC_FLOAT_IEEE_NORMAL	Codificación de coma flotante IEEE normal
MQENC_FLOAT_IEEE_REVERSED	Codificación de coma flotante IEEE inversa
 MQENC_FLOAT_S390	z/OS codificación de coma flotante de arquitectura
MQENC_NATIVE	Codificación de máquina nativa

Para formar un valor para la propiedad, la aplicación puede añadir tres de estas constantes del modo siguiente:

- Una constante cuyo nombre empieza con MQENC_INTEGER, para especificar la representación de enteros binarios
- Una constante cuyo nombre empieza con MQENC_DECIMAL, para especificar la representación de enteros decimales empaquetados
- Una constante cuyo nombre empieza con MQENC_FLOAT, para especificar la representación de números de coma flotante

De forma alternativa, la aplicación puede establecer la propiedad en MQENC_NATIVE, cuyo valor depende del entorno.

De forma predeterminada, la propiedad no está establecida.

Esta propiedad no es relevante cuando la aplicación se conecta a un bus de integración de servicios.

JMS_IBM_EXCEPTIONMESSAGE

Tipo de datos:

Serie

Propiedad de:

Mensaje

Texto que describe la razón por la cual el mensaje se ha enviado al destino de excepciones. Esta propiedad es de solo lectura.

Esta propiedad solo es relevante cuando una aplicación se conecta a un bus de integración de servicios y recibe un mensaje de un destino de excepciones.

JMS_IBM_EXCEPTIONPROBLEMDESTINATION

Tipo de datos:

Serie

Propiedad de:

Mensaje

El nombre del destino en el que estaba el mensaje antes de que se enviara al destino de excepciones.

Esta propiedad solo es relevante cuando una aplicación se conecta a un bus de integración de servicios y recibe un mensaje de un destino de excepciones.

JMS_IBM_EXCEPTIONREASON

Tipo de datos:

System.Int32

Propiedad de:

Mensaje

Un código de razón que indica la razón por la cual el mensaje se envió al destino de excepciones.

Esta propiedad solo es relevante cuando una aplicación se conecta a un bus de integración de servicios y recibe un mensaje de un destino de excepciones.

JMS_IBM_EXCEPTIONTIMESTAMP

Tipo de datos:

System.Int64

Propiedad de:

Mensaje

La hora cuando se envió el mensaje al destino de excepciones.

La hora se expresa en milisegundos desde 00:00:00 GMT del 1 de enero de 1970.

Esta propiedad solo es relevante cuando una aplicación se conecta a un bus de integración de servicios y recibe un mensaje de un destino de excepciones.

JMS_IBM_FEEDBACK

Tipo de datos:

System.Int32

Propiedad de:

Mensaje

Un código que indica la naturaleza de un mensaje de informe.

Los valores válidos de la propiedad son los códigos de comentarios (feedback) y códigos de razón que se pueden especificar en el campo **Feedback** de un descriptor de mensaje.

De forma predeterminada, la propiedad no está establecida.

JMS_IBM_FORMAT

Tipo de datos:

Serie

Propiedad de:

Mensaje

La naturaleza de los datos de aplicación en el mensaje.

Los valores válidos de la propiedad son los mismos que los valores que se pueden especificar en el campo **Format** de un descriptor de mensaje.

De forma predeterminada, la propiedad no está establecida.

Esta propiedad no es relevante cuando la aplicación se conecta a un bus de integración de servicios.

JMS_IBM_LAST_MSG_IN_GROUP

Tipo de datos:

System.Boolean

Propiedad de:

Mensaje

Indica si el mensaje es el último mensaje de un grupo de mensajes.

Establezca la propiedad en true si el mensaje es el último mensaje de un grupo de mensajes. De lo contrario, establezca la propiedad en false, o no establezca la propiedad. De forma predeterminada, la propiedad no está establecida.

El valor true corresponde al indicador de estado MQMF_LAST_MSG_IN_GROUP, que se puede especificar en el **MsgFlags** campo de un descriptor de mensaje..

Esta propiedad se ignora en el dominio de publicación/suscripción y no es relevante cuando una aplicación se conecta a un bus de integración de servicios.

JMS_IBM_MSGTYPE

Tipo de datos:

System.Int32

Propiedad de:

Mensaje

El tipo del mensaje.

Los valores válidos de la propiedad son los siguientes:

Valor válido	Significado
MQMT_DATAGRAM	El mensaje es uno que no requiere una respuesta.
MQMT_REQUEST	El mensaje es uno que requiere una respuesta.
MQMT_REPLY	El mensaje es un mensaje de respuesta.
MQMT_REPORT	El mensaje es un mensaje de informe.

Estos valores corresponden a los tipos de mensaje que se pueden especificar en el campo de **MsgType** de un descriptor de mensaje.

De forma predeterminada, la propiedad no está establecida.

Esta propiedad no es relevante cuando la aplicación se conecta a un bus de integración de servicios.

JMS_IBM_PUTAPPLTYPE

Tipo de datos:

System.Int32

Propiedad de:

Mensaje

El tipo de la aplicación que envió el mensaje.

Los valores válidos de la propiedad son los tipos de aplicación que se pueden especificar en el campo **PutApplType** de un descriptor de mensaje.

De forma predeterminada, la propiedad no está establecida.

Esta propiedad no es relevante cuando la aplicación se conecta a un bus de integración de servicios.

JMS_IBM_PUTDATE

Tipo de datos:

Serie

Propiedad de:

Mensaje

La fecha cuando se envió el mensaje.

Los valores válidos de la propiedad son los mismos que los valores que se pueden especificar en el campo **PutDate** de un descriptor de mensaje.

De forma predeterminada, la propiedad no está establecida.

Esta propiedad no es relevante cuando la aplicación se conecta a un bus de integración de servicios.

JMS_IBM_PUTTIME

Tipo de datos:

Serie

Propiedad de:

Mensaje

La hora cuando se envió el mensaje.

Los valores válidos de la propiedad son los mismo que los valores que se pueden especificar en el campo **PutTime** de un descriptor de mensaje.

De forma predeterminada, la propiedad no está establecida.

Esta propiedad no es relevante cuando la aplicación se conecta a un bus de integración de servicios.

JMS_IBM_REPORT_COA

Tipo de datos:

System.Int32

Propiedad de:

Mensaje

Solicitar los mensajes de informe 'confirmar en llegada', especificando cuántos datos de aplicación del mensaje original se deben incluir en un mensaje de informe.

Los valores válidos de la propiedad son los siguientes:

Valor válido	Significado
MQRO_COA	Solicitar mensajes de informe 'confirmar en llegada', si datos de aplicación del mensaje original incluidos en un mensaje de informe.
MQRO_COA_WITH_DATA	Solicitar mensajes de informe 'confirmar en llegada', con los primeros 100 bytes de datos de aplicación del mensaje original incluidos en un mensaje de informe.
MQRO_COA_WITH_FULL_DATA	Solicitar mensajes de informe 'confirmar en llegada', con todos los datos de aplicación del mensaje original incluidos en un mensaje de informe.

Estos valores corresponden a opciones de informe que se pueden especificar en el campo **Report** de un descriptor de mensaje. Si desea más información sobre estas opciones, consulte [Informe \(MQLONG\)](#).

De forma predeterminada, la propiedad no está establecida.

JMS_IBM_REPORT_COD

Tipo de datos:

System.Int32

Propiedad de:

Mensaje

Solicitar mensaje de informe 'confirmar en llegada', que especifica cuántos datos de aplicación del mensaje original se deben incluir en un mensaje de informe.

Los valores válidos de la propiedad son los siguientes:

Valor válido

MQRO_COD

Significado

Solicitar mensajes de informe 'confirmar en llegada', con ninguno de los datos de aplicación del mensaje original incluido en un mensaje de informe.

MQRO_COD_WITH_DATA

Solicitar mensajes de informe 'confirmar en llegada', con los primeros 100 bytes de datos de aplicación del mensaje original incluidos en un mensaje de informe.

MQRO_COD_WITH_FULL_DATA

Solicitar mensajes de informe 'confirmar en llegada', con todos los datos de aplicación del mensaje original incluidos en un mensaje de informe.

Estos valores corresponden a opciones de informe que se pueden especificar en el campo **Report** de un descriptor de mensaje.

De forma predeterminada, la propiedad no está establecida.

JMS_IBM_REPORT_DISCARD_MSG**Tipo de datos:**

System.Int32

Propiedad de:

Mensaje

Solicitar que el mensaje se descarte si no se puede entregar a su destino previsto.

Establezca la propiedad en MQRO_DISCARD_MSG para solicitar que el mensaje se descarte si no se puede entregar a su destino previsto. Si necesita que el mensaje se coloque en una cola de mensajes no entregados en su lugar, o que se envíe a un destino de excepciones, no establezca la propiedad. De forma predeterminada, la propiedad no está establecida.

El valor MQRO_DISCARD_MSG corresponde a una opción de informe que se puede especificar en el campo **Report** de un descriptor de mensaje.

JMS_IBM_REPORT_EXCEPTION**Tipo de datos:**

System.Int32

Propiedad de:

Mensaje

Solicitar mensajes de informe de excepción, que especifican cuántos datos de aplicación del mensaje original se deben incluir en un mensaje de informe.

Los valores válidos de la propiedad son los siguientes:

Valor válido

MQRO_EXCEPTION

Significado

Solicitar mensajes de informe de excepción, sin ningún dato de aplicación del mensaje original incluido en un mensaje de informe.

MQRO_EXCEPTION_WITH_DATA

Solicitar mensajes de informe de excepción, con los primeros 100 bytes de datos de aplicación del mensaje original incluidos en un mensaje de informe.

MQRO_EXCEPTION_WITH_FULL_DATA

Solicitar mensajes de informe de excepción, con todos los datos de aplicación del mensaje original incluidos en un mensaje de informe.

Estos valores corresponden a opciones de informe que se pueden especificar en el campo **Report** de un descriptor de mensaje.

De forma predeterminada, la propiedad no está establecida.

JMS_IBM_REPORT_EXPIRATION

Tipo de datos:

System.Int32

Propiedad de:

Mensaje

Solicitar mensajes de informe de caducidad, que especifican cuántos datos de aplicación del mensaje original se deben incluir en un mensaje de informe.

Los valores válidos de la propiedad son los siguientes:

Valor válido	Significado
MQRO_EXPIRATION	Solicitar mensajes de informe de caducidad, sin ninguno de los datos de aplicación del mensaje original incluido en un mensaje de informe.
MQRO_EXPIRATION_WITH_DATA	Solicitar mensajes de informe de caducidad, con los primeros 100 bytes de datos de aplicación del mensaje original incluidos en un mensaje de informe.
MQRO_EXPIRATION_WITH_FULL_DATA	Solicitar mensajes de informe de caducidad, con todos los datos de aplicación del mensaje original incluidos en un mensaje de informe.

Estos valores corresponden a opciones de informe que se pueden especificar en el campo **Report** de un descriptor de mensaje.

De forma predeterminada, la propiedad no está establecida.

JMS_IBM_REPORT_NAN

Tipo de datos:

System.Int32

Propiedad de:

Mensaje

Solicitar mensajes de informe de notificación de acción negativa.

Establezca la propiedad en MQRO_NAN para solicitar mensajes de informe de notificación de acción negativa. Si no necesita mensajes de informe de notificación de acción negativa, no establezca la propiedad. De forma predeterminada, la propiedad no está establecida.

El valor MQRO_NAN corresponde a una opción de informe que se puede especificar en el campo **Report** de un descriptor de mensaje.

JMS_IBM_REPORT_PAN

Tipo de datos:

System.Int32

Propiedad de:

Mensaje

Solicitar mensajes de informe de notificación de acción positiva.

Establezca la propiedad en MQRO_PAN para solicitar mensajes de informe de notificación de acción positiva. Si no necesita mensajes de informe de notificación de acción positiva, no establezca la propiedad. De forma predeterminada, la propiedad no está establecida.

El valor MQRO_PAN corresponde a una opción de informe que se puede especificar en el campo **Report** de un descriptor de mensaje.

JMS_IBM_REPORT_PASS_CORREL_ID

Tipo de datos:

System.Int32

Propiedad de:

Mensaje

Solicite que el identificador de correlación de cualquier mensaje de informe o de respuesta sea el mismo que el identificador de correlación del mensaje original.

Los valores válidos de la propiedad son los siguientes:

Valor válido

MQRO_PASS_CORREL_ID

Significado

Solicite que el identificador de correlación de cualquier mensaje de informe o de respuesta sea el mismo que el identificador de correlación del mensaje original.

MQRO_COPY_MSG_ID_TO_CORREL_ID

Solicitar que el identificador de correlación de cualquier mensaje de informe o de respuesta sea el mismo que el identificador de correlación del mensaje original.

Estos valores corresponden a opciones de informe que se pueden especificar en el **Report** campo de un descriptor de mensaje..

El valor predeterminado de la propiedad es MQRO_COPY_MSG_ID_TO_CORREL_ID.

JMS_IBM_REPORT_PASS_MSG_ID

Tipo de datos:

System.Int32

Propiedad de:

Mensaje

Solicite que el identificador de mensaje de cualquier mensaje de informe o respuesta sea el mismo que el identificador de mensaje del mensaje original.

Los valores válidos de la propiedad son los siguientes:

Valor válido

MQRO_PASS_MSG_ID

Significado

Solicite que el identificador de mensaje de cualquier mensaje de informe o respuesta sea el mismo que el identificador de mensaje del mensaje original.

MQRO_NEW_MSG_ID

Solicite que se genere un nuevo identificador de mensaje para cada mensaje de informe o respuesta.

Estos valores corresponden a opciones de informe que se pueden especificar en el campo **Report** de un descriptor de mensaje.

El valor predeterminado de la propiedad es MQRO_NEW_MSG_ID.

JMS_IBM_RETAIN

Tipo de datos:

System.Int32

Propiedad de:

Mensaje

Establecer esta propiedad indica al gestor de colas que trate un mensaje como una Publicación retenida. Cuando un suscriptor recibe mensajes de temas, podría recibir mensajes adicionales inmediatamente después de la suscripción, más allá de los mensajes recibidos en releases anteriores. Estos mensajes son las publicaciones retenidas opcionales para los temas suscritos. Para cada tema que coincida con la suscripción, si hay una publicación retenida, la publicación pasa a estar disponible para la entrega al consumidor de mensajes de la suscripción.

RETAIN_PUBLICATION es el único valor válido para esta propiedad. De forma predeterminada, esta propiedad no está establecida.

Nota: Esta propiedad solo es relevante en el dominio de publicación/suscripción únicamente.

JMS_IBM_SYSTEM_MESSAGEID

Tipo de datos:

Serie

Propiedad de:

Mensaje

Un identificador que identifica el mensaje de forma exclusiva dentro del bus de integración de servicios. Esta propiedad es de solo lectura.

Esta propiedad solo es relevante cuando una aplicación se conecta a un bus de integración de servicios.

JMSX_APPID

Tipo de datos:

Serie

Propiedad de:

Mensaje

El nombre de la aplicación que envió el mensaje.

Esta propiedad es la propiedad definida de JMS con el nombre JMS JMSXAppID. Para obtener más información sobre la propiedad, consulte la *Especificación Java del servicio de mensajes, versión 1.1*.

De forma predeterminada, la propiedad no está establecida.

Esta propiedad no es válida para una conexión en tiempo con un intermediario.

JMSX_DELIVERY_COUNT

Tipo de datos:

System.Int32

Propiedad de:

Mensaje

El número de intentos para entregar el mensaje.

Esta propiedad es la propiedad definida de JMS con el nombre de JMS JMSXDeliveryCount. Para obtener más información sobre la propiedad, consulte el *Java Message Service Especificación, versión 1.1*.

De forma predeterminada, la propiedad no está establecida.

Esta propiedad no es válida para una conexión en tiempo con un intermediario.

JMSX_GROUPID

Tipo de datos:

Serie

Propiedad de:

Mensaje

El identificador del grupo de mensajes al que pertenece el mensaje.

Esta propiedad es la propiedad definida de JMS con el nombre de JMS JMSXGroupID. Para obtener más información sobre la propiedad, consulte el *Java Message Service Especificación, versión 1.1*.

De forma predeterminada, la propiedad no está establecida.

Esta propiedad no es válida para una conexión en tiempo con un intermediario.

JMSX_GROUPSEQ

Tipo de datos:

System.Int32

Propiedad de:

Mensaje

El número de secuencia del mensaje de un grupo de mensajes.

Esta propiedad es la propiedad definida de JMS con el nombre de JMS JMSXGroupSeq. Para obtener más información sobre la propiedad, consulte el *Java Message Service Especificación, versión 1.1*.

De forma predeterminada, la propiedad no está establecida.

Esta propiedad no es válida para una conexión en tiempo con un intermediario.

JMSX_USERID

Tipo de datos:

Serie

Propiedad de:

Mensaje

El identificador de usuario asociado con la aplicación que envió el mensaje.

Esta propiedad es la propiedad definida de JMS con el nombre de JMS JMSXUserID. Para obtener más información sobre la propiedad, consulte el *Java Message Service Especificación, versión 1.1*.

De forma predeterminada, la propiedad no está establecida.

Esta propiedad no es válida para una conexión en tiempo con un intermediario.

XMSC_ASYNC_EXCEPTIONS

Tipo de datos:

System.Int32

Propiedad de:

ConnectionFactory

Objetos aplicables:

Nombre largo de la herramienta de administración JMS: ASYNCEXCEPTION

Nombre abreviado de la herramienta de administración JMS: AEX

Esta propiedad determina si XMS informa de un ExceptionListener solo cuando se interrumpe una conexión, o cuando se produce cualquier excepción de forma asíncrona en una llamada de la API XMS.

Esta propiedad se aplica a todas las conexiones creadas a partir de esta ConnectionFactory que tienen un ExceptionListener registrado.

Los valores válidos para esta propiedad son:

XMSC_ASYNC_EXCEPTIONS_ALL

Cualquier excepción detectada de forma asíncrona, fuera del ámbito de una llamada a API síncrona, y todas las excepciones de interrupción de conexión se envían al ExceptionListener.

XMSC_ASYNC_EXCEPTIONS_CONNECTIONBROKEN

Solo las excepciones que indican una conexión interrumpida se envían al ExceptionListener. Cualquier otra excepción que se produce durante el proceso asíncrono no se notifica al ExceptionListener y, por lo tanto, la aplicación no está informada de estas excepciones.

De forma predeterminada, esta propiedad está establecida en XMSC_ASYNC_EXCEPTIONS_ALL.

XMSC_CLIENT_ID

Tipo de datos:

Serie

Propiedad de:

ConnectionFactory

Objetos aplicables:

Nombre largo de la herramienta de administración JMS: CLIENTID

Nombre abreviado de la herramienta de administración JMS: CID

El identificador de cliente para una conexión.

Un identificador de cliente se utiliza solo para dar soporte a suscripciones duraderas en el dominio de publicación/suscripción y se ignora en el dominio punto a punto. Para obtener más información sobre cómo establecer identificadores de cliente, consulte [ConnectionFactoryes](#) y [objetos Connection](#).

Esta propiedad no es relevante para una conexión en tiempo real con un intermediario.

XMSC_CONNECTION_TYPE

Tipo de datos:

System.Int32

Propiedad de:

ConnectionFactory

El tipo del servidor de mensajería al que se conecta una aplicación.

Los valores válidos de la propiedad son los siguientes:

Valor válido	Significado
XMSC_CT_RTT	Una conexión en tiempo real con un intermediario.
XMSC_CT_WMQ	Una conexión con un gestor de colas de IBM MQ .
XMSC_CT_WPM	Una conexión a un WebSphere Application Server service integration bus.

De forma predeterminada, la propiedad no está establecida.

XMSC_DELIVERY_MODE

Tipo de datos:

System.Int32

Propiedad de:

Destino

Nombre utilizado en un URI:

persistence (para un destino IBM MQ)

deliveryMode (para un destino de proveedor de mensajería WebSphere predeterminado)

Objetos aplicables:

Nombre largo de la herramienta de administración JMS: PERSISTENCE

Nombre abreviado de la herramienta de administración JMS: PER

La modalidad de entrega de mensajes enviados al destino.

Los valores válidos de la propiedad son los siguientes:

Valor válido	Significado
XMSC_DELIVERY_NOT_PERSISTENT	Un mensaje enviado al destino es no persistente. Se ignora la modalidad de entrega predeterminada del productor de mensajes, o cualquier modalidad de entrega especificada en la llamada Send. Si el destino es una cola IBM MQ , el valor del atributo de cola <i>DefPersistence</i> también se ignora.
XMSC_DELIVERY_PERSISTENT	Un mensaje enviado al destino es persistente. Se ignora la modalidad de entrega predeterminada del productor de mensajes, o cualquier modalidad de entrega especificada en la llamada Send. Si el destino es una cola IBM MQ , el valor del atributo de cola <i>DefPersistence</i> también se ignora.
XMSC_DELIVERY_AS_APP	Un mensaje enviado al destino tiene la modalidad de entrega especificada en la llamada Send. Si la llamada Send no especifica ninguna modalidad de entrega, en su lugar, se utiliza la modalidad de entrega predeterminada del productor de mensajes. Si el destino es una cola IBM MQ , el valor del atributo de cola <i>DefPersistence</i> se ignora.
XMSC_DELIVERY_AS_DEST	Si el destino es una cola IBM MQ , un mensaje colocado en la cola tiene la modalidad de entrega especificada por el valor del atributo de cola <i>DefPersistence</i> . Se ignora la modalidad de entrega predeterminada del productor de mensajes, o cualquier modalidad de entrega especificada en la llamada Send. Si el destino no es una cola IBM MQ , el significado es el mismo que el de XMSC_DELIVERY_AS_APP.

El valor predeterminado es XMSC_DELIVERY_AS_APP.

XMSC_IC_PROVIDER_URL**Tipo de datos:**

Serie

Propiedad de:

InitialContext

Se utiliza para localizar el directorio de denominación JNDI de forma que no es necesario que el servicio de denominación COS esté en el mismo servidor que el servicio web.

XMSC_IC_SECURITY_AUTHENTICATION**Tipo de datos:**

Serie

Propiedad de:

InitialContext

Basado en la Java interfaz de contexto SECURITY_AUTHENTICATION. Esta propiedad solo es aplicable al contexto de denominación COS.

XMSC_IC_SECURITY_CREDENTIALS**Tipo de datos:**

Serie

Propiedad de:

InitialContext

Basado en la Java interfaz de contexto SECURITY_CREDENTIALS. Esta propiedad solo es aplicable al contexto de denominación COS.

XMSC_IC_SECURITY_PRINCIPAL**Tipo de datos:**

Serie

Propiedad de:

InitialContext

Basado en la Java interfaz de contexto SECURITY_PRINCIPAL. Esta propiedad solo es aplicable al contexto de denominación COS.

XMSC_IC_SECURITY_PROTOCOL**Tipo de datos:**

Serie

Propiedad de:

InitialContext

Basado en la Java interfaz Context SECURITY_PROTOCOL Esta propiedad solo es aplicable al contexto de denominación COS.

XMSC_IC_URL**Tipo de datos:**

Serie

Propiedad de:

InitialContext

Para contextos de LDAP y FileSystem, la dirección del repositorio que contiene objetos administrados.

Para contextos de LDAP y FileSystem, la dirección del repositorio que contiene objetos administrados.

XMSC_IS_SUBSCRIPTION_MULTICAST**Tipo de datos:**

System.Boolean

Propiedad de:

MessageConsumer

Indica si se están enviando mensajes al consumidor de mensajes mediante WebSphere MQ Multicast Transport. Esta propiedad es de solo lectura.

El valor de la propiedad es true si los mensajes se van a entregar al consumidor de mensajes mediante WebSphere MQ Multicast Transport. De lo contrario, el valor es false.

Esta propiedad solo es relevante para una conexión en tiempo real con un intermediario.

XMSC_IS_SUBSCRIPTION_RELIABLE_MULTICAST

Tipo de datos:

System.Boolean

Propiedad de:

MessageConsumer

Indica si los mensajes se están entregando al destinatario con WebSphere MQ Multicast Transport una calidad de servicio fiable. Esta propiedad es de solo lectura.

El valor de la propiedad es true si los mensajes se van a entregar al consumidor de mensajes mediante WebSphere MQ Multicast Transport con una calidad de servicio fiable. De lo contrario, el valor es false.

Esta propiedad solo es relevante para una conexión en tiempo real con un intermediario.

XMSC_JMS_MAJOR_VERSION

Tipo de datos:

System.Int32

Propiedad de:

ConnectionMetaData

El número de versión principal de la JMS especificación en la que XMS se basa. Esta propiedad es de solo lectura.

XMSC_JMS_MINOR_VERSION

Tipo de datos:

System.Int32

Propiedad de:

ConnectionMetaData

El número de versión menor de la JMS especificación en la que XMS se basa. Esta propiedad es de solo lectura.

XMSC_JMS_VERSION

Tipo de datos:

Serie

Propiedad de:

ConnectionMetaData

El identificador de versión de la JMS especificación en la que XMS se basa. Esta propiedad es de solo lectura.

XMSC_MAJOR_VERSION

Tipo de datos:

System.Int32

Propiedad de:

ConnectionMetaData

El número de versión del XMS cliente. Esta propiedad es de solo lectura.

XMSC_MINOR_VERSION

Tipo de datos:

System.Int32

Propiedad de:

ConnectionMetaData

El número de versión del XMS cliente. Esta propiedad es de solo lectura.

XMSC_PASSWORD

Tipo de datos:

Matriz de bytes

Propiedad de:

ConnectionFactory

Una contraseña que se puede utilizar para autenticar la aplicación cuando intenta conectarse a un servidor de mensajería. La contraseña se utiliza con la propiedad XMSC_USERID.

De forma predeterminada, la propiedad no está establecida.

Multi Si se está conectando a IBM MQ en Multiplataformas, y establece la propiedad XMSC_USERID de la fábrica de conexiones, debe coincidir con el **userid** del usuario conectado. Si no establece estas propiedades, el gestor de colas utiliza el **userid** del usuario conectado de forma predeterminada. Si necesita una autenticación de nivel de conexión adicional de usuarios individuales, puede escribir una salida de autenticación de cliente, que se configura en IBM MQ. Para obtener más información sobre cómo crear una salida de autenticación de cliente, consulte Planificación de la autenticación para una aplicación cliente.

z/OS Para autenticar el usuario cuando se conecta a IBM MQ for z/OS tendrá que utilizar una salida de seguridad.

XMSC_PRIORITY

Tipo de datos:

System.Int32

Propiedad de:

Destino

Nombre utilizado en un URI:

priority

La prioridad de los mensajes enviados al destino.

Los valores válidos de la propiedad son los siguientes:

Valor válido	Significado
Un entero dentro del rango de 0, la prioridad más baja, a 9, la prioridad más alta.	Un mensaje enviado al destino tiene la prioridad especificada. La prioridad predeterminada del productor de mensajes, o cualquier prioridad especificada en la llamada Send, se ignora. Si el destino es una cola IBM MQ, el valor del atributo de cola DefPriority también se ignora.
XMSC_PRIORITY_AS_APP	Un mensaje enviado al destino tiene la prioridad especificada en la llamada Send. Si la llamada Send no especifica ninguna prioridad, en su lugar, se utiliza la prioridad predeterminada del productor de mensajes. Si el destino es una cola IBM MQ, el valor del atributo de cola DefPriority se ignora.
XMSC_PRIORITY_AS_DEST	Si el destino es una cola IBM MQ, un mensaje colocado en la cola tiene la prioridad especificada por el valor del atributo de cola DefPriority. La prioridad predeterminada del productor de mensajes, o cualquier prioridad especificada en la llamada Send, se ignora. Si el destino no es una cola IBM MQ, el significado es el mismo que el de XMSC_PRIORITY_AS_APP.

El valor predeterminado es XMSC_PRIORITY_AS_APP.

WebSphere MQ Real-Time Transport y WebSphere MQ Multicast Transport no realizan ninguna acción basándose en la prioridad de un mensaje.

XMSC_PROVIDER_NAME

Tipo de datos:

Serie

Propiedad de:

ConnectionMetaData

XMS El proveedor del cliente. Esta propiedad es de solo lectura.

XMSC_RTT_BROKER_PING_INTERVAL

Tipo de datos:

System.Int32

Propiedad de:

ConnectionFactory

El intervalo de tiempo, en milisegundos, transcurrido el cual XMS .NET comprueba la conexión con un servidor de mensajería en tiempo real para detectar cualquier actividad. Si no se detecta ninguna actividad, el cliente inicia una conexión ping; la conexión se cierra si no se detecta ninguna respuesta de ping.

El valor predeterminado de la propiedad es 30000.

XMSC_RTT_CONNECTION_PROTOCOL

Tipo de datos:

System.Int32

Propiedad de:

ConnectionFactory

El protocolo de comunicación utilizado para una conexión en tiempo real con un intermediario.

El valor de la propiedad debe ser XMSC_RTT_CP_TCP, lo que significa una conexión en tiempo real con un intermediario sobre TCP/IP. El valor predeterminado es XMSC_RTT_CP_TCP.

XMSC_RTT_HOST_NAME

Tipo de datos:

Serie

Propiedad de:

ConnectionFactory

El nombre de host o la dirección IP del sistema en el que se ejecuta un intermediario.

Esta propiedad se utiliza con la propiedad XMSC_RTT_PORT para identificar el intermediario.

De forma predeterminada, la propiedad no está establecida.

XMSC_RTT_LOCAL_ADDRESS

Tipo de datos:

Serie

Propiedad de:

ConnectionFactory

El nombre de host o la dirección IP de la interfaz de red local que se va a utilizar para una conexión en tiempo real con un intermediario.

Esta propiedad solo es útil si el sistema en el cual se está ejecutando la aplicación tiene dos o más interfaces de red y necesita poder especificar qué interfaz se debe utilizar para una conexión en tiempo

real. Si el sistema tiene solo una interfaz de red, solo se puede utilizar dicha interfaz. Si el sistema tiene dos o más interfaces de red y la propiedad no está establecida, la interfaz se selecciona de forma aleatoria.

De forma predeterminada, la propiedad no está establecida.

XMSC_RTT_MULTICAST

Tipo de datos:

System.Int32

Propiedad de:

ConnectionFactory y Destination

Nombre utilizado en un URI:

multicast

El valor de multidifusión para una fábrica de conexiones o destino. Solo un destino que es un tema puede tener esta propiedad.

Una aplicación utiliza esta propiedad para habilitar la multidifusión con una conexión en tiempo real con un intermediario y, si la multidifusión está habilitada, para especificar la forma precisa en la cual se utiliza la multidifusión para entregar mensajes del intermediario a un consumidor de mensajes. La propiedad no tiene ningún efecto sobre cómo un productor de mensajes envía mensajes al intermediario.

Los valores válidos de la propiedad son los siguientes:

Valor válido

XMSC_RTT_MULTICAST_DISABLED

XMSC_RTT_MULTICAST_ASCF

XMSC_RTT_MULTICAST_ENABLED

XMSC_RTT_MULTICAST_RELIABLE

Significado

Los mensajes no se entregan a un consumidor de mensajes utilizando WebSphere MQ Multicast Transport. Este valor es el valor predeterminado para un objeto ConnectionFactory.

Los mensajes se entregan a un consumidor de mensajes de acuerdo con el valor de multidifusión para la fábrica de conexiones asociada al consumidor de mensajes. El valor de multidifusión para la fábrica de conexiones se indica en el momento cuando se crea la conexión. Este valor solo es válido para un objeto Destination, y es el valor predeterminado para un objeto Destination.

Si el tema se configura para la multidifusión en el intermediario, los mensajes se entregan a un consumidor de mensajes utilizando WebSphere MQ Multicast Transport. Se utiliza una calidad de servicio fiable si el tema está configurado para la multidifusión fiable.

Si el tema se ha configurado para la multidifusión fiable en el intermediario, los mensajes se entregan a un consumidor de mensajes utilizando WebSphere MQ Multicast Transport con una calidad de servicio fiable. Si el tema no está configurado para la multidifusión fiable, no puede crear un consumidor de mensajes para el tema.

Valor válido

XMSC_RTT_MULTICAST_NOT_RELIABLE

Significado

Si el tema se configura para la multidifusión en el intermediario, los mensajes se entregan a un consumidor de mensajes utilizando WebSphere MQ Multicast Transport. No se utiliza una calidad de servicio fiable, aunque el tema se haya configurado para la multidifusión fiable.

XMSC_RTT_PORT**Tipo de datos:**

System.Int32

Propiedad de:

ConnectionFactory

El número del puerto en el cual un intermediario escucha solicitudes entrantes. En el intermediario, debe configurar el nodo de proceso de mensajes Real-timeInput o Real-timeOptimizedFlow para escuchar en este puerto.

Esta propiedad se utiliza con la propiedad XMSC_RTT_HOST_NAME para identificar el intermediario.

El valor predeterminado de la propiedad es XMSC_RTT_DEFAULT_PORT, o 1506.

XMSC_TIME_TO_LIVE**Tipo de datos:**

System.Int32

Propiedad de:

Destino

Nombre utilizado en un URI:

caducidad (para un destino IBM MQ)

timeToLive (para un destino de proveedor de mensajería predeterminado de WebSphere)

El tiempo de vida para los mensajes enviados al destino.

Los valores válidos de la propiedad son los siguientes:

Valor válido

0

Un entero positivo

Significado

Un mensaje enviado al destino nunca caduca.

Un mensaje enviado al destino tiene el tiempo de vida especificado en milisegundos. Se ignora el periodo de vida predeterminado del productor de mensajes, o cualquier periodo de vida especificado en la llamada Send.

XMSC_TIME_TO_LIVE_AS_APP

Un mensaje enviado al destino tiene el periodo de tiempo especificado en la llamada Send. Si la llamada Send no especifica ningún periodo de tiempo, en su lugar, se utiliza el periodo de tiempo predeterminado del productor de mensajes.

El valor predeterminado es XMSC_TIME_TO_LIVE_AS_APP.

XMSC_USERID**Tipo de datos:**

Serie

Propiedad de:

ConnectionFactory

Un identificador de usuario que se puede utilizar para autenticar la aplicación cuando intenta conectarse a un servidor de mensajería. El identificador de usuario se utiliza con la propiedad `XMSC_PASSWORD`.

De forma predeterminada, la propiedad no está establecida.

Multi Si está conectándose a IBM MQ for Multiplatforms, y establece la propiedad `XMSC_USERID` de la fábrica de conexiones, debe coincidir con el **userid** del usuario conectado. Si no establece estas propiedades, el gestor de colas utiliza el **userid** del usuario conectado de forma predeterminada. Si necesita una autenticación de nivel de conexión adicional de usuarios individuales, puede escribir una salida de autenticación de cliente que se configura en IBM MQ. Para obtener más información sobre cómo crear una salida de autenticación de cliente, consulte [Planificación de la autenticación para una aplicación cliente](#).

z/OS Para autenticar el usuario cuando se conecta a IBM MQ for z/OS tendrá que utilizar una salida de seguridad.

XMSC_VERSION

Tipo de datos:

Serie

Propiedad de:

ConnectionMetaData

El identificador de versión del XMSCcliente. Esta propiedad es de solo lectura.

XMSC_WMQ_TIPO_DE_APLICACIÓN_DE_EQILIBRIO

Tipo de datos:

System.Int32

Propiedad de:

ConnectionFactory

Los valores válidos de la propiedad son los siguientes:

Valor válido

`XMSC_WMQ_BALANCEO_TIPO_APLICACIÓN_SIMPLE`

`XMSC_WMQ_EQILIBRIO_TIPO_APLICACIÓN_SOLICITUD_RESPUESTA`

Significado

Equilibrio simple; no se aplican reglas específicas además de las descritas en [Influir en el reequilibrio de aplicaciones en clústeres uniformes](#). Éste es el valor predeterminado.

Equilibrio de solicitud-respuesta; después de cada llamada `MQPUT`, se espera una llamada `MQGET` coincidente para un mensaje de respuesta. El equilibrio se retrasa hasta que se recibe dicho mensaje o hasta que se supera el mensaje de solicitud `EXPIRY`

Además, esta propiedad se puede configurar en el `client.ini` archivo. El orden de preferencia es:

1. Propiedades configuradas en la aplicación
2. Coincide con la [estrofa](#) denominada «Application» en el `mqclient.ini` archivo.
3. [Estrofa de valores predeterminados de la aplicación](#) en el `mqclient.ini` archivo.

XMSC_WMQ_OPCIONES_DE_EQILIBRIO

Tipo de datos:

System.Int32

Propiedad de:

ConnectionFactory

Los valores válidos de la propiedad son los siguientes:

Valor válido

XMSC_WMQ OPCIONES_DE_EQUILIBRIO_NINGUNA

XMSC_WMQ OPCIONES_DE_EQUILIBRIO_IGNOREAR_TRANS ACCIONES

Valor correspondiente

No se han establecido opciones. Se trata del valor predeterminado

Establecer esta opción permite que las aplicaciones se reequilibren incluso si están en medio de una transacción.

Además, esta propiedad se puede configurar en el `client.ini` archivo. El orden de preferencia es:

1. Propiedades configuradas en la aplicación
2. Coincide con [la estrofa](#) denominada «Application» en el `mqclient.ini` archivo.
3. [Estrofa de valores predeterminados de la aplicación](#) en el `mqclient.ini` archivo.

XMSC_WMQ_TIEMPO_DE_ESPERA_DE_EQUILIBRIO**Tipo de datos:**

System.Int32

Propiedad de:

ConnectionFactory

Los valores válidos de la propiedad son los siguientes:

Valor válido

XMSC_WMQ_BALANCING_TIMEOUT_IMMEDIATE

XMSC_WMQ_BALANCING_TIMEOUT_AS_DEFAULT

XMSC_WMQ_BALANCING_TIMEOUT_NEVER

Significado

Se produce un tiempo de espera inmediato

El valor de tiempo de espera predeterminado establecido. Se trata del valor predeterminado

No se produce ningún tiempo de espera

Nota: Debe proporcionar un valor sólo a partir de los valores definidos o un valor de 0-999999999 segundos.

Además, esta propiedad se puede configurar en el `client.ini` archivo. El orden de preferencia es:

1. Propiedades configuradas en la aplicación
2. Coincide con [la estrofa](#) denominada «Application» en el `mqclient.ini` archivo.
3. [Estrofa de valores predeterminados de la aplicación](#) en el `mqclient.ini` archivo.

XMSC_WMQ_BROKER_CONTROLQ**Tipo de datos:**

Serie

Propiedad de:

ConnectionFactory

El nombre de la cola de control que utiliza un broker.

El valor predeterminado de la propiedad es `SYSTEM.BROKER.CONTROL.QUEUE`.

Esta propiedad solo es relevante en el dominio de publicación/suscripción.

XMSC_WMQ_BROKER_PUBQ

Tipo de datos:

Serie

Propiedad de:

ConnectionFactory

El nombre de la cola supervisada por un intermediario donde las aplicaciones envían mensajes que publican.

El valor predeterminado de la propiedad es SYSTEM.BROKER.DEFAULT.STREAM.

Esta propiedad solo es relevante en el dominio de publicación/suscripción.

XMSC_WMQ_BROKER_QMGR

Tipo de datos:

Serie

Propiedad de:

ConnectionFactory

El nombre del gestor de colas al que está conectado un intermediario.

De forma predeterminada, la propiedad no está establecida.

Esta propiedad solo es relevante en el dominio de publicación/suscripción.

XMSC_WMQ_BROKER_SUBQ

Tipo de datos:

Serie

Propiedad de:

ConnectionFactory

El nombre de la cola de suscriptor para un consumidor de mensajes no duraderos.

El nombre de la cola de suscriptor debe empezar con los caracteres siguientes:

SYSTEM.JMS.ND.

Si desea que todos los consumidores de mensajes no duraderos compartan una cola de suscriptor, especifique el nombre completo de la cola compartida. Debe existir una cola con el nombre especificado antes de que una aplicación pueda crear un consumidor de mensajes no duraderos.

Si desea que cada consumidor de mensajes no duraderos recupere mensajes de su propia cola de suscriptores exclusiva, especifique un nombre de cola que termine con un asterisco (*). A continuación, cuando una aplicación crea un consumidor de mensajes no duradero, el cliente de XMS crea una cola dinámica para uso exclusivo del consumidor de mensajes. El cliente de XMS utiliza el valor de la propiedad para establecer el contenido del campo **DynamicQName** en el descriptor de objetos que se utiliza para crear la cola dinámica.

El valor predeterminado de la propiedad es SYSTEM.JMS.ND.SUBSCRIBER.QUEUE, lo que significa que XMS utiliza el enfoque de cola compartida de forma predeterminada.

Esta propiedad solo es relevante en el dominio de publicación/suscripción.

XMSC_WMQ_BROKER_VERSION

Tipo de datos:

System.Int32

Propiedad de:

ConnectionFactory y Destination

Nombre utilizado en un URI:

brokerVersion

El tipo de intermediario utilizado por la aplicación para una conexión o para el destino. Solo un destino que es un tema puede tener esta propiedad.

Los valores válidos de la propiedad son los siguientes:

Valor válido	Significado
XMSC_WMQ_BROKER_V1	La aplicación utiliza un intermediario de publicación/suscripción de IBM MQ . La aplicación también puede utilizar este valor si migra de la publicación/suscripción de IBM MQ a WebSphere Message Broker pero no ha cambiado la aplicación.
XMSC_WMQ_BROKER_V2	La aplicación está utilizando un intermediario de IBM Integration Bus.
XMSC_WMQ_BROKER_UNSPECIFIED	Después de que se haya migrado el intermediario, establezca esta propiedad para que las cabeceras RFH2 dejen de utilizarse. Después de la migración, esta propiedad deja de ser relevante.

El valor predeterminado para una fábrica de conexiones es XMSC_WMQ_BROKER_UNSPECIFIED pero, de forma predeterminada, la propiedad no está establecida para un destino. Establecer la propiedad para un destino sustituirá cualquier valor especificado por la propiedad de la fábrica de conexiones.

XMSC_WMQ_CCDTURL

Tipo de datos:

System.String

Propiedad de:

ConnectionFactory

Objetos aplicables:

Nombre largo de la herramienta de administración JMS: CCDTURL

Nombre abreviado de la herramienta de administración JMS: CCDT

Un localizador uniforme de recursos (URL) que identifica el nombre y la ubicación del archivo que contiene la tabla de la definición de canal de cliente y, también, especifica cómo se puede acceder al archivo.

De forma predeterminada, esta propiedad no está establecida.

XMSC_WMQ_CCSID

Tipo de datos:

System.Int32

Propiedad de:

Destino

Nombre utilizado en un URI:

CCSID

El identificador (CCSID) del conjunto de caracteres codificados, o página de códigos, en el que se encuentran las cadenas de datos de caracteres del cuerpo de un mensaje cuando el XMS cliente reenvía el mensaje al destinatario. Si se establece para un mensaje individual, la propiedad JMS_IBM_CHARACTER_SET altera temporalmente el CCSID especificado para el destino por esta propiedad.

El valor predeterminado de la propiedad es 1208.

Esta propiedad es relevante solo para los mensajes enviados al destino, no para los mensajes recibidos del destino.

XMSC_WMQ_CHANNEL

Tipo de datos:

Serie

Propiedad de:

ConnectionFactory

Objetos aplicables:

Nombre largo de la herramienta de administración JMS: CHANNEL

Nombre abreviado de la herramienta de administración JMS: CHAN

El nombre del canal que se va a utilizar para una conexión.

De forma predeterminada, la propiedad no está establecida.

Esta propiedad solo es relevante cuando una aplicación se conecta a un gestor de colas en modalidad de cliente.

XMSC_WMQ_CLIENT_RECONNECT_OPTIONS

Tipo de datos:

Serie

Propiedad de:

ConnectionFactory

Objetos aplicables:

Nombre largo de la herramienta de administración JMS: CLIENTRECONNECTOPTIONS

Nombre abreviado de la herramienta de administración JMS: CROPT

Esta propiedad especifica las opciones de reconexión de cliente para nuevas conexiones creadas por esta fábrica. Se encuentra en XMSC, y es una de:

- WMQ_CLIENT_RECONNECT_AS_DEF (default). Utilice el valor especificado en el archivo `mqclient.ini`. Establezca el valor utilizando la propiedad **DefRecon** dentro de la stanza Channels. Se puede establecer en uno de estos valores:
 1. Sí. Se comporta como la opción WMQ_CLIENT_RECONNECT
 2. NO. Valor predeterminado. No especifica ninguna opción de reconexión.
 3. QMGR. Se comporta como la opción WMQ_CLIENT_RECONNECT_Q_MGR
 4. DISABLED. Se comporta como la opción WMQ_CLIENT_RECONNECT_DISABLED
- WMQ_CLIENT_RECONNECT. Reconectarse a cualquiera de los gestores de colas especificados en la lista de nombres de conexión.
- WMQ_CLIENT_RECONNECT_Q_MGR. Se reconecta al mismo gestor de colas al que estaba conectado originalmente. Devuelve MQRC_RECONNECT_QMID_MISMATCH si el gestor de colas al que intenta conectarse (especificado en la lista de nombres de conexión) tiene un QMID diferente al del gestor de colas al que estaba conectado originalmente.
- WMQ_CLIENT_RECONNECT_DISABLED. La reconexión está inhabilitada.

XMSC_WMQ_CLIENT_RECONNECT_TIMEOUT

Tipo de datos:

Serie

Propiedad de:

ConnectionFactory

Objetos aplicables:

Nombre largo de la herramienta de administración JMS: CLIENTRECONNECTTIMEOUT

Nombre abreviado de la herramienta de administración JMS: CRT

La propiedad XMSC_WMQ_CLIENT_RECONNECT_TIMEOUT sólo es válida para el cliente XMS .NET gestionado.

Esta propiedad especifica la duración de tiempo, en segundos, que una conexión de cliente intenta reconectarse.

Después de intentar reconectarse para esta duración de tiempo, el cliente fallará con MQRC_RECONNECT_FAILED. El valor predeterminado para esta propiedad es XMSC_WMQ_CLIENT_RECONNECT_TIMEOUT_DEFAULT.

El valor predeterminado de esta propiedad es 1800.

XMSC_WMQ_CONNECTION_MODE

Tipo de datos:

System.Int32

Propiedad de:

ConnectionFactory

La modalidad a través de la cual una aplicación se conecta a un gestor de colas.

Los valores válidos de la propiedad son los siguientes:

Valor válido	Significado
XMSC_WMQ_CM_BINDINGS	Una conexión con un gestor de colas en modalidad de enlaces, para un rendimiento óptimo. Este valor es el valor predeterminado para C/C++.
XMSC_WMQ_CM_CLIENT	Una conexión con un gestor de colas en modalidad cliente, para garantizar una pila totalmente gestionada. Este valor es el valor predeterminado para .NET.
XMSC_WMQ_CM_CLIENT_UNMANAGED (solo para .NET)	Una conexión con un gestor de colas que fuerza una pila de cliente no gestionada.

XMSC_WMQ_CONNECTION_NAME_LIST

Tipo de datos:

Serie

Propiedad de:

ConnectionFactory

Objetos aplicables:

Nombre largo de la herramienta de administración JMS: CONNECTIONNAMELIST

Nombre abreviado de la herramienta de administración JMS: CNLIST

Esta propiedad especifica los hosts a los que el cliente intenta reconectarse después de que se hayan interrumpido sus conexiones.

La lista de nombres de conexión es una lista separada por comas de pares de host/puerto ip. El valor predeterminado para esta propiedad es WMQ_CONNECTION_NAME_LIST_DEFAULT.

Por ejemplo, 127.0.0.1(1414) , host2.example.com(1400)

El valor predeterminado de esta propiedad es localhost(1414).

XMSC_WMQ_DUR_SUBQ

Tipo de datos:

Serie

Propiedad de:

Destino

El nombre de la cola de suscriptor para un suscriptor duradero que está recibiendo mensajes del destino. Solo un destino que es un tema puede tener esta propiedad.

El nombre de la cola de suscriptor debe empezar con los caracteres siguientes:

SYSTEM.JMS.D.

Si desea que todos los suscriptores duraderos compartan una cola de suscriptor, especifique el nombre completo de la cola compartida. Una cola con el nombre especificado debe existir antes de que una aplicación pueda crear un suscriptor duradero.

Si desea que cada suscriptor duradero recupere mensajes de su propia cola de suscriptor exclusiva, especifique un nombre de cola que termine con un asterisco (*). A continuación, cuando una aplicación crea un suscriptor duradero, el cliente XMS crea una cola dinámica para uso exclusivo del suscriptor duradero. El cliente de XMS utiliza el valor de la propiedad para establecer el contenido del campo **DynamicQName** en el descriptor de objetos que se utiliza para crear la cola dinámica.

El valor predeterminado de la propiedad es SYSTEM.JMS.D.SUBSCRIBER.QUEUE, lo que significa que XMS utiliza el enfoque de cola compartida de forma predeterminada.

Esta propiedad solo es relevante en el dominio de publicación/suscripción.

XMSC_WMQ_ENCODING

Tipo de datos:

System.Int32

Propiedad de:

Destino

Cómo se representan los datos numéricos del cuerpo de un mensaje cuando el XMS cliente reenvía el mensaje al destinatario. Si se establece para un mensaje individual, la propiedad JMS_IBM_ENCODING anula la codificación especificada para el destino por esta propiedad. La propiedad especifica la representación de enteros binarios, enteros decimales empaquetados y números de coma flotante.

Los valores válidos de la propiedad son los mismos que los valores que se pueden especificar en el **Encoding** campo de un descriptor de mensaje.

Una aplicación puede utilizar las constantes con nombre siguientes para establecer la propiedad:

Constante con nombre	Significado
MQENC_INTEGER_NORMAL	Codificación de entero normal
MQENC_INTEGER_REVERSED	Codificación de entero inverso
MQENC_DECIMAL_NORMAL	Codificación de decimal empaquetado normal
MQENC_DECIMAL_REVERSED	Codificación de decimal empaquetado inverso
MQENC_FLOAT_IEEE_NORMAL	Codificación de coma flotante IEEE normal
MQENC_FLOAT_IEEE_REVERSED	Codificación de coma flotante IEEE inversa
 MQENC_FLOAT_S390	z/OS codificación de coma flotante de arquitectura
MQENC_NATIVE	Codificación de máquina nativa

Para formar un valor para la propiedad, la aplicación puede añadir tres de estas constantes del modo siguiente:

- Una constante cuyo nombre empieza con MQENC_INTEGER, para especificar la representación de enteros binarios
- Una constante cuyo nombre empieza con MQENC_DECIMAL, para especificar la representación de enteros decimales empaquetados
- Una constante cuyo nombre empieza con MQENC_FLOAT, para especificar la representación de números de coma flotante

De forma alternativa, la aplicación puede establecer la propiedad en MQENC_NATIVE, cuyo valor depende del entorno.

El valor predeterminado de la propiedad es MQENC_NATIVE.

Esta propiedad es relevante solo para los mensajes enviados al destino, no para los mensajes recibidos del destino.

XMSC_WMQ_FAIL_IF QUIESCE

Tipo de datos:

System.Int32

Propiedad de:

ConnectionFactory y Destination

Nombre utilizado en un URI:

failIfQuiesce

Objetos aplicables:

Nombre largo de la herramienta de administración JMS: FAILIFQUIESCE

Nombre abreviado de la herramienta de administración JMS: FIQ

Indica si las llamadas a determinados métodos fallan si el gestor de colas al que está conectada la aplicación está en un estado de inmovilización.

Los valores válidos de la propiedad son los siguientes:

Valor válido	Significado
XMSC_WMQ_FIQ_YES	Las llamadas a determinados métodos fallan si el gestor de colas está en un estado de inmovilización. Cuando la aplicación detecta que el gestor de colas está en fase de inmovilización, la aplicación puede completar su tarea inmediata y cerrar la conexión, lo que permite que se detenga el gestor de colas.
XMSC_WMQ_FIQ_NO	No falla ninguna llamada de método porque el gestor está en un estado de inmovilización. Si especifica esta valor, la aplicación no puede detectar que el gestor de colas está en fase de inmovilización. La aplicación podría seguir realizando operaciones en el gestor de colas y, por lo tanto, impedir que se detenga el gestor de colas.

El valor predeterminado para una fábrica de conexiones es XMSC_WMQ_FIQ_YES pero, de forma predeterminada, la propiedad no está establecida para un destino. Establecer la propiedad para un destino sustituirá cualquier valor especificado por la propiedad de la fábrica de conexiones.

XMSC_WMQ_MESSAGE_BODY

Tipo de datos:

System.Int32

Propiedad de:

Destino

Esta propiedad determina si una XMS aplicación procesa el campo « MQRFH2 » de un IBM MQ mensaje como parte de la carga útil del mensaje (es decir, como parte del cuerpo del mensaje).

Nota: Al enviar mensajes a un destino, la propiedad XMSC_WMQ_MESSAGE_BODY reemplaza a la propiedad de destino XMS existente XMSC_WMQ_TARGET_CLIENT.

Los valores válidos para esta propiedad son:

XMSC_WMQ_MESSAGE_BODY_JMS

Recibir: el tipo de mensaje y el cuerpo de XMS de entrada están determinados por el contenido de la MQRFH2 (si está presente) o la MQMD (si no hay ninguna MQRFH2) en el mensaje IBM MQ recibido.

Enviar: El cuerpo del mensaje XMS de salida contiene una cabecera MQRFH2 prependeda y generada automáticamente basándose en las propiedades y los campos de cabecera de XMS Message.

XMSC_WMQ_MESSAGE_BODY_MQ

Recibir: El tipo de mensaje XMS entrante es siempre `ByteMessage`, independientemente del contenido del mensaje IBM MQ recibido o del campo de formato del MQMD recibido. El cuerpo del mensaje XMS son los datos de mensaje sin modificar devueltos por la llamada de API del proveedor de mensajería subyacente. El juego de caracteres y la codificación de los datos en el cuerpo del mensaje se determina mediante los campos `CodedCharSetId` y `Encoding` del MQMD. El formato de los datos del cuerpo del mensaje se determina mediante el campo `Format` del MQMD.

Enviar: el cuerpo del mensaje XMS de salida contiene la carga útil de la aplicación tal cual; y no se añade ninguna cabecera IBM MQ generada automáticamente al cuerpo.

XMSC_WMQ_MESSAGE_BODY_UNSPECIFIED

Recibir: el cliente XMS determina un valor adecuado para esta propiedad. En la vía de acceso de recepción, este valor es el valor de propiedad `WMQ_MESSAGE_BODY_JMS`.

Enviar: El cliente XMS determina un valor adecuado para esta propiedad. En la vía de acceso de envío, este valor es el valor de propiedad `XMSC_WMQ_TARGET_CLIENT`.

De forma predeterminada, esta propiedad está establecida en `XMSC_WMQ_MESSAGE_BODY_UNSPECIFIED`.

XMSC_WMQ_MQMD_MESSAGE_CONTEXT

Tipo de datos:

`System.Int32`

Propiedad de:

Destino

Determina qué nivel de contexto del mensaje debe establecer la XMS aplicación. La aplicación debe estar en ejecución con la autoridad de contexto adecuada para que esta propiedad tenga efecto.

Los valores válidos para esta propiedad son:

XMSC_WMQ_MDCTX_DEFAULT

Para mensajes de salida, la llamada de la API `MQOPEN` y la estructura `MQPMO` no especifican ninguna opción de contexto de mensaje explícita.

XMSC_WMQ_MDCTX_SET_IDENTITY_CONTEXT

La llamada de la API `MQOPEN` especifica la opción de contexto de mensaje `MQOO_SET_IDENTITY_CONTEXT` y la estructura `MQPMO` especifica `MQPMO_SET_IDENTITY_CONTEXT`.

XMSC_WMQ_MDCTX_SET_ALL_CONTEXT

La llamada de la API `MQOPEN` especifica la opción de contexto de mensaje `MQOO_SET_ALL_CONTEXT` y la estructura `MQPMO` especifica `MQPMO_SET_ALL_CONTEXT`.

De forma predeterminada, esta propiedad está establecida en `XMSC_WMQ_MDCTX_DEFAULT`.

Nota: Esta propiedad no es relevante cuando una aplicación se conecta a WebSphere Application Server service integration bus.

Las propiedades siguientes requieren que la propiedad `XMSC_WMQ_MQMD_MESSAGE_CONTEXT` esté establecida en el valor de propiedad `XMSC_WMQ_MDCTX_SET_IDENTITY_CONTEXT` o el valor de propiedad `XMSC_WMQ_MDCTX_SET_ALL_CONTEXT` al enviar un mensaje para poder tener el efecto deseado:

- `JMS_IBM_MQMD_USERIDENTIFIER`
- `JMS_IBM_MQMD_ACCOUNTINGTOKEN`
- `JMS_IBM_MQMD_APPLIDENTITYDATA`

Las propiedades siguientes requieren que la propiedad `XMSC_WMQ_MQMD_MESSAGE_CONTEXT` esté establecida en el valor de propiedad `XMSC_WMQ_MDCTX_SET_ALL_CONTEXT` al enviar un mensaje para poder tener el efecto deseado.

- JMS_IBM_MQMD_PUTAPPLTYPE
- JMS_IBM_MQMD_PUTAPPLNAME
- JMS_IBM_MQMD_PUTDATE
- JMS_IBM_MQMD_PUTTIME
- JMS_IBM_MQMD_APPLORIGINDATA

XMSC_WMQ_MQMD_READ_ENABLED

Tipo de datos:

System.Int32

Propiedad de:

Destino

Esta propiedad determina si una XMS aplicación puede extraer los valores de los campos MQMD o no.

Los valores válidos para esta propiedad son:

XMSC_WMQ_READ_ENABLED_NO

Al enviar mensajes, las propiedades JMS_IBM_MQMD* en un mensaje enviado no se actualizan para reflejar los valores de campo actualizados en el MQMD.

Al recibir mensajes, ninguna de las propiedades JMS_IBM_MQMD* está disponible en un mensaje recibido, aunque el emisor haya establecido algunas o todas las propiedades.

XMSC_WMQ_READ_ENABLED_YES

Al enviar mensajes, todas las propiedades JMS_IBM_MQMD* en un mensaje enviado se actualizan para reflejar los valores de campo actualizados en el MQMD, incluyendo aquellas propiedades que el emisor no ha establecido de forma explícita.

Al recibir mensajes, todas las propiedades JMS_IBM_MQMD* están disponibles en un mensaje recibido, incluyendo aquellas propiedades que el emisor no ha establecido explícitamente.

De forma predeterminada, esta propiedad está establecida en XMSC_WMQ_READ_ENABLED_NO.

XMSC_WMQ_MQMD_WRITE_ENABLED

Tipo de datos:

System.Int32

Propiedad de:

Destino

Esta propiedad determina si una XMS aplicación puede establecer los valores de los campos MQMD o no.

Los valores válidos para esta propiedad son:

XMSC_WMQ_WRITE_ENABLED_NO

Todas las propiedades JMS_IBM_MQMD* se ignoran y sus valores no se copian en la estructura MQMD subyacente.

XMSC_WMQ_WRITE_ENABLED_YES

Se procesan las propiedades JMS_IBM_MQMD*. Sus valores se copian en la estructura MQMD subyacente.

De forma predeterminada, esta propiedad está establecida en XMSC_WMQ_WRITE_ENABLED_NO.

XMSC_WMQ_PUT_ASYNC_ALLOWED

Tipo de datos:

System.Int32

Propiedad de:

Destino

Esta propiedad determina si los productores de mensajes están permitidos para utilizar operaciones de transferencia asíncrona para enviar mensajes a este destino.

Los valores válidos para esta propiedad son:

XMSC_WMQ_PUT_ASYNC_ALLOWED_AS_DEST

Determine si las operaciones de transferencia asíncrona están permitidas consultando la definición de cola o tema.

XMSC_WMQ_PUT_ASYNC_ALLOWED_AS_Q_DEF

Determine si las operaciones de transferencia asíncrona están permitidas consultando la definición de cola.

XMSC_WMQ_PUT_ASYNC_ALLOWED_AS_TOPIC_DEF

Determine si las operaciones de transferencia asíncrona están permitidas consultando la definición de tema.

XMSC_WMQ_PUT_ASYNC_ALLOWED_DISABLED

Las operaciones de transferencia asíncrona no están permitidas.

XMSC_WMQ_PUT_ASYNC_ALLOWED_ENABLED

Las operaciones de transferencia asíncrona están permitidas.

De forma predeterminada, esta propiedad está establecida en XMSC_WMQ_PUT_ASYNC_ALLOWED_AS_DEST.

Nota: Esta propiedad no es relevante cuando una aplicación se conecta a WebSphere Application Server service integration bus.

XMSC_WMQ_READ_AHEAD_ALLOWED

Tipo de datos:

System.Int32

Propiedad de:

Destino

Esta propiedad determina si los consumidores de mensajes y los navegadores de colas están autorizados para utilizar la lectura anticipada para obtener mensajes no persistentes y no transaccionales de este destino en un almacenamiento intermedio interno antes de recibirlos.

Los valores válidos para esta propiedad son:

XMSC_WMQ_READ_AHEAD_ALLOWED_AS_Q_DEF

Determine si se permite la lectura anticipada consultando la definición de cola.

XMSC_WMQ_READ_AHEAD_ALLOWED_AS_TOPIC_DEF

Determine si se permite la lectura anticipada consultando la definición de tema.

XMSC_WMQ_READ_AHEAD_ALLOWED_AS_DEST

Determine si se permite la lectura anticipada consultando la definición de cola o tema.

XMSC_WMQ_READ_AHEAD_ALLOWED_DISABLED

La lectura anticipada no está permitida mientras se consumen o navegan mensajes.

XMSC_WMQ_READ_AHEAD_ALLOWED_ENABLED

La lectura anticipada está permitida.

De forma predeterminada, esta propiedad está establecida en XMSC_WMQ_READ_AHEAD_ALLOWED_AS_DEST.

XMSC_WMQ_READ_AHEAD_CLOSE_POLICY

Tipo de datos:

System.Int32

Propiedad de:

Destino

Esta propiedad determina, para los mensajes que se están entregando a un escucha de mensajes asíncronos, qué sucede con los mensajes en el almacenamiento de lectura anticipada interno, cuando se cierra el consumidor de mensajes.

Esta propiedad es aplicable en la especificación de opciones de cierre de cola al consumir mensajes de un destino y no es aplicable al enviar mensajes a un destino.

Esta propiedad se ignora para los Examinadores de colas porque, durante la exploración, los mensajes siguen estando disponibles en las colas.

Los valores válidos para esta propiedad son:

XMSC_WMQ_READ_AHEAD_CLOSE_POLICY_DELIVER_CURRENT

Solo se completa la invocación del escucha de mensajes actual antes de la devolución, posiblemente dejando mensajes en el almacenamiento intermedio de lectura anticipada interno que, después, se descartan.

XMSC_WMQ_READ_AHEAD_CLOSE_POLICY_DELIVER_ALL

Todos los mensajes del almacenamiento intermedio de lectura anticipada interno se entregan al escucha de mensajes de aplicación antes de la devolución.

De forma predeterminada, esta propiedad está establecida en XMSC_WMQ_READ_AHEAD_CLOSE_POLICY_DELIVER_CURRENT.

Nota:

Terminación de aplicación anómala

Todos los mensajes del almacenamiento intermedio de lectura anticipada se pierden cuando una aplicación XMS termina de forma abrupta.

Implicaciones para las transacciones

La lectura anticipada está inhabilitada cuando las aplicaciones utilizan transacciones. De esta forma, la aplicación no está viendo ninguna diferencia en el comportamiento cuando utilizando sesiones con transacción.

Implicaciones de las modalidades de conocimiento de sesión

La lectura anticipada está habilitada en una sesión sin transacción cuando las modalidades de acuse de recibo es XMSC_AUTO_ACKNOWLEDGE o bien XMSC_DUPS_OK_ACKNOWLEDGE. La lectura anticipada está inhabilitada si la modalidad de acuse de recibo de sesión es XMSC_CLIENT_ACKNOWLEDGE independientemente de sesiones con transacción o sesiones sin transacción.

Implicaciones para los navegadores de colas y los selectores de navegador de colas

Los navegadores de colas y los selectores de navegador de colas, utilizados en aplicaciones XMS , obtienen la ventaja de rendimiento de la lectura anticipada. El cierre del navegador de colas no degrada el rendimiento porque el mensaje sigue estando disponible en la cola para operaciones posteriores. No hay otras implicaciones para los navegadores de colas y los selectores de navegador de colas aparte de las ventajas de rendimiento de la lectura anticipada.

XMSC_WMQ_HOST_NAME

Tipo de datos:

Serie

Propiedad de:

ConnectionFactory

Objetos aplicables:

Nombre largo de la herramienta de administración JMS: HOSTNAME

Nombre abreviado de la herramienta de administración JMS: HOST

El nombre de host o la dirección IP del sistema en el cual se ejecuta un gestor de colas.

Esta propiedad solo se utiliza cuando una aplicación se conecta a un gestor de colas en modalidad cliente. La propiedad se utiliza con la propiedad `XMSC_WMQ_PORT` para identificar el gestor de colas.

El valor predeterminado de la propiedad es `localhost`.

XMSC_WMQ_LOCAL_ADDRESS

Tipo de datos:

Serie

Propiedad de:

ConnectionFactory

Objetos aplicables:

Nombre largo de la herramienta de administración JMS: LOCALADDRESS

Nombre abreviado de la herramienta de administración JMS: LA

Para una conexión a un gestor de colas, esta propiedad especifica la interfaz de red local que se va a utilizar, o el puerto local o el rango de puertos locales que se van a utilizar, o ambos.

El valor de la propiedad es una serie con el formato siguiente:

[nombre_host][(puerto_bajo)[,puerto_alto]]

Los significados de las variables son los siguientes:

nombre_host

El nombre de host o la dirección IP de la interfaz de red local que se va a utilizar para la conexión.

Proporcionar esta información solo es necesario si el sistema en el cual se está ejecutando la aplicación tiene dos o más interfaces de red y tendrá que poder especificar qué interfaz se debe utilizar para la conexión. Si el sistema tiene solo una interfaz de red, solo se puede utilizar dicha interfaz. Si el sistema tiene dos o más interfaces de red y no especifica qué interfaz se debe utilizar, la interfaz se selecciona de forma aleatoria.

puerto_bajo

El número del puerto local que se va a utilizar para la conexión.

Si *puerto_alto* también se especifica, *puerto_bajo* se interpreta como el número de puerto inferior en un rango de números de puerto.

puerto_alto

El número de puerto superior en un rango de números de puerto. Se debe utilizar uno de los puertos del rango especificado para la conexión.

La longitud máxima de la serie es 48 caracteres.

Aquí aparecen algunos ejemplos de valores válidos de la propiedad:

JUPITER
9.20.4.98
JUPITER(1000)
9.20.4.98(1000,2000)
(1000)
(1000,2000)

De forma predeterminada, la propiedad no está establecida.

Esta propiedad solo es relevante cuando una aplicación se conecta a un gestor de colas en modalidad de cliente.

XMSC_WMQ_MESSAGE_SELECTION

Tipo de datos:

System.Int32

Propiedad de:

ConnectionFactory

Determina si la selección de mensajes la realiza el XMS cliente o el broker.

Los valores válidos de la propiedad son los siguientes:

Valor válido	Significado
XMSC_WMQ_MSEL_CLIENT	La selección de mensajes ha sido realizada por el cliente de XMS.
XMSC_WMQ_MSEL_BROKER	La selección de mensajes ha sido realizada por el intermediario.

El valor predeterminado es XMSC_WMQ_MSEL_CLIENT.

Esta propiedad solo es relevante en el dominio de publicación/suscripción. La selección de mensajes del intermediario no está soportada si la propiedad [XMSC_WMQ_BROKER_VERSION](#) está establecida en XMSC_WMQ_BROKER_V1.

XMSC_WMQ_MSG_BATCH_SIZE

Tipo de datos:

System.Int32

Propiedad de:

ConnectionFactory

El número máximo de mensajes que se va a recuperar de una cola en un lote cuando se utiliza la entrega de mensajes asíncrona.

Cuando una aplicación está utilizando la entrega de mensajes asíncrona, bajo determinadas condiciones, el cliente XMS recupera un lote de mensajes de una cola antes de enviar cada mensaje individualmente a la aplicación. Esta propiedad especifica el número máximo de mensajes que puede haber en un lote.

El valor de la propiedad es un entero positivo y el valor predeterminado es 10. Considere definir la propiedad en un valor diferente solo si tiene un problema de rendimiento específico que debe abordar.

Si una aplicación está conectada a un gestor de colas sobre una red, aumentar el valor de esta propiedad puede reducir las sobrecargas de la red y los tiempos de respuesta, pero puede aumentar la cantidad de memoria necesaria para almacenar mensajes en el sistema cliente. En cambio, si se reduce el valor de esta propiedad se podría aumentar las sobrecargas de la red y los tiempos de respuesta, pero se podría reducir la cantidad de memoria necesaria para almacenar los mensajes.

XMSC_WMQ_POLLING_INTERVAL

Tipo de datos:

System.Int32

Propiedad de:

ConnectionFactory

Si cada escucha de mensajes dentro de una sesión no tiene ningún mensaje adecuado en su cola, este valor es el intervalo máximo, en milisegundos, que transcurre antes de que cada escucha de mensajes intente de nuevo obtener un mensaje de su cola.

Si con frecuencia sucede esto de que no haya disponible ningún mensaje adecuado para ninguno de los escuchas de mensajes de una sesión, considere aumentar el valor de esta propiedad.

El valor de la propiedad es un entero positivo. El valor predeterminado es 5000.

XMSC_WMQ_PORT

Tipo de datos:

System.Int32

Propiedad de:

ConnectionFactory

Objetos aplicables:

Nombre largo de la herramienta de administración JMS: PORT

Nombre abreviado de la herramienta de administración JMS: PORT

El número del puerto en el cual un gestor de colas escucha solicitudes entrantes.

Esta propiedad solo se utiliza cuando una aplicación se conecta a un gestor de colas en modalidad cliente. La propiedad se utiliza con la propiedad XMSC_WMQ_HOST_NAME para identificar el gestor de colas.

El valor predeterminado de la propiedad es XMSC_WMQ_DEFAULT_CLIENT_PORT, o 1414.

XMSC_WMQ_PROVIDER_VERSION

Tipo de datos:

Serie

Propiedad de:

ConnectionFactory

La versión, release, nivel de modificación y fixpack del gestor de colas al que tiene intención conectarse la aplicación. Los valores válidos para esta propiedad son:

- No especificado

O una serie en uno de los formatos siguientes

- V.R.M.F
- V.R.M
- V.R
- V

donde V, R, M y F son valores enteros mayores que o igual a cero.

De forma predeterminada, esta propiedad está establecida en "sin especificar".

La versión del cliente IBM MQ también desempeña un papel principal con respecto a si una aplicación cliente XMS puede utilizar características específicas de IBM MQ.

Nota: Una propiedad de sistema XMSC_WMQ_OVERRIDEPROVIDERVERSION sustituye la propiedad XMSC_WMQ_PROVIDER_VERSION. Esta propiedad se puede utilizar si no puede cambiar el valor de la fábrica de conexiones.

XMSC_WMQ_PUB_ACK_INTERVAL

Tipo de datos:

System.Int32

Propiedad de:

ConnectionFactory

El número de mensajes publicados por un editor antes de que el XMS cliente solicite un acuse de recibo al bróker.

Si reduce el valor de esta propiedad, el cliente solicita acuses de recibo con más frecuencia y, por lo tanto, el rendimiento del publicador disminuye. Si se aumenta el valor, el cliente tarda más tiempo en emitir una excepción si el intermediario no se ejecuta correctamente.

El valor de la propiedad es un entero positivo. El valor predeterminado es 25.

XMSC_WMQ_QMGR_CCSID

Tipo de datos:

System.Int32

Propiedad de:

ConnectionFactory

El identificador (CCSID) del conjunto de caracteres codificados, o página de códigos, en el que se intercambian los campos de datos de caracteres definidos en la interfaz de cola de mensajes (MQI) entre el XMS cliente y el IBM MQ cliente. Esta propiedad no se aplica a las series de datos de caracteres en los cuerpos de mensajes.

Cuando la aplicación XMS se conecta a un gestor de colas en la modalidad de cliente, el cliente de XMS se enlaza al cliente de IBM MQ. La información intercambiada entre los dos clientes contiene campos de datos de caracteres que están definidos en la MQI. Bajo circunstancias normales, el cliente de IBM MQ da por supuesto que estos campos están en la página de códigos del sistema en el cual se están ejecutando los clientes. Si el cliente de XMS proporciona y espera recibir estos campos en una página de códigos distinta, debe establecer esta propiedad para informar de ello al cliente de IBM MQ.

Cuando el cliente de IBM MQ envía estos campos de datos de caracteres al gestor de colas, los datos incluidos se deben convertir, si es necesario, a la página de códigos utilizada por el gestor de colas. De forma similar, cuando el cliente de IBM MQ recibe estos campos del gestor de colas, los datos aquí incluidos se deben convertir, si es necesario, a la página de códigos en la cual el cliente de XMS espera recibir los datos. El cliente de IBM MQ utiliza esta propiedad para realizar estas conversiones de datos.

De forma predeterminada, la propiedad no está establecida.

Establecer esta propiedad es equivalente a establecer la variable de entorno MQCCSID para un cliente IBM MQ que da soporte a aplicaciones cliente IBM MQ nativas. Para obtener más información sobre esta variable de entorno, consulte [MQCCSID](#).

XMSC_WMQ_QUEUE_MANAGER

Tipo de datos:

Serie

Propiedad de:

ConnectionFactory

Objetos aplicables:

Nombre largo de la herramienta de administración JMS: QMANAGER

Nombre abreviado de la herramienta de administración JMS: QMGR

El nombre del gestor de colas al que conectarse.

De forma predeterminada, la propiedad no está establecida.

XMSC_WMQ_RECEIVE_CCSID

La propiedad de destino que define el CCSID de destino para la conversión de mensajes del gestor de colas. El valor se ignora, a menos que XMSC_WMQ_RECEIVE_CONVERSION se establezca en WMQ_RECEIVE_CONVERSION_QMGR.

Tipo de datos:

Entero

Valor:

Cualquier entero positivo.

El valor predeterminado es 1208.

La especificación de un valor GMO_CONVERT en un mensaje es opcional. Si se especifica un valor GMO_CONVERT, la conversión se realiza de acuerdo con el valor especificado.

XMSC_WMQ_RECEIVE_CONVERSION

La propiedad de destino que determina si el gestor de colas va a realizar la conversión de datos.

Tipo de datos:

Entero

Valores:

XMSC_WMQ_RECEIVE_CONVERSION_CLIENT_MSG (DEFAULT): Realizar conversión de datos sólo en el cliente XMS . La conversión siempre se realiza utilizando la página de códigos 1208.

XMSC_WMQ_RECEIVE_CONVERSION_QMGR: Realizar la conversión de datos en el gestor de colas antes de enviar un mensaje al cliente XMS .

XMSC_WMQ_RECEIVE_EXIT

Tipo de datos:

Serie

Propiedad de:

ConnectionFactory

Identifica una salida de recepción de canal que se va a ejecutar.

El valor de la propiedad es una serie que identifica una salida de recepción de canal y tiene el formato siguiente:

libraryName(entryPointName)

donde,

- **libraryName** es la vía de acceso completa del archivo .dll de salida gestionada
- **nombrepuntoentrada** es el nombre de clase calificado por el espacio de nombres

Por ejemplo: C:\MyReceiveExit.dll(MyReceiveExitNameSpace.MyReceiveExitClassName)

De forma predeterminada, la propiedad no está establecida.

Esta propiedad solo es relevante cuando una aplicación se conecta a un gestor de colas en modalidad de cliente gestionado. Asimismo, solo están soportadas las salidas gestionadas.

XMSC_WMQ_RECEIVE_EXIT_INIT

Tipo de datos:

Serie

Propiedad de:

ConnectionFactory

Los datos de usuario que se pasan a una salida de recepción de canal cuando se llama.

El valor de la propiedad es una serie. De forma predeterminada, la propiedad no está establecida.

Esta propiedad solo es relevante cuando una aplicación se conecta a un gestor de colas en modalidad de cliente gestionado y la propiedad [“XMSC_WMQ_RECEIVE_EXIT” en la página 2119](#) está establecida.

XMSC_WMQ_RESOLVED_QUEUE_MANAGER

Tipo de datos:

Serie

Propiedad de:

ConnectionFactory

Esta propiedad se utiliza para obtener el nombre del gestor de colas que está conectado.

Cuando se utiliza con una CCDT (tabla de definición de canal de cliente), este nombre podría ser diferente del nombre del gestor de colas especificado en la fábrica de conexiones.

XMSC_WMQ_RESOLVED_QUEUE_MANAGER_ID

Tipo de datos:

Serie

Propiedad de:

ConnectionFactory

Esta propiedad se llena con el ID del gestor de colas después de la conexión.

XMSC_WMQ_SECURITY_EXIT

Tipo de datos:

Serie

Propiedad de:

ConnectionFactory

Identifica una salida de seguridad de canal.

El valor de la propiedad es una serie que identifica una salida de seguridad de canal y tiene el formato siguiente:

libraryName(entryPointName)

donde,

- **libraryName** es la vía de acceso completa del archivo .dll de salida gestionada.
- **nombrepuntoentrada** es el nombre de clase calificado por el espacio de nombres

Por ejemplo, C:\MySecurityExit.dll(MySecurityExitNameSpace.MySecurityExitClassName)

La longitud máxima de la serie es 128 caracteres.

De forma predeterminada, la propiedad no está establecida.

Esta propiedad solo es relevante cuando una aplicación se conecta a un gestor de colas en modalidad de cliente gestionado. Asimismo, solo están soportadas las salidas gestionadas.

XMSC_WMQ_SECURITY_EXIT_INIT

Tipo de datos:

Serie

Propiedad de:

ConnectionFactory

Los datos de usuario que se pasan a una salida de seguridad de canal cuando se llama.

La longitud máxima de la serie de datos de usuario es 32 caracteres.

De forma predeterminada, la propiedad no está establecida.

Esta propiedad solo es relevante cuando una aplicación se conecta a un gestor de colas en modalidad de cliente gestionado y la propiedad [“XMSC_WMQ_SECURITY_EXIT”](#) en la página 2120 está establecida.

XMSC_WMQ_SEND_EXIT

Tipo de datos:

Serie

Propiedad de:

ConnectionFactory

Identifica una salida de emisión de canal.

El valor de la propiedad es una serie. Una salida de emisión de canal tiene el formato siguiente:

libraryName(entryPointName)

donde,

- **libraryName** es la vía de acceso completa del archivo .dll de salida gestionada.
- **nombrepuntoentrada** es el nombre de clase calificado por el espacio de nombres

Por ejemplo, C:\MySendExit.dll(MySendExitNameSpace.MySendExitClassName)

De forma predeterminada, la propiedad no está establecida.

Esta propiedad solo es relevante cuando una aplicación se conecta a un gestor de colas en modalidad de cliente gestionado. Asimismo, solo están soportadas las salidas gestionadas.

XMSC_WMQ_SEND_EXIT_INIT

Tipo de datos:

Serie

Propiedad de:

ConnectionFactory

Los datos de usuario que se pasan a las salidas de emisión de canal cuando se llaman.

El valor de la propiedad es una serie de uno o más elementos de datos de usuario separados por comas. De forma predeterminada, la propiedad no está establecida.

Las reglas para especificar datos de usuario que se pasan a una secuencia de salidas de emisión de canal son las mismas que las reglas para especificar datos de usuario que se pasan a una secuencia de salidas de recepción de canal. Por lo tanto, para las reglas consulte [“XMSC_WMQ_RECEIVE_EXIT_INIT”](#) en la [página 2119](#).

Esta propiedad solo es relevante cuando una aplicación se conecta a un gestor de colas en modalidad de cliente gestionado y la propiedad [“XMSC_WMQ_SEND_EXIT”](#) en la [página 2120](#) está establecida.

XMSC_WMQ_SEND_CHECK_COUNT

Tipo de datos:

System.Int32

Propiedad de:

ConnectionFactory

El número de llamadas de envío que se van a permitir entre las comprobaciones de errores de colocación asíncrona dentro de una sesión XMS única sin transacción.

De forma predeterminada, esta propiedad está establecida en 0.

XMSC_WMQ_SHARE_CONV_ALLOWED

Tipo de datos:

System.Int32

Propiedad de:

ConnectionFactory

Objetos aplicables:

Nombre largo de la herramienta de administración JMS: SHARECONVALLOWED

Nombre abreviado de la herramienta de administración JMS: SCALD

Si una conexión de cliente puede compartir su socket con otras conexiones de primer nivel XMS del mismo proceso al mismo gestor de colas, siempre que las definiciones de los canales coincidan. Esta propiedad se proporciona para permitir un aislamiento completo de conexiones en sockets separados si es necesario para el desarrollo de aplicaciones, el mantenimiento o por motivos operativos. Si se establece esta propiedad simplemente se indica a XMS para que el socket subyacente se comparta. No indica cuántas conexiones comparten un solo socket. El número de conexiones que comparten un socket se determina mediante el valor SHARECNV que se negocia entre el cliente de IBM MQ y el servidor IBM MQ.

Una aplicación puede establecer las constantes con nombre siguiente para establecer la propiedad:

- XMSC_WMQ_SHARE_CONV_ALLOWED_FALSE - Las conexiones no comparten un socket.
- XMSC_WMQ_SHARE_CONV_ALLOWED_TRUE - Las conexiones comparten un socket.

De forma predeterminada, la propiedad está establecida en XMSC_WMQ_SHARE_CONV_ALLOWED_ENABLED.

Esta propiedad solo es relevante cuando una aplicación se conecta a un gestor de colas en modalidad de cliente.

XMSC_WMQ_SSL_CERT_STORES

Tipo de datos:

Serie

Propiedad de:

ConnectionFactory

Las ubicaciones de los servidores que contienen las listas de revocaciones de certificados (CRL) que se van a utilizar en una conexión SSL a un gestor de colas.

El valor de la propiedad es una lista de uno o más URL separados por comas. Cada URL tiene el formato siguiente:

```
[user[/password]@]ldap://[serveraddress][:portnum][,...]
```

Este formato es compatible con, pero con ampliaciones, el formato MQJMS básico.

Es válido tener un vacío serveraddress. En este caso, XMS presupone que el valor es la serie "localhost".

Una lista de ejemplos es:

```
myuser/mypassword@ldap://server1.mycom.com:389
ldap://server1.mycom.com
ldap://
ldap://:389
```

Solo .NET para: Las conexiones gestionadas a IBM MQ (WMQ_CM_CLIENT) y las conexiones no gestionadas a IBM MQ (WMQ_CM_CLIENT_UNMANAGED) admiten conexiones TLS / SSL.

De forma predeterminada, la propiedad no está establecida.

Conceptos relacionados

[Soporte SSL y TLS para el cliente no gestionado de .NET](#)

[Soporte SSL y TLS para el cliente gestionado de .NET](#)

XMSC_WMQ_SSL_CIPHER_SPEC

Tipo de datos:

Serie

Propiedad de:

ConnectionFactory

El nombre de la CipherSpec que se va a utilizar en una conexión segura a un gestor de colas.

Las especificaciones de cifrado que puede utilizar con el soporte de IBM MQ TLS se enumeran en la tabla siguiente. Cuando solicite un certificado personal, especifique un tamaño de clave para el par de claves pública y privada. El tamaño de clave que se utiliza durante el reconocimiento SSL es el tamaño almacenado en el certificado, a menos que esté determinado por la CipherSpec, tal como está indicado en la tabla. De forma predeterminada, esta propiedad no está establecida.

Nombre de CipherSpec	Protocolo utilizado	Algoritmo hash	Algoritmo de cifrado	Bits de cifrado	FIPS ¹	Suite B de 128 bits	Suite B de 192 bits
TLS_RSA_WITH_AES_128_CBC_SHA	TLS 1.0	SHA-1	AES	128	Sí	No	No
TLS_RSA_WITH_AES_256_CBC_SHA ²	TLS 1.0	SHA-1	AES	256	Sí	No	No
TLS_RSA_WITH_DES_CBC_SHA	TLS 1.0	SHA-1	DES	56	No	No	No
TLS_RSA_WITH_3DES_EDE_CBC_SHA ⁴	TLS 1.0	SHA-1	3DES	168	Sí	No	No
TLS_RSA_WITH_AES_128_GCM_SHA256	TLS 1.2	SHA-256	AES	128	Sí	No	No
TLS_RSA_WITH_AES_256_GCM_SHA384	TLS 1.2	SHA-384	AES	256	Sí	No	No
TLS_RSA_WITH_AES_128_CBC_SHA256	TLS 1.2	SHA-256	AES	128	Sí	No	No
TLS_RSA_WITH_AES_256_CBC_SHA256	TLS 1.2	SHA-256	AES	256	Sí	No	No
ECDHE_ECDSA_RC4_128_SHA256	TLS 1.2	SHA-256	RC4	128	No	No	No
ECDHE_ECDSA_3DES_EDE_CBC_SHA256	TLS 1.2	SHA-256	3DES	168	Sí	No	No
ECDHE_RSA_RC4_128_SHA256	TLS 1.2	SHA-256	RC4	128	No	No	No
ECDHE_RSA_3DES_EDE_CBC_SHA256	TLS 1.2	SHA-256	3DES	168	Sí	No	No
ECDHE_ECDSA_AES_128_CBC_SHA256	TLS 1.2	SHA-256	AES	128	Sí	No	No
ECDHE_ECDSA_AES_256_CBC_SHA384	TLS 1.2	SHA-384	AES	256	Sí	No	No
ECDHE_RSA_AES_128_CBC_SHA256	TLS 1.2	SHA-256	AES	128	Sí	No	No
ECDHE_RSA_AES_256_CBC_SHA384	TLS 1.2	SHA-384	AES	256	Sí	No	No
ECDHE_ECDSA_AES_128_GCM_SHA256	TLS 1.2	SHA-256	AES	128	Sí	Sí	No
ECDHE_ECDSA_AES_256_GCM_SHA384	TLS 1.2	SHA-384	AES	256	Sí	No	Sí
ECDHE_RSA_AES_128_GCM_SHA256	TLS 1.2	SHA-256	AES	128	Sí	No	No
ECDHE_RSA_AES_256_GCM_SHA384	TLS 1.2	SHA-384	AES	256	Sí	No	No
TLS_RSA_WITH_NULL_SHA256	TLS 1.2	SHA-256	Ninguna	0	No	No	No
ECDHE_RSA_NULL_SHA256	TLS 1.2	SHA-256	Ninguna	0	No	No	No

Nombre de CipherSpec	Protocolo utilizado	Algoritmo hash	Algoritmo de cifrado	Bits de cifrado	FIPS ¹	Suite B de 128 bits	Suite B de 192 bits
ECDHE_ECDSA_NULL_SHA256	TLS 1.2	SHA-256	Ninguna	0	No	No	No
TLS_RSA_WITH_NULL_NULL	TLS 1.2	Ninguna	Ninguna	0	No	No	No
TLS_RSA_WITH_RC4_128_SHA256	TLS 1.2	SHA-256	RC4	128	No	No	No

Notas:

1. Especifica si el sistema de control de la velocidad del vehículo (CipherSpec) cumple con Federal Information Processing Standards (FIPS) 140-2. Para obtener una explicación FIPS e información sobre cómo configurar IBM MQ para un funcionamiento FIPS 140-2 conforme, consulte [Federal Information Processing Standards \(FIPS\)](#).
2. CipherSpec no se puede utilizar para proteger una conexión desde a IBM MQ Explorer un gestor de colas, a menos que se apliquen los archivos de políticas sin restricciones adecuados al JRE utilizado por IBM MQ Explorer.
3. Este CipherSpec fue FIPS 140-2 certificado antes del 19 de mayo de 2007.
4. Cuando IBM MQ está configurado para un funcionamiento FIPS 140-2 compatible, este CipherSpec se puede utilizar para transferir hasta 32 GB de datos antes de que la conexión se termine con el error AMQ9288. Para evitar este error, evite utilizar el algoritmo DES triple (que está obsoleto) o habilite el restablecimiento de la clave secreta cuando utilice este CipherSpec en una FIPS 140-2 configuración.

Conceptos relacionados

[Integridad de datos de mensajes](#)

Tareas relacionadas

[Seguridad](#)

[Especificación de CipherSpecs](#)

XMSC_WMQ_SSL_CIPHER_SUITE

Tipo de datos:

Serie

Propiedad de:

ConnectionFactory

El nombre de la CipherSuite que se va a utilizar en una conexión TLS a un gestor de colas. El protocolo utilizado en la negociación de la conexión segura depende de la CipherSuite especificada.

Esta propiedad tiene los valores canónicos siguientes:

- SSL_RSA_WITH_DES_CBC_SHA
- SSL_RSA_EXPORT1024_WITH_DES_CBC_SHA
- SSL_RSA_EXPORT1024_WITH_RC4_56_SHA
- SSL_RSA_EXPORT_WITH_RC4_40_MD5
- SSL_RSA_WITH_RC4_128_MD5
- SSL_RSA_WITH_RC4_128_SHA
- SSL_RSA_WITH_3DES_EDE_CBC_SHA
- SSL_RSA_WITH_AES_128_CBC_SHA
- SSL_RSA_WITH_AES_256_CBC_SHA
- SSL_RSA_WITH_DES_CBC_SHA

- `SSL_RSA_WITH_3DES_EDE_CBC_SHA`

Este valor se puede proporcionar como alternativa a `XMSC_WMQ_SSL_CIPHER_SPEC`.

Si se ha especificado un valor no vacío para `XMSC_WMQ_SSL_CIPHER_SPEC`, este valor sustituye el valor para `XMSC_WMQ_SSL_CIPHER_SUITE`. Si `XMSC_WMQ_SSL_CIPHER_SPEC` no tiene un valor, se utiliza el valor de `XMSC_WMQ_SSL_CIPHER_SUITE` como conjunto de cifrado que se proporcionará a IBM Global Security Kit (GSKit). En este caso, el valor se asigna al valor equivalente CipherSpec, tal y como se describe en [CipherSuite y CipherSpec asignaciones de nombres para XMS conexiones a un IBM MQ gestor de colas](#).

Si tanto `XMSC_WMQ_SSL_CIPHER_SPEC` como `XMSC_WMQ_SSL_CIPHER_SUITE` están vacíos, el campo `pChDef->SSLCipherSpec` se llena con espacios.

Solo .NET para: Las conexiones gestionadas a IBM MQ (`WMQ_CM_CLIENT`) y las conexiones no gestionadas a IBM MQ (`WMQ_CM_CLIENT_UNMANAGED`) admiten conexiones TLS / SSL.

De forma predeterminada, la propiedad no está establecida.

Conceptos relacionados

[Soporte SSL y TLS para el cliente no gestionado de .NET](#)

[Soporte SSL y TLS para el cliente gestionado de .NET](#)

XMSC_WMQ_SSL_CRYPTO_HW

Tipo de datos:

Serie

Propiedad de:

`ConnectionFactory`

Detalles de configuración para el hardware de cifrado conectado al sistema cliente.

Esta propiedad tiene los valores canónicos siguientes:

- `GSK_ACCELERATOR_RAINBOW_CS_OFF`
- `GSK_ACCELERATOR_RAINBOW_CS_ON`
- `GSK_ACCELERATOR_NCIPHER_NF_OFF`
- `GSK_ACCELERATOR_NCIPHER_NF_ON`

Existe un formato especial para el hardware de cifrado PKCS11 (donde `DriverPath`, `TokenLabel` y `TokenPassword` son series especificadas por usuario):

```
GSK_PKCS11=PKCS#11 DriverPath; PKCS#11 TokenLabel;PKCS#11 TokenPassword
```

XMS no interpreta ni altera el contenido de la serie. Copia el valor proporcionado, hasta un límite de 256 caracteres de byte único, en el campo `MQSCO.CryptoHardware`.

Solo .NET para: Las conexiones gestionadas a IBM MQ (`WMQ_CM_CLIENT`) y las conexiones no gestionadas a IBM MQ (`WMQ_CM_CLIENT_UNMANAGED`) admiten conexiones TLS / SSL.

De forma predeterminada, la propiedad no está establecida.

Conceptos relacionados

[Soporte SSL y TLS para el cliente no gestionado de .NET](#)

[Soporte SSL y TLS para el cliente gestionado de .NET](#)

XMSC_WMQ_SSL_FIPS_REQUIRED

Tipo de datos:

Boolean

Propiedad de:

`ConnectionFactory`

El valor de esta propiedad determina si una aplicación puede o no utilizar conjuntos de cifrado no FIPS conformes. Si esta propiedad se establece en «true», solo FIPS se utilizan algoritmos para la conexión cliente-servidor.

Esta propiedad puede tener los valores siguientes, que se convierten a los dos valores canónicos para MQSCO.FipsRequired:

<i>Tabla 916. Tabla de valores para la propiedad MQSCO.FlipsRequired</i>		
Valor	Descripción	Valor correspondiente de MQSCO.FipsRequired
falso	Se puede utilizar cualquier CipherSpec.	MQSSL_FIPS_NO (el valor predeterminado)
true	Solo se pueden utilizar FIPS algoritmos criptográficos certificados en el CipherSpec que se aplica a esta conexión de cliente.	MQSSL_FIPS_YES

XMS copia el valor relevante en MQSCO.FipsRequired antes de llamar a MQCONN.

Solo .NET para: Las conexiones gestionadas a IBM MQ (WMQ_CM_CLIENT) y las conexiones no gestionadas a IBM MQ (WMQ_CM_CLIENT_UNMANAGED) admiten conexiones TLS / SSL.

Conceptos relacionados

[Soporte SSL y TLS para el cliente no gestionado de .NET](#)

[Soporte SSL y TLS para el cliente gestionado de .NET](#)

XMSC_WMQ_SSL_KEY_REPOSITORY

Tipo de datos:

Serie

Propiedad de:

ConnectionFactory

La ubicación de un archivo de base de datos de claves en el cual se almacenan claves y certificados.

XMS copia la serie, hasta un límite de 256 caracteres de byte único, en el campo MQSCO.KeyRepository. IBM MQ interpreta esta serie como un nombre de archivo, incluida la vía de acceso completa.

Solo .NET para: Las conexiones gestionadas a IBM MQ (WMQ_CM_CLIENT) y las conexiones no gestionadas a IBM MQ (WMQ_CM_CLIENT_UNMANAGED) admiten conexiones TLS / SSL.

De forma predeterminada, la propiedad no está establecida.

V 10.0.0 Desde IBM MQ 10.0.0, puede configurar rutas personalizadas para los certificados de TLS con el fin de gestionar de forma flexible el almacenamiento de certificados y especificar rutas fuera del almacén de claves del sistema para XMS.NET las aplicaciones. Para ello, puede utilizar Dotnet Password Encryptor (**rundotnetcred**), que admite el cifrado seguro de contraseñas y la gestión de datos de contraseñas cifradas. Para obtener más información, consulte [Ruta personalizada para certificados en XMS .NET para la conectividad TLS mediante contraseñas de texto sin formato o cifradas](#).

Conceptos relacionados

[Soporte SSL y TLS para el cliente no gestionado de .NET](#)

[Soporte SSL y TLS para el cliente gestionado de .NET](#)

XMSC_WMQ_SSL_KEY_RESETCOUNT

Tipo de datos:

System.Int32

Propiedad de:

ConnectionFactory

El KeyResetCount representa el número total de bytes sin cifrado enviados y recibidos en una conversación SSL antes de renegociar la clave secreta. El número de bytes incluye la información de control que envía el MCA.

XMS copia el valor que proporcione para esta propiedad en MQSCO.KeyResetCount antes de llamar a MQCONN.

El parámetro MQSCO.KeyRestCount solo está disponible desde IBM MQ versión 6. Si IBM MQ versión 5.3, si esta propiedad está establecida, XMS no intenta establecer la conexión con el gestor de colas y lanza una excepción adecuada en su lugar.

Solo .NET para: Las conexiones gestionadas a IBM MQ (WMQ_CM_CLIENT) y las conexiones no gestionadas a IBM MQ (WMQ_CM_CLIENT_UNMANAGED) admiten conexiones TLS / SSL.

El valor predeterminado de esta propiedad es cero, lo que significa que las claves secretas nunca se renegocian.

Conceptos relacionados

[Soporte SSL y TLS para el cliente no gestionado de .NET](#)

[Soporte SSL y TLS para el cliente gestionado de .NET](#)

XMSC_WMQ_SSL_PEER_NAME

Tipo de datos:

Serie

Propiedad de:

ConnectionFactory

El nombre de igual que se va a utilizar en una conexión SSL a un gestor de colas.

No hay ninguna lista de valores canónicos para esta propiedad. En su lugar, debe crear esta cadena de acuerdo con las reglas para SSLPEER.

Un ejemplo de un nombre de igual es:

```
"CN=John Smith, O=IBM ,OU=Test , C=GB"
```

XMS copia la serie en la página de códigos de byte único y coloca los valores correctos en MQCD.SSLPeerNamePtr y MQCD.SSLPeerNameLength antes de llamar a MQCONN.

Esta propiedad solo es relevante si la aplicación se conecta a un gestor de colas en modalidad de cliente.

Solo .NET para: Las conexiones gestionadas a IBM MQ (WMQ_CM_CLIENT) y las conexiones no gestionadas a IBM MQ (WMQ_CM_CLIENT_UNMANAGED) admiten conexiones TLS / SSL.

De forma predeterminada, la propiedad no está establecida.

Conceptos relacionados

[Soporte SSL y TLS para el cliente no gestionado de .NET](#)

[Soporte SSL y TLS para el cliente gestionado de .NET](#)

Referencia relacionada

[SSLPEERNAME](#)

XMSC_WMQ_SSL_KEY_REPOSITORY_PASSWORD

Tipo de datos:

Serie

Propiedad de:

Certificado TLS

Propiedad que permite a los clientes establecer las contraseñas necesarias para abrir el certificado TLS antes del establecimiento de la conexión.

Los siguientes ejemplos muestran el uso de la propiedad de contraseña del repositorio de claves:

```
 properties.Add(XMSC.WMQ_SSL_KEY_REPOSITORY_PASSWORD, "123456")
```

```
 properties.Add(XMSC.WMQ_SSL_KEY_REPOSITORY_PASSWORD, "123456")
```

Esta propiedad solo es relevante si la aplicación se conecta a un gestor de colas en modalidad de cliente.

De forma predeterminada, la propiedad no está establecida.

XMSC_WMQ_SYNCPOINT_ALL_GETS

Tipo de datos:

System.Boolean

Propiedad de:

ConnectionFactory

Indica si se deben recuperar todos los mensajes de colas dentro del control de punto de sincronización.

Los valores válidos de la propiedad son los siguientes:

Valor válido	Significado
false	Cuando las circunstancias son apropiadas, el cliente de XMS puede recuperar mensajes de colas fuera del control de punto de sincronización.
true	El cliente de XMS debe recuperar todos los mensajes de las colas dentro del control de punto de sincronización.

El valor predeterminado es false.

XMSC_WMQ_TARGET_CLIENT

Tipo de datos:

System.Int32

Propiedad de:

Destino

Nombre utilizado en un URI:

targetClient

Indica si los mensajes enviados al destino contienen una cabecera MQRFH2.

Si una aplicación envía un mensaje que contiene una cabecera MQRFH2, la aplicación receptora debe poder manejar la cabecera.

Los valores válidos de la propiedad son los siguientes:

Valor válido	Significado
XMSC_WMQ_TARGET_DEST_JMS	Los mensajes enviados al destino contienen una cabecera MQRFH2. Especifique este valor si la aplicación está enviando los mensajes a otra aplicación XMS , una aplicación IBM MQ classes for JMS o una aplicación IBM MQ nativa que está diseñada para manejar una cabecera MQRFH2 .
XMSC_WMQ_TARGET_DEST_MQ	Los mensajes enviados al destino no contienen una cabecera MQRFH2. Especifique este valor si la aplicación está enviando los mensajes a una aplicación IBM MQ nativa que no se ha diseñado para manejar una cabecera MQRFH2.

El valor predeterminado es XMSC_WMQ_TARGET_DEST_JMS.

XMSC_WMQ_TEMP_Q_PREFIX

Tipo de datos:

Serie

Propiedad de:

ConnectionFactory

El prefijo utilizado para formar el nombre de la IBM MQ cola dinámica que se crea cuando la aplicación genera una XMS cola temporal.

Las reglas para formar el prefijo son las mismas que las reglas para formar el contenido del **DynamicQName** campo en un descriptor de objeto, pero el último carácter no en blanco debe ser un asterisco (*). De forma predeterminada, la propiedad no está establecida. Si no está configurado, se utiliza el siguiente valor:

-  AMQ . * en múltiples plataformas
-  CSQ . * en z/OS

Esta propiedad solo es relevante en el dominio punto a punto.

XMSC_WMQ_TEMP_TOPIC_PREFIX

Tipo de datos:

Serie

Propiedad de:

ConnectionFactory, Destination

TEMP/TEMPTOPICPREFIX/unique_idAl crear temas temporales, XMS se genera una cadena de tema con el formato «», o, si esta propiedad contiene el valor predeterminado, se genera esta cadena: «TEMP/unique_id». Si se especifica un valor no vacío se permite que se definan colas de modelo específicas para crear las colas gestionadas para suscriptores de temas temporales creados en esta conexión.

Cualquier serie no nula que conste sólo de caracteres válidos para una serie de tema IBM MQ es un valor válido para esta propiedad.

De forma predeterminada, esta propiedad está establecida en "" (serie vacía).

Nota: Esta propiedad solo es relevante en el dominio de publicación/suscripción.

XMSC_WMQ_TEMPORARY_MODEL

Tipo de datos:

Serie

Propiedad de:

ConnectionFactory

El nombre de la IBM MQ cola modelo a partir de la cual se crea una cola dinámica cuando la aplicación crea una XMS cola temporal.

El valor predeterminado de la propiedad es SYSTEM.DEFAULT.MODEL.QUEUE.

Esta propiedad sólo es relevante en el dominio punto a punto.

XMSC_WMQ_WILDCARD_FORMAT

Tipo de datos:

System.Int32

Propiedad de:

ConnectionFactory, Destination

Esta propiedad determina qué versión de sintaxis de comodín se va a utilizar.

Cuando se utiliza la publicación/suscripción con IBM MQ '*' y '?' se tratan como comodines. Considerando que '#' y '+' se tratan como comodines cuando se utiliza la publicación/suscripción con IBM Integration Bus. Esta propiedad sustituye la propiedad XMSC_WMQ_BROKER_VERSION.

Los valores válidos para esta propiedad son:

XMSC_WMQ_WILDCARD_TOPIC_ONLY

Reconoce sólo los comodines de nivel de tema, es decir, '#' y '+' se tratan como comodines. Este valor es el mismo que XMSC_WMQ_BROKER_V2.

XMSC_WMQ_WILDCARD_CHAR_ONLY

Reconoce sólo los caracteres comodín, es decir, "*" y '?' se tratan como comodines. Este valor es el mismo que XMSC_WMQ_BROKER_V1.

De forma predeterminada, esta propiedad se establece en XMSC_WMQ_WILDCARD_TOPIC_ONLY.

XMSC_WMQ_CLAVE_DE_CIFRADO

Tipo de datos:

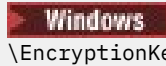
Serie

Propiedad de:

Certificado TLS

Propiedad que permite a los clientes especificar la ubicación donde se guarda el archivo de la clave de cifrado.

Los siguientes ejemplos muestran el uso de la propiedad de clave de cifrado:

```
 properties.Add(XMSC.WMQ_ENCRYPTION_KEY, "C:\\CustomPath_Certificates\\EncryptionKey\\mykey.txt")
```

```
 properties.Add(XMSC.WMQ_ENCRYPTION_KEY, "C:\\CustomPath_Certificates\\EncryptionKey\\mykey.txt")
```

Esta propiedad solo es relevante si la aplicación se conecta a un gestor de colas en modalidad de cliente.

De forma predeterminada, la propiedad no está establecida.

Nota: Si la propiedad de la clave de cifrado no está configurada en la .NET aplicación cliente, las bibliotecas comprueban si el valor establecido en la propiedad de contraseña está en texto sin formato o en un formato codificado c Base64.

- Si el valor está en texto sin formato, las bibliotecas lo tratan como una contraseña de texto sin formato.
- Si el valor está codificado en Base64, se utiliza la clave predeterminada para descifrarlo y establecer la conexión TLS.

Tras la confirmación, la aplicación continúa con la conexión TLS.

XMSC_WPM_BUS_NAME

Tipo de datos:

Serie

Propiedad de:

ConnectionFactory y Destination

Nombre utilizado en un URI:

busName

Para una fábrica de conexiones, el nombre del bus de integración de servicios al que se conecta la aplicación o, para un destino, el nombre del bus de integración de servicios en el cual existe el destino.

Para un destino que es un tema, esta propiedad es el nombre del bus de integración de servicios en el cual existe el espacio de tema asociado. Este espacio de tema se especifica mediante la propiedad XMSC_WPM_TOPIC_SPACE.

Si la propiedad no está establecida para un destino, se da por supuesto que la cola o el espacio de tema asociado existe en el bus de integración de servicios al cual se conecta la aplicación.

De forma predeterminada, la propiedad no está establecida.

XMSC_WPM_CONNECTION_PROTOCOL

Tipo de datos:

System.Int32

Propiedad de:

Conexión

El protocolo de comunicaciones utilizado para la conexión al motor de mensajería. Esta propiedad es de solo lectura.

Los valores posibles de la propiedad son solo siguientes:

Valor	Significado
XMSC_WPM_CP_HTTP	La conexión utiliza HTTP sobre TCP/IP.
XMSC_WPM_CP_TCP	La conexión utiliza TCP/IP.

XMSC_WPM_CONNECTION_PROXIMITY

Tipo de datos:

System.Int32

Propiedad de:

ConnectionFactory

El valor de proximidad de conexión para la conexión. Esta propiedad determina cómo de cerca debe estar el motor de mensajería al que se conecta la aplicación en el servidor de programa de arranque.

Los valores válidos de la propiedad son los siguientes:

Valor válido	Valor de proximidad de conexión
XMSC_WPM_CONNECTION_PROXIMITY_BUS	Bus
XMSC_WPM_CONNECTION_PROXIMITY_CLUSTER	Clúster
XMSC_WPM_CONNECTION_PROXIMITY_HOST	Host
XMSC_WPM_CONNECTION_PROXIMITY_SERVER	Servidor

El valor predeterminado es XMSC_WPM_CONNECTION_PROXIMITY_BUS.

XMSC_WPM_DUR_SUB_HOME

Tipo de datos:

Serie

Propiedad de:

ConnectionFactory

Nombre utilizado en un URI:

durableSubscriptionHome

El nombre del motor de mensajería donde se gestionan todas las suscripciones duraderas para una conexión o destino. Los mensajes que se van a entregar a los suscriptores duraderos se almacenan en el punto de publicación del mismo motor de mensajería.

Se debe especificar un inicio de suscripción duradera para una conexión antes de que una aplicación pueda crear un suscriptor duradero que utilice la conexión. Cualquier valor especificado para un destino altera temporalmente el valor especificado para la conexión.

De forma predeterminada, la propiedad no está establecida.

Esta propiedad solo es relevante en el dominio de publicación/suscripción.

XMSC_WPM_HOST_NAME

Tipo de datos:

Serie

Propiedad de:

Conexión

El nombre de host o la dirección IP del sistema que contiene el motor de mensajería al que se conecta la aplicación. Esta propiedad es de solo lectura.

XMSC_WPM_LOCAL_ADDRESS

Tipo de datos:

Serie

Propiedad de:

ConnectionFactory

Para una conexión a un bus de integración de servicios, esta propiedad especifica la interfaz de red local que se va a utilizar, o el puerto local o el rango de puertos locales que se va a utilizar, o ambos.

El valor de la propiedad es una serie con el formato siguiente:

[nombre_host][(puerto_bajo)[,puerto_alto]]

Los significados de las variables son los siguientes:

nombre_host

El nombre de host o la dirección IP de la interfaz de red local que se va a utilizar para la conexión.

Proporcionar esta información solo es necesario si el sistema en el cual se está ejecutando la aplicación tiene dos o más interfaces de red y tendrá que poder especificar qué interfaz se debe utilizar para la conexión. Si el sistema tiene solo una interfaz de red, solo se puede utilizar dicha interfaz. Si el sistema tiene dos o más interfaces de red y no especifica qué interfaz se debe utilizar, la interfaz se selecciona de forma aleatoria.

puerto_bajo

El número del puerto local que se va a utilizar para la conexión.

Si *puerto_alto* también se especifica, *puerto_bajo* se interpreta como el número de puerto inferior en un rango de números de puerto.

puerto_alto

El número de puerto superior en un rango de números de puerto. Se debe utilizar uno de los puertos del rango especificado para la conexión.

Aquí aparecen algunos ejemplos de valores válidos de la propiedad:

JUPITER
9.20.4.98
JUPITER(1000)
9.20.4.98(1000,2000)
(1000)
(1000,2000)

De forma predeterminada, la propiedad no está establecida.

XMSC_WPM_ME_NAME

Tipo de datos:

Serie

Propiedad de:

Conexión

El nombre del motor de mensajería al que se conecta la aplicación. Esta propiedad es de solo lectura.

XMSC_WPM_NON_PERSISTENT_MAP**Tipo de datos:**

System.Int32

Propiedad de:

ConnectionFactory

El nivel de fiabilidad de los mensajes no persistente que se envían utilizando la conexión.

Los valores válidos de la propiedad son los siguientes:

Valor válido

XMSC_WPM_MAPPING_AS_DESTINATION

XMSC_WPM_MAPPING_BEST_EFFORT_NON_PERSISTENT

XMSC_WPM_MAPPING_EXPRESS_NON_PERSISTENT

XMSC_WPM_MAPPING_RELIABLE_NON_PERSISTENT

XMSC_WPM_MAPPING_RELIABLE_PERSISTENT

XMSC_WPM_MAPPING_ASSURED_PERSISTENT

Nivel de fiabilidad

Se determina mediante el nivel de fiabilidad predeterminado especificado para el espacio de cola o tema en el bus de integración de servicios.

Mejor esfuerzo no persistente

Express no persistente

Fiable no persistente

Fiable persistente

Seguro persistente

El valor predeterminado es XMSC_WPM_MAPPING_EXPRESS_NON_PERSISTENT.

XMSC_WPM_PERSISTENT_MAP**Tipo de datos:**

System.Int32

Propiedad de:

ConnectionFactory

El nivel de fiabilidad de los mensajes persistentes que se envían mediante la conexión.

Los valores válidos de la propiedad son los siguientes:

Valor válido

XMSC_WPM_MAPPING_AS_DESTINATION

XMSC_WPM_MAPPING_BEST_EFFORT_NON_PERSISTENT

Nivel de fiabilidad

Se determina mediante el nivel de fiabilidad predeterminado especificado para el espacio de cola o tema en el bus de integración de servicios.

Mejor esfuerzo no persistente

Valor válido**Nivel de fiabilidad**

XMSC_WPM_MAPPING_EXPRESS_NON_PERSISTENT

Express no persistente

XMSC_WPM_MAPPING_RELIABLE_NON_PERSISTENT

Fiable no persistente

XMSC_WPM_MAPPING_RELIABLE_PERSISTENT

Fiable persistente

XMSC_WPM_MAPPING_ASSURED_PERSISTENT

Seguro persistente

El valor predeterminado es XMSC_WPM_MAPPING_RELIABLE_PERSISTENT.

XMSC_WPM_PORT**Tipo de datos:**

System.Int32

Propiedad de:

Conexión

El número del puerto en el que estaba a la escucha el motor de mensajería al que está conectada la aplicación. Esta propiedad es de solo lectura.

XMSC_WPM_PROVIDER_ENDPOINTS**Tipo de datos:**

Serie

Propiedad de:

ConnectionFactory

Una secuencia de una o más direcciones de punto final de servidores de programa de arranque. Las direcciones de punto final están separadas con comas.

Un servidor de programa de arranque es un servidor de aplicaciones que es responsable de seleccionar el motor de mensajería al que se conecta la aplicación. La dirección de punto final de un servidor de programa de arranque tiene el formato siguiente:

nombre_host:número_puerto:nombre_cadena

Los significados de los componentes de una dirección de punto final son los siguientes:

nombre_host

El nombre de host o la dirección IP del sistema en el que reside el servidor de programa de arranque. Si no se especifica ningún nombre de host ni ninguna dirección IP, el valor predeterminado es localhost.

número_puerto

El número de puerto en el cual escucha el servidor de programa de arranque solicitudes entrantes. Si no se especifica ningún número de puerto, el valor predeterminado es 7276.

nombre_cadena

El nombre de una cadena de transporte del programa de arranque utilizada por el servidor de programa de arranque. Los valores válidos son los siguientes:

Valor válido**Nombre de la cadena de transporte del programa de arranque**

XMSC_WPM_BOOTSTRAP_HTTP

BootstrapTunneledMessaging

XMSC_WPM_BOOTSTRAP_HTTPS

BootstrapTunneledSecureMessaging

XMSC_WPM_BOOTSTRAP_SSL

BootstrapSecureMessaging

Valor válido	Nombre de la cadena de transporte del programa de arranque
XMSC_WPM_BOOTSTRAP_TCP	BootstrapBasicMessaging

Si no se especifica ningún nombre, el valor predeterminado esXMSC_WPM_BOOTSTRAP_TCP.

Si no se especifica ninguna dirección de punto final, el valor predeterminado es localhost:7276:BootstrapBasicMessaging.

XMSC_WPM_SSL_CIPHER_SUITE

Tipo de datos:

Serie

Propiedad de:

ConnectionFactory

El nombre del servidor de mensajes (CipherSuite) que se utilizará en una conexión de servidor de mensajes (TLS) con un WebSphere Application Server service integration bus motor de mensajería. El protocolo utilizado en la negociación de la conexión segura depende de la CipherSuite especificada.

Tabla 917. CipherSuite Opciones para la conexión a un motor WebSphere Application Server service integration bus de mensajería

Suite de cifrado	Protocolo utilizado
TLS_RSA_WITH_DES_CBC_SHA	TLSv1
TLS_RSA_WITH_3DES_EDE_CBC_SHA	TLSv1
TLS_RSA_WITH_AES_128_CBC_SHA	TLSv1
TLS_RSA_WITH_AES_256_CBC_SHA	TLSv1

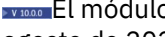
Notas:

-  TLS_RSA_WITH_AES_128_CBC_SHA y TLS_RSA_WITH_AES_256_CBC_SHA CipherSuites solo son compatibles Windows con. (Esto viene dictado por GSKit.)
-  TLS_RSA_WITH_3DES_EDE_CBC_SHA está en desuso. Sin embargo, se sigue pudiendo utilizar para transferir hasta 32 GB de datos antes de que se termine la conexión con el error AMQ9288. Para evitar este error, tendrá que evitar utilizar el triple DES, o habilitar el restablecimiento de clave secreta cuando se utiliza esta CipherSpec.

No hay ningún valor predeterminado para esta propiedad. Si desea utilizar SSL o TLS, debe especificar un valor para esta propiedad, de lo contrario, la aplicación no puede conectarse correctamente al servidor.

XMSC_WPM_SSL_FIPS_REQUIRED

Notas:

-  En AIX, Linux, y Windows, IBM MQ garantiza FIPS 140-3 el cumplimiento de la normativa a través del GSKit 9 módulo IBM Crypto for C (ICC) criptográfico. La certificación del NIST correspondiente al FIPS 140-3 módulo puede consultarse en <https://csrc.nist.gov/projects/cryptographic-module-validation-program/certificate/4755>.
-  FIPS Actualmente no hay soporte disponible para Linux s390x. Los clientes de esta plataforma que necesiten FIPS asistencia técnica deben seguir utilizando una versión anterior de IBM MQ. La compatibilidad con FIPS 140-3 se habilitará en una futura versión de FixPack una vez que el IBM Crypto for C (ICC) módulo criptográfico haya obtenido su certificación en esta plataforma.
-  El módulo FIPS 140-3 criptográfico integrado IBM Semeru Runtime fue aprobado por el NIST en agosto de 2024. IBM MQ 10.0.0 incorpora compatibilidad con la gestión de IBM MQ classes for JMS conexiones de IBM MQ classes for Java cliente mediante TLS para FIPS 140-3 en Java 8 y IBM Semeru

Runtime 11+. La certificación del NIST correspondiente al FIPS 140-3 módulo se puede consultar en <https://csrc.nist.gov/projects/cryptographic-module-validation-program/certificate/4755>.

- En IBM MQ el caso de Containers, las imágenes de IBM MQ Operator contenedor de 3.2.0 y del gestor de colas 9.4.0.0 en adelante se basan en UBI 9. FIPS 140-3 El cumplimiento de IBM MQ la normativa para productos envasados está actualmente pendiente.

• V 10.0.0 Si FIPS está habilitado, los procesos IBM MQ de control de contenedores utilizan un módulo FIPS 140-3 certificado de OpenSSL. Para obtener más información sobre la certificación del NIST, consulte: <https://access.redhat.com/compliance/fips>. IBM MQ Los gestores de colas que se ejecutan en imágenes de contenedor tienen el mismo nivel FIPS de certificación que la versión de la plataforma de IBM MQ la imagen base.

Tipo de datos:

Boolean

Propiedad de:

ConnectionFactory

El valor de esta propiedad determina si una aplicación puede o no utilizar conjuntos de cifrado noFIPS compatibles. Si esta propiedad se establece en verdadero, solo se utilizan FIPS algoritmos para la conexión cliente-servidor. Establecer el valor de esta propiedad en TRUE impide que la aplicación utilice conjuntos de cifrado noFIPS compatibles.

De forma predeterminada, la propiedad está establecida en FALSE (es decir, FIPS modo desactivado).

XMSC_WPM_SSL_KEY_REPOSITORY

Tipo de datos:

Serie

Propiedad de:

ConnectionFactory

Una vía de acceso del archivo que es el archivo de conjunto de claves que contiene las claves públicas o privadas que se van a utilizar en la conexión segura.

Establecer la propiedad del archivo de conjunto de claves en el valor especial de XMSC_WPM_SSL_MS_CERTIFICATE_STORE especifica el uso de la base de datos de claves Microsoft Windows. El uso de la base de datos de claves Microsoft Windows, que se encuentra en **Panel de control > Opciones de Internet > Contenido > Certificados**, elimina la necesidad de una base de datos de archivos de claves separados. El uso de esta constante en Windows x64 y otras plataformas no está permitido.

De forma predeterminada, la propiedad no está establecida.

XMSC_WPM_SSL_KEYRING_LABEL

Tipo de datos:

Serie

Propiedad de:

ConnectionFactory

El certificado que se va a utilizar al autenticarse con el servidor. Si no se especifica ningún valor, se utiliza el certificado predeterminado.

De forma predeterminada, la propiedad no está establecida.

XMSC_WPM_SSL_KEYRING_PW

Tipo de datos:

Serie

Propiedad de:

ConnectionFactory

La contraseña para el archivo de conjunto de claves.

Esta propiedad se puede utilizar como alternativa al uso de [XMSC_WPM_SSL_KEYRING_STASH_FILE](#) para configurar la contraseña del archivo de conjunto de claves.

De forma predeterminada, la propiedad no está establecida.

XMSC_WPM_SSL_KEYRING_STASH_FILE

Tipo de datos:

Serie

Propiedad de:

ConnectionFactory

El nombre de un archivo binario que contiene la contraseña del archivo del repositorio de claves.

Esta propiedad se puede utilizar como alternativa al uso de [XMSC_WPM_SSL_KEYRING_PW](#) para configurar la contraseña para el archivo de conjunto de claves.

De forma predeterminada, la propiedad no está establecida.

XMSC_WPM_TARGET_GROUP

Tipo de datos:

Serie

Propiedad de:

ConnectionFactory

El nombre de un grupo de destino de motores de mensajería. La naturaleza del grupo de destino viene determinada por la propiedad [XMSC_WPM_TARGET_TYPE](#).

Establezca esta propiedad si desea restringir la búsqueda de un motor de mensajería a un subgrupo de los motores de mensajería en el bus de integración de servicios. Si desea que la aplicación pueda conectarse a cualquier motor de mensajería del bus de integración de servicios, no establezca esta propiedad.

De forma predeterminada, la propiedad no está establecida.

XMSC_WPM_TARGET_SIGNIFICANCE

Tipo de datos:

System.Int32

Propiedad de:

ConnectionFactory

La importancia del grupo de destinos de motores de mensajería.

Los valores válidos de la propiedad son los siguientes:

Valor válido	Significado
XMSC_WPM_TARGET_SIGNIFICANCE_PREFERRED	Se selecciona un motor de mensajería en el grupo de destinos, si hay uno disponible. De lo contrario, se selecciona un motor de mensajería fuera del grupo de destinos, siempre que esté en el mismo bus de integración de servicios.
XMSC_WPM_TARGET_SIGNIFICANCE_OBLIGATORIO	El motor de mensajería seleccionado debe estar en el grupo de destinos. Si un motor de mensajería del grupo de destino no está disponible, el proceso de conexión falla.

El valor predeterminado de la propiedad es XMSC_WPM_TARGET_SIGNIFICANCE_PREFERRED.

XMSC_WPM_TARGET_TRANSPORT_CHAIN

Tipo de datos:

Serie

Propiedad de:

ConnectionFactory

El nombre de la cadena de transporte de entrada que debe utilizar la aplicación para conectarse a un motor de mensajería.

El valor de la propiedad puede ser el nombre de cualquier cadena de transporte de entrada que está disponible en el servidor de aplicaciones que aloja el motor de mensajería. La constante con nombre siguiente se proporciona para una de las cadenas de transporte de entrada predefinidas:

Constante con nombre	Nombre de la cadena de transporte
XMSC_WPM_TARGET_TRANSPORT_CHAIN_BASIC	InboundBasicMessaging

El valor predeterminado de la propiedad es XMSC_WPM_TARGET_TRANSPORT_CHAIN_BASIC.

XMSC_WPM_TARGET_TYPE

Tipo de datos:

System.Int32

Propiedad de:

ConnectionFactory

El tipo del grupo de destinos de motores de mensajería. Esta propiedad determina la naturaleza del grupo de destino identificado por la propiedad [XMSC_WPM_TARGET_GROUP](#).

Los valores válidos de la propiedad son los siguientes:

Valor válido	Significado
XMSC_WPM_TARGET_TYPE_BUSMEMBER	El nombre del grupo de destinos es el nombre de un miembro de bus. El grupo de destinos está formado por todos los motores de mensajería del miembro de bus.
XMSC_WPM_TARGET_TYPE_CUSTOM	El nombre del grupo de destinos es el nombre de un grupo definido por usuario de motores de mensajería. El grupo de destinos es para todos los motores de mensajería que están registrados con el grupo definido por usuario.
XMSC_WPM_TARGET_TYPE_ME	El nombre del grupo de destinos es el nombre de un motor de mensajería. El grupo de destinos es el motor de mensajería especificado.

De forma predeterminada, la propiedad no está establecida.

XMSC_WPM_TEMP_Q_PREFIX

Tipo de datos:

Serie

Propiedad de:

ConnectionFactory

El prefijo utilizado para formar el nombre de la cola temporal que se crea en el bus de integración de servicios cuando la aplicación crea una XMS cola temporal. El prefijo puede contener hasta 12 caracteres.

El nombre de una cola temporal empieza con los caracteres "_Q" seguidos por el prefijo. El resto del nombre está formado por caracteres generados por el sistema.

De forma predeterminada, la propiedad no está establecida, lo que significa que el nombre de una cola temporal no tiene un prefijo.

Esta propiedad sólo es relevante en el dominio punto a punto.

XMSC_WPM_TEMP_TOPIC_PREFIX

Tipo de datos:

Serie

Propiedad de:

ConnectionFactory

El prefijo utilizado para formar el nombre de un tema temporal que crea la aplicación. El prefijo puede contener hasta 12 caracteres.

El nombre de un tema temporal empieza con los caracteres "_T" seguidos por el prefijo. El resto del nombre está formado por caracteres generados por el sistema.

De forma predeterminada, la propiedad no está establecida, lo que significa que el nombre de un tema temporal no tiene prefijo.

Esta propiedad solo es relevante en el dominio de publicación/suscripción.

XMSC_WPM_TOPIC_SPACE

Tipo de datos:

Serie

Propiedad de:

Destino

Nombre utilizado en un URI:

topicSpace

El nombre del espacio de tema que contiene el tema. Solo un destino que es un tema puede tener esta propiedad.

De forma predeterminada, la propiedad no está establecida, lo que significa que se presupone el espacio de tema predeterminado.

Esta propiedad solo es relevante en el dominio de publicación/suscripción.

Referencia de aplicaciones en desarrollo de Managed File Transfer

Información de referencia para ayudarle a desarrollar aplicaciones para Managed File Transfer.

Ejemplos de uso de `fteCreateTransfer` para iniciar programas

Puede utilizar el mandato **`fteCreateTransfer`** para especificar programas a ejecutar antes o después de una transferencia.

Además de utilizar **`fteCreateTransfer`**, hay otras maneras de invocar un programa antes o después de una transferencia. Para obtener más información, consulte [Especificar programas para ejecutar con MFT](#).

Todos estos ejemplos utilizan la siguiente sintaxis para especificar un programa:

```
[type:]commandspec[, [retrycount]][, [retrywait]][, successrc]]
```

Para obtener más información sobre esta sintaxis, consulte [**`fteCreateTransfer`**: iniciar una nueva transferencia de archivos](#).

Ejecutar un programa ejecutable

El ejemplo siguiente especifica un programa ejecutable llamado mycommand y pasa dos argumentos, a y b, al programa.

```
mycommand(a,b)
```

Para ejecutar este programa en el agente de origen AGENT1 antes de que se inicie la transferencia, utilice el siguiente mandato:

```
fteCreateTransfer -sa AGENT1 -da AGENT2 -presrc mycommand(a,b)
destinationSpecification sourceSpecification
```

Ejecutar y reintentar un programa ejecutable

El ejemplo siguiente especifica un programa ejecutable llamado simple, que no acepta ningún argumento. Se especifica un valor de 1 para retrycount y un valor de 5 para retrywait. Estos valores significan que el programa se reintentará una vez si no devuelve un código de retorno satisfactorio, después de una espera de cinco segundos. No se especifica ningún valor para successrc, por lo que el único código de retorno satisfactorio es el valor predeterminado de 0.

```
executable:simple,1,5
```

Para ejecutar este programa en el agente de origen AGENT1 después de que finalice la transferencia, utilice el siguiente mandato:

```
fteCreateTransfer -sa AGENT1 -da AGENT2 -postsrc executable:simple,1,5
destinationSpecification sourceSpecification
```

Ejecutar un Ant script y especificar códigos de retorno correctos

El siguiente ejemplo especifica un Ant script llamado myscript y pasa dos propiedades al script. El script se ejecuta utilizando el mandato **fteAnt**. El valor de successrc se especifica como >2&<7&!5|0|14, lo que indica que los códigos de retorno 0, 3, 4, 6 y 14 indican éxito.

```
antscript:myscript(prop1=fred,prop2=bob),,,>2&<7&!5|0|14
```

Para ejecutar este programa en el agente de destino AGENT2 antes de que se inicie la transferencia, utilice el siguiente mandato:

```
fteCreateTransfer -sa AGENT1 -da AGENT2 -predst
"antscript:myscript(prop1=fred,prop2=bob),,,>2&<7&!5|0|14"destinationSpecification sourceSpecification
```

Ejecutar un Ant script y especificar los objetivos a llamar

El siguiente ejemplo especifica un Ant script llamado script2 y dos objetivos, target1 y target2, a los que llamar. También se pasa la propiedad prop1, con un valor de recmf(F,B). Los paréntesis, las comas (,) y las barras invertidas (\) son caracteres especiales en los comandos MFT, por lo que deben precederse de una barra invertida (\).  En Windows, las rutas de archivo pueden especificarse utilizando barras invertidas dobles (\\) como separador o barras simples (/). En el siguiente ejemplo, la coma (,) y los paréntesis se escapan utilizando un carácter de barra invertida (\).

```
antscript:script2(target1,target2,prop1=recmf(F\,B\)),,,>2&<7&!5|0|14
```

Para ejecutar este programa en el agente de destino AGENT2 después de que finalice la transferencia, utilice el siguiente mandato:

```
fteCreateTransfer -sa AGENT1 -da AGENT2
-postdst "antscript:script2(target1,target2,prop1=recmf(F\,B\)),,,>2&<7&!5|0|14"
destinationSpecification sourceSpecification
```


Uso de metadatos en un Ant script

Puede especificar una Ant tarea como cualquiera de las siguientes llamadas para una transferencia:

- Origen previo
- Origen posterior
- predestinación
- destino de la publicación

Cuando se ejecuta la Ant tarea, los metadatos de usuario de la transferencia se ponen a disposición mediante variables de entorno. Puede acceder a estos datos utilizando, por ejemplo, el siguiente código:

```
<property environment="environment" />
<echo>${environment.mymetadata}</echo>
```

donde mymetadata es el nombre de algunos metadatos insertados en la transferencia.

Ejecutar un script JCL

El ejemplo siguiente especifica un script JCL llamado ZOSBATCH. Se especifica un valor de 3 para `retrycount`, un valor de 30 para `retrywait` y un valor de 0 para `successrc`. Estos valores significan que el script se reintentará tres veces si no devuelve un código de retorno satisfactorio de 0, con una espera de treinta segundos entre cada intento.

```
jcl:ZOSBATCH,3,30,0
```

Donde ZOSBATCH es un miembro de un PDS denominado MYSYS.JCL, y el archivo `agent.properties` contiene la línea `commandPath=...:/'MYSYS.JCL':...`

Para ejecutar este programa en el agente de origen AGENT1 después de que finalice la transferencia, utilice el siguiente mandato:

```
fteCreateTransfer -sa AGENT1 -da AGENT2 -postsrc jcl:ZOSBATCH,3,30,0
destinationSpecification sourceSpecification
```

Tareas relacionadas

[Especificación de programas que se van a ejecutarse con MFT](#)

Referencia relacionada

[**fteCreateTransfer**: iniciar una nueva transferencia de archivos](#)

fteAnt: ejecutar tareas Ant en MFT

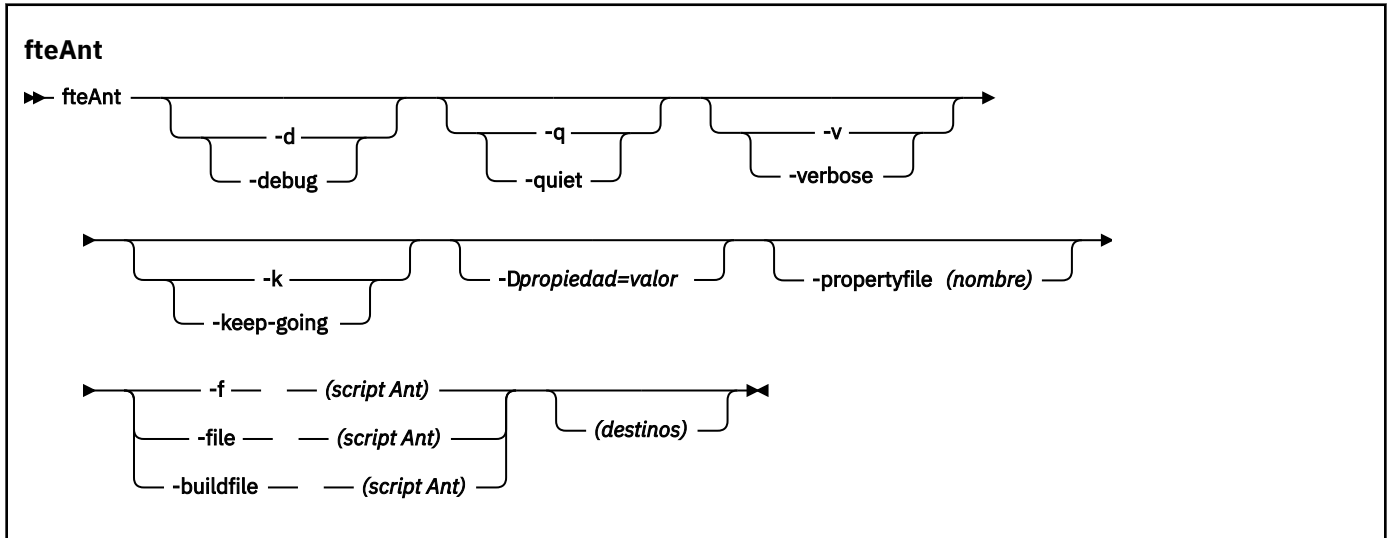
El **fteAnt** comando ejecuta Ant scripts en un entorno que tiene Managed File TransferAnt tareas disponibles. A diferencia del mandato **ant** estándar, **fteAnt** requiere definir un archivo de script.

AntMFT tareas y parámetros anidados

Managed File Transfer proporciona una serie de Ant tareas que puede utilizar para acceder a las funciones de transferencia de archivos. También hay disponible un conjunto de parámetros anidados; estos parámetros describen conjuntos anidados de elementos que son comunes en varias de las tareas Ant proporcionadas.

La sintaxis del **fteAnt** comando, los parámetros, el ejemplo de uso y los códigos de retorno se describen en el resto de este tema. Para obtener más información sobre las Ant tareas y los parámetros anidados que proporciona MFT, véanse los subtemas.

fteAnt sintaxis



Parámetros

debug o -d

Opcional. Generar salida de depuración.

-quiet o -q

Opcional. Generar salida mínima.

-verbose o -v

Opcional. Generar salida detallada.

-keep-going o -k

Opcional. Ejecutar todos los destinos que no dependen de destinos anómalos.

-D propiedad=valor

Opcional. Utilizar un *valor* para una determinada *propiedad*. Las prioridades establecidas con **-D** tienen prioridad sobre las establecidas en un archivo de propiedades.

Utilice la propiedad **com.ibm.wmqfte.propertyset** para especificar el conjunto de opciones de configuración que se utilizan para Ant las tareas. Utilice el nombre de un gestor de colas de coordinación no predeterminado como el valor de esta propiedad. Ant Las tareas utilizan entonces el conjunto de opciones de configuración asociadas a este gestor de colas de coordinación no predeterminado. Si no especifica esta propiedad, se utiliza el conjunto predeterminado de opciones de configuración que se basan en el gestor de colas de coordinación predeterminado. Si se especifica el **cmdqm** atributo para una Ant tarea, este atributo tiene prioridad sobre el conjunto de opciones de configuración especificadas para el **fteAnt** comando. Este comportamiento se aplica independientemente de si está utilizando el conjunto predeterminado de opciones de configuración o especificando un conjunto con la propiedad **com.ibm.wmqfte.propertyset**.

-propertyfile (nombre)

Opcional. Carga todas las propiedades de un archivo con las propiedades **-D** prioritarias.

-f (script Ant), -file (script Ant), or -buildfile (script Ant)

Necesario. Especifica el nombre del Ant script que se va a ejecutar.

destinos

Opcional. El nombre de uno o más destinos para ejecutar desde el script Ant. Si no especifica un valor para este parámetro, se ejecuta el destino predeterminado para el script.

-version

Opcional. Muestra el Managed File Transfer comando y Ant las versiones.

-? o -h

Opcional. Muestra la sintaxis del mandato.

Ejemplo

En este ejemplo, se ejecuta el mandato **copy** de destino del script Ant `fte_script.xml` y el mandato graba la salida de depuración en la salida estándar.

```
fteAnt -d -f fte_script.xml copy
```

Códigos de retorno

0

El mandato se ha completado satisfactoriamente.

1

El mandato no ha finalizado correctamente.

También se pueden especificar otros códigos de retorno de estado desde scripts Ant, por ejemplo, utilizando la Ant tarea `fail`.

Consulte [Fail](#) para obtener más información.

Conceptos relacionados

[Cómo empezar a utilizar scripts Ant con MFT](#)

Tareas relacionadas

[Utilización de Apache Ant con MFT](#)

Referencia relacionada

[Tareas de ejemplo Ant para MFT](#)

Tarea **fte:awaitoutcome** Ant

Espera hasta que se complete una operación **fte:filecopy**, **fte:filemove** o **fte:call**.

Atributos

id

Necesario. Identifica la transferencia de la que se espera un resultado. Normalmente, es una propiedad establecida por el atributo `idProperty` de las tareas [fte:filecopy](#), [fte:filemove](#) o [fte:call](#).

rcproperty

Necesario. Da nombre a una propiedad en la que se almacena el código de retorno de la tarea **fte:awaitoutcome**.

timeout

Opcional. La cantidad máxima de tiempo, en segundos, de espera hasta que se complete la operación. El tiempo mínimo de espera excedido es un segundo. Si no especifica un valor de tiempo de espera, la tarea **fte:awaitoutcome** espera de forma indefinida hasta que se determine el resultado de la operación.

Ejemplo

En este ejemplo, se inicia una copia de archivo y el identificador se almacena en la propiedad `copy.id`. Mientras la copia está en proceso, puede realizarse otro proceso. La sentencia **fte:awaitoutcome** se utiliza para esperar hasta que se complete la operación de copiar. La sentencia **fte:awaitoutcome** identifica qué operación hay que esperar para utilizar el identificador almacenado en la propiedad `copy.id`. La sentencia **fte:awaitoutcome** almacena un código de retorno indicando el resultado de la operación de copiar en una propiedad denominada `copy.result`.

```
<-- issue a file copy request -->
<fte:filecopy
src="AGENT1@QM1"
dst="AGENT2@QM2"
idproperty="copy.id"
outcome="defer">

<fte:filespec
```

```
srcfilespec="/home/fteuser1/file.bin"
dstdir="/home/fteuser2"/>

</fte:filecopy>

<fte:awaitoutcome id="{copy.id}" rcProperty="copy.rc"/>

<echo>Copy id="{copy.id}" rc="{copy.rc}"</echo>
```

Tareas relacionadas

[Utilización de Apache Ant con MFT](#)

Tarea fte: call Ant

Puede utilizar la tarea **fte:call** para llamar a scripts y programas de forma automática.

Esta tarea le permite enviar una solicitud **fte:call** a un agente. El agente procesa esta solicitud ejecutando un script o un programa y devolviendo el resultado. Los mandatos para llamar deben ser accesibles para el agente. Asegúrese de que el valor de propiedad `commandPath` del archivo `agent.properties` incluye la ubicación de los mandatos para llamar. La información de vía de acceso especificada por el elemento anidado del mandato debe ser relativa a las ubicaciones especificadas por la propiedad `commandPath`. De forma predeterminada, `commandPath` está vacío, por tanto, el agente no puede llamar a ningún mandato. Para obtener más información sobre esta propiedad, consulte [PropiedadcommandPath MFT](#).

Para obtener más información sobre el archivo `agent.properties`, consulte [El archivo MFT agent.properties](#).

Atributos

agent

Necesario. Especifica el agente al que enviar la solicitud **fte:call**. Especifique la información de agente con el formato: *agentname@qmgrname* donde *agentname* es el nombre del agente y *qmgrname* es el nombre del gestor de colas al que está conectado directamente este agente.

cmdqm

Opcional. El gestor de colas de mandatos al que se somete la solicitud. Especifique esta información en el formato *qmgrname@host@port@channel*, donde:

- *qmgrname* es el nombre del gestor de colas
- *host* es el nombre de host opcional del sistema donde se está ejecutando el gestor de colas
- *port* es el número de puerto opcional en el que escucha el gestor de colas
- *channel* es el canal SVRCONN opcional a utilizar

Si omite la información de *host*, *port* o *channel* para el gestor de colas de mandatos, se utiliza la información de conexión especificada en el archivo `command.properties`.



Atención: Si no se especifica ningún valor para:

- variable *host*, se utiliza la modalidad de enlaces
- *port*, se utiliza el valor 1414
- *channel*, la variable `SYSTEM.DEF.SVRCONN`.

Consulte [El archivo command.properties de MFT](#) para obtener más información.

Sin embargo, no puede omitir los atributos en el medio, por ejemplo, *qmgrname@host@@channel*. Puede tener, por ejemplo, *qmgrname@host*, o *qmgrname@host@port*, o *qmgrname@hostport@@channel*.

MFT divide el atributo especificado utilizando el delimitador @. En función del número de señales encontradas, toma la primera señal como *qmgrname*, la segunda como *host*, la tercera como *puerto* y finalmente *canal*.

Para obtener más información, consulte [El archivo MFT command.properties](#).

Puede utilizar la propiedad **com.ibm.wmqfte.propertySet** para especificar qué archivo `command.properties` se debe utilizar. Para obtener más información, consulte [com.ibm.wmqfte.propertySet](#).

Si no utiliza el atributo `cmdqm`, la tarea utiliza de forma predeterminada la propiedad `com.ibm.wmqfte.ant.commandQueueManager`, si se ha establecido esta propiedad. Si la propiedad `com.ibm.wmqfte.ant.commandQueueManager` no está establecida, se intenta una conexión con el gestor de colas predeterminado, definido en el archivo `command.properties`. El formato de la propiedad `com.ibm.wmqfte.ant.commandQueueManager` es el mismo que el atributo `cmdqm`, es decir, `qmgrname@host@port@channel`.

idproperty

Opcional a menos que haya especificado una propiedad `outcome` de `defer`. Especifica el nombre de una propiedad a la que se asigna el identificador de transferencia. Los identificadores de transferencia se generan en el punto en que se somete una solicitud de transferencia y se pueden utilizar identificadores de transferencia para realizar un seguimiento del progreso de una transferencia, diagnosticar problemas con una transferencia y cancelarla.

No puede especificar esta propiedad si también ha especificado una propiedad `outcome` de `ignore`. Pero debe especificar `idproperty` si también ha especificado una propiedad `outcome` de `defer`.

jobname

Opcional. Asigna un nombre de trabajo a la solicitud **fte:call**. Puede utilizar nombres de trabajo para crear grupos lógicos de transferencias. Utilice la tarea “Tarea `fte: uuid Ant`” en la [página 2157](#) para generar nombres de trabajo pseudoexclusivos. Si no utiliza el atributo `jobname`, la tarea utiliza de forma predeterminada el valor de la propiedad `com.ibm.wmqfte.ant.jobName`, si se ha establecido esta propiedad. Si no se establece esta propiedad, no se asociará ningún nombre de trabajo a la solicitud **fte:call**.

origuser

Opcional. Especifica el identificador del usuario de origen para asociar a la solicitud **fte:call**. Si no utiliza el atributo `origuser`, la tarea adopta de forma predeterminada el ID de usuario que se utiliza para ejecutar el script Ant.

outcome

Opcional. Determina si la tarea espera a que se complete la operación **fte:call** antes de devolver el control al script Ant. Especifique una de las opciones siguientes:

await

La tarea espera a que se complete la operación **fte:call** antes de volver. Cuando se especifica una propiedad `outcome` de `await`, el atributo `idproperty` es opcional.

defer

La tarea vuelve tan pronto como se ha enviado la solicitud **fte:call** y presupone que el resultado de la operación de llamada se trata más adelante utilizando las tareas [awaitoutcome](#) o [ignoreoutcome](#). Cuando se especifica una propiedad `outcome` de `defer`, el atributo `idproperty` es necesario.

ignore

Si el resultado de la operación **fte:call** no es importante, puede especificar un valor de `ignore`. La tarea volverá en cuanto se haya sometido la solicitud **fte:call**, sin asignar ningún recurso para realizar el seguimiento del resultado del mandato. Cuando se especifica una propiedad `outcome` de `ignore`, el atributo `idproperty` no se puede especificar.

Si no especifica el atributo `outcome`, la tarea adopta de forma predeterminada el valor `await`.

rcproperty

Opcional. Especifica el nombre de una propiedad a la que se asigna el código de resultado de la solicitud **fte:call**. El código de resultado refleja el resultado global de la solicitud **fte:call**.

No puede especificar esta propiedad si también ha especificado una propiedad `outcome` de `ignore` o `defer`. Sin embargo, debe especificar `rcproperty` si ha especificado un resultado de `await`.

Parámetros especificados como elementos anidados

fte:command

Especifica el mandato invocado por el agente. Sólo puede asociar un único elemento **fte:command** con una operación **fte:call** determinada. El mandato que se va a invocar debe estar situado en la vía de acceso especificada por la propiedad `commandPath` en el archivo `agent.properties` del agente.

fte:metadata

Puede especificar metadatos para asociarlos con la operación de llamada. Estos metadatos se graban en los mensajes de registro generados por la operación de llamada. Sólo puede asociar un único bloque de metadatos con un elemento de transferencia determinado, sin embargo, este bloque puede contener muchos fragmentos de metadatos.

Ejemplo

Este ejemplo muestra cómo invocar un mandato en AGENT1 que se ejecuta en el gestor de colas QM1. El mandato para invocar es el script `command.sh` y el script se invoca con un único argumento de `xyz`. El mandato `command.sh` está situado en la vía de acceso especificada por la propiedad `commandPath` en el archivo `agent.properties` del agente.

```
<fte:call cmdqm="QM0@localhost@1414@SYSTEM.DEF.SVRCONN"
  agent="AGENT1@QM1"
  rcproperty="call.rc"
  origuser="bob"
  jobname="{i}job.id}">

  <fte:command command="command.sh" successrc="1" retrycount="5" retrywait="30">
    <fte:arg value="xyz"/>
  </fte:command>

  <fte:metadata>
    <fte:entry name="org.foo.accountName" value="BDG3R"/>
  </fte:metadata>

</fte:call>
```

Tareas relacionadas

[Utilización de Apache Ant con MFT](#)

fte: cancelar tarea Ant

Cancela una transferencia gestionada o una llamada gestionada de Managed File Transfer. Se puede haber creado una transferencia gestionada mediante las tareas **fte:filecopy** o **fte:filemove**. Se puede haber creado una llamada gestionada mediante la tarea **fte:call**.

Atributos

agent

Necesario. Especifica el agente al que se somete la solicitud **fte:cancel**. El valor tiene el formato: *agentname@qmgrname* donde *agentname* es el nombre del agente y *qmgrname* es el nombre del gestor de colas al que está conectado directamente este agente.

cmdqm

Opcional. El gestor de colas de mandatos al que se somete la solicitud. Especifique esta información en el formato *qmgrname@host@port@channel*, donde:

- *qmgrname* es el nombre del gestor de colas
- *host* es el nombre de host opcional del sistema donde se está ejecutando el gestor de colas
- *port* es el número de puerto opcional en el que escucha el gestor de colas
- *channel* es el canal SVRCONN opcional a utilizar

Si omite la información de *host*, *port* o *channel* para el gestor de colas de mandatos, se utiliza la información de conexión especificada en el archivo `command.properties`.



Atención: Si no se especifica ningún valor para:

- variable *host* , se utiliza la modalidad de enlaces
- *port* , se utiliza el valor 1414
- *channel* , la variable SYSTEM.DEF.SVRCONN .

Consulte [El archivo command.properties de MFT](#) para obtener más información.

Sin embargo, no puede omitir los atributos en el medio, por ejemplo, `qmgrname@host@@channel`. Puede tener, por ejemplo, `qmgrname@host`, o `qmgrname@host@port`, o `qmgrname@hostport@@channel`.

MFT divide el atributo especificado utilizando el delimitador @ . En función del número de señales encontradas, toma la primera señal como *qmgrname*, la segunda como *host*, la tercera como *puerto* y finalmente *canal*.

Para obtener más información, consulte [El archivo MFT command.properties](#).

Puede utilizar la propiedad **com.ibm.wmqfte.propertySet** para especificar qué archivo `command.properties` se debe utilizar. Para obtener más información, consulte [com.ibm.wmqfte.propertySet](#).

Si no utiliza el atributo `cmdqm`, la tarea utiliza de forma predeterminada la propiedad `com.ibm.wmqfte.ant.commandQueueManager`, si se ha establecido esta propiedad. Si la propiedad `com.ibm.wmqfte.ant.commandQueueManager` no está establecida, se intenta una conexión con el gestor de colas predeterminado, definido en el archivo `command.properties`. El formato de la propiedad `com.ibm.wmqfte.ant.commandQueueManager` es el mismo que el atributo `cmdqm` , es decir, `qmgrname@host@port@channel`.

id

Necesario. Especifica el identificador de transferencia de la transferencia que se desea cancelar. Los identificadores de transferencia se generan en el punto en que las tareas [fte:filecopy](#) y [fte:filemove](#) someten una solicitud de transferencia.

origuser

Opcional. Especifica el identificador del usuario de origen para asociar con la solicitud **cancel**. Si no se utiliza el atributo `origuser` , la tarea utiliza de forma predeterminada el ID de usuario que se utiliza para ejecutar el script Ant .

Ejemplo

El ejemplo envía una solicitud **fte:cancel** al gestor de colas de mandatos `qm0`. La solicitud **fte:cancel** tiene como destino `agent1` en el gestor de colas `qm1` para el identificador de transferencia rellenado por la variable `transfer.id`. La solicitud se ejecuta mediante el ID de usuario "bob".

```
<fte:cancel cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
  agent="agent1@qm1"
  id="{transfer.id}"
  origuser="bob"/>
```

Tareas relacionadas

[Utilización de Apache Ant con MFT](#)

Tarea Ant fte:filecopy

La tarea **fte:filecopy** copia archivos entre agentes de Managed File Transfer. El archivo no se suprime del agente de origen.

Atributos

cmdqm

Opcional. El gestor de colas de mandatos al que se somete la solicitud. Especifique esta información en el formato `qmgrname@host@port@channel`, donde:

- *qmgrname* es el nombre del gestor de colas
- *host* es el nombre de host opcional del sistema donde se está ejecutando el gestor de colas
- *port* es el número de puerto opcional en el que escucha el gestor de colas
- *channel* es el canal SVRCONN opcional a utilizar

Si omite la información de *host*, *port* o *channel* para el gestor de colas de mandatos, se utiliza la información de conexión especificada en el archivo `command.properties`.



Atención: Si no se especifica ningún valor para:

- variable *host*, se utiliza la modalidad de enlaces
- *port*, se utiliza el valor 1414
- *channel*, la variable `SYSTEM.DEF.SVRCONN`.

Consulte [El archivo `command.properties` de MFT](#) para obtener más información.

Sin embargo, no puede omitir los atributos en el medio, por ejemplo, `qmgrname@host@@channel`. Puede tener, por ejemplo, `qmgrname@host`, o `qmgrname@host@port`, o `qmgrname@hostport@@channel`.

MFT divide el atributo especificado utilizando el delimitador @. En función del número de señales encontradas, toma la primera señal como *qmgrname*, la segunda como *host*, la tercera como *puerto* y finalmente *canal*.

Para obtener más información, consulte [El archivo MFT `command.properties`](#).

Puede utilizar la propiedad **`com.ibm.wmqfte.propertySet`** para especificar qué archivo `command.properties` se debe utilizar. Para obtener más información, consulte [com.ibm.wmqfte.propertySet](#).

Si no utiliza el atributo `cmdqm`, la tarea utiliza de forma predeterminada la propiedad `com.ibm.wmqfte.ant.commandQueueManager`, si se ha establecido esta propiedad. Si la propiedad `com.ibm.wmqfte.ant.commandQueueManager` no está establecida, se intenta una conexión con el gestor de colas predeterminado, definido en el archivo `command.properties`. El formato de la propiedad `com.ibm.wmqfte.ant.commandQueueManager` es el mismo que el atributo `cmdqm`, es decir, `qmgrname@host@port@channel`.

dst

Necesario. Especifica el agente de destino para la operación de copiar. Especifique esta información con el formato: `agentname@qmgrname` donde `agentname` es el nombre del agente de destino y `qmgrname` es el nombre del gestor de colas al que está conectado directamente este agente.

idproperty

Opcional a menos que haya especificado una propiedad `outcome` de `defer`. Especifica el nombre de una propiedad a la que se asigna el identificador de transferencia. Los identificadores de transferencia se generan en el punto en que se somete una solicitud de transferencia y se pueden utilizar identificadores de transferencia para realizar un seguimiento del progreso de una transferencia, diagnosticar problemas con una transferencia y cancelarla.

No puede especificar esta propiedad si también ha especificado una propiedad `outcome` de `ignore`. Pero debe especificar `idproperty` si también ha especificado una propiedad `outcome` de `defer`.

jobname

Opcional. Asigna un nombre de trabajo a la solicitud de copiar. Puede utilizar nombres de trabajo para crear grupos lógicos de transferencias. Utilice la tarea “Tarea fte: uuid Ant” en la [página 2157](#) para generar nombres de trabajo pseudoexclusivos. Si no utiliza el atributo `jobname`, la tarea utiliza de forma predeterminada el valor de la propiedad `com.ibm.wmqfte.ant.jobName`, si se ha establecido esta propiedad. Si no establece esta propiedad, no se asociará ningún nombre de trabajo a la solicitud de copiar.

origuser

Opcional. Especifica el identificador del usuario de origen para asociar a la solicitud de copiar. Si no utiliza el atributo `origuser`, la tarea utiliza de forma predeterminada el ID de usuario que se utiliza para ejecutar el script Ant.

outcome

Opcional. Determina si la tarea espera a que se complete la operación de copia antes de devolver el control al script Ant . Especifique una de las opciones siguientes:

await

La tarea espera a que se complete la operación de copiar antes de volver. Cuando se especifica una propiedad outcome de await, el atributo idproperty es opcional.

defer

La tarea vuelve tan pronto como se ha enviado la solicitud de copia y presupone que el resultado de la operación de copia se trata más adelante utilizando las tareas “Tarea fte: awaitoutcome Ant” en la página 2143 o “Tarea fte: ignoreoutcome Ant” en la página 2155 . Cuando se especifica una propiedad outcome de defer, el atributo idproperty es necesario.

ignore

Si el resultado de la operación de copiar no es importante, puede especificar un valor de ignore. La tarea volverá en cuanto se haya sometido la solicitud de copiar, sin asignar ningún recurso para realizar el seguimiento del resultado de la transferencia. Cuando se especifica una propiedad outcome de ignore, el atributo idproperty no se puede especificar.

Si no especifica el atributo outcome, la tarea adopta de forma predeterminada el valor await.

priority

Opcional. Especifica la prioridad para asociar a la solicitud de copiar. En general, las solicitudes de prioridad más altas tienen prioridad sobre las solicitudes de prioridad más bajas. El valor de prioridad debe estar en el rango entre 0 y 9 (inclusive). Un valor de prioridad de 0 es la prioridad más baja y un valor de 9 es la prioridad más alta. Si no especifica el atributo priority, la transferencia adopta una prioridad de 0.

rcproperty

Opcional. Especifica el nombre de una propiedad a la que se asigna el código de resultado de la solicitud de copiar. El código de resultado refleja el resultado global de la solicitud de copiar.

No puede especificar esta propiedad si también ha especificado una propiedad outcome de ignore o defer. Sin embargo, debe especificar rcproperty si especifica un resultado de await.

transferRecoveryTimeout

Opcional. Establece la cantidad de tiempo, en segundos, durante el cual un agente de origen sigue intentando recuperar una transferencia de archivo estancada. Especifique una de las opciones siguientes:

-1

El agente sigue intentando recuperar la transferencia estancada hasta que ésta se lleve a cabo. La utilización de esta opción equivale al comportamiento predeterminado del agente cuando la propiedad no se ha establecido.

0

El agente detiene la transferencia de archivo tan pronto como se inicia la recuperación.

>0

El agente sigue intentando recuperar la transferencia estancada durante el periodo de tiempo en segundos según se haya establecido mediante el valor entero positivo especificado. Por ejemplo,

```
<fte:filecopy cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
src="agent1@qm1" dst="agent2@qm2"
rcproperty="copy.result" transferRecoveryTimeout="21600">

  <fte:filespec srcfilespec="/home/fteuser1/file.bin" dstfile="/home/fteuser2/
file.bin"/>

</fte:filecopy>
```

indica que el agente sigue intentando recuperar la transferencia durante 6 horas desde que se inició la recuperación. El valor máximo para este atributo es 999999999.

Si se especifica el valor de tiempo de espera de recuperación de transferencia de esta forma, se establece en base a la transferencia. Para establecer un valor global para todas las transferencias en una red Managed File Transfer , puede añadir una propiedad a las [Propiedades de tiempo de](#)

espera de recuperación de transferencia. Para obtener más información, consulte [Opción de tiempo de espera para transferencias en recuperación](#).

src

Necesario. Especifica el agente de origen para la operación de copiar. Especifique esta información en el formato: *nombreagente@nombregestorcolas* donde *nombreagente* es el nombre del agente de origen y *nombregestorcolas* es el nombre del gestor de colas con el que está conectado directamente este agente.

Parámetros especificados como elementos anidados

fte:filespec

Necesario. Debe especificar al menos una especificación de archivo que identifique los archivos para copiar. Puede especificar que más de una especificación de archivo sea necesaria. Consulte [“Elemento anidado fte: filespec Ant” en la página 2157](#) para obtener más información.

fte:metadata

Puede especificar metadatos para asociarlos con la operación de copiar. Estos metadatos se transportan con la transferencia y se graban en los mensajes de registro generados por la transferencia. Sólo puede asociar un único bloque de metadatos con un elemento de transferencia determinado, sin embargo, este bloque puede contener muchos fragmentos de metadatos. Para obtener más información, consulte el tema [fte:metadata](#).

fte:presrc

Especifica una invocación de programa que tendrá lugar en el agente de origen antes de comenzar la transferencia. Sólo puede asignar un único elemento `fte:presrc` con una transferencia determinada. Consulte el tema [invocación de programas](#) para obtener más información.

fte:predst

Especifica una invocación de programa que tendrá lugar en el agente de destino antes de comenzar la transferencia. Sólo puede asignar un único elemento `fte:predst` con una transferencia determinada. Consulte el tema [invocación de programas](#) para obtener más información.

fte:postsrc

Especifica una invocación de programa que tendrá lugar en el agente de origen después de haber completado la transferencia. Sólo puede asignar un único elemento `fte:postsrc` con una transferencia determinada. Consulte el tema [invocación de programas](#) para obtener más información.

fte:postdst

Especifica una invocación de programa que tendrá lugar en el agente de destino después de haber completado la transferencia. Sólo puede asignar un único elemento `fte:postdst` con una transferencia determinada. Consulte el tema [invocación de programas](#) para obtener más información.

Si `fte:presrc`, `fte:predst`, `fte:postsrc`, `fte:postdst` y `exits` no devuelven un estado de éxito, las reglas se implantarán en el orden especificado:

1. Ejecute las salidas de inicio del origen. Si las salidas de inicio del origen fallan, la transferencia falla y no se ejecuta nada más.
2. Ejecute la llamada previa al origen (si existe). Si la llamada previa al origen falla, la transferencia falla y no se ejecuta nada más.
3. Ejecute las salidas de inicio del destino. Si las salidas de inicio del destino fallan, la transferencia falla y no se ejecuta nada más.
4. Ejecute la llamada previa al destino (si existe). Si la llamada previa al destino falla, la transferencia falla y no se ejecuta nada más.
5. Realice las transferencias de archivos.
6. Ejecute las salidas de finalización del destino. No existe ningún estado de anomalía para estas salidas.
7. Si la transferencia es satisfactoria (si algunos archivos se transfieren satisfactoriamente, se considera satisfactoria), ejecute la llamada posterior al destino (si está presente). Si la llamada posterior al destino falla, la transferencia falla.
8. Ejecute las salidas de finalización del origen. No existe ningún estado de anomalía para estas salidas.

9. Si la transferencia es satisfactoria, ejecute la llamada posterior al origen (si está presente). Si la llamada posterior al origen falla, la transferencia falla.

Ejemplos

En este ejemplo se muestra una transferencia básica de archivos entre agent1 y agent2. El mandato para iniciar la transferencia de archivos se envía a un gestor de colas denominado qm0, utilizando una conexión de modalidad de transporte de cliente. El resultado de la operación de transferencia de archivos se asigna a la propiedad denominada copy.result.

```
<fte:filecopy cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
  src="agent1@qm1" dst="agent2@qm2"
  rcproperty="copy.result">

  <fte:filespec srcfilespec="/home/fteuser1/file.bin" dstfile="/home/fteuser2/file.bin"/>

</fte:filecopy>
```

En este ejemplo se muestra la misma transferencia de archivos, pero con la incorporación de metadatos y un inicio de programa que tiene lugar en el agente de origen, después de que haya terminado la transferencia.

```
<fte:filecopy cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
  src="agent1@qm1" dst="agent2@qm2"
  rcproperty="copy.result">

  <fte:metadata>
    <fte:entry name="org.example.departId" value="ACCOUNTS"/>
    <fte:entry name="org.example.batchGroup" value="A1"/>
  </fte:metadata>

  <fte:filespec srcfilespec="/home/fteuser1/file.bin" dstfile="/home/fteuser2/file.bin"/>

  <fte:postsrc command="/home/fteuser2/scripts/post.sh" successsrc="1" >
    <fte:arg value="/home/fteuser2/file.bin"/>
  </fte:postsrc>
</fte:filecopy>
```

Conceptos relacionados

[Opción de tiempo de espera para transferencias de archivos en la recuperación](#)

Tareas relacionadas

[Utilización de Apache Ant con MFT](#)

Tarea Ant fte:filemove

La tarea **fte:filemove** mueve archivos entre agentes de Managed File Transfer. Cuando un archivo se ha transferido satisfactoriamente desde el agente de origen hasta el agente de destino, el archivo de suprime del agente de origen.

Atributos

cmdqm

Opcional. El gestor de colas de mandatos al que se somete la solicitud. Especifique esta información en el formato *qmgrname@host@port@channel*, donde:

- *qmgrname* es el nombre del gestor de colas
- *host* es el nombre de host opcional del sistema donde se está ejecutando el gestor de colas
- *port* es el número de puerto opcional en el que escucha el gestor de colas
- *channel* es el canal SVRCONN opcional a utilizar

Si omite la información de *host*, *port* o *channel* para el gestor de colas de mandatos, se utiliza la información de conexión especificada en el archivo *command.properties*.



Atención: Si no se especifica ningún valor para:

- variable *host* , se utiliza la modalidad de enlaces
- *port* , se utiliza el valor 1414
- *channel* , la variable SYSTEM.DEF.SVRCONN .

Consulte [El archivo command.properties de MFT](#) para obtener más información.

Sin embargo, no puede omitir los atributos en el medio, por ejemplo, *qmgrname@host@@channel*. Puede tener, por ejemplo, *qmgrname@host*, o *qmgrname@host@port*, o *qmgrname@hostport@@channel*.

MFT divide el atributo especificado utilizando el delimitador @ . En función del número de señales encontradas, toma la primera señal como *qmgrname*, la segunda como *host*, la tercera como *puerto* y finalmente *canal*.

Para obtener más información, consulte [El archivo MFT command.properties](#).

Puede utilizar la propiedad **com.ibm.wmqfte.propertySet** para especificar qué archivo *command.properties* se debe utilizar. Para obtener más información, consulte [com.ibm.wmqfte.propertySet](#).

Si no utiliza el atributo *cmdqm*, la tarea utiliza de forma predeterminada la propiedad *com.ibm.wmqfte.ant.commandQueueManager*, si se ha establecido esta propiedad. Si la propiedad *com.ibm.wmqfte.ant.commandQueueManager* no está establecida, se intenta una conexión con el gestor de colas predeterminado, definido en el archivo *command.properties*. El formato de la propiedad *com.ibm.wmqfte.ant.commandQueueManager* es el mismo que el atributo *cmdqm* , es decir, *qmgrname@host@port@channel*.

dst

Necesario. Especifica el agente de destino para la operación de copiar. Especifique esta información con el formato: *agentname@qmgrname* donde *agentname* es el nombre del agente de destino y *qmgrname* es el nombre del gestor de colas al que está conectado directamente este agente.

idproperty

Opcional a menos que haya especificado una propiedad *outcome* de *defer*. Especifica el nombre de una propiedad a la que se asigna el identificador de transferencia. Los identificadores de transferencia se generan en el punto en que se somete una solicitud de transferencia y se pueden utilizar identificadores de transferencia para realizar un seguimiento del progreso de una transferencia, diagnosticar problemas con una transferencia y cancelarla.

No puede especificar esta propiedad si también ha especificado una propiedad *outcome* de *ignore*. Pero debe especificar *idproperty* si también ha especificado una propiedad *outcome* de *defer*.

jobname

Opcional. Asigna un nombre de trabajo a la solicitud de mover. Puede utilizar nombres de trabajo para crear grupos lógicos de transferencias. Utilice la tarea *fte:uuid* para generar nombres de trabajo pseudoexclusivos. Si no utiliza el atributo *jobname*, la tarea utiliza de forma predeterminada el valor de la propiedad *com.ibm.wmqfte.ant.jobName*, si se ha establecido esta propiedad. Si no establece esta propiedad, no se asociará ningún nombre de trabajo a la solicitud de mover.

origuser

Opcional. Especifica el identificador del usuario de origen para asociar a la solicitud de mover. Si no utiliza el atributo *origuser* , la tarea utiliza de forma predeterminada el ID de usuario que se utiliza para ejecutar el script Ant .

outcome

Opcional. Determina si la tarea espera a que se complete la operación de traslado antes de devolver el control al script Ant . Especifique una de las opciones siguientes:

await

La tarea espera a que se complete la operación de mover antes de volver. Cuando se especifica una propiedad *outcome* de *await*, el atributo *idproperty* es opcional.

defer

La tarea vuelve tan pronto como se ha enviado la solicitud de traslado y presupone que el resultado de la operación de traslado se trata más adelante utilizando la tarea “Tarea fte: awaitoutcome Ant” en la página 2143 o “Tarea fte: ignoreoutcome Ant” en la página 2155 . Cuando se especifica una propiedad outcome de defer, el atributo idproperty es necesario.

ignore

Si el resultado de la operación de mover no es importante, puede especificar un valor de ignore. La tarea volverá en cuanto se haya sometido la solicitud de mover, sin asignar ningún recurso para realizar el seguimiento del resultado de la transferencia. Cuando se especifica una propiedad outcome de ignore, el atributo idproperty no se puede especificar.

Si no especifica el atributo outcome, la tarea adopta de forma predeterminada el valor await.

priority

Opcional. Especifica la prioridad para asociar a la solicitud de mover. En general, las solicitudes de prioridad más altas tienen prioridad sobre las solicitudes de prioridad más bajas. El valor de prioridad debe estar en el rango entre 0 y 9 (inclusive). Un valor de prioridad de 0 es la prioridad más baja y un valor de 9 es la prioridad más alta. Si no especifica el atributo priority, la transferencia adopta una prioridad de 0.

rcproperty

Opcional. Especifica el nombre de una propiedad a la que se asigna el código de resultado de la solicitud de mover. El código de resultado refleja el resultado global de la solicitud de mover.

No puede especificar esta propiedad si también ha especificado una propiedad outcome de ignore o defer. Sin embargo, debe especificar rcproperty si ha especificado un resultado de await.

transferRecoveryTimeout

Opcional. Establece la cantidad de tiempo, en segundos, durante el cual un agente de origen sigue intentando recuperar una transferencia de archivo estancada. Especifique una de las opciones siguientes:

-1

El agente sigue intentando recuperar la transferencia estancada hasta que ésta se lleve a cabo. La utilización de esta opción equivale al comportamiento predeterminado del agente cuando la propiedad no se ha establecido.

0

El agente detiene la transferencia de archivo tan pronto como se inicia la recuperación.

>0

El agente sigue intentando recuperar la transferencia estancada durante el periodo de tiempo en segundos según se haya establecido mediante el valor entero positivo especificado. Por ejemplo,

```
<fte:filemove cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
  src=agent1@qm1 dst="agent2@qm2"
  rcproperty="move.result" transferRecoveryTimeout="21600">

  <fte:filespec srcfilespec="/home/fteuser1/file.bin" dstfile="/home/fteuser2/
file.bin"/>

</fte:filemove
```

indica que el agente sigue intentando recuperar la transferencia durante 6 horas desde que se inició la recuperación. El valor máximo para este atributo es 999999999.

Si se especifica el valor de tiempo de espera de recuperación de transferencia de esta forma, se establece en base a la transferencia. Para establecer un valor global para todas las transferencias en una red Managed File Transfer , puede añadir una propiedad a las Propiedades de tiempo de espera de recuperación de transferencia. Para obtener más información, consulte [Opción de tiempo de espera para transferencias en recuperación](#).

src

Necesario. Especifica el agente de origen para la operación de mover. Especifique esta información con el formato: *agentname@qmgrname* donde *agentname* es el nombre del agente de origen y *qmgrname* es el nombre del gestor de colas al que está conectado directamente este agente.

Parámetros especificados como elementos anidados

fte:filespec

Necesario. Debe especificar al menos una especificación de archivo que identifique los archivos para mover. Puede especificar que más de una especificación de archivo sea necesaria. Consulte [“Elemento anidado fte: filespec Ant” en la página 2157](#) para obtener más información.

fte:metadata

Opcional. Puede especificar metadatos para asociarlos con la operación de mover archivo. Estos metadatos se transportan con la transferencia y se graban en los mensajes de registro generados por la transferencia. Sólo puede asociar un único bloque de metadatos con un elemento de transferencia determinado, sin embargo, este bloque puede contener muchos fragmentos de metadatos. Para obtener más información, consulte el tema [fte:metadata](#).

fte:presrc

Opcional. Especifica una invocación de programa que tendrá lugar en el agente de origen antes de comenzar la transferencia. Sólo puede asignar un único elemento `fte:presrc` con una transferencia determinada. Consulte el tema [invocación de programas](#) para obtener más información.

fte:predst

Opcional. Especifica una invocación de programa que tendrá lugar en el agente de destino antes de comenzar la transferencia. Sólo puede asignar un único elemento `fte:predst` con una transferencia determinada. Consulte el tema [invocación de programas](#) para obtener más información.

fte:postsrc

Opcional. Especifica una invocación de programa que tendrá lugar en el agente de origen después de haber completado la transferencia. Sólo puede asignar un único elemento `fte:postsrc` con una transferencia determinada. Consulte el tema [invocación de programas](#) para obtener más información.

fte:postdst

Opcional. Especifica una invocación de programa que tendrá lugar en el agente de destino después de haber completado la transferencia. Sólo puede asignar un único elemento `fte:postdst` con una transferencia determinada. Consulte el tema [invocación de programas](#) para obtener más información.

Si `fte:presrc`, `fte:predst`, `fte:postsrc`, `fte:postdst` y `exits` no devuelven un estado de éxito, las reglas se implantarán en el orden especificado:

1. Ejecute las salidas de inicio del origen. Si las salidas de inicio del origen fallan, la transferencia falla y no se ejecuta nada más.
2. Ejecute la llamada previa al origen (si existe). Si la llamada previa al origen falla, la transferencia falla y no se ejecuta nada más.
3. Ejecute las salidas de inicio del destino. Si las salidas de inicio del destino fallan, la transferencia falla y no se ejecuta nada más.
4. Ejecute la llamada previa al destino (si existe). Si la llamada previa al destino falla, la transferencia falla y no se ejecuta nada más.
5. Realice las transferencias de archivos.
6. Ejecute las salidas de finalización del destino. No existe ningún estado de anomalía para estas salidas.
7. Si la transferencia es satisfactoria (si algunos archivos se transfieren satisfactoriamente, la transferencia se considera satisfactoria), ejecute la llamada posterior al destino (si está presente). Si la llamada posterior al destino falla, la transferencia falla.
8. Ejecute las salidas de finalización del origen. No existe ningún estado de anomalía para estas salidas.
9. Si la transferencia es satisfactoria, ejecute la llamada posterior al origen (si está presente). Si la llamada posterior al origen falla, la transferencia falla.

Ejemplos

En este ejemplo se muestra una operación básica de movimiento de archivos entre `agent1` y `agent2`. El mandato para iniciar el movimiento de archivos se envía a un gestor de colas denominado `qm0`, utilizando

una conexión de modalidad de transporte. El resultado de la operación de transferencia de archivos se asigna a la propiedad denominada `move.result`.

```
<fte:filemove cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
               src="agent1@qm1" dst="agent2@qm2"
               rcproperty="move.result">

  <fte:filespec srcfilespec="/home/fteuser1/file.bin" dstfile="/home/fteuser2/file.bin"/>

</fte:filemove>
```

Conceptos relacionados

[Opción de tiempo de espera para transferencias de archivos en la recuperación](#)

Tareas relacionadas

[Utilización de Apache Ant con MFT](#)

Tarea fte: ignoreoutcome Ant

Ignore el resultado de un mandato **fte:filecopy**, **fte:filemove** o **fte:call**. Cuando especifica una tarea **fte:filecopy**, **fte:filemove** o **fte:call** para tener un resultado de defer, la tarea Ant asigna recursos para realizar el seguimiento de este resultado. Si ya no está interesado en el resultado, puede utilizar la tarea **fte:ignoreoutcome** para liberar esos recursos.

Atributos

id

Necesario. Identifica el resultado que ha perdido su interés. Normalmente se especifica este identificador utilizando una propiedad que se establece utilizando el atributo `idproperty` de la tarea [“Tarea Ant fte:filecopy” en la página 2147](#), [“Tarea Ant fte:filemove” en la página 2151](#) o [“Tarea fte:call Ant” en la página 2144](#).

Ejemplo

En este ejemplo se muestra cómo puede utilizar la tarea `fte:ignoreoutcome` para liberar los recursos asignados al seguimiento del resultado de la tarea [“Tarea Ant fte:filecopy” en la página 2147](#) anterior.

```
<!-- issue a file copy request -->
<fte:filecopy cmdqm="qm1@localhost@1414@SYSTEM.DEF.SVRCONN"
               src="agent1@qm1" dst="agent1@qm1"
               idproperty="copy.id"
               outcome="defer"/>

<!-- do some other things -->

<!-- decide that the result of the copy is not interesting -->
<fte:ignoreoutcome id="${copy.id}"/>
```

Tareas relacionadas

[Utilización de Apache Ant con MFT](#)

Tarea fte: ping Ant

Esta tarea de IBM MQ Managed File Transfer Ant hace ping a un agente para obtener una respuesta y, por lo tanto, determina si el agente puede procesar transferencias.

Nota: IBM WebSphere MQ File Transfer Edition (FTE) ya no es un producto compatible. Para migrar desde FTE al componente Managed File Transfer en IBM MQ, consulte [Migración de Managed File Transfer](#).

Atributos

agent

Necesario. Especifica el agente al que se somete la solicitud **fte:ping**. El valor tiene el formato: `agentname@qmgrname` donde `agentname` es el nombre del agente y `qmgrname` es el nombre del gestor de colas al que está conectado directamente este agente.

cmdqm

Opcional. El gestor de colas de mandatos al que se somete la solicitud. Especifique esta información en el formato *qmgrname@host@port@channel*, donde:

- *qmgrname* es el nombre del gestor de colas
- *host* es el nombre de host opcional del sistema donde se está ejecutando el gestor de colas
- *port* es el número de puerto opcional en el que escucha el gestor de colas
- *channel* es el canal SVRCONN opcional a utilizar

Si omite la información de *host*, *port* o *channel* para el gestor de colas de mandatos, se utiliza la información de conexión especificada en el archivo `command.properties`.



Atención: Si no se especifica ningún valor para:

- variable *host*, se utiliza la modalidad de enlaces
- *port*, se utiliza el valor 1414
- *channel*, la variable `SYSTEM.DEF.SVRCONN`.

Consulte [El archivo `command.properties` de MFT](#) para obtener más información.

Sin embargo, no puede omitir los atributos en el medio, por ejemplo, *qmgrname@host@@channel*. Puede tener, por ejemplo, *qmgrname@host*, o *qmgrname@host@port*, o *qmgrname@hostport@@channel*.

MFT divide el atributo especificado utilizando el delimitador @. En función del número de señales encontradas, toma la primera señal como *qmgrname*, la segunda como *host*, la tercera como *puerto* y finalmente *canal*.

Para obtener más información, consulte [El archivo MFT `command.properties`](#).

Puede utilizar la propiedad **`com.ibm.wmqfte.propertySet`** para especificar qué archivo `command.properties` se debe utilizar. Para obtener más información, consulte [com.ibm.wmqfte.propertySet](#).

Si no utiliza el atributo `cmdqm`, la tarea utiliza de forma predeterminada la propiedad `com.ibm.wmqfte.ant.commandQueueManager`, si se ha establecido esta propiedad. Si la propiedad `com.ibm.wmqfte.ant.commandQueueManager` no está establecida, se intenta una conexión con el gestor de colas predeterminado, definido en el archivo `command.properties`. El formato de la propiedad `com.ibm.wmqfte.ant.commandQueueManager` es el mismo que el atributo `cmdqm`, es decir, *qmgrname@host@port@channel*.

rcproperty

Necesario. Nombra una propiedad para almacenar el código de retorno de la tarea **ping**.

timeout

Opcional. El período máximo de tiempo, en segundos, que la tarea espera a que el agente responda. El tiempo de espera mínimo es cero segundos, aunque también se puede especificar un tiempo de espera de menos uno; de este modo, el mandato espera de forma indefinida a que el agente responda. Si no se especifica ningún valor para `timeout`, el valor predeterminado es esperar hasta cinco segundos a que el agente responda.

Ejemplo

Este ejemplo envía una solicitud **fte:ping** al agente `agent1` alojado en `qm1`. La solicitud **fte:ping** espera 15 segundos a que el agente responda. El resultado de la solicitud **fte:ping** se almacena en una propiedad llamada `ping.rc`.

```
<fte:ping agent="agent1@qm1" rcproperty="ping.rc" timeout="15"/>
```

Códigos de retorno

0

El mandato se ha completado satisfactoriamente.

Se ha excedido el tiempo de espera del mandato.

Tareas relacionadas

[Utilización de Apache Ant con MFT](#)

Tarea fte: uuid Ant

Genera un identificador exclusivo pseudoaleatorio y lo asigna a una propiedad determinada. Por ejemplo, puede utilizar este identificador para generar nombres de trabajo para otras operaciones de transferencia de archivos.

Atributos

longitud

Necesario. La longitud numérica del UUID que hay que generar. Este valor de longitud no incluye la longitud de cualquier prefijo, que especifica el parámetro **prefix**.

property

Necesario. El nombre de la propiedad a la que asignar el UUID generado.

prefix

Opcional. Un prefijo para añadir al UUID generado. Este prefijo no cuenta como parte de la longitud del UUID, tal como lo especifica el parámetro **length**.

Ejemplo

Este ejemplo define un UUID que empieza por las letras ABC seguido de 16 caracteres hexadecimales pseudoaleatorios. El UUID se asigna a una propiedad denominada `uuid.property`.

```
<fte:uuid length="16" property="uuid.property" prefix="ABC"/>
```

Tareas relacionadas

[Utilización de Apache Ant con MFT](#)

Elemento anidado fte: filespec Ant

El parámetro **fte:filespec** se utiliza como elemento anidado en otras tareas. Utilice **fte:filespec** para describir una correlación entre uno o varios archivos, directorios  o conjuntos de datos de origen y un destino. Normalmente este elemento se utiliza cuando se expresa un conjunto de archivos o directorios  o conjuntos de datos para mover o para copiar.

Anidado por:

- La tarea [fte:filecopy](#)
- La tarea [fte:filemove](#)

Atributos de especificación de origen

Debe especificar uno de los valores `srcfilespec` o `srcqueue`.

srcfilespec

Especifica el origen de la operación de archivo. El valor de este atributo puede incluir un comodín.

srcqueue

Especifica que el origen de la transferencia es una cola. La transferencia mueve datos desde los mensajes almacenados en la cola especificada por este atributo. No puede especificar este atributo si la tarea **fte:filespec** está anidada dentro de la tarea **fte:filecopy**.

El atributo `srcqueue` no está soportado cuando el agente de origen es un agente de puente de protocolo.

Atributos de especificación de destino

Debe especificar una de las palabras clave `dstdir`, `dstds`, `dstfilespace`, `dstfile`, `dstqueue` o `dstpds`.

dstdir

Especifica un directorio como destino de una operación de archivo.

dstds

Especifica un conjunto de datos como destino de una operación de archivo.

Este atributo solo está soportado cuando el agente de destino se ejecuta en la plataforma z/OS.

dstfile

Especifica un archivo como destino de una operación de archivo.

dstfilespace

Especifica un espacio de archivos como destino para una operación de archivos.

Este atributo sólo se aplica si el agente de destino es un agente web de IBM MQ 8.0 que tiene acceso al espacio de archivos de la pasarela web.

dstpds

Especifica un conjunto de datos particionados como destino de una operación de archivo.

Este atributo solo está soportado cuando el agente de destino se ejecuta en la plataforma z/OS.

dstqueue

Especifica una cola como destino de una operación de un archivo a un mensaje. Puede incluir, si lo desea, un nombre de gestor de colas en esta especificación, utilizando el formato `QUEUE@QUEUEMANAGER`. Si no especifica un nombre de gestor de colas, se utiliza el gestor de colas de agente de destino si no ha establecido la propiedad de agente `enableClusterQueueInputOutput` en `true` (verdadero). Si la propiedad `enableClusterQueueInputOutput` se establece en `true`, el agente de destino utiliza procedimientos de IBM MQ estándares para determinar dónde está ubicada la cola. Debe especificar un nombre de cola válido que exista en el gestor de colas.

Si especifica el atributo `dstqueue`, no puede especificar los atributos `srcqueue` porque estos atributos son mutuamente excluyentes.

El atributo `dstqueue` no está soportado cuando el agente de destino es un agente de puente de protocolo.

Atributos de opciones de origen

srcencoding

Opcional. La codificación del juego de caracteres que el archivo utiliza para realizar transferencias.

Sólo se puede especificar este atributo cuando el atributo `conversion` tiene un valor de `text`.

Si no se especifica el atributo `srcencoding`, se utiliza el juego de caracteres del sistema de origen para las transferencias de texto.

srceol

Opcional. El delimitador de fin de la línea utilizado por el archivo que se está transfiriendo. Los valores válidos son los siguientes:

- CRLF - Utilizar un carácter de retorno de carro seguido por un carácter de salto de línea como delimitador de fin de la línea. Este convenio es típico de sistemas Windows.
- LF - Utilizar un carácter de salto de línea como delimitador de fin de la línea. Este convenio es típico de sistemas UNIX.

Puede especificar este atributo únicamente cuando el atributo `conversion` se establece en un valor de `text`. Si no se especifica el atributo `srceol`, las transferencias de texto determinan automáticamente el valor correcto basado en el sistema operativo del agente de origen.

srckeeptrailingspaces

Opcional. Determina si los espacios finales se conservan en los registros de origen leídos desde un conjunto de datos de formato de longitud fija como parte de una transferencia en modalidad de texto. Los valores válidos son los siguientes:

- **true** - los espacios finales se conservan.
- **false** - los espacios finales se eliminan.

Si no especifica el atributo **srckeeptrailingspaces**, se especifica un valor predeterminado de **false**.

Sólo puede especificar este atributo si también especifica el atributo **srcfilespec** y establece el atributo **conversion** en un valor de **text**.

srcmsgdelimbytes

Opcional. Especifica uno o más valores de byte que insertar como delimitador al añadir varios mensajes un archivo de binario. Cada valor se debe especificar como dos dígitos hexadecimales en el rango 00-FF, con el prefijo **x**. Si se especifican varios bytes, se deben separar con comas. Por ejemplo, **srcmsgdelimbytes="x08,xA4"**. Sólo puede especificar el atributo **srcmsgdelimbytes** si también ha especificado el atributo **srcqueue**. No puede especificar el atributo **srcmsgdelimbytes** si también ha especificado el valor **text** para el atributo **conversion**.

srcmsgdelimtext

Opcional. Especifica una secuencia de texto que insertar como delimitador al añadir varios mensajes a un archivo de texto. Se pueden incluir secuencias de escape Java en los literales String del delimitador. Por ejemplo, **srcmsgdelimtext="\u007d\n"**. El delimitador de texto es insertado después de cada mensaje por el agente de origen. El delimitador de texto se cifra en formato binario utilizando el cifrado de origen de la transferencia. Cada mensaje se lee en formato binario, el delimitador cifrado se añade en formato binario al mensaje y el resultado se transfiere en formato binario al agente de destino. Si la página de códigos del agente de origen incluye estados de desplazamiento a teclado estándar y desplazamiento desde teclado estándar, el agente presupone que cada mensaje está en el estado de desplazamiento desde teclado estándar al final del mensaje. En el agente de destino, los datos binarios se convierten de la misma manera que una transferencia de texto de archivo a archivo. También puede especificar el atributo **srcmsgdelimtext** si ha especificado también el atributo **srcqueue** y un valor de **text** para el atributo **conversion**.

srcmsgdelimposition

Opcional. Especifica la posición en la que se inserta el delimitador de texto o binario. Los valores válidos son los siguientes:

- **prefix** - los delimitadores se insertan en el archivo de destino antes de los datos de cada mensaje.
- **postfix** - los delimitadores se insertan en el archivo de destino después de los datos de cada mensaje.

Sólo puede especificar el atributo **srcmsgdelimposition** si también ha especificado uno de los atributos **srcmsgdelimbytes** o **srcmsgdelimtext**.

srcmsggroups

Opcional. Especifica que los mensajes sean agrupados por ID de grupo de IBM MQ. El primer grupo completo se escribe en el archivo de destino. Si no se especifica este atributo, se escribirán todos los mensajes en la cola de origen en el archivo de destino. Sólo puede especificar el atributo **srcmsggroups** si ha especificado también el atributo **srcqueue**.

srcqueuetimeout

Opcional. Especifica el tiempo, en segundos, que esperar hasta que se cumpla una de las siguientes condiciones:

- Para que un nuevo mensaje se grabe en la cola.
- Si se ha especificado el atributo **srcmsggroups**, para que un grupo completo se grabe en la cola.

Si no se cumple ninguna de estas condiciones dentro del tiempo especificado por el valor de **srcqueuetimeout**, el agente de origen deja de leer de la cola y completa la transferencia. Si no se especifica el atributo **srcqueuetimeout**, el agente de origen deja de leer inmediatamente de la cola de origen si la cola de origen está vacía o, en el caso de haber especificado el atributo **srcmsggroups**, si

no hay ningún grupo completo en la cola. Puede especificar el atributo *srcqueuetimeout* únicamente si también ha especificado el atributo *srcqueue*.

Para obtener información sobre cómo establecer el valor *srcqueuetimeout*, consulte [Orientación para especificar un tiempo de espera en una transferencia de mensaje a archivo](#).

srcrcdelimbytes

Opcional. Especifica uno o más valores de byte para insertar como el delimitador al añadir varios registros de un archivo de origen orientado a registros a un archivo binario. Debe especificar cada valor como dos dígitos hexadecimales en el rango 00-FF, con el prefijo x. Si se especifican varios bytes, se deben separar con comas. Por ejemplo:

```
srcrcdelimbytes="x08,xA4"
```

Solo se puede especificar el atributo *srcrcdelimbytes* si el archivo de origen de la transferencia es un archivo orientado a registro como, por ejemplo, un conjunto de datos de z/OS, y el archivo de destino es un archivo normal no orientado a registro. No puede especificar el atributo *srcrcdelimbytes* si también ha especificado el valor *text* para el atributo *conversion*.

srcrcdelimpos

Opcional. Especifica la posición en la que se inserta el delimitador binario. Los valores válidos son los siguientes:

- *prefix* - los delimitadores se insertan en el archivo de destino antes de los datos de cada registro de archivo de origen orientado a registros.
- *postfix* - los delimitadores se insertan en el archivo de destino después de los datos de cada registro de archivo de origen orientado a registros.

Puede especificar el atributo *srcrcdelimpos* sólo si también ha especificado el atributo *srcrcdelimbytes*.

Atributos de opciones de destino

dstencoding

Opcional. La codificación de juego de caracteres que se utiliza para el archivo transferido.

Sólo se puede especificar este atributo cuando el atributo *conversion* tiene un valor de *text*.

Si no se especifica el atributo *dstencoding*, se utiliza el juego de caracteres del sistema de destino para las transferencias de texto.

dsteol

Opcional. El delimitador de fin de la línea que se utiliza para el archivo transferido. Los valores válidos son los siguientes:

- *CRLF* - Utilizar un carácter de retorno de carro seguido por un carácter de salto de línea como delimitador de fin de la línea. Este convenio es típico de sistemas Windows.
- *LF* - Utilizar un carácter de salto de línea como delimitador de fin de la línea. Este convenio es típico de sistemas UNIX.

Sólo se puede especificar este atributo cuando el atributo *conversion* tiene un valor de *text*.

Si no se especifica el atributo *dsteol*, las transferencias de texto determinan automáticamente el valor correcto basado en el sistema operativo del agente de destino.

dstmsgdelimbytes

Opcional. Especifica el delimitador hexadecimal que utilizar al dividir un archivo binario en varios mensajes. Todos los mensajes tienen el mismo ID de grupo de IBM MQ; el último mensaje del grupo tiene el distintivo de IBM MQ *LAST_MSG_IN_GROUP* definido. El formato para especificar un byte hexadecimal como delimitador es *xNN*, donde N es un carácter en el rango entre 0 y 9 o a y f. Puede especificar una secuencia de bytes hexadecimales como delimitador especificando una lista separada por comas de bytes hexadecimal, por ejemplo: *x3e,x20,x20,xbf*.

Sólo puede especificar el atributo `dstmsgdelimbytes` si también ha especificado el atributo `dstqueue` y la transferencia está en modalidad binaria. Sólo puede especificar uno de los atributos `dstmsgsize`, `dstmsgdelimbytes` y `dstmsgdelimpattern`.

dstmsgdelimpattern

Opcional. Especifica la expresión regular Java que se usa al dividir un archivo de texto en varios mensajes. Todos los mensajes tienen el mismo ID de grupo de IBM MQ; el último mensaje del grupo tiene el distintivo de IBM MQ `LAST_MSG_IN_GROUP` definido. El formato para especificar una expresión regular como delimitador es una expresión regular entre paréntesis, (*regular_expression*), o entre comillas dobles, "*regular_expression*". Si desea más información, consulte [Expresiones regulares utilizadas por MFT](#).

De forma predeterminada, la longitud de la serie que la expresión regular puede encontrar está limitada por el agente de destino a cinco caracteres. Puede cambiar este comportamiento utilizando la propiedad de agente **maxDelimiterMatchLength**. Para obtener más información, consulte [Propiedades avanzadas de agente MFT](#).

Sólo puede especificar el atributo `dstmsgdelimpattern` si también ha especificado el atributo `dstqueue` y la transferencia está en modalidad de texto. Sólo puede especificar uno de los atributos `dstmsgsize`, `dstmsgdelimbytes` y `dstmsgdelimpattern`.

dstmsgdelimposition

Opcional. Especifica la posición en la que está previsto que esté el delimitador de texto o binario. Los valores válidos son los siguientes:

- `prefix` - Los delimitadores se esperan al principio de cada línea.
- `postfix` - Los delimitadores se esperan al final de cada línea.

Sólo puede especificar el atributo `dstmsgdelimposition` si ha especificado también el atributo `dstmsgdelimpattern`.

dstmsgincludedelim

Opcional. Especifica si desea incluir en los mensajes el delimitador que se utiliza para dividir el archivo en varios mensajes. Si se especifica el atributo `dstmsgincludedelim`, el delimitador se incluye al final del mensaje que contiene los datos de archivo que preceden al delimitador. De forma predeterminada, el delimitador no se incluye en los mensajes. Sólo puede especificar el atributo `dstmsgincludedelim` si también ha especificado uno de los atributos `dstmsgdelimpattern` y `dstmsgdelimbytes`.

dstmsgpersist

Opcional. Especifica si los mensajes escritos en la cola de destino son persistentes. Los valores válidos son los siguientes:

- `true` - Escribir mensajes persistentes en la cola de destino. Éste es el valor predeterminado.
- `false` - Escribir mensajes no persistentes en la cola de destino.
- `qdef` - El valor de persistencia se toma del atributo `DefPersistence` de la cola de destino.

Puede especificar este atributo únicamente cuando también se especifica el atributo `dstqueue`.

dstmsgprops

Opcional. Especifica si el primer mensaje escrito en la cola de destino tiene la propiedad de mensaje de IBM MQ definida. Los valores posibles son:

- `true` - Definir propiedades de mensajes en el primer mensaje creado por la transferencia.
- `false` - No definir propiedades de mensajes en el primer mensaje creado por la transferencia. Éste es el valor predeterminado.

Para obtener más información, consulte [Propiedades de mensajes deMQ establecidas por MFT en mensajes escritos en colas de destino](#).

Puede especificar este atributo únicamente cuando también se especifica el atributo `dstqueue`.

dstmsgsize

Opcional. Especifica si dividir el archivo en varios mensajes de longitud fija. Todos los mensajes tienen el mismo ID de grupo de IBM MQ; el último mensaje del grupo tiene el distintivo de IBM MQ `LAST_MSG_IN_GROUP` definido. El tamaño de los mensajes se especifica por el valor de `dstmsgsize`. El formato de `dstmsgsize` es *longitudunidades*, where *longitud* es un valor de entero positivo y *unidades* es uno de los valores siguientes:

- B - Bytes. El valor mínimo permitido es dos veces el valor máximo de bytes por carácter de la página de códigos de los mensajes de destino.
- K - Kibibytes. Esto equivale a 1024 bytes.
- M - Mebibytes. El equivalente a 1024 kibibytes.

Si el archivo se transfiere en modalidad de texto, y está en un juego de caracteres de doble byte o un juego de caracteres de varios bytes, el archivo se divide en mensajes en el límite de caracteres más cercano al tamaño de mensaje especificado.

Sólo puede especificar el atributo `dstmsgsize` si ha especificado también el atributo `dstqueue`. Sólo puede especificar uno de los atributos `dstmsgsize`, `dstmsgdelimbytes` y `dstmsgdelimpattern`.

dstunsupportedcodepage

Opcional. Especifica la acción que se va a emprender si el gestor de colas de destino, tal como se especifica en el atributo `dstqueue`, no da soporte a la página de códigos utilizada cuando se transfieren datos de archivos a una cola como una transferencia de texto. Los valores válidos para este atributo son los siguientes:

- `binary` - continuar la transferencia pero no aplicar la conversión de página de códigos a los datos que se están transfiriendo. Especificar este valor equivale a no establecer el atributo de conversión en `text`.
- `fail` - no continuar con la operación de transferencia. El archivo se graba como si no hubiera conseguido transferirse. Este es el valor predeterminado.

Sólo puede especificar el atributo `dstunsupportedcodepage` si también ha especificado el atributo `dstqueue` y un valor de `text` para el atributo `conversion`.

dsttruncaterecords

Opcional. Especifica que los registros de destino que sean mayores que el atributo de conjunto de datos `LRECL` se truncan. Si se establece en `true`, los registros se truncan. Si se establece en `false`, los registros se ajustan automáticamente. El valor predeterminado es `false`. Este parámetro sólo es válido para transferencias en modalidad de texto, en las que el destino sea un conjunto de datos.

Otros atributos

checksum

Opcional. Determina el algoritmo que se utiliza para calcular la suma de comprobación de los archivos transferidos.

- MD5 - utilizar el algoritmo de hash MD5.
- NONE - no utilizar un algoritmo de suma de comprobación.

Si no especifica el atributo `checksum`, se utiliza un valor predeterminado de MD5.

conversion

Opcional. Especifica el tipo de conversión que se aplica al archivo cuando se está transfiriendo. Los valores posibles son:

- `binary` - no aplicar ninguna conversión.
- `text` - aplicar la conversión de código de páginas entre los sistemas de origen y destino. Aplicar también la conversión de delimitadores de línea. Los atributos `srcencoding`, `dstencoding`, `srceol` y `dsteol` influyen sobre la conversión que se aplica.

Si no especifica el atributo `conversion`, se especifica un valor predeterminado de `binary`.

overwrite

Opcional. Determina si la operación puede sobrescribir un  conjunto de datos o un archivo de destino existente. Cuando se especifica un valor de `true`, se sobrescriben los  conjuntos de datos o el archivo de destino existentes. Cuando se especifica un valor de `false`, la existencia de un archivo  o un conjunto de datos duplicado en los resultados en el destino hace que la operación falle. Si no se especifica el atributo `overwrite`, se especifica un valor predeterminado de `false`.

recurse

Opcional. Determina si la transferencia de archivos se repite en los subdirectorios. Cuando se especifica un valor de `true`, la transferencia se repite en los subdirectorios. Cuando se especifica un valor de `false`, la transferencia no se repite en los subdirectorios. Si no se especifica el atributo `recurse`, se especifica un valor predeterminado de `false`.

Ejemplo

Este ejemplo especifica un `fte:filespec` con un archivo de origen de `file1.bin` y un archivo de destino de `file2.bin`.

```
<fte:filespec srcfilespec="/home/fteuser/file1.bin" dstfile="/home/fteuser/file2.bin"/>
```

Tareas relacionadas

[Utilización de Apache Ant con MFT](#)

fte:metadata Ant elementos anidados

Los metadatos contienen información adicional definida por el usuario, especialmente en tareas como **fte:call**, que pueden no implicar una transferencia de archivos.

Consulte “Metadatos para salidas de usuario de MFT” en la [página 2168](#) para obtener más información sobre cómo Managed File Transfer utiliza los metadatos con las salidas de usuario MFT.

Anidado por:

- La tarea [fte:filecopy](#)
- La tarea [fte:filemove](#)
- La tarea [fte:call](#)

Parámetros especificados como elementos anidados

fte:entry

Debe especificar al menos una entrada dentro del `fte:metadata` elemento anidado. Puede optar por especificar más de una entrada. Las entradas asocian un nombre de clave a un valor. Las claves deben ser únicas en un bloque de `fte:metadata`.

Atributos de entrada

name

Necesario. El nombre de la clave que pertenece a esta entrada. Este nombre debe ser exclusivo para todos los parámetros anidados **entry** dentro de un elemento `fte:metadata`.

valor

Necesario. El valor que se asigna a esta entrada.

Ejemplo

Este ejemplo muestra una `fte:metadata` definición que contiene dos entradas.

```
<fte:metadata>  
  <fte:entry name="org.foo.partColor" value="red"/>
```

```
<fte:entry name="org.foo.partSize" value="medium"/>
</fte:metadata>
```

Ejemplo

Este ejemplo muestra el uso de metadatos dentro de un fte:call task. Esta tarea no realiza una transferencia de archivos, pero aún así puede transportar metadatos.

```
<fte:call
  agent="1@MFTQM"
  idproperty="doCall.id"
  rcproperty="doCall.rc">

  <fte:command command="foo" />

  <fte:metadata>
    <fte:entry name="org.foo.item.color" value="red"/>
    <fte:entry name="org.foo.item.size" value="medium"/>
  </fte:metadata>

</fte:call>
```

foo.cmd Archivo

```
@echo off
echo Program: %~dpnx0
echo Parameters: %*
echo Environment:
set org.foo
exit /b 0
```

Salida estándar generada por foo.cmd, utilizando IBM MQ Explorer

```
Program: C:\IBM\MQdata\mqft\Commands\foo.cmd
Parameters:
Environment:
org.foo.item.color=red
org.foo.item.size=medium
```

Tareas relacionadas

[Utilización de Apache Ant con MFT](#)

Elementos anidados de invocación de programa

Se pueden iniciar programas utilizando uno de cinco elementos anidados: **fte:presrc**, **fte:predst**, **fte:postdst**, **fte:postsrc** y **fte:command**. Estos elementos anidados ordenan a un agente que llame a un programa externo como parte del proceso. Para poder iniciar un programa, antes debe asegurarse de que el mandato está en una ubicación especificada por la propiedad **commandPath** en el archivo **agent.properties** del agente que ejecute el mandato.

Aunque cada elemento de invocación de programa tiene un nombre diferente, comparten el mismo conjunto de atributos y el mismo conjunto de elementos anidados. Los programas se pueden iniciar mediante las tareas Ant **fte:filecopy**, **fte:filemove** y **fte:command**.

No puede invocar programas desde un agente de puente Connect:Direct.

Tareas Ant que pueden invocar programas:

- La tarea **fte:filecopy** anida parámetros de invocación de programa mediante los elementos anidados **fte:predst**, **fte:postdst**, **fte:presrc** y **fte:postsrc**.
- La tarea **fte:filemove** anida parámetros de invocación de programa mediante los elementos anidados **fte:predst**, **fte:postdst**, **fte:presrc** y **fte:postsrc**.
- La tarea **fte:call** anida parámetros de invocación de programa mediante el elemento anidado **fte:command**.

Atributos

En una ventana de comandos, vaya al directorio donde está instalado . [*5737H41 From a command window, go to the directory where is installed.*]

Necesario. Nombra el programa que se invoca. Para que el agente pueda ejecutar un mandato, el mandato debe encontrarse en una ubicación especificada por la propiedad `commandPath` en el archivo `agent.properties` del agente. Para obtener más información, consulte la propiedad `MFTcommandPath`. La información sobre la vía de acceso especificada en el atributo `command` se considera relativa a una ubicación especificada por la propiedad `commandPath`. Cuando `type` es `executable`, se espera un programa ejecutable; en caso contrario, se espera un script apropiado al tipo de llamada.

retrycount

Opcional. El número de veces que se reintenta llamar al programa si el programa no devuelve un código de retorno satisfactorio. El programa especificado por el atributo `command` se invoca como máximo este número de veces. El valor asignado a este atributo debe ser un valor no negativo. Si no especifica el atributo `retrycount`, se utiliza un valor predeterminado de cero.

retrywait

Opcional. El tiempo de espera, en segundos, antes de intentar de nuevo la invocación de programa. Si el programa especificado por el atributo `command` no devuelve un código de retorno satisfactorio y el atributo `retrycount` especifica un valor distinto de cero, este parámetro determina el tiempo de espera entre reintentos. El valor asignado a este atributo debe ser un valor no negativo. Si no especifica el atributo `retrywait`, se utiliza un valor predeterminado de cero.

successrc

Opcional. El valor de este atributo se utiliza para determinar cuándo la invocación de programa se ejecuta satisfactoriamente. El código de retorno del proceso para el mandato se evalúa mediante esta expresión. El valor puede estar compuesto por una o más expresiones combinadas con un carácter de barra vertical (|) para indicar OR booleano, o un carácter de ampersand (&) para indicar AND booleano. Cada expresión puede ser uno de los tipos siguientes de expresión:

- Un número que indica una prueba de igualdad entre el código de retorno del proceso y el número.
- Un número precedido por el carácter ">" para indicar una comprobación de mayor que entre el número y el código de retorno del proceso.
- Un número precedido por el carácter "<" para indicar una comprobación menor que entre el número y el código de retorno del proceso.
- Un número que tiene como prefijo un carácter "!" para indicar una prueba de 'no igual a' entre el número y el código de retorno de proceso.

Por ejemplo: `>2&<7&!5|0|14` se interpreta como que los siguientes códigos de retorno son correctos: 0, 3, 4, 6, 14. Los demás códigos de retorno se interpretan como no satisfactorios. Si no especifica el atributo `successrc`, se utiliza un valor predeterminado de cero. Esto significa que se interpreta que el mandato se ha ejecutado satisfactoriamente si, y solamente si, dicho mandato devuelve un código de cero.

Tipo

Opcional. El valor de este atributo especifica qué tipo de programa se está invocando. Especifique una de las opciones siguientes:

executable

La tarea invoca un programa ejecutable. Puede tener argumentos adicionales especificados mediante el elemento anidado `arg`. Se espera que el programa sea accesible en `commandPath` y, en caso necesario, que tenga establecido el permiso de ejecución. UNIX Los scripts se pueden llamar siempre que especifiquen un programa de shell (por ejemplo, la primera línea del archivo de script de shell es: `#!/bin/sh`). La salida del comando escrita en `stderr` o `stdout` se envía al Managed File Transfer registro de la llamada. No obstante, la cantidad de salida de datos está limitada por la configuración del agente. El valor predeterminado es 10K bytes de datos, pero puede alterar temporalmente este valor predeterminado utilizando la propiedad de agente: `maxCommandOutput`.

antscript

La tarea ejecuta el script Ant especificado utilizando el **fteAnt** comando. Se pueden especificar propiedades mediante el elemento anidado **property**. Ant Los objetivos se pueden especificar utilizando el elemento anidado **target**. Se espera que el Ant guion esté disponible en **commandPath**. Ant La salida escrita en **stderr** o **stdout** se envía al Managed File Transfer registro de la llamada. No obstante, la cantidad de salida de datos está limitada por la configuración del agente. El valor predeterminado es 10K bytes de datos, pero puede alterar temporalmente este valor mediante la propiedad de agente: **maxCommandOutput**.

jcl

El valor **jcl** solo está soportado en z/OS y ejecuta el script JCL de z/OS especificado. El JCL se somete como un trabajo y requiere que haya una tarjeta de trabajo. Cuando el trabajo se somete satisfactoriamente, la salida del mandato JCL, grabada en el registro de Managed File Transfer, contiene el siguiente texto: **JOB nombre_trabajo(id_trabajo)**, donde:

- **nombre_trabajo** es el nombre del trabajo identificado por la tarjeta de trabajo del JCL.
- **id_trabajo** es el identificador de trabajo generado por el sistema z/OS.

Si el trabajo no se puede someter satisfactoriamente, el mandato de script JCL no se ejecuta correctamente y graba un mensaje en el registro indicando la razón del error (por ejemplo, no hay ninguna tarjeta de trabajo). Para saber si el trabajo se ha ejecutado o se ha completado satisfactoriamente, utilice un servicio del sistema, como, por ejemplo, SDSF. Managed File Transfer no proporciona esta información porque sólo somete el trabajo; a continuación, el sistema determina cuándo debe ejecutarse el trabajo y cómo se presenta la salida del trabajo. Puesto que un script JCL se somete como un trabajo por lotes, no es aconsejable especificar **jcl** para un elemento anidado **presrc** o **predst**, porque lo único que sabe es que el trabajo se ha sometido satisfactoriamente pero no si se ha ejecutado satisfactoriamente hasta el final antes de que se inicie la transferencia. No hay ningún elemento anidado que sea válido con un tipo de **jcl**.

El siguiente ejemplo muestra un trabajo de JCL:

```
//MYJOB JOB
//*
//MYJOB EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=H
//SYSUT1 DD DSN=FRED.DEMO.TXT,DISP=SHR
//SYSUT2 DD DSN=BOB.DEMO.TXT,DISP=(NEW,CATLG),
// RECFM=VB,LRECL=133,BLKSIZE=2048,
// SPACE=(TRK,(30,5),RLSE)
//SYSIN DD DUMMY
```

Parámetros especificados como elementos anidados

fte:arg

Sólo es válido cuando el valor del atributo **type** es **executable**. Utilice elementos **fte:arg** anidados para especificar argumentos para el programa que se está invocando como parte de la invocación de programa. Los argumentos del programa se crean a partir de los valores especificados por los elementos **fte:arg** en el orden en que se encuentran los elementos **fte:arg**. Si lo desea, puede especificar cero o más elementos **fte:arg** como elementos anidados de una invocación de programa.

fte:property

Sólo es válido cuando el valor del atributo **type** es **antscript**. Utilice los name atributos **value** y de los elementos **fte:property** anidados para pasar pares nombre-valor al Ant script. Puede optar por especificar cero o más **property** elementos como elementos anidados de una invocación de programa.

fte:target

Sólo es válido cuando el valor del atributo **type** es **antscript**. Especifique un destino en el Ant script que se va a llamar. Si lo desea, puede especificar cero o más elementos **fte:target** como elementos anidados de una invocación de programa.

Atributos de Arg

valor

Necesario. El valor del argumento que se pasa al programa que se está invocando.

Atributos de Property

nombre

Necesario. El nombre de una propiedad que se pasará al Ant script.

valor

Necesario. El valor que se asociará con el nombre de la propiedad que se pasa al Ant script.

Ejemplos

Este ejemplo muestra una invocación de programa `fte:postsrc` que se especifica como parte de una tarea `fte:filecopy`. La invocación de programa es para un programa denominado `post.sh` y se le suministra un único argumento de `/home/fteuser2/file.bin`.

```
<fte:filecopy
cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
src="agent1@qm1" dst="agent2@qm2"
rcproperty="copy.result">
<fte:filespec srcfilespec="/home/fteuser1/file.bin" dstfile="/home/fteuser2/file.bin"/>
<fte:postsrc command="post.sh" successsrc="1" >
  <fte:arg value="/home/fteuser2/file.bin"/>
</fte:postsrc>
</fte:filecopy>
```

Este ejemplo muestra una invocación de programa `fte:command` que se especifica como parte de una tarea `fte:call`. La invocación de programa es para un archivo ejecutable llamado `command.sh`, al que no se pasa ningún argumento de línea de mandatos. Si `command.sh` no devuelve un código de retorno satisfactorio de 1, se intenta ejecutar de nuevo el mandato al cabo de 30 segundos.

```
<fte:call
cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
agent="agent1@qm1"
rcproperty="call.rc"
origuser="bob"
jobname="${job.id}">
<fte:command command="command.sh" successsrc="1" retrycount="5" retrywait="30"/>
</fte:call>
```

Este ejemplo muestra una invocación de programa `fte:command` que se especifica como parte de una tarea `fte:call`. La invocación del programa es para los objetivos de copia y compresión en un Ant script llamado `script.xml`, al que se le pasan dos propiedades.

```
<fte:call
cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
agent="agent1@qm1"
rcproperty="call.rc"
origuser="bob"
jobname="${job.id}">
<fte:command command="script.xml" type="antscript">
  <property name="src" value="AGENT5@QM5"/>
  <property name="dst" value="AGENT3@QM3"/>
  <target name="copy"/>
  <target name="compress"/>
</fte:command>
</fte:call>
```

Tareas relacionadas

Especificación de programas que se van a ejecutarse con MFT

[Utilización de Apache Ant con MFT](#)

MFT Salidas de usuario para personalización de referencia

Información de referencia para ayudarle a configurar las salidas de usuario para Managed File Transfer.

Conceptos relacionados

Salidas de usuario de origen y destino de MFT

Metadatos para salidas de usuario de MFT

Hay tres tipos diferentes de metadatos que se pueden suministrar a las rutinas de salida para Managed File Transfer: entorno, transferencia y metadatos de archivo. Estos metadatos se presentan como mapas de pares de clave-valor Java.

Metadatos del entorno

Los metadatos del entorno se transfieren a todas las rutinas de salida de usuario y describen el entorno de ejecución del agente desde el que se invoca la rutina de salida de usuario. Estos metadatos son de sólo lectura y no se pueden actualizar mediante ninguna rutina de salida de usuario.

Tabla 918. Metadatos del entorno	
Clave	Descripción
AGENT_CONFIGURATION_DIRECTORY_KEY	El nombre del directorio que contiene la información de configuración del agente.
AGENT_PRODUCT_DIRECTORY_KEY	El nombre del directorio en el que se ha instalado el código del agente.
AGENT_VERSION_KEY	Número de versión para el tiempo de ejecución del agente que invoca la rutina de salida.

Los nombres de clave y valor suministrados en la Tabla 1 son constantes que se definen en la interfaz EnvironmentMetaDataConstants.

Metadatos de transferencia

Los metadatos de transferencia se transfieren a todas las rutinas de salida de usuario. Los metadatos constan de valores suministrados por el sistema y valores suministrados por el usuario. Si cambia cualquier valor proporcionado por el sistema, estos cambios serán ignorados. Los valores proporcionados por el usuario iniciales para la salida de usuario de inicio de transferencia de origen se basan en los valores proporcionados al definir la transferencia. El agente de origen puede cambiar los valores suministrados por el usuario como parte del proceso de la salida de usuario de inicio de la transferencia de origen. Esta rutina de salida se invoca antes de que se inicie toda la transferencia de archivos. Estos cambios se utilizan en llamadas posteriores a otras rutinas de salida que están relacionadas con esa transferencia. Los metadatos de transferencia se aplican a una transferencia entera.

Aunque las salidas de usuario pueden leer valores de los metadatos de transferencia, únicamente la salida de usuario de inicio de la transferencia de origen puede cambiar los metadatos de transferencia.

No puede utilizar metadatos de transferencia para difundir información entre transferencias de archivos diferentes.

Los metadatos de transferencia proporcionados por el sistema se detallan en la Tabla 2:

Tabla 919. Metadatos de transferencia	
Clave	Descripción
DESTINATION_AGENT_KEY	El nombre del agente que es el destino de la transferencia.
JOB_NAME_KEY	El nombre de trabajo asociado a la solicitud de transferencia
MQMD_USER_KEY	El campo de usuario MQMD del mensaje empleado para someter la solicitud de transferencia
ORIGINATING_HOST_KEY	El nombre de host especificado como el nombre de host de origen en la solicitud de transferencia

Tabla 919. Metadatos de transferencia (continuación)	
Clave	Descripción
ORIGINATING_USER_KEY	El nombre de usuario especificado como el ID de usuario de origen en la solicitud de transferencia
SOURCE_AGENT_KEY	El nombre del agente que es el origen de la transferencia
TRANSFER_ID_KEY	El identificador de la transferencia

Los nombres de clave y valor suministrados en la Tabla 2 son constantes que se definen en la interfaz TransferMetaDataConstants.

Metadatos de archivo

Los metadatos de archivo se transfieren a la salida del inicio de la transferencia de origen como parte de la especificación de archivo. Existen metadatos de archivo separados para los archivos de origen y destino.

No puede utilizar metadatos para difundir información entre transferencias de archivos diferentes.

Tabla 920. Metadatos de archivo		
Clave	Valores permitidos	Descripción
CONVERT_LINE_SEPARATORS		El valor de clave utilizado para las transferencias de texto para indicar si las secuencias de separador de línea CRLF (retorno de carro, salto de línea) o LF (salto de línea) en los datos de origen se convierten a la secuencia de separador de línea de separación en el destino.
DELIMITER_KEY		El valor de clave utilizado para definir un delimitador para separar datos de registro cuando se transfieren datos orientado a registros a archivos normales. También se utiliza para transferencias de mensaje a archivo y de archivo a mensaje.
DELIMITER_POSITION_KEY	DELIMITER_POSITION_PREFIX_VALUE DELIMITER_POSITION_POSTFIX_VALUE	Utilizar con DELIMITER_KEY para definir la posición del delimitador; prefijo o postfijo.
DELIMITER_TYPE_KEY	DELIMITER_TYPE_BINARY_VALUE DELIMITER_TYPE_TEXT_VALUE DELIMITER_TYPE_SIZE_VALUE	Utilice esta opción con DELIMITER_KEY para definir el tipo de delimitador.
DESTINATION_EXIST_KEY	DESTINATION_EXIST_KEY_ERROR_VALUE DESTINATION_EXIST_KEY_OVERWRITE_VALUE	Determina el comportamiento de la transferencia de archivos si el archivo de destino existe.
FILE_ALIAS_KEY		Valor de clave utilizado para definir un alias para el archivo que se está transfiriendo.
FILE_CHECKSUM_METHOD_KEY	FILE_CHECKSUM_METHOD_NONE_VALUE FILE_CHECKSUM_METHOD_MD5_VALUE	Determina el método de suma de comprobación que se utiliza cuando se transfiere el archivo.
FILE_CONVERSION_KEY	FILE_CONVERSION_TEXT_VALUE FILE_CONVERSION_BINARY_VALUE	Determina el tipo de conversión que se aplica al contenido del archivo.
FILE_ENCODING_KEY		Determina la codificación que se utiliza para un archivo de texto.

Tabla 920. Metadatos de archivo (continuación)

Clave	Valores permitidos	Descripción
FILE_END_OF_LINE_KEY	FILE_END_OF_LINE_LF_VALUE FILE_END_OF_LINE_CRLF_VALUE	Determina la secuencia de caracteres que indica el final de una línea: <LF> o <CR><LF>.
FILE_SPACE_ALIAS		Determina el alias de un archivo en el espacio de archivos. Nota: Estos metadatos pueden utilizarse sólo si FILE_TYPE_KEY es FILE_TYPE_FILE_SPACE_VALUE
FILE_SPACE_NAME		Determina el nombre del espacio de archivos. Nota: Estos metadatos pueden utilizarse sólo si FILE_TYPE_KEY es FILE_TYPE_FILE_SPACE_VALUE
FILE_TYPE_KEY	FILE_TYPE_FILE_VALUE FILE_TYPE_DIRECTORY_VALUE FILE_TYPE_DATASET_VALUE FILE_TYPE_PDS_VALUE FILE_TYPE_QUEUE_VALUE FILE_TYPE_FILE_SPACE_VALUE	Determina la especificación del archivo de destino, cola o espacio de archivos.
GROUP_ID_KEY		Valor de clave utilizado para transferencias de mensaje a archivo para determinar el grupo de mensajes a leer de la cola de origen. Este atributo sólo es válido cuando el valor de USE_GROUPS_KEY es USE_GROUPS_TRUE_VALUE.
INCLUDE_DELIMITER_IN_MESSAGE_KEY	INCLUDE_DELIMITER_IN_MESSAGE_TRUE_VALUE INCLUDE_DELIMITER_IN_MESSAGE_FALSE_VALUE	Valor de clave utilizado para transferencias de archivo a mensaje para determinar si se deben incluir los delimitadores que se han utilizado para dividir el archivo en varios mensajes al final de los mensajes. Este atributo sólo es válido cuando el valor de DELIMITER_TYPE_KEY es DELIMITER_TYPE_BINARY_VALUE DELIMITER_TYPE_TEXT_VALUE.
INSERT_RECORD_LINE_SEPARATOR_KEY		Valor de clave utilizado para las transferencias de texto desde archivos orientados a registros para especificar si se insertan separadores de línea en los datos después de cada registro.
KEEP_TRAILING_SPACES_KEY	KEEP_TRAILING_SPACES_TRUE_VALUE KEEP_TRAILING_SPACES_FALSE_VALUE	Valor de clave utilizado para determinar si los espacios de cola se eliminan de los registros leídos en los conjuntos de datos de formato de longitud fija.
NEW_RECORD_ON_LINE_SEPARATOR_KEY		Valor de clave utilizado para transferencias de texto a archivos orientados a registros para especificar si los separadores de línea en los datos se incluyen en los datos de registro o producen un nuevo registro (y no se graban).
PERSISTENT_KEY	PERSISTENT_TRUE_VALUE PERSISTENT_FALSE_VALUE PERSISTENT_QDEF_VALUE	Valor de clave utilizado para transferencias de archivo a mensaje para determinar si los mensajes son persistentes.

Tabla 920. Metadatos de archivo (continuación)		
Clave	Valores permitidos	Descripción
SET_MQ_PROPS_KEY	SET_MQ_PROPS_TRUE_VALUE SET_MQ_PROPS_FALSE_VALUE	Valor de clave utilizado para transferencias de archivo a mensaje para determinar si las propiedades de mensaje IBM MQ se establecen en el primer mensaje de un archivo y los mensajes se graban en la cola cuando se produce un error.
UNRECOGNISED_CODE_PAGE_KEY	UNRECOGNISED_CODE_PAGE_FAIL_VALUE UNRECOGNISED_CODE_PAGE_BINARY_VALUE	Valor de clave utilizado para transferencias de archivo a mensaje para determinar si falla una transferencia de modalidad de texto o se realiza una conversión, en el caso de que el gestor de colas de destino no reconozca la página de códigos de los datos.
USE_GROUPS_KEY	USE_GROUPS_TRUE_VALUE USE_GROUPS_FALSE_VALUE	Valor de clave utilizado para transferencias de mensaje a archivo para determinar si sólo se debe transferir un grupo completo de mensaje de la cola de origen.
WAIT_TIME_KEY		Valor de clave utilizado para transferencias de mensaje a archivo para determinar el tiempo, en segundos, que el agente de origen debe esperar a que se produzca uno de los casos siguientes: • Aparezca un mensaje en la cola de origen, si la cola está vacía o se ha quedado vacía, cuando el valor de USE_GROUPS_KEY es FALSE. • Aparezca un grupo completo en la cola de origen, si el valor de USE_GROUPS_KEY es TRUE.

Los nombres de clave y valor suministrados en la Tabla 3 son constantes que se definen en la interfaz FileMetaDataConstants.

Conceptos relacionados

[“Java interfaces para salidas MFT de usuario” en la página 2178](#)

Use los temas de esta sección para obtener información de referencia sobre las interfaces Java de las rutinas de salida de usuario.

Tareas relacionadas

[Personalización de MFT con salidas de usuario](#)

Referencia relacionada

[“Salidas de usuario del supervisor de recursos de MFT” en la página 2172](#)

Las salidas de usuario del supervisor de recursos permiten configurar que el código personalizado se ejecute cuando se cumple una condición desencadenante de un supervisor antes de que se inicie la tarea asociada.

[“Propiedades de agente MFT para salidas de usuario” en la página 2176](#)

Además de las propiedades estándar del archivo `agent.properties`, hay varias propiedades avanzadas específicas para las rutinas de salida de usuario. Estas propiedades no están incluidas de forma predeterminada; por tanto, si desea utilizar alguna de ellas, deberá editar manualmente el archivo `agent.properties`. Si realiza algún cambio en el archivo `agent.properties` mientras se ejecuta dicho agente, detenga y vuelva a iniciar el agente para recuperar los cambios.

Salidas de usuario del supervisor de recursos de MFT

Las salidas de usuario del supervisor de recursos permiten configurar que el código personalizado se ejecute cuando se cumple una condición desencadenante de un supervisor antes de que se inicie la tarea asociada.

No es recomendable invocar transferencias nuevas directamente desde el código de salida de usuario. En algunas circunstancias esto hace que los archivos se transfieren varias veces porque las salidas de usuario no son resistentes a los reinicios de agente.

Las salidas de usuario del supervisor de recursos utilizan la infraestructura existente para salidas de usuario. Las salidas de usuario de supervisor se invocan después de que se haya desencadenado un supervisor, pero antes de que se haya ejecutado la tarea correspondiente mediante la tarea del supervisor. Esto permite que la salida de usuario modifique la tarea que se va a ejecutar y decida si una tarea debe continuar o no. Puede modificar la tarea de supervisor actualizando los metadatos de supervisor, que a su vez se utilizan para la sustitución de variable en el documento de la tarea creado mediante la creación del supervisor original. O bien, la salida de supervisor puede sustituir o actualizar la serie XML de definición de tarea pasada como parámetro. La salida de supervisor puede devolver un código de resultado de 'proceed' (proseguir) o 'cancel' (cancelar) para la tarea. Si se devuelve un código de resultado de cancelación, la tarea no se iniciará y el supervisor no se volverá a iniciar hasta que el recurso supervisado coincida con las condiciones desencadenantes. Si el recurso no ha cambiado, el desencadenante no se iniciará. Al igual que sucede con las otras salidas de usuario, pueden encadenarse conjuntamente salidas de usuario. Si una de las salidas devuelve un código de resultado de cancelación, se cancela el resultado global y no se inicia la tarea.

- Una correlación de metadatos de entorno (lo mismo que las otras salidas de usuario)
- Una correlación de metadatos de supervisor incluidos los metadatos del sistema inmutables y los metadatos del sistema mutables. Los metadatos del sistema inmutables son los siguientes:
 - FILENAME - nombre del archivo que cumplía la condición desencadenante
 - FILEPATH - vía de acceso al archivo que cumplía la condición desencadenante
 - FILESIZE (en bytes - puede que estos metadatos no estén disponibles) - tamaño del archivo que cumplía la condición desencadenante
 - LASTMODIFIEDDATE (Local) - fecha en la que se modificó por última vez el archivo que cumplía la condición desencadenante. Esta fecha se expresa como fecha local del huso horario en el que se ejecuta el agente y se formatea como fecha ISO 8601.
 - LASTMODIFIEDTIME (Local) - hora, en formato local, a la que se modificó por última vez el archivo que cumplía la condición desencadenante. Esta hora se expresa como hora local del huso horario en el que se ejecuta el agente y tiene un formato como el de la hora ISO 8601.
 - LASTMODIFIEDDATEUTC - fecha, en formato universal, en la que se modificó por última vez el archivo que cumplía la condición desencadenante. Esta fecha se expresa como fecha local convertida al huso horario UTC y se formatea como fecha ISO 8601.
 - LASTMODIFIEDTIMEUTC - hora, en formato universal, a la que se modificó por última vez el archivo que cumplía la condición desencadenante. Esta hora se expresa como la hora local convertida al huso horario UTC y tiene un formato de hora ISO 8601.
 - AGENTNAME - el nombre del agente supervisor
- Una serie XML que representa la tarea que se va a ejecutar como resultado del desencadenante de supervisor.

Las salidas de supervisor devuelven los siguientes datos:

- Un indicador que especifica si se debe seguir avanzando (proseguir o cancelar)
- Una serie para insertar en el mensaje de registro que ha cumplido el desencadenante

Como resultado de la ejecución del código de salida de supervisor, puede que también se hayan actualizado los metadatos del supervisor y la serie XML de definición de tarea que se transfirieron originalmente como parámetros.

El valor de la propiedad de agente `monitorExitClasses` (en el archivo `agent.properties`) especifica qué clases de salida de supervisor se deben cargar; cada clase de salida va separada por una coma. Por ejemplo:

```
monitorExitClasses=testExits.TestExit1,testExits.testExit2
```

La interfaz con la salida de usuario de supervisor es la siguiente:

```
package com.ibm.wmqfte.exitroutine.api;

import java.util.Map;

/**
 * An interface that is implemented by classes that want to be invoked as part of
 * user exit routine processing. This interface defines a method that will be
 * invoked immediately prior to starting a task as the result of a monitor trigger
 */
public interface MonitorExit {

    /**
     * Invoked immediately prior to starting a task as the result of a monitor
     * trigger.
     *
     * @param environmentMetaData
     *      meta data about the environment in which the implementation
     *      of this method is running. This information can only be read,
     *      it cannot be updated by the implementation. The constant
     *      defined in <code>EnvironmentMetaDataConstants</code> class can
     *      be used to access the data held by this map.
     *
     * @param monitorMetaData
     *      meta data to associate with the monitor. The meta data passed
     *      to this method can be altered, and the changes will be
     *      reflected in subsequent exit routine invocations. This map
     *      also contains keys with IBM reserved names. These entries are
     *      defined in the <code>MonitorMetaDataConstants</code> class and
     *      have special semantics. The the values of the IBM reserved names
     *      cannot be modified by the exit
     *
     * @param taskDetails
     *      An XML String representing the task to be executed as a result of
     *      the monitor triggering. This XML string may be modified by the
     *      exit
     *
     * @return
     *      a monitor exit result object which is used to determine if the
     *      task should proceed, or be cancelled.
     */
    MonitorExitResult onMonitor(Map<String, String> environmentMetaData,
                               Map<String, String> monitorMetaData,
                               Reference<String> taskDetails);
}
```

Las constantes de los valores reservados de IBM en los metadatos del supervisor son las siguientes:

```
package com.ibm.wmqfte.exitroutine.api;

/**
 * Constants for IBM reserved values placed into the monitor meta data
 * maps used by the monitor exit routines.
 */
public interface MonitorMetaDataConstants {

    /**
     * The value associated with this key is the name of the trigger
     * file associated with the monitor. Any modification performed
     * to this property by user exit routines will be ignored.
     */
    final String FILE_NAME_KEY = "FILENAME";
}
```

```

/**
 * The value associated with this key is the path to the trigger
 * file associated with the monitor. Any modification performed
 * to this property by user exit routines will be ignored.
 */
final String FILE_PATH_KEY = "FILEPATH";

/**
 * The value associated with this key is the size of the trigger
 * file associated with the monitor. This will not be present in
 * the cases where the size cannot be determined. Any modification
 * performed to this property by user exit routines will be ignored.
 */
final String FILE_SIZE_KEY = "FILESIZE";

/**
 * The value associated with this key is the local date on which
 * the trigger file associated with the monitor was last modified.
 * Any modification performed to this property by user exit routines
 * will be ignored.
 */
final String LAST_MODIFIED_DATE_KEY = "LASTMODIFIEDDATE";

/**
 * The value associated with this key is the local time at which
 * the trigger file associated with the monitor was last modified.
 * Any modification performed to this property by user exit routines
 * will be ignored.
 */
final String LAST_MODIFIED_TIME_KEY = "LASTMODIFIEDTIME";

/**
 * The value associated with this key is the UTC date on which
 * the trigger file associated with the monitor was last modified.
 * Any modification performed to this property by user exit routines
 * will be ignored.
 */
final String LAST_MODIFIED_DATE_KEY_UTC = "LASTMODIFIEDDATEUTC";

/**
 * The value associated with this key is the UTC time at which
 * the trigger file associated with the monitor was last modified.
 * Any modification performed to this property by user exit routines
 * will be ignored.
 */
final String LAST_MODIFIED_TIME_KEY_UTC = "LASTMODIFIEDTIMEUTC";

/**
 * The value associated with this key is the name of the agent on which
 * the monitor is running. Any modification performed to this property by
 * user exit routines will be ignored.
 */
final String MONITOR_AGENT_KEY = "AGENTNAME";
}

```

Ejemplo de rutina de salida de usuario

En este ejemplo de clase se implementa la interfaz MonitorExit. Este ejemplo añade una variable de sustitución personalizada a los metadatos de supervisor denominados *REDIRECTEDAGENT* que se llenarán con un valor de LONDON si la hora del día es impar y un valor de PARIS para horas pares. El código de resultado de salida de supervisor está definido para devolver siempre un valor proceed.

```

package com.ibm.wmqfte.monitor;

import java.util.Calendar;
import java.util.Map;

import com.ibm.wmqfte.exitroutine.api.MonitorExit;
import com.ibm.wmqfte.exitroutine.api.MonitorExitResult;
import com.ibm.wmqfte.exitroutine.api.Reference;

/**
 * Example resource monitor user exit that changes the monitor mutable
 * metadata value between 'LONDON' and 'PARIS' depending on the hour of the day.
 */

```

```

public class TestMonitorExit implements MonitorExit {

    // custom variable that will substitute destination agent
    final static String REDIRECTED_AGENT = "REDIRECTEDAGENT";

    public MonitorExitResult onMonitor(
        Map<String, String> environmentMetaData,
        Map<String, String> monitorMetaData,
        Reference<String> taskDetails) {

        // always succeed
        final MonitorExitResult result = MonitorExitResult.PROCEED_RESULT;

        final int hour = Calendar.getInstance().get(Calendar.HOUR_OF_DAY);

        if (hour%2 == 1) {
            monitorMetaData.put(REDIRECTED_AGENT, "LONDON");
        } else {
            monitorMetaData.put(REDIRECTED_AGENT, "PARIS");
        }

        return result;
    }
}

```

La tarea correspondiente para un supervisor que emplea la variable de sustitución *REDIRECTEDAGENT* podría ser similar a la siguiente:

```

<?xml version="1.0" encoding="UTF-8"?>
<request version="4.00"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <managedTransfer>
    <originator>
      <hostName>reportserver.com</hostName>
      <userID>USER1</userID>
    </originator>
    <sourceAgent agent="AGENT1"
      QMgr="QM1"/>
    <destinationAgent agent="{REDIRECTEDAGENT}"
      QMgr="QM2"/>
    <transferSet>
      <item mode="binary" checksumMethod="MD5">
        <source recursive="false" disposition="delete">
          <file>c:\sourcefiles\reports.doc</file>
        </source>
        <destination type="file" exist="overwrite">
          <file>c:\destinationfiles\reports.doc</file>
        </destination>
      </item>
    </transferSet>
  </managedTransfer>
</request>

```

Antes de iniciar esta transferencia, el valor del atributo «agent» del elemento « <destinationAgent> » se sustituye por «» LONDON o PARIS«».

Debe especificar la variable de sustitución en la clase de salida del supervisor y la serie XML de definición de tarea en mayúsculas.

Conceptos relacionados

[Personalización de MFT con salidas de usuario](#)

[“Metadatos para salidas de usuario de MFT” en la página 2168](#)

Hay tres tipos diferentes de metadatos que se pueden suministrar a las rutinas de salida para Managed File Transfer: entorno, transferencia y metadatos de archivo. Estos metadatos se presentan como mapas de pares de clave-valor Java.

[“Java interfaces para salidas MFT de usuario” en la página 2178](#)

Use los temas de esta sección para obtener información de referencia sobre las interfaces Java de las rutinas de salida de usuario.

[Salidas de usuario de origen y destino de MFT](#)

Referencia relacionada

[“Propiedades de agente MFT para salidas de usuario” en la página 2176](#)

Además de las propiedades estándar del archivo `agent.properties`, hay varias propiedades avanzadas específicas para las rutinas de salida de usuario. Estas propiedades no están incluidas de forma predeterminada; por tanto, si desea utilizar alguna de ellas, deberá editar manualmente el archivo `agent.properties`. Si realiza algún cambio en el archivo `agent.properties` mientras se ejecuta dicho agente, detenga y vuelva a iniciar el agente para recuperar los cambios.

Propiedades de agente MFT para salidas de usuario

Además de las propiedades estándar del archivo `agent.properties`, hay varias propiedades avanzadas específicas para las rutinas de salida de usuario. Estas propiedades no están incluidas de forma predeterminada; por tanto, si desea utilizar alguna de ellas, deberá editar manualmente el archivo `agent.properties`. Si realiza algún cambio en el archivo `agent.properties` mientras se ejecuta dicho agente, detenga y vuelva a iniciar el agente para recuperar los cambios.

Las variables de entorno se pueden utilizar en algunas Managed File Transfer propiedades que representan ubicaciones de archivos o directorios. Esto permite que las ubicaciones de archivos o directorios que se utilizan al ejecutar componentes del producto, varíen dependiendo de los cambios del entorno, como por ejemplo qué usuario está ejecutando el proceso. Para obtener más información, consulte [Variables de entorno en las propiedades MFT](#).

Propiedades de la rutina de salida del usuario

Las rutinas de salida de usuario se llaman en el orden que se indica en la siguiente tabla. Para obtener más información sobre el `agent.properties` archivo, consulte [Propiedades avanzadas del agente: rutina de salida de usuario](#).

Tabla 921. Propiedades de agente para salidas de usuario	
Nombre de propiedad	Descripción
<code>sourceTransferEndExitClasses</code>	Especifica una lista de clases separadas por comas, que implementan una rutina de salida de finalización de transferencia de origen.
<code>sourceTransferStartExitClasses</code>	Especifica una lista de clases separadas por comas, que implementan una rutina de salida de inicio de transferencia de origen.
<code>destinationTransferStartExitClasses</code>	Especifica una lista de clases separadas por comas, que implementan una rutina de salida de usuario de inicio de transferencia.
<code>destinationTransferEndExitClasses</code>	Especifica una lista de clases separadas por comas, que implementan una rutina de salida de usuario de transferencia de destino.
<code>exitClassPath</code>	Especifica una lista de directorios, delimitados por caracteres y específicos de la plataforma, que actúan como la vía de acceso de clases para rutinas de salida de usuario. El directorio de salidas del agente se busca antes que cualquier entrada en esta ruta de clases.  En Windows, las rutas de archivo pueden especificarse utilizando barras invertidas dobles (\\) como separador o barras simples (/). Los paréntesis, las comas (,) y las barras invertidas (\) son caracteres especiales en los comandos MFT, por lo que deben precederse de una barra invertida (\). Por ejemplo: <pre>exitClassPath=C:/mycomp/mqft/exits/encryptFileExit.jar; C:/mycomp/mqft/exits/fileFilter.jar.</pre> El valor de esta propiedad puede contener variables de entorno.
<code>exitNativeLibraryPath</code>	Especifica una lista de directorios, delimitados por caracteres y específicos de la plataforma, que funcionan como vía de acceso de biblioteca nativa para rutinas de salida de usuario. El valor de esta propiedad puede contener variables de entorno.
<code>monitorExitClasses</code>	Especifica una lista de clases separadas por comas, que implementan una rutina de salida de supervisor. Para obtener más información, consulte Salidas de usuario del supervisor de recursos de MFT .

Tabla 921. Propiedades de agente para salidas de usuario (continuación)

Nombre de propiedad	Descripción
protocolBridgeCredentialExitClasses	Especifica una lista de clases separadas por comas que implementan una rutina de salida de usuario de credenciales de puente de protocolo. Para obtener más información, consulte Correlacionar credenciales para un servidor de archivos utilizando clases de salida .
protocolBridgePropertiesExitClasses	Especifica una lista separada por comas de clases que implementan una rutina de salida de usuario de las propiedades de servidor de puente de protocolo. Para obtener más información, consulte ProtocolBridgePropertiesExit2: buscar propiedades de servidor de archivos de protocolo .
IOExitClasses	Especifica una lista separada por comas de clases que implementan una rutina de salida de usuario de E/S. Especifique sólo las clases que implementan la interfaz IOExit, es decir, no especifique clases que implementan las otras interfaces de salida de usuario de E/S, por ejemplo IOExitResourcePath e IOExitChannel. Para obtener más información, consulte Utilización de salidas de usuario de E/S de transferencia de MFT .

Orden de invocación de salida

Las salidas de origen y de destino se invocan en el orden siguiente:

1. SourceTransferStartExit
2. DestinationTransferStartExit
3. DestinationTransferEndExit
4. SourceTransferEndExit

Encadenamiento de salidas de origen y de destino

Si se especifican varias salidas, la primera salida de la lista se invoca primero, seguida de la segunda salida y así sucesivamente. Los cambios realizados por la primera salida se transfieren como entrada a la salida que se invoque posteriormente y así sucesivamente. Por ejemplo, si hay dos salidas de inicio de transferencia de origen, los cambios realizados en los metadatos de la transferencia por la primera salida se entran en la segunda salida. Cada salida devuelve su propio resultado. Si todas las salidas de un tipo determinado devuelven PROCEED como código de resultado de salida, el resultado global será PROCEED. Si una o varias salidas devuelven CANCEL_TRANSFER, el resultado global será CANCEL_TRANSFER. Todos los códigos de resultado y series devuelvas por las salidas se generan como salida en el registro de transferencias.

Si el resultado global de la salida de inicio de la transferencia de origen es PROCEED, la transferencia proseguirá utilizando los cambios realizados por las salidas. Si el resultado global es CANCEL_TRANSFER, se invocarán las salidas de finalización de la transferencia de origen y se cancelará la transferencia. El estado de terminación en el registro de transferencias es "cancelado".

Si el resultado global de las salidas de inicio de la transferencia de destino es PROCEED, la transferencia proseguirá utilizando los cambios realizados por las salidas. Si el resultado global es CANCEL_TRANSFER, se invocarán las salidas de finalización de la transferencia de destino y a continuación, las salidas de finalización de la transferencia de origen. Finalmente, se cancelará la transferencia. El estado de terminación en el registro de transferencias es "cancelado".

Si una salida de origen o destino necesita transferir información a las siguientes salidas en la cadena o en el orden de ejecución, deberá realizarlo actualizando los metadatos de la transferencia. El uso de los metadatos de transferencia es específico de la implementación de salida. Por ejemplo, si una salida establece el resultado de retorno en CANCEL_TRANSFER y necesita comunicarse con las siguientes salidas que la transferencia ha cancelado, deberá realizarlo estableciendo un valor de metadatos de transferencia que sea comprensible para las otras salidas.

Ejemplo

```
sourceTransferStartExitClasses=com.ibm.wmqfte.test.MFTTestSourceTransferStartExit
sourceTransferEndExitClasses=com.ibm.wmqfte.test.MFTTestSourceTransferEndExit
destinationTransferStartExitClasses=com.ibm.wmqfte.test.MFTTestDestinationTransferStartExit
destinationTransferEndExitClasses=com.ibm.wmqfte.test.MFTTestDestinationTransferEndExit
exitClassPath=C:/mycomp/mqft/exits/encryptFileExit.jar;C:/mycomp/mqft/exits/fileFilter.jar
```

Conceptos relacionados

[Personalización de MFT con salidas de usuario](#)

[“Metadatos para salidas de usuario de MFT” en la página 2168](#)

Hay tres tipos diferentes de metadatos que se pueden suministrar a las rutinas de salida para Managed File Transfer: entorno, transferencia y metadatos de archivo. Estos metadatos se presentan como mapas de pares de clave-valor Java.

[“Java interfaces para salidas MFT de usuario” en la página 2178](#)

Use los temas de esta sección para obtener información de referencia sobre las interfaces Java de las rutinas de salida de usuario.

Referencia relacionada

[“Salidas de usuario del supervisor de recursos de MFT” en la página 2172](#)

Las salidas de usuario del supervisor de recursos permiten configurar que el código personalizado se ejecute cuando se cumple una condición desencadenante de un supervisor antes de que se inicie la tarea asociada.

[Variables de entorno en MFT propiedades](#)

[El archivo MFT agent.properties](#)

Java interfaces para salidas MFT de usuario

Use los temas de esta sección para obtener información de referencia sobre las interfaces Java de las rutinas de salida de usuario.

Tareas relacionadas

[Personalización de MFT con salidas de usuario](#)

Referencia relacionada

[“Interfaz DestinationTransferStartExit.java” en la página 2182](#)

[“Interfaz DestinationTransferEndExit.java” en la página 2181](#)

[“Interfaz IOExit.java” en la página 2184](#)

[“Interfaz IOExitChannel.java” en la página 2186](#)

[“Interfaz IOExitLock.java” en la página 2188](#)

[“Interfaz IOExitPath.java” en la página 2188](#)

[“Interfaz IOExitProperties.java” en la página 2190](#)

[“Interfaz IOExitRecordChannel.java” en la página 2192](#)

[“Interfaz IOExitRecordResourcePath.java” en la página 2194](#)

[“Interfaz IOExitResourcePath.java” en la página 2195](#)

[“Interfaz IOExitWildcardPath.java” en la página 2200](#)

[“Interfaz MonitorExit.java” en la página 2200](#)

[“Interfaz ProtocolBridgeCredentialExit.java” en la página 2201](#)

[“Interfaz ProtocolBridgeCredentialExit2.java” en la página 2202](#)

[“Interfaz ProtocolBridgePropertiesExit2.java” en la página 2203](#)

[“Interfaz SourceTransferStartExit.java” en la página 2207](#)

[“Interfaz SourceTransferEndExit.java” en la página 2205](#)

Interfaz CDCredentialExit.java

CDCredentialExit.java

```
/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * © Copyright IBM Corp. 2011, 2026. All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with
 * IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.util.Map;

/**
 * An interface that is implemented by classes that are invoked as part of
 * user exit routine processing. This interface defines methods that are
 * invoked by a Connect:Direct bridge agent to map the IBM MQ user ID of the transfer to credentials
 * that are used to access the Connect:Direct node.
 * There will be one instance of each implementation class per Connect:Direct bridge agent. The methods
 * can be called from different threads so the methods must be synchronized.
 */
public interface CDCredentialExit {

    /**
     * Invoked once when a Connect:Direct bridge agent is started. It is intended to initialize
     * any resources that are required by the exit
     *
     * @param bridgeProperties
     *      The values of properties defined for the Connect:Direct bridge.
     *      These values can only be read, they cannot be updated by
     *      the implementation.
     *
     * @return
     *      true if the initialisation is successful and false if unsuccessful
     *      If false is returned from an exit the Connect:Direct bridge agent does not
     *      start.
     */
    public boolean initialize(final Map<String, String> bridgeProperties);

    /**
     * Invoked once per transfer to map the IBM MQ user ID in the transfer message to the
     * credentials to be used to access the Connect:Direct node.
     *
     * @param mqUserId The IBM MQ user ID from which to map to the credentials to be used
     *                  to access the Connect:Direct node
     * @param snode    The name of the Connect:Direct SNODE specified as the cdNode in the
     *                  file path. This is used to map the correct user ID and password for the
     *                  SNODE.
     * @return         A credential exit result object that contains the result of the map and
     *                  the credentials to use to access the Connect:Direct node
     */
    public CDCredentialExitResult mapMQUserId(final String mqUserId, final String snode);

    /**
     * Invoked once when a Connect:Direct bridge agent is shutdown. This method releases
     * any resources that were allocated by the exit
     *
     * @param bridgeProperties
     *      The values of properties defined for the Connect:Direct bridge.
     *      These values can only be read, they cannot be updated by
     *      the implementation.
     *
     * @return
     */
    public void shutdown(final Map<String, String> bridgeProperties);
}
```

Interfaz CredentialExitResult.java

CredentialExitResult.java

```
/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * © Copyright IBM Corp. 2008, 2026. All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with
 * IBM Corp.
 */

package com.ibm.wmqfte.exitroutine.api;

/**
 * The result of invoking a Credential mapMQUserId exit method. It is composed of a result
 * code, which determines whether the mapping of the user id was successful, and an optional
 * Credentials object if the mapping is successful.
 */
public class CredentialExitResult {

    private final CredentialExitResultCode resultCode;
    private final Credentials credentials;

    /**
     * Constructor. Creates a credential exit result object with a specified result
     * code and optionally credentials.
     *
     * @param resultCode
     *         The result code to associate with the exit result being created.
     *
     * @param credentials
     *         The credentials to associate with the exit result being created.
     *         A value of <code>null</code> can be specified to indicate no
     *         credentials. If the resultCode is USER_SUCCESSFULLY_MAPPED the
     *         credentials must be set to a non-null value,
     */
    public CredentialExitResult(CredentialExitResultCode resultCode, Credentials credentials) {
        this.resultCode = resultCode;
        this.credentials = credentials;
    }

    /**
     * Returns the result code associated with this credential exit result
     *
     * @return the result code associated with this exit result.
     */
    public CredentialExitResultCode getResultCode() {
        return resultCode;
    }

    /**
     * Returns the credentials associated with this credential exit result
     *
     * @return the explanation associated with this credential exit result.
     */
    public Credentials getCredentials() {
        return credentials;
    }
}
```

Tareas relacionadas

[Personalización de MFT con salidas de usuario](#)

Referencia relacionada

[“Interfaz SourceTransferStartExit.java” en la página 2207](#)

[“Interfaz DestinationTransferStartExit.java” en la página 2182](#)

[“Interfaz DestinationTransferEndExit.java” en la página 2181](#)

[“Interfaz MonitorExit.java” en la página 2200](#)

[“Interfaz ProtocolBridgeCredentialExit.java” en la página 2201](#)

Interfaz DestinationTransferEndExit.java

DestinationTransferEndExit.java

```
/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * © Copyright IBM Corp. 2008, 2026. All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with
 * IBM Corp.
 */
package com.ibm.wmqfte.exitpoint.api;

/**
 * An interface that is implemented by classes that want to be invoked as part of
 * user exit routine processing. This interface defines a method that will be
 * invoked immediately after completing a transfer on the agent acting as the
 * destination of the transfer.
 */
public interface DestinationTransferEndExit {

    /**
     * Invoked immediately after the completion of a transfer on the agent acting as
     * the destination of the transfer.
     *
     * @param transferExitResult
     *      a result object reflecting whether or not the transfer completed
     *      successfully.
     *
     * @param sourceAgentName
     *      the name of the agent acting as the source of the transfer.
     *
     * @param destinationAgentName
     *      the name of the agent acting as the destination of the
     *      transfer. This is the name of the agent that the
     *      implementation of this method will be invoked from.
     *
     * @param environmentMetaData
     *      meta data about the environment in which the implementation
     *      of this method is running. This information can only be read,
     *      it cannot be updated by the implementation. The constants
     *      defined in <code>EnvironmentMetaDataConstants</code> class can
     *      be used to access the data held by this map.
     *
     * @param transferMetaData
     *      meta data to associate with the transfer. The information can
     *      only be read, it cannot be updated by the implementation. This
     *      map may also contain keys with IBM reserved names. These
     *      entries are defined in the <code>TransferMetaDataConstants</code>
     *      class and have special semantics.
     *
     * @param fileResults
     *      a list of file transfer result objects that describe the source
     *      file name, destination file name and result of each file transfer
     *      operation attempted.
     *
     * @return
     *      an optional description to enter into the log message describing
     *      transfer completion. A value of <code>null</code> can be used
     *      when no description is required.
     */
    String onDestinationTransferEnd(TransferExitResult transferExitResult,
                                   String sourceAgentName,
                                   String destinationAgentName,
                                   Map<String, String>environmentMetaData,
                                   Map<String, String>transferMetaData,
                                   List<FileTransferResult>fileResults);
}
```

Tareas relacionadas

[Personalización de MFT con salidas de usuario](#)

Referencia relacionada

[“Interfaz SourceTransferStartExit.java” en la página 2207](#)

[“Interfaz SourceTransferEndExit.java” en la página 2205](#)

[“Interfaz DestinationTransferStartExit.java” en la página 2182](#)

[“Interfaz MonitorExit.java” en la página 2200](#)

[“Interfaz ProtocolBridgeCredentialExit.java” en la página 2201](#)

Interfaz DestinationTransferStartExit.java

DestinationTransferStartExit.java

```
/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * © Copyright IBM Corp. 2008, 2026. All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with
 * IBM Corp.
 */
package com.ibm.wmqfte.exitpoint.api;

/**
 * An interface that is implemented by classes that want to be invoked as part of
 * user exit routine processing. This interface defines a method that will be
 * invoked immediately prior to starting a transfer on the agent acting as the
 * destination of the transfer.
 */
public interface DestinationTransferStartExit {

    /**
     * Invoked immediately prior to starting a transfer on the agent acting as
     * the destination of the transfer.
     *
     * @param sourceAgentName
     *     the name of the agent acting as the source of the transfer.
     *
     * @param destinationAgentName
     *     the name of the agent acting as the destination of the
     *     transfer. This is the name of the agent that the
     *     implementation of this method will be invoked from.
     *
     * @param environmentMetaData
     *     meta data about the environment in which the implementation
     *     of this method is running. This information can only be read,
     *     it cannot be updated by the implementation. The constants
     *     defined in <code>EnvironmentMetaDataConstants</code> class can
     *     be used to access the data held by this map.
     *
     * @param transferMetaData
     *     meta data to associate with the transfer. The information can
     *     only be read, it cannot be updated by the implementation. This
     *     map may also contain keys with IBM reserved names. These
     *     entries are defined in the <code>TransferMetaDataConstants</code>
     *     class and have special semantics.
     *
     * @param fileSpecs
     *     a list of file specifications that govern the file data to
     *     transfer. The implementation of this method can modify the
     *     entries in this list and the changes will be reflected in the
     *     files transferred. However, new entries may not be added and
     *     existing entries may not be removed.
     *
     * @return
     *     a transfer exit result object which is used to determine if the
     *     transfer should proceed, or be cancelled.
     */
    TransferExitResult onDestinationTransferStart(String sourceAgentName,
                                                  String destinationAgentName,
                                                  Map<String, String> environmentMetaData,
```

```
Map<String, String> transferMetaData,  
List<Reference<String>> fileSpecs);
```

Tareas relacionadas

[Personalización de MFT con salidas de usuario](#)

Referencia relacionada

[“Interfaz SourceTransferStartExit.java” en la página 2207](#)

[“Interfaz SourceTransferEndExit.java” en la página 2205](#)

[“Interfaz DestinationTransferEndExit.java” en la página 2181](#)

[“Interfaz MonitorExit.java” en la página 2200](#)

[“Interfaz ProtocolBridgeCredentialExit.java” en la página 2201](#)

Interfaz FileTransferResult.java

FileTransferResult.java

```
/*  
 * Licensed Materials - Property of IBM  
 *  
 * "Restricted Materials of IBM"  
 *  
 * 5724-H72  
 *  
 * © Copyright IBM Corp. 2008, 2026. All Rights Reserved.  
 *  
 * US Government Users Restricted Rights - Use, duplication or  
 * disclosure restricted by GSA ADP Schedule Contract with  
 * IBM Corp.  
 */  
  
package com.ibm.wmqfte.exitroutine.api;  
  
/**  
 * Result information about a file transfer.  
 */  
public interface FileTransferResult {  
  
    /** An enumeration for the <code>getCorrelatorType()</code> method. */  
    public enum CorrelationInformationType {  
        /** No correlation information is available for this result */  
        NONE,  
        /**  
         * The correlation information relates to work done in  
         * IBM Sterling File Gateway.  
         */  
        SFG  
    }  
  
    /**  
     * Returns the source file specification, from which the file was transferred.  
     *  
     * @return the source file specification, from which the file was  
     * transferred.  
     */  
    String getSourceFileSpecification();  
  
    /**  
     * Returns the destination file specification, to which the file was transferred.  
     *  
     * @return the destination file specification, to which the file was  
     * transferred. A value of <code>null</code> may be returned  
     * if the transfer did not complete successfully.  
     */  
    String getDestinationFileSpecification();  
  
    /**  
     * Returns the result of the file transfer operation.  
     *  
     * @return the result of the file transfer operation.  
     */  
    FileExitResult getExitResult();  
}
```

```

    * @return an enumerated value that identifies the product to which this correlating
    *         information relates.
    */
    CorrelationInformationType getCorrelatorType();

    /**
    * @return the first string component of the correlating identifier that relates
    *         this transfer result to work done in another product. A value of null
    *         may be returned either because the other product does not utilize a
    *         string based correlation information or because there is no correlation
    *         information.
    */
    String getString1Correlator();

    /**
    * @return the first long component of the correlating identifier that relates
    *         this transfer result to work done in another product. A value of zero
    *         is returned when there is no correlation information or the other
    *         product does not utilize long based correlation information or because
    *         the value really is zero!
    */
    long getLong1Correlator();
}

```

Tareas relacionadas

[Personalización de MFT con salidas de usuario](#)

Referencia relacionada

[“Interfaz SourceTransferStartExit.java” en la página 2207](#)

[“Interfaz DestinationTransferStartExit.java” en la página 2182](#)

[“Interfaz DestinationTransferEndExit.java” en la página 2181](#)

[“Interfaz MonitorExit.java” en la página 2200](#)

[“Interfaz ProtocolBridgeCredentialExit.java” en la página 2201](#)

Interfaz IOExit.java

IOExit.java

```

/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * © Copyright IBM Corp. 2011, 2026. All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with
 * IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.io.IOException;
import java.util.Map;

import com.ibm.wmqfte.exitroutine.api.IOExitRecordResourcePath.RecordFormat;

/**
 * An interface that is implemented by classes that you want to be invoked as
 * part of user exit routine processing. This interface defines methods that
 * will be invoked during transfers to perform the underlying file system I/O
 * work for WMQFTE transfers.
 * <p>
 * The {@link #initialize(Map)} method will be called once when the exit is
 * first installed. The WMQFTE agent properties are passed to this method, thus
 * enabling the exit to understand its environment.
 * <p>
 * The {@link #isSupported(String)} method will be invoked during WMQFTE
 * transfers to determine whether the user exit should be used. If the
 * {@link #isSupported(String)} method returns a value of {@code true}, the
 * {@link #newPath(String)} method will be invoked for the paths specified for
 * the transfer request. The returned {@link IOExitPath} instance from a

```

```

* {@link #newPath(String)} method invocation will then be used by the WMQFTE
* transfer to obtain information about the resource and to transfer data to or
* from the resource.
* <p>
* To obtain transfer context for an I/O exit, a {@link SourceTransferStartExit}
* or {@link DestinationTransferStartExit} as appropriate, should be installed
* to enable information to be seen by this exit. The
* {@link SourceTransferStartExit} or {@link DestinationTransferStartExit} are
* passed the transfer's environment, metadata, and a list of file
* specifications for the transfer. The paths for the file specifications are
* the paths passed to the I/O exit's {@link #newPath(String)} method.
* <p>
* Note also that the {@link #isSupported(String)} and {@link #newPath(String)}
* methods might be called at other times by a WMQFTE agent and not just during
* transfers. For example, at transfer setup time the I/O system is queried to
* resolve the full resource paths for transfer.
*/
public interface IOExit {

    /**
     * Invoked once when the I/O exit is first required for use. It is intended
     * to initialize any resources that are required by the exit.
     *
     * @param agentProperties
     *     The values of properties defined for the WMQFTE agent. These
     *     values can only be read, they cannot be updated by the
     *     implementation.
     * @return {@code true} if the initialization is successful and {@code
     *     false} if unsuccessful. If {@code false} is returned from an
     *     exit, the exit will not be used.
     */
    boolean initialize(final Map<String, String> agentProperties);

    /**
     * Indicates whether this I/O user exit supports the specified path.
     * <p>
     * This method is used by WMQFTE to determine whether the I/O user exit
     * should be used within a transfer. If no I/O user exit returns true for
     * this method, the default WMQFTE file I/O function will be used.
     *
     * @param path
     *     The path to the required I/O resource.
     * @return {@code true} if the specified path is supported by the I/O exit,
     *     {@code false} otherwise
     */
    boolean isSupported(String path);

    /**
     * Obtains a new {@link IOExitPath} instance for the specified I/O resource
     * path.
     * <p>
     * This method will be invoked by WMQFTE only if the
     * {@link #isSupported(String)} method has been called for the path and
     * returned {@code true}.
     *
     * @param path
     *     The path to the required I/O resource.
     * @return A {@link IOExitPath} instance for the specified path.
     * @throws IOException
     *     If the path cannot be created for any reason.
     */
    IOExitPath newPath(String path) throws IOException;

    /**
     * Obtains a new {@link IOExitPath} instance for the specified I/O resource
     * path and passes record format and length information required by the
     * WMQFTE transfer.
     * <p>
     * Typically this method will be called for the following cases:
     * <ul>
     * <li>A path where a call to {@link #newPath(String)} has previously
     * returned a {@link IOExitRecordResourcePath} instance and WMQFTE is
     * re-establishing a new {@link IOExitPath} instance for the path, from an
     * internally-serialized state. The passed recordFormat and recordLength
     * will be the same as those for the original
     * {@link IOExitRecordResourcePath} instance.</li>
     * <li>A transfer destination path where the source of the transfer is
     * record oriented. The passed recordFormat and recordLength will be the
     * same as those for the source.</li>
     * </ul>
     * The implementation can act on the record format and length information as
     * deemed appropriate. For example, for a destination agent if the

```

```

* destination does not already exist and the source of the transfer is
* record oriented, the passed recordFormat and recordLength information
* could be used to create an appropriate record-oriented destination path.
* If the destination path already exists, the passed recordFormat and
* recordLength information could be used to perform a compatibility check
* and throw an {@link IOException} if the path is not compatible. A
* compatibility check could ensure that a record oriented path's record
* format is the same as the passed record format or that the record length
* is greater or equal to the passed record length.
* <p>
* This method will be invoked by WMQFTE only if the
* {@link #isSupported(String)} method has been called for the path and
* returned {@code true}.
*
* @param path
*         The path to the required I/O resource.
* @param recordFormat
*         The advised record format.
* @param recordLength
*         The advised record length.
* @return A {@link IOExitPath} instance for the specified path.
* @throws IOException
*         If the path cannot be created for any reason. For example,
*         the passed record format or length is incompatible with the
*         path's actual record format or length.
*/
IOExitPath newPath(String path, RecordFormat recordFormat, int recordLength)
    throws IOException;

```

Tareas relacionadas

[Utilización de salidas de usuario de E/S de transferencia de MFT](#)

[Personalización de MFT con salidas de usuario](#)

Interfaz IOExitChannel.java

IOExitChannel.java

```

/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * © Copyright IBM Corp. 2011, 2026. All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with
 * IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.io.IOException;
import java.nio.ByteBuffer;

/**
 * Represents a channel that enables data to be read from or written to an
 * {@link IOExitResourcePath} resource.
 */
public interface IOExitChannel {

    /**
     * Obtains the data size for the associated {@link IOExitResourcePath} in
     * bytes.
     *
     * @return The data size in bytes.
     * @throws IOException
     *         If a problem occurs while attempting obtain the size.
     */
    long size() throws IOException;

    /**
     * Closes the channel, flushing any buffered write data to the resource and
     * releasing any locks.
     *
     * @throws RecoverableIOException
     */
}

```

```

*           If a recoverable problem occurs while closing the resource.
*           This means that WMQFTE can attempt to recover the transfer.
* @throws IOException
*           If some other I/O problem occurs. For example, the channel might
*           already be closed.
*/
void close() throws RecoverableIOException, IOException;

/**
 * Reads data from this channel into the given buffer, starting at this
 * channel's current position, and updates the current position by the
 * amount of data read.
 * <p>
 * Data is copied into the buffer starting at its current position and up to
 * its limit. On return, the buffer's position is updated to reflect the
 * number of bytes read.
 *
 * @param buffer
 *         The buffer that the data is to be copied into.
 * @return The number of bytes read, which might be zero, or -1 if the end of
 *         data has been reached.
 * @throws RecoverableIOException
 *         If a recoverable problem occurs while reading the data. For a
 *         WMQFTE transfer this means that it will attempt to recover.
 * @throws IOException
 *         If some other I/O problem occurs. For a WMQFTE transfer this
 *         means that it will be failed.
 */
int read(ByteBuffer buffer) throws RecoverableIOException, IOException;

/**
 * Writes data to this channel from the given buffer, starting at this
 * channel's current position, and updates the current position by the
 * amount of data written. The channel's resource is grown to accommodate
 * the data, if necessary.
 * <p>
 * Data is copied from the buffer starting at its current position and up to
 * its limit. On return, the buffer's position is updated to reflect the
 * number of bytes written.
 *
 * @param buffer
 *         The buffer containing the data to be written.
 * @return The number of bytes written, which might be zero.
 * @throws RecoverableIOException
 *         If a recoverable problem occurs while writing the data. For a
 *         WMQFTE transfer this means that it will attempt to recover.
 * @throws IOException
 *         If some other I/O problem occurs. For a WMQFTE transfer this
 *         means that it will be failed.
 */
int write(ByteBuffer buffer) throws RecoverableIOException, IOException;

/**
 * Forces any updates to this channel's resource to be written to its
 * storage device.
 * <p>
 * This method is required to force changes to both the resource's content
 * and any associated metadata to be written to storage.
 *
 * @throws RecoverableIOException
 *         If a recoverable problem occurs while performing the force.
 *         For a WMQFTE transfer this means that it will attempt to
 *         recover.
 * @throws IOException
 *         If some other I/O problem occurs. For a WMQFTE transfer this
 *         means that it will be failed.
 */
void force() throws RecoverableIOException, IOException;

/**
 * Attempts to lock the entire resource associated with the channel for
 * shared or exclusive access.
 * <p>
 * The intention is for this method not to block if the lock is currently
 * unavailable.
 *
 * @param shared
 *         {@code true} if a shared lock is required, {@code false} if an
 *         exclusive lock is required.
 * @return A {@link IOExitLock} instance representing the newly acquired
 *         lock or null if the lock cannot be obtained.
 * @throws IOException

```

```

    *           If a problem occurs while attempting to acquire the lock.
    */
    IOExitLock tryLock(boolean shared) throws IOException;
}

```

Tareas relacionadas

[Utilización de salidas de usuario de E/S de transferencia de MFT](#)

[Personalización de MFT con salidas de usuario](#)

Interfaz *IOExitLock.java*

IOExitLock.java

```

/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * © Copyright IBM Corp. 2011, 2026. All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with
 * IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.io.IOException;

/**
 * Represents a lock on a resource for either shared or exclusive access.
 * {@link IOExitLock} instances are returned from
 * {@link IOExitChannel#tryLock(boolean)} calls and WMQFTE will request the
 * release of the lock at the appropriate time during a transfer. Additionally, when
 * a {@link IOExitChannel#close()} method is called it will be the
 * responsibility of the channel to release any associated locks.
 */
public interface IOExitLock {

    /**
     * Releases the lock.
     * <p>
     * After this method has been successfully called the lock is to be deemed as invalid.
     *
     * @throws IOException
     *         If the channel associated with the lock is not open or
     *         another problem occurs while attempting to release the lock.
     */
    void release() throws IOException;

    /**
     * Indicates whether this lock is valid.
     * <p>
     * A lock is considered valid until its @ {@link #release()} method is
     * called or the associated {@link IOExitChannel} is closed.
     *
     * @return {@code true} if this lock is valid, {@code false} otherwise.
     */
    boolean isValid();

    /**
     * @return {@code true} if this lock is for shared access, {@code false} if
     *         this lock is for exclusive access.
     */
    boolean isShared();
}

```

Tareas relacionadas

[Utilización de salidas de usuario de E/S de transferencia de MFT](#)

[Personalización de MFT con salidas de usuario](#)

Interfaz *IOExitPath.java*

IOExitPath.java

```
/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * © Copyright IBM Corp. 2011, 2026. All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with
 * IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

/**
 * Represents an abstract path that can be inspected and queried by WMQFTE for
 * transfer purposes.
 * <p>
 * There are two types of path supported:
 * <ul>
 * <li>{@link IOExitResourcePath} - Represents a path that denotes a data
 * resource. For example, a file, directory, or group of database records.</li>
 * <li>{@link IOExitWildcardPath} - Represents a wildcard path that can be
 * expanded to multiple {@link IOExitResourcePath} instances.</li>
 * </ul>
 */
public abstract interface IOExitPath {

    /**
     * Obtains the abstract path as a {@link String}.
     *
     * @return The abstract path as a {@link String}.
     */
    String getPath();

    /**
     * Obtains the name portion of this abstract path as a {@link String}.
     * <p>
     * For example, a UNIX-style file system implementation evaluates the
     * path {@code /home/fteuser/file1.txt} as having a name of {@code
     * file1.txt}.
     *
     * @return the name portion of this abstract path as a {@link String}.
     */
    String getName();

    /**
     * Obtains the parent path for this abstract path as a {@link String}.
     * <p>
     * For example, a UNIX-style file system implementation evaluates the
     * path {@code /home/fteuser/file1.txt} as having a parent path of {@code
     * /home/fteuser}.
     *
     * @return The parent portion of the path as a {@link String}.
     */
    String getParent();

    /**
     * Obtains the abstract paths that match this abstract path.
     * <p>
     * If this abstract path denotes a directory resource, a list of paths
     * for all resources within the directory are returned.
     * <p>
     * If this abstract path denotes a wildcard, a list of all paths
     * matching the wildcard are returned.
     * <p>
     * Otherwise null is returned, because this abstract path probably denotes a
     * single file resource.
     *
     * @return An array of {@link IOExitResourcePath}s that
     * match this path, or null if this method is not applicable.
     */
    IOExitResourcePath[] listPaths();
}
```

Tareas relacionadas

Utilización de salidas de usuario de E/S de transferencia de MFT

Personalización de MFT con salidas de usuario

Interfaz *IOExitProperties.java*

IOExitProperties.java

```
/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * © Copyright IBM Corp. 2011, 2026. All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with
 * IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

/**
 * Properties that determine how WMQFTE treats an {@link IOExitPath} for certain
 * aspects of I/O. For example, whether to use intermediate files.
 */
public class IOExitProperties {

    private boolean rereadSourceOnRestart = true;
    private boolean rechecksumSourceOnRestart = true;
    private boolean rechecksumDestinationOnRestart = true;
    private boolean useIntermediateFileAtDestination = true;
    private boolean requiresSingleThreadedChannelIO = false;

    /**
     * Determines whether the I/O exit implementation expects the resource to be
     * re-read from the start if a transfer is restarted.
     *
     * @return {@code true} if, on restart, the I/O exit expects the source
     * resource to be opened at the beginning and re-read from the
     * beginning (the {@link IOExitPath#openForRead(long)} method is
     * always invoked with 0L as an argument). {@code false} if, on
     * restart, the I/O exit expects the source to be opened at the
     * offset that the source agent intends to start reading from (the
     * {@link IOExitPath#openForRead(long)} method can be invoked with a
     * non-zero value as its argument).
     */
    public boolean getRereadSourceOnRestart() {
        return rereadSourceOnRestart;
    }

    /**
     * Sets the value to determine whether the I/O exit implementation expects
     * the resource to be re-read from the beginning if a transfer is restarted.
     * <p>
     * The default is {@code true}. The I/O exit should call this method when
     * required to change this value.
     *
     * @param rereadSourceOnRestart
     *        {@code true} if, on restart, the I/O exit expects the source
     * resource to be opened at the beginning and re-read from the
     * beginning (the {@link IOExitPath#openForRead(long)} method
     * is always invoked with 0L as an argument). {@code false}
     * if, on restart, the I/O exit expects the source to be opened
     * at the offset that the source agent intends to start reading
     * from (the {@link IOExitPath#openForRead(long)} method can be
     * invoked with a non-zero value as its argument).
     */
    public void setRereadSourceOnRestart(boolean rereadSourceOnRestart) {
        this.rereadSourceOnRestart = rereadSourceOnRestart;
    }

    /**
     * Determines whether the I/O exit implementation requires the source
     * resource to be re-checkedsummed if the transfer is restarted.
     * Re-checksumming takes place only if the
     */
}
```

```

* {@link #getRereadSourceOnRestart()} method returns {@code true}.
*
* @return {@code true} if, on restart, the I/O exit expects the already-
*         transferred portion of the source to be re-checksummed for
*         inconsistencies. Use this option in environments
*         where the source could be changed during a restart. {@code
*         false} if, on restart, the I/O exit does not require the
*         already-transferred portion of the source to be re-checksummed.
*/
public boolean getRechecksumSourceOnRestart() {
    return rechecksumSourceOnRestart;
}

/**
 * Sets the value to determine whether the I/O exit implementation requires
 * the source resource to be re-checksummed if the transfer is restarted.
 * Re-checksumming takes place only if the
 * {@link #getRereadSourceOnRestart()} method returns {@code true}.
 * <p>
 * The default is {@code true}. The I/O exit should call this method when
 * required to change this value.
 *
 * @param rechecksumSourceOnRestart
 *        {@code true} if, on restart, the I/O exit expects the already-
 *        transferred portion of the source to be re-checksummed
 *        for inconsistencies. Use this option in environments
 *        where the source could be changed during a restart.
 *        {@code false} if, on restart, the I/O exit does not
 *        require the already-transferred portion of the source to be
 *        re-checksummed.
 */
public void setRechecksumSourceOnRestart(boolean rechecksumSourceOnRestart) {
    this.rechecksumSourceOnRestart = rechecksumSourceOnRestart;
}

/**
 * Determines whether the I/O exit implementation requires the destination
 * resource to be re-checksummed if the transfer is restarted.
 *
 * @return {@code true} if, on restart, the I/O exit expects the already-
 *         transferred portion of the destination to be re-checksummed to
 *         check for inconsistencies. This option should be used in
 *         environments where the destination could have been changed while
 *         a restart is occurring. {@code false} if, on restart, the I/O exit
 *         does not require the already transferred portion of the
 *         destination to be re-checksummed.
 */
public boolean getRechecksumDestinationOnRestart() {
    return rechecksumDestinationOnRestart;
}

/**
 * Sets the value to determine whether the I/O exit implementation requires
 * the destination resource to be re-checksummed if the transfer is
 * restarted.
 * <p>
 * The default is {@code true}. The I/O exit should call this method when
 * required to change this value.
 *
 * @param rechecksumDestinationOnRestart
 *        {@code true} if, on restart, the I/O exit expects the already-
 *        transferred portion of the destination to be re-checksummed
 *        for inconsistencies. Use this option in environments
 *        where the destination could have been changed during a
 *        restart. {@code false} if, on restart, the I/O exit does not
 *        require the already-transferred portion of the destination
 *        to be re-checksummed.
 */
public void setRechecksumDestinationOnRestart(
    boolean rechecksumDestinationOnRestart) {
    this.rechecksumDestinationOnRestart = rechecksumDestinationOnRestart;
}

/**
 * Determines whether the I/O exit implementation requires the use of an
 * intermediate file when writing the data at the destination. The
 * intermediate file mechanism is typically used to prevent an incomplete
 * destination resource from being processed.
 *
 * @return {@code true} if data should be written to an intermediate file at
 *         the destination and then renamed (to the requested destination
 *         path name as specified in the transfer request) after the transfer is

```

```

*      complete. {@code false} if data should be written directly to the
*      requested destination path name without the use of an
*      intermediate file.
*/
public boolean getUseIntermediateFileAtDestination() {
    return useIntermediateFileAtDestination;
}

/**
 * Sets the value to determine whether the I/O exit implementation requires
 * the use of an intermediate file when writing the data at the destination.
 * The intermediate file mechanism is typically used to prevent an
 * incomplete destination resource from being processed.
 *
 * <p>
 * The default is {@code true}. The I/O exit should call this method when
 * required to change this value.
 *
 * @param useIntermediateFileAtDestination
 *      {@code true} if data should be written to an intermediate file
 *      at the destination and then renamed (to the requested
 *      destination path name as specified in the transfer request) after
 *      the transfer is complete. {@code false} if data should be written
 *      directly to the requested destination path name without the
 *      use of an intermediate file
 */
public void setUseIntermediateFileAtDestination(
    boolean useIntermediateFileAtDestination) {
    this.useIntermediateFileAtDestination = useIntermediateFileAtDestination;
}

/**
 * Determines whether the I/O exit implementation requires
 * {@link IOExitChannel} instances to be accessed by a single thread only.
 *
 * @return {@code true} if {@link IOExitChannel} instances are to be
 *      accessed by a single thread only.
 */
public boolean requiresSingleThreadedChannelIO() {
    return requiresSingleThreadedChannelIO;
}

/**
 * Sets the value to determine whether the I/O exit implementation requires
 * channel operations for a particular instance to be accessed by a
 * single thread only.
 *
 * <p>
 * For certain I/O implementations it is necessary that resource path
 * operations such as open, read, write, and close are invoked only from a
 * single execution {@link Thread}. When set {@code true}, WMQFTE ensures
 * that the following are invoked on a single thread:
 *
 * <ul>
 * <li>{@link IOExitResourcePath#openForRead(long)} method and all methods of
 * the returned {@link IOExitChannel} instance.</li>
 * <li>{@link IOExitResourcePath#openForWrite(boolean)} method and all
 * methods of the returned {@link IOExitChannel} instance.</li>
 * </ul>
 *
 * <p>
 * This has a slight performance impact, hence enable single-threaded channel
 * I/O only when absolutely necessary.
 *
 * <p>
 * The default is {@code false}. The I/O exit should call this method when
 * required to change this value.
 *
 * @param requiresSingleThreadedChannelIO
 *      {@code true} if {@link IOExitChannel} instances are to be
 *      accessed by a single thread only.
 */
public void setRequiresSingleThreadedChannelIO(boolean requiresSingleThreadedChannelIO) {
    this.requiresSingleThreadedChannelIO = requiresSingleThreadedChannelIO;
}
}

```

Tareas relacionadas

[Utilización de salidas de usuario de E/S de transferencia de MFT](#)

[Personalización de MFT con salidas de usuario](#)

Interfaz *IOExitRecordChannel.java*

IOExitRecordChannel.java

```
/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * © Copyright IBM Corp. 2011, 2026. All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with
 * IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.io.IOException;
import java.nio.ByteBuffer;

/**
 * Represents a channel that enables records of data to be read from or written
 * to an {@link IOExitRecordResourcePath} resource.
 * <p>
 * This is an extension of the {@link IOExitChannel} interface such that the
 * {@link #read(java.nio.ByteBuffer)} and {@link #write(java.nio.ByteBuffer)}
 * methods are expected to deal in whole records of data only. That is, the
 * {@link java.nio.ByteBuffer} returned from the read method and passed to the
 * write method is assumed to contain one or more complete records.
 */
public interface IOExitRecordChannel extends IOExitChannel {

    /**
     * Reads records from this channel into the given buffer, starting at this
     * channel's current position, and updates the current position by the
     * amount of data read.
     * <p>
     * Record data is copied into the buffer starting at its current position
     * and up to its limit. On return, the buffer's position is updated to
     * reflect the number of bytes read.
     * <p>
     * Only whole records are copied into the buffer.
     * <p>
     * For a fixed-record-format resource, this might be multiple records. The
     * amount of data in the return buffer does not necessarily need to be a
     * multiple of the record length, but the last record is still to be treated
     * as a complete record and padded as required by the caller.
     * <p>
     * For a variable-format resource, this is a single whole record of a size
     * corresponding to the amount of return data or multiple whole records with
     * all except the last being treated as records of maximum size.
     *
     * @param buffer
     *      The buffer that the record data is to be copied into.
     * @return The number of bytes read, which might be zero, or -1 if the end of
     *      data has been reached.
     * @throws RecoverableIOException
     *      If a recoverable problem occurs while reading the data. For a
     *      WMQFTE transfer this means that it will attempt to recover.
     * @throws IOException
     *      If some other I/O problem occurs, for example, if the passed
     *      buffer is insufficient to contain at least one complete
     *      record). For a WMQFTE transfer this means that it will be
     *      failed.
     */
    int read(ByteBuffer buffer) throws RecoverableIOException, IOException;

    /**
     * Writes records to this channel from the given buffer, starting at this
     * channel's current position, and updates the current position by the
     * amount of data written. The channel's resource is grown to accommodate
     * the data, if necessary.
     * <p>
     * Record data is copied from the buffer starting at its current position
     * and up to its limit. On return, the buffer's position is updated to
     * reflect the number of bytes written.
     * <p>
     * The buffer is expected to contain only whole records.
     * <p>
     * For a fixed-record-format resource, this might be multiple records and if
     */
}
```

```

* there is insufficient data in the buffer for a complete record, the
* record is to be padded as required to complete the record.
* <p>
* For a variable-record format resource the buffer is normally expected to
* contain a single record of length corresponding to the amount of data
* within the buffer. However, if the amount of data within the buffer
* exceeds the maximum record length, the implementation can either:
* <ol>
* <li>throw an {@link IOException} indicating that it cannot handle the
* situation.</li>
* <li>Consume a record's worth of data from the buffer, leaving the remaining
* data within the buffer.</li>
* <li>Consume all the buffer data and just write what it can to the current
* record. This effectively truncates the data.</li>
* <li>Consume all the buffer data and write to multiple records.</li>
* </ol>
*
* @param buffer
*         The buffer containing the data to be written.
* @return The number of bytes written, which might be zero.
* @throws RecoverableIOException
*         If a recoverable problem occurs while writing the data. For a
*         WMQFTE transfer this means that it will attempt to recover.
* @throws IOException
*         If some other I/O problem occurs. For a WMQFTE transfer this
*         means that it will be failed.
*/
int write(ByteBuffer buffer) throws RecoverableIOException, IOException;
}

```

Tareas relacionadas

[Utilización de salidas de usuario de E/S de transferencia de MFT](#)

[Personalización de MFT con salidas de usuario](#)

Interfaz IOExitRecordResourcePath.java

IOExitRecordResourcePath.java

```

/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * © Copyright IBM Corp. 2011, 2026. All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with
 * IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.io.IOException;

/**
 * Represents a path that denotes a record-oriented data resource (for example,
 * a z/OS data set). It allows the data to be located, the record format to be
 * understood, and {@link IOExitRecordChannel} instances to be created for read
 * or write operations.
 */
public interface IOExitRecordResourcePath extends IOExitResourcePath {

    /**
     * Record formats for record-oriented resources.
     */
    public enum RecordFormat {
        FIXED, VARIABLE
    }

    /**
     * Obtains the record length for records that are maintained by the resource
     * denoted by this abstract path.
     * <p>
     * For a resource with fixed-length records, the data for each record read

```

```

* and written is assumed to be this length.
* <p>
* For a resource with variable-length records, this is the maximum length
* for a record's data.
* <p>
* This method should return a value greater than zero, otherwise it can
* result in the failure of a WMQFTE transfer that involves this abstract
* path.
*
* @return The record length, in bytes, for records maintained by the
*         resource.
*/
int getRecordLength();

/**
* Obtains record format, as a {@link RecordFormat} instance, for records
* that are maintained by the resource denoted by this abstract path.
*
* @return A {@link RecordFormat} instance for the record format for records
*         that are maintained by the resource denoted by this abstract
*         path.
*/
RecordFormat getRecordFormat();

/**
* Opens a {@link IOExitRecordChannel} instance for reading data from the
* resource denoted by this abstract path. The current data byte position
* for the resource is expected to be the passed position value, such that
* when {@link IOExitRecordChannel#read(java.nio.ByteBuffer)} is called,
* data starting from that position is read.
* <p>
* Note that the data byte read position will be on a record boundary.
*
* @param position
*         The required data byte read position.
* @return A new {@link IOExitRecordChannel} instance allowing data to be
*         read from the resource denoted by this abstract path.
* @throws RecoverableIOException
*         If a recoverable problem occurs while attempting to open the
*         resource for reading. This means that WMQFTE can attempt to
*         recover the transfer.
* @throws IOException
*         If some other I/O problem occurs.
*/
IOExitRecordChannel openForRead(long position)
    throws RecoverableIOException, IOException;

/**
* Opens a {@link IOExitRecordChannel} instance for writing data to the
* resource denoted by this abstract path. Writing of data, using the
* {@link IOExitRecordChannel#write(java.nio.ByteBuffer)} method, starts at
* either the beginning of the resource or end of the current data for the
* resource, depending on the specified append parameter.
*
* @param append
*         When {@code true} indicates that data written to the resource
*         should be appended to the end of the current data. When
*         {@code false} indicates that writing of data is to start at
*         the beginning of the resource; any existing data is lost.
* @return A new {@link IOExitRecordChannel} instance allowing data to be
*         written to the resource denoted by this abstract path.
* @throws RecoverableIOException
*         If a recoverable problem occurs while attempting to open the
*         resource for writing. This means that WMQFTE can attempt to
*         recover the transfer.
* @throws IOException
*         If some other I/O problem occurs.
*/
IOExitRecordChannel openForWrite(boolean append)
    throws RecoverableIOException, IOException;
}

```

Tareas relacionadas

Utilización de salidas de usuario de E/S de transferencia de MFT

[Personalización de MFT con salidas de usuario](#)

Interfaz IOExitResourcePath.java

IOExitResourcePath.java

```
/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * © Copyright IBM Corp. 2011, 2026. All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with
 * IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.io.IOException;

/**
 * Represents a path that denotes a data resource (for example, a file,
 * directory, or group of database records). It allows the data to be located
 * and {@link IOExitChannel} instances to be created for read or write
 * operations.
 * <p>
 * There are two types of data resources as follows:
 * <ul>
 * <li>Directory - a container for other data resources. The
 * {@link #isDirectory()} method returns {@code true} for these.</li>
 * <li>File - a data container. This allows data to be read from or written to
 * it. The {@link #isFile()} method returns {@code true} for these.</li>
 * </ul>
 */
public interface IOExitResourcePath extends IOExitPath {

    /**
     * Creates a new {@link IOExitResourcePath} instance for a child path of the
     * resource denoted by this abstract path.
     * <p>
     * For example, with a UNIX-style path, {@code
     * IOExitResourcePath("/home/fteuser/test").newPath("subtest")} could be
     * equivalent to: {@code IOExitResourcePath("/home/fteuser/test/subtest")}
     *
     * @param child
     *     The child path name.
     * @return A new {@link IOExitResourcePath} instance that represents a child
     *     of this path.
     */
    IOExitResourcePath newPath(final String child);

    /**
     * Creates the directory path for the resource denoted by this abstract
     * path, including any necessary but nonexistent parent directories. If the
     * directory path already exists, this method has no effect.
     * <p>
     * If this operation fails, it might have succeeded in creating some of the
     * necessary parent directories.
     *
     * @throws IOException
     *     If the directory path cannot be fully created, when it does
     *     not already exist.
     */
    void makePath() throws IOException;

    /**
     * Obtains the canonical path of the abstract path as a {@link String}.
     * <p>
     * A canonical path is defined as being absolute and unique. For example,
     * the path can be represented as UNIX-style relative path: {@code
     * test/file.txt} but the absolute and unique canonical path representation
     * is: {@code /home/fteuser/test/file.txt}
     *
     * @return The canonical path as a {@link String}.
     * @throws IOException
     *     If the canonical path cannot be determined for any reason.
     */
    String getCanonicalPath() throws IOException;

    /**
     * Tests if this abstract path is an absolute path.
     */
}
```



```

* <p>
* For example, a UNIX-style path, {@code /home/fteuser/test} is an absolute
* path, whereas {@code fteuser/test} is not.
*
* @return {@code true} if this abstract path is an absolute path, {@code
*         false} otherwise.
*/
boolean isAbsolute();

/**
* Tests if the resource denoted by this abstract path exists.
*
* @return {@code true} if the resource denoted by this abstract path
*         exists, {@code false} otherwise.
* @throws IOException
*         If the existence of the resource cannot be determined for any
*         reason.
*/
boolean exists() throws IOException;

/**
* Tests whether the calling application can read the resource denoted by
* this abstract path.
*
* @return {@code true} if the resource for this path exists and can be
*         read, {@code false} otherwise.
* @throws IOException
*         If a problem occurs while attempting to determine if the
*         resource can be read.
*/
boolean canRead() throws IOException;

/**
* Tests whether the calling application can modify the resource denoted by
* this abstract path.
*
* @return {@code true} if the resource for this path exists and can be
*         modified, {@code false} otherwise.
* @throws IOException
*         If a problem occurs while attempting to determine if the
*         resource can be modified.
*/
boolean canWrite() throws IOException;

/**
* Tests whether the specified user is permitted to read the resource
* denoted by this abstract path.
* <p>
* When WMQFTE invokes this method, the user identifier is the MQMD user
* identifier for the requesting transfer.
*
* @param userId
*         User identifier to test for access.
* @return {@code true} if the resource for this abstract path exists and is
*         permitted to be read by the specified user, {@code false}
*         otherwise.
* @throws IOException
*         If a problem occurs while attempting to determine if the user
*         is permitted to read the resource.
*/
boolean readPermitted(String userId) throws IOException;

/**
* Tests whether the specified user is permitted to modify the resource
* denoted by this abstract path.
* <p>
* When WMQFTE invokes this method, the user identifier is the MQMD user
* identifier for the requesting transfer.
*
* @param userId
*         User identifier to test for access.
* @return {@code true} if the resource for this abstract path exists and is
*         permitted to be modified by the specified user, {@code false}
*         otherwise.
* @throws IOException
*         If a problem occurs while attempting to determine if the user
*         is permitted to modify the resource.
*/
boolean writePermitted(String userId) throws IOException;

/**
* Tests if the resource denoted by this abstract path is a directory-type

```

```

* resource.
*
* @return {@code true} if the resource denoted by this abstract path is a
*         directory type resource, {@code false} otherwise.
*/
boolean isDirectory();

/**
 * Creates the resource denoted by this abstract path, if it does not
 * already exist.
 *
 * @return {@code true} if the resource does not exist and was successfully
 *         created, {@code false} if the resource already existed.
 * @throws RecoverableIOException
 *         If a recoverable problem occurs while attempting to create
 *         the resource. This means that WMQFTE can attempt to recover
 *         the transfer.
 * @throws IOException
 *         If some other I/O problem occurs.
 */
boolean createNewPath() throws RecoverableIOException, IOException;

/**
 * Tests if the resource denoted by this abstract path is a file-type
 * resource.
 *
 * @return {@code true} if the resource denoted by this abstract path is a
 *         file type resource, {@code false} otherwise.
 */
boolean isFile();

/**
 * Obtains the last modified time for the resource denoted by this abstract
 * path.
 * <p>
 * This time is measured in milliseconds since the epoch (00:00:00 GMT,
 * January 1, 1970).
 *
 * @return The last modified time for the resource denoted by this abstract
 *         path, or a value of 0L if the resource does not exist or a
 *         problem occurs.
 */
long lastModified();

/**
 * Deletes the resource denoted by this abstract path.
 * <p>
 * If the resource is a directory, it must be empty for the delete to work.
 *
 * @throws IOException
 *         If the delete of the resource fails for any reason.
 */
void delete() throws IOException;

/**
 * Renames the resource denoted by this abstract path to the specified
 * destination abstract path.
 * <p>
 * The rename should still be successful if the resource for the specified
 * destination abstract path already exists and it is possible to replace
 * it.
 *
 * @param destination
 *         The new abstract path for the resource denoted by this
 *         abstract path.
 * @throws IOException
 *         If the rename of the resource fails for any reason.
 */
void renameTo(IOExceptionResourcePath destination) throws IOException;

/**
 * Creates a new path to use for writing to a temporary resource that did
 * not previously exist.
 * <p>
 * The implementation can choose the abstract path name for the temporary
 * resource. However, for clarity and problem diagnosis, the abstract path
 * name for the temporary resource should be based on this abstract path
 * name with the specified suffix appended and additional characters to make
 * the path unique (for example, sequence numbers), as required.
 * <p>
 * When WMQFTE transfers data to a destination it normally attempts to first
 * write to a temporary resource then on transfer completion renames the

```

```

* temporary resource to the required destination. This method is called by
* WMQFTE to create a new temporary resource path. The returned path should
* be new and the resource should not previously exist.
*
* @param suffix
*         Recommended suffix to use for the generated temporary path.
*
* @return A new {@link IOExitResourcePath} instance for the temporary
*         resource path, that did not previously exist.
* @throws RecoverableIOException
*         If a recoverable problem occurs whilst attempting to create
*         the temporary resource. This means that WMQFTE can attempt to
*         recover the transfer.
* @throws IOException
*         If some other I/O problem occurs.
*/
IOExitResourcePath createTempPath(String suffix)
    throws RecoverableIOException, IOException;

/**
* Opens a {@link IOExitChannel} instance for reading data from the resource
* denoted by this abstract path. The current data byte position for the
* resource is expected to be the passed position value, such that when
* {@link IOExitChannel#read(java.nio.ByteBuffer)} is called, data starting
* from that position is read.
*
* @param position
*         The required data byte read position.
* @return A new {@link IOExitChannel} instance allowing data to be read
*         from the resource denoted by this abstract path.
* @throws RecoverableIOException
*         If a recoverable problem occurs while attempting to open the
*         resource for reading. This means that WMQFTE can attempt to
*         recover the transfer.
* @throws IOException
*         If some other I/O problem occurs.
*/
IOExitChannel openForRead(long position) throws RecoverableIOException,
    IOException;

/**
* Opens a {@link IOExitChannel} instance for writing data to the resource
* denoted by this abstract path. Writing of data, using the
* {@link IOExitChannel#write(java.nio.ByteBuffer)} method, starts at either
* the beginning of the resource or end of the current data for the
* resource, depending on the specified append parameter.
*
* @param append
*         When {@code true} indicates that data written to the resource
*         should be appended to the end of the current data. When
*         {@code false} indicates that writing of data is to start at
*         the beginning of the resource; any existing data is lost.
* @return A new {@link IOExitChannel} instance allowing data to be written
*         to the resource denoted by this abstract path.
* @throws RecoverableIOException
*         If a recoverable problem occurs whilst attempting to open the
*         resource for writing. This means that WMQFTE can attempt to
*         recover the transfer.
* @throws IOException
*         If some other I/O problem occurs.
*/
IOExitChannel openForWrite(boolean append) throws RecoverableIOException,
    IOException;

/**
* Tests if the resource denoted by this abstract path is in use by another
* application. Typically, this is because another application has a lock on
* the resource either for shared or exclusive access.
*
* @return {@code true} if resource denoted by this abstract path is in use
*         by another application, {@code false} otherwise.
*/
boolean inUse();

/**
* Obtains a {@link IOExitProperties} instance for properties associated
* with the resource denoted by this abstract path.
*
* <p>
* WMQFTE will read these properties to govern how a transfer behaves when
* interacting with the resource.
*
* @return A {@link IOExitProperties} instance for properties associated

```

```

    *           with the resource denoted by this abstract path.
    */
    IOExitProperties getProperties();
}

```

Tareas relacionadas

[Utilización de salidas de usuario de E/S de transferencia de MFT](#)
[Personalización de MFT con salidas de usuario](#)

Interfaz *IOExitWildcardPath.java*

IOExitWildcardPath.java

```

/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * Copyright IBM Corp. 2011, 2026. All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with
 * IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

/**
 * Represents a path that denotes a wildcard. This can be used to match multiple
 * resource paths.
 */
public interface IOExitWildcardPath extends IOExitPath {

```

Tareas relacionadas

[Utilización de salidas de usuario de E/S de transferencia de MFT](#)
[Personalización de MFT con salidas de usuario](#)

Interfaz *MonitorExit.java*

MonitorExit.java

```

/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * Copyright IBM Corp. 2009, 2026. All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with
 * IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.util.Map;

/**
 * An interface that is implemented by classes that want to be invoked as part of
 * user exit routine processing. This interface defines a method that will be
 * invoked immediately prior to starting a task as the result of a monitor trigger
 */
public interface MonitorExit {

    /**
     * Invoked immediately prior to starting a task as the result of a monitor
     * trigger.
     *
     * @param environmentMetaData
     */

```

```

*      meta data about the environment in which the implementation
*      of this method is running. This information can only be read,
*      it cannot be updated by the implementation. The constant
*      defined in <code>EnvironmentMetaDataConstants</code> class can
*      be used to access the data held by this map.
*
* @param monitorMetaData
*      meta data to associate with the monitor. The meta data passed
*      to this method can be altered, and the changes will be
*      reflected in subsequent exit routine invocations. This map
*      also contains keys with IBM reserved names. These entries are
*      defined in the <code>MonitorMetaDataConstants</code> class and
*      have special semantics. The values of the IBM reserved names
*      cannot be modified by the exit
*
* @param taskDetails
*      An XML String representing the task to be executed as a result of
*      the monitor triggering. This XML string may be modified by the
*      exit
*
* @return    a monitor exit result object which is used to determine if the
*            task should proceed, or be cancelled.
*/
MonitorExitResult onMonitor(Map<String, String> environmentMetaData,
                           Map<String, String> monitorMetaData,
                           Reference<String> taskDetails);
}

```

Tareas relacionadas

[Supervisión de recursos de MFT](#)

[Personalización de MFT con salidas de usuario](#)

Referencia relacionada

[“Interfaz SourceTransferStartExit.java” en la página 2207](#)

[“Interfaz SourceTransferEndExit.java” en la página 2205](#)

[“Interfaz DestinationTransferStartExit.java” en la página 2182](#)

[“Interfaz DestinationTransferEndExit.java” en la página 2181](#)

[“Interfaz ProtocolBridgeCredentialExit.java” en la página 2201](#)

Interfaz ProtocolBridgeCredentialExit.java

ProtocolBridgeCredentialExit.java

```

/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * © Copyright IBM Corp. 2008, 2026. All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with
 * IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.util.Map;

/**
 * An interface that is implemented by classes that are to be invoked as part of
 * user exit routine processing. This interface defines methods that will
 * be invoked by a protocol bridge agent to map the MQ user ID of the transfer to credentials
 * that are to be used to access the protocol server.
 * There will be one instance of each implementation class per protocol bridge agent. The methods
 * can be called from different threads so the methods must be synchronized.
 */
public interface ProtocolBridgeCredentialExit {

    /**
     * Invoked once when a protocol bridge agent is started. It is intended to initialize

```

```

    * any resources that are required by the exit
    *
    * @param bridgeProperties
    *         The values of properties defined for the protocol bridge.
    *         These values can only be read, they cannot be updated by
    *         the implementation.
    *
    * @return  true if the initialization is successful and false if unsuccessful
    *         If false is returned from an exit the protocol bridge agent will not
    *         start
    */

    public boolean initialize(final Map<String> bridgeProperties);

    /**
     * Invoked once for each transfer to map the MQ user ID in the transfer message to the
     * credentials to be used to access the protocol server
     *
     * @param mqUserId The MQ user ID from which to map to the credentials to be used
     *                access the protocol server
     * @return  A credential exit result object that contains the result of the map and
     *          the credentials to use to access the protocol server
     */

    public CredentialExitResult mapMQUserId(final String mqUserId);

    /**
     * Invoked once when a protocol bridge agent is shutdown. It is intended to release
     * any resources that were allocated by the exit
     *
     * @param bridgeProperties
     *         The values of properties defined for the protocol bridge.
     *         These values can only be read, they cannot be updated by
     *         the implementation.
     *
     * @return
     */

    public void shutdown(final Map<String> bridgeProperties);
}

```

Tareas relacionadas

[Personalización de MFT con salidas de usuario](#)

[Correlacionar credenciales para un servidor de archivos utilizando clases de salida](#)

Interfaz *ProtocolBridgeCredentialExit2.java*

ProtocolBridgeCredentialExit2.java

```

/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * © Copyright IBM Corp. 2011, 2026. All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with
 * IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

/**
 * An interface that is implemented by classes that are invoked as part of user
 * exit routine processing. This interface defines methods that are invoked by a
 * protocol bridge agent to map the MQ user ID of the transfer to credentials
 * used to access a specified protocol bridge server. There will be one instance
 * of each implementation class for each protocol bridge agent. The methods can
 * be called from different threads so the methods must be synchronized.
 */
public interface ProtocolBridgeCredentialExit2 extends
    ProtocolBridgeCredentialExit {

```

```

/**
 * Invoked once for each transfer to map the MQ user ID in the transfer
 * message to the credentials used to access a specified protocol server.
 *
 * @param endPoint
 *         Information that describes the protocol server to be accessed.
 * @param mqUserId
 *         The MQ user ID from which to map the credentials used to
 *         access the protocol server.
 * @return A {@link CredentialExitResult} instance that contains the result
 *         of the map and the credentials to use to access the protocol
 *         server.
 */
public CredentialExitResult mapMQUserId(
    final ProtocolServerEndPoint endPoint, final String mqUserId);
}

```

Tareas relacionadas

[Personalización de MFT con salidas de usuario](#)

[Correlacionar credenciales para un servidor de archivos utilizando clases de salida](#)

Interfaz ProtocolBridgePropertiesExit2.java

ProtocolBridgePropertiesExit2.java

```

/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * © Copyright IBM Corp. 2011, 2026. All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with
 * IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.util.Map;
import java.util.Properties;

/**
 * An interface that is implemented by classes that are to be invoked as part of
 * user exit routine processing. This interface defines methods that will be
 * invoked by a protocol bridge agent to look up properties for protocol servers
 * that are referenced in transfers.
 * <p>
 * There will be one instance of each implementation class for each protocol
 * bridge agent. The methods can be called from different threads so the methods
 * must be synchronised.
 */
public interface ProtocolBridgePropertiesExit2 {

    /**
     * Invoked once when a protocol bridge agent is started. It is intended to
     * initialize any resources that are required by the exit.
     *
     * @param bridgeProperties
     *         The values of properties defined for the protocol bridge.
     *         These values can only be read, they cannot be updated by the
     *         implementation.
     * @return {@code true} if the initialization is successful and {@code
     *         false} if unsuccessful. If {@code false} is returned from an exit
     *         the protocol bridge agent will not start.
     */
    public boolean initialize(final Map<String, String> bridgeProperties);

    /**
     * Invoked when the Protocol Bridge needs to access the protocol bridge credentials XML file.
     *
     * @return a {@link String} object giving the location of the ProtocolBridgeCredentials.xml
     */
    public String getCredentialLocation ();
}

```

```

/**
 * Obtains a set of properties for the specified protocol server name.
 * <p>
 * The returned {@link Properties} must contain entries with key names
 * corresponding to the constants defined in
 * {@link ProtocolServerPropertyConstants} and in particular must include an
 * entry for all appropriate constants described as required.
 *
 * @param protocolServerName
 *     The name of the protocol server whose properties are to be
 *     returned. If a null or a blank value is specified, properties
 *     for the default protocol server are to be returned.
 * @return The {@link Properties} for the specified protocol server, or null
 *     if the server cannot be found.
 */
public Properties getProtocolServerProperties(
    final String protocolServerName);

/**
 * Invoked once when a protocol bridge agent is shut down. It is intended to
 * release any resources that were allocated by the exit.
 *
 * @param bridgeProperties
 *     The values of properties defined for the protocol bridge.
 *     These values can only be read, they cannot be updated by the
 *     implementation.
 */
public void shutdown(final Map<String, String> bridgeProperties);
}

```

Tareas relacionadas

[ProtocolBridgePropertiesExit: buscar propiedades de servidor de archivos de protocolo](#)

[Personalización de MFT con salidas de usuario](#)

[Correlacionar credenciales para un servidor de archivos utilizando clases de salida](#)

Clase *SourceFileExitFileSpecification.java*

SourceFileExitFileSpecification.java

```

/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * © Copyright IBM Corp. 2012, 2026. All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with
 * IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.util.Map;

/**
 * A specification of the file names to use for a file transfer, as evaluated by the
 * agent acting as the source of the transfer.
 */
public final class SourceFileExitFileSpecification {

    private final String sourceFileSpecification;
    private final String destinationFileSpecification;
    private final Map<String, String> sourceFileMetaData;
    private final Map<String, String> destinationFileMetaData;

    /**
     * Constructor. Creates a source file exit file specification.
     *
     * @param sourceFileSpecification
     *     the source file specification to associate with the source file
     *     exit file specification.
     *
     * @param destinationFileSpecification
     *     the destination file specification to associate with the
     */
}

```



```

        *           source file exit file specification.
        *
        * @param sourceFileMetaData
        *           the source file meta data.
        *
        * @param destinationFileMetaData
        *           the destination file meta data .
        */
public SourceFileExitFileSpecification(final String sourceFileSpecification,
                                       final String destinationFileSpecification,
                                       final Map<String, String> sourceFileMetaData,
                                       final Map<String, String> destinationFileMetaData) {
    this.sourceFileSpecification = sourceFileSpecification;
    this.destinationFileSpecification = destinationFileSpecification;
    this.sourceFileMetaData = sourceFileMetaData;
    this.destinationFileMetaData = destinationFileMetaData;
}

/**
 * Returns the destination file specification.
 *
 * @return the destination file specification. This represents the location,
 *         on the agent acting as the destination for the transfer, where the
 *         file should be written. Exit routines installed into the agent
 *         acting as the destination for the transfer may override this value.
 */
public String getDestination() {
    return destinationFileSpecification;
}

/**
 * Returns the source file specification.
 *
 * @return the source file specification. This represents the location where
 *         the file data will be read from.
 */
public String getSource() {
    return sourceFileSpecification;
}

/**
 * Returns the file meta data that relates to the source file specification.
 *
 * @return the file meta data that relates to the source file specification.
 */
public Map<String, String> getSourceFileMetaData() {
    return sourceFileMetaData;
}

/**
 * Returns the file meta data that relates to the destination file specification.
 *
 * @return the file meta data that relates to the destination file specification.
 */
public Map<String, String> getDestinationFileMetaData() {
    return destinationFileMetaData;
}
}

```

Conceptos relacionados

“Metadatos para salidas de usuario de MFT” en la página 2168

Hay tres tipos diferentes de metadatos que se pueden suministrar a las rutinas de salida para Managed File Transfer: entorno, transferencia y metadatos de archivo. Estos metadatos se presentan como mapas de pares de clave-valor Java.

Interfaz *SourceTransferEndExit.java*

SourceTransferEndExit.java

```

/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * © Copyright IBM Corp. 2008, 2026. All Rights Reserved.

```

```

*
* US Government Users Restricted Rights - Use, duplication or
* disclosure restricted by GSA ADP Schedule Contract with
* IBM Corp.
*/
package com.ibm.wmqfte.exitpoint.api;

/**
 * An interface that is implemented by classes that want to be invoked as part of
 * user exit routine processing. This interface defines a method that will be
 * invoked immediately after completing a transfer on the agent acting as the
 * source of the transfer.
 */
public interface SourceTransferEndExit {

    /**
     * Invoked immediately after the completion of a transfer on the agent acting as
     * the source of the transfer.
     *
     * @param transferExitResult
     *        a result object reflecting whether or not the transfer completed
     *        successfully.
     *
     * @param sourceAgentName
     *        the name of the agent acting as the source of the transfer.
     *        This is the name of the agent that the implementation of this
     *        method will be invoked from.
     *
     * @param destinationAgentName
     *        the name of the agent acting as the destination of the
     *        transfer.
     *
     * @param environmentMetaData
     *        meta data about the environment in which the implementation
     *        of this method is running. This information can only be read,
     *        it cannot be updated by the implementation. The constants
     *        defined in <code>EnvironmentMetaDataConstants</code> class can
     *        be used to access the data held by this map.
     *
     * @param transferMetaData
     *        meta data to associate with the transfer. The information can
     *        only be read, it cannot be updated by the implementation. This
     *        map may also contain keys with IBM reserved names. These
     *        entries are defined in the <code>TransferMetaDataConstants</code>
     *        class and have special semantics.
     *
     * @param fileResults
     *        a list of file transfer result objects that describe the source
     *        file name, destination file name and result of each file transfer
     *        operation attempted.
     *
     * @return
     *        an optional description to enter into the log message describing
     *        transfer completion. A value of <code>null</code> can be used
     *        when no description is required.
     */
    String onSourceTransferEnd(TransferExitResult transferExitResult,
                               String sourceAgentName,
                               String destinationAgentName,
                               Map<String, String>environmentMetaData,
                               Map<String, String>transferMetaData,
                               List<FileTransferResult>fileResults);

}

```

Tareas relacionadas

[Personalización de MFT con salidas de usuario](#)

Referencia relacionada

[“Interfaz SourceTransferStartExit.java” en la página 2207](#)

[“Interfaz DestinationTransferStartExit.java” en la página 2182](#)

[“Interfaz DestinationTransferEndExit.java” en la página 2181](#)

[“Interfaz MonitorExit.java” en la página 2200](#)

[“Interfaz ProtocolBridgeCredentialExit.java” en la página 2201](#)

Interfaz SourceTransferStartExit.java

SourceTransferStartExit.java

```
/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * © Copyright IBM Corp. 2008, 2026. All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with
 * IBM Corp.
 */
package com.ibm.wmqfte.exitpoint.api;

import java.util.List;
import java.util.Map;

/**
 * An interface that is implemented by classes that want to be invoked as part of
 * user exit routine processing. This interface defines a method that will be
 * invoked immediately prior to starting a transfer on the agent acting as the
 * source of the transfer.
 */
public interface SourceTransferStartExit {

    /**
     * Invoked immediately prior to starting a transfer on the agent acting as
     * the source of the transfer.
     *
     * @param sourceAgentName
     *         the name of the agent acting as the source of the transfer.
     *         This is the name of the agent that the implementation of this
     *         method will be invoked from.
     *
     * @param destinationAgentName
     *         the name of the agent acting as the destination of the
     *         transfer.
     *
     * @param environmentMetaData
     *         meta data about the environment in which the implementation
     *         of this method is running. This information can only be read,
     *         it cannot be updated by the implementation. The constants
     *         defined in <code>EnvironmentMetaDataConstants</code> class can
     *         be used to access the data held by this map.
     *
     * @param transferMetaData
     *         meta data to associate with the transfer. The meta data passed
     *         to this method can be altered, and the changes to will be
     *         reflected in subsequent exit routine invocations. This map may
     *         also contain keys with IBM reserved names. These entries are
     *         defined in the <code>TransferMetaDataConstants</code> class and
     *         have special semantics.
     *
     * @param fileSpecs
     *         a list of file specifications that govern the file data to
     *         transfer. The implementation of this method can add entries,
     *         remove entries, or modify entries in this list and the changes
     *         will be reflected in the files transferred.
     *
     * @return
     *         a transfer exit result object which is used to determine if the
     *         transfer should proceed, or be cancelled.
     */
    TransferExitResult onSourceTransferStart(String sourceAgentName,
                                             String destinationAgentName,
                                             Map<String, String> environmentMetaData,
                                             Map<String, String> transferMetaData,
                                             List<SourceFileExitFileSpecification> fileSpecs);
}
```

Tareas relacionadas

[Personalización de MFT con salidas de usuario](#)

Referencia relacionada

[“Clase SourceFileExitFileSpecification.java” en la página 2204](#)

[“Interfaz SourceTransferEndExit.java” en la página 2205](#)

[“Interfaz DestinationTransferStartExit.java” en la página 2182](#)

[“Interfaz DestinationTransferEndExit.java” en la página 2181](#)

[“Interfaz MonitorExit.java” en la página 2200](#)

[“Interfaz ProtocolBridgeCredentialExit.java” en la página 2201](#)

Interfaz TransferExitResult.java

TransferExitResult.java

TransferExitResult.java tiene este aspecto:

```
/*
 * Licensed Materials - Property of IBM
 *
 * "Restricted Materials of IBM"
 *
 * 5724-H72
 *
 * © Copyright IBM Corp. 2008, 2026. All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with
 * IBM Corp.
 */

package com.ibm.wmqfte.exitroutine.api;

/**
 * The result of invoking a transfer exit routine. It is composed of a result
 * code, which determines if the transfer should proceed, and an optional explanatory
 * message. The explanation, if present, is entered into the log message.
 */
public class TransferExitResult {

    private final TransferExitResultCode resultCode;
    private final String explanation;

    /**
     * For convenience, a static "proceed" result with no associated explanation
     * message.
     */
    public static final TransferExitResult PROCEED_RESULT =
        new TransferExitResult(TransferExitResultCode.PROCEED, null);

    /**
     * Constructor. Creates a transfer exit result object with a specified result
     * code and explanation.
     *
     * @param resultCode
     *         The result code to associate with the exit result being created.
     *
     * @param explanation
     *         The explanation to associate with the exit result being created.
     *         A value of <code>null</code> can be specified to indicate no
     *         explanation.
     */
    public TransferExitResult(TransferExitResultCode resultCode, String explanation) {
        this.resultCode = resultCode;
        this.explanation = explanation;
    }

    /**
     * Returns the explanation associated with this transfer exit result.
     *
     * @return the explanation associated with this exit result.
     */
    public String getExplanation() {
        return explanation;
    }
}
```

```

/**
 * Returns the result code associated with this transfer exit result.
 *
 * @return the result code associated with this exit result.
 */
public TransferExitResultCode getResultCode() {
    return resultCode;
}

/**
 * Returns a string representation of this TransferExitResult.
 * @return a string representation of this TransferExitResult.
 */
@Override
public String toString() {
    return "[" + getResultCode() + ", " + getExplanation() + "];"
}
}

```

Tareas relacionadas

[Personalización de MFT con salidas de usuario](#)

Referencia relacionada

[“Interfaz SourceTransferStartExit.java” en la página 2207](#)

[“Interfaz DestinationTransferStartExit.java” en la página 2182](#)

[“Interfaz DestinationTransferEndExit.java” en la página 2181](#)

[“Interfaz MonitorExit.java” en la página 2200](#)

[“Interfaz ProtocolBridgeCredentialExit.java” en la página 2201](#)

Formatos de mensaje para los mensajes que puede colocar en la cola de mandatos del agente de MFT

Estos esquemas XML definen los formatos de los mensajes que se pueden colocar en la cola de mandatos del agente para solicitar que el agente realice una acción. El mensaje XML se puede colocar en la cola de mandatos de agente mediante los mandatos de una línea de mandato o mediante un aplicación.

- [Formato de mensaje de solicitud de transferencia de archivos](#)
- [Formatos de mensajes de solicitud del supervisor de MFT](#)
- [Formato de mensaje de solicitud de ping a un agente MFT](#)
- [Formato de mensaje de respuesta del agente MFT](#)

Referencia de la REST API de mensajería

Información de referencia sobre messaging REST API.

Para obtener más información sobre cómo utilizar messaging REST API, consulte [Mensajería utilizando REST API](#).

Recursos de REST API

Esta colección de temas proporciona información de referencia para cada uno de los recursos de messaging REST API .

Para obtener más información sobre cómo utilizar messaging REST API, consulte [Mensajería utilizando REST API](#).

/mensajería/qmgr/ {qmgrName} /cola/ {queueName} /mensaje

Permite colocar messaging REST API mensajes en una cola, o bien examinar o extraer de forma destructiva mensajes de una cola, utilizando el /messaging/qmgr/{qmgrName}/queue/{queueName}/message recurso.

Nota: La función de segmentación de mensajes no es compatible con el IBM MQ messaging REST API.

POST/messaging/qmgr/{qmgrName}/queue/{queueName}/message

Puede utilizar el método POST HTTP con el `/messaging/qmgr/{qmgrName}/queue/{queueName}/message` recurso para enviar mensajes a la cola especificada en el gestor de colas especificado.

Envía un IBM MQ mensaje que contiene el cuerpo de la solicitud HTTP al gestor de colas y a la cola especificados. El gestor de colas debe estar en la misma máquina que el servidor mqweb. El método solo admite cuerpos de solicitud HTTP basados en texto. Los mensajes se envían como MQSTR o JMS TextMessage mensajes formateados y se colocan utilizando el contexto del usuario actual.

La API REST V3 añade la posibilidad de especificar propiedades de mensajes definidas por el usuario e incluir la prioridad de los mensajes. Los encabezados de solicitud `ibm-mq-md-priority` e `ibm-mq-usr` solo están disponibles con la API REST V3. El encabezado de solicitud `ibm-mq-md-correlationId` tiene un formato diferente en la API REST V3. El encabezado puede ser un ID específico de la aplicación o, si es una cadena codificada, requiere el ID: prefijo. Si su solicitud POST contiene mensajes definidos por el usuario o un ID de correlación específico de la aplicación, el mensaje se formatea como un JMS TextMessage.

- [“URL del recurso” en la página 2210](#)
- [“Cabeceras de solicitud” en la página 2211](#)
- [“Formato del cuerpo de la solicitud” en la página 2213](#)
- [“Requisitos de seguridad” en la página 2213](#)
- [“Códigos de estado de respuesta” en la página 2214](#)
- [“Cabeceras de respuesta” en la página 2214](#)
- [“Formato del cuerpo de la respuesta” en la página 2215](#)
- [“Ejemplos” en la página 2215](#)

URL del recurso

`https://host:port/ibmmq/rest/v1/messaging/qmgr/{qmgrName}/queue/{queueName}/message`

`https://host:port/ibmmq/rest/v2/messaging/qmgr/{qmgrName}/queue/{queueName}/message`

`https://host:port/ibmmq/rest/v3/messaging/qmgr/{qmgrName}/queue/{queueName}/message`

qmgrName

Especifica el nombre del gestor de colas al que conectarse para el envío de mensajes. El gestor de colas debe estar en la misma máquina que el servidor mqweb.

Desde IBM MQ 9.3.3, puede conectarse a un gestor de colas local o a un gestor de colas remoto. El nombre que especifique para **qmgrName** depende de cómo esté configurado su servidor mqweb:

- Si su servidor mqweb está configurado para conectarse únicamente a gestores de colas locales, utilice el nombre del gestor de colas.
- Si su servidor mqweb está configurado para conectarse a gestores de colas remotos, utilice el nombre único del gestor de colas tal y como se especifica en la información de conexión del gestor de colas remoto.

Puede determinar si su servidor mqweb está configurado para conectarse a gestores de colas locales o remotos utilizando el **dspmweb properties** comando y visualizando la **mqRestMessagingConnectionMode** propiedad.

El nombre del gestor de colas es sensible a las mayúsculas y minúsculas.

Si el nombre de gestor de colas incluye una barra inclinada, un punto o un signo de porcentaje, estos caracteres se deben codificar en el URL:

- Una barra inclinada se debe codificar como %2F.
- Un punto se debe codificar como %2E.
- Un signo de porcentaje se debe codificar como %25.

queueName

Especifica el nombre de la cola en la que se colocará el mensaje.

La cola debe definirse como local, remota o un alias del gestor de colas especificado; también puede hacer referencia a una cola en clúster.

El nombre de la cola distingue entre mayúsculas y minúsculas.

Si el nombre de la cola incluye una barra inclinada o un signo de porcentaje, estos caracteres deben codificarse con e URL :

- Una barra inclinada, /, debe codificarse como %2F.
- Un signo de porcentaje, %, debe codificarse como %25.

Puede utilizar HTTP en lugar de HTTPS si habilita las conexiones HTTP. Para obtener más información sobre la habilitación de HTTP, consulte [Configuración de puertos HTTP y HTTPS](#).

Cabeceras de solicitud

Los siguientes encabezados deben enviarse con la solicitud:

Autorización

Se debe enviar esta cabecera si utiliza autenticación básica. Para obtener más información, consulte [Utilización de la autenticación básica HTTP con REST API](#).

Content-Type

Este encabezado debe enviarse con uno de los siguientes valores:

- text/plain; charset=utf-8
- text/html; charset=utf-8
- text/xml; charset=utf-8
- application/json; charset=utf-8
- application/xml; charset=utf-8

ibm-mq-rest-csrf-token

Esta cabecera se debe establecer, pero el valor puede ser cualquiera, incluso puede estar en blanco.

Los siguientes encabezados pueden enviarse opcionalmente con la solicitud:

Accept-Language

Este encabezado especifica el idioma requerido para cualquier excepción o mensaje de error devuelto en el cuerpo del mensaje de respuesta.

REST API V2 REST API V1 ibm-mq-md-correlationId

Este encabezado establece el ID de correlación del mensaje creado. El encabezado se especifica como una cadena codificada en hexadecimal de 48 caracteres, que representa 24 bytes. No anteponga el valor con "ID: ", la API REST añade esa cadena automáticamente.

Por ejemplo: ibm-mq-md-correlationId:
414d5120514d4144455620202020202067d8bf5923582e02

REST API V3 ibm-mq-md-correlationId

Este encabezado establece el ID de correlación del mensaje creado. El identificador de correlación puede adoptar una de las siguientes formas:

- Una cadena codificada en hexadecimal de 48 caracteres, que representa 24 bytes, precedida por la cadena "ID: ". Por ejemplo: ibm-mq-md-correlationId:
ID: 414d5120514d4144455620202020202067d8bf5923582e02

- Un valor específico de la aplicación. El valor es una cadena específica de la aplicación: `ibm-mq-md-correlationId: My-Custom-CorrelId`

Si se especifica este tipo de ID de correlación, el destino del mensaje se establece como WMQ_CLIENT_JMS_COMPLIANT y, por lo tanto, incorpora un encabezado « MQRFH2 ».

ibm-mq-md-caducidad

Este encabezado establece la duración de caducidad del mensaje creado. La caducidad de un mensaje comienza desde el momento en que el mensaje llega a la cola. Como resultado, se ignora la latencia de la red. El encabezado debe especificarse como uno de los siguientes valores:

ilimitado

El mensaje no caduca.

Este es el valor predeterminado.

Valor entero

Milisegundos antes de que caduque el mensaje.

Limitado al rango 0 - 99999999900.

ibm-mq-md-persistencia

Este encabezado establece la persistencia del mensaje creado. El encabezado se especifica como uno de los siguientes valores:

nonPersistent

El mensaje no sobrevive a fallos del sistema ni a reinicios del gestor de colas.

Este es el valor predeterminado.

persistentes

El mensaje sobrevive a los fallos del sistema o a los reinicios del gestor de colas.

REST API V3 ibm-mq-md-prioridad

Para la API REST V3, este encabezado establece la prioridad del mensaje creado. El encabezado se especifica como uno de los siguientes valores:

asDestination

El mensaje utiliza la prioridad especificada en el atributo DEFPTY del objeto de cola IBM MQ subyacente.

Este es el valor predeterminado.

Valor entero

Especifique la prioridad real como un número entero en el rango 0-9.

Por ejemplo: `ibm-mq-md-priority: asDestination`

Para la API REST V1 y la API REST V2, la prioridad del mensaje para POST es siempre 4.

ibm-mq-md-replyTo

Este encabezado establece el destino de respuesta para el mensaje creado. El formato del encabezado utiliza la notación estándar de proporcionar la cola de respuesta y un gestor de colas opcional: `replyQueue[@replyQmgr]`

Por ejemplo: `ibm-mq-md-replyTo: myReplyQueue@myReplyQMGr`

REST API V3 ibm-mq-usuario

Este encabezado establece las propiedades definidas por el usuario del mensaje de solicitud.

Las propiedades tienen la siguiente sintaxis:

`ibm-mq-usr: property_name; user_value; user_type`

nombre_propiedad

El nombre de la propiedad de usuario que se está especificando. Debe ser un nombre de propiedad JMS válido.

valor_usuario

El valor de la propiedad.

tipo_de_usuario

El tipo de propiedad:

- boolean (verdadero/falso, MQBOOL)
- byte (entero de 8 bits, MQINT8)
- short (entero de 16 bits, MQINT16)
- integer (entero de 32 bits, MQINT32)
- long (entero de 64 bits, MQINT64)
- float (32 bits reales, MQFLOAT32)
- double (64 bits reales, MQFLOAT64)
- string (cadena citada)

Puedes establecer varias propiedades en un mensaje. Puede especificar varias propiedades separadas por comas en un único encabezado de solicitud ibm-mq-usr, o bien puede utilizar dos o más instancias independientes del encabezado de solicitud ibm-mq-usr.

Por ejemplo, puede establecer varias propiedades definidas por el usuario en un solo encabezado: `ibm-mq-usr: userPropertyA;5;byte,userPropertyB;-10;integer`

O bien, puede establecer varias propiedades definidas por el usuario en instancias separadas del encabezado: `ibm-mq-usr: userPropertyA;5;byte ibm-mq-usr: userPropertyB;-10;integer`

Formato del cuerpo de la solicitud

El cuerpo de la solicitud debe ser texto y utilizar la codificación UTF-8. No se requiere una estructura de texto específica. Se crea un mensaje MQSTR con formato que contiene el texto del cuerpo de la solicitud y se coloca en la cola especificada.

 Si se utilizan las propiedades definidas por el usuario de la API REST V3 o las características de ID de correlación específicas de la aplicación, se crea un JMS `TextMessage` mensaje formateado que contiene el texto del cuerpo de la solicitud y se coloca en la cola especificada.

Para obtener más información, consulte [los ejemplos](#).

Requisitos de seguridad

La persona que llama debe estar autenticada en el servidor mqweb. Los roles MQWebAdmin y MQWebAdminRO no son aplicables para messaging REST API. Para obtener más información sobre la seguridad para REST API, consulte [Seguridad de IBM MQ Console y REST API](#).

Una vez autenticado en el servidor mqweb, el usuario puede utilizar tanto el messaging REST API como el administrative REST API.

El principal de seguridad de la persona que realiza la llamada debe tener la capacidad de enviar mensajes a la cola especificada:

- La cola especificada por la parte `{queueName}` del URL de recurso debe estar habilitada para PUT.
-   Para la cola especificada por la `{queueName}` parte del recurso URL, se debe conceder +PUT autoridad al principal de seguridad del llamante.
-  Para la cola especificada por la `{queueName}` parte del recurso URL, se debe conceder UPDATE acceso al principal de seguridad del llamante.

Este principal de seguridad puede ser el usuario que inició el servidor mqweb o el usuario que ha iniciado sesión en el servidor mqweb. Si el administrador de colas al que se conecta es un administrador de colas remoto, el principal de seguridad podría ser, en su lugar, el usuario proporcionado por las credenciales en la información de conexión del administrador de colas remoto, u otro usuario determinado por las reglas de seguridad del canal. Para obtener más información, consulte [Determinación del principal de seguridad utilizado por el messaging REST API](#).



En AIX, Linux, and Windows, puede otorgar autorización a los principales de seguridad para que utilicen recursos de IBM MQ utilizando el mandato **setmqaut**. Para obtener más información, consulte **setmqaut** (conceder o revocar la autorización).

En z/OS, consulte [Configuración de la seguridad en z/OS](#).

Si utiliza Advanced Message Security (AMS) con la messaging REST API, tenga en cuenta que todos los mensajes se cifran utilizando el contexto del servidor mqweb, no el contexto del usuario que publica el mensaje.

Códigos de estado de respuesta

201

Mensaje creado y enviado correctamente.

400

Se han proporcionado datos no válidos.

Por ejemplo, se especificó un valor de encabezado de solicitud no válido.

401

No está autenticado.

El llamante debe autenticarse en el servidor mqweb y debe ser miembro de uno o más de los roles MQWebAdmin, MQWebAdminRO o MQWebUser. También se debe especificar el `ibm-mq-rest-csrf-token` encabezado. Para obtener más información, consulte [“Requisitos de seguridad” en la página 2213](#).

403

No autorizado.

La persona que llama está autenticada en el servidor mqweb y está asociada a un principal válido. Sin embargo, el director no tiene acceso a todos los recursos IBM MQ necesarios, o a un subconjunto de ellos, o no desempeña esa MQWebUser función. Para obtener más información sobre el acceso necesario, consulte [“Requisitos de seguridad” en la página 2213](#).

404

No existe la cola.

405

La cola está inhibida por PUT.

415

El encabezado o el cuerpo del mensaje es un tipo de medio no compatible.

Por ejemplo, el Content-Type encabezado está configurado con un tipo de medio no compatible.

500

Problema con el servidor o código de error de IBM MQ.

502

El principal de seguridad actual no puede enviar el mensaje, ya que el proveedor de mensajería no admite la función requerida. Por ejemplo, si la ruta de clase del servidor mqweb no es válida.

503

El gestor de colas no se está ejecutando.

Cabeceras de respuesta

Los siguientes encabezados se devuelven con la respuesta:

Idioma del contenido

Especifica el identificador de idioma del mensaje de respuesta en caso de que se produzcan errores o excepciones. Se utiliza junto con el Accept-Language encabezado de solicitud para indicar el idioma requerido para cualquier condición de error o excepción. Se utiliza el servidor mqweb predeterminado si el idioma solicitado no es compatible.

Content-Length

Especifica la longitud del cuerpo de la respuesta HTTP, incluso cuando no hay contenido. Si se realiza correctamente, el valor es cero.

Content-Type

Especifica el tipo de cuerpo de la respuesta. Si se realiza correctamente, el valor es `text/plain; charset=utf-8`. En caso de errores o excepciones, el valor es `application/json; charset=utf-8`.

REST API V2 REST API V1 **ibm-mq-md-messageId**

Especifica el ID de mensaje que se asigna a IBM MQ este mensaje. Al igual que el `ibm-mq-md-correlationId` encabezado de solicitud, se representa como una cadena codificada en hexadecimal de 48 caracteres, que representa 24 bytes.

Por ejemplo:

```
ibm-mq-md-messageId: 414d5120514d4144455620202020202067d8ce5923582f07
```

REST API V3 **ibm-mq-md-messageId**

Especifica el ID de mensaje que se asigna a IBM MQ este mensaje. Al igual que el `ibm-mq-md-correlationId` encabezado de solicitud, se representa como una cadena codificada en hexadecimal de 48 caracteres, que representa 24 bytes, precedida por la cadena `ID:`.

Por ejemplo:

```
ibm-mq-md-messageId: ID:414d5120514d4144455620202020202067d8ce5923582f07
```

ibm-mq-resuelto-qmgr

Especifica el nombre del gestor de colas que se utilizó para la solicitud. Si se utilizó un nombre único en el recurso URL, este encabezado identifica el nombre del gestor de colas asociado a ese nombre único. Si el nombre único utilizado en el recurso URL hace referencia a un grupo de gestores de colas, este encabezado identifica qué gestor de colas dentro del grupo se utilizó.

Formato del cuerpo de la respuesta

El cuerpo de la respuesta está vacío si el mensaje se envía correctamente. Si se produce un error, el cuerpo de la respuesta contiene un mensaje de error. Para obtener más información, consulte [REST API manejo de errores](#).

Ejemplos

Los ejemplos siguientes utilizan el URL del recurso v2. Si está utilizando una versión de IBM MQ anterior a IBM MQ 9.1.5 debe utilizar el URL de recurso v1 en su lugar. Es decir, en el URL de recurso, sustituya v1 donde el URL de ejemplo utiliza v2.

El siguiente ejemplo inicia sesión con un usuario llamado `mquser` con la contraseña `mquser`. En cURL,, la solicitud de inicio de sesión podría parecerse al siguiente Windows ejemplo. El token LTPA se almacena en el `cookiejar.txt` archivo utilizando el `-c` indicador:

```
curl -k "https://localhost:9443/ibmmq/rest/v2/login" -X POST
-H "Content-Type: application/json" --data '{"username":"mquser","password":"mquser"}'
-c c:\cookiejar.txt
```

Una vez que el usuario ha iniciado sesión, el token LTPA y el `ibm-mq-rest-csrf-token` encabezado HTTP se utilizan para autenticar las solicitudes posteriores. El `ibm-mq-rest-csrf-token` puede ser cualquier valor, incluido un espacio en blanco.

- El siguiente Windows ejemplo de cURL envía un mensaje a la cola Q1 en el gestor de QM1colas, utilizando las opciones predeterminadas. El mensaje contiene el texto «¡Hola, mundo!»:

```
curl -k "https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/queue/Q1/message"
-X POST -b c:\cookiejar.txt -H "ibm-mq-rest-csrf-token: token_value"
-H "Content-Type: text/plain; charset=utf-8" --data "Hello World!"
```

- El siguiente Windows ejemplo de cURL envía un mensaje persistente a la cola Q1 en el gestor de QM1colas, con una caducidad de 2 minutos. El mensaje contiene el texto «¡Hola, mundo!» :

```
curl -k "https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/queue/Q1/message"
-X POST -b c:\cookiejar.txt -H "ibm-mq-rest-csrf-token: token_value"
-H "Content-Type: text/plain; charset=utf-8" -H "ibm-mq-md-persistence: persistent"
-H "ibm-mq-md-expiry: 120000" --data "Hello World!"
```

- El siguiente Windows ejemplo de cURL envía un mensaje no persistente a la cola Q1 en el gestor de QM1colas, sin caducidad y con un ID de correlación definido. El mensaje contiene el texto «¡Hola, mundo!» :

```
curl -k "https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/queue/Q1/message"
-X POST -b c:\cookiejar.txt -H "ibm-mq-rest-csrf-token: token_value"
-H "Content-Type: text/plain; charset=utf-8" -H "ibm-mq-md-persistence: nonPersistent"
-H "ibm-mq-md-expiry: unlimited" -H "ibm-mq-md-correlationId:
414d5120514d414445562020202020202067d8b
f5923582e02" --data "Hello World!"
```

GET/messaging/qmgr/{qmgrName}/queue/{queueName}/message

Puede utilizar el método GET HTTP con el /messaging/qmgr/{qmgrName}/queue/{queueName}/message recurso para examinar los mensajes del gestor de colas y la cola asociados.

Busca el primer mensaje disponible del gestor de colas y la cola especificados. El gestor de colas debe estar en la misma máquina que el servidor mqweb. El cuerpo del mensaje se devuelve en el cuerpo de la respuesta HTTP. El mensaje debe tener un formato de MQSTR o JMS TextMessage y se recibe utilizando el contexto del usuario actual.

Todos los mensajes se dejan en la cola y se devuelve un código de estado adecuado a la persona que llama por cualquier mensaje inadecuado. Por ejemplo, un mensaje que no tiene formato MQSTR JMS TextMessage o.

La API REST V3 añade la posibilidad de especificar propiedades de mensajes definidas por el usuario e incluir la prioridad de los mensajes. Los encabezados de respuesta ibm-mq-md-priority e ibm-mq-usr solo están disponibles con la API REST V3. El encabezado de solicitud ibm-mq-md-correlationId tiene un formato diferente en la API REST V3. El encabezado puede ser un identificador específico de la aplicación o, si es una cadena codificada, conserva el ID: prefijo. El encabezado de respuesta y el parámetro de consulta ibm-mq-md-messageId tienen un formato diferente en la API REST V3, ya que conservan el ID: prefijo.

- [“URL del recurso” en la página 2216](#)
- [“Parámetros de consulta opcionales” en la página 2217](#)
- [“Cabeceras de solicitud” en la página 2218](#)
- [“Formato del cuerpo de la solicitud” en la página 2218](#)
- [“Requisitos de seguridad” en la página 2218](#)
- [“Códigos de estado de respuesta” en la página 2219](#)
- [“Cabeceras de respuesta” en la página 2220](#)
- [“Formato del cuerpo de la respuesta” en la página 2222](#)
- [“Ejemplos” en la página 2222](#)

URL del recurso

`https://host:port/ibmmq/rest/v1/messaging/qmgr/{qmgrName}/queue/{queueName}/message`

`https://host:port/ibmmq/rest/v2/messaging/qmgr/{qmgrName}/queue/{queueName}/message`

`https://host:port/ibmmq/rest/v3/messaging/qmgr/{qmgrName}/queue/{queueName}/message`

qmgrName

Especifica el nombre del gestor de colas al que conectarse para el envío de mensajes. El gestor de colas debe estar en la misma máquina que el servidor mqweb.

Desde IBM MQ 9.3.3, puede conectarse a un gestor de colas local o a un gestor de colas remoto. El nombre que especifique para **qmgrName** depende de cómo esté configurado su servidor mqweb:

- Si su servidor mqweb está configurado para conectarse únicamente a gestores de colas locales, utilice el nombre del gestor de colas.
- Si su servidor mqweb está configurado para conectarse a gestores de colas remotos, utilice el nombre único del gestor de colas tal y como se especifica en la información de conexión del gestor de colas remoto.

Puede determinar si su servidor mqweb está configurado para conectarse a gestores de colas locales o remotos utilizando el **dspmweb properties** comando y visualizando la **mqRestMessagingConnectionMode** propiedad.

El nombre del gestor de colas es sensible a las mayúsculas y minúsculas.

Si el nombre de gestor de colas incluye una barra inclinada, un punto o un signo de porcentaje, estos caracteres se deben codificar en el URL:

- Una barra inclinada (/) debe codificarse como %2F.
- Un signo de porcentaje (%) debe codificarse como %25.

queueName

Especifica el nombre de la cola desde la que se buscará el mensaje.

La cola debe definirse como local o como un alias que apunte a una cola local.

El nombre de la cola distingue entre mayúsculas y minúsculas.

Si el nombre de la cola incluye una barra inclinada o un signo de porcentaje, estos caracteres deben codificarse con e URL :

- Una barra inclinada, /, debe codificarse como %2F.
- Un signo de porcentaje, %, debe codificarse como %25.

Puede utilizar HTTP en lugar de HTTPS si habilita las conexiones HTTP. Para obtener más información sobre la habilitación de HTTP, consulte [Configuración de puertos HTTP y HTTPS](#).

Parámetros de consulta opcionales

REST API V2 REST API V1 **correlationId=valor hexadecimal**

Especifica que el método HTTP devuelve el siguiente mensaje con el ID de correlación correspondiente.

hexValue

El parámetro de consulta debe especificarse como una cadena codificada en hexadecimal de 48 caracteres, que representa 24 bytes.

Por ejemplo:

```
../message?correlationId=414d5120514d4144455620202020202067d8bf5923582e02
```

REST API V3 **correlationId=ID: valor hexadecimal o correlation Id=valor específico de la aplicación**

Especifica que el método HTTP devuelve el siguiente mensaje con el ID de correlación correspondiente.

hexValue

El parámetro de consulta debe especificarse como una cadena codificada en hexadecimal de 48 caracteres, que representa 24 bytes, y precedida por la cadena "ID: ".

Por ejemplo:

```
../message?correlationId=ID:414d5120514d4144455620202020202067d8bf5923582e02
```

valor_específico_de_la_aplicación

El parámetro de consulta se puede especificar como una cadena específica de la aplicación.

Por ejemplo:

```
../message?correlationId=My-Custom-CorrelId
```

REST API V2 REST API V1 **messageId=valor hexadecimal**

Especifica que el método HTTP devuelve el siguiente mensaje con el ID de mensaje correspondiente.

hexValue

El parámetro de consulta debe especificarse como una cadena codificada en hexadecimal de 48 caracteres, que representa 24 bytes.

Por ejemplo:

```
../message?messageId=414d5120514d41444556202020202067d8ce5923582f07
```

REST API V3 **messageId=ID: valor hexadecimal**

Especifica que el método HTTP devuelve el siguiente mensaje con el ID de mensaje correspondiente.

hexValue

El parámetro de consulta debe especificarse como una cadena codificada en hexadecimal de 48 caracteres, que representa 24 bytes, y precedida por la cadena "ID: ".

Por ejemplo:

```
../message?messageId=ID:414d5120514d41444556202020202067d8ce5923582f07
```

Cabeceras de solicitud

Los siguientes encabezados deben enviarse con la solicitud:

Autorización

Se debe enviar esta cabecera si utiliza autenticación básica. Para obtener más información, consulte [Utilización de la autenticación básica HTTP con REST API](#).

ibm-mq-rest-csrf-token

Esta cabecera se debe establecer, pero el valor puede ser cualquiera, incluso puede estar en blanco.

Los siguientes encabezados pueden enviarse opcionalmente con la solicitud:

Aceptar conjunto de caracteres

Este encabezado se puede utilizar para indicar qué conjunto de caracteres es aceptable para la respuesta. Si se especifica, este encabezado debe establecerse como UTF-8.

Accept-Language

Este encabezado especifica el idioma requerido para cualquier excepción o mensaje de error devuelto en el cuerpo del mensaje de respuesta.

Formato del cuerpo de la solicitud

Ninguna.

Requisitos de seguridad

La persona que llama debe estar autenticada en el servidor mqweb. Los roles MQWebAdmin y MQWebAdminRO no son aplicables para messaging REST API. Para obtener más información sobre la seguridad para REST API, consulte [Seguridad de IBM MQ Console y REST API](#).

Una vez autenticado en el servidor mqweb, el usuario puede utilizar tanto el messaging REST API como el administrative REST API.

Se debe otorgar al principal de seguridad del proceso de llamada la capacidad de examinar mensajes de la cola especificada:

- La cola especificada por la parte *{queueName}* del URL de recurso debe estar habilitada para BROWSE.
-  **ALW**  **MQ Appliance** Para la cola especificada por la parte *{queueName}* del URL de recurso, se debe otorgar la autorización +GET, +INQ y +BROWSE al principal de seguridad del llamante.
-  **z/OS** Para la cola especificada por la parte *{queueName}* del URL de recurso, UPDATE, debe otorgarse acceso al principal de seguridad del llamante.

Este principal de seguridad puede ser el usuario que inició el servidor mqweb o el usuario que ha iniciado sesión en el servidor mqweb. Si el administrador de colas al que se conecta es un administrador de colas remoto, el principal de seguridad podría ser, en su lugar, el usuario proporcionado por las credenciales en la información de conexión del administrador de colas remoto, u otro usuario determinado por las reglas de seguridad del canal. Para obtener más información, consulte [Determinación del principal de seguridad utilizado por el messaging REST API](#).

 **ALW** En AIX, Linux, and Windows, puede otorgar autorización a los principales de seguridad para que utilicen recursos de IBM MQ utilizando el mandato **setmqaut**. Para obtener más información, consulte **setmqaut** (conceder o revocar la autorización).

En z/OS, consulte [Configuración de la seguridad en z/OS](#).

Códigos de estado de respuesta

200

Mensaje recibido correctamente.

204

No hay mensajes disponibles.

400

Se han proporcionado datos no válidos.

Por ejemplo, se especificó un valor de parámetro de consulta no válido.

401

No está autenticado.

El llamante debe autenticarse en el servidor mqweb y debe ser miembro de uno o más de los roles MQWebAdmin, MQWebAdminRO o MQWebUser. También se debe especificar el `ibm-mq-rest-csrf-token` encabezado. Para obtener más información, consulte [“Requisitos de seguridad” en la página 2218](#).

403

No autorizado.

La persona que llama está autenticada en el servidor mqweb y está asociada a un principal válido. Sin embargo, el director no tiene acceso a todos los recursos IBM MQ necesarios, o a un subconjunto de ellos, o no desempeña esa MQWebUser función. Para obtener más información sobre el acceso necesario, consulte [“Requisitos de seguridad” en la página 2218](#).

404

No existe la cola.

500

Problema con el servidor o código de error de IBM MQ.

501

No se pudo construir la respuesta HTTP.

Por ejemplo, el mensaje recibido tiene un tipo incorrecto, o tiene el tipo correcto pero no se ha podido procesar el cuerpo.

502

El principal de seguridad actual no puede recibir el mensaje, ya que el proveedor de mensajería no admite la función requerida. Por ejemplo, si la ruta de clase del servidor mqweb no es válida.

503

El gestor de colas no se está ejecutando.

Cabeceras de respuesta

Los siguientes encabezados se devuelven con la respuesta:

Idioma del contenido

Especifica el identificador de idioma del mensaje de respuesta en caso de que se produzcan errores o excepciones. Se utiliza junto con el `Accept-Language` encabezado de solicitud para indicar el idioma requerido para cualquier condición de error o excepción. Se utiliza el servidor mqweb predeterminado si el idioma solicitado no es compatible.

Content-Length

Especifica la longitud del cuerpo de la respuesta HTTP, incluso cuando no hay contenido. El valor contiene la longitud (bytes) de los datos del mensaje.

Content-Type

Especifica el tipo de contenido devuelto en el cuerpo de la respuesta del mensaje recibido. Si se realiza correctamente, el valor es `text/plain; charset=utf-8`. En caso de errores o excepciones, el valor es `application/json; charset=utf-8`.

ibm-mq-md-correlationId

Especifica el ID de correlación del mensaje recibido. El encabezado se devuelve si el mensaje recibido contiene un ID de correlación válido. Se representa como una cadena codificada en hexadecimal de 48 caracteres, que representa 24 bytes.

Por ejemplo:

```
ibm-mq-md-correlationId: 414d5120514d4144455620202020202067d8bf5923582e02
```

ibm-mq-md-correlationId

Especifica el ID de correlación del mensaje recibido. El encabezado se devuelve si el mensaje recibido contiene un ID de correlación válido. El identificador de correlación puede adoptar una de las siguientes formas:

- Una cadena codificada en hexadecimal de 48 caracteres, que representa 24 bytes, precedida por la cadena "ID:". Por ejemplo:

```
ibm-mq-md-correlationId: ID:414d5120514d4144455620202020202067d8bf5923582e02
```

- Un valor específico de la aplicación. El valor es una cadena específica de la aplicación:

```
ibm-mq-md-correlationId: My-Custom-CorrelId
```

ibm-mq-md-caducidad

Especifica la duración restante hasta la expiración del mensaje recibido. El encabezado puede tener uno de los siguientes valores:

ilimitado

El mensaje no caduca.

Valor entero

Milisegundos restantes antes de que caduque el mensaje.

ibm-mq-md-messageId

Especifica el ID de mensaje que se asigna a IBM MQ este mensaje. Al igual que el `ibm-mq-md-correlationId` encabezado, se representa como una cadena codificada en hexadecimal de 48 caracteres, que representa 24 bytes.

Por ejemplo:

```
ibm-mq-md-messageId: 414d5120514d4144455620202020202067d8ce5923582f07
```


REST API V3 **ibm-mq-md-messageId**

Especifica el ID de mensaje que se asigna a IBM MQ este mensaje. Al igual que el `ibm-mq-md-correlationId` encabezado, se representa como una cadena codificada en hexadecimal de 48 caracteres, que representa 24 bytes, precedida por la cadena "ID: "

Por ejemplo:

```
ibm-mq-md-messageId: ID:414d5120514d4144455620202020202067d8ce5923582f07
```

ibm-mq-md-persistencia

Especifica la persistencia del mensaje recibido. El encabezado puede tener uno de los siguientes valores:

nonPersistent

El mensaje no sobrevive a fallos del sistema ni a reinicios del gestor de colas.

persistentes

El mensaje sobrevive a los fallos del sistema o a los reinicios del gestor de colas.

REST API V3 **ibm-mq-md-prioridad**

Devuelve la configuración de la prioridad del mensaje. Por ejemplo:

```
ibm-mq-md-priority: 3
```

ibm-mq-md-replyTo

Especifica el destino de respuesta para el mensaje recibido. El formato del encabezado utiliza la notación estándar de la cola de respuesta y el gestor de colas, `replyQueue@replyQMgr`.

Por ejemplo:

```
ibm-mq-md-replyTo: myReplyQueue@myReplyQMgr
```

ibm-mq-resuelto-qmgr

Especifica el nombre del gestor de colas que se utilizó para la solicitud. Si se utilizó un nombre único en el recurso URL, este encabezado identifica el nombre del gestor de colas asociado a ese nombre único. Si el nombre único utilizado en el recurso URL hace referencia a un grupo de gestores de colas, este encabezado identifica qué gestor de colas dentro del grupo se utilizó.

REST API V3 **ibm-mq-usuario**

Devuelve las propiedades definidas por el usuario del mensaje. Se pueden establecer varias propiedades en un mensaje, en cuyo caso habrá dos o más instancias separadas del encabezado de respuesta `ibm-mq-usr`.

Por ejemplo:

```
ibm-mq-usr: myIPProp;5;short
ibm-mq-usr: mySProp;"hi";string
ibm-mq-usr: myBProp>true;boolean
```

Las propiedades tienen la siguiente sintaxis:

```
ibm-mq-usr: property_name; user_value; user_type
```

nombre_propiedad

El nombre de la propiedad de usuario que se está especificando. Debe ser un nombre de propiedad JMS válido.

valor_usuario

El valor de la propiedad.

tipo_de_usuario

El tipo de propiedad:

- boolean (verdadero/falso, MQBOOL)
- byte (entero de 8 bits, MQINT8)

000000000000000000000000000000000000abcdabcd, de la cola Q1 en el gestor de QM1colas :

Eliminar /messaging/qmgr/{qmgrName}/queue/{queueName}/message

Obtiene de forma destructiva el siguiente mensaje disponible del gestor de colas y la cola especificados, y devuelve el cuerpo del mensaje en el cuerpo de la respuesta HTTP. El gestor de colas debe estar en la misma máquina que el servidor mqweb. El mensaje debe tener un formato de MQSTR o JMS TextMessage, y se recibe utilizando el contexto de usuario actual.

La API REST V3 añade la posibilidad de especificar propiedades de mensajes definidas por el usuario e incluir la prioridad de los mensajes. Los encabezados de respuesta `ibm-mq-md-priority` e `ibm-mq-user` solo están disponibles con la API REST V3. El encabezado de solicitud `ibm-mq-md-correlationId` tiene un formato diferente en la API REST V3. El encabezado puede ser un identificador específico de la aplicación o, si es una cadena codificada, conserva el ID: prefijo. El encabezado de respuesta y el parámetro de consulta `ibm-mq-md-messageId` tienen un formato diferente en la API REST V3, ya que conservan el ID: prefijo.

URL del recurso

qmgrName

Desde IBM MQ 9.3.3, puede conectarse a un gestor de colas local o a un gestor de colas remoto. El nombre que especifique para **mqggrName** depende de cómo esté configurado su servidor mqweb:

- Si su servidor mqweb está configurado para conectarse a gestores de colas remotos, utilice el nombre único del gestor de colas tal y como se especifica en la información de conexión del gestor de colas remoto.

Puede determinar si su servidor mqweb está configurado para conectarse a gestores de colas locales o remotos utilizando el **dspmweb properties** comando y visualizando la **mqRestMessagingConnectionMode** propiedad.

El nombre del gestor de colas es sensible a las mayúsculas y minúsculas.

Si el nombre de gestor de colas incluye una barra inclinada, un punto o un signo de porcentaje, estos caracteres se deben codificar en el URL:

- Una barra inclinada (/) debe codificarse como %2F.
- Un signo de porcentaje (%) debe codificarse como %25.

queueName

Especifica el nombre de la cola desde la que se obtendrá el siguiente mensaje.

La cola debe definirse como local o como un alias que apunte a una cola local.

El nombre de la cola distingue entre mayúsculas y minúsculas.

Si el nombre de la cola incluye una barra inclinada o un signo de porcentaje, estos caracteres deben codificarse con e URL :

- Una barra inclinada, /, debe codificarse como %2F.
- Un signo de porcentaje, %, debe codificarse como %25.

Puede utilizar HTTP en lugar de HTTPS si habilita las conexiones HTTP. Para obtener más información sobre la habilitación de HTTP, consulte [Configuración de puertos HTTP y HTTPS](#).

Parámetros de consulta opcionales

REST API V2 cor relationId=valor hexadecimal

Especifica que el método HTTP devuelve el siguiente mensaje con el ID de correlación correspondiente.

hexValue

El parámetro de consulta debe especificarse como una cadena codificada en hexadecimal de 48 caracteres, que representa 24 bytes.

Por ejemplo:

```
../message?correlationId=414d5120514d4144455620202020202067d8bf5923582e02
```

REST API V3 correlationId=ID: valor hexadecimal o correlation Id=valor específico de la aplicación

Especifica que el método HTTP devuelve el siguiente mensaje con el ID de correlación correspondiente.

hexValue

El parámetro de consulta debe especificarse como una cadena codificada en hexadecimal de 48 caracteres, que representa 24 bytes, y precedida por la cadena "ID: ".

Por ejemplo:

```
../message?correlationId=ID:414d5120514d4144455620202020202067d8bf5923582e02
```

valor específico de la aplicación

El parámetro de consulta se puede especificar como una cadena específica de la aplicación.

Por ejemplo:

```
../message?correlationId=My-Custom-CorrelId
```

REST API V2 REST API V1 **messageId=valor hexadecimal**

Especifica que el método HTTP devuelve el siguiente mensaje con el ID de mensaje correspondiente.

hexValue

El parámetro de consulta debe especificarse como una cadena codificada en hexadecimal de 48 caracteres, que representa 24 bytes.

Por ejemplo:

```
../message?messageId=414d5120514d41444556202020202067d8ce5923582f07
```

REST API V3 **messageId=ID: valor hexadecimal**

Especifica que el método HTTP devuelve el siguiente mensaje con el ID de mensaje correspondiente.

hexValue

El parámetro de consulta debe especificarse como una cadena codificada en hexadecimal de 48 caracteres, que representa 24 bytes, y precedida por la cadena "ID: ".

Por ejemplo:

```
../message?messageId=ID:414d5120514d41444556202020202067d8ce5923582f07
```

espera =valor entero

Especifica que el método HTTP esperará *integerValue* milisegundos hasta que el siguiente mensaje esté disponible.

integerValue

El parámetro de consulta debe especificarse como un valor entero que represente la duración en milisegundos. El valor máximo es 2147483647.

Por ejemplo:

```
../message?wait=120000
```

Cabeceras de solicitud

Los siguientes encabezados deben enviarse con la solicitud:

Autorización

Se debe enviar esta cabecera si utiliza autenticación básica. Para obtener más información, consulte [Utilización de la autenticación básica HTTP con REST API](#).

ibm-mq-rest-csrf-token

Esta cabecera se debe establecer, pero el valor puede ser cualquiera, incluso puede estar en blanco.

Los siguientes encabezados pueden enviarse opcionalmente con la solicitud:

Aceptar conjunto de caracteres

Este encabezado se puede utilizar para indicar qué conjunto de caracteres es aceptable para la respuesta. Si se especifica, este encabezado debe establecerse como UTF-8.

Accept-Language

Este encabezado especifica el idioma requerido para cualquier excepción o mensaje de error devuelto en el cuerpo del mensaje de respuesta.

Formato del cuerpo de la solicitud

Ninguna.

Requisitos de seguridad

La persona que llama debe estar autenticada en el servidor mqweb. Los roles MQWebAdmin y MQWebAdminRO no son aplicables para messaging REST API. Para obtener más información sobre la seguridad para REST API, consulte [Seguridad de IBM MQ Console y REST API](#).

Una vez autenticado en el servidor mqweb, el usuario puede utilizar tanto el messaging REST API como el administrative REST API.

Se debe conceder al principal de seguridad de la persona que realiza la llamada la capacidad de obtener mensajes de la cola especificada:

- La cola especificada por la parte *{queueName}* del URL de recurso debe estar habilitada para GET.
-   Para la cola especificada por la parte *{queueName}* del URL de recurso, se debe otorgar la autorización +GET, +INQ y +BROWSE al principal de seguridad del llamante.
-  Para la cola especificada por la parte *{queueName}* del URL de recurso, UPDATE, debe otorgarse acceso al principal de seguridad del llamante.

Este principal de seguridad puede ser el usuario que inició el servidor mqweb o el usuario que ha iniciado sesión en el servidor mqweb. Si el administrador de colas al que se conecta es un administrador de colas remoto, el principal de seguridad podría ser, en su lugar, el usuario proporcionado por las credenciales en la información de conexión del administrador de colas remoto, u otro usuario determinado por las reglas de seguridad del canal. Para obtener más información, consulte [Determinación del principal de seguridad utilizado por el messaging REST API](#).

 En AIX, Linux, and Windows, puede otorgar autorización a los principales de seguridad para que utilicen recursos de IBM MQ utilizando el mandato **setmqaut**. Para obtener más información, consulte **setmqaut** (conceder o revocar la autorización).

En z/OS, consulte [Configuración de la seguridad en z/OS](#).

Códigos de estado de respuesta

200

Mensaje recibido correctamente.

204

No hay mensajes disponibles.

400

Se han proporcionado datos no válidos.

Por ejemplo, se especificó un valor de parámetro de consulta no válido.

401

No está autenticado.

El llamante debe autenticarse en el servidor mqweb y debe ser miembro de uno o más de los roles MQWebAdmin, MQWebAdminRO o MQWebUser. También se debe especificar el `ibm-mq-rest-csrf-token` encabezado. Para obtener más información, consulte [“Requisitos de seguridad” en la página 2225](#).

403

No autorizado.

La persona que llama está autenticada en el servidor mqweb y está asociada a un principal válido. Sin embargo, el director no tiene acceso a todos los recursos IBM MQ necesarios, o a un subconjunto de ellos, o no desempeña esa MQWebUser función. Para obtener más información sobre el acceso necesario, consulte [“Requisitos de seguridad” en la página 2225](#).

404

No existe la cola.

405

La cola está inhibida por GET.

500

Problema con el servidor o código de error de IBM MQ.

501

No se pudo construir la respuesta HTTP.

Por ejemplo, el mensaje recibido tiene un tipo incorrecto, o tiene el tipo correcto pero no se ha podido procesar el cuerpo.

502

El principal de seguridad actual no puede recibir el mensaje, ya que el proveedor de mensajería no admite la función requerida. Por ejemplo, si la ruta de clase del servidor mqweb no es válida.

503

El gestor de colas no se está ejecutando.

Cabeceras de respuesta

Los siguientes encabezados se devuelven con la respuesta:

Idioma del contenido

Especifica el identificador de idioma del mensaje de respuesta en caso de que se produzcan errores o excepciones. Se utiliza junto con el `Accept-Language` encabezado de solicitud para indicar el idioma requerido para cualquier condición de error o excepción. Se utiliza el servidor mqweb predeterminado si el idioma solicitado no es compatible.

Content-Length

Especifica la longitud del cuerpo de la respuesta HTTP, incluso cuando no hay contenido. El valor contiene la longitud (bytes) de los datos del mensaje.

Content-Type

Especifica el tipo de contenido devuelto en el cuerpo de la respuesta del mensaje recibido. Si se realiza correctamente, el valor es `text/plain; charset=utf-8`. En caso de errores o excepciones, el valor es `application/json; charset=utf-8`.

REST API V2 REST API V1 **ibm-mq-md-correlationId**

Especifica el ID de correlación del mensaje recibido. El encabezado se devuelve si el mensaje recibido contiene un ID de correlación válido. Se representa como una cadena codificada en hexadecimal de 48 caracteres, que representa 24 bytes.

Por ejemplo:

```
ibm-mq-md-correlationId: 414d5120514d4144455620202020202067d8bf5923582e02
```

REST API V3 **ibm-mq-md-correlationId**

Especifica el ID de correlación del mensaje recibido. El encabezado se devuelve si el mensaje recibido contiene un ID de correlación válido. El identificador de correlación puede adoptar una de las siguientes formas:

- Una cadena codificada en hexadecimal de 48 caracteres, que representa 24 bytes, precedida por la cadena "ID: ". Por ejemplo:

```
ibm-mq-md-correlationId: ID:414d5120514d4144455620202020202067d8bf5923582e02
```

- Un valor específico de la aplicación. El valor es una cadena específica de la aplicación:

```
ibm-mq-md-correlationId: My-Custom-CorrelId
```

ibm-mq-md-caducidad

Especifica la duración restante hasta la expiración del mensaje recibido. El encabezado puede tener uno de los siguientes valores:

ilimitado

El mensaje no caduca.

Valor entero

Milisegundos restantes antes de que caduque el mensaje.

REST API V2 REST API V1 **ibm-mq-md-messageId**

Especifica el ID de mensaje que se asigna a IBM MQ este mensaje. Al igual que el `ibm-mq-md-correlationId` encabezado, se representa como una cadena codificada en hexadecimal de 48 caracteres, que representa 24 bytes.

Por ejemplo:

```
ibm-mq-md-messageId: 414d5120514d4144455620202020202067d8ce5923582f07
```

REST API V3 **ibm-mq-md-messageId**

Especifica el ID de mensaje que se asigna a IBM MQ este mensaje. Al igual que el `ibm-mq-md-correlationId` encabezado, se representa como una cadena codificada en hexadecimal de 48 caracteres, que representa 24 bytes, precedida por la cadena "ID: "

Por ejemplo:

```
ibm-mq-md-messageId: ID:414d5120514d4144455620202020202067d8ce5923582f07
```

ibm-mq-md-persistencia

Especifica la persistencia del mensaje recibido. El encabezado puede tener uno de los siguientes valores:

nonPersistent

El mensaje no sobrevive a fallos del sistema ni a reinicios del gestor de colas.

persistentes

El mensaje sobrevive a los fallos del sistema o a los reinicios del gestor de colas.

REST API V3 **ibm-mq-md-prioridad**

Devuelve la configuración de la prioridad del mensaje. Por ejemplo:

```
ibm-mq-md-priority: 3
```

ibm-mq-md-replyTo

Especifica el destino de respuesta para el mensaje recibido. El formato del encabezado utiliza la notación estándar de la cola de respuesta y el gestor de colas, `replyQueue@replyQmgr`.

Por ejemplo:

```
ibm-mq-md-replyTo: myReplyQueue@myReplyQMGr
```

ibm-mq-resuelto-qmgr

Especifica el nombre del gestor de colas que se utilizó para la solicitud. Si se utilizó un nombre único en el recurso URL, este encabezado identifica el nombre del gestor de colas asociado a ese nombre único. Si el nombre único utilizado en el recurso URL hace referencia a un grupo de gestores de colas, este encabezado identifica qué gestor de colas dentro del grupo se utilizó.

REST API V3 **ibm-mq-usuario**

Devuelve las propiedades definidas por el usuario del mensaje. Se pueden establecer varias propiedades en un mensaje, en cuyo caso habrá dos o más instancias separadas del encabezado de respuesta `ibm-mq-usr`.

Por ejemplo:

```
ibm-mq-usr: myIPProp;5;short
ibm-mq-usr: mySProp;"hi";string
ibm-mq-usr: myBProp>true;boolean
```

Las propiedades tienen la siguiente sintaxis:

```
ibm-mq-usr: property_name; user_value; user_type
```


nombre_propiedad

El nombre de la propiedad de usuario que se está especificando. Debe ser un nombre de propiedad JMS válido.

valor_usuario

El valor de la propiedad.

tipo_de_usuario

El tipo de propiedad:

- boolean (verdadero/falso, MQBOOL)
- byte (entero de 8 bits, MQINT8)
- short (entero de 16 bits, MQINT16)
- integer (entero de 32 bits, MQINT32)
- long (entero de 64 bits, MQINT64)
- float (32 bits reales, MQFLOAT32)
- double (64 bits reales, MQFLOAT64)
- string (cadena citada)

Formato del cuerpo de la respuesta

Si la operación se realiza correctamente, el cuerpo de la respuesta contiene el cuerpo del mensaje recibido. Si se produce un error, el cuerpo de la respuesta contiene un mensaje de error con formato JSON. Ambas respuestas están UTF-8 codificadas. Para obtener más información, consulte [REST API manejo de errores](#).

Tenga en cuenta que al recibir un mensaje solo se admiten IBM MQMQSTR mensajes con JMS TextMessage formato y. Posteriormente, todos los mensajes se reciben bajo punto de sincronización y todos los mensajes no gestionados se dejan en la cola. La cola de IBM MQ puede configurarse para trasladar estos mensajes con formato incorrecto a un destino alternativo. Para obtener más información, consulte [Manejo de mensajes de error en clases de JMS de IBM MQ](#).

Ejemplos

Los ejemplos siguientes utilizan el URL del recurso v2. Si está utilizando una versión de IBM MQ anterior a IBM MQ 9.1.5 debe utilizar el URL de recurso v1 en su lugar. Es decir, en el URL de recurso, sustituya v1 donde el URL de ejemplo utiliza v2.

El siguiente ejemplo inicia sesión con un usuario llamado mquser con la contraseña mquser. En cURL,, la solicitud de inicio de sesión podría parecerse al siguiente Windows ejemplo. El token LTPA se almacena en el cookiejar.txt archivo utilizando el -c indicador:

```
curl -k "https://localhost:9443/ibmmq/rest/v2/login" -X POST
-H "Content-Type: application/json" --data "{\"username\":\"mquser\", \"password\":\"mquser\"}"
-c c:\cookiejar.txt
```

Una vez que el usuario ha iniciado sesión, el token LTPA y el ibm-mq-rest-csrf-token encabezado HTTP se utilizan para autenticar las solicitudes posteriores. El ibm-mq-rest-csrf-token_token_value puede ser cualquier valor, incluido un espacio en blanco.

- El siguiente Windows ejemplo de cURL elimina el siguiente mensaje disponible de la cola Q1 en el gestor de QM1colas, utilizando las opciones predeterminadas:

```
curl -k "https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/queue/Q1/message"
-X DELETE -b c:\cookiejar.txt -H "ibm-mq-rest-csrf-token: token-value"
-H "Accept: text/plain"
```


qmgrName

Especifica el nombre del gestor de colas al que conectarse para el envío de mensajes. El gestor de colas debe estar en la misma máquina que el servidor mqweb.

Desde IBM MQ 9.3.3, puede conectarse a un gestor de colas local o a un gestor de colas remoto. El nombre que especifique para **qmgrName** depende de cómo esté configurado su servidor mqweb:

- Si su servidor mqweb está configurado para conectarse únicamente a gestores de colas locales, utilice el nombre del gestor de colas.
- Si su servidor mqweb está configurado para conectarse a gestores de colas remotos, utilice el nombre único del gestor de colas tal y como se especifica en la información de conexión del gestor de colas remoto.

Puede determinar si su servidor mqweb está configurado para conectarse a gestores de colas locales o remotos utilizando el **dspmweb properties** comando y visualizando la **mqRestMessagingConnectionMode** propiedad.

El nombre del gestor de colas es sensible a las mayúsculas y minúsculas.

Si el nombre de gestor de colas incluye una barra inclinada, un punto o un signo de porcentaje, estos caracteres se deben codificar en el URL:

- Una barra inclinada (/) debe codificarse como %2F.
- Un signo de porcentaje (%) debe codificarse como %25.

queueName

Especifica el nombre de la cola desde la que se van a explorar los mensajes.

La cola debe definirse como local o como un alias que apunte a una cola local.

El nombre de la cola distingue entre mayúsculas y minúsculas.

Si el nombre de la cola incluye una barra inclinada o un signo de porcentaje, estos caracteres deben codificarse con e URL :

- Una barra inclinada, /, debe codificarse como %2F.
- Un signo de porcentaje, %, debe codificarse como %25.

Puede utilizar HTTP en lugar de HTTPS si habilita las conexiones HTTP. Para obtener más información sobre la habilitación de HTTP, consulte [Configuración de puertos HTTP y HTTPS](#).

Parámetros de consulta opcionales

REST API V2 **correlationId=valor hexadecimal**

Especifica que el método HTTP devuelve el siguiente mensaje con el ID de correlación correspondiente.

hexValue

El parámetro de consulta debe especificarse como una cadena codificada en hexadecimal de 48 caracteres, que representa 24 bytes.

Por ejemplo:

```
../messagelist?correlationId=414d5120514d4144455620202020202067d8bf5923582e02
```

REST API V3 **correlationId=ID: valor hexadecimal o correlation Id=valor específico de la aplicación**

Especifica que el método HTTP devuelve una lista de mensajes con el ID de correlación correspondiente.

hexValue

El parámetro de consulta debe especificarse como una cadena codificada en hexadecimal de 48 caracteres, que representa 24 bytes, y precedida por la cadena "ID: ".

Por ejemplo:

```
../message?correlationId=ID:414d5120514d4144455620202020202067d8bf5923582e02
```

valor_específico_de_la_aplicación

El parámetro de consulta se puede especificar como una cadena específica de la aplicación.

Por ejemplo:

```
../message?correlationId=My-Custom-CorrelId
```

REST API V2 REST API V1 **messageId=valor hexadecimal**

Especifica que el método HTTP devuelve el siguiente mensaje con el ID de mensaje correspondiente.

hexValue

El parámetro de consulta debe especificarse como una cadena codificada en hexadecimal de 48 caracteres, que representa 24 bytes.

Por ejemplo:

```
../message?messageId=414d5120514d41444556202020202067d8ce5923582f07
```

REST API V3 **messageId=ID: valor hexadecimal**

Especifica que el método HTTP devuelve el siguiente mensaje con el ID de mensaje correspondiente.

hexValue

El parámetro de consulta debe especificarse como una cadena codificada en hexadecimal de 48 caracteres, que representa 24 bytes, y precedida por la cadena "ID: ".

Por ejemplo:

```
../message?messageId=ID:414d5120514d41444556202020202067d8ce5923582f07
```

límite=valor entero

Especifica que el cuerpo de la respuesta del método HTTP está limitado a *integerValue* elementos JSON.

integerValue

El parámetro de consulta debe especificarse como un valor entero que represente el número máximo de elementos que contiene el cuerpo de la respuesta JSON.

El valor predeterminado es 10 y el valor máximo es 2147483647.

Por ejemplo:

```
../messagelist?limit=250
```

Cabeceras de solicitud

Los siguientes encabezados deben enviarse con la solicitud:

Autorización

Se debe enviar esta cabecera si utiliza autenticación básica. Para obtener más información, consulte [Utilización de la autenticación básica HTTP con REST API](#).

ibm-mq-rest-csrf-token

Esta cabecera se debe establecer, pero el valor puede ser cualquiera, incluso puede estar en blanco.

Los siguientes encabezados pueden enviarse opcionalmente con la solicitud:

Accept conjunto de caracteres

Este encabezado se puede utilizar para indicar qué conjunto de caracteres es aceptable para la respuesta. Si se especifica, este encabezado debe establecerse como UTF-8.

Accept-Language

Este encabezado especifica el idioma requerido para cualquier excepción o mensaje de error devuelto en el cuerpo del mensaje de respuesta.

Formato del cuerpo de la solicitud

Ninguna.

Requisitos de seguridad

La persona que llama debe estar autenticada en el servidor mqweb. Los roles MQWebAdmin y MQWebAdminRO no son aplicables para messaging REST API. Para obtener más información sobre la seguridad para REST API, consulte [Seguridad de IBM MQ Console y REST API](#).

Una vez autenticado en el servidor mqweb, el usuario puede utilizar tanto el messaging REST API como el administrative REST API.

Se debe otorgar al principal de seguridad del proceso de llamada la capacidad de examinar mensajes de la cola especificada:

- La cola especificada por la parte *{queueName}* del URL de recurso debe estar habilitada para BROWSE.
-  **MQ Appliance** Para la cola especificada por la parte *{queueName}* del URL de recurso, se debe otorgar la autorización +GET, +INQ y +BROWSE al principal de seguridad del llamante.
-  Para la cola especificada por la parte *{queueName}* del URL de recurso, UPDATE, debe otorgarse acceso al principal de seguridad del llamante.

Este principal de seguridad puede ser el usuario que inició el servidor mqweb o el usuario que ha iniciado sesión en el servidor mqweb. Si el administrador de colas al que se conecta es un administrador de colas remoto, el principal de seguridad podría ser, en su lugar, el usuario proporcionado por las credenciales en la información de conexión del administrador de colas remoto, u otro usuario determinado por las reglas de seguridad del canal. Para obtener más información, consulte [Determinación del principal de seguridad utilizado por el messaging REST API](#).

 En AIX, Linux, and Windows, puede otorgar autorización a los principales de seguridad para que utilicen recursos de IBM MQ utilizando el mandato **setmqaut**. Para obtener más información, consulte [setmqaut](#) (conceder o revocar la autorización).

En z/OS, consulte [Configuración de la seguridad en z/OS](#).

Códigos de estado de respuesta

200

Lista de mensajes recibida correctamente.

400

Se han proporcionado datos no válidos.

Por ejemplo, se especificó un valor de parámetro de consulta no válido.

401

No está autenticado.

El llamante debe autenticarse en el servidor mqweb y debe ser miembro de uno o más de los roles MQWebAdmin, MQWebAdminRO o MQWebUser. También se debe especificar el `ibm-mq-rest-csrf-token` encabezado. Para obtener más información, consulte [“Requisitos de seguridad” en la página 2233](#).

403

No autorizado.

La persona que llama está autenticada en el servidor mqweb y está asociada a un principal válido. Sin embargo, el director no tiene acceso a todos los recursos IBM MQ necesarios, o a un subconjunto de ellos, o no desempeña esa MQWebUser función. Para obtener más información sobre el acceso necesario, consulte [“Requisitos de seguridad” en la página 2233](#).

404

No existe la cola.

500

Problema con el servidor o código de error de IBM MQ.

501

No se pudo construir la respuesta HTTP.

Por ejemplo, el mensaje recibido tiene un tipo incorrecto, o tiene el tipo correcto pero no se ha podido procesar el cuerpo.

502

El principal de seguridad actual no puede recibir el mensaje, ya que el proveedor de mensajería no admite la función requerida. Por ejemplo, si la ruta de clase del servidor mqweb no es válida.

503

El gestor de colas no se está ejecutando.

Cabeceras de respuesta

Idioma del contenido

Especifica el identificador de idioma del mensaje de respuesta en caso de que se produzcan errores o excepciones. Se utiliza junto con el Accept-Language encabezado de solicitud para indicar el idioma requerido para cualquier condición de error o excepción. Se utiliza el servidor mqweb predeterminado si el idioma solicitado no es compatible.

Content-Length

Especifica la longitud del cuerpo de la respuesta HTTP, incluso cuando no hay contenido. El valor contiene la longitud de los datos del mensaje, en bytes.

Content-Type

Especifica el tipo de cuerpo de la respuesta. El valor es `application/json; charset=utf-8`.

ibm-mq-resuelto-qmgr

Especifica el nombre del gestor de colas que se utilizó para la solicitud. Si se utilizó un nombre único en el recurso URL, este encabezado identifica el nombre del gestor de colas asociado a ese nombre único. Si el nombre único utilizado en el recurso URL hace referencia a un grupo de gestores de colas, este encabezado identifica qué gestor de colas dentro del grupo se utilizó.

Formato del cuerpo de la respuesta

Si se obtiene un resultado satisfactorio, el cuerpo de la respuesta es una respuesta UTF-8 codificada. La respuesta contiene un objeto JSON externo que contiene una única matriz JSON llamada `messages`. Cada elemento de la matriz es un objeto JSON que contiene información sobre un mensaje de la cola. Cada elemento contiene los siguientes atributos:

REST API V2 REST API V1 correlationId

Especifica el ID de correlación del mensaje. Se devuelve el valor si el mensaje contiene un ID de correlación válido.

REST API V3 correlationId

Especifica el ID de correlación del mensaje. Se devuelve el valor si el mensaje contiene un ID de correlación válido. El identificador de correlación tiene como prefijo la cadena «ID:», o puede ser un valor específico de la aplicación.

REST API V2 REST API V1 messageId

Especifica el ID de mensaje que IBM MQ asigna a este mensaje. Se representa como una cadena codificada en hexadecimal de 48 caracteres, que representa 24 bytes.

REST API V3 messageId

Especifica el ID de mensaje que IBM MQ asigna a este mensaje. Se representa como una cadena codificada en hexadecimal de 48 caracteres, que representa 24 bytes. El ID del mensaje tiene como prefijo la cadena «ID:».

Especifica el campo de formato MQMD. En circunstancias normales, los mensajes de texto contendrán el IBM MQMQSTR valor.

Si se produce un error, el cuerpo de la respuesta contiene un mensaje de error con formato JSON. Para obtener más información, consulte [REST API manejo de errores](#).

Los ejemplos siguientes utilizan el URL del recurso v2. Si está utilizando una versión de IBM MQ anterior a IBM MQ 9.1.5 debe utilizar el URL de recurso v1 en su lugar. Es decir, en el URL de recurso, sustituya v1 donde el URL de ejemplo utiliza v2.

```
curl -k "https://localhost:9443/ibmmq/rest/v2/login" -X POST
-H "Content-Type: application/json" --data "{\"username\":\"mquser\",\"password\":\"mquser\"}"
-c c:\cookiejar.txt
```

- El siguiente Windows ejemplo de cURL muestra los siguientes diez mensajes disponibles de la cola Q1 en el gestor de QM1colas, utilizando las opciones predeterminadas:

```
curl -k "https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/queue/Q1/messagelist"  
-X GET -b c:\cookiejar.txt -H "ibm-mq-rest-csrf-token: token-value"  
-H "Accept: application/json"
```

```
curl -k "https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/queue/Q1/messagelist?limit=200"
-X GET -b c:\cookiejar.txt -H "ibm-mq-rest-csrf-token: token-value"
-H "Accept: application/json"
```

```
curl -k "https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/queue/Q1/messageList?correlationId=0000000000000000000000000000000000000000abcdabcd"  
-X GET -b c:\cookiejar.txt -H "ibm-mq-rest-csrftoken: token-value"  
-H "Accept: application/json"
```

El siguiente Windows ejemplo de cURL es igual que el anterior, pero utiliza la API REST V3. El ejemplo solo muestra los mensajes con el ID de correlación correspondiente, 0000000000000000000000000000000000000000abcdabcd, de la cola Q1 en el gestor de QM1colas :

[illegible]

/messaging/qmgr/{qmgrName}/topic/{topicString}/message

Puede utilizar el método POST HTTP con el /messaging/qmgr/{qmgrName}/topic/{topicString}/message recurso para publicar mensajes en el tema especificado en el gestor de colas especificado.

POST/messaging/qmgr/{qmgrName}/topic/{topicString}/message

Puede utilizar el método POST HTTP con el /messaging/qmgr/{qmgrName}/topic/{topicString}/message recurso para publicar mensajes en el tema especificado en el gestor de colas especificado.

Publica un mensaje de texto en el cuerpo de la solicitud HTTP al gestor de colas y al tema especificados. El gestor de colas debe estar en la misma máquina que el servidor mqweb, y solo se admiten mensajes de texto. Los mensajes se publican como MQSTR o JMS TextMessage mensajes formateados utilizando el contexto de usuario actual, y tienen una prioridad de mensaje predeterminada de 4 para la API REST V2, y asDestination para la API REST V3.

La API REST V3 añade la posibilidad de especificar propiedades de mensajes definidas por el usuario e incluir la prioridad de los mensajes. Los encabezados de solicitud ibm-mq-md-priority e ibm-mq-usr solo están disponibles con la API REST V3. El encabezado de solicitud ibm-mq-md-correlationId tiene un formato diferente en la API REST V3. El encabezado puede ser un ID específico de la aplicación o, si es una cadena codificada, requiere el ID: prefijo. Si su solicitud POST contiene mensajes definidos por el usuario o un ID de correlación específico de la aplicación, el mensaje se formatea como un JMS TextMessage.

- “URL del recurso” en la [página 2236](#)
- “Cabeceras de solicitud” en la [página 2237](#)
- “Formato del cuerpo de la solicitud” en la [página 2239](#)
- “Requisitos de seguridad” en la [página 2239](#)
- “Códigos de estado de respuesta” en la [página 2240](#)
- “Cabeceras de respuesta” en la [página 2240](#)
- “Formato del cuerpo de la respuesta” en la [página 2241](#)
- “Ejemplos” en la [página 2241](#)

URL del recurso

`https://host:port/ibmmq/rest/v2/messaging/qmgr/{qmgrName}/topic/{topicString}/message`

`https://host:port/ibmmq/rest/v3/messaging/qmgr/{qmgrName}/topic/{topicString}/message`

qmgrName

Especifica el nombre de un gestor de colas al que conectarse para el envío de mensajes.

Desde IBM MQ 9.3.3, puede conectarse a un gestor de colas local o a un gestor de colas remoto. El nombre que especifique para **qmgrName** depende de cómo esté configurado su servidor mqweb:

- Si su servidor mqweb está configurado para conectarse únicamente a gestores de colas locales, utilice el nombre del gestor de colas.
- Si su servidor mqweb está configurado para conectarse a gestores de colas remotos, utilice el nombre único del gestor de colas tal y como se especifica en la información de conexión del gestor de colas remoto.

Puede determinar si su servidor mqweb está configurado para conectarse a gestores de colas locales o remotos utilizando el **dspmweb properties** comando y visualizando la **mqRestMessagingConnectionMode** propiedad.

El nombre es sensible a las mayúsculas y minúsculas.

Si el nombre incluye una barra inclinada, un punto o un signo de porcentaje, estos caracteres deben codificarse con e URL :

- Una barra inclinada se debe codificar como %2F.
- Un punto se debe codificar como %2E.
- Un signo de porcentaje se debe codificar como %25.

topicString

Especifica la cadena de tema en la que se publicará el mensaje.

La cadena del tema distingue entre mayúsculas y minúsculas. La cadena de temas puede contener varios niveles de temas, separados por el delimitador de barra inclinada.

Si la cadena del tema contiene un signo de porcentaje, un punto o un signo de interrogación, estos caracteres deben codificarse con e URL :

- Un signo de porcentaje se debe codificar como %25.
- Un punto se debe codificar como %2E.
- Un signo de interrogación se debe codificar como %3F.

Si la cadena del tema comienza o termina con una barra inclinada, debe codificarse con un %2F.

Por ejemplo, para publicar en la cadena de tema:

- sport/football En el gestor de MY .QMGRcolas, utilice el siguiente comando « URL »:

```
https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/MY%2EQMGR/topic/sport/football/message
```

- /sport/football En el gestor de MY .QMGRcolas, utilice el siguiente comando « URL »:

```
https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/MY%2EQMGR/topic/%2Fsport/football/message
```

Puede utilizar HTTP en lugar de HTTPS si habilita las conexiones HTTP. Para obtener más información sobre la habilitación de HTTP, consulte [Configuración de puertos HTTP y HTTPS](#).

Cabeceras de solicitud

Los siguientes encabezados deben enviarse con la solicitud:

Autorización

Se debe enviar esta cabecera si utiliza autenticación básica. Para obtener más información, consulte [Utilización de la autenticación básica HTTP con REST API](#).

Content-Type

Este encabezado debe enviarse con uno de los siguientes valores:

- text/plain;charset=utf-8
- text/html;charset=utf-8
- text/xml;charset=utf-8
- application/json;charset=utf-8
- application/xml;charset=utf-8

ibm-mq-rest-csrf-token

Esta cabecera se debe establecer, pero el valor puede ser cualquiera, incluso puede estar en blanco.

Los siguientes encabezados pueden enviarse opcionalmente con la solicitud:

Accept-Language

Este encabezado especifica el idioma requerido para cualquier excepción o mensaje de error devuelto en el cuerpo del mensaje de respuesta.

ibm-mq-md-caducidad

Este encabezado establece la duración de caducidad del mensaje creado. El vencimiento de un mensaje comienza desde el momento en que el mensaje llega al gestor de colas. Como resultado, se ignora la latencia de la red. El encabezado debe especificarse como uno de los siguientes valores:

ilimitado

El mensaje no caduca.

Este es el valor predeterminado.

Valor entero

Milisegundos antes de que caduque el mensaje.

Limitado al rango 0 - 99999999900.

ibm-mq-md-persistencia

Este encabezado establece la persistencia del mensaje creado. El encabezado debe especificarse como uno de los siguientes valores:

nonPersistent

El mensaje no sobrevive a fallos del sistema ni a reinicios del gestor de colas.

Este es el valor predeterminado.

persistentes

El mensaje sobrevive a los fallos del sistema o a los reinicios del gestor de colas.

REST API V3 ibm-mq-md-prioridad

Este encabezado establece la prioridad del mensaje creado. El encabezado debe especificarse como uno de los siguientes valores:

asDestination

El mensaje utiliza la prioridad especificada en el atributo DEFPTY del objeto de cola IBM MQ subyacente.

Valor entero

Especifique la prioridad real como un número entero en el rango 0-9.

Por ejemplo:

```
ibm-mq-md-priority: asDestination
```

ibm-mq-md-replyTo

Este encabezado establece el destino de respuesta para el mensaje creado. El formato del encabezado utiliza la notación estándar de proporcionar la cola de respuesta y un gestor de colas opcional: `replyQueue[@replyQMgr]`

Por ejemplo:

```
ibm-mq-md-replyTo: myReplyQueue@myReplyQMgr
```

REST API V3 ibm-mq-usuario

Establezca las propiedades definidas por el usuario del mensaje de solicitud. Se pueden establecer múltiples propiedades en un mensaje. Puede especificar varias propiedades separadas por comas en un único encabezado de solicitud `ibm-mq-usr`, o puede utilizar dos o más instancias separadas del encabezado de solicitud `ibm-mq-usr`.

Por ejemplo:

```
ibm-mq-usr: myIPprop;5;short
ibm-mq-usr: mySProp;"hi";string
ibm-mq-usr: myBProp>true;boolean
ibm-mq-usr: myA;5;byte,myB;-10;integer
```

Las propiedades tienen la siguiente sintaxis:

```
ibm-mq-usr: property_name; user_value; user_type
```

nombre_propiedad

El nombre de la propiedad de usuario que se está especificando. Debe ser un nombre de propiedad JMS válido.

valor_usuario

El valor de la propiedad.

tipo_de_usuario

El tipo de propiedad:

- `boolean` (verdadero/falso, MQBOOL)
- `byte` (entero de 8 bits, MQINT8)
- `short` (entero de 16 bits, MQINT16)
- `integer` (entero de 32 bits, MQINT32)
- `long` (entero de 64 bits, MQINT64)
- `float` (32 bits reales, MQFLOAT32)
- `double` (64 bits reales, MQFLOAT64)
- `string` (cadena citada)

Formato del cuerpo de la solicitud

El cuerpo de la solicitud debe ser texto y utilizar la codificación UTF-8. No se requiere una estructura de texto específica. Se crea un mensaje MQSTR con formato que contiene el texto del cuerpo de la solicitud y se publica en el tema especificado.

 Si se utilizan las propiedades definidas por el usuario de la API REST V3 o las características de ID de correlación específicas de la aplicación, se crea un mensaje JMS `TextMessage` con formato que contiene el texto del cuerpo de la solicitud y se coloca en la cola especificada.

Para obtener más información, consulte [los ejemplos](#).

Requisitos de seguridad

La persona que llama debe estar autenticada en el servidor mqweb. Los roles MQWebAdmin y MQWebAdminRO no son aplicables para messaging REST API. Para obtener más información sobre la seguridad para REST API, consulte [Seguridad de IBM MQ Console y REST API](#).

Una vez autenticado en el servidor mqweb, el usuario puede utilizar tanto el messaging REST API como el administrative REST API.

Se debe conceder al principal de seguridad de la persona que realiza la llamada la capacidad de publicar mensajes en el tema especificado:

- El tema especificado por la `{topicString}` parte del recurso URL debe estar PUBLISH habilitado.
-   Para el tema especificado por la `{topicString}` parte del recurso URL, se debe +PUB conceder autoridad al principal de seguridad del llamante.
-  Para el tema especificado por la `{topicString}` parte del recurso URL, se debe UPDATE conceder acceso al principal de seguridad del llamante.

Este principal de seguridad puede ser el usuario que inició el servidor mqweb o el usuario que ha iniciado sesión en el servidor mqweb. Si el administrador de colas al que se conecta es un administrador de colas remoto, el principal de seguridad podría ser, en su lugar, el usuario proporcionado por las credenciales en la información de conexión del administrador de colas remoto, u otro usuario determinado por las reglas de seguridad del canal. Para obtener más información, consulte [Determinación del principal de seguridad utilizado por el messaging REST API](#).



En AIX, Linux, and Windows, puede otorgar autorización a los principales de seguridad para que utilicen recursos de IBM MQ utilizando el mandato **setmqaut**. Para obtener más información, consulte **setmqaut** (conceder o revocar la autorización).

En z/OS, consulte [Configuración de la seguridad en z/OS](#).

Si utiliza Advanced Message Security (AMS) con la messaging REST API, tenga en cuenta que todos los mensajes se cifran utilizando el contexto del servidor mqweb, no el contexto del usuario que publica el mensaje.

Códigos de estado de respuesta

201

Mensaje creado y publicado correctamente.

400

Se han proporcionado datos no válidos.

Por ejemplo, se especificó un valor de encabezado de solicitud no válido.

401

No está autenticado.

El llamante debe autenticarse en el servidor mqweb y debe ser miembro de uno o más de los roles MQWebAdmin, MQWebAdminRO o MQWebUser. También se debe especificar el `ibm-mq-rest-csrf-token` encabezado. Para obtener más información, consulte [“Requisitos de seguridad” en la página 2239](#).

403

No autorizado.

La persona que llama está autenticada en el servidor mqweb y está asociada a un principal válido. Sin embargo, el director no tiene acceso a todos los recursos IBM MQ necesarios, o a un subconjunto de ellos, o no desempeña esa MQWebUser función. Para obtener más información sobre el acceso necesario, consulte [“Requisitos de seguridad” en la página 2239](#).

404

El gestor de colas no existe.

405

El tema está inhibido por PUBLISH.

415

El encabezado o el cuerpo del mensaje es un tipo de medio no compatible.

Por ejemplo, el Content-Type encabezado está configurado con un tipo de medio no compatible.

500

Problema con el servidor o código de error de IBM MQ.

502

El principal de seguridad actual no puede publicar el mensaje, ya que el proveedor de mensajería no admite la función requerida. Por ejemplo, si la ruta de clase del servidor mqweb no es válida.

503

El gestor de colas no se está ejecutando.

Cabeceras de respuesta

Los siguientes encabezados se devuelven con la respuesta:

Idioma del contenido

Especifica el identificador de idioma del mensaje de respuesta en caso de que se produzcan errores o excepciones. Se utiliza junto con el Accept-Language encabezado de solicitud para indicar el idioma requerido para cualquier condición de error o excepción. Se utiliza el servidor mqweb predeterminado si el idioma solicitado no es compatible.

Content-Length

Especifica la longitud del cuerpo de la respuesta HTTP, incluso cuando no hay contenido. Si se realiza correctamente, el valor es cero.

Content-Type

Especifica el tipo de cuerpo de la respuesta. Si se realiza correctamente, el valor es `text/plain; charset=utf-8`. En caso de errores o excepciones, el valor es `application/json; charset=utf-8`.

ibm-mq-resuelto-qmgr

Especifica el nombre del gestor de colas que se utilizó para la solicitud. Si se utilizó un nombre único en el recurso URL, este encabezado identifica el nombre del gestor de colas asociado a ese nombre único. Si el nombre único utilizado en el recurso URL hace referencia a un grupo de gestores de colas, este encabezado identifica qué gestor de colas dentro del grupo se utilizó.

Formato del cuerpo de la respuesta

El cuerpo de la respuesta está vacío si el mensaje se publica correctamente. Si se produce un error, el cuerpo de la respuesta contiene un mensaje de error. Para obtener más información, consulte [REST API manejo de errores](#).

Ejemplos

El siguiente ejemplo inicia sesión con un usuario llamado `mqluser` con la contraseña `mqluser`. En `cURL`, la solicitud de inicio de sesión podría parecerse al siguiente Windows ejemplo. El token LTPA se almacena en el `cookiejar.txt` archivo utilizando el `-c` indicador:

```
curl -k "https://localhost:9443/ibmmq/rest/v1/login" -X POST
-H "Content-Type: application/json" --data "{\"username\":\"mqluser\", \"password\":\"mqluser\"}"
-c c:\cookiejar.txt
```

Una vez que el usuario ha iniciado sesión, el token LTPA y el `ibm-mq-rest-csrf-token` encabezado HTTP se utilizan para autenticar las solicitudes posteriores. El `ibm-mq-rest-csrf-token_value` puede ser cualquier valor, incluido un espacio en blanco.

- El siguiente Windows ejemplo de `cURL` publica un mensaje en el tema string `myTopic` en el gestor de QM1colas, utilizando las opciones predeterminadas. El mensaje contiene el texto «¡Hola, mundo!» :

```
curl -k "https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/topic/myTopic/message"
-X POST -b c:\cookiejar.txt -H "ibm-mq-rest-csrf-token: token_value"
-H "Content-Type: text/plain; charset=utf-8" --data "Hello World!"
```

- El siguiente Windows ejemplo de `cURL` publica un mensaje persistente en el tema string `myTopic/thisTopic` en el gestor de QM1colas, con una caducidad de 2 minutos. El mensaje contiene el texto «¡Hola, mundo!» :

```
curl -k "https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/topic/myTopic%2FthisTopic/message"
-X POST -b c:\cookiejar.txt -H "ibm-mq-rest-csrf-token: token_value"
-H "Content-Type: text/plain; charset=utf-8" -H "ibm-mq-md-persistence: persistent"
-H "ibm-mq-md-expiry: 120000" --data "Hello World!"
```


Esta información se ha desarrollado para productos y servicios ofrecidos en los Estados Unidos.

Es posible que IBM no ofrezca los productos, servicios o las características que se tratan en este documento en otros países. Consulte al representante local de IBM para obtener información sobre los productos y servicios que actualmente pueden adquirirse en su zona. Las referencias a programas, productos o servicios de IBM no pretenden establecer ni implicar que sólo puedan utilizarse dichos productos, programas o servicios de IBM. En su lugar podrá utilizarse cualquier producto, programa o servicio equivalente que no infrinja ninguno de los derechos de propiedad intelectual de IBM. No obstante, es responsabilidad del usuario evaluar y verificar el funcionamiento de cualquier producto, programa o servicio no IBM.

IBM puede tener patentes o solicitudes de patentes pendientes que cubran el tema principal descrito en este documento. El suministro de este documento no le otorga ninguna licencia sobre estas patentes. Puede enviar consultas sobre licencias, por escrito, a:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Para consultas sobre licencias relacionadas con información de doble byte (DBCS), póngase en contacto con el Departamento de propiedad intelectual de IBM de su país o envíe las consultas por escrito a:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokio 103-8510, Japón

El párrafo siguiente no se aplica al Reino Unido ni a ningún otro país donde estas disposiciones contradigan la legislación vigente: INTERNATIONAL BUSINESS MACHINES CORPORATION PROPORCIONA ESTA PUBLICACIÓN "TAL CUAL" SIN NINGÚN TIPO DE GARANTÍA, YA SEA EXPLÍCITA O IMPLÍCITA, INCLUYENDO, PERO SIN LIMITARSE A, LAS GARANTÍAS IMPLÍCITAS DE NO INCUMPLIMIENTO, COMERCIALIZACIÓN O IDONEIDAD PARA UNA FINALIDAD DETERMINADA. Algunas legislaciones no contemplan la exclusión de garantías, ni implícitas ni explícitas, en determinadas transacciones, por lo que puede haber usuarios a los que no les afecte dicha norma.

Esta información puede contener imprecisiones técnicas o errores tipográficos. La información aquí contenida está sometida a cambios periódicos; tales cambios se irán incorporando en nuevas ediciones de la publicación. IBM puede realizar mejoras y/o cambios en los productos y/o programas descritos en esta publicación en cualquier momento sin previo aviso.

Las referencias hechas en esta publicación a sitios web que no son de IBM se proporcionan sólo para la comodidad del usuario y no constituyen de modo alguno un aval de esos sitios web. Los materiales de estos sitios web no forman parte de los materiales para este producto IBM, por lo que la utilización de dichos sitios web es a cuenta y riesgo del usuario.

IBM puede utilizar o distribuir cualquier información que el usuario le proporcione del modo que considere apropiado sin incurrir por ello en ninguna obligación con respecto al usuario.

Los titulares de licencias de este programa que deseen información del mismo con el fin de permitir: (i) el intercambio de información entre los programas creados de forma independiente y otros programas (incluido este) y (ii) el uso mutuo de la información intercambiada, deben ponerse en contacto con:

IBM Corporation
Software Interoperability Coordinator, Department 49XA

3605 Highway 52 N
Rochester, MN 55901
U.S.A.

Dicha información puede estar disponible, sujeta a los términos y condiciones apropiados, incluyendo, en algunos casos, el pago de una cantidad.

El programa bajo licencia que se describe en esta información y todo el material bajo licencia disponible para el mismo lo proporciona IBM bajo los términos del Acuerdo de cliente de IBM, el Acuerdo de licencia de programas internacional de IBM o cualquier acuerdo equivalente entre las partes.

Los datos de rendimiento incluidos en este documento se han obtenido en un entorno controlado. Por consiguiente, los resultados obtenidos en otros entornos operativos pueden variar de manera significativa. Es posible que algunas mediciones se hayan realizado en sistemas en nivel de desarrollo y no existe ninguna garantía de que estas mediciones serán las mismas en sistemas disponibles generalmente. Además, es posible que algunas mediciones se hayan estimado mediante extrapolación. Los resultados reales pueden variar. Los usuarios de este documento deben verificar los datos aplicables a su entorno específico.

La información relativa a productos que no son de IBM se obtuvo de los proveedores de esos productos, sus anuncios publicados u otras fuentes de disponibilidad pública. IBM no ha comprobado estos productos y no puede confirmar la precisión de su rendimiento, compatibilidad o alguna reclamación relacionada con productos que no sean de IBM. Todas las preguntas sobre las prestaciones de productos que no son de IBM deben dirigirse a los proveedores de dichos productos.

Todas las declaraciones relacionadas con una futura intención o tendencia de IBM están sujetas a cambios o se pueden retirar sin previo aviso y sólo representan metas y objetivos.

Este documento contiene ejemplos de datos e informes que se utilizan diariamente en la actividad de la empresa. Para ilustrar los ejemplos de la forma más completa posible, éstos incluyen nombres de personas, empresas, marcas y productos. Todos estos nombres son ficticios y cualquier similitud con los nombres y direcciones utilizados por una empresa real es puramente casual.

LICENCIA DE DERECHOS DE AUTOR:

Esta información contiene programas de aplicación de ejemplo en lenguaje fuente que ilustran técnicas de programación en diversas plataformas operativas. Puede copiar, modificar y distribuir estos programas de ejemplo de cualquier forma sin pagar ninguna cuota a IBM para fines de desarrollo, uso, marketing o distribución de programas de aplicación que se ajusten a la interfaz de programación de aplicaciones para la plataforma operativa para la que se han escrito los programas de ejemplo. Los ejemplos no se han probado minuciosamente bajo todas las condiciones. IBM, por tanto, no puede garantizar la fiabilidad, servicio o funciones de estos programas.

Puede que si visualiza esta información en copia software, las fotografías e ilustraciones a color no aparezcan.

Información acerca de las interfaces de programación

La información sobre la interfaz de programación, si se proporciona, tiene por objeto ayudarle a crear software de aplicación para su uso con este programa.

Este libro contiene información sobre las interfaces de programación previstas que permiten al cliente escribir programas para obtener los servicios de IBM MQ.

Sin embargo, esta información puede contener también información de diagnóstico, modificación y ajustes. La información de diagnóstico, modificación y ajustes se proporciona para ayudarle a depurar el software de aplicación.

Importante: No utilice esta información de diagnóstico, modificación y ajuste como interfaz de programación, ya que está sujeta a cambios.

Marcas registradas

IBM, el IBM logotipo, ibm.com, son marcas comerciales de IBM Corporation, registradas en numerosas jurisdicciones de todo el mundo. La lista actualizada de IBM marcas comerciales está disponible en la web en "Información sobre derechos de autor y marcas comerciales" www.ibm.com/legal/copytrade.shtml. Otros nombres de productos y servicios pueden ser marcas registradas de IBM o de otras empresas.

Microsoft y Windows son marcas registradas de Microsoft Corporation en Estados Unidos y/o otros países.

UNIX es una marca registrada de Open Group en Estados Unidos y en otros países.

Linux es una marca registrada de Linus Torvalds en Estados Unidos y en otros países.

Este producto incluye software desarrollado por el Proyecto Eclipse (<https://www.eclipse.org/>).

Java y todas las marcas registradas y logotipos son marcas registradas de Oracle o sus afiliados.



Número Pieza:

(1P) P/N: